Modeling and Reasoning about Incomplete, Uncertain, and Approximate Historical Dates

Ryan Shaw

School of Information and Library Science
University of North Carolina at Chapel Hill

1 Introduction

From partially unreadable dates on documents, to dates estimated based on contextual evidence, to the inherently approximate dates of cultural diffusion based on archaeological evidence, historical datasets are full of incomplete, uncertain, and approximate descriptions of time. Conventions abound for recording such descriptions, making them difficult to compare when aggregating data in historical knowledge graphs. Recently some progress has been made on this problem, in the form of the Extended Date and Time Format (EDTF), developed at the Library of Congress (Network Development and MARC Standards Office, 2019) and now incorporated into the ISO 8601 standard for representing dates and times (ISO, 2019b). EDTF specifies a standard syntax for expressing uncertain or approximate Gregorian calendar dates, dates with missing parts, sets of possible dates, and open-ended or recurring intervals of time. But while a standard syntax is useful, alone it is insufficient for computational management of historical dates. Neither the SPARQL standard nor any other database query engines recognize EDTF expressions as a special datatype. As a result using EDTF typically requires custom parsing and comparison logic even for basic functionality such as filtering and sorting by date.

Projects working with graph technologies have the option of modeling incomplete, uncertain, and approximate historical dates using an ontology that defines concepts for temporal entities and relations, such as the Time Ontology in OWL (OWL-Time) or the CIDOC Conceptual Reference Model (Bekiari et al., 2021; Cox and Little, 2020). Doing so makes it possible to query over incomplete, uncertain, and approximate historical dates using SPARQL and to infer possible temporal sequences using reasoners. But temporal ontologies can be dauntingly complex even for experienced data modelers, and as there are a number of possible ways to model incomplete, uncertain, and approximate historical dates, there is no guarantee of interoperability. The EDTF Ontology project provides practical tools for computationally managing historical dates by bridging the gap between EDTF expressions and ontology-driven temporal modeling and reasoning. The project has developed a OWL-Time-compatible OWL ontology for modeling EDTF concepts in a standard way (Shaw, 2021b) and a set of rules in Notation3 (Arndt et al., 2021) for automatically inferring graphs of temporal concepts and relations from EDTF expressions (Shaw, 2021a). This extended abstract explains

and demonstrates the modeling constructs, focusing on the handling of incomplete, uncertain, and approximate dates.

2 Extended Date and Time Format (EDTF)

EDTF was originally created at the Library of Congress with input from the bibliographic community and others interested in a standard way to express incomplete, uncertain, and approximate dates and times (Network Development and MARC Standards Office, 2019). It was later incorporated into the ISO 8601 standard (ISO, 2019a,b). The EDTF specification defines three "conformance levels" (0, 1, and 2) in increasing order of complexity. EDTF Level 0 is a profile (subset) of ISO 8601 Part 1: Basic rules, which specifies syntax for expressing Gregorian calendar dates and 24-hour clock times with (optional) offsets from UTC time (ISO, 2019a). Level 0 date and time expressions mostly have XML Schema datatype (XSD) equivalents such as `xsd:date`, `xsd:dateTimeStamp`, `xsd:gYear`, and `xsd:gYearMonth`. But Level 0 also specifies syntax for expressing time intervals where both the beginning and end of the interval are dates (not date-times). There is no XSD equivalent for representing a time interval as a string expression.

EDTF Levels 1 and 2 are profiles of ISO 8601 Part 2: Extensions (ISO, 2019b). EDTF Level 1 includes everything in Level 0 and adds support for expressing "qualified" (uncertain, approximate, or both uncertain and approximate) dates, dates where smaller components are left unspecified (for example `2020-XX` meaning "some month in the year 2020"), intervals where either the beginning or the end is qualified or unknown, and "open" intervals lacking either a beginning or an end or both. EDTF Level 2 includes everything in Levels 0 and 1 and adds support for expressing approximate years by specifying a number of significant digits and qualifying or leaving unspecified date components (rather than entire dates). EDTF Level 2 also introduces syntax for expressing for sets of dates. These can be either single-choice set expressions (for example `[1960,1968]` meaning "either in 1960 or in 1968") or inclusive set expressions (for example `{1960,1968}` meaning "in 1960 and again in 1968").

3 Time ontology in OWL (OWL-Time)

OWL-Time defines OWL classes and properties for modeling time in terms of "temporal entities" (Cox and Little, 2020). A temporal entity may be an interval extended in time or a point in time (an instant). An instant is treated as equivalent to an interval with a duration of zero, while a "proper" interval has an end distinct from its beginning and thus a non-zero (positive) duration. A proper interval's relative position in time can be described by relating it to other proper intervals using properties based on the topological temporal relations defined by Allen (1984): before, meets, overlaps, starts, during, finishes, and equals. It is also possible to describe the absolute position in time of a temporal entity (interval or instant) using a standard temporal reference system. Most commonly the temporal reference system will be the conventional Gregorian calendar and 24-hour clock with timezone offset from UTC, but other calendar systems and even non-calendric temporal reference systems such as geologic time or Unix time are possible as well.

The ability to model interrelated temporal entities with absolute temporal positions described using different temporal reference systems is one of the strengths of OWL-Time. Modeling a temporal entity's position in time by explicitly linking it to values in

```
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact xsd:date expression of position in time
ex:when
  a time:Instant ;
  time:inXSDDateTimeStamp  "1914-06-28"^^xsd:date .

# Explicit OWL-Time model of position in time
ex:when
  a time:Instant ;
  time:inDateTime [
    a time:DateTimeDescription ;
    time:hasTRS i8:Gregorian ;
    time:unitType time:unitDay ;
    time:day "---28"^^xsd:gDay ;
    time:month "--06"^^xsd:gMonth ;
    time:year "1914"^^xsd:gYear
  ] .
```

Listing 1: Using `xsd:date` vs. using `time:DateTimeDescription` to represent a position in time

a temporal reference system makes it possible to computationally compare and relate temporal positions that do not have a standard lexical representation. Even when temporal positions do have a standard lexical representation—such as the ISO 8601-1 standard for representing dates of the Gregorian calendar and times based on the 24-hour clock—it can be convenient to work with an explicit model of their elements, similar to working with a parsed date-time object in a programming language like Python or JavaScript. But literal date-time values using a syntax like ISO 8601 are far more concise, so OWL-Time also provides properties for describing the temporal position of an instant by linking it to an `xsd:date` or `xsd:dateTimeStamp` literal (Listing 1).

4 Describing instants, intervals and sets with EDTF

As explained above, OWL-Time allows compact XSD representations of dates and times to be modeled in a "parsed" form called a date-time description. The EDTF Ontology aims to do the same for compact EDTF representations of "extended" dates and times. OWL-Time defines datatype properties such as `time:inXSDDate` for compactly expressing the temporal position of an instant using XML Schema datatypes. Similarly, the EDTF Ontology defines the `edtfo:inEDTFDateTime` datatype property for compactly expressing the temporal position of an instant using EDTF (Listing 2).

But instants are not the only kinds of temporal entities that can have their positions described using EDTF. Since it is also possible to describe temporal intervals using EDTF, the EDTF Ontology defines the `edtfo:hasEDTFDateTimeDescription` datatype property for this purpose (Listing 3).

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact EDTF expression of position in time
ex:when
  a time:Instant ;
  edtfo:inEDTFDateTime "1876-XX" .

# Explicit OWL-Time model of position in time
ex:when
  a time:Instant ;
  time:inDateTime [
    a time:DateTimeDescription ;
    time:hasTRS i8:Gregorian ;
    time:unitType time:unitMonth ;
    time:year "1876"^^xsd:gYear
  ] .
```

Listing 2: Using `edtfo:inEDTFDateTime` to express the temporal position of an instant

EDTF Level 2 also specifies syntax for expressing sets of dates. OWL-Time has no notion of an "set of instants." But because OWL classes can be understood as defining sets of things, sets of instants can be modeled as subclasses of `time:Instant`. Thus the EDTF Ontology defines the `edtfo:hasEDTFDateTimeSetDescription` datatype property for using an EDTF expression to describe an `owl:Class` that is a subclass of `time:Instant` (Listing 4).

One interesting result of modeling sets of instants as classes is that the only difference between EDTF single-choice and inclusive set expressions is that while an inclusive set expression describes a subclass of `time:Instant`, a single-choice set expression both describes a subclass of `time:Instant` and asserts that some specific instant is a member of that subclass. Since the set of instants is modeled as a disjoint union of class expressions (Patel-Schneider et al., 2012, §9.1.4), an OWL reasoner can use the model (along with additional information) to deduce when something happened. For example, the EDTF expression `[1960,1968]` asserts that some instant (let's call it `ex:when`) belongs to a class such that every instant in the class is either in 1960 or in 1968, and no instant can be both in 1960 and in 1968. If additional information is provided asserting that `ex:when` is not in 1960 (i.e. `ex:when` is not a member of the class of instants in 1960), then an OWL reasoner can deduce that `ex:when` must be in 1968.

5 Uncertainty and approximation

EDTF Levels 1 and 2 support qualification operators that can be used to qualify dates or parts of dates as uncertain, approximate, or both uncertain and approximate.

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact EDTF expression of position in time
ex:when
  a time:ProperInterval ;
  edtfo:hasEDTFDateTimeDescription "1954/1968~" .

# Explicit OWL-Time model of position in time
ex:when
  a time:ProperInterval ;
  time:hasBeginning [
    a time:Instant ;
    time:inDateTime [
      a time:DateTimeDescription ;
      time:hasTRS i8:Gregorian ;
      time:unitType time:unitYear ;
      time:year "1954"^^xsd:gYear
    ]
  ] ;
  time:hasEnd [
    a time:Instant ;
    time:inDateTime [
      a time:DateTimeDescription ;
      time:hasTRS i8:Gregorian ;
      time:unitType time:unitYear ;
      time:year "1968"^^xsd:gYear
    ] {| a edtfo:ApproximateStatement |}
  ] .
```

Listing 3: Using `edtfo:hasEDTFDateTimeDescription` to express the temporal position of an interval

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# Compact EDTF expression of position in time
:When
  a owl:Class ;
  rdfs:subClassOf time:Instant ;
  edtfo:hasEDTFDateTimeSetDescription "{1960,1961-12}" .

# Explicit OWL-Time model of position in time
ex:When
  a owl:Class ;
  owl:disjointUnionOf (
    [ rdfs:subClassOf time:Instant ;
      owl:equivalentClass [
        a owl:Restriction ;
        owl:onProperty time:inDateTime ;
        owl:someValuesFrom ex:1960
      ]
    ]
    [ rdfs:subClassOf time:Instant ;
      owl:equivalentClass [
        a owl:Restriction ;
        owl:onProperty time:inDateTime ;
        owl:someValuesFrom ex:1961_12
      ]
    ]
  ) .

# The class of all date-time descriptions
# with year precision where the year is 1960
# (full model omitted for brevity)
ex:1960 rdfs:subClassOf time:DateTimeDescription .

# The class of all date-time descriptions
# with month precision where the year is 1961
# and the month is December
# (full model omitted for brevity)
ex:1961_12 rdfs:subClassOf time:DateTimeDescription .
```

Listing 4: Using `edtfo:hasEDTFDateTimeSetDescription` to describe a set of instants

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact EDTF expression of position in time
ex:when
  a time:Instant ;
  edtfo:inEDTFDateTime "1777-07?" .

# Explicit OWL-Time model of position in time
ex:when
  a time:Instant ;
  time:inDateTime ex:1777_07 .

ex:1777_07
  a time:DateTimeDescription ;
  time:hasTRS i8:Gregorian ;
  time:unitType time:unitMonth ;
  time:year "1777"^xsd:gYear ;
  time:month "--07"^^xsd:gMonth .

ex:temporalPositionStatement
  a edtfo:UncertainStatement ;
  rdf:subject ex:when ;
  rdf:predicate time:inDateTime ;
  rdf:object ex:1777_07 .
```

Listing 5: Using RDF reification to qualify a time assertion as uncertain

In Level 1 these operators can only qualify entire date expressions, while in Level 2 they can qualify parts of date expressions. This kind of qualification is precisely what the RDF reification vocabulary was developed for (Guha and Brickley, 2014, §5.3). Reification means making statements about statements. In this case, we are creating a statement describing the temporal position of an instant, and then creating another statement that qualifies the former statement as uncertain, approximate, or both. The EDTF Ontology defines two classes that can be used to type instances of `rdf:Statement` as approximate or uncertain: `edtfo:UncertainStatement` and `edtfo:ApproximateStatement` (Listing 5). Both of these classes are subclasses of `edtfo:QualifiedStatement`, making it possible to query for all qualified statements regardless of whether they are uncertain or approximate or both.

An advantage of this approach is that modeling a qualified date expression produces all the statements produced by modeling an unqualified date expression, so those statements can be queried without concern for qualification. When one wants to filter out or otherwise treat qualified statements differently, the separate qualification

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact EDTF expression of position in time
ex:when
  a time:Instant ;
  edtfo:inEDTFDateTime "1777-07?" .

# Explicit OWL-Time model of position in time
ex:when
  a time:Instant ;
  time:inDateTime [
    a time:DateTimeDescription ;
    time:unitType time:unitMonth ;
    time:year "1777"^xsd:gYear ;
    time:month "--07"^^xsd:gMonth
  ] {| a edtfo:UncertainStatement |} .
```

Listing 6: Using RDF-star annotation syntax to qualify a time assertion as uncertain

statements can be taken into consideration. Some disadvantages of this approach are that it is verbose and it requires that both the subject and the object of the qualified statement be identified (i.e. they cannot be blank / anonymous nodes). Fortunately, the new RDF-star specification addresses both of these issues (Arndt et al., 2021) (Listing 6).

6 Unspecified parts of dates

EDTF Levels 1 and 2 allow for parts of date expressions to be left unspecified. In Level 1 only smaller parts can be left unspecified; for example if the year is specified the day and month can be left unspecified, but if the month is specified the year cannot be left unspecified. In Level 2 any part can be left unspecified without restriction. Unspecified parts of date expressions are easily modeled using OWL-Time: to model the fact that the month is left unspecified we simply assert that a date-time description has month-level precision, but omit any statement specifying the month (Listing 2). One complication is that EDTF Levels 1 and 2 also allow the rightmost digits of years to be left unspecified (for example `198X` meaning "some year in the 1980s" or `20XX` meaning "some year in the 21st century"). In order to handle these cases, it is necessary to specify parts of date-time descriptions with units larger than a year. For example, the EDTF expression `198X` can be modeled as a date-time description with year-level precision (since it is referring to a specific year) that specifies the decade (1980s) but does not specify the year. The EDTF Ontology defines three properties for specifying the parts of a date-time description at granularities larger than a year: `edtfo:decade`, `edtfo:century`, and `edtfo:millennium`. These take integer values

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact EDTF expression of position in time
ex:when
  a time:Instant ;
  edtfo:inEDTFDateTime "197X" .

# Explicit OWL-Time model of position in time
ex:when
  a time:Instant ;
  time:inDateTime [
    a edtfo:DecadeDescription , time:DateTimeDescription ;
    time:hasTRS i8:Gregorian ;
    time:unitType time:unitYear ;
    edtfo:decade 197
  ] .
```

Listing 7: Some unspecified year in the 1970s

such as `197` to refer to the 1970s or `19` to refer to the 20th century (Listing 7).

EDTF Level 2 also allows for digits within a month to be left unspecified (for example 1X means "October, November, or December" and X1 means "January or November"). These cases differ from the one above in that we cannot capture the unspecified parts of the date by creating a description with some parts left out: January and November are not parts of some larger unit that does not include the other months. Thus partially-specified months are treated as single-choice set expressions (Listing 8).

7 Open temporal intervals

EDTF Level 1 specifies syntax for expressing open intervals of time, i.e. intervals that do not have a beginning or that do not have an end (or that do not have either!). These are distinguished from intervals for which either the beginning or end is simply not known. The latter are easily modeled using OWL-Time as time intervals for which the beginning or end is left unstated: under the open world assumption, if nothing is said about an endpoint of an interval, that just means we know nothing about it, not that it doesn't exist. Open intervals, however, are trickier, because of that same open world assumption. Simply omitting the missing endpoint would make open intervals indistinguishable from intervals with an unknown endpoint. Rather than asserting that we do not know anything about the endpoint, we want to express that the interval does not have an endpoint. For this purpose the EDTF Ontology defines two subclasses of `time:ProperInterval` for modeling open intervals: `edtfo:OpenBeginningInterval` and `edtfo:OpenEndInterval` (Listing 9). The

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# Compact EDTF expression of position in time
ex:when
  a time:Instant ;
  edtfo:inEDTFDateTime "1984-X1" .

# Explicit OWL-Time model of position in time
ex:when
  a [
    a owl:Class ;
    owl:disjointUnionOf (
      [ rdfs:subClassOf time:Instant ;
        owl:equivalentClass [
          a owl:Restriction ;
          owl:onProperty time:inDateTime ;
          owl:someValuesFrom ex:1984_01
        ]
      ]
      [ rdfs:subClassOf time:Instant ;
        owl:equivalentClass [
          a owl:Restriction ;
          owl:onProperty time:inDateTime ;
          owl:someValuesFrom ex:1984_11
        ]
      ]
    )
  ] .

# The class of all date-time descriptions
# with month precision where the year is 1984
# and the month is January
# (full model omitted for brevity)
ex:1984_01 rdfs:subClassOf time:DateTimeDescription .

# The class of all date-time descriptions
# with month precision where the year is 1984
# and the month is November
# (full model omitted for brevity)
ex:1984_11 rdfs:subClassOf time:DateTimeDescription .
```

Listing 8: Either January or November 1984

```
@base <https://periodo.github.io/edtf-ontology/> .
@prefix edtfo: <edtfo.ttl#> .
@prefix ex: <http://example.org/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix i8: <http://www.opengis.net/def/uom/ISO-8601/0/> .

# Compact EDTF expression of position in time
ex:when
  a time:ProperInterval ;
  edtfo:hasEDTFDateTimeDescription "1976/" .

# Explicit OWL-Time model of position in time
ex:when
  a edtfo:OpenEndInterval , time:ProperInterval ;
  time:hasBeginning [
    a time:Instant ;
    time:inDateTime [
      a time:DateTimeDescription ;
      time:hasTRS i8:Gregorian ;
      time:unitType time:unitYear ;
      time:year "1976"^^xsd:gYear
    ]
  ] .
```

Listing 9: An open-ended temporal interval

former adds the constraint that instances must have zero `time:hasBeginning` values, while the latter adds the constraint that instances must have zero `time:hasEnd` values.

References

Allen, J. F. (1984). Towards a general theory of action and time. Artificial Intelligence 23(2), 123–154. https://doi.org/10.1016/0004-3702(84)90008-0.

Arndt, D., J. Broekstra, B. DuCharme, O. Lassila, P. F. Patel-Schneider, E. Prud'hommeaux, J. Ted Thibodeau, and B. Thompson (2021). RDF-star and SPARQL-star. Draft community group report, W3C. https://w3c.github.io/rdf-star/cg-spec/2021-07-01.html.

Arndt, D., W. Van Woensel, D. Tomaszuk, and G. Kellogg (2021). Notation3. Draft community group report, W3C. https://w3c.github.io/N3/spec/.

Bekiari, C., G. Bruseker, M. Doerr, C.-E. Ore, S. Stead, and A. Velios (2021). Definition of the CIDOC conceptual reference model version 7.1.1. Technical report, ICOM/CIDOC CRM Special Interest Group.

Cox, S. and C. Little (2020). Time ontology in OWL. Candidate recommendation, W3C. https://www.w3.org/TR/2020/CR-owl-time-20200326/.

Guha, R. and D. Brickley (2014). RDF schema 1.1. W3C recommendation, W3C. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

ISO (2019a). Date and time — Representations for information interchange — Part 1: Basic rules. Standard ISO 8601-1:2019, International Organization for Standardization, Geneva, CH. https://www.iso.org/standard/70907.html.

ISO (2019b). Date and time — Representations for information interchange — Part 2: Extensions. Standard ISO 8601-2:2019, International Organization for Standardization, Geneva, CH. https://www.iso.org/standard/70908.html.

Network Development and MARC Standards Office (2019). Extended Date/Time Format (EDTF). Specification, Library of Congress, Washington, DC. https://www.loc.gov/standards/datetime/.

Patel-Schneider, P., B. Motik, and B. Parsia (2012). OWL 2 Web Ontology Language structural specification and functional-style syntax (second edition). W3C recommendation, W3C. https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

Shaw, R. (2021a). EDTF in RDF/OWL. https://github.com/periodo/edtf-ontology.

Shaw, R. (2021b). Extended Date/Time Format (EDTF) concepts. Draft specification, PeriodO. https://periodo.github.io/edtf-ontology/.