A General Methodology For Equipping Ontologies With Time

Hans-Ulrich Krieger

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI)
What Is This Talk All About?

- representing changing relationships over time important for
  - reasoning & querying services on top of RDF & OWL
  - practical applications, e.g., business intelligence
  - Semantic Web & Web 2.0 in general
- DLs unable to represent diachronic relations *directly*
  - no *built-in* mechanism to handle changing relationships
  - temporal DLs are no exception
  - extending relation instances with time leads to massive proliferation of objects
- 4D view makes it easy to extend ontologies with time
- **preferable**: a temporal “annotation” mechanism plus lightweight temporal reasoning services
Example: Synchronic Relation

*Tony Blair was born on May 6, 1953.*

output of an IE system (RDF triples):

\[
\langle \text{tb}, \text{rdf:type, Person}\rangle \\
\langle \text{tb}, \text{hasName, "Tony Blair"}\rangle \\
\langle \text{tb}, \text{dateOfBirth, "1953-05-06"}\rangle
\]

dateOfBirth is a synchronic relation, often functional temporal entity stored as range value of relation instance representation perfectly captures the intended meaning.
Example: Diachronic Relation

most relationships vary with time

Christopher Gent was Vodafone’s chairman until July 2003. Later, Chris became the chairman of GlaxoSmithKline with effect from 1st January 2005.

informal IE output:

[????-??-??, 2003-07-??]: <cg, isChairman, vf>
[2005-01-01, ????-??-??]: <cg, isChairman, gsk>
applying \textit{synchronic} representation scheme from above gives:

\begin{verbatim}
<cg, isChairman, vf>
<cg, hasTime, [????-??-??, 2003-07-??]>
<cg, isChairman, gsk>
<cg, hasTime, [2005-01-01, ????-??-??]>
\end{verbatim}

resulting RDF graph mixes up association between fact and extent:

\begin{verbatim}
[????-??-??, 2003-07-??]: <cg, isChairman, vf>
[2005-01-01, ????-??-??]: <cg, isChairman, vf>
[????-??-??, 2003-07-??]: <cg, isChairman, gsk>
[2005-01-01, ????-??-??]: <cg, isChairman, gsk>
\end{verbatim}
Encoding 1: Equip Relation with Temporal Argument

obvious extension, used in temporal data bases and logic programming community

\[ \text{hasCeo}(c, p) \leftrightarrow \text{hasCeo}(c, p, t) \text{ or } \text{hasCeo}(c, p, s, t) \]

DLs do not support relations with more than two arguments, i.e., encoding not applicable to OWL
so what are *Temporal Description Logics* (e.g., Lutz 2004)?

TDLs = DLs + concrete domain (Baader & Hanschke 1991)

TDLs are great aiming at representing **synchronous** relations

temporal features are **functional** relations

descriptive inventory: paths, additional constructors (e.g., <)

example:

\[ \text{Human} \sqsubseteq \exists (\text{hasMother}.\text{dateOfBirth} < \text{dateOfBirth}) \sqcap \exists (\text{hasFather}.\text{dateOfBirth} < \text{dateOfBirth}) \]
Encoding 2: Apply Meta-Logical Predicate

use **holds** to encode temporally constant information
hasCeo must be reinterpreted as a functional fluent
used by situation calculus, Allen logic, KIF
complex relation arguments not possible in OWL
annotation properties in OWL not possible for relation instances

\[
\text{hasCeo}(c, p, t) \mapsto \text{holds}(\text{hasCeo}(c, p), t)
\]
Encoding 3: Reify Original Relation

relation reification loses original relation
needs introduction of a new class for each relation
requires massive ontology rewriting
new individual, four additional relation instances
similarities to reification in RDF

\[
\text{hasCeo}(c, p, t) \iff \exists hc. \\
\text{type}(hc, \text{HasCeo}) \land \text{hasTime}(hc, t) \land \\
\text{company}(hc, c) \land \text{person}(hc, p)
\]
domain argument often anchor for reasoning and querying
so wrap range arguments in a new container object
same container class can be applied to each relation instance
ontology rewriting still needed
related to relation reification, but does not lose relation name

\[
\text{hasCeo}(c, p, t) \iff \exists et . \\
\text{type}(et, \text{EntityTime}) \land \text{hasTime}(et, t) \land \\
\text{hasCeo}(c, et) \land \text{hasEntity}(et, p)
\]
distinction between *endurants* and *perdurants* in philosophy

perdurantist view: all entities only exist for some period of time

perdurant \(\approx 4D\) trajectory in spacetime

time slice = temporal part of a 4D slice

of special interest: slices where specific information stays constant
we usually only have partial information for a given perdurant
Encoding 5: Encode Perdurantist/4D View in OWL

Welty & Fikes 2006: OWL implementation of perdurantist view
time slice encodes time dimension of spacetime
relations from source ontology no longer connect original entities
encoding requires ontology rewriting

\[
\text{hasCeo}(c, p, t) \iff \exists ts1, ts2. \\
\text{hasCeo}(ts1, ts2) \land \\
\text{type}(ts1, \text{TimeSlice}) \land \text{hasTimeSlice}(c, ts1) \land \text{hasTime}(ts1, t) \land \\
\text{type}(ts2, \text{TimeSlice}) \land \text{hasTimeSlice}(p, ts2) \land \text{hasTime}(ts2, t)
\]
reinterpret perdurantist view:

what has originally been an entity becomes a time slice

original entities now describe the “behavior” of perdurants at a certain moment in time (e.g., being a person)

time slices of a perdurant need not to be of the same type, e.g., perdurant DFKI has slices for Company and AcademicInstitution

cooccurring information in such a slice stays constant

encoding does NOT need rewriting of original ontology

\[
\text{hasCeo}(c, p, t) \iff \\
\text{hasCeo}(c, p) \land \text{hasTime}(c, t) \land \text{hasTime}(p, t) \land \\
\text{hasTimeSlice}(C, c) \land \text{hasTimeSlice}(P, p)
\]

time slices c, p are linked to perdurants C, P (created only once)
DC’s CEO Jürgen Schrempp announces that he will resign by 31st December 2005.
I believe [that] Jürgen Schrempp was the CEO of DC from 1995 until 2005.
1. find out which relations will undergo a temporal change
2. identify domain and range class(es) for these relations
3. make these classes time slices using `owl:equivalentClass`

example: PROTON upper ontology (proton.semanticweb.org/)
   1. most properties in PROTON are diachronic properties
   2. `psys:Entity` is the class of choice, both for domain and range
   3. `fourd:TimeSlice ≡ psys:Entity`

\[
\begin{align*}
4D & \\
\downarrow & \\
\text{Time} \rightarrow & \text{PROTime} \leftarrow \text{Allen} \\
\uparrow & \\
\text{PROTON} &
\end{align*}
\]
General Integration Scheme

1. **always use 4D**
   Perdurant: hasTimeSlice; TimeSlice: timeSliceOf, hasTime

2. **choose Time**
   an arbitrary time ontology, e.g., OWL-Time

3. **choose upper/domain ontology**
   the original ontology that lacks time, e.g., PROTON

4. **use Allen (optional)**
   13 relations, plus 6 super-relations defined over time slices

5. **add axiom** fourd:TimeSlice ≡ $c_1 \sqcup \ldots \sqcup c_n$
   $c_1, \ldots, c_n$: maximal incompatible classes that need to be extended by a temporal dimension
Outlook: Temporal Extensions to OWL

additional arguments, going beyond binary relations/triples
Hayes-/ter Horst-style rules can be extended by a temp. dimension
only lightweight reasoning needed

**example 1:** owl:inverseOf
ceoOf(js, dc, 1995, 2005)
→ hasCeo(dc, js, 1995, 2005)

**example 2:** owl:SymmetricProperty
marriedWith(bbt, aj, 2000, 2003)
→ marriedWith(aj, bbt, 2000, 2003)

**example 3:** owl:TransitiveProperty
contains(dfki, room1.26, s, t) & contains(room1.26, chair42, u, v)
→ contains(dfki, chair42, max(s, u), min(t, v))
Paper: Further Issues

- sophisticated time ontology
  - temporal underspecification
  - granularity of time
- more on Hayes-/ter Horst-style entailment rules
- comparison how extended tuples ease the writing of custom rules (and querying), compared to RDF triples
Thank you!

Questions?