# 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):

```
<tb, rdf:type, Person>
<tb, hasName, "Tony Blair">
<tb, dateOfBirth, "1953-05-06">
```

`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>`

# Example: Diachronic Relation, cont.

applying *synchronic* representation scheme from above gives:

<cg, isChairman, vf>
<cg, hasTime, [????-??-??, 2003-07-??]>
<cg, isChairman, gsk>
<cg, hasTime, [2005-01-01, ????-??-??]>

resulting RDF graph mixes up association between fact and extent:

[????-??-??, 2003-07-??]: <cg, isChairman, vf>
~~[2005-01-01, ????-??-??]: <cg, isChairman, vf>~~
~~[????-??-??, 2003-07-??]: <cg, isChairman, gsk>~~
[2005-01-01, ????-??-??]: <cg, isChairman, gsk>

# Encoding 1: Equip Relation with Temporal Argument

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

$$\text{hasCeo(c, p)} \longmapsto \text{hasCeo(c, p, }\underline{\text{t}}\text{)} \;\textbf{or}\; \text{hasCeo(c, p, }\underline{\text{s, t}}\text{)}$$

DLs do **not** support relations with more than two arguments, i.e., encoding **not** applicable to OWL

# Side Note: Temporal Description Logics

so what are *Temporal Description Logics* (e.g., Lutz 2004)?

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

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

temporal features are **functional** relations

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

example:

$$\begin{aligned}\text{Human} \sqsubseteq\ & \exists(\text{hasMother.dateOfBirth} < \text{dateOfBirth}) \sqcap \\ & \exists(\text{hasFather.dateOfBirth} < \text{dateOfBirth})\end{aligned}$$

# 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

$$\text{hasCeo}(c, p, t) \longmapsto \underline{\text{holds}}(\text{hasCeo}(c, p), t)$$

# 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

$$
\begin{aligned}
&\underline{\text{hasCeo}}(\text{c, p, t}) \ \longmapsto\ \exists \text{hc}\,.\\
&\quad \text{type}(\text{hc}, \underline{\text{HasCeo}}) \wedge \text{hasTime}(\text{hc, t}) \wedge\\
&\quad \text{company}(\text{hc, c}) \wedge \text{person}(\text{hc, p})
\end{aligned}
$$

# Encoding 4: Wrap Range Arguments

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

$$
\begin{array}{l}
\mathsf{hasCeo}(\mathsf{c}, \underline{\mathsf{p}, \mathsf{t}}) \;\longmapsto\; \exists \mathsf{et}\,. \\
\quad \mathsf{type}(\mathsf{et}, \underline{\mathsf{EntityTime}}) \wedge \mathsf{hasTime}(\mathsf{et}, \mathsf{t}) \wedge \\
\quad \mathsf{hasCeo}(\mathsf{c}, \mathsf{et}) \wedge \mathsf{hasEntity}(\mathsf{et}, \mathsf{p})
\end{array}
$$

# Perdurants and Time Slices: Encoding 5+6

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

$$
\begin{aligned}
&\underline{\mathrm{hasCeo(c,\ p,\ t)}} \longmapsto \exists \mathrm{ts1,\ ts2}\,. \\
&\quad \mathrm{hasCeo(ts1,\ ts2)} \wedge \\
&\quad \mathrm{type(ts1,\ TimeSlice)} \wedge \mathrm{hasTimeSlice(c,\ ts1)} \wedge \mathrm{hasTime(ts1,\ t)} \wedge \\
&\quad \mathrm{type(ts2,\ TimeSlice)} \wedge \mathrm{hasTimeSlice(p,\ ts2)} \wedge \mathrm{hasTime(ts2,\ t)}
\end{aligned}
$$

# Encoding 6: Reinterpret Perdurantist/4D View

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

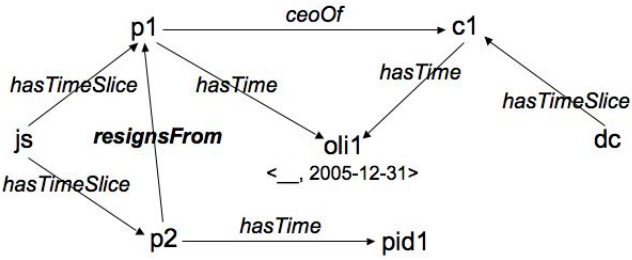
encoding does NOT need rewriting of original ontology

$$
\begin{aligned}
&\mathrm{hasCeo}(c, p, \underline{t}) \longmapsto \\
&\quad \mathrm{hasCeo}(c, p) \wedge \mathrm{hasTime}(c, t) \wedge \mathrm{hasTime}(p, t) \wedge \\
&\quad \mathrm{hasTimeSlice}(C, c) \wedge \mathrm{hasTimeSlice}(P, p)
\end{aligned}
$$

time slices c, p are linked to perdurants C, P (created only once)

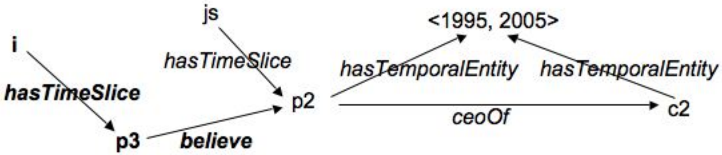
# Example I

*DC's CEO Jürgen Schrempp announces that he will resign by 31st December 2005.*



# Example II

*I believe [that] Jürgen Schrempp was the CEO of DC from 1995 until 2005.*



# Equiping OWL Ontologies With Time: Example

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 $\equiv$ psys:Entity

$$\begin{tikzcd} & \textbf{4D} \arrow[d] & \\ \textbf{Time} \arrow[r] & \text{PROTime} & \textbf{Allen} \arrow[l] \\ & \textbf{PROTON} \arrow[u] & \end{tikzcd}$$

# 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 $\equiv c_1 \sqcup \ldots \sqcup c_n$
   $c_1, \ldots, c_n$: 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)
$\longrightarrow$ hasCeo(dc, js, 1995, 2005)

**example 2:** `owl:SymmetricProperty`
marriedWith(bbt, aj, 2000, 2003)
$\longrightarrow$ marriedWith(aj, bbt, 2000, 2003)

**example 3:** `owl:TransitiveProperty`
contains(dfki, room1.26, s, t) & contains(room1.26, chair42, u, v)
$\longrightarrow$ 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?