A Reusable Time Ontology

Richard Fikes   Qing Zhou

Knowledge Systems Laboratory, Stanford University
Stanford, CA, 94305
fikes@ksl.stanford.edu zhou@ksl.stanford.edu

Abstract

In this paper we present a reusable time ontology for large-scale knowledge system applications that provides an easily understandable, flexible, formally defined, and effective means of representing knowledge about time. Our underlying time theory treats both time points and time intervals as primitive elements on a time line, and the ontology contains a class hierarchy, relations, axioms and instances built on those primitives. The ontology distinguishes between closed and open intervals, as opposed to many previous time ontologies. However, we provide flexibility of usage by providing two sets of relations on intervals: one that assumes that distinction and one that does not. Time granularity is implemented in the ontology to facilitate representing time in varying granularity in a layered model and switching from or relating one granularity to a coarser or finer one. The ontology also includes a representation of periodic intervals and of the standard components and properties of calendars such as calendar months, calendar days, and weekdays.

Introduction

The representation of time is fundamental to any knowledge base that includes representations of change and action. Ontologies providing representations for time are therefore particularly important building blocks for a broad range of knowledge system applications. Although such time ontologies are available from both academic (e.g., Ontolingua [3]) and commercial sources (e.g., Cycorp [4]), they all have significant deficiencies in expressive power, formality, and/or coherence. Our objective is to provide a time ontology that overcomes those deficiencies and provides knowledge engineers with an effective and easily understandable means of representing knowledge about time. The advantages of a standard time ontology include saving repetitious knowledge building efforts and facilitating potential merging of multiple ontologies built by different authors but using the same time ontology. Our time ontology is targeted toward practical use, and we made many design decisions during its development to facilitate knowledge engineering that we will describe later in this paper.

A temporal ontology is based on a temporal logic. There are many different time theories in the literature [1], and there are difficulties associated with each of them. For example, instant-based time theories [5,6,7] are not natural for representing events that have a duration, and Allen's interval-based theory has trouble with the Dividing Instant Problem [8]. In [9, 10], the approach of treating both instants and intervals as independent primitives is presented, which is the approach we adopted. It is of interest to study, as we have here, what kinds of instants (instants of zero duration or instants of unit duration and which unit) and what kinds of intervals (convex or non-convex, open or closed) should be in a library ontology.

Since our time ontology must allow knowledge engineers to specify precisely and easily wide-ranging granular systems, different time granularities must be supported in our ontology. Varying granularity doesn't merely mean one can use different time units, but it involves semantic issues of a layered temporal model and switching from one representation to a coarser or finer one. (See, for example, several formalisms for quantitative and qualitative time granularity in [14,15,16].)

Periodic intervals and calendar dates are also representation problems dealt with in our ontology. In [11,12], temporal formalisms provide frameworks for specification of periodic events and operations on them. Calendar dates have been studied more in the temporal database research area. (For example, [13] discusses calendar support for database systems using user-defined calendars.)

Ontology Overview

Our time ontology is implemented in KIF (Knowledge Interchange Format) [17] augmented with the frame language specified in the knowledge model of the OKBC (Open Knowledge Base Connectivity) knowledge server API [18]. The ontology is available from the Ontolingua server [3] in Stanford's Knowledge Systems Laboratory, and a source file is available at http://ksl.stanford.edu/ontologies/time. The ontology is compatible with the HPKB-Upper-Level-Kernel ontology in the Ontolingua library, which was developed in DARPA's High Performance Knowledge Base (HPKB) program [19].

Choice of Time Theory

Our ontology is based on the notion of a time line analogous to a continuous number line. Time instants and intervals are the temporal entities about which assertions are made. Each time point on the time line is analogous to a real number, and each connected time interval is analogous to an interval on the number line, one of [a, b] (both "a" and "b" are included in the interval), (a, b] ("a" is not included in the interval, but "b" is), [a, b) ("a" is included in the interval, but "b" is not), and (a, b) (neither "a" nor "b" is included in the interval), where a < b. We also define intervals that are not connected. Taking this approach implies that time is continuous and linear in our ontology. This is an intuitive assumption that is also useful in practice.

Ontology Class Hierarchy Structure

Time-Point and Time-Interval are the two fundamental classes in our ontology. Time-Point is the class of time points on the time line corresponding to real numbers on the number line. "13:00:00 exactly on Jan 3, 1970" is an instance of this class, whose corresponding real number is 219600, assuming "00:00:00 exactly on Jan 1, 1970" is the zero point and second is the unit of measurement.

Class Time-Interval is the class of sets of two or more time points. Each time interval has two distinguished points called the Starting-Point and the Ending-Point. Note that instances of Time-Interval do not directly correspond to connected intervals on the number line. "The time when I went jogging this week" is an instance of Time-Interval, but it is not connected. This generalized class makes it much easier to represent events such as "I play tennis every day from 6 to 7 p.m." or "Let's meet every Wednesday ", etc. With only connected time-intervals, we can still find ways to express the above scenarios, but only in awkward ways.

We have made Time-Point and Time-Interval disjoint classes. It would be theoretically sound to let Time-Interval be the class of all sets of points on the time line, so that Time-Point becomes a subclass of Time-Interval where there is only a single time point in the set. That choice, however, leads to a more cumbersome recursive formalization in which the start and end points of an interval are themselves intervals each of which has its own start and end points, etc. Our formalization also supports the requirement of the Allen relations [2] on Time-Interval that the starting point and ending point of a time interval be distinct so that the relations are disjoint from each other.

Down in the hierarchy tree, Convex-Time-Interval, which corresponds to connected intervals on the number line, is now a subclass of Time-Interval. Useful subclasses of Convex-Time-Interval are Calendar-Month and Calendar-Day. (Other subclasses such as Calendar-Year or Calendar-Week can be easily added.) Calendar-Month has 12 subclasses, Calendar-Month-January through Calendar-Month-December, Calendar-Day has subclasses, Calendar-Day-1 through Calendar-Day-31, and Calendar-Sunday through Calendar-Saturday. "January, 1999" is an instance of Calendar-Month-January, etc.

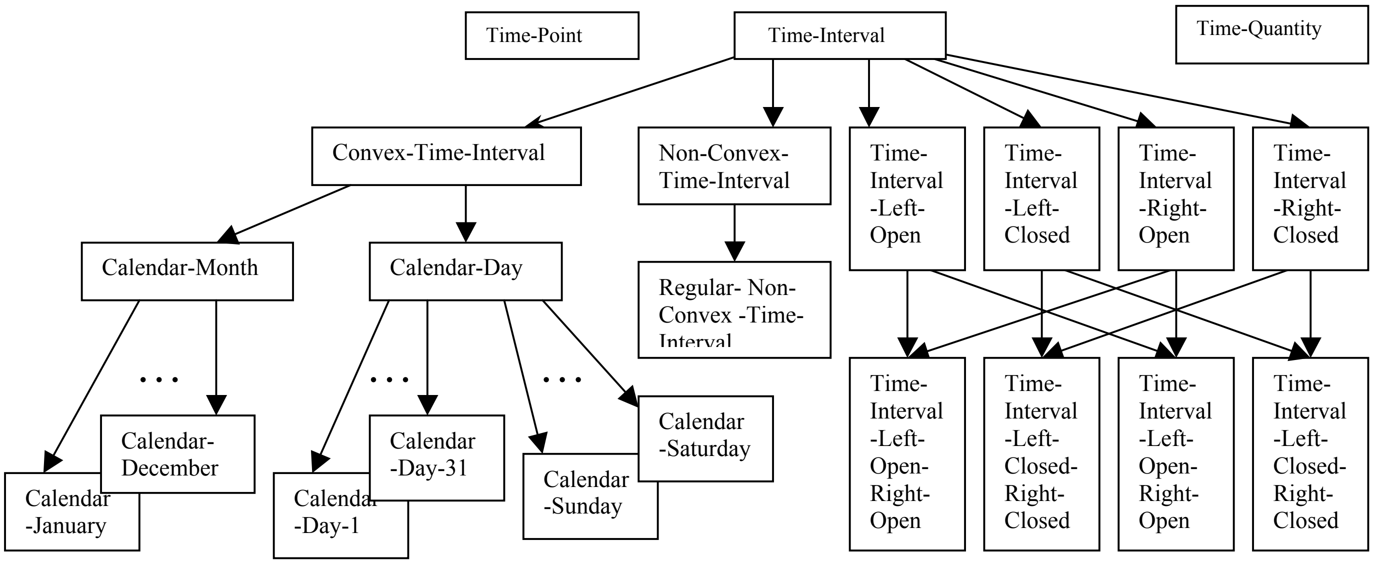

Figure 1. Class Hierarchy of Time Ontology

Non-Convex-Time-Interval and Convex-Time-Interval are a disjoint and complete decomposition of Time-Interval. Non-Convex-Time-Interval is the class of time intervals that are not connected, i.e. with "holes" in them.

Regular-Non-Convex-Time-Interval is a subclass of Non-Convex-Time-Interval. This class is handy for representing regularly recurring events. For example, "every Wednesday in September" can be an instance of Regular-Non-Convex-Time-Interval. It consists of 4 or 5 (depending on which year) connected intervals, each of which represents a Wednesday. The connected intervals contained in an instance of Non-Convex-Time-Interval must all be instances of the same subclass of Convex-Time-Interval, as in the above example, class Calendar-Wednesday.

Class Time-Quantity is a subclass of Physical-Quantity from library ontology "Physical-Quantities" [20]. A time quantity is an "amount" of time that is represented by a real number and a time unit. Relation Magnitude from the Physical-Quantities ontology has three arguments, a physical quantity, a unit, and a real number. If Q is a time quantity of 90 minutes, then "(Magnitude Q Minute 90)" and "(Magnitude Q Hour 1.5)" would both be true. Conversion between different time units can be done using Magnitude. Adding two time quantities is analogous to adding two real numbers.

Functions and Relations

We defined the following functions on domain Time-Point. Function Location-Of maps a time point to a time quantity (i.e., a duration) that is the amount of time from "point zero" on the time line to the time point. That time quantity locates the time point on the time line and can be used to determine relations between two time points. Functions Year-Of, Month-Of, Day-Of, Week-Day-Of, Hour-Of, Minute-Of and Second-Of are functions defined for convenience in terms of the value of Location-Of. In many domains, it is natural to specify time points by year, month, day, etc. In that case, the function value of Location-Of can be calculated from those specified function values. Year-Of, Hour-Of, Minute-Of and Second-Of have range Integer, Month-Of and Day-Of and Week-Day-Of have range Calendar-Month-Type, Calendar-Day-Type and Calendar-Week-Day-Type respectively. Calendar-Month-Type is a class of classes whose instances are the 12 subclasses of Calendar-Month. Calendar-Day-Type is also a class of classes whose 31 instances are classes Calendar-Day-1 through Calendar-Day-31. Similarly for Calendar-Week-Day-Type.

Three binary relations are defined on Time-Point: Before, After, and Equal-Point, which correspond to "<", ">", and "=" on the number line, respectively.

Three basic functions on domain Time-Interval are Starting-Point (range Time-Point), Ending-Point (range Time-Point), and Duration (range Time-Quantity). Starting-Point is defined by:

```
(=> (and (Time-Point ?s) (Time-Interval ?i))
    (<=> (Starting-Point ?i ?s)
         (and
           (not (exists (?j)
                  (and (Time-Point ?j)
                       (Before ?j ?s)
                       (Point-In-Interval
                         ?j ?i))))
           (forall (?p)
             (=> (Time-Point ?p)
                 (or
                   (exists (?k)
                     (and (Time-Point ?k)
                          (Before ?k ?p)
                          (Point-In-Interval
                            ?k ?i)))
                   (Equal-Point ?p ?s)
                   (Before ?p ?s)))))))
```

That definition says that a time point S is the starting point of a time interval I if and only if S is the greatest lower bound of the points in I. Ending-Point is similarly defined as the least upper bound of the points in the interval.

A duration is intended to represent an "amount of time" such as a century, 25 minutes, or "as long as it takes for the kettle to boil". For Convex-Time-Interval, the function value for Duration is the length of the time elapsed between the two end time points, i.e., the difference between the Location-Of function values of the two end points. The value of Duration of a non-convex time interval is the sum of durations of all convex time intervals contained in the non-convex time interval. (Additional functions on Time-Interval (i.e., In-Year, In-Month, In-Day, In-Hour, In-Minute and In-Second) are inlcuded to make it easier to conclude attributes associated with a time interval.)

Point-In-Interval, as in the above axiom for Starting-Point, is a primitive1 relation in our ontology, with domain Time-Point and range Time-Interval.

We define the entire set of 13 Allen relations on Time-Interval as well as the relations on Time-Interval defined in the HPKB-Upper-Level-Kernel ontology for compatibility. These definitions are based on comparisons of their starting points and ending points. For example, Precedes is defined by:

```
(<=> (Precedes ?i ?j)
     (and (Time-Interval ?i)
          (Time-Interval ?j)
          (Before (Ending-Point ?i)
                  (Starting-Point ?j))))
```

Binary relation Contains-Convex-Time-Interval is defined with domain Non-Convex-Time-Interval and range Convex-Time-Interval. It is defined by:

[1] A primitive relation is one that is not defined by any other relations.

```
(<=> (Contains-Convex-Time-Interval ?i ?j)
     (and (Non-Convex-Time-Interval ?i)
          (Convex-Time-Interval ?j)
          (=> (and (Time-Point ?p)
                   (Point-In-Interval
                     ?p ?j))
              (Point-In-Interval ?p ?i))))
```

Binary relation Characteristic-Time-Interval-Of is defined with domain Regular-Non-Convex-Time-Interval and range Convex-Time-Interval-Type. All subclasses of Convex-Time-Interval are instance of Convex-Time-Interval-Type. For example, this relation holds for "every Wednesday" and Calendar-Wednesday.

Two ternary relations Plus and Minus for Time-Point and Time-Interval are analogous to '□' and '–' in set theory.

Consider an example of defining "a week in January" using some of the classes and relations. This is a class each of whose instances is a particular week in January of a particular year. The definition would be as follows:

```
(and (subclass-of WiJ Convex-Time-Interval)
     (=> (WiJ ?w)
         (and
           (Duration ?w
                     (The-Quantity Day 7))
           (exists (?j)
             (and (Calendar-January ?j)
                  (or (During ?w ?j)
                      (Costarts ?w ?j)
                      (Cofinishes ?w
                                  ?j)))))))
```

"During", "Costarts" and "Cofinishes" are three Allen relations. "The-Quantity" is the constructor for time quantities. The definition says that each week in January W is a convex time interval whose duration is 7 days and for which there exists a "calendar January" J such that W is during J or W costarts J or W cofinishes J.

Infinity and Density

In order to address time infinity, we defined two instances of Time-Point: Infinite-Past and Infinite-Future. For example Infinite-Past is defined by:

```
(and (Time-Point Infinite-Past)
     (=> (Time-Point ?p)
         (not (Before ?p Infinite-Past))))
```

For time density, we have the following axiom:

```
(=> (and (Time-Point ?i)
         (Time-Point ?j)
         (Before ?i ?j))
    (exists ?k (and (Before ?i ?k)
                    (Before ?k ?j))))
```

Some Design Issues

Closed interval or Open interval?

The time ontology from Cycorp [4] and the previous time ontologies from Ontolingua [3] do not specify whether Time-Interval is open or closed. In many knowledge domains, one does not care about single points in an interval, so open or closed intervals do not matter. But there are domains in which people do care about end points, E.g., describing a ball tossed straight upwards. We want to express the interval in which the ball has a positive velocity. Any time ontology with only closed intervals is insufficient. However, if we want to express the interval in which the ball has nonnegative velocity, then any ontology with only open intervals does not work. Though we can say the ball has non-zero acceleration in the open interval and on the point when the ball has zero velocity, this representation is of course cumbersome.

So, in our ontology, we have subclasses of Time-Interval: Time-Interval-Left-Open, Time-Interval-Left-Closed, Time-Interval-Right-Open, and Time-Interval-Right-Closed. Then, Time-Interval-Open is a subclass of both Time-Interval-Left-Open and Time-Interval-Right-Open, and Time-Interval-Closed is a subclass of both Time-Interval-Left-Closed and Time-Interval-Right-Closed. We then defined finer grained relations for these subclasses. E.g., relation Meets is defined by

```
(<=> (Meets ?i ?j)
     (and (Time-Interval ?i)
          (Time-Interval ?j)
          (Equal-Point
            (Ending-Point ?i)
            (Starting-Point ?j))))
```

Meets has three subrelations, Meets-2Open, Meets-2Closed and Meets-OpenClosed. E.g., Meets-OpenClosed is defined by

```
(<=> (Meets-OpenClosed ?i ?j)
     (and
       (or (and
             (Time-Interval-Right-Open ?i)
             (Time-Interval-Left-Open ?j))
           (and
             (Time-Interval-Right-Open ?i)
             (Time-Interval-Left-Open ?j)))
       (Meets ?i ?j)))
```

Differentiating the three cases of "meets" allows users to give them different properties. For example, users might want "meets" in Meets-2Closed to mean "overlaps" in some knowledge domains. Having both the parent relation and subrelations gives users much flexibility. Users in domains where it is not necessary to distinguish open and closed intervals will not use the open/closed subclasses of Time-Interval, and they will choose to use relation Meets. Users working with domains of physics experiment will use the subclasses and thus the subrelations of Meets to represent motions of a metal ball.

Time Granularity

In section "Functions and Relations" above, we said that the Location-Of function value of a time point locates this time point on the time line as a real number locates a point on a number line. However, we cannot always identify this real number, i.e. we cannot measure time with indefinite accuracy. So, one must address the issue of time granularity in a time ontology. When we are describing some ideal physics experiment, time can be specified with perfect accuracy. At other times, "day" would be an appropriate primitive time unit for an American history book, "minute" works for a daily class schedule, "microsecond" is used in measuring CPU time, etc. Since our ontology is intended to be multi-use, it must be able to handle different levels of granularity.

In our ontology, granularity is specified for time points, not for time intervals. A time point with a certain level of granularity is a single time point with the uncertainty that it may be anywhere in a certain time interval. For example, time point "Jan 1st, 2002" with day granularity is a single time point that can be any point within the convex time interval starting midnight of Dec 31st, 2001, and ending midnight of Jan 1st, 2002. For time points with day granularity, the time line, in both directions and starting from "point zero", is evenly divided into time slots of duration one day. The Location-Of function value for a time point only determines the slot in which the time point locates, in much the same way as truncating real numbers to integers, 1.56 to 1, -5.99 to –5, etc.

Function Granularity-Of is defined on domain Time-Point, with range Time-Granularity. Class Time-Granularity is defined in our ontology. There can be an arbitrary number of levels of granularity since every time quantity can correspond to a level of granularity. But in practice, we only have granularities for commonly used time units. Year-Granularity, Month-Granularity, Day-Granularity, Hour-Granularity, Minute-Granularity and Second-Granularity are defined as instances of Time-Granularity in our ontology. But in our ontology, other levels of granularity can be added when we need finer granularity than second or a granularity for Day-On-Planet-X of, say, 12.3456 hours. Function Time-Unit-Of is defined on domain Time-Granularity with range Time-Unit, a subclass of Unit-Of-Measure in ontology "Physical-Quantities"[20]. The function Time-Unit-Of maps Year-Granularity into Year, etc. Then there is a special instance of Granularity, Infinitely-Fine-Granularity, that does not have a Time-Unit-Of function value. When users do not care to use granularity, as when describing an ideal physics experiment, all time points are assumed to have infinitely fine granularity.

The three basic binary relations on Time-Points (i.e., Before, After, and Equal-Point) are fundamental to defining any relation on Time-Interval on different levels of granularity. For example, relation Equal-Point is defined as follows. Two time points are equal either when they both have infinitely fine granularity and exactly coincide with each other on the time line, or when they have the same granularity and fall in the same time slot. That definition is as follows:

```
(=> (and (Time-Point ?i) (Time-Point ?j))
   (<=> (Equal-Point ?i ?j)
        (or
          (and
            (Granularity-Of
              ?i
              Infinitely-Fine-Granularity)
            (Granularity-Of
              ?j
              Infinitely-Fine-Granularity)
            (= (Location-Of ?i)
               (Location-Of ?j)))
          (and
            (Granularity-Of ?i ?gran)
            (Granularity-Of ?j ?gran)
            (= (LINLT
                 (Magnitude
                   (Location-Of ?i)
                   (Time-Unit-Of ?gran)))
               (LINLT
                 (Magnitude
                   (Location-Of ?j)
                   (Time-Unit-Of
                     ?gran))))))))
```

LINLT is the "largest integer not larger than" relation.

Two time points on two different levels of granularity cannot be said to be equal to each other. However, we define Before or After to hold for two time points on different levels of granularity when the two time slots containing the time points don't overlap. For example, time point "Year 1970" is before time point "Jan 1st, 1999".

In previous ontologies, e.g. Simple Time [3], there were no axioms dealing with the granularity problem. When comparing "today" and "today at 10 a.m.", the result would depend on the default value that specifies the hour of "today". If the default value is zero, it would be inferred that "today" is before "today at 10 a.m.", which is not a desirable result.

Class Calendar-Month

The need to associate attributes with each calendar month leads to using Calendar-January instead of 1 and Calendar-February instead of 2, etc. for the months of the year. These classes in the ontology store many calendar facts about each month, for example its number of days. In previous ontologies, relation Month-Length was defined on domain Integer (e.g., "(Month-Length 1 31)"). Here, we define the relation on the 12 subclasses of Calendar-Month and thus make Calendar-Month-Type the domain. We use a similar design for Calendar-Day and their subclasses.

As an example of the use of these classes, consider representing the holidays during the year. We could do that by making Holiday a subclass of Regular-Non-Convex-Time-Interval, and define functions In-Month and In-Day on domain Holiday with range Calendar-Month-Type and Calendar-Day-Type, respectively. "New Year's

Day" would have In-Month function value Calendar-January and In-Day function value Calendar-Day-1. We could define a ternary relation In-Week-Day with its three arguments being an instance of Holiday, an instance of Calendar-Week-Day-Type, and an integer, respectively. "Father's Day", which is the second Sunday in June, could then be defined by:

```
(and (Holiday FsD)
     (In-Month FsD Calendar-June)
     (In-Week-Day FsD Calendar-Sunday 2))
```

At the same time, we could define relation Has-Holiday with range Holiday on domain Calendar-Month-Type, Calendar-Day-Type and Calendar-Week-Day-Type. For example, Has-Holiday would hold for Calendar-January and "New Year's Day". Holiday information can be well integrated into the calendar in this representation.

Summary and Future Work

To summarize, we developed a formal time ontology upon which other knowledge engineers can build larger ontologies that utilize the notion of time. Knowledge engineers can choose to use the portion of our ontology that fits their needs, whether they need open or closed intervals, or they need granularity of a second or a year, or they need to specify calendar weekdays.

Future work includes building special purpose reasoners for this time ontology, so that it can be used more efficiently in large-scale systems. Other tasks include building a more complete set of relations across different time granularities, and adding branching time to support hypothetical reasoning when needed.

References

[1] P. Hayes; A Catalog of Temporal Theories; Tech report UIUC-BI-AI-96-01, University of Illinois; 1995.

[2] J. Allen; Maintaining Knowledge about Temporal Intervals; in Communications of the ACM, 26(11), pp.832-843, November 1983

[3] Ontology Simple Time. Available through http://ontolingua.stanford.edu

[4] Time ontology as part of HPKB-Upper-Level-Kernel. Available at http://ontolingua.stanford.edu and at http://www.cyc.com/products2.html#kb

[5] D. McDermott; A Temporal Logic for Reasoning about Processes and Plans; in Cognitive Science, 6:101-155, 1982

[6] Y. Shoham; Temporal Logics in AI: Semantical and Ontological considerations; in Artificial Intelligence, 33:89-104, 1987.

[7] F. Bacchus, J. Tenenberg, and J. Koomen; A Non-reified Temporal Logic; in Proc. KR'89, pages 2-10, 1989.

[8] A. Galton; A Critical Examination of Allen's Theory of Action and Time; in Artificial Intelligence, 42:159-188, 1990.

[9] A. Bochman; Concerted Instance-Interval Temporal Semantics: Temporal Ontologies; in Notre Dame Journal of Formal Logic, 31(3):403-414, 1990.

[10] L. Vila, E. Schwalb; A Theory of Time and Temporal Incidence Based on Instants and Periods.

[11] P. Terenziani; Reasoning about Periodic Events.

[12] D. Cukierman, J. Delgrande; A Language to Express Time Intervals and Repetition.

[13] M. Soo; Multiple Calendar Support for Conventional Database Management Systems; Proc. Int. Workshop on an Infrastructure for Temporal Databases, 1993.

[14] A. Montanari; Metric and Layered Temporal Logic for Time Granularity; 1996.

[15] C. Bettini, X. S. Wang, S. Jajodia; A General Framework for Time Granularity and its Application to Temporal Reasoning; Annals of Mathematics and Artificial Intelligence, 1(22):29-58, 1998.

[16] J. Euzenat; Granularity in Relational Formalisms; 1998.

[17] M. R. Genesereth, R. Fikes; Knowledge Interchange Format, Version 3.0 Reference Manual; KSL Technical Report KSL-92-86, 1992.

[18] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, & J. P. Rice; OKBC: A Programmatic Foundation for Knowledge Base Interoperability; Proceedings of the Fifteenth National Conference on Artificial Intelligence; Madison, Wisconsin; July 26-30, 1998. Also, KSL Technical Report KSL-98-08 (http://www.ksl.stanford.edu/KSL_Abstracts/KSL-98-08.html).

[19] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Gunning, M. Burke; The DARPA High-Performance Knowledge Bases Project; AI Magazine, 19(4): Winter 1998, 25-50

[20] Ontology Physical-Quantities. Available through http://ontolingua.stanford.edu