Exploiting inference to improve temporal RDF annotations and queries for machine reading

Robert C. Schrag
Digital Sandbox, Inc.
McLean, VA USA
bschrag@dsbox.com

***Abstract*—We describe existing and anticipated future benefits of an end-to-end methodology for annotating formal RDF statements representing temporal knowledge to be extracted from text, as well as for authoring and validating test and/or application queries to exercise that knowledge. Extraction is driven by a target ontology of temporal and domain concepts supporting an intelligence analyst's timeline tool. Both the tool and the methodology are supported at several points by an implemented temporal reasoning engine, in a way that we argue ultimately advances machine reading technology by increasing both sophistication and quality expectations about temporal annotations and extraction.**

***Index Terms*—temporal knowledge representation and reasoning, extracting formal knowledge from text, machine reading, annotation interfaces and validation**

I. Introduction

Machine reading—that is, automatic extraction of formal knowledge from natural language text—has been a longstanding goal of artificial intelligence. Effective extraction into RDF has the potential to make targeted knowledge accessible in the semantic web. We recently supported a large-scale evaluation of temporal knowledge extraction from text by providing RDF/OWL ontology for target statements and a corresponding reasoning engine for query answering. Along the way, we discovered…

- How inference could improve annotation—the manual extraction of formal temporal statements—and question authoring for evaluation or for applications.
- How, coupled with annotation and question authoring processes, inference could ultimately drive more sophisticated machine reading capabilities.

II. Temporal knowledge representation and reasoning for timeline development

Our temporal logic is based loosely on the event calculus [10], as follows.

A time interval is a convex collection of time points—intuitively, an unbroken segment of a time line. Time intervals begin and end with time points, which may be constrained relative to each other or relative to a calendar. The ontology includes a rich set of relational properties over time points and intervals, and the reasoning engine will calculate tightest inferable bounds between any two points and will detect contradictory time point relation sets.

A fluent is an object-level, domain statement (e.g., FluentA: attendsSchool(Jansa LubljanaUniversity)) whose truth value is a function of time. It is taken to be true at time points where it holds and not to be true at time points where it does not hold. We reify a fluent in an observation—a meta-level statement whose object is a fluent, whose subject is a time interval, and whose predicate is a holds property (e.g., holdsThroughout(FluentA Interval1), when FluentA is observed over Interval1, corresponding to, say, September, 1980).

The events of interest to us, which we call transition events, occur at individual time points and may cause one or more fluents to change truth value. We represent events (like the birth of Jansa) as objects with attribute properties like agent, patient, and location, and we relate events to time intervals with an occurs property (e.g., occursAt(BirthOfJansa Point2), where Point2 is associated with an interval corresponding to the date September 17, 1958). As usual with the event calculus, such events can initiate fluents (e.g., occursAt(BirthOfJansa Point2) initiates FluentB: alive(Jansa Interval3), where Interval3 is begun by Point2) or terminate them (e.g., DeathOfJansa…). The temporal reasoning engine implements appropriate axioms to perform fluent initiation and termination.

Note that an observer may report information about the temporal extent of a fluent without communicating anything about initiation or termination. E.g., if text says *Abdul and Hasan lived next door to each other in Beirut in 1999,* we don't know when Abdul or Hasan may have moved to or from Beirut. When text says *Abdul moved to Beirut in 1995 and emigrated in 2007,* we use the properties clippedBackward and clippedForward regarding the fluent residesInGPE-spec(Abdul BeirutLebanon) to indicate initiation and termination by anonymous (unrepresented) transition events, so that we can also initiate or terminate temporally under-constrained like-fluent observations (e.g. *Abdul lived in Beirut during the 1990s*).

The reasoning engine's implementation, using AllegroGraph, Allegro Prolog, and Allegro Common Lisp from Franz, Inc., can answer any conjunctive query. While not yet heavily optimized, it is at least fast enough to support machine reading system evaluation over newspaper articles where cross-document entity co-reference is not required.

The combined extraction and reasoning capability was conceived to support an intelligence analyst's timeline tool in which a GUI would be populated with statements about entities (e.g., persons) of interest extracted from text. Our evaluation of machine reading capabilities was based on queries similar to those we would have expected from such a tool's API. It also supposed the analysts could formulate their own, arbitrary questions, such as Query 1: *Find all persons who were born in Ljubljana in the 1950s and attended Ljubljana University in the 1980s, the titles that they held, the organizations in which they held these titles, and the maximal known time periods over which they attended and held these titles.*

III. Lessons Learned and Realized in Implementation

This indirect, query answering style of machine reading evaluation makes it especially important that we perform effective quality control of formal queries in the context of the formal statements we expect to be extracted from answer-bearing documents. We thus developed the test query validation approach illustrated in Figure 1. Considering Query 1's formalization (see Figure 10 in section IV.B), it's worth noting that we used the methodology illustrated here to debug a number of subtle errors occurring in our earlier (manual) formulations. When each such formulation did not result in the answers expected, we traced inference to identify a point of failure, corrected this, and then iterated until correct.

Our machine reading technologists told us early on that they preferred macro-level relational interfaces that would streamline away micro-level details of time points and intervals. We thus provide a language of flexible specification strings (spec strings) that expand to create time points, intervals, and relations inside our reasoning engine. We also provide ontology to associate the temporal aspects Completed, Ongoing, and Future with fluents (e.g., *he used to live there* vs. *he is living there* vs. *he is going to live there*) and with the reporting dates of given articles, to afford a relational interface reasonably close in form to natural language sources. For the Completed or Future aspects, we also can capture quantitative lag or lead information (e.g., *he lived there five years ago* or *he will move in five days from now*).

IV. Lessons Learned with Implementation Proposed

Here, we propose some further approach elements that we expect to lead to high-quality temporal annotations, including…

A. Interfaces and workflows deliberately designed to support capture of all statements specified as extraction targets (see section A)

B. Graphical time map display including fluents and events (see section B)
   - On-line inference to elucidate inter-relationships and potential contradictions
   - Visual feedback to let users help assure quality themselves
   - Time map-based widgets supporting user knowledge entry

C. Technology adaptable to test or application question authoring (see section C)

D. Quantitative temporal relation annotation evaluation (see section V.A).

A. Annotation workflows

Fluents are simple statements that we can readily represent in RDF. The example in Figure 2 focuses on the fluent about one Janez Jansa attending Ljubljana University—only on the fluent, not the full observation including temporal information (i.e., only that Jansa attends the school, not when). The technology needed to annotate such information is well understood and (excepting perhaps the last bullet about modality) has been well enough exercised that we may routinely expect good results. This includes multi-frame GUIs where a user can produce stand-off annotations by highlighting text and by clicking in drop-down boxes select relations, classes, and instances. In part because these tools have preceded reading for formal knowledge extraction, they may not use our intended representation internally—i.e., they may for historical reasons internally use a representation (e.g., XML that is not RDF) tailored to linguistic phenomena rather than associated with any formal ontology.

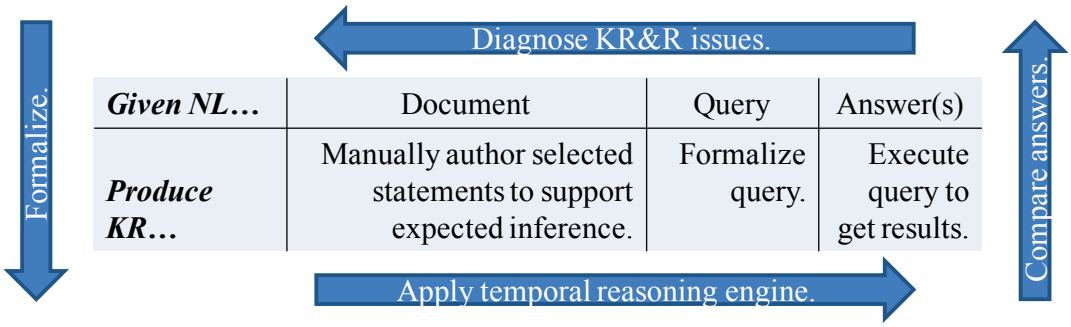


Figure 1. We validate test queries by making sure that natural language (NL) and formal knowledge representation (KR) versions of documents, queries, and answers agree, diagnosing and debugging as necessary.

1. Select relation.
2. Specify argument identifiers, respecting co-reference.
3. Select / highlight / designate corresponding text.
4. Capture any counter-factual modality info.

Figure 2. The workflow to annotate a fluent

1. Select one of time interval or point.
2. Capture any beginning date and backward clipping info.
3. Capture any ending date and **forward clipping info**.
4. Capture any duration info.
5. If ending point is unconstrained w.r.t. reporting date:
   a. Capture reporting aspect.
   b. Capture any reporting lag info.
6. Capture any other relative temporal info available.

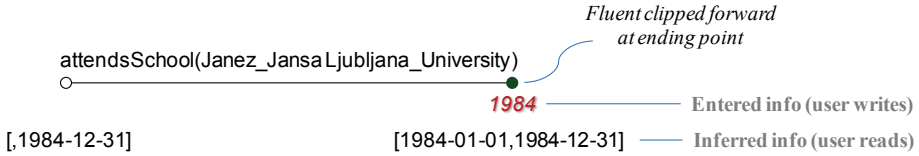


Figure 3. The workflow to annotate a holds statement (a fluent observation)

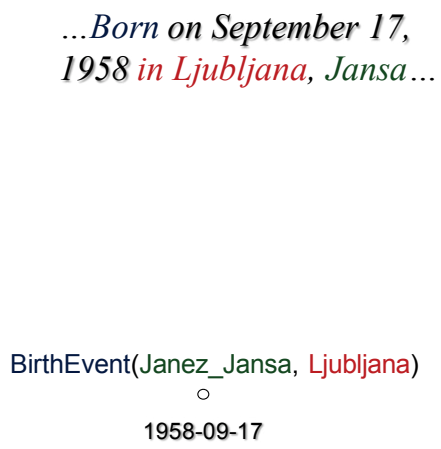
Capturing the temporal information associated with the given observation of a fluent in a holds statement requires following a sequence of actions and decisions in a deliberately designed workflow, as outlined in Figure 3. We have highlighted, by color- and typeface-coding, some temporal elements in the source text, along with corresponding steps in the workflow and elements of the associated graphical representation.

Addressing the workflow step by step, we see that:

- There is no reason to believe Jansa attended school for only one day (the time unit associated with a time "point" in our machine reading evaluation), so we choose a time interval (and the predicate holdsThroughout) rather than a time point (and holdsAt). Schrag [8] argues that holdsAt almost never is appropriate, and in future this step may be omitted.
- We find no beginning date information. (In the absence of such information, there is no benefit to asserting backward clipping.)
- We find (and have highlighted) a coarse-grained ending date (1984). We indicate that our fluent is clipped forward, assuming that Jansa no longer attends the school after graduation.
- There is no duration information. (We don't know how long Jansa was at school.)
- The ending point is well before the reporting date, so we skip to the next step.
- There is no other relevant temporal information.
- To indicate clipping, the graphic fills the time point symbol (making it solid).

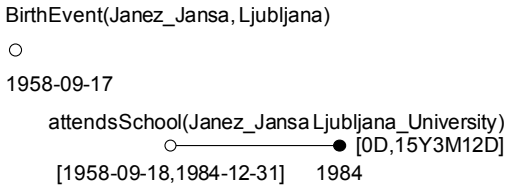
Our reasoning engine expands the entered coarse date 1984 into earliest and latest possible calendar dates bounding the observation's ending point. It also infers an upper bound on its beginning time point.

As illustrated in Figure 4, we invoke a similar workflow for event occurrence. Because our representation for events is simpler than that for observations, this workflow has fewer steps. Our ontology treats birth as a fluent transition event—it occurs at a given time point, and it causes a transition of the vital status of the person born (from FuturePerson to Alive). Our graphical representation here accordingly just depicts a single time point (not an interval). We can use basically the same workflow to capture a non-transition event (e.g., a legal trial) that occurs over more than one time point.



1. Select event type.
2. Specify argument identifiers, respecting co-reference.
3. Select / highlight / designate corresponding text.
4. Capture any hypothetical modality info.
5. Capture any date info.
6. If an event's date is otherwise unconstrained w.r.t. reporting date:
   a. Capture reporting aspect.
   b. Capture any reporting lag info.
7. Capture any other relative temporal info available.

Figure 4. Workflow to annotate a transition event



Automatically:

- Display in order any time points that are ordered unambiguously.
- Display inferred bounds.
  - *Rules:* Can't attend school before being alive; being born makes one alive.
- Highlight bounds contradictions.

On demand:

- Trace back from bounds to user-entered information.
  - Date of the birth event
- Display / hide entered or inferred bounds on…
  - Beginning points, ending points
  - Durations
- Focus on a particular time window, location, person, …
- Highlight time points that are ordered / unordered w.r.t. to a selected, reference time point.

Figure 5. A time map with both a fluent and a transition event

B. Displaying integrated time maps

Figure 5 illustrates a time map including both the birth event and the school attendance fluent from earlier figures. It also suggests functional requirements to be satisfied automatically/by default and upon user demand. Note that we now have automatically displayed—from on-line temporal inference—a lower bound on the fluent observation's beginning date: Jansa could not have attended school until after he was born. (The "day" time point granularity used in our machine reading evaluation leads to some non-intuitive effects, like not being alive until the day after one is born. We can easily correct this using an interval constraint propagation arithmetic including infinitesimals [6][7][8].) We've also indicated bounds on the fluent observation's duration (calculated as ending date bounds interval minus starting date bounds interval). Propagating effects like this can maximize visual feedback to users, expanding their basis for quality judgments about the information they enter. If any inferred bound seemed odd, a user could click on it to identify which of his/her own entered information (then highlighted in the display) might be responsible. The time map display tool would automatically launch such an interaction when it detected a contradiction among inputs.

The time map displayed in Figure 6 includes all the information from the source text that is necessary to answer Query1. The last fluent observation (at bottom right, where Jansa is prime minister) exercises workflow steps that earlier time map elements don't. We have no ending date for this observation, but we do have present tense reference to Jansa as *the prime minister,* so we appeal to the reporting aspect Ongoing. From the source text *he was elected prime minister on November 9, 2004,* we can bound the observation's beginning point.

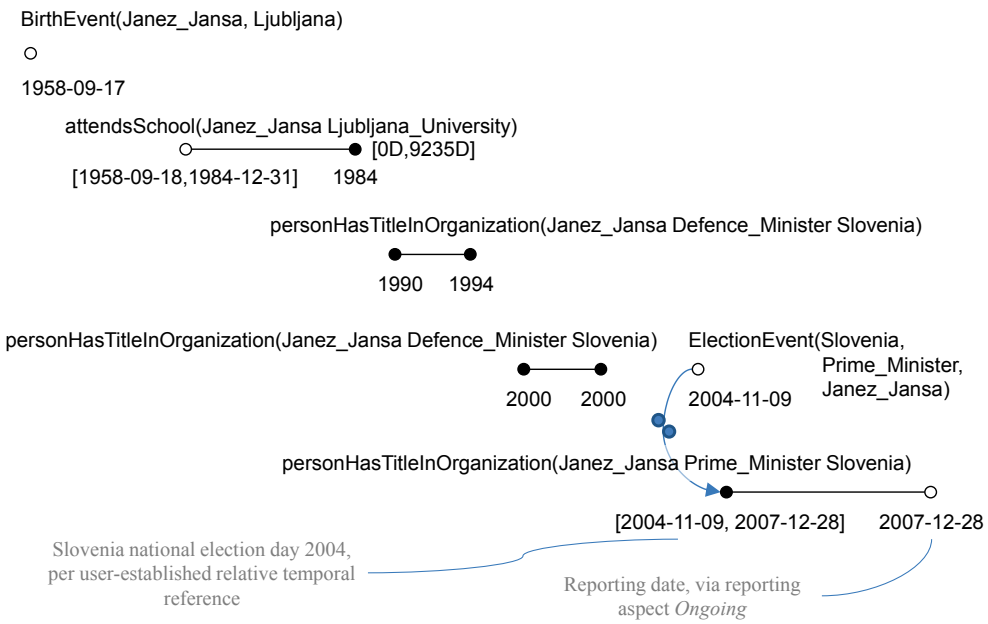


Figure 6. A time map with more statements extracted from the same article

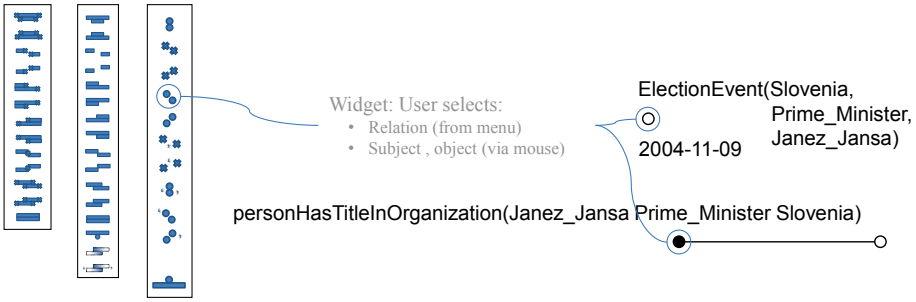


Figure 7. Using the GUI to establish relative temporal reference

Our user also has entered the election event. An election is not necessarily a fluent transition event, at least in that an elected candidate does not always take office immediately. So, we rely on the user to establish relative temporal reference between the election event and the fluent observation's beginning. See the depicted constraint, whose entry is illustrated in Figure 7. Establishing relative temporal reference requires the selection of a pair of time points and/or intervals to be related and of an appropriate temporal relation between them. Here, we just need the time point at which the election occurs to be less than or equal to the time point at which Jansa takes office.

While a few common relations may be all that most users will ever need, we do have a lot of relations [8] that a user could in principle choose from. We should be able to provide access to these effectively, so that our user is empowered without being overwhelmed.

Figure 8 shows formal statements that would be created directly by the user's actions (i.e., not also including those created indirectly by inference) in entering the information reflected in our finished time map. We have highlighted fluents and some other key statements, each of which appears near several related statements. Our time map represents Jansa's birth event in a non-standard way, repeated here in different color type, beside the italicized, standard statements. We have not similarly formalized the event of Jansa's election as PM, and Figure 8 includes just statements about that event's point of occurrence.

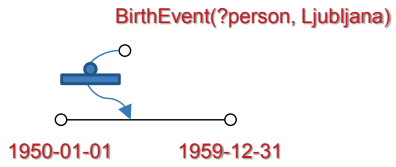
Clearly, we can do a lot of formal work for the user behind the scenes.

F_school: attendsSchool (Janez_Jansa Ljubljana_University)
holdsThroughout(F_school I_school)
clippedForward(F_school I_school)
hasTimeIntervalSpecString(I_school [,1984])
hasPersonBorn(birth Janez_Jansa)
occursAt(birth P_birth)
hasTimePointSpecString(P_birth 1958-09-17)
hasPersonBorn(birth Janez_Jansa)
hasBirthEventGPE-spec(birth GPEspec)
hasCityTownOrVillage(GPEspec ljubljana_Ljubljana_Slovenia)
hasNationState(GPEspec Slovenia)
type(Defence_Minister MinisterTitle)

BirthEvent(Janez_Jansa, Ljubljana)

F_PTIO_DM_1: personHasTitleInOrganization(Janez_Jansa Defence_Minister Slovenia)
holdsThroughout(F_PTIO_DM_1 I_PTIO_DM_1)
clippedBackward(F_PTIO_DM_1 I_PTIO_DM_1)
clippedForward(F_PTIO_DM_1 I_PTIO_DM_1)
hasTimeIntervalSpecString(I_PTIO_DM_1 [1990,1994])
F_PTIO_DM_2: personHasTitleInOrganization(Janez_Jansa Defence_Minister Slovenia)
holdsThroughout(F_PTIO_DM_2 I_PTIO_DM_2)
clippedBackward(F_PTIO_DM_2 I_PTIO_DM_2)
clippedforward(F_PTIO_DM_2 I_PTIO_DM_2)
hasTimeIntervalSpecString(I_PTIO_DM_2 [2000,2000])
F_PTIO_PM: personHasTitleInOrganization(Janez_Jansa Prime_Minister Slovenia)
holdsThroughout(F_PTIO_PM I_PTIO_PM)
clippedBackward(F_PTIO_PM I_PTIO_PM)
hasBeginningTimePoint(I_PTIO_PM I_PTIO_PM_beginning)
hasTimePointSpecString(Slovenia_2004_Election_Day 2004-11-09)
timePointGreaterThanOrEqualTo(I_PTIO_PM_beginning Slovenia_2004_Election_Day)
hasReportingAspect(I_PTIO_PM Ongoing)
ref(annotation I_PTIO_PM)
annotation(document annotation)
hasReportingChronusSpecString(document 2007-12-28)

Figure 8. Formal statements associated with the time map in Figure 6



Query 1: *Find all* ***persons who were born in Ljubljana in the 1950s*** *and* ***attended Ljubljana University in the 1980s****,* the titles that they held, the organizations in which they held these titles, *and the maximal known time periods over which* ***they attended*** *and* held these titles.

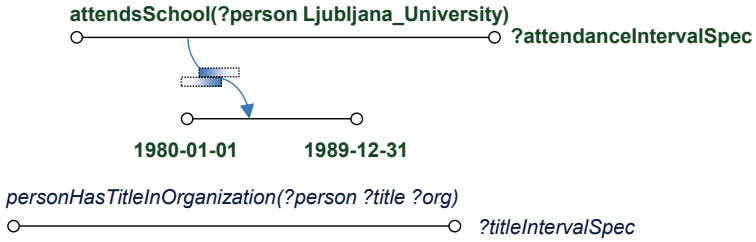


Figure 9. The time map covering our Query1

C. Adaptation to test or application question authoring

We might reuse much of the same machinery in a question authoring interface, in which a user can formalize a query, as illustrated for Query1 in Figure 9. This time map display is even less cluttered than the one for this query's supporting statements, for a couple of reasons.

- We are making general statements, rather than specific ones, so don't use as many dates or long identifiers. Rather, we use variables (here beginning with ?).
- We are asking about only one answer (set of variable values satisfying the query) at a time. The supporting statements in our earlier time map include three separate sets of bindings for the variables ?title and ?org.

We have introduced intervals to represent *the 1950s* and *the 1980s,* and we have selected time point/interval relationships appropriate to the query's conditions. These relationships are associated with particular idioms used in our formalization in Figure 10.

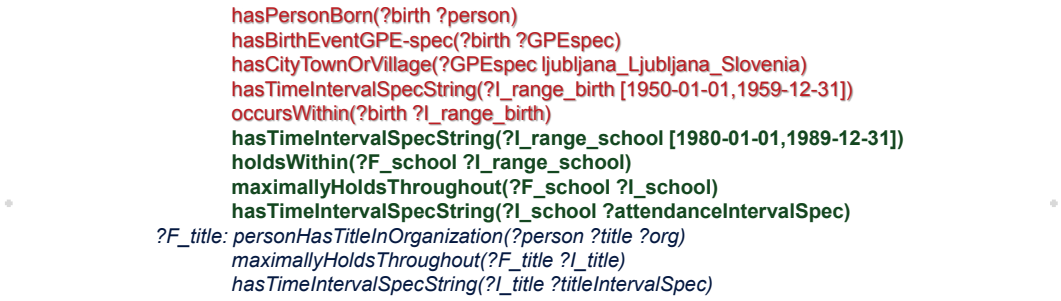


Figure 10. Formalization of the query in Figure 9's time map, covering Query 1

Our query asks about the "maximal known time periods" over which the fluents hold, and we associate (via a query authoring workflow step) an "interval spec" variable with each fluent's observation interval. Per our formalization, this will be bound, on successful query execution, to a string that describes lower and upper bounds on the observation interval's beginning point, ending point, and duration. The formalization uses the properties occursWithin (for *born in the 1950s*) and holdsWithin (for *attended school in the 1980s*) to accommodate the temporal relations selected for the query authoring time map. We know to use maximallyHoldsThroughout (vice the less restrictive holdsThroughout) for the fluents' observation intervals because the query's author has included (via the invoked widget) associated spec string variables.

Thus, it appears that we might enable non-specialists to author effective test queries (or, in a transition/application setting, domain queries), without requiring the intervention of a KR specialist. One angle on this proposed work might be to determine the extent to which readers who are not (temporal) knowledge representation specialists can perform such tasks consistently—alternatively, to determine the amount of training (e.g., pages of documentation, number of successfully completed test exercises) required to qualify an otherwise-non-specialist to perform the task well. That said, rather than "dumb down" the task, to accommodate non-expert readers, we propose to ratchet up annotator performance expectations—to achieve the highest-quality results possible so that we can drive research regarding extraction of temporal knowledge by machines from text to new levels of sophistication. The machine reading researchers whose systems are under evaluation quite reasonably ask, before they embark on a mission of technological advancement, "Is this task feasible for humans, with acceptable consistency?" We'd like to answer that question in the best way that we can.

V. Related Work and Proposed Advances

Beyond test questions and answers, the entire machine reading community would benefit from having a large volume of good temporal logic annotations available. Time is a key topic in language understanding, engendering much current community interest. TimeML [4], which emphasizes XML annotation structures rather than RDF ontology and relationships, has been used in the TempEval temporal annotation activities (see, e.g., www.timeml.org/tempeval2/) and advanced as an international standard [5]. We are interested in exploring the synergy between this work and ours.

Others have applied limited temporal reasoning in post-processing of temporal annotations, to…

A. Compute the closure of qualitative pairwise time interval relations, as one step in assessing a machine reader's precision and recall performance (see section A)
B. Ascertain the global coherence of captured qualitative relations (see section B).

Our implementation can go further, as described below.

A. Quantitative temporal relation annotation evaluation

Evaluating temporal annotations typically has been limited to (Allen's [1]) qualitative relations (e.g., before, overlaps, contains), and quantitative information about dates and durations typically has been evaluated only locally—at the level of temporal expressions (AKA "TIMEXs" [3]). The reasoning applied has been strictly interval-based, neglecting important quantitative information about dates and durations widely available in text. This approach is taken by Setzer et al. [9], e.g.

Our temporal reasoning engine, which is point-based, naturally accommodates arbitrary bounds on the metric durations that separate time points and uses global constraint propagation to calculate earliest and latest possible dates/times for any point (including the beginning and ending points of all temporal intervals), as well as tightest bounds on durations.

This approach also usually affords sufficient global perspective for a robust recall statistic. Adapting the standard approach for evaluating interval relations [11], we can discard from our gold standard annotations any redundant relations until we determine a set spanning globally calculated bounds. Then we can count members of this spanning set whose addition to a user's candidate set results in tightening of bounds in the latter, to determine recall.

Only when every member of a set of points is unrelated to the calendar (i.e., we have only point ordering and interval duration information) do we lack calendar bounds supporting meaningful recall assessment. Then, however, by choosing any point in a connected set to serve as a reference (in place of the calendar), we can apply the same approach as above.

It may reasonably be argued that at some threshold of representational complexity the brute force transitive closure-and-spanning tree approach to computing recall and precision of an extracted knowledge base (set of statements) must become impractical. Our quantitative temporal statements are certainly richer than the typical qualitative ones, and (depending on knowledge base size) we may be pushing up

against this threshold with them. Our query answering evaluation paradigm is more broadly applicable, presuming inference over extracted statements remains tractable—for the queries of interest.

B. Ascertaining global coherence

Waiting until annotation is done to infer bounds and detect contradictions neglects opportunities to give annotator's (e.g., time map-based) feedback and receive their best-effort corrections. As Bittar et al. [2] comment, "Manually eliminating incoherencies is an arduous task, and performing an online coherence check during annotation of relations would be extremely useful in a manual annotation tool." We propose this.

VI. SUMMARY

We have outlined existing and anticipated future benefits of an end-to-end methodology for…

- Annotating formal RDF statements representing temporal knowledge to be extracted from text
- Authoring and validating test and/or application queries to exercise that knowledge.

These capabilities are supported by an implemented temporal reasoning engine. They and the engine are intended to support a timeline tool conceived for use by intelligence analysts. We have explained how these benefits can advance machine reading technology by increasing both sophistication and quality expectations about temporal annotations and extraction.

ACKNOWLEDGMENT

Thanks to other participants in DARPA's Machine Reading research program—especially to other members of the SAIC evaluation team, including Global InfoTek (the author's former employer).

The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. (Approved for Public Release, Distribution Unlimited)

REFERENCES

[1] J. Allen, "Maintaining knowledge about temporal intervals," in Communications of the ACM. 26, pp. 832–843, November 1983.

[2] A. Bittar, P. Amsili, P. Denis, L. Danlos, "French TimeBank: An ISO-TimeML annotated reference corpus," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pp. 130–134, 2011.

[3] L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson, "TIDES 2005 standard for the annotation of temporal expressions," MITRE Corporation, 2005.

[4] J. Pustejovsky et al., "TimeML: Robust specification of event and temporal expressions in text," AAAI Technical Report SS-03-07, 2003.

[5] J. Pustejovsky, K. Lee, H. Bunt, and L. Romary, "ISO-TimeML: an international standard for semantic annotation," in Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC), Malta. May 18-21, 2010.

[6] R. Schrag, J. Carciofini, and M. Boddy, "Beta-TMM Manual (version b19)," Technical Report CS-R92-012, Honeywell SRC, 1992.

[7] R. Schrag, M. Boddy, and J. Carciofini. "Managing disjunction for practical temporal reasoning," in Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR-92), pp 36–46, 1992.

[8] R. Schrag, "Best-practice time point ontology for event calculus-based temporal reasoning," 7th International Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS), 2012.

[9] A. Setzer, R. Gaizauskas, and M. Hepple, "The role of inference in the temporal annotation and analysis of text," Language Resources and Evaluation v. 39, pp. 243–265, 2005.

[10] M. Shanahan, "The event calculus explained," in Artificial Intelligence Today, ed. M. Wooldridge and M. Veloso, Springer Lecture Notes in Artificial Intelligence no. 1600, pp.409-430, 1999.

[11] X. Tannier and P Muller, "Evaluation metrics for automatic temporal annotation of texts," in Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), 2008.