CHRONOS: Improving the Performance of Qualitative Temporal Reasoning in OWL

Eleftherios Anagnostopoulos, Euripides G.M. Petrakis, Sotirios Batsakis
School. of Electronic and Computer Engineering
Technical University of Crete (TUC)
Chania, Crete, Greece, GR-73100
Email: eanagnostopoulos@hotmail.com,{petrakis, batsakis}@intelligence.tuc.gr

***Abstract*—We investigate on potential improvements to reasoning about qualitative temporal information in OWL. Building upon path consistency, the new reasoner design, referred to as CHRONOS, computes a minimal set of relation compositions at run-time (based on a tractable subset of Allen relations) while being sound and complete. The experimental results demonstrate that the two implementation variants of CHRONOS discussed in this work run up to two orders of magnitude faster than SOWL, a temporal reasoner implemented in SWRL.**

***Keywords*-Temporal Ontology; Qualitative Temporal Reasoning; Performance;**

I. Introduction

Semantic Web has motivated research on tools and methodologies capable of extracting, processing and representing the meaning of Web pages as well as for reasoning and querying using this content. Ontologies, in particular, offer the means for representing high level concepts, their properties and their inter-relationships by means of binary relations between concepts or, between concepts and a numerical (concrete) domain [1]. Dynamic ontologies will in addition enable representation of information evolving time. Representation of dynamic features calls for mechanisms allowing for the representation of the notions of time (and of properties varying in time) within an ontology. Methods for achieving this goal include (among others) N-ary relations [2] and the 4D-fluents (perdurantist) approach [3].

Typically, temporal information is expressed quantitatively (i.e., using time instants or temporal intervals whose left and right endpoints are defined numerically). However, temporal information can also be expressed using qualitative (in addition to quantitative) temporal expressions of time instants or intervals by means of their relations to other instants or intervals, when precise quantitative information is unknown (e.g., "event A took place *before* 2014" or "event A took place *after* event *B*"). Temporal knowledge is expressed using a vocabulary of qualitative relationships such as the Allen's thirteen interval relations [4]. The motivation for using a qualitative approach is that it is closer to the way humans represent and reason about commonsense knowledge and that it is possible to deal with incomplete knowledge.

Reasoning is the process of making logical inferences from a given set of facts or statements. Qualitative reasoning, in particular, is the process of making inferences over symbolic (qualitative) information without making numerical computations. Although qualitative (Allen) relations can be defined in an ontology (as object properties) it is not always possible to directly encode the semantics of these relations using the expressivity of OWL and Description Logics (DL) that OWL is based on. There might be inconsistencies within a set of temporal relations that will not be detected by an ordinary OWL reasoner (e.g., Pellet) or, an OWL reasoner might not compute all temporal inferences. CHRONOS addresses both these problems. It extends the OWL-DL reasoning features of Pellet with qualitative temporal reasoning capabilities.

Representing temporal information in ontologies resorts to OWL. However, the syntactic restriction of OWL to binary relations complicates the representation of temporal properties: A property holding for a specific time instant or interval is in fact a ternary relation involving three objects (an object, a subject and a time instant or interval) which cannot be represented directly by a single OWL statement. A standard fix to this problem is to represent the temporal relation by a set of binary ones. Approaches for dealing with ternary relations include (among others) N-ary relations [2] and the 4D-fluents (perdurantist) approach [3].

In our previous contributions [5], [6], both the N-ary relations and the 4D-fluents approaches have been enhanced with qualitative (in addition to quantitative) temporal expressions allowing for the representation of temporal intervals with unknown starting and ending points by means of their relation (e.g., "before", "after") to other time intervals. Reasoning is realized by means of SWRL rules implementing the allowable compositions over the supported (tractable) relation set combined with OWL 2.0 constructs (e.g., for declaring disjoint properties) ensuring path consistency while being sound and complete [7]. Reasoning is embedded within the ontology and can be executed by a general purpose reasoner such as Pellet[1].

[1] http://clarkparsia.com/pellet/

Although fast and intuitive, this approach still suffers from the following disadvantages: (a) Relying on a general purpose reasoner, it is not possible to accommodate any optimization within the implementation of path consistency (resorting entirely to the performance of the SWRL interpreter in use) and, (b) Implementing path consistency in SWRL calls for additional temporal relations thus complicating the representation. CHRONOS [8] tackles both these problems.

Earlier work on qualitative temporal reasoning includes also GQR [9]. GQR is general purpose reasoner for general spatial and temporal calculi and information expressed in XML. Compared to GQR, CHRONOS is a dedicated temporal reasoner for qualitative temporal information expressed in OWL DL. Taking advantage of OWL semantics, temporal reasoning in separated from standard OWL DL reasoning which is handled by Pellet. CHRONOS is independent of temporal model representing the evolution of concepts in time (i.e., works with both the N-ary and the 4D-fluents representations), a feature that cannot be easily handled by GQR. It builds-upon path consistency [7] and is implemented in Java.

Along these lines, the present work suggests an optimization to CHRONOS based on the observation that reasoning can be computed using a minimal subset of the set of possible compositions of basic interval relations. This reasoner variant of CHRONOS is faster than its original implementation and also, faster than SOWL [6], a temporal reasoner implemented in SWRL, and scales-up well for large data sets.

II. Background and Related Work

The OWL-Time temporal ontology[2] describes the temporal content of Web pages and the temporal properties of Web services. Apart from language constructs for the representation of time in ontologies, there is still a need for mechanisms for the representation of the evolution of concepts (e.g., events) in time. Temporal relations are in fact ternary (i.e., properties of objects that change in time involve also a temporal value in addition to the object and the subject) and cannot be expressed directly in OWL. Representing temporal relations in OWL calls for specific methods such as 4D-fluents [3] or the N-ary relations [2] with the later being the one implemented in this work.

The N-ary relations approach suggests representing an non-binary relation as two properties each related with a new object, which in turn is related with the temporal interval over which this relation holds. As shown in Fig.1, to add the time dimension to an ontology, classes *Event* and *TimeInterval* are introduced. Class *Event* represents temporal relations (it can be specialized into sub-classes such as *EmploymentEvent* and class *TimeInterval*) is the domain class of time intervals. Properties having a temporal dimension are called *fluent properties* and connect initial objects with instances of class *Event*.

A temporal property between two individuals (e.g., Employee works for Company) holds as long as that event endures. The N-ary property is represented as a class rather than as a property. Instances of such classes correspond to instances of the relation. Additional properties introduce additional binary links to each argument of the relation. For properties that change in time, their domains and ranges have to be adjusted taking into account the classes of intermediate objects representing the relation. For example, the *"worksFor"* relation in Fig. 1 is no longer a relation having as object an individual of class "Company" and subject of class "Employee" as they are now related to the new object *"EmploymentEvent"*. The new domain is the union of the old domain with the class that represents the N-ary property (Event class). Likewise, the new range is a union of the old one with the Event class.

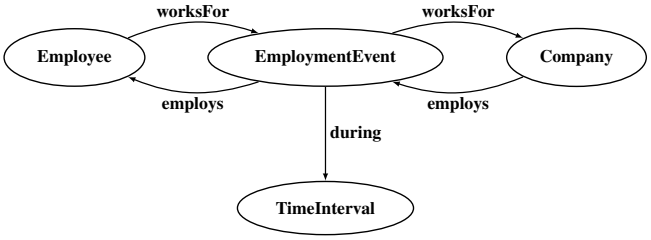

Figure 1: N-ary relations example

SOWL [5], [6] is an ontology for representing and reasoning over spatio-temporal information in OWL. Representing qualitative temporal information (in addition to quantitative information) is a distinctive feature of SOWL. In SOWL the temporal representation was enhanced with qualitative temporal relations (i.e., relations holding between time intervals whose starting and ending points are not specified) by introducing temporal relationships as object relations between time intervals. A temporal relation can be one of the thirteen interval relations of Allen [4] of Fig. 2. Definitions for temporal entities (e.g., intervals) are provided by incorporating OWL-Time into the same ontology.

Temporal reasoning in SOWL is realized by introducing a set of SWRL rules for asserting inferred temporal relations of Allen. Reasoners that support DL-safe rules (i.e., rules that apply only on named individuals in the knowledge base) such as Pellet can be used for inference and consistency checking over temporal relations.

Relations between intervals are expressed as relations between their starting and ending points which in turn are expressed as a function of the three possible relations between points (time instants) namely *equals*, *before* and *after* denoted by "=", "<" and ">" respectively, forming the so called *"point algebra"* [7]. The relations *after, metby, overlappedby, startedby, contains* and *finishedby* are the inverse of *before, meets, overlaps, starts, during* and *finishes*

[2]http://www.w3.org/TR/owl-time/

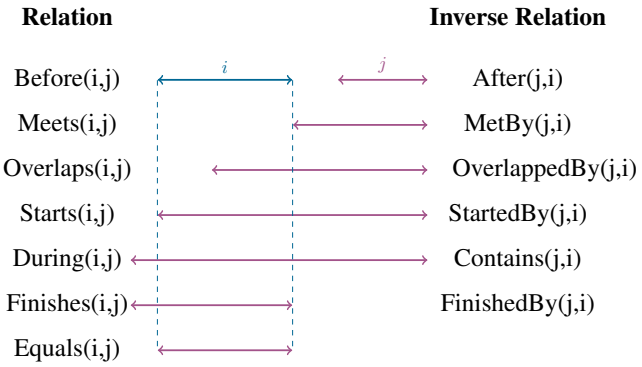

Figure 2: Temporal Relations of Allen

and are defined accordingly (by interchanging s_1, s_2 and e_1, e_2 in their respective definitions). Basic properties are that the relation *equals* is symmetric and relations *before* and *after* are transitive. These temporal relations and the corresponding reasoning mechanism are integrated both in the 4D-fluents and the N-ary based ontologies. Qualitative relations can be asserted as well, i.e., when the specific time points or the temporal extends of intervals are unknown but their relative position is known. Whenever a temporal relation between two events is uncertain, disjunctions of some of the thirteen basic relations can be used to represent what is known.

III. CHRONOS

CHRONOS is a system for reasoning over temporal information in OWL ontologies. Following the example of Pellet-Spatial [10] for qualitative spatial information, CHRONOS is a dedicated temporal reasoning engine implemented in Java which allows one to incorporate certain optimizations in implementing path consistency [7]. It supports checking of consistency and computation of new temporal inferences resulting from a set of asserted relations. CHRONOS defines an RDF/OWL vocabulary for expressing qualitative temporal relations between time intervals, using the Temporal Relation Model of Allen.

As illustrated in Fig. 3, the temporal relations of Allen are defined as simple object properties with no extra characteristics (e.g., *inverse, transitive*). One can either use the vocabulary provided, or use his own by defining sub-property axioms (e.g., *ObjectProperty :* B *subPropertyOf :* $Before$). Non-temporal relations are represented as ordinary OWL assertions. An interval is represented as an OWL individual (e.g., *Individual :* $interval_1$ *oftype Class :* $Interval$) and a temporal relation between two intervals is represented as an OWL object property assertion (e.g., *Individual :* $interval_1$ *ObjectProperty :* $Before$ *Individual :* $interval_2$).

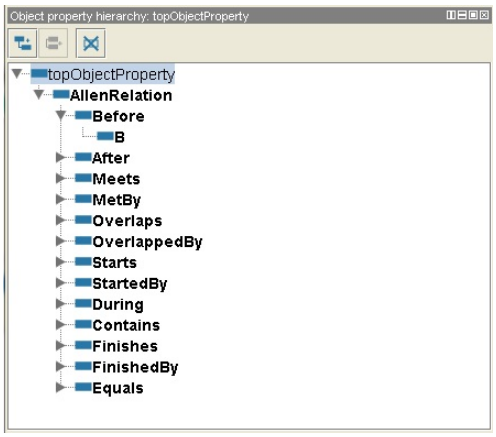

Figure 3: Temporal Allen Relations as object properties, in Protégé.

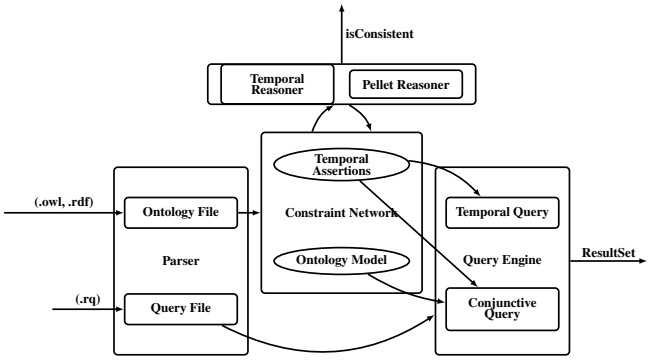

Figure 4: CHRONOS architecture

A. Architecture

The architecture of CHRONOS is illustrated in Fig. 4. The *Parser* for loading and processing ontologies in main memory. It implements an *Ontology Parser* and a *Query Parser* for handling of ontologies and queries respectively. The Ontology Parser separates temporal relations from non-temporal OWL triples. All temporal triples are removed from the RDF graph and are stored in a *constraint network*. The rest of the graph, contains only non-spatial OWL assertions and are handled by Pellet reasoner (i.e., are stored in the Pellet's knowledge base). The Query Parser implements an ARQ[3] parser enabling CHRONOS to process queries of temporal information, as well as conjunctive temporal and non-temporal semantic queries.

A *Constraint Network (CN)* is a set of variables together with a set of constraints defined on these variables. The constraint network of CHRONOS comprises of the temporal OWL triples in the ontology graph (extracted by the Ontology Parser), as well as of non-temporal OWL assertions, which are stored in Pellet's knowledge base, as it is typical in standard (i.e., non-temporal) reasoning. CHRONOS separates temporal from semantic DL reasoning, which is managed by Pellet, and uses an exclusive reasoner for temporal calculus.

[3]http://jena.apache.org/documentation/query/

B. Reasoning

Temporal reasoning (i.e., inferring implied relations and detecting inconsistencies) can be viewed as a form of a constraint satisfaction problem which is NP hard in general case. CHRONOS handles tractable cases of such problems by limiting asserted temporal relations to the basic Allen relations [11]. Temporal reasoning is then achieved by applying the path consistency algorithm [7]. Path consistency computes all inferred relations using compositions of existing relations defined, until a fixed point is reached (i.e., algorithm does not yield new inferences) or until an inconsistency is detected (i.e., yield the empty relation $\emptyset$ as a result). Path consistency, when applied on a set of assertions containing only basic relations, retains tractability and guarantees soundness and completeness of reasoning [11].

The possible compositions of basic Allen temporal relations are defined in the composition table [4]. The composition table represents the result of the composition of two temporal relations. For example, if relation R_1 holds between $interval_x$ and $interval_y$ and relation R_2 holds between $interval_y$ and $interval_z$, then the entry of the composition table corresponding to row R_1 and column R_2 denotes the possible relation(s) holding between $interval_x$ and $interval_z$:

$$R_1(x,y) \circ R_2(y,z) \rightarrow R_3(x,z)$$

The following is an example of such a temporal inference rule:

$$before(x,y) \wedge equals(y,z) \rightarrow before(x,z)$$

A series of compositions of relations may imply relations which are inconsistent with existing ones (for example the rule referred to above will yield a contradiction if *after(x,z)* has been asserted into the ontology for specific values of *x,y,z*). Consistency checking is achieved by ensuring path consistency [7]. Path consistency is implemented by consecutively applying the following formula:

$$\forall x,y,k\ R_s(x,y) \leftarrow R_i(x,y) \cap (R_j(x,k) \circ R_k(k,y))$$

representing intersection of compositions of relations with existing relations (symbol $\cap$ denotes intersection, symbol $\circ$ denotes composition and R_i, R_j, R_k, R_s denote temporal relations). The formula is applied until a fixed point is reached (i.e., the consecutive application of the rules above doesn't yield new inferences) or until the empty set is reached, implying that the ontology is inconsistent.

1) Handling Disjunctions of Basic Relations: The composition of basic relations may infer disjunctions of such relations because disjunctive entries exist in the composition table of Allen relations. For example, the composition of *overlaps* and *during* yields disjunction of three possible relations (*starts, overlaps* and *during*) as a result:

$$Overlaps(x,y) \circ During(y,z) \rightarrow$$
$$\{Overlaps \cup Starts \cup During\}(x,z)$$

Disjunctions of relations are represented using new relations, whose compositions must also be defined and asserted into the knowledge base. Composing disjunctions of relations can be achieved in two different ways: a) By pre-computing the composition of all disjunctions and storing the result in a structure, which contains all possible compositions between basic and disjunctive relations. However, this structure may hold up to $2^{13} \times 2^{13}$ entries (i.e., up to 2^{13} disjunctions can appear). b) By decomposing disjunctive relations into basic ones and computing each composition "on the fly" (i.e., at run-time). In most cases, the number of compositions are much less resulting in faster reasoning times.

Table I: Comparison between CHRONOS$_{x13}$ and CHRONOS$_{x28}$

	CHRONOS$_{x13}$	**CHRONOS**$_{x28}$
Model	13 Temporal Allen relations, 2^{13} possible disjunctions.	
Compositions of Disjunctions	Disjunctive relations are decomposed into basic ones and computation of compositions involves a simple look-up operation in the 13×13 composition table of Allen relations. In practice, results in less than $2^{13} \times 2^{13}$ combinations of disjunctions of basic relations.	A tractable set of 28 basic and disjunctive relations is used. Their 28×28 compositions are encoded as functions and are used at run-time saving memory and computation time.
Memory Structure	A composition table with 13×13 entries, the compositions of the basic interval relations. Compositions are not stored but are computed at run-time, using simple look-up operations in the 13×13 table.	CHRONOS$_{x28}$ compositions are not stored but are encoded as functions.
Inverse of Disjunctions:	CHRONOS$_{x13}$ computes the inverse of disjunctive relations at run-time, using the inverse of the basic relations for computing compositions.	Based on the tractable set of CHRONOS$_{x28}$, the possible 28 inverse relations are encoded as functions.

CHRONOS$_{x13}$: The first implementation of CHRONOS [12] follows the latter approach. Computing compositions on the fly results in less than $2^{13} \times 2^{13}$ combinations of disjunctions of basic relations. Disjunctive relations are decomposed into basic ones (which are computed at run-time) and each composition involves a simple look-up operation in the 13×13 composition table of Allen relations. CHRONOS suggests reducing the number of basic relations from 12 (the size of possible compositions is then 12×12 in which case up to 2^{12} disjunctions can appear), by replacing the directional relation *Equals* with OWL axiom *sameAs*.

For example, composition of {*overlaps, starts, during*} and *during* computes as

$$
\begin{gathered}
(Overlaps \cup Starts \cup During) \circ During \\
\rightarrow (Overlaps \circ During) \cup (Starts \circ During) \cup \\
(During \circ During) \\
\rightarrow (Overlaps \cup Starts \cup During) \cup (During) \cup (During) \\
\rightarrow Overlaps \cup Starts \cup During \cup During \cup During \\
\rightarrow Overlaps \cup Starts \cup During
\end{gathered}
$$

CHRONOS$_{x28}$: The second implementation variant of CHRONOS which is proposed here relies on the observation that reasoning over Allen relations can be based on a tractable set of 28 relations which are identified in [8]: Disjunctions of the 13 basic relations can generate the following additional 15 disjunctions of basic relations

$$
\begin{gathered}
\{Before\}, \{After\}, \{Meets\}, \{MetBy\}, \{Overlaps\}, \\
\{OverlappedBy\}, \{Starts\}, \{StartedBy\}, \{During\}, \\
\{Contains\}, \{Finishes\}, \{FinishedBy\}, \{Equals\}, \\
\{Finishes, FinishedBy, Equals\}, \\
\{Overlaps, Starts, During\}, \{Starts, StartedBy, Equals\}, \\
\{OverlappedBy, During, Finishes\}, \\
\{OverlappedBy, StartedBy, Contains\}, \\
\{Before, Meets, Overlaps\}, \\
\{Overlaps, Contains, FinishedBy\}, \\
\{After, MetBy, OverlappedBy\}, \\
\{Before, Meets, Overlaps, Starts, During\}, \\
\{After, MetBy, OverlappedBy, During, Finishes\}, \\
\{After, MetBy, OverlappedBy, StartedBy, Contains\}, \\
\{Before, Meets, Overlaps, Contains, FinishedBy\}, \\
\{Overlaps, OverlappedBy, Starts, StartedBy, During, \\
Contains, Finishes, FinishedBy, Equals\}, \\
\{Before, Meets, Overlaps, OverlappedBy, Starts, \\
StartedBy, During, Contains, Finishes, \\
FinishedBy, Equals\}, \\
\{After, MetBy, Overlaps, OverlappedBy, Starts, \\
StartedBy, During, Contains, Finishes, \\
FinishedBy, Equals\}
\end{gathered}
$$

Composing pairs of these relations yields $28 \times 28 = 784$ possible results. CHRONOS$_{28}$ does not implement any static structure for storing these results. Instead, all 784 compositions are implemented as functions and are used at run-time by path-consistency for generating the new compositions. Compared to CHRONOS$_{13}$ larger chunks of information (i.e., disjunctions of compositions) are encoded as functions and used at run-time, saving computation time. Notice that, path consistency [7] uses *Inverses of Disjunctions* of basic relations. The inverse of a disjunctive relation, is the disjunction of the inverse of the same relations. For example, the inverse of the disjunctive relation {*before, meets*} is the disjunction of their inverse relations {*after, metby*} respectively. The possible twenty-eight inverse relations have also been pre-computed and are used at run-time, as functions. Table I presents a brief comparison between CHRONOS$_{x13}$ and CHRONOS$_{x28}$. As it is noticed in [7], path consistency is tractable for basic temporal Allen relations.

C. Query Engine

Both implementations of CHRONOS can answer conjunctive queries specifying temporal and non-temporal patterns (i.e., triple patterns for temporal relations joined with triple patterns for semantic RDF/OWL relations). More specifically, this module can process queries written in SPARQL satisfying the following conditions: Each property used in the predicate position is either an (object or datatype) property defined in the ontology or one of the following built-in properties: {*rdf : type, owl : sameIndividualAs, owl : differentFrom*} and, At least one of the triples must contain a temporal object property in the predicate position (otherwise it is a non-temporal query).

The query engine of CHRONOS can process temporal queries specifying temporal properties in an RDF graph or Allen relations between temporal intervals, by implementing a dual stage query answering technique: The set of query answers returned by the first stage in response to a given a temporal query, are fed to the second stage whose purpose is to guarantee that the non-temporal part of the query is satisfied as well.

IV. Evaluation

In the following, the efficiency of our reasoning approach is assessed both theoretically and experimentally. We carried-out several groups of experiments using synthetic (but realistic) data set, demonstrating the run-time efficiency

of our reasoning engine with SOWL [5], [6], a temporal reasoner implement in SWRL.

If only the basic interval relations are supported reasoning has polynomial time complexity: Within n intervals, any time interval can be related to every other interval with one basic Allen relation (basic Allen relations are mutually exclusive), at most $O(n^2)$ relations can be inferred. In the worst case, n^2 relations are asserted into the knowledge base from n individuals and path consistency has $O(n^3)$ time complexity. This upper bound is pessimistic, since an inconsistency detection may terminate the reasoning process early, or the asserted relations may yield a small number of inferences. In the average case, the number of temporal relations is in the order of n and path consistency has $O(n)$ time complexity for both implementations of CHRONOS.

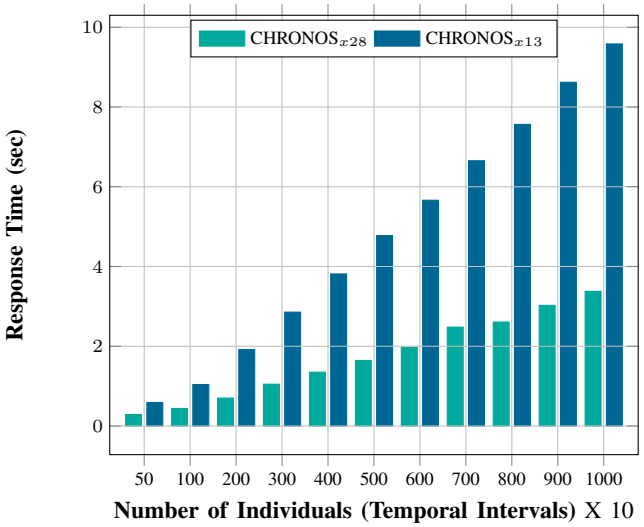

Figure 5: Average case performance of CHRONOS$_{x28}$ and CHRONOS$_{x13}$

We carried out two different experiments, corresponding to measurements of performance in the average and in the worst case. The average case performance is encountered when less than n^2 relations are inferred from an input set of n individuals (i.e., time intervals or time instants). In our experiments, kn relations ($k = 13$) are asserted. This is for example the case of a random set of individuals. The worst case performance is encountered when the number of asserted relations are in the order of n^2. This is for example the case of intervals given in a certain arrangement (i.e., each one is *before* another). The data sets are simple ontologies containing between 500 and 10,000 random intervals. Every individual can be related with only one individual. All running times reported are averages over 10 ontologies. In all experiments we compare the running time of all reasoner implementations as a function of the number of individuals. All experiments were run on a PC with 4 GB RAM running Windows 7 at 1.90 GHz.

Fig. 5 illustrates reasoning times of the CHRONOS implementations, in average case as a function of the number of individuals (i.e., temporal intervals) in the input data set. Both implementations exhibit near linear time performance with CHRONOS$_{x28}$ performing almost three times faster than CHRONOS$_{x13}$.

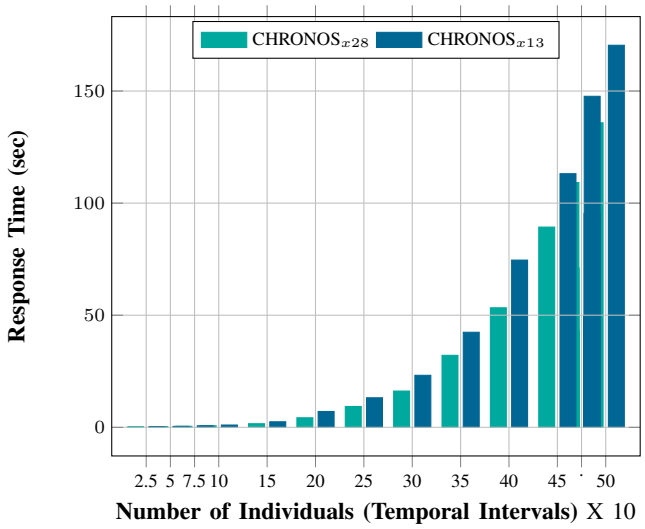

Figure 6: Worst case performance of CHRONOS$_{x28}$ and CHRONOS$_{x13}$

Fig. 6 illustrates reasoning times of CHRONOS$_{x13}$ and CHRONOS$_{x28}$ in the worst case. The experimental results confirm the theoretic polynomial time complexity of reasoning exhibiting a close to n^3 performances. CHRONOS$_{28}$ is again the better variant.

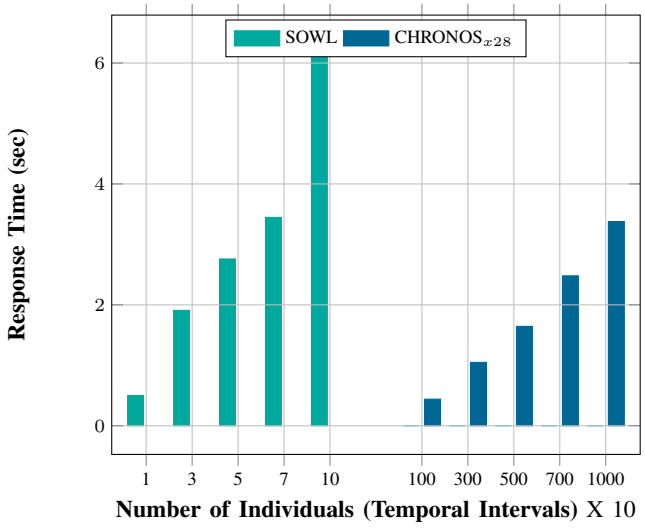

Figure 7: Average case performance of CHRONOS$_{x28}$ and SOWL

The purpose of the following set of experiments is to demonstrate the superiority of two implementation variants of CHRONOS over SOWL [5], a temporal reasoner implemented in SWRL, in both the average and worst cases, as a function of the number of temporal intervals in the ontology.

Both implementations of CHRONOS clearly outperform

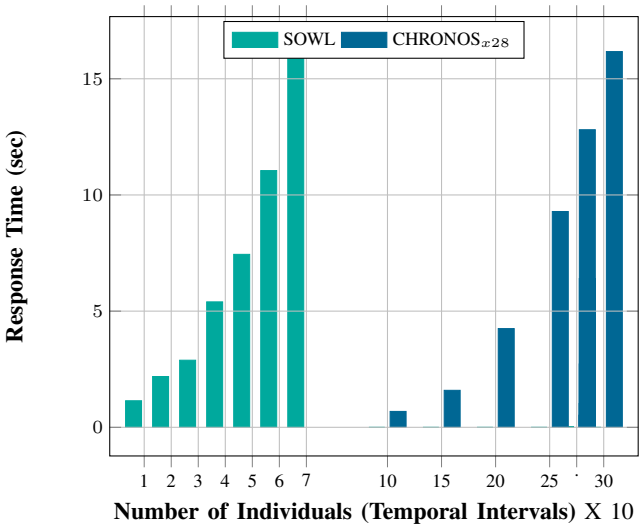

Figure 8: Worst case performance of CHRONOS$_{x28}$ and SOWL

their SWRL competitor in all cases. The run-time performance of SOWL declines drastically for large data sets as the large number of inferred relations caused the memory to overflow. Although the SOWL reasoner may perform better on computers with more memory, CHRONOS scales-up much better than SOWL with the size of the input (i.e., the performance gap between the two reasoner implementations increases with the size of the data set) and will run much faster than SOWL for large data sets even on average computers.

V. Conclusions and Future Work

We propose CHRONOS, a stand-alone temporal OWL reasoner following the example of Pellet-Spatial [10] (for spatial information) for reasoning over temporal knowledge in ontologies using OWL. CHRONOS implements certain optimizations and exhibited increased performance improvements over SOWL in both the average and the worst cases. Extending CHRONOS to handle both temporal instants in addition to temporal intervals (as SOWL does), investigating on more effective reasoner designs for temporal information and examining their performance on real applications are issues for future research.

Acknowledgements

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement no 604691 (project FI-STAR).

References

[1] A. Artale and E. Franconi, "A survey of temporal extensions of description logics." *Ann. Math. Artif. Intell.*, vol. 30, no. 1-4, pp. 171–210, 2000. [Online]. Available: http://dblp.uni-trier.de/db/journals/amai/amai30.html#ArtaleF00

[2] N. Noy and A. Rector, "Defining N-ary Relations on the Semantic Web," April 2006. [Online]. Available: http://www.w3.org/TR/swbp-n-aryRelations/

[3] C. A. Welty and R. Fikes, "A Reusable Ontology for Fluents in OWL." in *FOIS*, ser. Frontiers in Artificial Intelligence and Applications, B. Bennett and C. Fellbaum, Eds., vol. 150. IOS Press, 2006, pp. 226–236. [Online]. Available: http://dblp.uni-trier.de/db/conf/fois/fois2006.html#WeltyF06

[4] J. F. Allen, "Maintaining Knowledge About Temporal Intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[5] S. Batsakis and E. G. M. Petrakis, "SOWL: A Framework for Handling Spatio-temporal Information in OWL 2.0," in *RuleML Europe*, ser. Lecture Notes in Computer Science, N. Bassiliades, G. Governatori, and A. Paschke, Eds., vol. 6826. Springer, 2011, pp. 242–249. [Online]. Available: http://dblp.uni-trier.de/db/conf/ruleml/ruleml2011e.html#BatsakisP11

[6] S. Batsakis, K. Stravoskoufos, and E. G. M. Petrakis, "Temporal Reasoning for Supporting Temporal Queries in OWL," in *KES (1)*, ser. Lecture Notes in Computer Science, A. Kónig, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, Eds., vol. 6881. Springer, 2011, pp. 558–567. [Online]. Available: http://dblp.uni-trier.de/db/conf/kes/kes2011-1.html#BatsakisSP11

[7] P. van Beek and R. Cohen, "Exact and Approximate Reasoning about Temporal Relations," *Computational Intelligence*, vol. 6, pp. 132–144, 1990.

[8] E. Anagnostopoulos, "CHRONOS: A Reasoning Engine for Qualitative Temporal Information in OWL," Chania, Crete, Greece, November 2013, diploma Thesis. [Online]. Available: http://www.intelligence.tuc.gr/publications.php?pub_author=261&pub_type=All&pub_subject=All

[9] M. Westphal, S. Wlfl, and Z. Gantner, "GQR: A Fast Solver for Binary Qualitative Constraint Networks," in *AAAI Spring Symposium: Benchmarking of Qualitative Spatial and Temporal Reasoning Systems*. AAAI, 2009, pp. 51–52. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaaiss/aaaiss2009-2.html#WestphalWG09

[10] M. Stocker and E. Sirin, "PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine," in *6th Intern. Workshop on OWL: Experiences and Directions (OWLED 2009)*. Springer-Verlag New York, Inc., Aug. 2009, pp. 2–31. [Online]. Available: http://www.webont.org/owled/2009/papers/owled2009_submission_20.pdf

[11] B. Nebel and H. Burckert, "Reasoning About Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra," *Journal of the ACM (JACM)*, vol. 42, no. 1, pp. 43–66, 1995.

[12] E. Anagnostopoulos, S. Batsakis, and E. G. M. Petrakis, "Chronos: A reasoning engine for qualitative temporal information in owl." in *KES*, ser. Procedia Computer Science, J. Watada, L. C. Jain, R. J. Howlett, N. Mukai, and K. Asakura, Eds., vol. 22. Elsevier, 2013, pp. 70–77. [Online]. Available: http://dblp.uni-trier.de/db/conf/kes/kes2013.html#AnagnostopoulosBP13