titative temporal information [7]. Moreover, we are also applying LaTeR to model-based diagnosis of dynamic systems. In both cases, LaTeR high-level language provide a useful interface for obtaining a loosely coupled integration, and LaTeR's efficient treatment of queries (and updates) provides crucial advantages. A discussion on such applications can be found in [5].

A prototype of LaTeR has been implemented in C on Sun workstations, under the UNIX operating system.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[2] J. Allen. Time and time again: the many ways to represent time. *Int. J. Intelligent Systems*, 6(4):341–355, 1991.

[3] R. Arthur and J. Stillman. Temporal reasoning for planning and scheduling. Technical report, AI Lab, General Elettric Research Center, 1992.

[4] V. Brusoni, L. Console, B. Pernici, and P. Terenziani. LaTeR: a general purpose manager of temporal information. In *Methodologies for Intelligent Systems 8*, pages 255–264. Lecture Notes in Computer Science 869, Springer Verlag, 1994.

[5] V. Brusoni, L. Console, B. Pernici, and P. Terenziani. Dealing with time in knowledge based systems: a loosely coupled approach. In *Proc. FLAIRS '95*, Melbourne, FL, 1995.

[6] V. Brusoni, L. Console, and P. Terenziani. On the computational complexity of querying bounds on differences constraints. *Artificial Intelligence (to appear)*, 1995.

[7] L. Console, B. Pernici, and P. Terenziani. Towards the development of a general temporal manager for temporal databases: a layered and modular approach. In *Proc. of the Int. Work. on an Infrastructure for Temporal Databases*, Arlington, Texas, 1993.

[8] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.

[9] T. Dean and D. McDermott. Temporal data base management. *Artificial Intelligence*, 32:1–56, 1987.

[10] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[11] A. Gerevini and L. Schubert. Efficient temporal reasoning through timegraphs. In *Proc. 13th IJCAI*, pages 648–654, Chambery, 1993.

[12] H. Kautz and P. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proc. AAAI 91*, pages 241–246, 1991.

[13] J. Koomen. The TIMELOGIC temporal reasoning system. Technical Report 231, Computer Science Department, University of Rochester, Rochester, NY, March 1989.

[14] L. McKenzie and R. Snodgrass. Evaluation of relational algebras incorporating the time dimension in databases. *ACM Computing Surveys*, 23(4):501–543, 1991.

[15] I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In *Proc. AAAI 91*, pages 260–267, 1991.

[16] R. Snodgrass, editor. *Proc. of the Int. Work. on an infrastructure for Temporal Databases*. 1993.

[17] A. Tansell, R. Snodgrass, J. Clifford, S. Gadia, and A. Segev. *Temporal Databases: Theory, design and implementation*. Benjamin Cummings, 1993.

[18] P. VanBeek. Approximation algorithms for temporal reasoning. In *Proc. 11th IJCAI*, pages 1291–1297, 1989.

[19] P. VanBeek. Temporal query processing with indefinite information. *Artificial Intelligence in Medicine*, 3:325–339, 1991.

[20] M. Vilain. A system for reasoning about time. In *Proc. AAAI 82*, pages 197–201, 1982.

[21] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. AAAI 86*, pages 377–382, 1986.

[22] M. Vilain, H. Kautz, and P. VanBeek. Constraint propagation algorithms for temporal reasoning: a revised report. In D.S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about physical systems*, pages 373–381. Morgan Kaufmann, 1989.

[23] Ed Yampratoom and J. Allen. Performance of temporal reasoning systems. *SIGART Bulletin*, pages 26–29, 1993.

long less than 100 and more than 75% for sequences long from 100 to around 200.

A more detailed evaluation of the results can be found in [6].

5 Comparisons with Related Work

Different criteria can be considered in order to compare the temporal managers developed in the artificial intelligence literature. A first important criteria concerns completeness. In LaTeR, as in many artificial intelligence approaches, we choose to retain completeness. since it seems important to us in order to provide users and applications with uncontestable and reliable results. This rises a trade-off between expressive power and computational complexity of complete temporal reasoning. As e.g. in Timegraph [11] and in Tachyon [3] we chose to limit the expressive power in order to retain tractability. In particular, the expressive power of LaTeR is comparable to that of Tachyon, which deals with temporal constraints that can be mapped onto conjunctions of bounds on differences, too.

In [23], Allen distinguishes between two different class of temporal managers:

(i) managers that use a constraint satisfaction technique at assertion time, building an all-to-all graph with the constraints between each pair of temporal entities in the knowledge base;

(ii) managers that build partial graph structures, which need further processing at query time.

For instance, Allen classified TimeLogic [13], MATS [12] and Tachyon [3] as systems of the first type, and Timegraph [11] and TMM [9] as systems of the second type. In [23], Allen, considering only atomic queries (i.e., queries for extracting the constraints between two entities in the graph, or yes/no queries without conjunction) pointed out that the approaches computing the all-to-all graph are more efficient than those computing only partial graphs when dealing with queries. In fact, in these approaches, queries can be answered in constant time, by reading the values from the graph, while in the approaches in (ii) some further reasoning may be needed. On the other hand, the approaches in (i) are less efficient when dealing with assertions (updates), since the whole all-to-all graph has to be computed after each assertion.

LaTeR is a system computing the all-to-all graph (which is the minimal network in the case of LaTeR) that reconciles the advantages of both types of approaches, thanks to its efficient treatment of complex queries and of assertions as hypothetical queries. This result has been obtained via the treatment of complex types of queries.

As shown in van Beek's work [19], as soon as one considers non-atomic queries (even only conjunctions of yes/no queries), two problems arise: on the one hand, the distinction between queries about necessity and queries about consistency is needed; on the other hand, constraint propagation may be required. Van Beek's work has two major limitations with respect to the work in this paper: (i) it deals with qualitative information only and (ii) it performs constraint propagation on the whole network (global propagation) both for queries about necessity and queries about consistency (notice, however, that Van Beek allows the use of all logical connectives in the query language, although answering queries becomes exponential). On the other hand, we showed that propagation is needed only for queries about consistency (and hypothetical queries, which are not considered in [19]) and, even in such a case, local propagation is sufficient.

Thus, LaTeR retains the efficient query processing typical of approaches computing the all-to-all graph also in case complex queries. Furthermore, since in LaTeR assertions followed by queries can be simulated by hypothetical queries (which are answered by local temporal reasoning), LaTeR does not have to recompute the whole all-to-all graph at each assertion, so that also assertions are managed efficiently.

Besides providing the computational advantages above, LaTeR treatment of different (and complex) types of queries seems to us a main feature of the system in itself, since queries (and assertions) constitute the main way of interacting with temporal managers. Thus, we believe that the expressive query language (and manipulation language) constitutes an advantage of LaTeR with respect to the other systems in the literature. For instance, high-level interface languages are widely used in the temporal databases community. However, most of the approaches to temporal databases only deal with time stamps associated with information and do not consider temporal relations between entities (see, e.g., [14]), so that temporal constraint propagation is not needed.

6 Conclusions

In the paper we showed how queries on an heterogeneous temporal knowledge base can be answered efficiently, independently of the dimension of the knowledge base

Currently, LaTeR is being loosely coupled with Oracle, in order to extend relational databases to deal also with (possibly imprecise) qualitative and quan-

the consistency of the knowledge base and thus consistency must be checked after each update and before answering the queries following the update itself.

Since answering queries in an inconsistent knowledge base is meaningless, the consistency check must be performed anyway. Moreover, the minimal network of the updated knowledge base can be produced by the same algorithms that check consistency. This means that the presence of updates does not affect the efficiency of our approach: consistency has to be checked anyway but this produces the minimal network and queries can be answered efficiently given the minimal network (see the previous section).

Our approach, on the other hand, suggests an efficient way for dealing with a class of updates, specifically updates that add new constraints (which are the most common in many applications, see the discussion in [5]). In fact, in such a case one can answer the queries following an update as hypothetical ones. More specifically, a query Q following an update U can be answered as the hypothetical query:

$$Q \ if \ U$$

which only involves local propagation. If a query Q follows a sequence of updates $U_1, \ldots, U_h$, this can be simulated as the query $Q \ if \ U_1, \ldots U_h$.

The advantage of such an approach is that during a session of interleaved queries and updates all the operations can be performed with local propagation and the actual update of the knowledge base (which can be very costly) can be delayed with respect to the query process (e.g., performed once and off-line at the end of the session).

Dealing with updates as hypothetical queries can provide significant computational advantages. However, when the sequence of updates and queries becomes very long and the updates involve significant parts of the knowledge base, such advantages may be lost. A detailed evaluation of the such computational advantages and trade-offs can be found in [6] where we compare:

- the case where the minimal network is recomputed after each update (and then queries are answered with local propagation as discussed in the previous section);

- the case where queries are dealt with as hypothetical ones.

The evaluation is performed by taking into account three different parameters:

- The length of the sequences ("k" in (7));

- The average dimension of updates/queries (we assume that updates and queries have the same average dimension), i.e., the average ratio between the dimension of queries/updates and the dimension of the knowledge base;

- How extensive the updates are, that is: how many entities involved in the $i-th$ update were not involved in the previous ones. At one extreme, all the updates may involve the same set of variables (i.e., the same part of the knowledge base is repeatedly changed); at the other extreme, each update may involve a part of the knowledge base that was not involved by any previous update and thus the updates in the sequence tend to involve larger and larger parts of the knowledge base as the length of the sequence increases (in general, both extreme cases are unlikely; taking this as a parameter allows us to consider all possibilities).

Two different evaluations are then performed:

- First of all we evaluated the break-even point between the two approaches that is: the maximum length of the sequence for which dealing with updates as hypothetical queries provides advantages, given the average dimension and extension of the updates or, conversely, which is the maximum dimension for the updates for which there are advantages, given the length of the sequence.

  For example, it turned out that for a sequence of 40 updates and queries in which one half of the variables involved in each update was not involved by previous ones (so that the updates tend to extend to significant parts of the knowledge base), dealing with updates as hypothetical queries provides advantages when the average dimension of each update/query is less that 7% of the knowledge base. Conversely, when the average dimension of each query/update is 1% of the knowledge base, the approach is advantageous when the length of the sequence is less than 320.

  From our experience in the practical application of LATER (see [5]), these dimensions are realistic in the sense that it is common that the dimension of updates/queries is around 1% of the dimension of the knowledge base and in any case never more than 5%.

- We evaluated how big the computational advantage is. For example, when the average dimension of update/queries is 1% of the knowledge base (and one half of the variables involved in each update were not involved in previous ones), the advantage is around 90% if the sequence is

Each one of the constraints in (6), taken in isolation, is consistent with (5), but the conjunction in (6) is inconsistent with (5). Thus constraint propagation is needed in order to check whether a set of constraints is consistent with a given knowledge base.

However, we proved that *global propagation* of the constraints in the query to the whole knowledge base is not needed for computing the answer. In fact, since the minimal network is available and since the goal is not to update the whole knowledge base but just to answer the query, *local propagation* is sufficient (local propagation concerns only the variables in the query). More formally, we proved the following theorem (the proof can be found in [6]):

Theorem 1 *Let S be a set of variables, K.B. a set of bounds of differences on such variables and N_S the consistent minimal network computed by the (complete) propagation algorithm. Let us consider a query $MAY(Q)$ on K.B. (where Q is a conjunction of atomic tests) referring to a set $G \subseteq S$ of variables (i.e., all the constraints in the query involve only variables in G).*
Let N'_S be the minimal network obtained by propagating the constraints in Q to N_S (i.e., to all the variables - global propagation in S) and N'_G the minimal network obtained by propagating the constraints in Q to N_G, where N_G is the restriction of N_S to the variables in G (local propagation);
then N'_S is consistent if and only if N'_G is consistent.

The theorem guarantees that in order to answer MAY queries of the form:

$$MAY(C_1 \ AND \ C_2 \ \dots \ AND \ C_n)$$

it is sufficient to propagate the constraints C_i in the query to the part of the minimal network whose nodes are the variables in the query (i.e., occurring in $C_1, C_2, \dots, C_n$).

Therefore conjunctive MAY queries can be answered in a time that is cubic in the number of variables in the query and that is independent of the dimension of the knowledge base.

3.2.3 Hypothetical Queries.

Hypothetical queries are queries of the form

$$Q \ if \ C$$

where Q is a query of one of the types discussed in the previous subsections and C is a conjunction of temporal constraints, expressed in LaTeR's high-level language.

For example, given the knowledge base in figure 1, the following queries could be asked:

HowLong John_work
        If Mary_work Lasting 4h, 50min?
Answer : 5h

MUST(Tom_work During Mary_work)
        If Mary_work Lasting 4h, 50min?
Answer : Yes

In principle, an hypothetical query should be answered in 3 steps: (i) adding the constraints C to the temporal knowledge base; (ii) computing the minimal network N' for the new knowledge base; (iii) answering the query Q given N'.

However, we proved the following theorem (see [6] for more details):

Theorem 2 *Given S, N_S, G, N_G, Q, N'_S and N'_G as in Theorem 1, then for each pair of variables $\langle X, Y \rangle$ in G, the maximal admissibility range for $X - Y$ provided by N'_G (minimal network computed with local propagation) is the same as the maximal admissibility range for $X - Y$ provided by N'_S (minimal network computed with global propagation).*

In other words, as regards the variables in G, local propagation to the part of the minimal network concerning the variables in G produces the same results as global propagation to the whole minimal network. This means that, for any query $Q \ If \ C$, it is sufficient to proceed as follows:

- perform local propagation of the constraints in C to the part of the minimal network involving the variables in $C \cup Q$;

- Answer Q as discussed in the previous subsections.

The theorem guarantees that this procedure provides the same result that would be obtained by propagating the constraints in C to the whole knowledge base before answering the query Q. Thus, also hypothetical queries are answered in LaTeR in a time which is independent of the dimension of the knowledge base (more specifically, in a time that is cubic in the number of variables in $C \cup Q$).

4 Dealing with updates

In the practical applications of temporal reasoning queries are interleaved with updates. In other words, a typical session with a temporal knowledge server could have the form of a sequence:

$$U_1, Q_1, U_2, Q_2, \dots, U_k, Q_k \qquad (7)$$

of alternated updates (U_i) and queries (Q_i). An update corresponds to the the addition or removal of some temporal assertion. Each update may affect

$MAY(\ Tom\_work\ During\ Mary\_work\ AND$
$\qquad start(John\_work)\ After\ 16^{20})$

which involves checking that the conjunction of the two assertions is consistent with the knowledge base. Given the example in figure 1 the answer to such a query is negative.

Queries about consistency/necessity are mapped into conjunctions of *atomic tests* each one of which is a check on the distance between two time points (and thus the difference between two variables in the minimal network). The mapping is the same used for translating assertions into bounds on differences sketched in section 2. For example, the conjunction of atomic tests corresponding to the queries (1) and (2) above are respectively:

$MUST(\ 0 < STW - SMW\ AND$
$\qquad 0 < EMW - ETW)$
$MAY(\ 0 < STW - SMW\ AND$
$\qquad 0 < EMW - ETW)$

(STW and ETW are as above; SMW and EMW are the starting and ending points of "$Mary\_work$").

In other words, high level queries about consistency (necessity) are answered by checking that a conjunction of bounds on differences (atomic tests) is consistent (follows necessarily) from the constraints in the knowledge base. Given the minimal network, atomic tests can be performed as local checks on such a network. However, different checks are performed in case of $MUST$ and MAY queries; as a result the computational complexity of the cases is different, as it will be discussed in the two following subsections.

3.2.1 MUST queries

Let us consider a query about necessity of the form $MUST(C_1\ AND\ C_2\ \ldots\ AND\ C_n)$, where each C_i is an atomic test of the form $c_i \leq X_i - Y_i \leq d_i$. We distinguish two cases:

- ($n = 1$), i.e., the query involves only one atomic test and thus has the form:

$$MUST(c \leq X - Y \leq d)$$

Let $[a, b]$ be the maximal admissibility range for the difference $X - Y$ (read from the minimal network). The query is satisfied **iff** all the values for $X - Y$ which satisfy the constraints are in $[c, d]$, that is:

$$MUST(c \leq X - Y \leq d)\ \Leftrightarrow\ [c, d] \supseteq [a, b]\quad (3)$$

Intuitively, since the maximal admissibility range $[a, b]$ includes *all* the values for $X - Y$ satisfying the constraints, then any interval $[c, d]$ such that $[c, d] \supseteq [a, b]$ includes all the values for

$X - Y$ satisfying all the constraints. A query involving one constraint can thus be answered in constant time with a simple lookup in the minimal network and a containment check.

- ($n > 1$), i.e., the query involves a conjunction of atomic tests and has the form

$$MUST(C_1\ AND\ C_2\ \ldots\ AND\ C_n).$$

In this case each one of the C_i can be checked independently of the others since the following property holds:

$$MUST(C_1\ AND\ C_2\ \ldots\ AND\ C_n)\ \Leftrightarrow$$
$$MUST(C_1)\ AND \ldots AND\ MUST(C_n)$$

Thus a query about necessity can be answered in time linear in the number of constraints (and thus in the number of variables) in the query.

3.2.2 MAY queries

Let us consider a query about possibility, i.e., of the form $MAY(C_1\ AND\ C_2\ \ldots\ AND\ C_n)$, where each C_i is an atomic test of the form $c_i \leq X_i - Y_i \leq d_i$. This case is more complex than the one of $MUST$ queries since the MAY operator does not distribute over a conjunction.

The base case, however, is similar, in the sense that when $n = 1$, the answer to a query of the form:

$$MAY(c \leq X - Y \leq d)$$

can be provided with a local check on the minimal network. Let $[a, b]$ be the maximal admissibility range for the difference $X - Y$ (read from the minimal network). The query is satisfied **iff** there is (at least) a value $p \in [c, d]$ for $X - Y$ which satisfies all the constraints, that is:

$$MAY(c \leq X - Y \leq d)\ \Leftrightarrow\ [c, d] \cap [a, b] \neq \emptyset\quad (4)$$

Intuitively, since the maximal admissibility range $[a, b]$ includes *only* values for $X - Y$ satisfying the constraints in the knowledge base, then any interval $[c, d]$ intersecting $[a, b]$ contains at least one value for $X - Y$ satisfying all the constraints.

The case where the consistency of a conjunction of constraints has to be checked is more complex since atomic tests are not independent of each other. For instance, consider the knowledge base formed by the following constraints:

$$\{0 \leq X - Z \leq 30,\ 5 \leq Z - W \leq 25,$$
$$10 \leq Y - X \leq 20,\ 15 \leq Y - Z \leq 30\}\quad (5)$$

and the query:

$$MAY(15 \leq Y - Z \leq 20\ AND\ 15 \leq X - Z \leq 20)\quad (6)$$

where STW and ETW (SJW and EJW) are the variables associated with the starting and ending points of "Tom_work" ("$John\_work$") respectively.

Given a knowledge base of temporal information (expressed as bounds on differences), its consistency must be checked before answering queries (or performing updates), since query processing is not interesting in an inconsistent knowledge base. LATER checks the consistency of a set of bounds on differences constraints using the complete algorithm discussed in [10], whose complexity is $O(N^3)$, where N is the number of variables. This algorithm produces the *minimal network* of the set of constraints, i.e., a compact representation of all the solutions. More specifically, for each pair $\langle X, Y \rangle$ of variables, the minimal network provides the *maximal admissibility range* $[a, b]$ for the difference $X - Y$. In other words $[a, b]$ is the set of all and only the values for $X - Y$ consistent with the knowledge base. LATER keeps track of such a network since, as we shall discuss in the following section, this provides interesting computational advantages during query processing.

3 Efficient Query Answering in LATER

At least three different types of high-level queries are important for querying a temporal knowledge base: queries for extracting some piece of information from the knowledge base (e.g., the duration of an event or the relation between two events), queries for checking whether a set of temporal constraints is consistent with or follows necessarily from the knowledge base and hypothetical queries.

LATER provides a high-level language for expressing all these types of queries. Queries in the high-level language are then translated into the corresponding low-level queries on bounds on differences constraints, which are answered efficiently, in a time that is independent of the dimension of the knowledge base. Let us consider the types of queries listed above one at a time.

3.1 Queries for extracting temporal information.

Different high-level primitives are provided: **When, HowLong, Delay** and **Relation**, which give as answer respectively (1) the temporal location of temporal entities (points or intervals), (2) the duration of time intervals, (3) the delay between two time points and (4) the temporal relations between two temporal entities. These queries can be answered by a simple lookup in the minimal network.

For example, given the knowledge base in figure 1, the following query could be asked:

HowLong John_work?
Answer : $4h, 50min - 5h$

This query can be answered by simply reading in the minimal network the maximal admissibility range of the difference between the variables corresponding to the end and start of "$John\_work$".

As a further example, the following query could be asked:

Relation Mary_work, John_work
Answer : $start(Mary\_work)$ *After*
$$start(John\_work)$$
$$end(Mary\_work) \ non\_strict \ Before$$
$$end(John\_work)$$

Also in such a case the answer can be read directly from the minimal network (and is then translated in the output format above). Notice that the answer corresponds to the following relation in Allen's interval algebra:

Mary_work (During OR Finishes) John_work

3.2 Queries about consistency/necessity.

A second important type of query is that of Yes/No queries for asking whether a set (conjunction) of constraints is true in the given knowledge base. Since in LATER temporal information may be imprecise, it is necessary to distinguish whether some conclusion must necessarily hold (i.e., it is entailed by the knowledge base) or whether it may hold (i.e., it is consistent with the knowledge base). This distinction is similar, e.g., to the one in [19]. Therefore, modal operators must be introduced in the query language in order to distinguish between queries asking whether a set of constraints is possible (consistent) given the knowledge base or whether it follows from the knowledge base.

In LATER queries about necessity/consistency are expressed by prefixing the $MUST$ or MAY operator to the primitives of the high level manipulation language. For instance, given the knowledge base in figure 1, one could ask:

$$MUST(Tom\_work \ During \ Mary\_work) \qquad (1)$$

$$MAY(Tom\_work \ During \ Mary\_work) \qquad (2)$$

(1) corresponds to asking whether the relation *Tom_work During Mary_work* is entailed by the knowledge base; (2) asks whether it is consistent with the knowledge base.

Given the knowledge base in figure 1, the answer to (1) is negative while the answer to (2) is positive.

Conjunction is also provided, so that one can ask for the necessity/consistency of a conjunction of temporal constraints. For example, one could ask the following query:

the approaches that maintain the minimal network and those that perform reasoning at query time [23] discussing how our approach strongly supports the former alternative (since we deal efficiently with complex queries and with a class of updates).

2 Representing time in LaTeR

LaTeR is a general purpose manager of temporal information conceived as a "knowledge server" that can be loosely-coupled with different Artificial Intelligence and database applications [4, 5]. We believe that a knowledge server must have a predictable behavior. This has at least two main consequences: (i) from the inferential point of view, complete temporal reasoning must be performed; (ii) from the computational point of view, reasoning must be performed in polynomial time. Moreover, a friendly interface language for interacting with the system must be available; in particular, a powerful query language must be provided and query processing must be performed very efficiently.

LaTeR is a two-level architecture: the higher level provides the manipulation and query interface language (to which we shall return in the following); the lower level is based on the use of a constraint framework.

LaTeR assumes that time is linear, totally ordered, continuous and metric. Time points are the basic entities; an interval I is defined as a convex set of time points with a starting and an ending point, denoted respectively as $start(I)$ and $end(I)$ (with $start(I) < end(I)$). The **distance between time points** is the basic primitive in our approach and is defined as follows:

Given two time points P_1 and P_2, the assertion distance(P_1,P_2,[a,b]) is true iff the distance between P_1 and P_2 is between a and b, where $a, b \in \mathcal{R}, a \leq b$.[1]

The notion of distance is isomorphic to the notion of difference between reals. Thus, standard and well-known constraint propagation techniques (see [8] or frameworks such as TCSP and STP [10]) can be used to implement such a notion: the variables correspond to the time points and each assertion distance(P_1,P_2,[a,b]) can be represented as a bound on the difference between the variables X_1 and X_2 corresponding to P_1 and P_2, i.e., as a linear inequality of the form:

$$a \leq X_2 - X_1 \leq b$$

In order to achieve the goal of tractable complete reasoning we limited the expressive power to deal

---

[1] We consider also the case where one of the extremes a and b or both of them are not included, i.e. the range is partially or completely open.

only with conjunctions of bounds of differences, in which complete constraint propagation is performed in $O(N^3)$ (where N is the number of variables). The expressive power of LaTeR's lower level is thus the one of STP [10].

LaTeR provides a high-level interface language for manipulating and querying a temporal knowledge base. Each assertion in such a language is translated (in constant time) into a set of lower-level constraints (bounds on differences). Given the restrictions above on the lower level, we have some restrictions on the expressive power of the interface language. In particular, the following types of information can be expressed: precise or imprecise location of time points and intervals, precise or imprecise duration of time intervals, precise or imprecise delay between time points, qualitative relations between points, intervals or points and intervals, limiting to the *continuous pointisable* relations [22] (as discussed in [19] this is not too restrictive in practice since many commonly used relations are indeed continuous pointisable). Figure 1 provides examples of assertions in LaTeR's high level language (see [4] for a definition of the language).

John_work Since $14^{00} - 14^{30}$ *Until* $18^{00} - 19^{00}$
start(Mary_work) $10 - 40$ *min After*
       start(John_work)
Mary_work Lasting AtLeast $4\,h, 40\,min$
end(Mary_work) non_strict Before
       end(John_work)
Tom_work Since 14^{15} *Until* 18^{30}
Tom_work During John_work

Figure 1: A simple knowledge base.

For example, the first assertion localizes (in an imprecise way) the interval of time corresponding to "*John_work*"; the second defines a delay between the starting point of "*Mary_work*" and the starting point of "*John_work*"; the third defines the duration of "*Mary_work*". The *non_strict* operator can be used in conjunction with the precedence (and containment) relations to express that the relation itself is not strict (in the example the meaning is that the end of "*Mary_work*" is before or equal the end of "*John_work*").

As an example of the translation of high-level assertions into bounds on differences, the last assertion in figure 1 is translated into bounds on differences as follows:

$$(0 < STW - SJW) \land (0 < EJW - ETW)$$

Efficient query answering in LaTeR*

V. Brusoni and **L. Console** and **P. Terenziani**
Dip. Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
E-mail: {brusoni,lconsole,terenz}@di.unito.it

Abstract

In the paper we address the problem of answering queries efficiently in heterogeneous temporal knowledge bases (in which qualitative and quantitative pieces of information are amalgamated). In particular, we first outline a powerful high-level language for querying a temporal knowledge base. We then show that, in our language, if the minimal network computed during consistency checking is maintained, then queries can be answered efficiently in time that depends only on the dimension of the query and is independent of the dimension of the knowledge base. Finally, we discuss how our approach can deal efficiently also with updates and, specifically, with sequences of interleaved updates of the knowledge base and queries.

1 Introduction

A lot of attention has been paid in the Artificial Intelligence community to the problem of dealing with time [2, 22]. In particular, most artificial intelligence approaches focus on reasoning issues [1, 15, 18, 20, 21]. On the other hand, the problems of (i) designing a high level language for manipulating and querying temporal knowledge bases and (ii) answering (complex) queries efficiently have been often disregarded. Some of these problems have been faced in the database community [14, 16, 17] where, however, reasoning and complexity issues received only a limited attention.

The aim of this paper is to reconcile these two complementary tendencies in a general-purpose manager of temporal information: LaTeR (*La*yered *T*emporal *R*easoner). In LaTeR heterogeneous temporal information (that is, qualitative and quantitative information) is amalgamated in a principled way start-

ing from the notion of distance between time points. LaTeR, moreover, provides a high-level language for manipulating temporal information; the expressive power of the language has been limited in such a way that complete constraint propagation can be performed in polynomial time (section 2 sketches those aspects of LaTeR that are relevant in this paper, see [4, 5] for more details).

The paper defines a powerful query language including modal operators for asking whether a set of assertions follows necessarily from a knowledge base or it is only possibly true and supporting yes/no queries, queries for extracting temporal information and hypothetical queries. We believe that having a powerful language for querying temporal knowledge bases is fundamental for the practical applicability of managers of temporal information.

The main goal of the paper is to propose an approach for answering queries efficiently in a temporal knowledge base (section 3). Notice that the problem is interesting only in case the knowledge base is consistent since answering queries such as those mentioned above in an inconsistent knowledge base is banal.

We show that in our language, if we maintain the propagated knowledge base ("minimal network") obtained as a result of checking consistency of the knowledge base, then the complexity of answering queries is independent of the dimension of the knowledge base and depends only on the dimension of the query, where the dimension of a knowledge base (query) corresponds to the number of temporal entities involved in the knowledge base (query).

A critical aspect when the minimal network is maintained is that of updating such a network each time the knowledge base is updated (since, in principle, the whole network has to be recomputed after every update). In the paper we discuss how updates to the temporal knowledge base and interleaved sequences of updates and queries can be dealt with efficiently in our approach (section 4).

In section 5 we compare our approach to related ones. In particular, we consider the trade-off between