Temporal Reasoning with Classes and Instances of Events

Paolo Terenziani
DISTA, Univ. del Piemonte Orientale "Amedeo Avogadro"
Corso Borsalino 54, Alessandria, Italy
Phone: +39 0131 287447 -- terenz@mfn.unipmn.it

Abstract

Representing and reasoning with both *temporal constraints* between classes of events (e.g., between the types of actions needed to achieve a goal) and *temporal constraints* between instances of events (e.g., between the specific actions being executed) is a ubiquitous task in many areas of computer science, such as planning, workflow, guidelines and protocol management. The temporal constraints between the classes of events must be *inherited* by the instances, and the *consistency* of both types of constraints must be checked. In this paper, we propose a general-purpose domain-independent knowledge server dealing with these issues. In particular, we propose a formalism to represent temporal constraints, we show two algorithms to deal with inheritance and to perform temporal consistency checking, and we study the properties of the algorithms.

Keywords: Temporal constraints between classes and instances of events, Inheritance, Consistency, Prediction

1 Introduction

The need to represent and reason with classes (i.e., sets of individuals), instances (specific individuals) and inheritance is ubiquitous in many fields of Computer Science (and, in particular, of Artificial Intelligence -AI) and in many applications. Thus, a lot of works in AI focused on this problem, in order to provide *once and for all* systems dealing with these phenomena [14]. KL-ONE [3] and KL-ONE-like hybrid knowledge representation systems (henceforth HKRS) are probably the most popular examples of these types of systems (see, e.g., the survey in [11]). HKRS were usually conceived as task and domain independent *knowledge servers*, providing other systems and problem solvers with facilities for storing and reasoning with classes, instances and inheritance [9]. This showed to be very advantageous both from the conceptual and from the engineering point of view: instead of having to deal from scratch with classes, instances and inheritance, programmers and knowledge engineers could use a HKRS to this purpose, and focus on the specific problems of their task/application domain. In HKRS, a terminological component (called T-Box) is used to represent the description of classes, and an assertional one (A-Box) is used to deal with the instances of the terminological classes. *Classification* is used to determine the exact place of each class in the class taxonomy. Inheritance of properties is supported, as well as integrated reasoning between instances and classes: the *realization* process [11] determines (considering inheritance of properties and the description of classes and instances) all the most specific classes of which a given assertional entity is an instance. HKRS are widely and fruitfully applied in different fields of AI and Computer Science (see, e.g., in [15] a survey on some paradigmatic applications).

There is an obvious temporal counterpart to the problem of dealing with classes, instances and inheritance: in many areas, such as planning, workflow management, protocol/guidelines management and so on one usually wants to specify the actions (henceforth, we use the cover term *event* to denote all types of actions –e.g., agentive or not) needed in order to achieve a given task, and the temporal constraints between them. Notice that an event in a general plan (or workflow, or protocol, or guideline) represents a *class* (set) of instances of events, in the sense that it has specific instantiations for specific executions of the plan itself. On the other hand, while executing (instantiating) a general plan, one has specific *instances* of the classes of events in the plan, which must *inherit* the temporal constraints from their super-classes. Obviously, the instances must respect (i.e., be *consistent* with) the constraints they inherit from their super-classes. Moreover, general plans (or workflows, or protocols, or guidelines) may have a *predictive* role, since they state that a given action has to be executed within a give range of time. However, surprisingly enough, this temporal counterpart has been quite neglected in the AI literature, in the sense that no general-purpose domain-independent knowledge server for *hybrid* (i.e., classes plus instances) *temporal reasoning* has been built (to the best of our knowledge), so that the temporal issues mentioned here have been and are still faced almost from scratch by programmers/researchers in different tasks and applications. In fact, although since the beginning of the 80's many general purpose knowledge servers (the so called *temporal managers*) have been built to deal with different types of qualitative and/or

quantitative temporal constraints (see, e.g., the surveys in [1,19]), none of them supports an explicit treatment of both classes and instances constraints, with the treatment of inheritance and consistency. In the paper, we sketch a *hybrid* temporal approach which overcomes such a limitation *providing users with a temporal corresponding of HKRS*.

In section 2, we discuss the phenomena to be addressed by an hybrid temporal manager. In section 3, we introduce two languages to deal with temporal constraints between instances and temporal constraints between classes respectively. Since the main goal of this paper is that of focusing on the *integration* of constraints between classes and between instances, we deliberately chose two languages which are based on a well-known constraint framework (i.e., *STP* [6]), whose properties are well known. In section 4, we deal with constraint inheritance and hybrid (classes/instances) temporal reasoning to check consistency in case the observations on instances are not complete. In section 5, we extend consistency checking to the case where observations are compete (i.e., when all the events which actually occurred have been observed). Finally, in section 6, we further carry on the parallelism between our approach and HKRS approaches, enlightening future research issues in the fields of knowledge representation and temporal reasoning.

2 Temporal constraints between classes and between instances of events

2.1Classes and Instances of Events

In the introduction, we sketched the temporal counterpart of the well known dichotomy between classes and instances. "Classes of events" may correspond to terminological classes (T-Box concepts), and "instances of events" to (A-Box, i.e., assertional) instances. For example, in a general guideline or plan (e.g., in the clinical field), one may represent the event (action) of "performing a laboratory test". Such an event stands for a class (of events), since it represents a set of individual occurrences of "performing a laboratory test", taking place at definite intervals of time. A specific person may execute, at a given time, a specific laboratory test. This is, of course, an instance event (i.e., a specific occurrence) of the class of events above. This can be graphically represented as in Figure 1, where LT1, LT2, ... LTk represent specific instances (*Instance-of* arcs) of the event class "Lab_Tests" occurring at specific intervals of time.

2.2Temporal Constraints Inheritance

Usually, general plans (guidelines, protocols, workflows) contain *temporal constraints* between (classes of) events. For example, in the CLASSES part of Figure 1, we graphically represent in a simplified way part of a guideline for the management of laboratory tests in an hospital. The general guideline represents the fact that the *reservation* (RS) of each test must be done between 1 and 7 days before the *lab-test* (LT), and that the results of the tests are *reported* (RP) within 1 and 48 hours after the end of the test. Of course, these are *temporal constraints between classes of events*, which might be instantiated many times, for different instantiations of the classes of events (see the INSTANCES part of Figure 1). However, it is important to notice that the temporal constraints at the class level are "relational" constraints, in the sense that they have to be inherited only by "corresponding" pair of instances of the related classes. For example, in Figure 1, the constraint between RS and LT states that each instance of Reservation must be 1-7 days before the *corresponding* instance of Lab-Test (and not before *all* LT instances!). In general, a temporal constraint R between two classes C1 and C2 of events involves an underlying relation pairing instances of the two classes. This correspondence relation has been recognised, e.g., by [10,16], who called it *correlation*. Correlation is symmetric and transitive [10,16]. Its nature depends on the problem and the context. Even in our simple clinical example, correlation may be specified in different ways[1]. Thus, in general, *different* rules could be devised to infer whether two instances of events are correlated or not, *depending on the specific context and domain*. Modelling correlation is outside the goals of this paper. Further discussions on correlation are in the conclusions and in [10,16]. In the example in Figure 1, we suppose that RSi is correlated to LTi which, in turn, is correlated to RPi, $1 \le i \le k$.

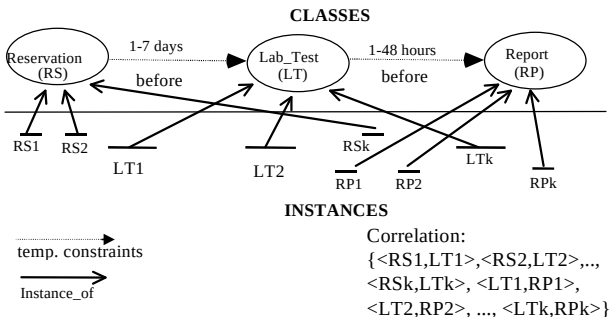

Figure 1 Temporal constraints between classes of events and between instances.

[1]Even in our simple example, correlation may depend on the level of detail used to describe events, and on assumptions on the specific application. E.g., an instance R of Reservation may be correlated to an instance L of Lab_Test if

(1) both L and R refer to the same patient code and to the same type of test (in case a patient cannot have multiple tests of the same type)

(2) both L and R refer to the same patient and to the same type of test in the same date (if a patient cannot have two tests of the same type exactly at the same time)

(3) both L and R refer to the same "unique code" (this is the extreme case: the correspondence is given by a code, unique for each execution of a lab test).

2.3 Reasoning: consistency checking

While most KL-ONE-like approaches support *classification* and *realization* [11], in the temporal case one is usually interested in *checking the consistency* of *temporal constraints*. In particular, temporal consistency can be checked on the classes alone, on the instances alone, or to the merge of the constraints on classes and the constraints on instances, (i.e., considering *inheritance*).

2.4 Reasoning: prediction

In KL-ONE-like systems, the descriptions of classes play a predictive role, in the sense that they predict a set of properties and property value restrictions for the instances. Analogously, in the temporal domain, a plan (or protocol, or guideline) is "predictive", in the sense that, if one has observed a given action E1 which is an instance of a class of events E in a plan, and the class E' follows E in the general plan, one expects to observe an instance of E' in a time consistent with the temporal constraints between the classes of events E and E' in the plan. In domains where one is certain to have a full and *complete observability* of events, the consistency check of the temporal constraints must take into account "prediction", since not having observed a given instance of event in a given range of time may indicate an inconsistency.

2.5 Hybrid Temporal Manager

To summarize, the goal of the work described in this paper is to propose a general purpose knowledge server (temporal manager) which offers a support for

- explicitly *representing* (i) temporal constraints between classes and instances of events, (ii) instance-of relations, and (iii) correlations

-*reasoning* about *inheritance* of temporal constraints, and performing *consistency checking* and *prediction*.

On the other hand, in this paper we do not deal with the representation of the internal description of events (which are considered as “primitive” entities; see also the discussions in section 6).

3 A hybrid approach to temporal reasoning

3.1 Language for temporal constraints between instances of events (ITL)

The basic notion in our temporal ontology are time points. A time interval I is a convex set of points between a starting (Start(I)) and an ending (End(I)) point. Different types of temporal constraints can involve instances of events. **Dates** locate instances of events in time and can be precise (see Ex.1) or imprecise (Ex.2,3).

(Ex.1) RS1 started on 10/1/98 at 10:00 and ended on 10/1/98 at 10:05

(Ex.2) RS2 started on 10/1/98 at 10:10-10:15 and ended on 10/1/98 at 10:20

(Ex.3) LT1 started on 15/1/98 at 9:00-9:40 and ended on 15/1/98 at 10:00

Dates can be expressed in our language for temporal constraints between instances of events (called ITL) using the predicate **date(E,L1,U1,L2,U2)**, stating that the starting point of E is between L1 and U1 and its ending point is between L2 and U2. Also **durations** can be precise or not (e.g., Ex.4)

(Ex.4) LT2 lasted at least 1 hour

Durations are represented in ITL by the predicate **duration(E,L1,U1)**, stating that L1 and U1 are the minimum and maximum durations of E respectively. **Delays** represent (in a precise or imprecise way) the temporal distance between pairs of instances (more precisely, between two of their endpoints; see, e.g., Ex5)

(Ex.5) RS2 started 4-5 minutes after the end of RS1

Delays are represented in ITL by the predicate **delay(P1,P2,L1,U1)**, stating that L1 and U1 are the minimum and maximum delay between P1 and P2, where P1 and P2 are endpoints of events. On the other hand, **qualitative temporal constraints** do not involve any metric of time, allowing one to deal with the relative position of two instances of events (ex.6). Currently, ITL considers the qualitative constraints of the *Continuous Interval Algebra* i.e., the subset of relations of Allen's Interval Algebra which can be mapped onto conjunctions of constraints between points, excluding disequality [19]. We chose such a subset because it has very interesting computational properties, and nevertheless it proved to be very important in many practical applications [5,18][2].

(Ex.6) LT1 was before LT2

E.g., the constraints in (Ex.1-Ex.6) can be represented in ITL as shown by (ITL1).

(ITL1):

date(RS1,10/1/98 at 10:00,10/1/98 at 10:00,10/1/98 at 10:05,10/1/98 at 10:05),
date(RS2,10/1/98 at 10:10,10/1/98 at 10:15,10/1/98 at 10:20,10/1/98 at 10:20),
date(LT1,15/1/98 at 9:00,15/1/98 at 9:40,15/1/98 at 10:00,15/1/98 at 10:00),
duration(LT2,1 hour,∞),
delay(End(RS1),Start(RS2),4 min,5 min),
before(LT1,LT2)

All the constraints in ITL above can be easily mapped onto distances between time points, or, better into bounds on differences (b.o.d.) constraints of the general form

$$L \leq X - Y \leq U \quad \text{where L, U are real numbers}$$

where X and Y represent time points and L and U their minimum and maximum temporal distance (i.e., onto the STP framework). The semantics of ITL predicates can be

[2]Notice that arbitrary disjunctions of temporal constraints (as in "LT1 was during LT2 or LT2 lasted at least 1 hour") cannot be specified in ITL, as well as some disjunctive relations in Allen's algebra such as "LT1 before or after LT2".

specified in terms of b.o.d. constraints on the distance between points as follows:

date(E,L1,U1,L2,U2) $\Leftrightarrow$ (L1 ≤ Start(E) - X0 ≤ U1) Λ (L2 ≤ End(E) - X0 ≤ U2)
duration(E,L1,U1) $\Leftrightarrow$ L1 ≤ End(E) - Start(E) ≤ U1
delay(P1,P2,L1,U1) $\Leftrightarrow$ L1 ≤ P2 - P1 ≤ U1

Notice that dates are represented by distances from a reference time point X0 for the whole knowledge base, and that P1 and P2 are starting/ending points of events. As examples of qualitative relations, let us consider *before* and *during* between two time intervals E1 and E2

before(E1,E2) $\Leftrightarrow$ 0 < Start(E2) - End(E1)
during(E1,E2) $\Leftrightarrow$ (0 < Start(E1) - Start(E2)) Λ (0 < End(E2) - End(E1))

Bounds on differences (and the *STP* framework) have been widely used in the AI literature in order to represent and reason with temporal constraints (consider, e.g., [5,6,7]). *Correct* and *complete* reasoning on b.o.d. can be performed efficiently using an all-to-all shortest path algorithm which provides an inconsistency or the upper and lower bounds for the distance between each pair of time points (also called *minimal network*), and which operates in a time that is cubic in the number of time points [6]. A simple test in the all-shortest-path algorithm allows it to detect inconsistencies, at no additional cost [6]. For example, reasoning on the b.o.d. corresponding to (ITL1) finds their consistency and infers that, e.g., RS2 started at 10:10 and LT2 started after 15/1/98 at 10:00.

The temporal high-level language we described until now is very similar to the ones of many STP-based temporal managers in the AI literature (see, e.g., [1,5,6,19]). In order to be able to integrate temporal constraints between classes and between instances, we must extend ITL. We introduce the predicate **Instance_of(E1,C1)** to state that E1 is a specific instance of the class of events C1. In the following, we suppose that we have the classes in Figure 1, and to have observed only the instances RS1 and RS2 (of Reservation) and LT1, LT2 (of Lab_Tests). The class/instance relations can be represented in ITL as shown by (ITL2):
(ITL2):

Instance_of(RS1, Reservation),
Instance_of(RS2, Reservation),
Instance_of(LT1, Lab_Tests),
Instance_of(LT2, Lab_Tests)

Predicate **COR** is introduced to represent correlations between instances of events. In our example, we assume (as in Figure 1) that LT1 is correlated to RS1 and LT2 is correlated to RS2. This can be expressed in ITL by (ITL3): COR(RS1,LT1), COR(RS2,LT2)

Finally, it is useful to indicate the set IKB_Elements of all the instances e.g., as shown in (ITL4)
(ITL4): {RS1,RS2,LT1,LT2}

In ITL, a Knowledge Base of temporal constraints between instances (IKB for short) is a quadruple <IKB_Elements, IKB_Instance_of, IKB_COR, IKB_Constraints>, where IKB_Elements is a set of instances of events, IKB_Instance_of is a set of Instance_of assertions, IKB_COR a set of correlations and IKB_Constraints a set of temporal constraints on instances of events. In our example, we have IKB = <ITL4,ITL2,ITL3,ITL1>.

Axioms (Ax1) and (Ax2) (and the logical formulae in section 3.2) are introduced to make explicit our intended semantics of an IKB: IKB is a representation of the instances of events which have been observed until NOW (where NOW is the system time when a call to the temporal manager is done). (Ax1) states that if one instance x of event has been observed (i.e., x∈ IKB_Elements), then it has been observed to start before (or equal to) NOW. Of course, the temporal reasoning algorithms we describe in the following sections have to respect such a semantics (in other words, they can be seen as a procedural implementation of such a semantics).
(Ax1) ∀x x∈ IKB_Elements ⇒ Start(x) - NOW ≤ 0

If we *hypothesize* that observations are complete, the fact that an instance x of event has not been observed (i.e., x∉ IKB_Elements) implies that it did not start until NOW (see Axiom Ax2).
(Ax2) ∀x x∉ IKB_Elements⇒ NOT(Start(x) - NOW ≤ 0)

3.2 Language for temporal constraints between classes of events (CTL)

In general, all the types of temporal constraints discussed above can also be expressed between classes of events. For instance, considering again the example in Figure 1, one could assert the following temporal constraints:
(Ex.7) Laboratory tests are made between 1 and 7 days after the reservation
(Ex.8) Laboratory tests last between 30 minutes and 48 hours
(Ex.9) Results are reported between 1 and 48 hours after the end of the tests
(Ex.10) Results are reported after the tests

Thus, we used the same predicates as above to express them into our temporal language for classes of events (CTL for short). Ex.7-Ex.10 are represented in CTL as follows:
(CTL1):

Cdelay(End(Reservation),Start(Lab_Tests),1day, 7day),
Cduration(Lab_Tests, 30 min, 48 hour),
Cdelay(End(Lab_Tests),Start(Report), 1 hour, 48 hour),
Cafter(Report,Lab_Tests)

The predicates on classes are basically the same as for instances (we put the prefix C to distinguish them); however, when applied to classes, durations, delays (here we consider just the delays between the starting points of two classes; the other cases are analogous) and qualitative relations have a different meaning, as shown below.

Cduration(C,L1,U1) ⇔
∀ E Instance_of(E,C) ⇒ L1 ≤ End(E)- Start(E) ≤ U1

Cdelay(Start(C1),Start(C2),L1,U1) ⇔
(∀ C1',C2' (Instance_of(C1',C1) Λ Instance_of(C2',C2) Λ COR(C1',C2')) ⇒ (L1 ≤ Start(C2') - Start(C1') ≤ U1) Λ
(∀ C1' Instance_of(C1',C1) ⇒
(∃ C2' Instance_of(C2',C2) Λ COR(C1',C2'))))

As example of qualitative relations, let us consider the relation "before":

Cbefore(C1,C2) ⇔ (∀ C1',C2' (Instance_of(C1',C1) Λ Instance_of(C2',C2) Λ COR(C1',C2')) ⇒
(0 < Start(C2') - End(C1')) Λ
(∀ C1' Instance_of(C1',C1) ⇒
(∃ C2' Instance_of(C2',C2) Λ COR(C1',C2'))))

While durations are simply inherited by all instances, qualitative relations and delays are only inherited by *correlated* pairs of instances (see section 2). The second conjuncts in the definition of Cdelay and Cbefore formalize the "predictive" character of delays and qualitative relations between classes of events. For example, given the constraint between classes Cbefore(C1,C2), the observation of an instance of C1 implies the later occurrence of a correlated instance of C2. Finally, the predicate EventClass is introduced in CTL in order to declare the classes of events being considered. Thus, in the example in Figure 1, we would have (CTL2) below

(CTL2): EventClass(Reservation),
EventClass(Lab_Tests), EventClass(Report)

Thus, in our language CTL, a KB of temporal constraints between classes of events (CKB for short) can be defined as a pair <CKB_EventClass, CKB_Constraints> (<CTL2,CTL1> in our example).

4 Hybrid consistency checking (no prediction)

If one has *only* constraints between classes, the fact that they are classes is irrelevant from the point of view of temporal reasoning; they can be interpreted as primitive (individual) events and standard temporal reasoning can be performed on them (see, e.g., [2] as regards temporal constraints in general plans). On the other hand, hybrid temporal reasoning takes in input a KB of temporal constraints between classes and a KB of temporal constraints between instances, and gives as output the upper and lower bounds on the distance between each pair of starting and ending points of instances (i.e., the minimal network) or an inconsistency. The procedure Integrated_Reasoning in Figure 2 deals with the case (common in many applications) in which observations are incomplete, i.e., instances of events can occur and not be observed (i.e., not be present in the IKB).

Before performing hybrid temporal reasoning, temporal constraints in the high-level language are translated into the corresponding b.o.d. constraints (steps (1) and (2)). In step (3), Set_NOW updates the constraints in IKB_Constraints adding the constraint represented in Axiom (Ax1)[3]. Then, temporal reasoning is performed separately on instances and on classes (using the all-shortest-path algorithm on b.o.d. constraints [6], called here *Temporal_reasoning*) to check whether each one of them is independently consistent and to infer the implied temporal constraints separately (see steps (4) and (5); let BOD_IKB_Con' and BOD_CKB_Con' the resulting sets of constraints). Step (6) performs the transitive closure of correlation relations. The rest of the procedure deals with the integration of the two levels of constraints. The basic idea is that of inheriting (accordingly with the semantics specified is section 3.2) the temporal constraints between classes on the instances of events, and then performing temporal reasoning on instances (applying again the all-shortest-path algorithm) on the union of the inherited plus the instances constraints. Step (7) implements the inheritance of durations of events. All distances t ≤ End(E) - Start(E) ≤ u between the ending point and the starting point of an event class E must be inherited by all the instances of the class. Thus, they are added to the constraints in BOD_IKB_Con'. For each pair of correlated instances E1 and E2, Step (8) deals with the inheritance of qualitative relations and delays from the corresponding classes of events. Finally, step (9) performs integrated reasoning at the level of instances, considering also the constraints inherited from the classes of events. The procedure stops reporting an inconsistency if a call to Temporal reasoning (steps 4, 5, and 9) finds it.

Procedure Integrated_Reasoning
(<CKB_EventClass, CKB_Constraints>,
<IKB_Elements,IKB_Instance_of, IKB_COR,
IKB_Constraints>)
(1) BOD_IKB_Con **:=** Transform(IKB_Constraints)**;**
(2) BOD_CKB_Con **:=** Transform(CKB_Constraints)**;**
(3) BOD_IKB_Con **:=**
Set_NOW(BOD_IKB_Con,NOW)**;**
(4) BOD_IKB_Con' **:=**
Temporal_reasoning(BOD_IKB_Con)**;**
(5) BOD_CKB_Con'**:=**
Temporal_reasoning(BOD_CKB_Con)**;**
(6) IKB_COR **:=** Closure(IKB_COR)**;**
(7) **Forall** C1 \ EventClass(C1) ∈ CKB_EventClass Λ
t ≤ End(C1) - Start(C1) ≤ u ∈ BOD_CKB_Con' **do**
Forall E \ Instance_of(E,C1) ∈ IKB_Instance_of **do**
BOD_IKB_Con'**:=** BOD_IKB_Con' ∪
{t ≤ End(E) - Start(E) ≤ u} **od od;**

[3] In our example, we have observed (the beginning of) LT2, but there is only the constraint before(LT1,LT2) concerning LT2 in IKB. Thus, in the mapping on b.o.d., we have *Start(LT2) - X0* < ∞, and the effect of Set_NOW is to change this constraint into *Start(LT2) - X0* ≤ *NOW*

(8) **Forall** E1,E2 ∈ IKB_Elements, E1≠E2\ COR(E1,E2)
Let C1∈ CKB_EventClass and
C1∈ CKB_EventClass the corresponding classes /* i.e., Instance_of(E1,C1) and Instance_of(E2,C2) hold */
Istantiate on E1 and E2 the constraints in CKB_Constraints between C1 and C2
(9) Minimal_Network **:=**
Temporal_reasoning(BOD_IKB_Con'**);**

Figure 2. Procedure Integrated_Reasoning

For example, let us apply Integrated_Reasoning to <CTL2,CTL1> and <ITL4,ITL2,ITL3,ITL1> described above, supposing that NOW=18/1/98 at 18:00. Step (7) inherits the constraints on the duration of LT1 and LT2 (which must last between 30 minutes and 48 hours). Step (8) inherits the delay of 1-7 days between correlated pairs of instances of Reservation and Lab_Tests. In the example, and taking minutes as the basic granularity, this corresponds to adding the constraints $30 \leq End(LT1)-Start(LT1) \leq 2880$, $30 \leq End(LT2)-Start(LT2) \leq 2880$, $1440 \leq Start(LT1)-End(RS1) \leq 10080$, and $1440 \leq Start(LT2)-End(RS2) \leq 10080$ into the temporal constraints between instances of events. The final application of Temporal_reasoning does not detect any inconsistency and provides, among the others, the constraints that: LT1 starts on 15/1/98 at 9-9:30; LT2 starts between 15/1/98 at 10:00 and 17/1/98 at 10:20 and ends between 15/1/98 at 11:00 and 19/1/98 at 10:20.

More generally, the following property holds:

Property 1 *The procedure Integrated_Reasoning is correct with respect to the logical semantics of the temporal language we introduced in subsections 3.1 and 3.2.*

Proof (Sketch) The proof is based on the fact that all and only the temporal constraints specified by the semantics of the constructs in CTL (Cduration, Cdelay etc.) are inherited at the level of instances of events (steps (7) and (8)), and then correct and complete temporal reasoning is performed at the level of instances of events via the all-to-all shortest path algorithm (step 9). ♦

Given the proof sketched above, Integrated_Reasoning is complete as regards consistency checking on the classes in CKB plus the instances in IKB. However, it does not consider the "predictive" part in the logical semantics of delays and qualitative relations between classes, since it does not add the predicted (correlated) instances into the IKB. However, this is reasonable in many applications. For example, in all the applications where observations are incomplete (i.e., where Axiom Ax2 does not hold), prediction has no impact on consistency checking. In fact, even if the predicted events should have occurred in the past (i.e., before NOW), not having them in the IKB does not imply an inconsistency: maybe they occurred and were not observed (inserted in the IKB). Thus, Property 2 holds:

Property 2 *In the case of incomplete observations, the procedure Integrated_Reasoning checks consistency in a correct and complete way with respect to the logical semantics of the temporal language.*

5 Hybrid consistency checking (complete observations)

In many applications, the "predictory" part of temporal constraints between classes must be considered. For example, in applications where one can hypothesize that observations are complete (i.e., Axiom Ax2 holds), "prediction" must be used to detect inconsistency. In fact, in such a case, the absence of the observation of an instance of an event which, according to the constraints among classes, should have already happened (and be observed), gives an inconsistency. This can be coped with as in Procedure Integrated_Predictory_Reasoning in Figure 3. The procedure first calls Integrated_Reasoning (step 1) and then consider "predictions". In the procedure, we denote by CKB_Connected(C) the set of all classes in CKB that can be reached (directly or indirectly) from the class C via some temporal constraint (a delay or a qualitative relation; i.e., CKB_Connected(C) represent the set of classes correlated to C). This can be easily computed a-priori by navigating the graph of constraints between classes (see Figure 1). Step (2) implements the "prediction" of new instances. For each instance E in IKB, it considers all the classes in CKB_Connected(C_E) which are connected to the class C_E of which E is an instance. For each one of these classes (say C), it looks whether there is an instance of C which is correlated to E in IKB. If there is not, in step (2.1.1) Add_Instance(C, IKB_Elements, IKB_Instance_of, IKB_COR,E) returns a new instance I' of C and inserts: I' into IKB_Elements, the relation Instance_of(I',C) into IKB_Instance_of, and COR(E,I') in IKB_COR; this amounts to create a new instance I' of C correlated to E, according to the "prediction" part of the semantics of constraints between classes.

Then, in step (2.1.3), the transitive closure of COR is computed, and in 2.1.4 Inh_constr_inst is invoked to consider all the constraints concerning (the starting and ending points of) C in CKB, and to let I' inherit them (inheritance here is analogous to steps (7) and (8) of the procedure Integrated_Reasoning). Step (3) executes temporal reasoning on the resulting set of constraints. Finally, step (4), for each one of the new instances *I* introduced into the IKB (the instances in NEW_INST), checks whether the resulting constraints in the IKB imply that *I* should have started necessarily before NOW. In such a case, an inconsistency is reported. NEC(KB,*test*) checks whether *test* is necessarily true given the constraints in KB (i.e., if *test* is *logically implied* by KB.

considering the minimal network of a KB of bounds on differences constraints, this can be done in a time linear in the time points in *test* [4]).

Procedure Integrated_Predictory_Reasoning
(<CKB_EventClass, CKB_Constraints>, <IKB_Elements,IKB_Instance_of, IKB_COR, IKB_Constraints>)
(0) NEW_INST := ∅;
(1) Integrated_Reasoning (CKB,IKB);
(2) **Forall** E ∈ IKB_Elements **do**
Let C_E ∈ CKB_EventClass the class such that Instance_of(E,C_E) ∈ IKB_Instance_of
Let CKB_Connected(C_E) the set of all classes in CKB_EventClass connected to C_E via temporal constraints
(2.1) **Forall** C ∈ CKB_Connected(C_E) **do**
if NOT (**Exists** E' such that Instance_of(E',C) ∈ IKB_Instance_of Λ COR(E',E) ∈ IKB_COR) **then**
begin
(2.1.1) I' **:=** Add_Instance(C, IKB_Elements, IKB_Instance_of, IKB_COR, E)**;**
(2.1.2) NEW_INST **:=** NEW_INST ∪ {I'}**;**
(2.1.3) IKB_COR := Closure(IKB_COR)**;**
(2.1.4) **Forall** bod ∈ BOD_CKB_Con concerning C **do** Inh_constr_inst(bod,BOD_IKB_Con') **od end od od;**
(3) Minimal_Network := Temporal_Reasoning(BOD_IKB_Con')**;**
(4) **Forall** I ∈ NEW_INST **do**
If NEC(BOD_IKB_Con', Before(start(I),NOW)) **then** INCONSISTENT**; od;**

Figure 3. Procedure Integrated_Predictory_Reasoning

Let us consider again our example. The procedure above inserts two instances RP1 and RP2 of Report into the IKB. RP1 is correlated to RS1 and LT1, and RP2 to RS2 and LT2. Considering the inherited temporal constraints, we infer that RP1 should start between 1 and 48 hours after the end of LT1, i.e., between 15/1/98 at 11:00 and 17/1/98 at 10:00, and RP2 should start between 15/1/98 at 12:00 and 21/1/98 at 10:20. In particular, the starting point of RP1 must be between 15/1/98 at 11:00 and 17/1/98 at 10:00 and thus it is necessarily before NOW (18/1/98 at 18:00 in our example). Thus, an inconsistency is detected. In general, Property 3 holds:

Property 3
The procedure Integrated_Predictory_Reasoning checks consistency in a correct and complete way with respect to the logical semantics of the temporal language we introduced in subsections 3.1 and 3.2.
Proof (Sketch) The proof is based on the fact that Integrated_Reasoning is correct, and its incompleteness is only due to the fact that it does not consider the "predictive" part of the semantics of constraints between classes, which is dealt with by step (2) of Integrated_Predictory_Reasoning. Then correct and complete temporal reasoning is performed at the level of instances of events via the all-to-all shortest path algorithm (step 3). Finally, step 4 is needed to force the fact that observations are complete (see Ax2), checking whether some predicted instance should have been observed necessarily before now. ♦

6 Conclusions and Developments

In planning, workflows, guidelines, protocols and so on, checking whether the temporal constraints in a general plan (protocol, guideline, workflow) are respected by the plan (protocol, guideline, workflow) instantiation is a fundamental task. Such a task involves integrated temporal reasoning considering both the temporal constraints between the classes of events and the (observed) temporal constraints between their instances.

The approach in this paper is, to the best of our knowledge, the first one proposing a *general-purpose* and *domain-independent* knowledge server supporting such a task, thus providing a temporal corresponding of HKRS systems. We think that the parallel between our approach and HKRS systems [11] could give rise to new interesting topics of research in temporal reasoning. For example, a main issue in HKRS concerns the relation between the expressiveness of the terminological and the assertional components [11], ranging from KL-ONE [3], in which the terminological component is very powerful and expressive and the assertional one very limited, to BACK [12], where the two components are *balanced* from both the expressive and computational point of view. Considering this issue, our approach is close to a balanced BACK-like approach. However, as in the research about HKRS's, a lot of work should be done in order to extend the classes and/or the instances languages and the temporal reasoning features (for the sake of simplicity, we currently adopted very simple STP-based languages, whose expressive limitations are well known within the temporal reasoning community), and considering the trade-off between expressive power and the complexity of (complete) reasoning.

Moreover, some HKRS have been extended with an Inferential Box, containing formulae or rules operating on the assertional component (consider, e.g., the I-Box in BACK [13]). Analogously, in our approach, one could introduce a further component, which manages the (domain and application dependent) rules which specify the *correlation* relations between instances of events. Finally, the integration of our approach with a classical HKRS (e.g., BACK) to represent the internal description of classes and instances of events (using concepts and roles) and to exploit the classification and realization facilities would be interesting. In such an extended approach, one could use (in the I-Box rules)

descriptions of the classes of events in T-Box in order to infer, e.g., correlation[4].

Finally, it would be interesting to extend the approach in this paper in order to cope with cases where a one-to-one correspondence between events cannot be assumed (e.g., coming back to the example in Figure 1, where the same reservation can be used for more than one laboratory test). Furthermore, we think that our approach is suitable to be extended in order to use constraints between classes as basic knowledge to be evaluated a-priori, and to be used to check consistency in an incremental way whenever new (constraints on) instances are added (e.g., to deal with least commitment temporal planning).

To conclude, we conceive our hybrid temporal reasoner as a domain and task independent *knowledge server* to be loosely *coupled* with other systems and problem solvers to deal with different problems in different areas (following the lines which have been pointed out by many applications of HKRS [11,15] and e.g., by [1,5] as regards applications of temporal managers dealing with instances only). Currently, we are studying to loosely couple our hybrid temporal manager with GLARE, a system for managing clinical guidelines we developed in cooperation with the physicians of Azienda Ospedaliera S. Giovanni Battista of Torino, Italy [8,17].

References

[1] J. Allen, "Time and Time again: the Many Ways to Represent Time", *Int'l J. Intelligent Systems*, vol. 6, no. 4, pp. 341-355, July 1991.

[2] J. Allen, "Planning as Temporal Reasoning", Proc. KR91, 3-14, 1991.

[3] R. Brachman, and J. Schmolze, "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science*, vol. 9, No. 2, pp. 171-216, April-June 1985.

[4] V. Brusoni, L. Console, and P. Terenziani. "On the computational complexity of querying bounds on differences constraints", *Artificial Intelligence 74(2)*:367-379, 1995.

[5] V. Brusoni, L. Console, B. Pernici, P. Terenziani, "LaTeR: Managing Temporal Information Efficiently", *IEEE Expert* 12(4), 56-64, 1997.

[6] R. Dechter, I. Meiri, J. Pearl, "Temporal Constraint Networks", *Artificial Intelligence* 49, 61-95, 1991.

[7] E. Davis, "Constraint Propagation with Interval Labels", *Artificial Intelligence* 32, 281-331, 1987.

[8] A. Guarnero, M. Marzuoli, G. Molino, P. Terenziani, M. Torchio, K. Vanni, "Contextual and Temporal Clinical Guidelines", *Journal of the American Medical Informatics Association*, 683-687, 1998.

[9] H.J. Levesque and R.J. Brachman, "Expressiveness and Tractability in Knowledge Representation and Reasoning", *Computational Intelligence 3*, 78-93, 1987.

[10] R.A. Morris, W.D. Shoaff, and L. Khatib, "Path Consistency in a Network of Non-convex Intervals", *Proc. thirteenth Int'l Joint Conf. on Artificial Intelligence*, pp. 655-660, Chambery, France, 1993.

[11] B.Nebel, *Reasoning and Revision in Hybrid Representation Systems*, LNCS 422, Springer-Verlag, 1990.

[12] B. Nebel and K. von Luck, "Hybrid Reasoning in BACK", In Z.W. Ras and L. Saitta eds., *Methodologies for Intelligent Systems 3*, North Holland, 260-269, 1988.

[13] J. Quantz and C. Kindermann, "Implementation of the BACK System Version 4", KIT-REPORT 78, Technische Universitat Berlin, December 1990.

[14] E. Rich, K. Knight, *Artificial Intelligence*, McGraw Hill, 1991.

[15] J. Schmolze, W.Mark "The NIKL Experience", *Computational Intelligence 6*, 48-69, 1991

[16] P. Terenziani, "Integrating calendar-dates and qualitative temporal constraints in the treatment of periodic events", *IEEE Trans. on Knowledge and Data Engineering 9(5)*, 1997.

[17] P. Terenziani, G. Molino, and M. Torchio, "A modular approach for representing and executing clinical guidelines", *Artificial Intelligence in Medicine 23*, 249-276, 2001.

[18] P. VanBeek, "Temporal Query Processing with Indefinite Information", *Artificial Intelligence in Medicine*, 3(6), 325-339, 1991.

[19] L.Vila, "A Survey on Temporal Reasoning in Artificial Intelligence", *AI Communications 7(1)*, 4-28, 1994.

[4] Considering for example footnote 1, the rule (2) of correlation could be implemented in the following way. One could describe Reservation (RS for short) and Lab-tests (LT) giving them (among the others) the attributes (slots) patient code (PC), type of test code (TTC) and date of the examination (EXAM_DATE), and insert in I-Box a rule such as

$\forall$ I1,I2 Instance_of(I1,RS) Λ Instance_of(I2,LT) Λ
I1.PC=I2.PC Λ I1.TTC=I2.TTC Λ
I1.EXAM_DATE = I2.EXAM_DATE $\Rightarrow$ COR(I1,I2)