Towards A Temporal Reasoning Approach Dealing with Instance-of, Part-of, and Periodicity

Paolo Terenziani*, Luca Anselma[#]
*Dipartimento di Informatica, Univ. del Piemonte Orientale "Amedeo Avogadro"
Spalto Marengo 33, Alessandria, Italy
Phone: +39 0131 287447 – terenz@mfn.unipmn.it
[#]Dipartimento di Informatica, Universita' di Torino,
Corso Svizzera 185, 10149 Torino, Italy
Phone: +39 011 6706729 – anselma@di.unito.it

Abstract

In many application areas, including planning, workflow, guidelines and protocol management, the description of the domain requires the use of *part-of* relations between events, the modeling of *periodic repetitions* and the treatment of "standard" *temporal constraints* between such events.

Events in plans, workflow etc. represent "classes", in the sense that they can be *instantiated* several times, once for each execution of the plan, guideline etc. Of course, such executions must respect (i.e., be *consistent* with) the temporal constraints explicitly or implicitly (e.g., by the *part-of* relation) conveyed by the class descriptions. In this paper, we propose a tractable domain-independent temporal server dealing with the above phenomena. We first sketch a representation formalism coping with *part-of* and *instance-of* relations, periodicity, and temporal constraints, and then we describe two algorithms to deal with inheritance and to perform temporal consistency checking.

Track 1: Temporal representation and Reasoning in AI

Keywords: Temporal constraint reasoning, temporal constraints between classes and instances of events, Periodic repetitions, Part-of relations, Inheritance, Consistency

1 Introduction

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. Moreover, *part-of* relations between events are often important in order to describe the domain in a structured and top-down fashion (obviously, part-of relations induce implicit temporal constraints between the composite event and its components). Moreover, in many domains (e.g., clinical therapies) events need to be repeated at regular (i.e., periodic) time, so that repetition and periodicity are an essential part of the description.

It is important to 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 respect (i.e, be consistent with) the temporal constraints from their classes. This also means that the (implicit) temporal constraints conveyed by the part-of relations must be respected, as well as those involved by periodicity and repetitions. Moreover, general plans (or workflows, or protocols, or guidelines) may have a *predictive* role. For example, in a general plan, they might state that a given action has to be executed within a give range of time after the execution of the preceding action.

Since the beginning of the 80's many domain-indendent knowledge servers (called *temporal managers*) have been built to deal in an efficient and ad-hoc way with different types of qualitative and/or quantitative temporal constraints (see, e.g., the surveys in [Allen, 91; Vila, 94]). However, to the best of our knowledge, there is no constraint-propagation-based temporal reasoner which can deal in an integrated way with the ***temporal constraints*** involved by the above temporal phenomena.

The final overall goal of our work about temporal reasoning is that of designing a tractable domain-independent and general-purpose temporal manager (**TEMPORE** – TEMPOral REasoner) coping with the temporal constraints between both classes and instances of events, taking into account also those constraints (implicitly or explicitly) conveyed by instance-of and part-of relations and by the specification of periodic repeated actions. This goal is motivated by our work in different applications areas (including train scheduling [Brusoni et al., 96] and multimedia presentations [Adali et al., 00]), and in particular in clinical guidelines automation [Terenziani et al., 01, 02a, 02b] (see subsection 6.1). The work described in this paper extends the initial work in [Terenziani, 02, Terenziani et al., 02b] along such a direction.

In section 2, we introduce the phenomena to be addressed by our temporal manager. In section 3, we describe the high-level language we introduced in order to represent the above phenomena, and the low-level language we use to map the high-level description in an internal form, on which temporal reasoning (via constraint propagation) can be performed. In Section 4 we deal with constraint inheritance and classes+instances temporal reasoning to check consistency in case the observations on instances are complete. In section 5, we extend the algorithm in section 4 in order to operate in the case when there is no observation in the future. Despite the quite wide range on phenomena we have faced, we have aimed to retain tractability and completeness of the reasoning process, and this fact lead us to introduce some restrictions on the expressiveness of the temporal language and some simplifying assumptions. Some extensions of our current approach to overcome the current limitations will be sketched in section 6, where we also discuss applications and future and related works.

2 An introduction to the problem

2.1 Classes and Instances of Events

In most domains, one has to deal with both classes and instances of events. For example, in a general guideline or plan, 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 (i.e., a specific occurrence) of the class of events above.

The possibility of representing part-of relations between events plays a fundamental role in the construction of adequate representation of "structured" events.

2.2 Temporal Constraints Inheritance

Usually, general plans (guidelines, protocols, workflows) contain *temporal constraints* between (classes of) events. Constraints about duration, delays and precedence are typical examples. For example, one might have that *reservation* of each test must be done between 1 and 7 days before the laboratory tests. Of course, these are *temporal constraints between classes of events*, which might be instantiated many times, for different instantiations of the classes of events. Moreover, 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" pairs of instances of the related classes. For example, each reservation must be 1-7 days before the *correlated* [Morris, 93; Terenziani, 97] instance of laboratory test (and not before *all* tests!). 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 [Morris, 93; Terenziani, 97]).

2.3 Periodic events

Repeated events are widespread in many application domains. In many cases (consider, e.g., clinical therapies), repeated events are periodic, since they occur at regular (i.e., periodic) times. In a broad sense, thus, periodic events are special kinds of classes of events, i.e., classes whose instances must respect a given periodic temporal pattern. In a very broad sense, constraints on the periodicity of repetitions are basically constraints that must be "inherited" by instances. However, it is a "non-classical" form of inheritance. In fact, while constraints about durations, delays and precedences regard single instances (durations) or pairs of instances (delays, precedences), periodicity constraints concern sets of instances, imposing constraints on their cardinality and on the temporal pattern they have to respect. Finally, notice that the interplay between part-of relations and periodic events might be quite complex to represent and manage. In fact, in the case of a composite periodic

event, the temporal pattern regards the components, which may, recursively, be composite and/or periodic events. Consider, for instance, the following example, concerning the clinical guideline about the treatment of multiple mieloma.

(Ex.1) *The therapy for multiple mieloma is made by six cycles of 5 days treatment, each one followed by a delay of 23 days (for a total time of 24 weeks). Within each cycle of 5 days, 2 inner cycles can be distinguished: the Alkeran treatment, to be provided twice a day, for each of the 5 days, and the Deltacorten treatment, to be provided once a day, for each of the 5 days. These two treatments must be performed in parallel.*

In Ex.1, e.g., the instances of the Alkeran treatment must respect the temporal pattern "twice a day, for 5 days", but such a pattern must be repeated for six cycles, each one followed by a delay of 23 days, since the Alkeran treatment is part of the general therapy for multiple mieloma.

2.4 Reasoning: consistency checking and prediction

Checking the consistency of *temporal constraints* is a fundamental task in many applications involving time. In particular, temporal consistency can be checked on the classes alone, on the instances alone, or on the merge of the constraints on classes and the constraints on instances, (i.e., considering *inheritance*). Notice that, when considering instances, one should also take into account the fact plans (or protocols, or guidelines) have a "predictive" role. For instance, if one has observed a given event E_1 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.
The problem is more complex in the (more common) case where the full observability only concerns the past (e.g, if *proactive* updates are not allowed, so that no event in the future can be instantiated). In such a case, not having observed a predicted instance does not rise an inconsistency (since such an instance could be observed in the future), unless the temporal constraints impose that it should have started before NOW.

2.5 Integrated Temporal Manager

To summarize, the goal of the work described in this paper is to propose a *tractable* knowledge server (temporal manager) which offers a support for:
(i) explicitly *representing* a description of the domain in terms of part-of and instance-of relations
(ii) explicitly *representing* the temporal constraints between classes events, including periodic repetitions, and the temporal constraints between instances events, including correlation relations
(iii) *reasoning* about *inheritance* of temporal constraints, and performing *consistency checking*.

3 An integrated approach to temporal reasoning

3.1 High-level language for temporal constraints about instances of events (ITL)

In this paper, we mainly focus on the issue of integrating temporal constraints deriving from sources of different nature (e.g., periodicity vs. part-of relations). For such a reason, we chose to adopt a quite "standard" language to represent qualitative and quantitative temporal constraints between instances of events. In particular, our high-level constraint language provides primitives in order to represent
(i) (possibly imprecise) dates
(ii) (possibly imprecise) delays between endpoints of events
(iii) (possibly imprecise) durations
(iv) (possibly imprecise) qualitative constraints between endpoints of events
ITL has been deliberately designed in such a way that only constraints that can be mapped onto conjunctions of bounds on differences (i.e., on an STP framework [Dechter et al., 91]) can be represented (see [Terenziani, 02] for more details).
Besides temporal constraints, our high-level language also allows one to specify:
(i) Instance-of relations (between an instance and its class)
(ii) Correlation relations (between pairs of instances)
Thus, in our language, a knowledge base IKB of instances is a quadruple <IKB_Elements, IKB_Instance_Of, IKB_COR, IKB_Constraints>, where the first term represents the set of instances.

3.2 High-level language for classes of events (CTL)

As regards qualitative and quantitative temporal constraints between classes, we basically retain the simple high-level constraint language in ITL. Thus, our language provides primitives in order to describe (possibly imprecise) dates, durations, and delays, as well as *continuous pointizable qualitative temporal constraints* [Vilain et al., 90]. Notice, however, that the semantics of such constraints is different depending on whether they apply to instances or to classes (see [Terenziani, 02]).

As required by many practical application areas (see, e.g., the clinical guideline example in Section 2.3), we propose a quite expressive high-level language in order to represent the constraints on the repetition of events, and on their periodicity. In our approach (which is also based on the results of the linguistic analysis by Van Eynde [87]), we deal with different components in the definitions of repeated actions, in order to deal also with complex cases such as Ex.1 (see Section 2.3) and Ex.2:

(Ex.2) *for six months, perform action A twice each five days for twenty days, and then suspend for ten days (and so on).*

Thus we can deal with *frame times* ("*the interval which contain all the instances of the event*" [VanEynde, 87], e.g., "*for six months*" in Ex.2 – henceforth called **FT** for short). The description of repeated periodic events splits frame times into a sequence of intervals when actions are performed (called *action-times* – **AT** for short; e.g. "*twenty days*" in Ex.2) and "pause" intervals (*delay time* – **DT** for short – e.g., "*ten days*" in Ex.2). In turn, action times are split into *I-times* (**IT** for short; e.g., "*five days*" in Ex.2) where actions are performed (if DT is null, AT coincides with IT). Finally, we call the number of actions in each I-time "*frequency*" (**freq** for short; e.g., two in Ex.2). The above terms are illustrated in Figure 1.

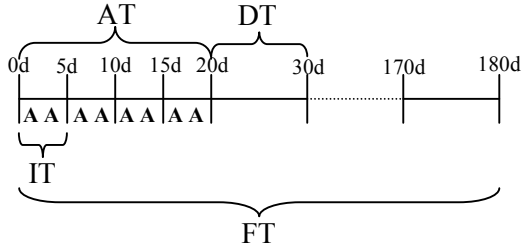

Figure 1: Representation of periodicity for Ex.2. A's represent action instances.

Thus, we introduce in our constraint language the primitive

Repetition_Periodicity(FT,AT,DT,IT,freq,C)

to state the constraints on the repetition of the (class of) event C.

Notice that, since we aimed at designing tractable algorithms to deal with correct and complete consistency checking, we had to impose the constraint that *FT, AT, DT, IT* and *freq* must be specified in an *exact* way.

Finally, in our language, we use the primitive

Part_Of(C1,C)

to state that a class C1 is part of a class C. We will see in the rest of the paper how we cope with the temporal constraints between the composite and the component classes of events induced by the Part_Of relation.

Thus, in our language, a knowledge base CKB of classes is a triple <CKB_Elements, CKB_Part_Of, CKB_Constraints>, where the first term represents the set of classes.

3.3 Internal representation for temporal constraints: the STPs-tree

The high-level constraint language about instances has been deliberately designed in such a way that all constraints can be easily mapped onto bounds on differences, and thus, internally represented as a "standard" STP framework [Dechter et al, 91].

The same point also holds for dates, delays, durations and qualitative temporal constraints between classes of events, that might be stored into a separate STP dealing with the constraints about classes (where each class is represented by a starting and an ending point)[1]. However, such a basic approach must be extended to cope with the temporal constraints about repeated events and with (the temporal constraints involved by) part-of relations.

As regards part-of relations, we represent the temporal constraints they involve as constraints between the starting/ending points of the composite event and the endpoints of its (direct) subactions. In particular, the starting point of a composite event is temporally equal to the starting point of its starting sub-event, while its ending point is after or equal to the ending point of its ending sub-event[2]. Therefore, the constraints

[1] Two different STP's are needed in order to model the fact that, basically, costraints beween instances are "existentially quantified", while constraints between classes are "universally quantified" [Terenziani, 02]).

[2] Unfortunately, if the temporal constraints between the sub-events are such that one cannot find its starting and ending sub-events, the above constraint involves a disjunction, and

involving classes of events (which are *not repeated*) can be homogeneously represented into a unique STP framework, containing a starting and an ending points for each composite and atomic event[3].

However, such an homogeneous approach does not work in the case of repeated/periodic events. In fact, in many real-world application domains (e.g., medical therapies), actions can be repeated a relevant number of times, so that an explicit representation of all repetitions could cause an unnecessary explosion of the number of points in the STP[4]. Such an explosion can be avoided via an *"intensional"* representation, in which a single entity (e.g., a time interval, possibly represented via its endpoints) is used to represent the set of all repetitions of a given event. However, such an intensional approach cannot be directly implemented in "classical" temporal constraint propagation approaches (such as, e.g., STP), which need an explicit (i.e., *"extensional"*) representation of all the temporal entities (time points and/or time intervals). For instance, the temporal constraint that the end of a repeated event E_1 is 1 hour before the starting point of a repeated event E_2 is actually a constraint between the end of the *last repetition* of E_1 and the *first repetition* of E_2. Furthermore, the components of the specification of the periodicity of repeated events (e.g., *AT,DT,IT*) are actually constraints on the single repetitions (rather than on the repeated event as whole). But, in the *intensional* approach, one does not want an explicit representation of each single repetition….

We thus chose to model repeated events as composite events and to represent the constraints regarding repeated actions into separate STP frameworks, one for each reapeated event. Thus, in our approach, the overall set of constraints between classes of events is represented by a ***tree of STP frameworks*** (*STPs-tree* henceforth) [Terenziani et al., 02b]. The root of the tree is the STP which homogeneously represents the constraints between all the classes events (composite and atomic), except repeated events. Each node in the tree is an STP, and has as many children as the number of repeated events it contains. Each arc in the tree connects a pair of endpoints in an STP (the starting and ending point of a repeated event) to the STP containing the constraints between its subactions, and is labeled with a list of properties describing the temporal constraints on the repetitions (i.e., *FT, AT, DT, IT, freq*). For example, in Figure 2, we show the STPs-tree representing temporal constraints involved by the example Ex.1 in Section 2.3.

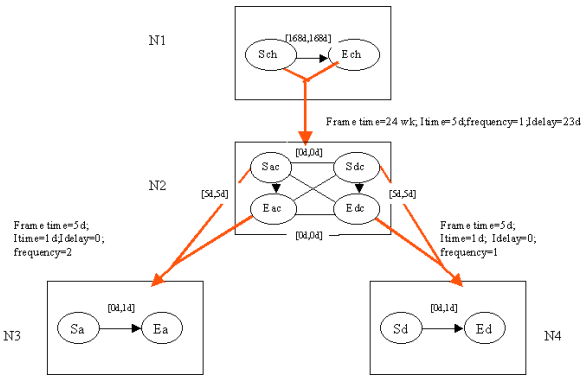

Figure 2: STPs-tree for the multiple mieloma chemotherapy guideline: a naïve graphical representation. Thiny lines and arcs between nodes in a STP represent bound on differences constraints. Arcs from a pair of nodes to a child STP represent repetitions. The overall therapy (node N1) is composed by 6 cycles of 5 days plus a delay of 23 days (for a global frame time of 24 weeks). In each cycle (node N2), two therapies are executed in parallel: Alkeran (node N3: Sa and Ea are the starting and ending nodes), to be repeated twice a day, and Deltacorten (node N4: Sd and Ed are the starting and ending nodes), to be repeated once a day. Arcs between any two nodes X and Y in a STP (say N2) of the STPs-tree are labeled by a pair [n,m] representing the minimal and maximal distance between X and Y.

4 Temporal reasoning with complete knowledge

In the rest of the paper, we describe two temporal reasoning algorithms to check the consistency of a knowledge base of temporal constraints regarding both classes and instances of events. For the sake of clarity, we propose an incremental presentation: in this section we describe our basic algorithm, which assumes full observability of instances even

cannot be modeled in the STP framework (for instance, an approach such as [Galipienso & Barber, 02] would be needed). On the other hand, in the STP framework, one can represent the weaker costraint that all sub-events start after or at the same time than the start of the event they compose, and end before or at the same time of its end.

[3] Alternatively, we could have chosen to partition the constraints about classes into a set of STP frameworks, one for each composite event. Such a clustering technique would save space and, apparently, make the reasoning process more efficient. However, the temporal constraints between the composite events and their components make the constraints in different STPs dependent on each other. Thus, determining the global consistency of such a set of STPs would need a potentially never-ending iteration of the reasoning process, until a state of quiescence is reached.

[4] Such a representation of each repetition would also make the problem of maintaining the coherence of data very expensive, in the case of updates of the knowledge base of constraints.

in the future and total ordering between instances of events (see footnote 5). In section 5 we extend such an algorithm in order to work in domains where there is no observability in the future. The problem of relaxing the assumptions of full observability in the past and of total ordering is briefly investigated in Section 6, and is one of the main goals of our future work. We intend that, whenever an inconsistency is detected, all the algorithms in Figure 3 report it and stop.

The *IntegratedReasoningF* procedure (where "F" stands for *Future instances of events*) performs integrated consistency checking. The basic idea is that of first "inheriting" the temporal constraints from classes to instances, and then performing consistency checking (applying the all-to-all shortest path algorithm) on instances (considering both input and inherited constraints). However, while inheritance from non-repeated classes of events can be dealt with as in [Terenziani, 02], the "inheritance" of constraints from repeated (periodic) events involves first finding the instances of an STPs-tree node that are related to a specific repetition and then adding the constraints of inclusion in the proper I-time (i.e., the IT of such a repetition).

IntegratedReasoningF takes in input the high-level description of the classes and the high-level description of the instances. For the sake of brevity, we assume that all the input instances are correlated. This is not restrictive: since *correlation* partitions instances (and their constraints) into independent sets [Morris et al., 93; Terenziani, 97], the procedure can be iterated on each partition.

In step (1) the procedure performs the translation from classes' high level language (CTL) into internal representation (i.e. an STPs tree as described in section 3.3). In step (2), it checks the consistency of the classes alone (see [Terenziani et al., 02b]). Analogously, step (3) translates the high-level constraints on instances into the corresponding STP framework; in step (4) constraint propagation is performed on the STP about instances alone (using Floyd-Warshall's all-to-all shortest path algorithm). Steps (2) and (4) are also useful to obtain an early detection of inconsistencies.

In step (5) the recursive procedure *InheritF_STP* is invoked on the STPs tree's root. It visits the STPs-tree in a depth-first fashion and performs constraints inheritance for each STPs-tree node. The procedure "marks" the instances of classes it takes into account (including instances of repeated events). Thus, the presence of non-marked instances in step (6) denotes a violation on the number of repetitions (i.e., more repetitions than required have been observed) and is used to report an inconsistency. Finally, in step (7) the procedure performs integrated reasoning at instances level, considering all the constraints inherited from the classes of events, including periodicity constraints.

InheritF_STP is logically divided in two parts: the former (steps (1-19)) performs the inheritance of periodicity constraints, the latter (steps 20-21) performs inheritance of the other temporal constraints (e.g., delays, durations).

Periodicity constraints locate each repetition of a (possibly composite) event into a specific AT and IT. To inherit such constraints, the cue issue is to understand, for each instance, which repetition and thus which I-time it "belongs to". For instance, considering the example Ex.2, the 10^{th} instance of the event A must be located between 30 and 35 days after the beginning of the frame time.

In procedure *InheritF_STP*, the variable *STP_Instances* collects the instances of all the classes of events represented by an STP *X*. In the case of a class representing a repeated event, one repetition at a time is considered. For each class *C1* of *X* (step (2)), if *C1* is not a repeated class, *FindAndMarkFirstInstance* (step (4)) looks for and marks the first unmarked instance of *C1*, i.e. an instance *E1* such that the starting points of all other instances of *C1* are after *E1*'s starting point (and which has not been taken into account yet). Here we use the assumption of total ordering (see footnote 5). If such an instance is not found, an inconsistency is reported.

If *C1* is a repeated class, *Add_Instance* adds a dummy instance of *C1*, that represents the whole sequence of repetitions and is useful to add periodicity constraints related to *C1* (as a matter of fact, one does not have in the IKB an instance of the repeated event as a whole, but just the instances of its individual repetitions). Then, *InheritF_STP* is called recursively on the child node for each repetition (step (12)). This is done via the three-level "*for*" statement in steps (9-10-11): in fact, there are *FT/(AT+DT)* periods in the frame time, and, in each one of the *action times* in such periods, there are *AT/IT I-times*, and each one of them should contain *freq* repetitions. *InheritF_STP* returns in *SubPlanInsts* the instances belonging to a single repetition (notice that repeated actions may be nested). These instances must respect the periodicity constraints (steps (13-15)), i.e. they must be included in their own I-Time. Such a constraint is imposed by adding to STP_IKB_Con the constraint that, for

each instance *subE1* which is part of a repetition, it should start no more than *Offset* after the beginning of the dummy point representing the starting point of the whole repetition, and it should end no more than *Offset+IT* after it (where the $freqi^{th}$ instance of *subE1* in the ITi^{th} I-time in the ATi^{th} action-time has *Offset* equal to *ATi*(AT+DT)+ITi*IT*). Step (16) is used in order to force that the different repetitions within the same I-time cannot overlap with each others.

The inheritance of constraints not involving periodicity is easier. Step (20) inherits monadic constraints, i.e. constraints regarding durations of events (the duration of a repeated class is its frame time). In step (21) binary constraints are inherited, according to the semantics of classes constraints ([Terenziani et al., 02]).

Complexity. *IntegratedReasoningF runs in a time* $O(\max(N^3,M^3))$, *where N and M represent the number of instances in IKB and in CKB respectively.*

Basically, the complexity of inheriting the constraints of periodicity is dominated by step (16), and is $O(N^2)$ at worst. Analogously, the complexity of inheriting the other temporal constraints is $O(N^2)$ at worst (see step 21). Notice that, in all cases, the marking procedure grants that no instance is considered more than once. Thus, the complexity of performing Floyd-Warshall all-to-all shortest path algorithms (see steps 2, 4, and 7 in the procedure IntegratedReasoningF; Floyd-Warshall's algorithm is cubic on the number of nodes) dominates.

Procedure IntegratedReasoningF(<CKB_Elements, CKB_Part_Of, CKB_Constraints>, <IKB_Elements, IKB_Instance_of, IKB_COR, IKB_Constraints>**)**

(1) CKB_STPs_tree := Transform(CKB_ Constraints);
(2) CKB_STPs_tree' := Propagate(CKB_STPs_tree);
(3) STP_IKB_Con := Transform(IKB_ Constraints)
(4) IKB_Minimal_Network := All-Shortest-Paths(STP_IKB_Con);
(5) InheritF_STP(Root(CKB_STPs_tree'), <IKB_Elements, IKB_Instance_of, IKB_COR, IKB_Minimal_Network>)
(6) **if** exists in IKB_Elements a non-marked instance **then return** INCONSISTENT
(7) IKB_Minimal_Network := All-Shortest-Paths(IKB_Minimal_Network)

Procedure InheritF_STP(X=<CKB_Elements, STP_CKB_Con>, <IKB_Elements, IKB_Instance_of, IKB_COR, STP_IKB_Con>**)**

(1) STP_Instances := ∅
(2) **forall** C1 ∈ CKB_ Elements **do**
(3) **if** Non-Repeated(C1) **then begin**
(4) E1:= FindAndMarkFirstInstance(C1, IKB_Elements, IKB_Instance_of)
(5) **if** E1 = NULL **then** return INCONSISTENT
end
(6) **else** /* C1 is repeated */ **begin**
(7) E1 := Add_Instance(C1, IKB_Elements, IKB_Instance_of)
(8) - let FT, AT, DT, IT and freq the periodicity parameters for class C1
(9) **for** 0 ≤ ATi ≤ FT / (AT + DT) - 1 **do**
(10) **for** 0 ≤ ITi ≤ AT / IT - 1 **do**
(11) **for** 0 ≤ freqi ≤ freq - 1 **do**
(12) SubPlanInsts:= InheritF_STP(Child(X,C1), <IKB_Elements, IKB_Instance_of, IKB_COR, STP_IKB_Con>)
(13) Offset := ATi * (AT+DT) + ITi * IT
(14) **forall** subE1 ∈ SubPlanInsts
(15) STP_IKB_Con := STP_IKB_Con ∪ { Offset ≤ Start(subE1) - Start(E1) , End(subE1) - Start(E1) ≤ Offset + IT}
(16) add to STP_IKB_Con the constraints stating that this single repetition must be after previous one
(17) **od od od end**
(18) STP_ Instances := STP_ Instances ∪ { E1 }
(19) **od**
(20) **forall** E1 ∈ STP_ Instances | t ≤ End(C1) - Start(C1) ≤ u ∈ STP_CKB_Con **do**
- let C1 the class corresponding to E1 //i.e., Instance_of(E1, C1) holds
STP_IKB_Con := STP_IKB_Con ∪ {t ≤ End(E1) - Start(E1) ≤ u }
od
(21) **forall** E1, E2 ∈ STP_ Instances, E1 ≠ E2 **do**
- let C1, C2 the classes corresponding to E1 and E2
- instantiate on E1 and E2 the constraints in CKB_Constraints between C1 and C2
od
(22) **return** STP_ Instances

Figure 3. Constraint propagation algorithms for temporal reasoning

5 Temporal reasoning with no future events

In this section we relax the assumption of full observability and we assume that we have no events instances in the future.
For sake of brevity, we describe only the differences between the new procedure *IntegratedReasoning* and the previous version, *IntegratedReasoningF*. We add the step (3a) between steps 3 and 4 of *IntegratedReasoningF*:

(3a) **forall** E1 ∈ IKB_Elements
STP_IKB_Con := STP_IKB_Con ∪ {Start(E1) – NOW ≤ 0}

It states that all observed instances (the istances in *IKB*) must have started before NOW.

The procedure *InheritF_STP* must not report an inconsistency whenever doesn't find a foreseen event in the *IKB*. In fact, let us suppose that the *CKB* predicts the occurrence of an instance *E* of event, and that *E* is not present in *IKB*. This fact is inconsistent only in the case the temporal constraints in *IKB* and *CKB* impose that it must be observed before (or at the same time than) NOW. Otherwise, no inconsistency arises since *E* might be observed (and inserted in *IKB*) in the future.
Because of that, in *InheritF_STP* the predicted and "missing" instances are "hypothesized", collected in the variable *HYPOTH_INSTS* and provisionally inserted in the set of observed instances. This ca be easily done within the *FindAndMarkFirstInstance* procedure, which should introduce hypothesized instances whenever needed (obviously, step (5) of *InheritF_STP* must be removed from the new version). Constraints inheritance is applied to both "real" and "hypothesized" instances and constraint propagation is performed on both types of instances. Finally, the procedure checks whether "hypothesized" instances should *necessarily* occure before NOW. Only in such a case an inconsistency is reported.
Therefore we add at the end of *IntegratedReasoningF* the following steps:

(8) **forall** E1 ∈ HYPOTH_INSTS
if NEC(IKB_Minimal_Network, Start(E1) – NOW ≤ 0) **then return** INCONSISTENT

Notice that, in the case of repeated classes, it is not necessary to hypothesize all the repetitions of a class, but only the first repetition that includes missing instances: in fact, in the case these instances should occur *after* NOW, also the following repetitions should do and, anyhow, we have no inconsistencies. On the other hand, if these instances should occur *before* NOW, the first "non-complete" repetition is enough to detect the inconsistency.
Complexity. The step (3a) requires a time which is linear in the number of instances. As regards step (8), exploiting the locality properties of STP constraints proved in [Brusoni et al., 95], it can be done in a time linear in the number of instances in *HYPOTH_INSTS* (which is, at most, O(M), since we at most hypothesize one instance for each class).

6 Conclusions and Developments

In this paper, we describe a tractable temporal manager (TEMPORE) coping in an integrated way with different temporal aspects involved in real-world temporal applications. In particular, TEMPORE offers an expressive language for explicitly representing *part-of* and *instance-of* relations between events, as well as (a limited set of qualitative and quantitative) *temporal constraints* between *classes* and *instances* of events, including *periodic repetitions*. Moreover, it supports tractable temporal reasoning for consistency checking. It can also be conjectured that our temporal reasoning algorithms are *complete*, in the sense that the inheritance procedures add all the bounds-on-differences mappings of the temporal constraints between classes on the proper instances, and then, complete temporal reasoning on instances is performed via Floyd-Warshall algorithm (which is complete for STP problems [Dechter et al., 91]).

6.1 Applications

The requirement of coping with the temporal constraints rising from so many different aspects of reality (ranging from part-of and instance-of relations to standard qualitative and quantitative temporal constraints and to periodicity) emerged from our previous work in different application fields (including train scheduling [Brusoni et al., 96] and multimedia presentation [Adali et al., 00]), and, in particular, in clinical guideline management. Since 1997, we work in a joint project with Azienda Ospedaliera S. Giovanni Battista of Turin (one of the main hospitals in Italy) for the design and the development of GLARE (GuideLine Acquisition, Representation and Execution) [Terenziani et al., 01, 02a, 02b], a system which acquires and represents clinical guidelines, and then executes

them (in a semi-automatic way) on specific patients. In such an application, all the expressiveness of TEMPORE is needed. In fact, clinical guidelines describe general procedures (i.e., procedures about classes of events), and are structured via part-of relations. Many actions in the guidelines (expecially in the therapeutic part of guidelines) must be repeated periodically. Temporal constraints (e.g., delays) are used in order to constrain the execution of actions in the guidelines. Moreover, at execution time, one has to check whether the physician respected the procedure, including the temporal constraints it conveyes (thus, integrated classes+instances temporal reasoning is needed). Finally, *full observability* in the past and *total ordering* on correlated instances are reasonable assumptions considering the execution of clinical guidelines on hospitalized patients (notice that, in our application domain, instances are exactly located in time).

6.2 Related Works

To the best of our knowledge, there is no other *constraint-propagation-based* approach which offers explicit support in order to deal with the temporal constraints raising from all the different issues we considered in this paper. In the following, we mention some of the constraint-based temporal reasoning approaches facing some of such issues which seem to us more closely related to our work.

In Morris et al.'s approach [Morris et al., 93, 95, 96] only qualitative constraints between repeated events are dealt with. Repeated events are dealt with as "classes" of events, and different quantifiers are used to relate them. As regards instances of events, Morris et al. introduced the notion of *consistent scenario* [Morris et al., 95] and used it in a *"predictive"* perspective, sketching an algorithm for *generating* a consistent scenario of instances which are consistent with a knowledge base of temporal constraints between repeated events.

Loganantharaj mainly faced the problem of associating possibilistic distributions to qualitative temporal constraints between periodic events [Loganantharaj & Gimbrone, 95] and to metric constraints concerning the durations of events, which are also expressed using transition rules [Loganantharaj & Gimbrone, 97; Loganantharaj & Kurkovsky, 97]. Such constraints are used in a "predictive" way: temporal reasoning is used for projecting the constraints on the durations in the future using the current domain information.

In our previous work, we have defined an high-level language to deal with period-dependent qualitative temporal constraints between repeated events (which are coped with as "classes" of events) and constraint propagation algorithms operating on them [Terenziani, 95,97,03]. We also proposed an initial algorithm to check the consistency of a knowledge base of such constraints with a set of instances of events exactly located in time [Terenziani,00]. More recently, we started to approach the problem of integrating both qualitative and quantitative constraints between classes and instances of events in a more general perspective [Terenziani, 02], and proposed an approach to cope with qualitative and quantitative and periodic repetition constraints between classes of events in clinical guidelines [Terenziani et al., 02a, 02b].

The work in this paper is an extension and integration of the works in [Terenziani, 02, Terenziani et al., 02a, 02b] towards the general goal described in the introduction. In particular, the treatment of part-of relations, and of the "inheritance" of the periodicity constraints from classes of events to the corresponding instances (and of the interplay between part-of and periodicity – see e.g. the comments to Ex.1) are entirely new contributions of this paper.

6.3 Future work

Although we believe that our approach is significantly more expressive than related approaches in the literature, it is currently based on some limitating assumptions, some of which we aim to remove in our future work. However, we want to stress that most of such assumptions are needed in order to retain the tractability of the reasoning process, and are reasonable in real application domains (including the clinical guideline one: see section 6.1). We thus envision an extension of our approach along two mainstreams:

(1) "conservative" extensions (i.e., extension which still preserve the tractability of our approach)
(2) "general" extensions, with the loss of tractability

As concerns "conservative" extensions, we are currently trying to extend our approach to cope also with repetitions based on conditions (e.g., repeat therapy X until condition Y holds). The treatment of conditioned repetitions is not critical at the level of classes of events (see [Terenziani et al., 02b]), but its interplay with the checking of the consistency of instances wrt classes constraints, considering the predictive role of classes constraints, deserves

specific attention. Furthermore, we are also planning to apply the "local reasoning" properties we proved in [Brusoni et al., 95; Console & Terenziani, 99] to support efficient query aswering in TEMPORE.

As regards "general" extensions, the tractability of our current approach strongly relies on the assumptions that instances are totally ordered[5] and of "full observability" (i.e., if an instance is not explicitly present in the knowledge base, it didn't occur). Although both assumptions are reasonable in the clinical guideline context (which constitute our main application area), we aim to develop a domain-independent approach covering also domains where such assumptions cannot be made. Roughly speaking, the lack of total ordering between instances and the lack of complete observability have on the reasoning mechanism basically the same effect: associating instances to the corresponding repetitions becomes a combinatorial problem[6]. In this perspective, we look at our current approach as the "core" of a generalised approach in which extensive backtracking (or backjumping) techniques are used to cope with the combinatorial explosion, and we are investigating how and when temporal constraints in the knowledge base can be used in order to limit the explosion (e.g., with an early pruning of temporally inconsistent alternatives) and/or to optimize the backtracking algorithm.

References

[Adali et al., 00] S. Adali, L. Console, M.L. Sapino, M. Schenone, P. Terenziani. Representing and reasoning with temporal constraints in multimedia presentations. Proc. TIME 2000, Best Paper, Cape Breton, Canada, IEEE Press, 3-11, 00.

[Allen,91] 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.

[Brusoni et al., 95] V. Brusoni, L. Console, and P. Terenziani. "On the computational complexity of querying bounds on differences constraints", *Artificial Intelligence 74(2)*:367-379, 1995.

[Brusoni et al., 96] V. Brusoni, L. Console, E. Lamma, P. Mello, M. Milano, P. Terenziani. Resource-based vs. task-based approaches for scheduling problems. In Proc. 9th International Symposium on Methodologies for Intelligent Systems, Zakopane, Poland, 1996.

[Console & Terenziani, 99] L. Console, P. Terenziani. Efficient processing of queries and assertions about qualitative and quantitative temporal constraints. Computational Intelligence 15(4), 442-465, November 1999.

[Dechter et al., 91] R. Dechter, I. Meiri, J. Pearl, "Temporal Constraint Networks", *Artificial Intelligence* 49, 61-95, 1991.

[Galipienso & Barber, 02] M. I. A. Galipienso and F. Barber Sanchis, Representing and Reasoning with Disjunctive Temporal Constraints, Proc. Ninth International Symposium on Temporal Representation and reasoning (IEEE Press) 46-48.

[Loganantharaj & Gimbrone, 95] R. Loganantharaj, and S. Gimbrone. "Probabilistic Approach for Representing and Reasoning with Repetitive Events", In Proceedings *second International Workshop on Temporal Representation and Reasoning (TIME'95),* Melbourne, FL, pp. 26-30, 1995.

[Loganantharaj & Kurkovsky, 97] R. Loganantharaj, and S. Kurkovsky. "A new model for projecting temporal distance using fuzzy temporal constraints", In Proceedings I*EA/AIE'97*, 1997.

[Loganantharaj & Gimbrone, 97] R. Loganantharaj, and S. Gimbrone. "Issues on Synchronizing when Propagating Temporal Constraints", In *National Conference on Artificial Intelligence Workshop on Spatial and Temporal Reeasoning*, 1997.

[Morris et al., 93] 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.

[5] Actually, total ordering is only required considering correlated instances of the same class. We also suppose that two such instances cannot occur exactly at the same time. Notice that our current approach covers also the case when ordering emerges from the initial propagation of the instances constraints – without considering classes constraints.

[6] For example, as an an extreme case, given a specific observed instance *I* of a repeated event *E*, if we have no temporal information about when *I* occurred, we have no means to understand if I is the first, the second, etc. repetition of *E*, and we have to cope with all possible cases. Analogously, if *E* must be repeated *x* times, and we have just *x-1* (or, even worst, *x-i*) instances of *E* (not ordered in time), and we do not have full observability, we have to cover all possible alternatives (i.e, the missing instance may correspond to the first repetition, to the second, etc.).

[Morris et al., 95] R.A. Morris, L. Khatib, and G. Ligozat. "Generating Scenarios from specifications of Repeating Events", In Proceedings *second International Workshop on Temporal Representation and Reasoning (TIME'95)*, Melbourne, FL, pp. 41-48, 1995.

[Morris et al., 96] R.A. Morris, W.D. Shoaff, and L. Khatib. "Domain Independent Temporal Reasoning with Recurring Events", *Computational Intelligence* **12**(3):450-477, 1996.

[Terenziani, 95] P. Terenziani. Reasoning about Periodic Events. In Proc. TIME-95 International Workshop on Temporal Representation and Reasoning, pages 137-144, Melbourne, Florida, 1995.

[Terenziani, 97] 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.

[Terenziani, 00] P. Terenziani. Integrated Temporal Reasoning with Periodic Events. Computational Intelligence 16(2), 210-256, May 2000.

[Terenziani et al., 01] P. Terenziani, G. Molino, and M. Torchio, "A modular approach for representing and executing clinical guidelines", *Artificial Intelligence in Medicine 23*, 249-276, 2001.

[Terenziani, 02] P. Terenziani. Temporal Reasoning with classes and instances of events. Proc. *TIME* 2002, Manchester, UK, IEEE Press, 100-107, 2002.

[Terenziani et al., 02a] P. Terenziani, S. Montani, C. Carlini, G. Molino, M. Torchio. Supporting physicians in taking decisions in Clinical Guidelines: the GLARE's "what if" facility. *Journal of the American Association of Medical Informatics (JAMIA)*, Proc. Annual Fall Symposium, 2002.

[Terenziani et al., 02b] P. Terenziani, C. Carlini, S. Montani. Towards a Comprehensive Treatment of Temporal Constraints in Clinical Guidelines. Proc. *TIME* 2002, Manchester, UK, IEEE Press, 20-27, 2002.

[Terenziani, 03] P. Terenziani. Towards a Comprehensive Treatment of Temporal Constraints about Periodic Events. Accepted for pubblication in International Journal of Intelligent Systems.

[VanEynde, 87] F. Van Eynde. "Iteration, Habituality and Verb Form Semantics", In Proceedings *third Conference of European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark, pp. 270-277, 1987.

[Vila, 94] L.Vila, "A Survey on Temporal Reasoning in Artificial Intelligence", *AI Communications 7(1)*, 4-28, 1994.

[Vilain et al., 90] M. Vilain, H. Kautz, and P. VanBeek. "Constraint Propagation Algorithms for temporal reasoning: a Revised Report", D.S. Weld, J. deKleer, eds., *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, pp. 373-381, 1990.