Spatial and Temporal Reasoning

edited by

Oliviero Stock
IRST, Italy

KLUWER ACADEMIC PUBLISHERS
DORDRECHT / BOSTON / LONDON

2. Reasoning about Time and Actions in Artificial Intelligence: Major Issues

Alfonso Gerevini

2.1. Introduction

The ability to represent and manage temporal knowledge about the world is fundamental in humans as well as in artificial agents. Some examples of important "intelligent" activities that require time representation and reasoning include the following:

- Given that certain conditions, actions, and events occurred in the world prior to a time t, *predict* whether certain conditions will hold at t or whether certain actions can be executed or certain events will occur at or after t;
- Hypothesize conditions, actions, or events prior to a certain time t to *explain* why certain conditions, actions, or events occurred at or after t;
- Given a description of the world at a certain time t, *plan* a set of actions that can be executed in a certain order starting at t, to achieve a desired goal;
- *Schedule* a set of given activities to meet some constraints imposed on the order, duration, and temporal position or separation of the activities, such as a stipulated deadline;
- Given (possibly incomplete or uncertain) information about the temporal relations holding between events or facts in the represented domain, *answer temporal queries* about other implicit (entailed) relations; for example, queries about the possibility or the necessity that two particular future events will temporally overlap or about the shortest temporal distance separating two events that have occurred.

Temporal reasoning has been a major research field in artificial intelligence (AI) since the beginning of this discipline, and it has been investigated in the context of various AI areas where the tasks mentioned above are crucial. Such areas include knowledge representation (see, for example, Schmiedel, 1990; Miller, 1990; Schubert and Hwang, 1989; Artale and Franconi, 1994), natural language understanding (see Allen, 1984; Miller and Schubert, 1990; Song and Cohen, 1988), commonsense reasoning (for a survey see Sandewall and Shoham, 1995), qualitative reasoning (for a survey see Dague and MQD Group, 1995), diagnostic reasoning (e.g., Console et al., 1991; Nökel, 1991), plan generation (see Allen, 1991; Dean et al., 1988; Tsang, 1986; Vere, 1983), plan recognition (see Kautz, 1987, 1991; Weida and Litman, 1992; Song and Cohen, 1996), and scheduling (see Rit, 1986; Boddy, 1993; Frederking and Muscettola, 1992).

Since the field of temporal reasoning has been a central research topic in AI for many years, providing an exhaustive survey of all the results and proposed approaches would probably require an entire book by itself. The much less ambitious goal for this chapter is to serve as a base for orienting the reader in the field of temporal reasoning and as general background for further specific reading.

The first part of the chapter, after a short discussion on time models, introduces basic issues characterizing two major subfields of temporal reasoning: reasoning about time and reasoning about actions and change. Then the essential role of temporal reasoning in the important AI area of planning is discussed. Finally, the last part of the chapter focuses on reasoning about time, providing an overview of the main results.[1]

Note that the chapter mainly addresses reasoning issues, and says little about how temporal information is represented in a general language for knowledge representation. Also, the coverage of reasoning about time uses very restricted representations, where only disjunctions of basic relations between free variables are permitted. Syntax and semantics of more expressive sentential or quantificational temporal logics are not addresses. (The reader interested in first-order logics with temporal arguments, reified temporal logics, and temporal modal logic might consult the survey by Vila, 1994.)

2.2. Modeling Time

In order to build a time representation some ontological issues need to be considered.[2] For example, we have to choose the primitive *time unit*, which usually is either points or intervals.[3] Theoretically, this choice is not crucial, since it is possible to build a point-based theory of time that allows us to represent intervals in terms of points, or vice versa (see, for example, Galton, 1995b). However, the choice can be practically important. For instance, in

building temporal reasoning system, a point-based representation supports efficient implementations of basic temporal database management (Dean and McDermott, 1987; Stillman et al., 1993; Gerevini et al., 1995; Barber, 1993).

Another fundamental issue concerns the topology of time. For example, we can model the direction of time by imposing on the basic time units either a total order or a partial order. In the former case we have a *linear-time* model, where for each pair of different time units one precedes the other. In the later case we have a *branching-time* model with multiple future times (or past times) lying on different time lines.

A time model can be either *discrete* (such as the integers), implying that given a pair of ordered time units there exists at most a finite number of time units lying between them, or it can be *dense* (such as the real numbers), implying that for each pair of ordered time units there exists a third unit lying between them. A time model is *bounded* when there exists a lower bound on times (such as the natural numbers), or there exists an upper bound on times (such as the negative integers). By contrast, a time model is *unbounded* when there exists no lower or upper bound on times (such as the integers).

Finally, another important issue in modeling time is *granularity* (see, for example, Hobbs, 1985; Leban et al., 1986; Ladkin, 1987a; Dean, 1989; Euzenat, 1995; Ciapessoni et al., 1993; Wang et al., 1995). Time units can be grouped into clusters of the same size (for example, seconds, minutes, hours) where these clusters are hierarchically organized to form "layers of time" with different granularities (seconds can be grouped into clusters of sixty elements to form a layer of minutes). This organization can be exploited in the reasoning process because it can allow us to abstract away details that may be irrelevant at a certain level of conceptualization.

Thus we have a variety of options in modeling time. From our perspective the "best" model to adopt in a AI application is mainly a practical issue, depending on the particular domain and application, rather than a philosophical or physical issue concerned with modeling what time actually is in the real world.

2.3. Reasoning About Time

In reasoning about time, temporal knowledge is typically specified in terms of a collection of qualitative or metric relations constraining time points or intervals. The main reasoning tasks are concerned with determining the consistency of this collection, deducing new relations from those that are given, and deriving an interpretation of the point (interval) variables involved that satisfy the constraints imposed by the given relations. For example, consider the following story in an imaginary trains transportation

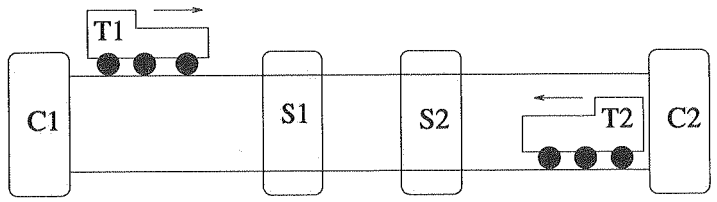

Figure 1. Pictorial description of the trains example

domain (see Figure 1):

> During its travel from city C1 to city C2, train T1 stopped first at the station S1 and then at station S2. Train T2 traveled in the opposite direction of T1 (i.e., from C2 to C1). During its trip T2 stopped first at S2 and then at S1. When T2 arrived at S1, T1 was stopping there too.

This simple story contains some explicit qualitative information about the ordering of the time intervals during which the events described occurred. In particular, from the first sentence we can infer that the intervals of time during which T1 stopped at S1 (at(T1,S1)) and at S2 (at(T1,S2)) are *contained* in the interval of time during which T1 traveled from C1 to C2 travel(T1,C1,C2); from the second sentence we can infer that at(T1,S1) is *before* at(T1,S2); from the third and the fourth sentences we can infer that at(T2,S1) and at(T2,S2) are *during* travel(T2,C2,C1) and that at(T2,S2) is *before* at(T2,S1); finally, from the the last sentence we can infer that the starting time of at(T1,S1) *precedes* the starting time of at(T2,S1) and that the starting time of at(T2,S1) *precedes* the end time of at(T1,S1).

Suppose that in addition we know that no more than one train can stop at S2 at the same time (say, because it is a very small station). Then we have that at(T2,S2) is *before or after* at(T1,S2). In performing the temporal reasoning tasks mentioned above in the context of the temporal information provided in the above story, we would determine that the story is temporally consistent. We would deduce new relations such as that travel(T1,C1,C2) intersects travel(T2,C2,C1), which are implicit in the story; we would strengthen explicit relations (e.g., we deduce that at(T2,S2) must be before at(T1,S2), ruling out the possibility that at(T2,S2) is after at(T1,S2)); and finally we would determine that the ordering of intervals in Figure 2 is consistent with all the (implicit and explicit) temporal relations in the story.

Note that if the supplementary information that T1 stopped at S2 *before* T2 were provided, then we would determine that the story is temporally

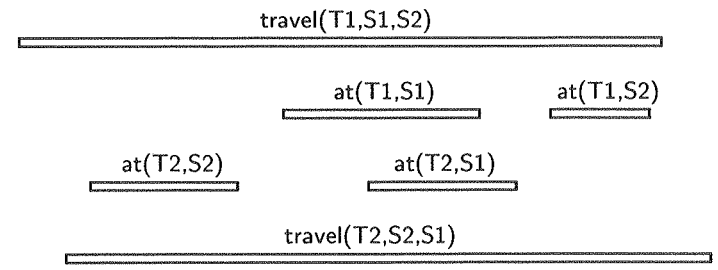

Figure 2. A consistent ordering (scenario) of the time intervals in the trains example

*in*consistent. This is because the explicit temporal relations in the story imply that T2 left S2 before T1 left S1, which precedes the arrival of T1 at S2.

In many applications useful temporal knowledge is also available in the form of quantitative constraints such as durations, deadlines, and temporal distances between starting times or end times of events. For example, our simple transportation domain could provide the following additional quantitative information imposing duration constraints on the travel times and stopping times of the trains:[4]

> The durations of travel(T1,C1,C2) and of travel(T2,C2,C1) should be less than forty-eight hours; the trip of T1 (T2) from C1 to S1 (from S1 to C1) takes six to eight hours; the trip of T1 (T2) from S1 to S2 (from S2 to S1) takes eight to twelve hours; the trip of T1 (T2) from S2 to C2 (from C2 to S2) takes fourteen to twenty hours; both T1 and T2 stop at each station for at least two hours.

Furthermore, suppose that our story is extended with the following information about metric times:

> T1 left C1 at midnight on December 11. It stopped at S1 for six hours, and then it arrived at C2 at 4:00 p.m. on December 12. Regarding T2's trip, we have that T2 left C2 at 11:00 a.m. on December 10; that it stopped three hours at S2 and 4 hours at S1 and that it took nine hours to reach C1 from S2.

The resulting story together with the temporal information provided by the domain give rise to a rich set of qualitative as well as quantitative temporal relations between points and intervals. Such constraints can be represented in a knowledge base, and temporal reasoning is required in order to answer queries of the following form:

> Is the temporal information provided consistent ?
> What are the possible arrival times of T2 at C1 ?
> How long did it take for T1 to go from S2 to C2 ?

Did T2 arrive at C2 before T1 arrived at C1 ?
Is it possible that on Dec. 11 at 2:00 p.m. T2 had not arrived at S1 yet ?

Important issues in the research on reasoning about time concern the computational complexity of reasoning relative to the class of relations that can be expressed, the development of efficient representations and algorithms, and the implementation of domain independent systems that can be used as specialized temporal reasoners in larger AI systems (see, for example, Dean and McDermott, 1987; Koomen, 1988; Gerevini et al., 1995). Section 2.6 gives an overview of the main results obtained in this subfield of temporal reasoning.

2.4. Reasoning About Actions and Change

Formal methods for reasoning about change are used in AI to model the dynamics of the world (domain) represented, where changes can be determined by the effects of actions performed by an agent or of (possibly) unforeseen "external" events. Central reasoning tasks involve prediction and retrodiction of certain properties of the world represented.

More precisely, given a set of axioms formalizing the effects of the actions in a particular domain and the relevant preconditions (a set of *action laws*), an ordered set of actions (*a schedule*), a set of properties holding before the actions are executed (an *initial state*), and possibly a set of *observations* indicating that certain conditions hold (or do not hold) at particular times after the initial state, the prediction task regards determining whether certain conditions hold at a certain time after the execution of the actions in the schedule (in the *final state*). Similarly, a retrodiction task has a set of action laws, a schedule, a final state, and a possibly empty set of observations, and we have to determine whether certain conditions held (or did not hold) at certain times before the actions of the schedule are executed. This formulation of the prediction and retrodiction tasks can be extended by adding possible external events (see, for example, Allen and Ferguson, 1994).

2.4.1. THE SITUATION CALCULUS

One of the earliest and most widely used formalisms for reasoning about actions and change is the *situation calculus* (McCarthy, 1963, 1968; McCarthy and Hayes, 1969). In the situation calculus (SC) knowledge about the changing world is organized into a potentially infinite number of *situations*, each of which corresponds to one possible way that the world could be at a given time and is described by a collection of "facts" (McCarthy and Hayes, 1969, p.477):

Knowledge of the changing world ⊃ infinite # of situations ⊃ a collection of "facts"
(Since it is incomplete)

> A situation s is the complete state of the universe at an instant of time. Since the universe is too large for complete description, we shall never completely describe a situation; we shall only give facts about situations. These facts will be used to deduce further facts about that situation, about future situations and about situations that persons can bring about from that situation.

Situations are temporally represented as time points in a discrete branching (partially ordered) time model, and the performance of an action can change some properties (*fluents*) in the current situation, determining a transition to a new situation. Fluents can be seen as functions defined on situations whose value (either `true` or `false` for *propositional fluents* and a situation for a *situational fluents*) is changed by the execution of an action or by the occurrence of an external event.

For example, in the context of the ever-popular *blocks world*, consider performing the action of moving block `A` on top of block `B` (`Puton(A,B)`) in a situation S1 where the blocks `A` and `B` do not have any block on the top of them (that is, where `clear(A)` and `clear(B)` are `true`, and `on(A,B)` is `false`). As a result of the execution of this action we transit from S1 to a new situation S2 in which `clear(B)` is `false` and `on(A,B)` is `true`, provided that the action `Puton` has been appropriately axiomatized. More formally, this can be represented by asserting

(1) $\texttt{Holds(clear(x),s)} \wedge \texttt{Holds(clear(y),s)} \wedge \texttt{x} \neq \texttt{y}$
$\rightarrow \texttt{Holds(on(x,y),Result(Puton(x,y),s))} \wedge$
$\neg\texttt{Holds(clear(y),Result(Puton(x,y),s))}$

(2) $\texttt{Holds(clear(A),S1)} \wedge \texttt{Holds(clear(B),S2)} \wedge \neg\texttt{Holds(on(A,B),S1)}$

(3) $\texttt{S2} = \texttt{Result(Puton(A,B),S1)}$,

from which we can derive

(4) $\texttt{Holds(on(A,B),S2)} \wedge \neg\texttt{Holds(clear(B),S2)}$,

where `Result` is a situational fluent mapping an action and a situation to another situation; `Holds(p,S)` is a fluent predicate indicating that the property `p` is true in the situation `S`; `x,y` and `s` are universally quantified variables of the appropriate sort.[5]

SC can be seen as a point-based temporal logic with a branching time model, where time is implicitly represented by the situations. In some alternative formalisms time it explicitly represented, and change is implicitly represented by asserting that a property holds at (during) a particular point (interval) of time, while it does not hold at another. In these

formalisms knowledge about actions and properties of the represented world is formalized through various temporal logics based on different time models (see, for example, Allen, 1984; Allen and Ferguson, 1994; McDermott, 1982; Shoham, 1987; Galton, 1990; van Benthem, 1991).

Allen and Ferguson (1994) (see their chapter in this book, Chapter 7) discuss the need for explicit temporal logics to develop general representations that support a wide range of reasoning tasks including prediction, retrodiction, and planning and that can model complex realistic scenarios where actions and events take time, external unforeseen events may interact with planned actions, actions may overlap or occur simultaneously, and the knowledge of the world is incomplete and unpredictable in detail.

On the other hand, some researchers rely on SC as a theoretical and computational foundation for the ambitious ultimate goal of modeling autonomous agents acting in complex dynamic environments. Several extensions to the original SC have been proposed to accommodate in this framework some important features of reasoning about action and change. For example, some very interesting extensions have recently been proposed by Pinto (1994) and Reiter (1996), who enriched SC with concurrent actions, external "natural actions" that occur in response to known laws of physics, and continuous time.

2.4.2. THE FRAME, RAMIFICATION, AND QUALIFICATION PROBLEMS

The central problems in formalizing reasoning about actions and change are the so-called *frame problem*, *ramification problem*, and *qualification problem*. Shoham and McDermott (1988) describe these problems as intrinsic problems that are independent of the particular formalism used (either change-based or time-based). Here we choose to briefly illustrate them in the context of SC, despite the limited expressiveness of state-based approaches compared to time-based approaches argued by several researchers.

The frame problem was first described by McCarthy and Hayes (1969). Given a situation S in which an action A is performed, it concerns the formalization of those fluents that do *not* change value in the situation `Result(A,S)` relative to their original value in S. For example, suppose that in the previous simple blocks world we have an additional clear block `C`, and that in the initial situation S1 each block has a color associated with it, that is,

$$\texttt{Holds(color(A,blue),S1)} \wedge \texttt{Holds(color(B,red),S1)} \wedge \\ \wedge \texttt{Holds(color(C,blue),S1)}.$$

Suppose also that the agent has the task of building a stack formed by at least two blocks of the same color. This can be accomplished by performing

in S1 either `Puton(A,C)` or `Puton(C,A)`. However, that these actions result in the desired final state cannot be formally proved because logically nothing ensures that while we move a block on top of another its color does not change. In particular, we cannot prove that `color(A,blue)` and `color(C,blue)` *persist* either if we perform `Puton(A,C)` or `Puton(C,A)`.

In order to guarantee the persistence of the fluents unaffected by an action, McCarthy and Hayes proposed introducing a set of axioms that explicitly list the fluents that do not change their value when the action is performed. Such axioms are called *frame axioms*. The major problem with the frame axioms is that they can be quite numerous ($O(m \cdot n)$ for m fluents and n possible actions in the represented world), making the formalization tedious to specify and the (automated) reasoning computationally expensive. In general, the frame problem is the difficulty of formalizing the properties of the world that are unaffected by the performance of an action, without explicitly enumerating *all* such properties.

The ramification problem regards formalizing the effects of an action. In particular, in the situation calculus the ramification problem is the difficulty of formally specifying *all* changes in fluents to be expected in the resultant situation when an action is performed. This can be a problem because an action can have arbitrarily complex effects, since a particular change can give rise to further changes and so on, creating an arbitrarily long chain of changes.

For example, consider the action of driving a car from a location `loc1` to another location `loc2`.[6] An obvious effect of this action is that the driven car is at `loc2`, and a consequence of this effect is that all the parts forming the car (the engine, the tires, and so on) change their original location from `loc1` to `loc2`. In formalizing the driving action, instead of enumerating the changes of location for all the object forming the car, it would be more natural to be able to automatically derive such changes from the ("primary") effect that the car is in a new location. This requires solving the ramification problem, which in general is to avoid exhaustively enumerating as effects of an action *all* the changes determined by performing the action.

The qualification problem was identified in (McCarthy and Hayes, 1969; McCarthy, 1977) and concerns the formalization of the preconditions of an action. In fact, depending on the context in which an action is formalized, specifying the conditions under which an action is executable may require consideration of arbitrarily many cases. For example, the previous action of driving a car has many potential preconditions, such as that the battery is not dead, the gas is not empty, the tail pipe is not clogged up by anything, the engine works, the wheels have not been stolen, and so on. In general, the qualification problem is to avoid exhaustively enumerating all the preconditions of an action. This problem is related to the difficulty of

making appropriate assumptions about what are the *relevant* preconditions for the formalization of an action in a specific domain.

A general concern of the research in reasoning about actions and change is the investigation of formalisms and inference methods based on non-monotonic logics (for a survey see Sandewall and Shoham, 1995), monotonic logics (see Schubert, 1990, 1994; Reiter, 1991; Morley et al., 1994), or probabilistic methods (see Pearl, 1988a; Dean and Kanazawa, 1989; Kanazawa, 1991; Dean and M. Wellman, 1991; Hanks, 1990a; Hanks and McDermott, 1994; Haddawy, 1996; McDermott's chapter in this book, Chapter 8) in which prediction, retrodiction, and other related reasoning tasks can be formalized without succumbing to the basic problems outlined above.

Two main methodologies have been followed: an *example-driven* methodology, which most of the researchers adopted, and a more recent *systematic* methodology.[7] Sandewall (1994a, 1994b) discusses the merits and the limits of the first, and proposes and uses the second as a methodology whereby one can obtain precise results on the *range of correct applicability* of several previously proposed theories of action and change, as well as new ones.

Various nonmonotonic logics for reasoning about actions and change were originally proposed in the context of the example-driven methodology, where the validation of the techniques proposed is usually restricted to the use of few representative examples, such as the famous *Yale shooting problem* (Hanks and McDermott, 1987). In this simple problem we have two possible actions, `Wait` and `Shoot`, and two fluents representing the properties that a gun is loaded (`Loaded`) and that the target is alive (`Alive`). In the initial situation `S0` both `Loaded` and `Alive` hold. The problem is to derive the "intended" conclusion that if `Wait` and `Shoot` are performed in succession starting from `S0`, then in the resulting situation `Alive` will not hold. The difficulty of this example arises from the fact that the formalization of `Wait` has no effects, and the effect $\neg$`Alive` of `Shoot` is conditioned to the fact that `Loaded` holds in the situation where the action is performed. Hence, logically it is possible that in the situation `Result(Shoot,Result(Wait,S0))` `Alive` still holds because during waiting the gun (spontaneously) became unloaded, that is, we have $\neg$`Hold(loaded,Result(Wait,S0))`.

The main distinguishing feature of the nonmonotonic logics proposed in the literature is the entailment method for selecting a more restricted subset of models (or a larger set of consequences) than those derivable in a classical logical framework. These models are selected according to some *preference relation*, and are designed to correspond to the set of models intended by a reasoner with common sense. Some prominent examples of these entailment methods are *circumscription* (see McCarthy, 1980, 1986; Hanks and McDermott, 1987; Baker, 1989; Kartha and Lifschitz, 1995), *chronological minimization* (see Kautz, 1986; Lifschitz, 1987b; Shoham, 1988a; Sandewall, 1989; Lin and Shoham, 1991), *causal minimization* (see Haugh, 1987; Lifschitz, 1987a), *nonnormal defaults* and *autoepistemic logic* (see, Morris, 1988; Gelfond, 1989).[8]

An alternative approach to nonmonotonic logics that relies on the use of monotonic logics is *explanation closure* (Haas, 1987; Schubert, 1990, 1994; Reiter, 1991). We discuss this approach in the next section as an efficient method for dealing with the frame problem in planning.

The systematic methodology relies on a semantics that is used to define what the intended models are and to characterize scenario descriptions in terms of a taxonomy of (ontological and epistemological) properties that they may have. The range of correct applicability of a logic of actions and change (the ability to correctly infer change and nonchange) can then be identified in terms of classes of scenarios that can be handled by the logic. Sandewall (1994a, 1994b) evaluated several logics in the context of the systematic approach, using an underlying "inertial" semantics according to which the only changes that occur in a world are those known to be directly induced by an agent's actions, except for some limited cases of indirect changes.

More recently, Sandewall (see Chapter 9 in this book) extended his assessment results to some approaches for dealing with the ramification problem. For this purpose he proposes and uses an underlying *transition cascade semantics*, according to which "an action is viewed as consisting of an initial state transition which represents the invocation of the action, followed by a succession of other state transitions which may be understood as representing causal chains".

In this section and in the previous one we have given a general picture of the major issues concerning formal reasoning about actions and change, and reasoning about (qualitative or quantitative) temporal information. In the next section we discuss the important role of these subfields of temporal reasoning in the AI area of planning.

2.5. Temporal Reasoning in Planning

Planning is an important and challenging research area of AI in which a large variety of techniques have been proposed in the last three decades (for a review of the field see Tate, 1990; Georgeff, 1990; Allen et al., 1990; AIJ, 1995). Our goal in this section is to briefly introduce some fundamental aspects of domain-independent planning that are closely related to temporal reasoning. Similar issues can arise in domain-dependent planners as well, planners that concentrate on using domain heuristics to attain computational efficiency.

Planning, and especially *well-founded* planning, is a computationally very hard problem (Bylander, 1993, 1994; Erol et al., 1992; Nebel and Bäckström, 1994), which involves both reasoning about actions and change and reasoning about time. The emphasis in well-founded planning is on constructing planners that can be *proved* to have certain desirable properties, such as soundness and completeness for their intended class of problems, or the ability to find optimal or near-optimal solutions. On the other hand, practical planning research seeks to provide planning frameworks and tools that are sufficiently expressive, flexible, and efficient to be effectively usable in applications such as planning robot actions, transportation planning, factory scheduling, genetic engineering, and conversation planning. A considerable effort in current planning research aims at bringing well-founded approaches closer to practicality.

A domain-independent planner needs to cope with the frame problem and to manage the temporal constraints associated with the actions in the plan or with some conditions of the represented world (such as the period of time during which a certain resource can be consumed or an action can be synchronized with an external events). The rest of this section is dedicated to illustrating these issues.

2.5.1. THE FRAME PROBLEM IN PLANNING

STRIPS (Fikes and Nilsson, 1971) is one of the earliest domain-independent planning formalism on which several recent planners are still based. In STRIPS-based planners a state of the world is represented as a set of formulas, and the effects of an action are specified through a list of formulas to be removed from the state in which A is executed (a *delete list*), plus a list of formulas to be added to such a state (an *add list*). The execution of an action in a certain state determines a transition to another state, whose set of formulas is derived by the add list and delete list of the action.

In STRIPS, as well as in modern STRIPS-descendants such as UCPOP (Penberthy and Weld, 1992), the frame problem is "solved" by the adoption of some strong assumptions about the world represented and on the nature of the possible actions.[9] In particular,

- The planner has complete knowledge of the the relevant fluents and actions. For example, for any fluent f in the domain, the planner can decide whether either f or $not f$ holds in the initial state; typically this assumption is realized by using the so-called *Closed World Assumption* (Genesereth and Nilsson, 1987).
- The world is changed only by the actions of the agent executing the plan. Everything remains unchanged unless it is explicitly forced to change by the declared effects of the actions.

- Concurrent actions can never occur, and all the actions have only deterministic effects.

By exploiting these assumptions, from a logical point of view the planner can easily solve (without resorting to any frame axiom) the *temporal projection problem*. In general, given a description of an initial state of the world and a description of which events (actions) occur (are performed), the problem of temporal projection concerns determining the (possible or necessary) consequences of these events (actions). In particular, deciding whether a precondition P of an action A does or does not hold after execution of a certain sequence of actions is an instance of this problem.[10] Temporal projection can be very important during planning, for example, to determine whether an action may be executed (has all its preconditions satisfied) at a certain point in the plan under construction.

When some of the previous assumptions are relaxed to handle more complex domains, with more flexible representations of actions and domain knowledge, the frame problem can become an essential difficulty for efficient planning (Chapman, 1987). The simple STRIPS assumption that everything remains unchanged unless it is explicitly declared to change can no longer be maintained. In particular, the closed-world assumption implicit in systems like UCPOP breaks down in worlds with disjunctive ambiguity since it can give rise to inconsistent theories (Genesereth and Nilsson, 1987).

Furthermore, as pointed out by Schubert (1990), Pednault's preservation conditions (1988, 1989), asserting necessary and sufficient conditions for a fluent to change, also cannot in general be formulated when knowledge of the domain is incomplete.

As a simple example in which this difficulty arises consider the *next-to problem* analyzed by Schubert (1990, 1994). Suppose for simplicity that in a robot's world the only possible action is `Goto(b)`, which has the only explicit effect of putting the robot next to a box `b`. However, in this world the robot going to a particular box B1 may "incidentally" also end up next to another box `B2` that happens to be near B1. So, in general we cannot say that the condition `nextto(ROBOT,b)` holds *if and only if* `Goto(b)` is executed for some box `b`. Similarly, it can be seen that necessary and sufficient conditions for `nextto(b)` becoming *false* also cannot be given. In other words, without exact (geometrical) knowledge of the world in which the robot acts, we cannot infer whether a next-to condition *persists* in a state reached by the execution of some `goto` action(s).

An interesting and efficient solution to this problem is to use *explanation closure* (EC) axioms, which were suggested in Schubert (1990, 1994), building on (Haas, 1987) as a monotonic solution to the frame problem for worlds with fully specified actions. EC axioms are axioms complementary to the effect axioms, stating the necessary conditions for a fluent to change.

For example, in the previous simple scenario, we can assert the following EC axiom stating that `nextto(ROBOT,b)` becomes true only if the robot performs either action `goto(b)` or action `goto(b')` for some box `b'` near to `b` (assuming that `nexto(b,b')` implies that `b` is near `b'`):

$$\forall(\texttt{a,x,s,s'})[[\texttt{Holds(nextto(R,x),s)} \wedge \neg\texttt{Holds(nextto(R,x),s')} \wedge \texttt{s'=Result(a,s)}] \rightarrow (\exists\ \texttt{y})[\texttt{a=Goto(R,y)} \wedge \texttt{Holds(near(x,y),s)}]]$$

Using this EC axiom, if, for example, we happen to know that the robot is *not* next to B1, and then it goes to B2, which is *not* near B1, we can *deductively* determine that the robot still is not next to B1 after he has arrived at B2.

Two of the merits of the explanation closure method are that EC axioms can be specified in standard first-order logic and that the number of EC axioms required is comparable to the number of effect axioms (rather than being much larger, as in the case of frame axioms). Furthermore, this approach allows a planner to reason monotonically about *nonchange*. From a computational point of view this can be a significant advantage with respect to traditional approaches based on nonmonotonic techniques (Schubert, 1994; Allen and Ferguson, 1994). Ass argued by Schubert (1990) it is one of the reasons making explanation closure a potentially very powerful method for coping with the frame problem in planning and in reasoning about change in general.

Reiter (1991) supplies a method of automatically generating explanation closure axioms from effect axioms, under certain assumptions about the form and completeness of the effect axioms. He combines effect axioms with explanation closure axioms to obtain a single *successor-state axiom* for the truth of each fluent (or negated fluent) at the end of any action. Pednault (1988, 1989) had previously discussed the use of such axioms for formalizing and extending STRIPS.

Recently, Reiter (1996) proposed a generalization of Green's classical formulation of deductive planning (Green, 1969), which uses successor state axioms in the context of the situation calculus enriched to handle concurrent actions, external natural actions, and continuous time.

Schubert (1994) shows that explanation closure is a powerful method that allows monotonic solution of many problems in Sandewall's test suite (Sandewall, 1994a) for reasoning about change. He also argues that the completeness assumption on which successor state axioms rely is too strong and can easily break down, for example, when effects of actions are somewhat unpredictable, incompletely known, or expressed in terms of "vague" predicates such as `nextto`.

Allen and Ferguson (1994; see their chapter in this book, Chapter 7), show how to use explanation closure in the context of a representation of events and actions based on interval temporal logic.

2.5.2. TEMPORAL CONSTRAINTS IN PLANNING

During the search process the planner must (partially or totally) order the set of actions forming the plan by imposing a collection of appropriate ordering constraints. Such constraints are essential to guarantee the correctness of the resulting plan, that is, to guarantee that if the actions are executed starting at the initial state and consistently with these constraints, then the goal(s) will be achieved.

In nonlinear planning (see, for example, Chapman, 1987; Sacerdoti, 1975; Tate, 1977; Weld, 1994; Wilkins, 1988) the actions in a plan are partially ordered and reasoning about ordering constraints is required, for example, when the planner attempts to establish a subgoal G by "reusing" an action already in the plan under construction. In particular, the planner needs to determine the set of actions already in the plan that have an effect matching G and that can *consistently precede* the action whose precondition is G.

Moreover, in nonlinear planners such as UCPOP it is important to assert constraints that prevent an (instantaneous) action A from lying within a certain interval between two other actions $A1$ and $A2$, that is, $A < A1$ *or* $A2 < A$ and $A1 < A2$. In particular, this is required during planning to "protect" an established condition, when an earlier action ($A1$) serves to achieve the preconditions of a later one ($A2$), and no further action (A) should be inserted between them that would subvert those preconditions (because one of the effects of A falsifies some protected precondition of $A2$). An essential task of the planner is to determine whether the set of all the ordering constraints is consistent, which is an NP-complete problem (Gerevini and Schubert, 1994b). In UCPOP this is accomplished by a search that attempts to find a disjunct for each disjunctive constraint so as to obtain a consistent set of simple ordering constraints (if one exists). This search can be exponential since the problem is NP-hard.

Another planning framework where temporal reasoning is crucial is Allen and Koomen's planning system (1983). Their framework uses a richer temporal representation than (traditional) nonlinear planners that is based on an interval algebra formed from subsets of thirteen basic relations (Allen, 1983) (see the following section). By allowing for temporal constraints among properties of domain objects and actions available to a planner, they are able to reason effectively about simultaneous actions and to construct plans that take into account interactions among contemplated actions (such as disjointness constraints that prevent actions from overlapping because they contend for some resource.)

The reasoning task of checking the consistency of these relations is also essential in the planning algorithm presented by Allen (1991). Unfortunately, since the temporal relations used belong to the interval

algebra, and reasoning with such an algebra is NP-hard (Vilain and Kautz, 1986), this task can be a bottleneck in the performance of the planning algorithm on large-scale problems.[11]

Other temporal constraints that a planner may need to manage concern the inherent duration of an action; goals with possible deadlines (as in the planners FORBIN (Dean et al., 1988) and ZENO (Penberthy and Weld, 1994)); and the coordination of an action with externally scheduled events (as in the DEVISER planning and scheduling system (Vere, 1983)).

Thus an important issue in designing an effective general planner is the efficient handling of a variety of qualitative and metric temporal constraints. Although in the worst case this often requires solving an NP-hard problem, recently some complete algorithms and data structures that work well in practice have been developed (see Gerevini and Schubert, 1995a; van Beek and Manchak, 1996; Ladkin and Reinefeld, ming; Nebel, 1996).

These techniques can manage a rich class of (qualitative) temporal constraints, including those mentioned above in connection with nonlinear planners, as well as those used in Allen and Koomen's framework. In particular, the techniques proposed by Gerevini and Schubert (1994a, 1995a) are very efficient in practice and are often able to decide disjunctions deductively by fast preprocessing techniques (linear or low-order polynomial time in the worst case). Their algorithms involve the use of a graphical representation of temporal information called *D-timegraphs* (where D stands for "disjunctive"), and unlike many complete temporal reasoning methods are *scalable* in the sense of being sufficiently economical in their use of storage and time to allow application to large data sets (say hundreds or thousands of relationships). The scalability of D-timegraphs makes them a promising approach for planning in temporally complex domains without sacrificing well-foundedness.

2.6. Reasoning About Temporal Relations

As mentioned at the end of Section 2.3, the research in AI on reasoning about time has mainly addressed computational issues. Our presentation in this section will adopt a similar perspective, focusing first on qualitative relations, then on metric constraints, and finally on a few prominent implemented reasoning systems. In the presentation we assume that time is linear, dense, and unbounded and that the basic time units are either points or intervals. Such models of time predominate in AI applications that involve reasoning about time. (Some interesting studies in the context of AI which investigate other kinds of time models are Vilain, 1982; Meiri, 1992, 1996; Ligozat, 1996b; Jonsson et al., 1996).

2.6.1. QUALITATIVE RELATIONS

Fundamental frameworks for reasoning about qualitative temporal relations are Allen's *interval algebra* (1983) and Vilain and Kautz's *point algebra* (1986, 1990). The interval algebra (IA) is a relation algebra (Tarski, 1941; Ladkin and Maddux, 1994; Hirsh, 1996) based on thirteen basic relations between intervals corresponding to all the possible ways in which the endpoints of two intervals can be ordered (see Figure 3).[12] By combining all possible sets of basic relations in disjunctive form, 2^{13} interval relations can be derived. For example, by combining *before* and *overlaps* we obtain the relation *before or overlaps*:

$$(I \text{ before } J) \text{ or } (I \text{ overlaps } J) \equiv I \text{ (before or overlaps) } J$$

The point algebra (PA) is a relation algebra based on the three basic relations between time points $<$, $=$, and $>$ and whose elements are the set of binary relations $\{<, \leq, =, \geq, >, \neq, ?\}$, where ? is the disjunction of the three basic relations.

Given a set of (assertions of) qualitative relations, fundamental reasoning tasks include *determining consistency* (satisfiability) of such a collection, *finding a consistent scenario* (an interpretation for all the temporal variables involved that is consistent with the information provided), and *deducing new relations* from those that are known (or computing their deductive closure).[13] These tasks are NP-hard problems if the assertions are in IA (Vilain et al., 1990), while they can be solved in polynomial time if the assertions are in PA (Ladkin and Maddux, 1988a; Vilain et al., 1990; van Beek, 1992, 1992; Gerevini and Schubert, 1995b). As a result of the latter fact, these problems are also tractable in a restriction of IA called simple interval algebra (SIA) by van Beek (1992). This consists of the set of relations in IA that can be translated into conjunctions of relations in PA between the endpoints of the intervals (Ladkin and Maddux, 1988a; van Beek, 1992; van Beek and Cohen, 1990). For example, the relation *before or overlaps* between the intervals I and J can be translated into

$$I^- < I^+ \quad \wedge \quad J^- < J^+ \quad \wedge \quad I^+ < J^+ \quad \wedge \quad I^- < J^- \quad \wedge \quad I^+ \neq J^-$$

where I^- and I^+ indicate the starting and the end time points of I (analogously for J).

Nebel and Bürckert (1995) identified a maximal tractable subalgebra of IA called the ORD-Horn subclass. This subclass subsumes SIA extending the percentage of IA relations that can be handled efficiently from 2.3 per cent (for SIA) to 10.6 per cent.[14] In particular, they proved that the consistency of a set of assertions of relations in the ORD-Horn class can be determined by imposing the *path-consistency* property on the set (Montanari, 1974; Mackworth, 1977). Such a property can be computed in

Relation	Inverse	Meaning
I before J	*J after I*	
I meets J	*J met-by I*	
I overlaps J	*J overlapped-by I*	
I during J	*J contains I*	
I starts J	*J started-by I*	
I finishes J	*J finished-by I*	
I equal J	*J equal I*	

Figure 3. The thirteen basic relations of the interval algebra

$O(n^3)$ time and $O(n^2)$ space, where n is the number of intervals involved in the input set of assertions.[15]

Recently, other tractable subclasses of IA containing more relations than those in ORD-Horn have been identified (Drakengren and Jonsson, 1996). However, such classes do not contain all of the thirteen basic relations, and hence they do not permit the specification of complete knowledge about the relation holding between two intervals of time.

A set of (asserted) IA-relations can be represented with a *constraint network* (Montanari, 1974) whose vertices represents interval (point) variables, and edges are labeled by the relations holding between these variables. Enforcing path-consistency for a given set of relations corresponds to enforcing path consistency for their network representation. This task requires deriving a (possibly) new equivalent network in which for every 3-vertex subnetwork formed by the vertices i, j, and k, the relation R_{ik} labeling the edge from i to k is "stronger" than the composition of the relations R_{ij} and R_{jk} labeling the edges from i to j, and from j to k, respectively.[16]

Allen (1983) originally proposed calculating the $2^{13} \times 2^{13}$ possible compositions of IA-relations dynamically, using a table storing the 13×13 compositions of the basic relations of IA. Other significantly improved methods have been proposed since then by Hogge (1987) and by Ladkin and Reinefeld (ming). Ladkin and Reinefeld compared these methods and

showed that their method of storing all the possible compositions in a table (requiring about 64 megabytes of memory) is much faster than any alternative.

Table 1 summarizes the computational complexity of the best known algorithms for reasoning about relations in PA, PA^c, SIA, SIA^c, and ORD-Horn, where PA^c is the (convex) subalgebra of PA containing all the relations of PA except $\neq$, and SIA^c is the (convex) subalgebra of SIA formed by the relations of IA that can be translated into conjunctions of relations in PA^c.

The algorithm for determining the consistency of a set of relations in PA^c/SIA^c and in PA/SIA is based on methods for finding the "strongest connected components" in a directed graph (see Tarjan, 1972) and then examining the edges connecting vertices within such components (van Beek, 1992).

The algorithm for finding a consistent scenario for a set of relations in PA^c/SIA^c and in PA/SIA is based on first determining the consistency of the set and then topologically sorting the vertices of a directed acyclic graph (van Beek, 1992).

The algorithms for computing the deductive closure of a set of relations in PA^c/SIA^c and for deciding the consistency of a set of relations in the ORD-Horn class are based on algorithms that enforce path-consistency (see Allen, 1983; Vilain et al., 1990; van Beek, 1992; van Beek and Manchak, 1996).

The algorithm for finding a consistent scenario for a set of relations in the ORD-Horn class is based on two main steps (Gerevini, 1997). The first one checks the consistency of the set by imposing the path-consistency property on the set. The second step finds a consistent scenario for a particular set of relations in PA involving the endpoints of the interval variables of the input set of ORD-Horn relations.[17]

The algorithms for computing the deductive closure of a sets of relations in PA/SIA are based on first enforcing path-consistency and then examining

Problem	PA^c/SIA^c	PA/SIA	ORD-Horn
Consistency	$O(n^2)$	$O(n^2)$	$O(n^3)$
Consistent scenario	$O(n^2)$	$O(n^2)$	$O(n^3)$
Deductive closure	$O(n^3)$	$O(n^4)$	$O(n^5)$

TABLE 1. Time complexity of the best known temporal reasoning algorithms for PA^c/SIA^c, PA/SIA, and ORD-Horn.

some special four-vertex subnetworks to reduce certain implicit $\neq$ relations to $<$ relations (van Beek, 1992; Gerevini and Schubert, 1995b).

Finally, a simple algorithm for computing the deductive closure of a set of relations in the ORD-Horn class is based on using a path-consistency algorithm to determine whether any basic relation is a feasible relation between two interval variables.

The ORD-Horn class, though computationally attractive, is not practically adequate for all AI applications because it excludes disjointness relations such as *before or after*, which are important in planning and scheduling. For example, planned actions often cannot be scheduled concurrently because they contend for the same resources (agents, vehicles, tools, pathways, and so on). Similarly, reasoning about disjoint actions or events can be important in natural language understanding (for example, contrast "John reviewed a paper, practiced piano, and vacuumed the floor; he had no time left to cook dinner" with "John sprawled on the sofa, read a magazine, and listened to music; he had lots of time left to cook dinner").

Golumbic and Shamir (1993), and Drakengren and Jonsson (1996) have investigated some interesting subclasses of IA that contain interval disjointness relations. In particular, they provide the computational complexity analysis of their classes, as well as new graph-based algorithms. From these results it is clear that as long as we cannot refer to endpoints of intervals but only to intervals as a whole, we can handle interval disjointness in combination with certain other interval relations polynomially (for instance, when all we can say is that an interval I is disjoint from another interval J, or that I is before J).

However, as wasdiscussed in Section 2.5.2, the ability to refer to endpoints of intervals is useful in plan reasoning, where it is important to be able to say that a certain time points (perhaps an instantaneous action, or the beginning or end of an action) must not lie within a certain interval (another action, or the interval between two actions).[18] Unfortunately, it turns out that the minimal expressive power needed to express temporal nonoverlap in a point-based representation leads to intractability (if $P \neq NP$) (Gerevini and Schubert, 1994b). This is true even when we restrict ourselves to sets of statements asserting strict exclusion of certain time points from certain time intervals whose endpoints are unordered. It is thus important to formulate techniques that work well on average (or in practice), without sacrificing completeness.

Typically, the algorithms for handling classes of relations involving temporal nonoverlap (and more generally the full class of relations in IA) are based on search methods that use backtracking (Bitner and Reingold, 1975; Shanahan and Southwick, 1989). In particular, Gerevini and Schubert (1994a) propose a selective backtracking method for checking the consistency of a set of relations involving temporal nonoverlap that

effectively addresses the problem of scalability and that can be extended to arbitrary disjunctions of point relations. Ladkin and Reinefeld (1992) proposed a method for finding a consistent scenario of a set of relations in IA that is based on chronological backtracking and that uses path-consistency algorithms as a pruning technique. At the time of writing their algorithm, recently also investigated by van Beek and Manchak (1996) and by Nebel (1996), appears to be the fastest known algorithm for relations in the full interval algebra.

2.6.2. METRIC TEMPORAL CONSTRAINTS

As was discussed in the trains example of Section 2.3, metric temporal informations of the form "the trip of train `T1` from `C1` to `S1` takes six to eight hours" or "the travel time of `T1` from `C1` to `C2` should be less than forty-eight hours" are practically very important.

A fundamental framework for reasoning about metric temporal relations is Dechter, Meiri and Pearl's (1991) TCSP (temporal constraint satisfaction problem) model. A TCSP consists of a set of temporal variables representing points in a continuous time domain and a set of distance constraints between temporal variables. Each constraint is of the form

$$y - x \in \mathrm{I}_1 \cup \mathrm{I}_2 \cup ... \cup \mathrm{I}_n,$$

where y and x are point variables, I_1, I_2, ..., I_n are continuous closed intervals, and n is any integer.[19] As for the case of sets of qualitative relations, a TCSP can be represented through a constraint network where vertices represent the variables and edges the constraints. Given a TCSP the main reasoning tasks are similar to those in the qualitative case. They include *determining the consistency* of the set of constraints, *finding a solution* (an instantiation of all the variables that is consistent with the constraints), computing the *minimal network* representation (an equivalent temporal constraint network representation where the intervals of the constraints are the tightest possible);[20] and enforcing *global consistency* of the input set of constraints (making the set of constraints completely explicit, allowing for extension of the set of "partial" solutions of each subset of the constraints to "full solutions" for the whole set) (Dechter, 1992).

These problems are NP-hard for TCSP, but they become polynomial for a useful restriction of TCSP called STP (Dechter et al., 1991) that is based on constraints of the form $y - x \in \mathrm{I}$ (STP-constraints), where x and y are point variables and I is a (possibly open or semiopen) interval in the time domain. For STP-constraints all the reasoning tasks mentioned above can be accomplished in $O(n^3)$ time and $O(n^2)$ space (Dechter et al., 1991; Gerevini, 1997), where n is the number of time points constrained.[21] Such

complexity bounds refer to algorithms that are mainly based on traditional techniques for directed weighted graphs (see Floyd and Warshall's all-pairs shortest paths algorithm, and Bellman and Ford's single-source shortest paths algorithm (Cormen et al., 1990)), or on variants of them.

An interesting property of STP is that the minimal network of a set of STP-constraints is globally consistent. Another practically important related property of STP is that this class of problems allows efficient query answering, such as checking whether a given set of STP-constraints is consistent with the an existing (minimal) network of STP-constraints or whether it is necessarily satisfied in it (Brusoni et al., 1995).

Recently, an extension of STP to include inequations (that is, binary constraints of the form $y - x \neq d$, where d is any value in a dense time domain – such as the rational or real numbers) has been investigated. This extension, called STP$^{\neq}$ by Gerevini and Cristani (1995), was first studied in Koubarakis (1992).[22] STP$^{\neq}$ provides a useful increase of expressiveness for domains where both "instantaneous" (point) events and extended events can occur, without losing tractability. In fact, the task of determining the consistency and of finding a solution for a given set of STP$^{\neq}$-constraints can be accomplished in $O(n^3 + k)$ time (Gerevini, 1997), the task of computing the minimal network in $O(n^3 + kn^2)$ time (Koubarakis, 1995; Gerevini and Cristani, 1995), and the task of enforcing global consistency in $O(kn^4)$ time (Koubarakis, 1995).

Table 2 summarizes time complexity of known algorithms for STP, STP$^{\neq}$, and TCSP. As for STP the complexity bounds for STP$^{\neq}$ refer to algorithms that are based on traditional graph algorithms and in addition to some specialized techniques for dealing with the inequations. The algorithms for TCSP use complete techniques based on backtracking, or approximate methods (see Dechter et al., 1991; Schwalb and Dechter, 1993, 1995).

The integration of qualitative and metric information is a practically

Problem	TCSP	STP$^{\neq}$	STP
Consistency	NP-hard	$O(n^3 + k)$	$O(n^3)$
Solution	NP-hard	$O(n^3 + k)$	$O(n^3)$
Minimal network	NP-hard	$O(n^3 + kn^2)$	$O(n^3)$
Global consistency	NP-hard	$O(kn^4)$	$O(n^3)$

TABLE 2. Time complexity of temporal reasoning for TCSP, STP$^{\neq}$ and STP. (n is the number of points constrained; k is the number of inequations.)

important issue especially in planning applications and in knowledge-based systems. Meiri's *general temporal constraint networks* (1996) and Kautz and Ladkin's *combined-metric-Allen* procedure (1991) are two prominent approaches allowing integration of STP-constraints with Allen's relations. Unfortunately, both of them are intractable, and Kautz and Ladkin's method is also incomplete.[23]

Very recently Jonsson and Bäckström (1996) proved the tractability of an interesting class of relations called *Horn disjunctive linear relations* (Horn DRLs). This class subsumes Nebel's ORD-Horn class as well as the STP$^{\neq}$-constraints. In particular, they proved that deciding the consistency of a set of assertions of Horn DRLs can be accomplished by using an algorithm based on linear programming. While this is indeed a significant result for the field, Jonsson and Bäckström's linear programming techniques can incur high computational costs.

It appears then that integration of qualitative and metric information is an aspect of temporal reasoning which deserves some further research. For example, such research might formulate new efficient representations and (complete or approximate) algorithms that work well in practice, or investigate other classes of relations that can express both qualitative and metric information (a recent related study on this subject is Hirsh, 1996).

2.6.3. IMPLEMENTED TEMPORAL REASONING SYSTEMS

The development of domain-independent temporal reasoning modules that can be used in AI applications as specialized reasoning engines is practically important. Several systems for representing and managing temporal relations have been implemented. Among them, two well-known examples are TIMELOGIC (Koomen, 1988), which is based on Allen's IA, and MATS (Kautz and Ladkin, 1991), which is based on both Allen's IA and on Dechter, Meiri, and Pearl's STP. These are two of the most ambitious systems, in terms of the relations they can handle. They implement polynomial constraint-propagation techniques, and their Lisp implementation is relatively simple. On the other hand, they have two drawbacks: they are incomplete, and they are very inefficient with large data sets (say, a set of input relations involving hundreds of intervals).

TIMELOGIC ensures completeness of consistency checking only when the input relations belong to the ORD-Horn subclass. MATS suffers the same limitation, and in addition its method for integrating metric and qualitative information is incomplete even when the input qualitative relations belong to SIA.[24] Some recently proposed techniques that improve previous methods for computing path-consistency could be used to improve the efficiency of these systems (van Beek and Manchak, 1996; Bessière, 1996; Ladkin and Reinefeld, ming).

Other systems based on graph algorithms have been developed with the aim of addressing scalability and supporting efficient reasoning with large data sets. Some important examples of such systems are TMM (Dean and McDermott, 1987; Dean, 1989; Boddy, 1993), IxTeT (Ghallab and Alaoui, 1989), Tachyon (Stillman et al., 1993, 1996) and TimeGraph I and II (Miller and Schubert, 1990; Gerevini et al., 1995; Gerevini and Schubert, 1995a).

TMM (time map manager) can handle the qualitative relations in PA^c (SIA^c) as well as metric temporal constraints expressing upper and lower bounds on the distance separating two time points, *dates* defined as offsets from a *global frame reference*, and *reldates* defined as offsets from a specified default date. TMM uses indexing techniques to efficiently manage large databases of temporal constraints (Dean, 1989). The system does not provide completeness guarantees, but it has the merit of being a more general reasoning tool than the other systems mentioned above. In particular, in addition to temporal constraints, it supports reasoning about change by managing *causal rules* of the form: *if a certain event occurs at time t, then a certain proposition becomes true at* $t + \epsilon$. Such rules are used by the system for accomplishing some temporal projection tasks. TMM has been used in planning and scheduling applications (Boddy, 1996).

Tachyon can manage STP-constraints as well as the qualitative relations in PA^c (SIA^c). It is based on a C++ implementation of the Bellman-Ford single source shortest-paths algorithm (Cormen et al., 1990), supporting consistency checking and the computation of the earliest or latest possible times of the point variables involved by the asserted constraints. Tachyon has a graphic interface for displaying and editing the temporal information represented through the constraint graph. It also has some diagnosis capabilities to detect negative cycles, when the set of the asserted constraints is inconsistent. This system has been used to support mixed-initiative planning in military applications (Stillman et al., 1996).

IxTeT (indexed time table) could originally handle only the qualitative relations in PA (SIA). Later it was extended to handle also metric constraints expressing upper and lower bounds on temporal distances between time points (Dousson et al., 1993). IxTeT uses a graph-based representation of the qualitative relations, which relies for efficiency on a maximum spanning tree, together with an indexing scheme of the vertices supporting very efficient computation of ancestor information. The system is not complete, since it cannot infer certain strict inequalities that are entailed by the relations represented in the graph. IxTeT has been used in the context of a situation recognition system for monitoring complex dynamic systems.

TimeGraph I was originally developed in the context of story comprehension and can efficiently manage large sets of relations in PA^c and SIA^c, as well as metric constraints such as upper and lower bounds

on the temporal distance between two time points and absolute times (dates). TimeGraph II is a complete reasoning system that provides useful extensions to TimeGraph I, including an increase of expressiveness allowing the system to represent relations in PA, SIA, and disjunctive relations expressing point-interval exclusion and interval disjointness. On the other hand, the current version of TimeGraph II does not support metric constraints.

In TimeGraph I-II temporal relations are represented through a *timegraph*, a graph partitioned into a collection of *time chains* (sequences of time points) which in TimeGraph II are automatically structured for efficiency. Efficient query handling is achieved through a time point numbering scheme, and a *metagraph* data structure that guides the search processes on the same chain and across the chains.[25]

TimeGraph II also supports D-timegraphs, timegraphs augmented by a collection of disjunctions of relations of the form $x < y$ *or* $w \leq z$. Although reasoning with such disjunctive information is in the worst case NP-hard, experimental results given in (Gerevini and Schubert, 1995a; Yampratoom and Allen, 1993) show that the method implemented in TimeGraph-II for deciding consistency is in practice very efficient.

2.7. Conclusions

Time representation and reasoning is fundamental in many applications of AI as well as of other disciplines of computer science, such as computational linguistics (see Song and Cohen, 1991; Lascarides and Oberlander, 1993; Hwang and Schubert, 1994), database design (for a review on temporal databases see Snodgrass, 1990; Kline, 1993; Özsoyŏglu and Snodgrass, 1995; Orgun 1996), and computational models for molecular biology (Benzen, 1959; Golumbic and Shamir, 1993). In this article we have selectively surveyed some aspects of temporal reasoning, focusing on two subareas of particular interest to AI: reasoning about time and reasoning about actions and change. After introducing the basic concepts and techniques of both subareas, we briefly illustrated their essential role in the area of planning. We also provided an overview of the main results concerning reasoning about time.

The research conducted on reasoning about time in the last three decades has yielded a rich collection of results and techniques, ranging from computational complexity analyses to implemented reasoning systems. We believe that in general the state of the art in reasoning about time is relatively mature and has much to offer to the development of practical AI applications. However, at the time of writing some important aspects of temporal representation and reasoning still deserve further research. These include the following:

- New efficient representations and algorithms for managing combined qualitative and metric temporal information;
- New methods for representing and managing relations involving non-convex intervals, which for example can be useful in the representation of periodic events (see Leban et al., 1986; Poesio and Brachman, 1991; Terenziani, 1996);
- Extensions of the TCSP framework to represent stochastic temporal information such as "the travel of the train from Venice to Milan takes on average three hours with a standard deviation of five minutes" or other probabilistic temporal information;[26]
- Efficient methods for incremental maintenance of a temporal constraint network (or of any other kinds of representation), when new constraints are added or removed (some examples of the few studies conducted in this direction are Bell and Tate, 1985; Cervoni et al., 1994; Gerevini et al., 1996);
- Efficient (complete or approximate) algorithms for reasoning with disjunctive information in the context of the TCSP framework, possibly extended to represent relations expressing interval disjointness or point-interval exclusion (see Schwalb and Dechter, 1995);
- The integration of a temporal reasoner into a more general system of knowledge representation and reasoning, capable for example of using specialized temporal and spatial modules for representing movement information;
- The use of temporal constraint reasoning in the context of formal methods for reasoning about actions and change (see Schwalb et al., 1994).

Concerning formal methods for reasoning about actions and change, current techniques still appear to be far from practicality. Despite, significant recent progress in this subfield (see Pinto, 1994; JLC, 1994; Sandewall, 1994a; Lin, 1996; Reiter, 1996), current approaches cannot address all the ontological and representational characteristics of complex real situations, which include actions with nondeterministic effects, side-effects, or delayed effects; probabilistic or partial information about the scenario descriptions; external change and exogenous events; multiple agents; concurrent actions; continuous change.

However, some researchers (see Sandewall and Shoham, 1995) claim that the development of a general-purpose method should not be the ultimate goal of future research. Instead, the investigation of complementary methods, capable of addressing different phenomena which characterize restricted classes of scenarios (and their possible integration) is probably a more promising direction.

Similarly, computational methods for reasoning about actions and change are in general still inadequate for dealing with problems of the

real world. In particular, in well-founded domain-independent planning the expressiveness of current planners is in general not yet fully satisfactory, and the performance of the algorithms and representation used is not yet adequate for problems of practical size. One very promising research direction concerns the development and use of techniques for *preprocessing* the specification of a planning domain (see Blum and Furst, 1995; Gerevini and Schubert, 1996; McDermott, 1996; Friedman and Weld, 1996). Such techniques can then be used to discover constraints on the structure of the search space, which can be exploited during planning in order to significantly cut down the search space.

Another intriguing recent development is the application of fast satisfiability testing techniques to planning (see Kautz and Selman, 1992, 1996; Kautz et al., 1996). The general idea is to convert a planning problem to a propositional satisfiability problem, and then using efficient (complete or stochastic) techniques to search for solutions (see Crawford and Auton, 1993; Selman et al., 1994).

Acknowledgments

I would like to thank Len Schubert for many helpful comments on an earlier version of this chapter. I am also grateful to George Ferguson for suggestions that helped to improve the quality of the presentation, to Peter Ladkin for some technical suggestions, and to the editor, for his invitation to contribute to this book.

Notes

[1] We choose to deepen this subfield, partly because the book does not contain a specific contribution on this subject and partly because this is one of the areas in which the author has conducted active research for several years.

[2] This section gives only quite a general introduction. For a more detailed treatment the interested reader is referred to Ladkin (1987b), Allen and Hayes (1989), van Benthem (1991), Galton (1995b), Galton's chapter in this book (Chapter 10).

[3] Other less commonly used primitive units are semintervals (Freksa, 1992a) and nonconvex intervals (Ladkin, 1986; Ligozat, 1990, citeyearLig91; Morris et al., 1993). Vilain (1982), Meiri (1996), and more recently Jonsson, Drakengren, and Bäcström (1996) studied frameworks where both points and intervals are primitive units.

[4] Suppose,for example, that the travel times of a train vary according to the kind and quantity of load transported.

[5] As observed in Sandewall and Shoham (1995) the predicate `Holds` is commonly used in the context of SC, though it was not used in the original formulation of SC.

[6] This example and the one that we will use to illustrate the qualification problem are borrowed from Shoham and Goyal (1988).

[7] Other methodologies are briefly reviewed in Sandewall's chapter of this book, Chapter 9.

[8] For a critical survey of these and other methods see Sandewall and Shoham (1995) and Sandewall (1994a).

[9] The actions that can be handled by UCPOP are a subset of those that can be expressed by Pednault's action description language (ADL) (1989), which is more expressive than the STRIPS's language. In particular, UCPOP operates with actions that have conditional effects, universally quantified goals, preconditions and effects, and disjunctive goals and preconditions (Penberthy and Weld, 1992).

[10] From a computational point of view, when the events (actions) are partially ordered, and their descriptions correspond to instances of STRIPS operators, temporal projection is NP-hard (Nebel and Bäckström, 1994; Dean and Boddy, 1988b).

[11] This is likely to be the case for instances of NP-complete reasoning problems that lie around critical values (a "phase transition", Cheeseman et al., 1991) of certain parameters of the problem space. Ladkin and Reinefeld (1992; ming), and more recently Nebel (1996) identified phase transitions for the problem of determining the consistency of a set of assertions of relations in the interval algebra.

[12] Allen and Hayes (1989) showed that one of these basic relations, meets, can be used to express all the others.

[13] van Beek calls the problem of computing the deductive closure computing the "minimal labels" (between all pairs of intervals or points) in van Beek (1992) and computing the "feasible relations" in (van Beek, 1992).

[14] IA contains $2^{13} = 8192$ relations, SIA contains 188 relations (Ladkin and Maddux, 1988a; van Beek and Cohen, 1990), and the ORD-Horn subclass is a strict superset of the relations in SIA containing 868 relations (Nebel and Bürckert, 1995).

[15] On parallel machines the complexity of iterative local path-consistency algorithms lies asymptotically between n^2 and $n^2 log\ n$ (Ladkin and Maddux, 1988a, 1994).

[16] Two networks are equivalent when the variables represented admit the same set of consistent interpretations in both the representations. A relation R_1 is stronger than a relation R_2 if R_1 implies R_2. Also recall that the relations of IA and PA form algebras, and hence they are closed under the operation of composition, as well as under the operations of converse and intersection. For more details the interested reader may consult (Tarski, 1941; Ladkin and Maddux, 1994; Nebel and Bürckert, 1995; Hirsh, 1996).

[17] Another algorithm for accomplishing this task has been proposed by Ligozat (1996a), without a worst-case complexity analysis.

[18] Note that two such relations used in conjunction can also express disjointness of two intervals.

[19] As discussed in (Dechter et al., 1991; Kautz and Ladkin, 1991), the TCSP model can easily be extended to the case where the intervals of the constraints are (semi)open, or have $-/+$ infinity as bounds.

[20] Two temporal constraint networks are equivalent when the corresponding set of constraints admit exactly the same solutions.

[21] Actually, the tasks of consistency checking and of finding a solution can be accomplished using $O(e)$ space, where e is the number of the input constraints.

[22] Another related study is Isli (1994).

[23] Combined-metric-Allen is not able to infer certain strict inequalities entailed by the input set of constraints.

[24] See Gerevini and Cristani (1995) for an example illustrating this source of incompleteness.

[25] Recently Delgrande and Gupta proposed an interesting extension to the chain partitioning of a timegraph that is based on "serial-parallel graphs" (Delgrande and Gupta, 1996).

[26] Some related studies on "fuzzy" temporal reasoning are 6Dubois and Prade 1989, Console et al. (1991), Godo and Vila (1995).

Part II

Spatial Representation and Reasoning