Chapter 13
Time for Planning

13.1 Introduction

The purpose of this chapter is to present temporal representations and temporal reasoning techniques that are useful to planning with time and resources. Temporal planning itself will not be covered until the next chapter.

The mathematical structure of time is generally a set with a transitive and asymmetric ordering operation. It can be discrete, dense or continuous, bounded or unbounded, totally ordered or branching.[1] We will rely here on a simple structure of time as modeled by the set $\Re$ of real numbers.

The chapter introduces, informally through examples, temporal references and temporal relations (Section 13.2). Temporal references are instants or intervals. Relations are, for example, before, overlap, or a numerical relation. In planning, temporal relations can be conveniently represented and handled with CSP-based approaches and techniques.[2] Two main formalisms for qualitative relations are developed: the time-point algebra, in Section 13.3.1, and the interval algebra in Section 13.3.2. The relationships between these two formalisms are discussed in Section 13.3.4. Finally, the quantitative temporal constraint networks are introduced in Section 13.4. The chapter ends with a discussion and exercises.

13.2 Temporal References and Relations

Consider the following assertion: "crane2 loads container5 onto robot1 during interval *i*." This assertion can be analyzed from the causal point of view: *what* changes are entailed by the assertion and *what* conditions are required for it to hold consistently. But it can also be analyzed from the temporal point of view: *when* other related assertions can or cannot take place. These *what* and *when* issues may or may not be decomposed into two separate computational processes, but they need to

1. For a discussion of the mathematical structure of time, see [474, 520].
2. Refer to Chapter 8 for a review of CSP concepts and algorithms.

be distinguished conceptually. Temporal planning, as any other form of temporal reasoning, has to deal with these two distinct issues.

1. *What*: reasoning about *change* and *causality* relations entailed by actions and events.
2. *When*: dealing with *temporal references* of propositions and temporal relations, our purpose in this chapter. These temporal references are time periods during which a proposition holds or time points at which a state variable changes its value. They are represented as instants or intervals. An *instant* is a variable ranging over the set $\Re$ of real numbers. An *interval* is a pair (x, y) of reals, such that $x \leq y$.

Typically, a planner that reasons about time operates on a temporal database that maintains temporal references for every domain proposition that varies in time: when does it hold or when does it change. These are the temporal references of the proposition. The planner asserts relations among these temporal references. The temporal database has to provide functions for querying, for updating, and for maintaining the consistency of the database. Let us illustrate through a few examples the different types of temporal references and relations we are considering.

Example 13.1 In the DWR domain, consider the operation of loading the container cont onto a robot rob at location loc.

- Let t_1 be the instant at which the robot enters the location loc, t_2 the instant at which the robot stops in loc,[3] and let i_1 be the interval $[t_1, t_2]$ corresponding to the robot *entering* loc.
- Let t_3 be the instant at which the crane starts picking up the container, t_4 the instant at which the crane finishes putting down the container onto the robot, and i_2 the interval $[t_3, t_4]$ corresponding to the crane *picking up and loading* cont.
- Let t_5 be the instant at which the container cont is loaded onto the robot, t_6 the instant at which the container stops being loaded onto the robot, and i_3 the interval $[t_5, t_6]$ corresponding to the container staying *loaded* on rob.

Let us assume that the crane can start picking up a container as soon as the robot on which it is to be loaded has entered the location. One possible position in time for these instants and intervals is depicted in Figure 13.1. Other possibilities would be to have $t_3 = t_1$ or $t_3 = t_2$ or $t_2 < t_3 < t_5$. Here we assume that the container is loaded as soon as the crane finishes putting it down, i.e., $t_4 = t_5$. The example does not mention any reason for the end of i_3; i.e., the value of t_6 can be arbitrarily large. ■

In this example, the set of instants $\{t_1, \ldots, t_6\}$ and the set of intervals $\{i_1, i_2, i_3\}$ are *temporal references* that specify when domain propositions are true. Here, these two

3. Presumably there is a specific loading/unloading position in loc at which the robot stops.

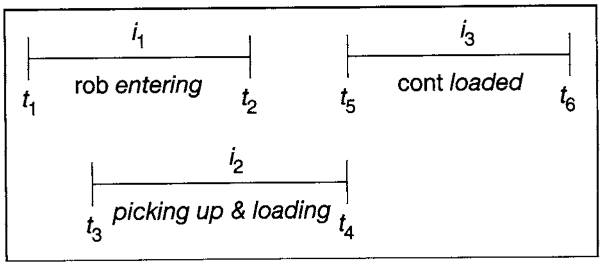

Figure 13.1 Entering and loading operations.

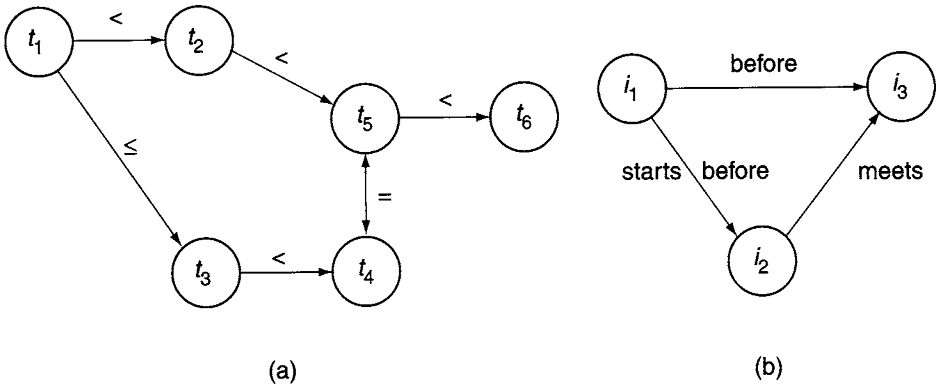

Figure 13.2 Networks of constraints over instants (a) and intervals (b).

sets play the same role. We may describe this example using just the three intervals: the *entering* of the robot during i_1, the *picking up and loading* of the container during i_2, and the container staying *loaded* during i_3. Intervals i_1 and i_2 refer to *activities*, while interval i_3 refers to a *proposition*. Note that the example does not give metrical information about duration or about absolute time positions: it is a qualitative representation.

The temporal relations between these temporal references can be expressed as binary constraints between instants or between intervals. The constraints for Example 13.1 are expressed in the two networks over instants and intervals, Figures 13.2(a) and (b), respectively. In these networks, nodes are labeled by temporal references, and arcs are labeled by temporal constraints (to be defined formally in upcoming sections), which express precedence and equality of time points or the relative positions of intervals.

Example 13.2 Consider a DWR domain where security is an important concern: every moved container has to be *inspected* and *sealed*. The inspection operation uses

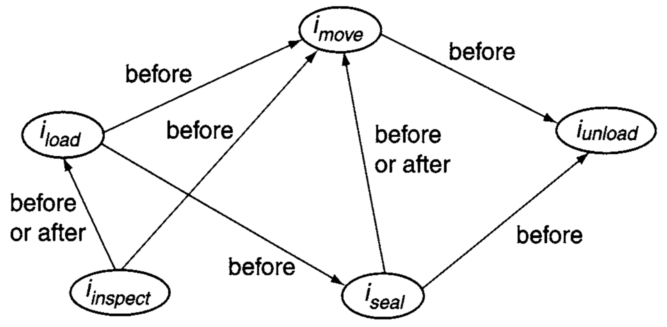

Figure 13.3 Network of constraints over intervals.

a sensor carried out by the crane, e.g., a camera that moves around the container. The sealing operation is performed by the robot itself. The following constraints must be met.

- Inspection is performed either *before or after* loading the container on the robot but *not while* holding it, and *before* the robot starts moving.
- Sealing is performed *after* loading the container on and *before* unloading it from the robot, and either *before or after* moving it to its destination, but *not while* in motion.

We may use five intervals (i_{load}, i_{move}, i_{unload}, $i_{inspect}$, and i_{seal}) to denote the *loading, moving, unloading, inspection,* and *sealing* operations, respectively. These intervals are nodes of a constraint network (Figure 13.3) whose arcs are labeled by the binary constraints between the intervals. The set of solutions of this constraint net corresponds to all feasible organizations of these five activities that meet the constraints.

■

An interval i is described by its two end points, noted i^- and i^+, together with the constraint $[i^- \leq i^+]$.[4] One could specify this example with a network of ten instants. A constraint such as $[i_{load}$ before $i_{move}]$ is translated into the conjunction of three constraints: $[i^+_{load} < i^-_{move}]$, $[i^-_{load} \leq i^+_{load}]$, and $[i^-_{move} \leq i^+_{move}]$. However, the constraint $[i_{move}$ before or after $i_{seal}]$ requires a more complex expression involving four instants: $[(i^+_{move} < i^-_{seal})$ or $(i^+_{seal} < i^-_{move})]$. This disjunction cannot be expressed into a single binary relation because it involves four instants; hence it cannot correspond to the label of an arc of a network of instants. Although the two networks of Figure 13.2 are equivalent, in general not every network of interval constraints can be translated into a network of instants (this will be detailed in Section 13.3.4). There are ways to combine instants and intervals into a general representation; however,

4. We will denote temporal constraints in the infix notation within square brackets as delimiters.

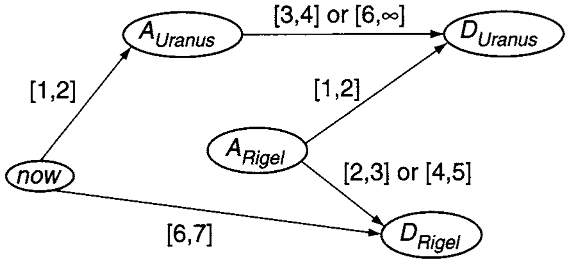

Figure 13.4 Network of numerical constraints over instants.

most temporal planning languages choose either instants or intervals for temporal references.

Let us now illustrate another type of *quantitative* temporal relation.

Example 13.3 Consider a DWR domain in which ships to be loaded and unloaded are not on dock indefinitely. The planner also may have to consider their schedules. Assume that two ships, the *Uranus* and the *Rigel*, are scheduled to be serviced. The *Uranus* arrival is expected within one or two days; it will leave either with a light cargo (then it will have to stay docked three to four days) or with a full load (then it will have to stay docked at least six days). The *Rigel* can be serviced either on an express dock (then it will stay docked two to three days) or on a normal dock (then it will stay four to five days).

Assume further that the *Uranus* has to depart one to two days after the arrival of the *Rigel*, which in turn has to depart six to seven days from now. Questions such as when the *Rigel* should arrive, whether it can be serviced on a normal dock, and whether the *Uranus* can take a full load can be answered by solving the constraint network in Figure 13.4. This network has five instants corresponding to the arrival and departure of the two ships (noted, respectively, A_{Rigel} and D_{Rigel} and similarly for the *Uranus*) and to the current time (noted *now*). It relates these instants with arcs labeled by numerical intervals or by disjunction of intervals, e.g., $(2 \leq D_{Rigel} - A_{Rigel} \leq 3) \vee (4 \leq D_{Rigel} - A_{Rigel} \leq 5)$. ■

To summarize, these examples illustrated two ways to express temporal information about activities and propositions of a planning domain: either with *instants* or with *intervals*. The first two examples consider *qualitative* constraints to relate two instants or two intervals, whereas this last example introduces *quantitative* and *absolute* time constraints. As illustrated in Example 13.2, intervals can be translated into instants, but some binary constraints on intervals cannot be translated into binary constraints on instants.

In the following sections, we will detail more formally how to manage qualitative constraints on instants and on intervals and quantitative constraints on instants.

13.3 Qualitative Temporal Relations

This section is concerned with qualitative temporal constraints. Two approaches will be considered, point algebra (PA) and interval algebra (IA), dealing with qualitative constraints on, respectively, time points and intervals. A geometric interpretation of the latter will then be introduced, enabling us to relate the two representations.

13.3.1 Point Algebra

Point algebra (PA) is a symbolic calculus that enables us to relate in time a set of instants with qualitative constraints without necessarily ordering them.

Two instants t_1 and t_2 that are set in time, with real values, can be related in only three possible ways: $[t_1 < t_2]$, $[t_1 > t_2]$, or $[t_1 = t_2]$. Two instants whose values are not known and whose relative positions are not precisely specified may be constrained, as in $[t_1 \leq t_2]$, $[t_1 \geq t_2]$, $[t_1 \neq t_2]$.

Let $P = \{<, =, >\}$ be the set of *primitive relation symbols* between instants (*primitives* for short). PA relies on the following set of qualitative constraints:

$$R = 2^P = \{\emptyset, \{<\}, \{=\}, \{>\}, \{<, =\}, \{>, =\}, \{<, >\}, P\}$$

Here, "$\emptyset$" denotes the empty constraint that cannot be satisfied and P the universal constraint.

Each element $r \in R$ is a set of primitives; r is a constraint interpreted as the *disjunction* of these primitives. For example, $[t \neq t']$ is denoted $[t\ r\ t']$ for $r = \{<, >\}$.

The usual operations on sets apply to R: $\cap, \cup$, etc. In addition, a composition operation, noted $\bullet$, is defined to handle transitivity:

$$\text{for } r, q \in R\text{: if } [t_1\ r\ t_2] \text{ and } [t_2\ q\ t_3], \text{ then } [t_1\ r \bullet q\ t_3].$$

This operation is computed according to the composition table for the three primitives (Figure 13.5) and to the distributivity property:

$$\text{for } r, s, q \in R\text{: } (r \cup q) \bullet s = (r \bullet s) \cup (q \bullet s), \text{ and similarly for } s \bullet (r \cup q).$$

Finally, the symmetrical constraint of $[t_1\ r\ t_2]$ is the constraint r' such that $[t_2\ r'\ t_1]$ iff $[t_1\ r\ t_2]$. r' is obtained by replacing in the set r the primitive $<$ by $>$ and symmetrically, while $=$ remains unchanged. Note that $(r \bullet q)' = q' \bullet r'$.

The set R with the two operations $(R, \cup, \bullet)$ is an *algebra* in the following sense: It is closed under the two operations, $\cup$ is an associative and commutative operation whose identity is $\emptyset$ (i.e., $(R, \cup)$ is a semigroup with a unit element), $\bullet$ is an associative operation whose identity is $\{=\}$, and the two operations are distributive.

•	<	=	>
<	<	<	P
=	<	=	>
>	P	>	>

Figure 13.5 Composition table for time-point algebra.

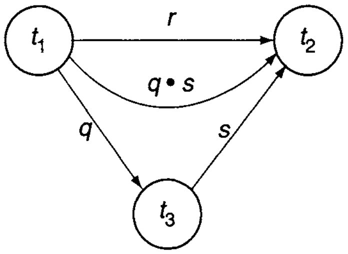

Figure 13.6 Transitive composition and conjunction.

However, the most useful operations, within a CSP framework, are $\cap$ and $\bullet$. Indeed, constraint propagation and consistency checking proceed by combining a given constraint r with the constraint entailed by transitive composition (Figure 13.6):

$$\text{if } [t_1\ r\ t_2] \text{ and } [t_1\ q\ t_3] \text{ and } [t_3\ s\ t_2], \text{ then } [t_1\ r \cap (q \bullet s)\ t_2].$$

If $r \cap (q \bullet s) = \emptyset$, then the constraints r, q, and s are inconsistent.

Example 13.4 Let us make explicit a few constraints in the PA network of Figure 13.2 (a) that are entailed by transitive composition.

$$(t_4, t_6) : [t_4 = t_5] \bullet [t_5 < t_6] \Rightarrow [t_4 < t_6]$$
$$(t_2, t_3) : [t_2 > t_1] \bullet [t_1 \leq t_3] \Rightarrow [t_2\ P\ t_3]$$
$$(t_2, t_4) : [t_2 < t_5] \bullet [t_5 = t_4] \Rightarrow [t_2 < t_4]$$

Another path $\langle t_2, t_3, t_4 \rangle$ gives for (t_2, t_4):

$$[t_2\ P\ t_3] \bullet [t_3 < t_4] \Rightarrow [t_2\ P\ t_4]$$

and the conjunction for the two constraints on (t_2, t_4) is $P \cap \{<\} = \{<\}$. ∎

More generally, let $X = \{t_1, t_2, \ldots, t_n\}$ be a set of instant variables, where the domain of each t_i is the set $\Re$ of real numbers. A *binary constraint network* for the point algebra is defined as a directed graph (X, C) where each arc in C is labeled by a constraint $r_{ij} \in R$: $[t_i \; r_{ij} \; t_j]$. We assume (X, C) to be symmetrical CSP, i.e., for all $i, j : r_{j,i} = r'_{ij}$. Note that if $(t_i, t_j) \notin C$, i.e., there is no explicit constraint in (X, C) between t_i and t_j, then $r_{ij} = P$, the universal constraint.

A tuple of reals $(v_1, \ldots, v_n)$ is a *solution* of (X, C) iff the n values $t_i = v_i$ satisfy all the constraints of C. The PA network (X, C) is *consistent* when a solution exists. Every pair (t_i, t_j) in a solution is related by a *single primitive relation* $p_{ij} \in P$ because it corresponds to a pair of $\Re^2$. This set of primitives characterizes entirely the relative positions of instants in a solution.

Proposition 13.1 *A PA network (X, C) is consistent iff there is a set of primitives $p_{ij} \in r_{ij}$, for each pair (i, j), such that every triple of primitives verifies $p_{ij} \in (p_{ik} \bullet p_{kj})$.*

Proof Assume the network to be consistent; then the set p_{ij} follows directly from the real values (v_i, v_j) of the solution. The solution meets all the constraints: $p_{ij} \in r_{ij}$; it also meets their composition. Conversely, assume that there is a set of primitives $p_{ij} \in r_{ij}$; then it is possible to find real values v_i, for $1 \leq i \leq n$, such that each pair of values is related by a primitive p_{ij}: there cannot be a conflict because $p_{ij} \in (p_{ik} \bullet p_{kj})$, and this set of values meets all the constraints. ■

In this qualitative calculus, we will not be interested in the real values of a solution but only in a set of primitives that meets the consistency condition.

Note that a PA network departs from the standard CSPs over finite domains because here the variable domains are not finite. However, most CSP definitions are easily extended to this case by viewing a constraint r_{ij} as a finite domain from which we are seeking a single element and a solution as being the set of primitives $p_{ij} \in r_{ij}$ meeting the consistency condition. In particular, we'll say that

- a primitive in $r_{ij} \in C$ is *redundant* if there is no solution where (t_i, t_j) are related by this primitive, and
- a network is *minimal* if it has no redundant primitive in a constraint.

Redundant primitives can be filtered out on the basis of the transitive closure propagation with a path-consistency algorithm in $O(n^3)$ (Figure 8.5). If the filtering leads to an empty constraint, r_{ij}, then the network is necessarily inconsistent. It turns out that this necessary condition is also a sufficient condition in this case: algorithm PC, incomplete in the general case, is complete for PA networks. Hence, a PA network (X, C) is consistent iff algorithm PC(X, C) does not return inconsistent. In planning, we will generate incrementally a network (X, C) by adding new instants and constraints while searching for a plan. In order to keep the current net consistent, we can use IPC, the incremental version of path consistency (Figure 8.6).

However, the path-consistency algorithms do not provide, after filtering, a minimal network. Indeed, the network in Figure 13.7 is path-consistent but the primitive $[t_1 = t_4]$ is redundant: there is no solution in which these two instants are equal. A 4-consistency algorithm is required for obtaining a minimal network [537, 538].

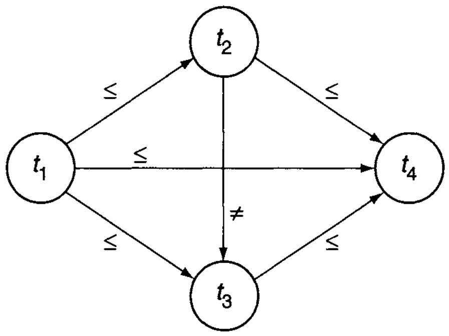

Figure 13.7 A path-consistent PA network.

Alternatively, one may restrict the set of constraints to $R - \{<, >\}$. The corresponding algebra, noted PA_c, is said to be convex. It is closed for the intersection and composition operations. Path consistency provides minimal networks for PA_c.

A more efficient approach for checking the consistency and for obtaining minimal PA networks is to use graph connectivity techniques. The *precedence graph* associated with a PA network (X, C) is a directed graph $G = (X, E)$ with the same set of vertices and with edges defined as follows.

- If r_{ij} is $\{<\}$ or $\{<, =\}$, then $(t_i, t_j) \in E$.
- If r_{ij} is $\{=\}$, then both $(t_i, t_j) \in E$ and $(t_j, t_i) \in E$.

E does not contain edges associated with the constraints $\neq$ or P. It can be shown that a PA network is consistent iff for any pair (t_i, t_j) such that $\{=\} \subseteq r_{ij}$, then t_i and t_j are in the same strongly connected component of G, i.e., there is a path in G from t_i to t_j and a path from t_j to t_i. Hence, consistency checking can be performed by computing strongly connected components of G and testing each constraint in C; both tasks require $O(|C|)$.

13.3.2 Interval Algebra

Interval algebra (IA) is similar to PA except that it deals with intervals instead of points. IA is a symbolic calculus that enables us to relate in time a set of intervals with qualitative constraints.

Two intervals i and j whose end points are precisely set in $\Re$ can be related qualitatively in only thirteen possible ways, seven of which are shown in Figure 13.8. These primitive relations correspond to all consistent cases, in PA, of precedence or equality of the four endpoints i^-, i^+, j^-, and j^+ of i and j, with the constraints $[i^- < i^+]$ and $[j^- < j^+]$. The thirteen relations are the following.

- b, m, o, s, d, and f stand respectively for *before, meet, overlap, start, during,* and *finish.*

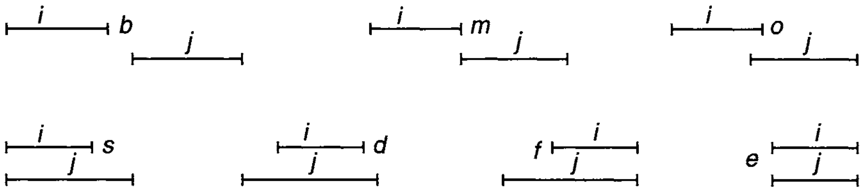

Figure 13.8 Seven primitive relations of IA.

- b', m', o', s', d', and f' stand respectively for *after, is-met-by, is-overlapped-by, is-started-by, includes,* and *is-finished-by.* These are the symmetrical relations of b, m, o, s, d, and f, i.e., $[i\ b'\ j]$ when $[j\ b\ i]$, $[i\ m'\ j]$ when $[j\ m\ i]$, etc.
- e stands for *equal,* which is identical to the symmetrical relation e'.

Two intervals whose relative positions are not precisely known can be related by any *disjunction* of such primitive relations.

Let $P = \{b, m, o, s, d, f, e, b', m', o', s', d', f'\}$ be the set of *primitive relation symbols* between intervals (*primitives* for short). IA relies on the following set of qualitative constraints:

$$R = 2^P = \{\emptyset; \{b\}; \{m\}; \{o\}; \ldots; \{b, m\}; \{b, o\}; \ldots; \{b, m, o\}; \ldots; P\}.$$

Each element $r \in R$ is a set of primitives; r is a constraint interpreted as the *disjunction* of these primitives. For example, $[i\ \{b, m\}\ i']$ denotes $[(i\ b\ i') \vee (i\ m\ i')]$. As in the previous section, "$\emptyset$" is the constraint that cannot be satisfied and P the universal constraint, which is always satisfied. Out of the 2^{13} constraints in R, one may select a subset of constraints that are useful for some applications and/or have a particular semantics, and give the constraints mnemonic names, e.g., while $= \{s, d, f\}$, disjoint $= \{b, b'\}$, and joint $= \{m, m', o, o', s, s', f, f', d, d', e\} = P -$ disjoint, etc.

The usual operations on sets apply to R: $\cap, \cup$, etc., as well as the composition operation, noted $\bullet$, for transitivity:

$$\text{for } r_{1,2}, r_{2,3} \in R\text{: if } [i_1\ r_{1,2}\ i_2] \text{ and } [i_2\ r_{2,3}\ i_3], \text{ then } [i_1\ (r_{1,2} \bullet r_{2,3})\ i_3].$$

This operation is computed from a composition table for the 13 primitives and from the distributivity property:

$$\text{for } r_1, r_2, r_3 \in R\text{: } (r_1 \cup r_2) \bullet r_3 = (r_1 \bullet r_2) \cup (r_2 \bullet r_3),$$

and similarly for $r_3 \bullet (r_1 \cup r_2)$.

Finally, the symmetrical constraint of $[i_1\ r\ i_2]$, i.e., r' such that $[i_2\ r'\ i_1]$, is obtained by replacing in r every primitive by its symmetrical primitive, i.e., b by b', o by o', etc.

$\bullet$	b	m	o	s	d	f	b'	d'	f'
b	b	b	b	b	$u \cup v$	$u \cup v$	P	b	b
m	b	b	b	m	v	v	$u' \cup v'$	b	b
o	b	b	u	o	v	v	$u' \cup v'$	$u \cup w'$	u
s	b	b	u	s	d	d	b'	$u \cup w'$	u
d	b	b	$u \cup v$	d	d	d	b'	P	$u \cup v$
f	m	m	v	d	d	f	b'	$u' \cup v'$	x
b'	P	$w \cup u'$	$w \cup u'$	$w \cup u'$	$w \cup u'$	b'	b'	b'	b'
d'	$u \cup w'$	v'	$o' \cup w'$	$o \cup w'$	y	v'	$u' \cup v'$	d'	d'
f'	b	m	o	o	v	x	$u' \cup v'$	d'	f'
s'	$u \cup w'$	$o \cup w'$	$o \cup w'$	$\{s,e,s'\}$	$o' \cup w$	o'	b'	d'	d'

Figure 13.9 Part of the composition table for IA.

Part of the composition table is given in Figure 13.9 where, for simplicity, curly brackets for sets are implicit and the following constraints in R are named: $u = \{b, m, o\}$; $v = \{o, s, d\}$; $w = \{d, f\}$; $x = \{f, e, f'\}$; and $y = P - \{b, m, b', m'\}$. Some (but not all) of the missing composition operations in the table can be completed with the general property of symmetrical constraints: $(r \cup q)' = q' \cup r'$, and by denoting that $\{e\}$ is the identity element for the composition.

Note that $\bullet$ is not a commutative operation in IA. For example, Figure 13.10 illustrates on the left of the figure. $\{d\} \cup \{b\} = \{b\}$; whereas on the right, $\{b\} \cup \{d\} = \{b, m, o, s, d\}$. This is because $[(i\ d\ j) \wedge (j\ b\ k)]$ constrains k to a single position with

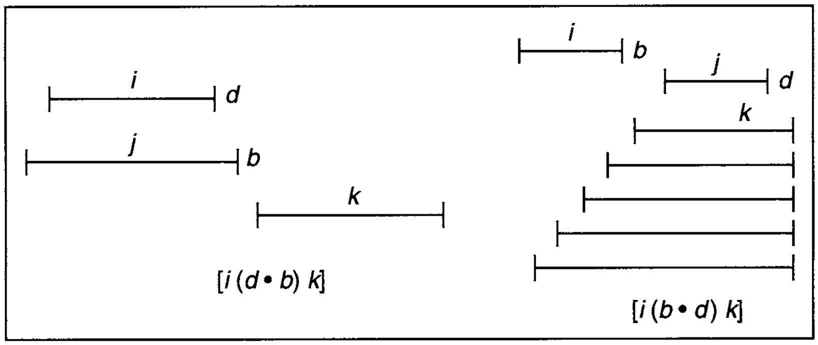

Figure 13.10 Two composition operations.

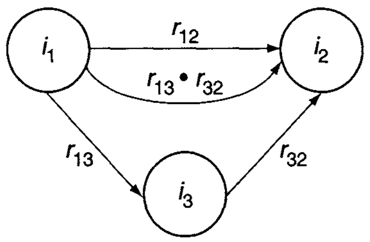

Figure 13.11 Transitive composition and conjunction over intervals.

respect to i, whereas $[(i\ b\ j) \wedge (j\ d\ k)]$ leaves five possibilities for k (drawn in the figure as separate segments) with respect to i. Each cell in the composition table can be obtained by just looking at the possible configurations of three intervals, i, j, and k. It can also be obtained with a PA net over the six end points i^-, i^+, j^-, j^+, k^-, and k^+ by finding all its subnetworks between i^-, i^+, k^-, and k^+ that correspond to valid PA networks for primitives in P.

As for PA, the set R in IA with the two operations $(R, \cup, \bullet)$ is an *algebra*: it is closed under two operations: $\cup$ is an associative and commutative operation, with identity $\emptyset$; $\bullet$ is associative, with identity $\{e\}$. The two operations are distributive.

Here also, the useful operation for constraint satisfaction relies on the transitive closure propagation (Figure 13.11, similar to Figure 13.6 for PA):

$$\text{if } [i_1\ r\ i_2] \text{ and } [i_1\ p\ i_3] \text{ and } [i_3\ q\ i_2], \text{ then } [i_1\ r \cap (p \bullet q)\ i_2].$$

If $r \cap (p \bullet q) = \emptyset$, then the three constraints are inconsistent.

Example 13.5 In the IA network for the situation described in Example 13.2 (see page 287) (Figure 13.12), suppose we want to know the constraint between the *inspect* and *unload* intervals. By transitivity through i_{move} we have:

$$[i_{inspect}\ (\{b\} \bullet \{b\})\ i_{unload}] = [i_{inspect}\ \{b\}\ i_{unload}].$$

The constraint between the *inspect* and *seal* intervals, computed by transitivity through i_{load}, is the universal constraint:

$$\{b, b'\} \bullet \{b\} = (\{b\} \bullet \{b\}) \cup (\{b'\} \bullet \{b\}) = \{b\} \cup P = P.$$

The same result is obtained by transitivity through i_{move}. ■

Let $X = \{i_1, i_2, \ldots, i_n\}$ be a set of interval variables. The domain of each interval is the half plane of $\Re^2$ defined by the inequality on the end points $i^- \leq i^+$. A *binary constraint network for interval algebra* is a directed graph (X, C) where each arc in C is labeled by a constraint $r_{ij} \in R$: $[i\ r_{ij}\ j]$. We assume (X, C) to be a symmetrical

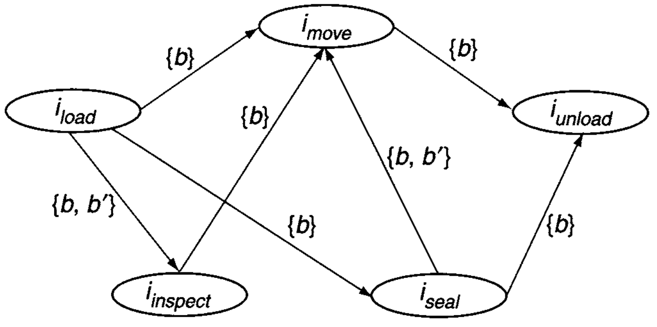

Figure 13.12 An IA network.

CSP, i.e., for all $i, j \in X : r_{j,i} = r'_{ij}$. Note that if $(i, j) \notin C$, i.e., there is no explicit constraint in (X, C) between intervals i and j, then $r_{ij} = P$, the universal constraint.

A tuple of pairs of reals $((i_1^-, i_1^+), \ldots, (i_n^-, i_n^+))$, with $i^- \leq i^+$, is a *solution* of (X, C) iff the n intervals $i = [i^-, i^+]$ satisfy all the constraints of C. The IA network (X, C) is *consistent* when a solution exists. Every pair (i, j) in a solution is related by a *single primitive relation* $p_{ij} \in P$.

Proposition 13.2 *An IA network (X, C) is consistent iff there is a set of primitives $p_{ij} \in r_{ij}$, for each pair (i, j), such that every triple of primitives verifies $p_{ij} \in (p_{ik} \bullet p_{kj})$.*

The proof is a direct transposition of the proof of Proposition 13.1 (see page 292) to IA. Note that it is essential in Proposition 13.2 that p_{ij} are *primitives*; in particular, the proposition does not hold if the intervals are related by a set of constraints meeting $r_{ij} \in (r_{ik} \bullet r_{kj})$.

The notions of redundant primitives in a constraint and of minimal networks for IA nets are as defined in standard CSPs. Redundant primitives can be filtered out by the transitive closure propagation with the path-consistency algorithm in $O(n^3)$. If the filtering leads to an empty constraint, then the network is necessarily inconsistent. However, this is no longer a sufficient condition: the path-consistency algorithm is not complete for IA networks. A counterexample is the network of Figure 13.13, which is path-consistent but is not consistent. Indeed, the consistency checking of IA networks is an NP-complete problem, as for general CSPs.

One may use the standard CSP algorithms such as backtrack search and forward checking (Figures 8.2 and 8.3, respectively) with some slight modifications: instead of choosing at each step a value for a variable, one chooses nondeterministically a primitive in a constraint and propagates this choice.

Let us close this section with some useful properties for temporal planning, with respect to the constraint in R that we proposed to name joint = $\{m, m', o, o', s, s', f, f', d, d', e\} = P - \{b, b'\}$. This constraint characterizes the case where intervals can have a nonempty intersection and enables us to detect various types of conflicts.

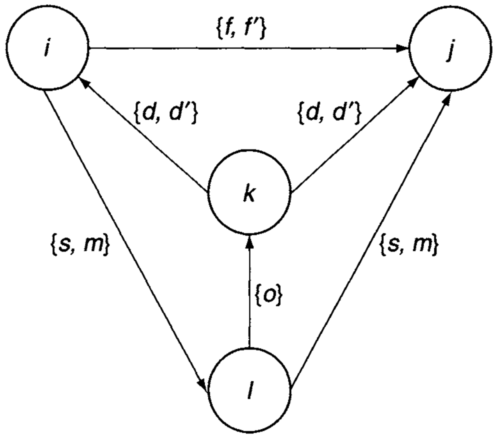

Figure 13.13 An inconsistent IA network.

Definition 13.1 Let (X, C) be a consistent IA network. Two interval variables $i, j \in X$ are *necessarily intersecting* iff for every solution of the network, the two corresponding intervals of $\Re$, $[i^-, i^+]$ and $[j^-, j^+]$, have a nonempty intersection, i.e., $[i^-, i^+] \cap [j^-, j^+] \neq \emptyset$. The two interval variables i and j are *possibly intersecting* iff there is a solution of the network that meets this equation. ■

Proposition 13.3 *Let r_{ij} be a constraint between two interval variables i and j in a consistent IA network (X, C). If $r_{ij} \subseteq$ joint, then i and j are necessarily intersecting.*

Proof For every solution of the network (X, C), the pair (i, j) is related by a primitive p_{ij} which is, by definition, included in the minimal constraint $\hat{r}_{ij}$ between i and j. Furthermore, in that solution, $[i^-, i^+] \cap [j^-, j^+] \neq \emptyset$ iff the corresponding primitive is neither b nor b'; any other primitive in joint corresponds to a nonempty intersection. Now, assume $r_{ij} \subseteq$ joint, because $\hat{r}_{ij} \subseteq r_{ij}$, then for every solution, the corresponding p_{ij} is neither b nor b', hence i and j are necessarily intersecting. ■

The converse of this proposition is weaker: if i and j are necessarily intersecting, then $\hat{r}_{ij} \subseteq$ joint, hence $r_{ij} \cap$ joint $\neq \emptyset$.

Proposition 13.4 *A pair (i, j) in a consistent IA network (X, C) is possibly intersecting iff the constraint $[i$ joint $j]$ is consistent with C, iff $C \not\models [i\,\{b, b'\}\,j]$.*

Proof When $[i$ joint $j]$ is consistent with the network, then $\hat{r}_{ij} \cap$ joint $\neq \emptyset$, hence there is a solution related by a primitive in joint. Conversely, if there is such a solution, then $\hat{r}_{ij} \cap$ joint $\neq \emptyset$. The second *iff* follows from the equivalence between $C \not\models [i\,\{b, b'\}\,j]$ and $\hat{r}_{ij} \cap$ joint $\neq \emptyset$. ■

Let us now generalize Definition 13.1 to a subset of interval variables.

Definition 13.2 A subset $I \subseteq X$ of interval variables is *necessarily intersecting* iff for every solution of the network $\bigcap_{i \in I}[i^-, i^+] \neq \emptyset$. The subset I is *possibly intersecting* iff there is a solution of the network for which $\bigcap_{i \in I}[i^-, i^+] \neq \emptyset$. ■

Proposition 13.5 *A subset of interval variables $I \subseteq X$ is necessarily intersecting iff every pair $i, j \in I$ is necessarily intersecting; I is possibly intersecting iff every pair in I is possibly intersecting.*

Proof The *only-if* part follows directly from the definition. Let us show the converse. Consider a solution in which the intervals $[i^-, i^+]$ are pairwise intersecting for all pairs in I. This means that the interval variables in I cannot be related in that solution with primitives b_{ij} or b'_{ij}. Hence the constraint $[i^+ < j^-]$, for any pair i and j in I, is inconsistent with the PA network corresponding for that solution to the set of end points of X. However, it is simple to show that the constraints $[i^- < j^+]$, for all pairs i and j in I, are consistent with this network. Let $\hat{\epsilon}^-$ be the biggest start point in I and $\hat{\epsilon}^+$ the smallest end point. We have $[\hat{\epsilon}^- < \hat{\epsilon}^+]$, $[i^- < \hat{\epsilon}^+]$, and $[\hat{\epsilon}^- < i^+]$ for every $i \in I$. Consequently, the intersection $\bigcap_{iC \in I}[i^-, i^+] \neq \emptyset$. ■

Proposition 13.5 enables us to reduce a global k-ary property of a subset of k interval variables to a local property of pairs.

13.3.3 A Geometric Interpretation of Interval Algebra

IA has a nice geometric model that helps in clarifying its relationship to PA. A temporal interval i is a geometric point in the Cartesian plane $(i^-, i^+) \in \Re^2$, with $i^- \leq i^+$. Conversely, a point (x, y) of the plane, with $x \leq y$, corresponds to a temporal interval. A set of points of the plane meeting $x \leq y$, such as a segment, a line, or a two-dimensional area, is associated with a set of temporal intervals. Hence, the relative position of two intervals can be described from the relative position of two geometric points in this plane.

In addition to a geometric point, six areas and six lines of the plane are defined with respect to an interval i, delimited with the x, y axes and the diagonal line $y = x$. These six areas, six lines, and a point are the thirteen zones labeled with the thirteen primitives of P (see Figure 13.14). Each zone delimits a set of intervals that have a particular relationship with i. For example, an interval j whose geometric mapping in the plane falls into the area labeled b with respect to i has its two end points before i^-, hence, by definition $[j\ b\ i]$. Similarly, an interval k whose geometric point falls in area d is such that $i^- < k^-$ and $k^+ < i^+$, i.e., $[k\ d\ i]$. Similar remarks explain the labeling of the other areas.

Among these zones, a line corresponds to a set of intervals that has one end point fixed as being equal to a end point of i and the other end point varying.

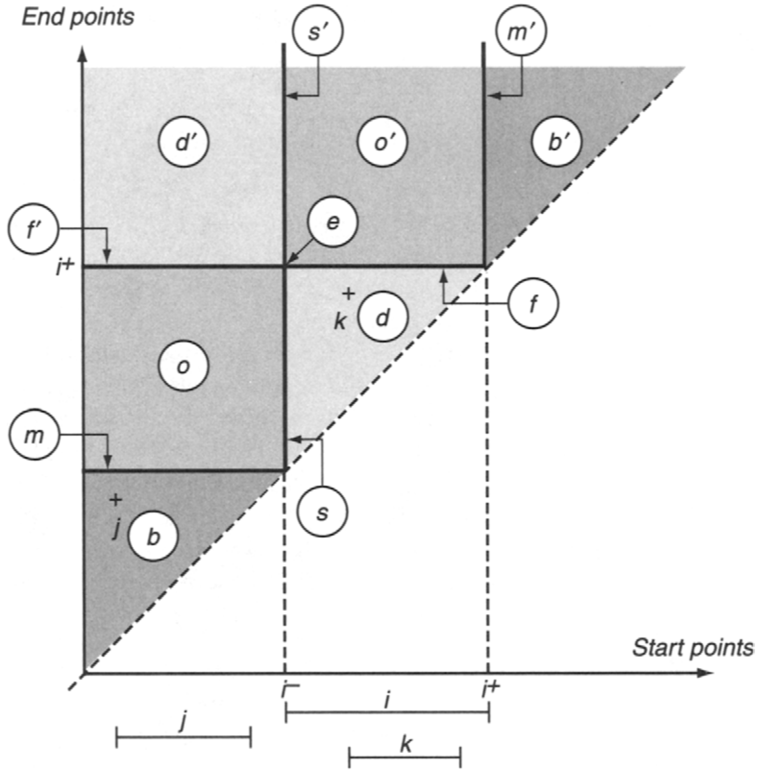

Figure 13.14 A geometric interpretation of interval algebra.

For example, the segment labeled s corresponds to the set of temporal intervals $\{j \mid j^- = i^-,\ j^+ > i^+\}$. A two-dimensional zone corresponds to a set of intervals whose two end points vary, e.g., the area labeled o corresponds to the intervals $\{j \mid j^- < i^-,\ i^- < j^+ < i^+\}$. Finally, the point labeled e is of dimension zero; it is the point i itself, and any interval mapped to this point is necessarily equal to i.

The position of an interval j that is not precisely known with respect to i can be modeled as one or several geometric zones (areas or lines) in the plane that are defined as the set of corresponding geometric points when j varies between some extreme positions. The intersection between these zones and the point, lines, and areas characteristic of i gives a disjunction of primitives corresponding to the constraint between i and j.

Example 13.6 In Figure 13.14, assume that an interval u starts with interval j and ends at some unknown point: it corresponds to a vertical half line starting at point j which crosses successively b, m, o, f', and d'; the constraint between u and i is thus $[u\ \{b, m, o, f', d'\}\ i]$. Similarly, an interval v that ends with k and starts at some unknown point will be described as a horizontal segment ending at point k that crosses the zones o, s, and d, hence $[v\ \{o, s, d\}\ i]$.

Consider an interval w that starts anywhere between the start points of j and k and ends between their end points, i.e., $j^- \leq w^- \leq k^-$ and $j^+ \leq w^+ \leq k^+$. Interval w corresponds to an area delimited by the equation $x \leq y$ and by the rectangle that has the line from point j to point k as one of its diagonals. This area intersects the zones labeled b, m, o, s, and d. Hence $[w\ \{b, m, o, s, d\}\ i]$. ■

It is interesting to note that the area associated with w in Example 13.6 is convex in the usual geometric sense, i.e., any point in a straight line between two points of the area belongs to it. Not every interval is mapped to a convex zone, i.e., an interval that is either entirely before or after i corresponds to the nonconvex union of the two zones labeled b and b'.

13.3.4 Interval Algebra versus Point Algebra

The convexity property of zones of the plane associated with intervals has an interesting and useful implication on IA. Let us define a graph G_{IA} that has 13 vertices labeled with the 13 primitives of P and that has an edge between two primitives iff the corresponding geometric zones are adjacent in the plane (Figure 13.15). A constraint $r \in R$ is said to be *convex* iff for any two primitives p_1 and p_2 in r all primitives along every shortest path in G_{IA} between p_1 and p_2 are also in r.

Example 13.7 $\{b, m, o, s, d\}$ is convex; $\{b, m, o, s, d, f\}$ is not convex because a minimal path from b to f goes through e and f'; but $\{b, m, o, s, d, f, e, f'\}$ is convex. ■

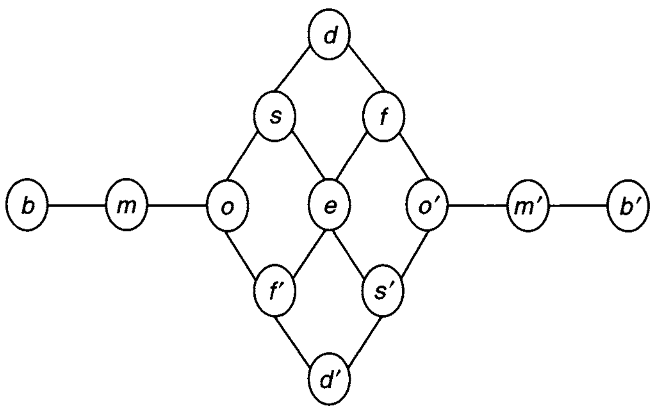

Figure 13.15 G_{IA} graph for convex constraints in interval algebra.

It can be shown that a constraint r is convex iff for every i and j such that $[j\ r\ i]$ the zone defined by j with respect to i is a convex zone in the plane.

As mentioned earlier, $\mathrm{IA_c}$ is the subset of the interval algebra restricted to convex constraints. Out of the 2^{13} constraints in IA, just 82 constraints are convex. $\mathrm{IA_c}$ is closed under the composition and intersection operations. Furthermore, checking the consistency of an $\mathrm{IA_c}$ network is a problem of polynomial complexity for which the path-consistency algorithm is complete and provides minimal networks.

A similar notion of convexity has been defined for PA. A constraint r in PA is said to be convex iff the set of points t that meet $[t\ r\ t_0]$, for t_0 fixed, is a convex part of $\Re$. It is easy to see that the only nonconvex constraint in PA is $\{<, >\}$. This explains the already introduced restriction of PA to the set $R - \{<, >\}$, denoted $\mathrm{PA_c}$, which is closed under the composition and intersection operations.

Each of the 82 constraints in $\mathrm{IA_c}$ can be expressed as a conjunction of $\mathrm{PA_c}$ constraints. For example:

$$[i\ \{b, m, o\}\ j] \equiv [i^- < i^+] \wedge [j^- < j^+] \wedge [i^- < j^-] \wedge [i^+ < j^+].$$

Hence an $\mathrm{IA_c}$ network can be translated into a $\mathrm{PA_c}$ network. It is no surprise that we have the same computational properties for $\mathrm{IA_c}$ as for $\mathrm{PA_c}$, namely, minimal nets obtained with path consistency.

Similarly, one may define another subset of IA by reference to the full PA. Let $\mathrm{IA_p}$ be the subset of IA constraints that can be expressed as conjunctions of PA constraints. For example:

$$[i\ \{b, o\}\ j] \equiv [i^- < i^+] \wedge [j^- < j^+] \wedge [i^- < j^-] \wedge [i^+ < j^+] \wedge [i^+ \neq j^-].$$

$\{b, o\}$ is an element of $\mathrm{IA_p}$, but clearly it is not in $\mathrm{IA_c}$ (from the last constraint $[i^+ \neq j^-]$, or directly from Figure 13.15, which shows that $\{b, o\}$ is not convex). Furthermore, there are elements of IA that are not in $\mathrm{IA_p}$. For example:

$$[i\ \{b, b'\}\ j] \equiv [i^- < i^+] \wedge [j^- < j^+] \wedge ([i^+ < j^-] \vee [j^+ \neq i^-]).$$

There are only 187 constraints in $\mathrm{IA_p}$. It is easy to show $\mathrm{IA_c} \subset IA_p \subset \mathrm{IA}$. The subset $\mathrm{IA_p}$ is also closed under composition and intersection. Because an $\mathrm{IA_p}$ network can be translated into a PA network, consistency checking for $\mathrm{IA_p}$ is a polynomial problem.

Let us mention finally that there exists other tractable subsets of IA, e.g., [163, 419], that are richer than $\mathrm{IA_p}$ and maximal for set inclusion.

13.4 Quantitative Temporal Constraints

The temporal primitive in this section is again the time point. We'll be developing a calculus for relating a set of instants with quantitative, absolute, and relative numerical constraints. We'll first consider the simple case where every constraint is a simple interval, and then we'll move to the general case where disjunctions of intervals are allowed.

13.4.1 Simple Temporal Constraints

Let $X = \{t_1, t_2, \ldots, t_n\}$ be a set of time-point variables; the domain of each t_i is the set $\Re$ of real numbers. Each t_i can be subjected to

- a unary constraint of the form $a_i \le t_i \le b_i$ and to
- binary constraints of the form $a_{ij} \le t_j - t_i \le b_{ij}$,

with $a_i, b_i, a_{ij}, b_{ij} \in \Re$.

Let us take an origin reference point t_0 such as to rewrite every unary constraint $a_i \le t_i \le b_i$ as a binary one: $a_i \le t_i - t_0 \le b_i$. Furthermore, in order to retrieve the general framework set in the two previous sections, let us denote a binary constraint on a pair (t_i, t_j) as a constraint $r_{ij} = [a_{ij}, b_{ij}]$ corresponding to an interval of reals.

A simple temporal constraint problem (STP) is a pair (X, C), where each element in C is an interval r_{ij} that constrains the relative distance of a pair of instants (t_i, t_j). The composition and intersection operations for STPs are defined as follows.

- Composition: $r_{ij} \bullet r_{jk} = [a_{ij} + a_{jk}\ ,\ b_{ij} + b_{jk}]$, which corresponds to the sum of the two constraints $a_{ij} \le t_j - t_i \le b_{ij}$ and $a_{jk} \le t_k - t_j \le b_{jk}$.
- Intersection: $r_{ij} \cap r'_{ij} = [max\{a_{ij} + a'_{ij}\}\ ,\ min\{b_{ij} + b'_{ij}\}]$ which denotes the conjunction of $a_{ij} \le t_j - t_i \le b_{ij}$ and $a'_{ij} \le t_j - t_i \le b'_{ij}$.

Note that the symmetrical of r_{ij} is $r_{ji} = [-b_{ij}, -a_{ij}]$.

An STP (X, C) is consistent if there is a solution that satisfies all the constraints. It is minimal if every point in an interval r_{ij} belongs to some solution. Constraints can be reduced with the familiar transitive closure propagation operation: $r_{ij} \leftarrow r_{ij} \cap (r_{ik} \bullet r_{kj})$.

The path-consistency algorithm is complete for STPs. Furthermore, the algorithm reaches a fixed point after a single iteration over all triples i, j, k (see the simplified version of the algorithm in Figure 13.16). This fixed point corresponds to the minimal network.

Example 13.8 Returning to the situation in Example 13.3 (see page 289), assume that the *Uranus* ship takes a light load and that the *Rigel* is serviced on a normal dock. This reduces the network of Figure 13.4 to a simple net, shown in Figure 13.17. We can apply the path-consistency algorithm to this network to discover that it is consistent and that the *Rigel* arrives one or two days after the arrival of the *Uranus* and departs two to three days after the *Uranus* departs. ■

It is interesting to note that the path-consistency algorithm verifies these nice properties of completeness and reaching a minimal network after a single iteration

```
PC($X, C$)
    for each $k : 1 \leq k \leq n$ do
        for each pair $i, j : 1 \leq i < j \leq n,, i \neq k, j \neq k$ do
            $r_{ij} \leftarrow r_{ij} \cap [r_{ik} \bullet r_{kj}]$
            if $r_{ij} = \emptyset$ then exit(inconsistent)
end
```

Figure 13.16 Path consistency algorithm for STPs.

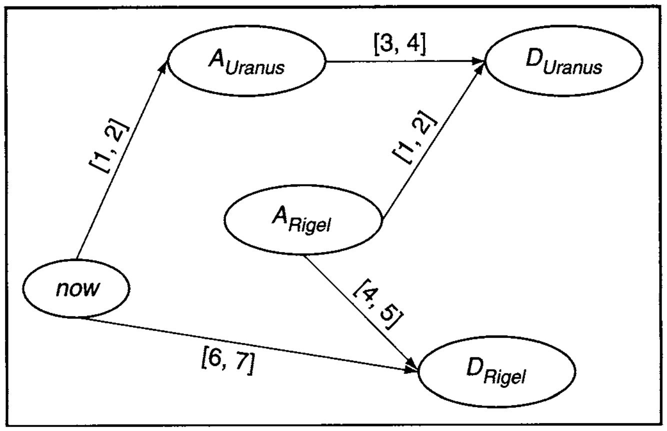

Figure 13.17 A simple temporal network.

whenever the transitive closure propagation uses a composition operation that distributes over an intersection, i.e., $r \bullet [r' \cap r''] = [r \bullet r'] \cap [r \bullet r'']$.

Distributivity is easily checked in STPs because

$$a + max\{a', a''\} = max\{a + a', a + a''\}$$

and similarly for *min*. Unfortunately, the distributivity property of $\bullet$ over $\cap$ does not hold, neither for PA nor for IA (see Exercises 13.9 and 13.10).

Another approach to consistency checking and finding minimal nets in STPs is the all-pairs minimal distance algorithm, Floyd-Warshall. Here, a network (X, C) is transformed into a distance graph $G = (X, E)$ as follows. Each constraint $r_{ij} = [a_{ij}, b_{ij}]$ of the network defines two labeled arcs in G: from t_i to t_j, labeled $l_{ij} = b_{ij}$, and from t_j to t_i, labeled $l_{ij} = -a_{ij}$. The algorithm (Figure 13.18) computes the minimal distances $d(i, j)$ between all pairs of vertices in G. The original network is inconsistent iff there is a negative cycle in G, i.e., a vertex i such that $d(i, i) < 0$.

```
Floyd-Warshall($X, E$)
    for each $i$ and $j$ in $X$ do
        if $(i,j) \in E$ then $d(i,j) \leftarrow l_{ij}$ else $d(i,j) \leftarrow \infty$
        $d(i,i) \leftarrow 0$
    for each $i,j,k$ in $X$ do
        $d(i,j) \leftarrow \min\{d(i,j), d(i,k) + d(k,j)\}$
end
```

Figure 13.18 Minimal distance algorithm.

Floyd-Warshall has a complexity of $O(n^3)$, as does PC. Path consistency remains more convenient when the network is incrementally maintained.

13.4.2 Temporal Constraint Networks

We consider here the general case of quantitative temporal constraint network problems, called TCSPs, where disjunctions of constraints are allowed on the distances between pairs of instants, as illustrated in Example 13.3 (see page 289). In a TCSP network, the constraint between a pair (t_i, t_j) corresponds to the disjunction $(a_1 \leq t_j - t_i \leq b_1)$ or ... or $(a_m \leq t_j - t_i \leq b_m)$. It is denoted $r = \{[a_1, b_1], \ldots, [a_m, b_m]\}$.

Union, intersection, and composition operations are easily generalized to these extended constraints. Let $r = \{I_1, \ldots, I_h\}$, $q = \{J_1, \ldots, J_l\}$, where $I_i = [a_i, b_i]$ and $J_j = [c_j, d_j]$ are intervals on the reals.

- Union: $r \cup q = \{I_1, \ldots, I_h, J_1, \ldots, J_l\}$.
- Intersection: $r \cap q = \{K_1, \ldots, K_p\}$, where $p \leq h + l$ and each $K_k = I_i \cap J_j$ whenever this intersection is not empty.
- Composition: $r \bullet q = \{K_1, \ldots, K_p\}$, where $p \leq h \times l$ and each $K_k = I_i \bullet J_j = [a_i + c_j, b_i + d_j]$.

The symmetrical constraint of r is $r' = \{[-b_1, -a_1], \ldots, [-b_m, -a_m]\}$.

The consistency and minimality of a TCSP network are defined as in the previous section. Here also the transitive closure propagation can be applied. However, the path-consistency algorithm is no longer complete. Nonetheless, this algorithm has been proved to terminate whenever the interpretation domain of time is the natural or the rational numbers and to provide a path-consistent network in a time complexity of $O(n^3 d^3)$, where n is the number of instants and d is the maximum range of any constraint. However, path consistency does not guarantee in general the consistency of the network.

A straightforward approach for solving a TCSP is the following.

- Decompose it into several simple networks by choosing one disjunct for each constraint.
- Solve each simple net by finding the corresponding minimal network.
- Combine the results with the union of the minimal intervals.

This systematic enumeration can be improved with a backtrack-search algorithm on a meta-CSP. Every variable in the meta-CSP is a constraint in the original TCSP, such as $r = \{I_1, \ldots, I_h\}$, whose domain is that set of intervals. Constraints of the meta-CSP are given implicitly by stating that any partial assignment should correspond to a consistent simple network. Each solution of the meta-CSP defines directly a solution of the temporal network. By using standard CSP heuristics and improvements, such as forward checking, backjumping, and intelligent backtracking, this backtrack-search algorithm can be reduced to a time complexity of $O(n^2 w^{|C|})$, where w is the maximum number of intervals in a disjunct.

13.5 Discussion and Historical Remarks

Time is an essential component for reasoning about action and change. Temporal reasoning has been extensively studied in AI from the viewpoint of the knowledge representation and nonmonotonic reasoning techniques for dealing with change, events, actions, and causality [12, 386, 473, 474, 475, 536].

This chapter and the rest of the discussion are focused on a relatively narrow aspect of temporal reasoning, namely, how to describe and manage consistently a set of temporal references to a set of propositions and objects and the constraints between these references. The propositions and objects are dealt with by a specialized system, such as a planner, while the task of the temporal reasoner is to maintain, through queries and consistent updates, the set of temporal references to these objects and their constraints. This task is of interest in planning and in several other applications areas, such as temporal databases [97], diagnosis [96], multimedia document management [2, 184], and process supervision [161].

As usual, the problem for this reasoning task is to find a good trade-off between the expressiveness of a representation and the computational complexity of handling it. The literature can be classified into approaches involving intervals or time points, qualitative or quantitative constraints, approaches integrating intervals and points, qualitative and quantitative models, and various generalization efforts, e.g., handling imprecise constraints or contingent temporal variables.

Interval algebra was introduced by Allen [16], who proposed a path-consistency filtering algorithm for handling it. Vilain and Kautz [537] introduced point algebra and showed that the consistency-checking problem of the IA is an NP-complete problem, while PA is a tractable problem. With van Beek, they considered the subset

PA_c and solved it by path consistency; they proposed a 4-consistency algorithm for the full IA calculus [538]. It turns out that an $O(n^2)$ algorithm is sufficient for the full IA [388]. Techniques such as minimal indexed spanning trees [225] enable an average-case linear performance for the incremental consistency checking of PA_c and IA_c. Efficient techniques for handling disjunction of constraints in PA have also been proposed [216].

A number of tractable subclasses of IA that are larger than IA_c and IA_p have been identified (e.g., [163, 419]). Starting from the work of Ligozat [363], several models integrating PA and IA have been proposed [242, 243], and their tractable subclasses have been characterized [287].

Early work on quantitative models, such as Malik and Binford [375], relied on linear equations and linear programming. The TCSP and STP models were introduced by Dechter *et al.* [152]. They are widely used, and several improvements have been proposed, e.g., to the STP management algorithms [114]. More recent approaches generalize these models to allow disjunctions of constraints on points, intervals, and durations [95, 250]. Uncertainty in quantitative temporal constraints has been studied by several authors [165, 166, 323, 534].

Among a number of models integrating qualitative and quantitative constraints, there are the proposals of Ghallab and Vidal [226], Meiri [389], and Pujari and Sattar [449]. Disjunctions of constraints in the integrated qualitative and quantitative framework are further introduced by Barber [47] within a general scheme called *labeled constraints* (where each disjunct is associated with a unique label).

An important issue for planning is that of *contingent* temporal variables, which are random variables that cannot be freely constrained, as normal CSP variables are, when propagating constraints and checking their consistency. For example, in a temporal network dealing with starting and ending points of actions, the former are controllable CSP variables, i.e., one controls the triggering point of actions, while the latter are contingent variables: they can be observed but usually they cannot be planned for, unless one assumes that there is a precise model of action durations. Models handling hybrid CSPs with controllable and contingent variables have been studied [533]; good solutions have been proposed [403, 530, 531, 532].

13.6 Exercises

13.1 Reformulate the path-consistency algorithm in Figure 8.5 and its incremental version in Figure 8.6 for PA networks. Filter the network of Figure 13.7 using this algorithm, and show that this network is path-consistent.

13.2 In Figure 13.7, relabel the constraint to be $[t_1 = t_4]$ and show that this leads to an inconsistent network.

13.3 Prove that the path-consistency algorithm is complete for PA networks.

13.4 Show that in PA the composition operation does not distribute over intersection, i.e., $r \bullet [r' \cap r''] \neq [r \bullet r'] \cap [r \bullet r'']$.

13.5 Perform Exercise 13.4 for IA.

13.6 Prove that the IA of Figure 13.13 is path-consistent but is not consistent.

13.7 Apply the path-consistency algorithm to the simple temporal constraint network of Figure 13.17. Show that the resulting network is minimal.

13.8 Modify the network of Figure 13.4 by considering that the *Uranus* ship takes a full load while the *Rigel* is serviced on a normal dock. Using the path-consistency algorithm, show that the network is inconsistent. Is it possible to meet all the constraints by serving the *Rigel* on an express dock while still providing a full load to the *Uranus*? If not, which constraint needs to be relaxed to provide a full load to the *Uranus*?

13.9 For the three PA constraints $r = \{<, >\}$, $r' = \{<, =\}$, and $r'' = \{=, >\}$, compare $r \bullet [r' \cap r'']$ to $[r \bullet r'] \cap [r \bullet r'']$.

13.10 Find three constraints r, r', and r'' in IA that provide a counterexample for the distributivity of $\bullet$ over $\cap$ in IA, i.e., $r \bullet [r' \cap r''] \neq [r \bullet r'] \cap [r \bullet r'']$.

13.11 Show that the composition operation does not distribute over intersection in general temporal constraint networks.

13.12 Prove that the consistency problem of general constraint networks is NP-hard. (Hint: Use reduction from the three-coloring problem.)

13.13 Detail explicitly the proof of Proposition 13.5 (see page 299) by showing (*ad absurdum*) that the constraints $[i^- < j^+]$, for all pairs i and j in I, are consistent with the PA network corresponding to the considered solution.