TIMEX – A Tool for Interval-Based Representation in Technical Applications

Jürgen Dorn
Technical University Vienna,
Christian Doppler Laboratory for Expert Systems,
Paniglgasse 16
A-1040 Vienna
email: dorn@vexpert.uucp

Abstract

TIMEX is a tool that extends Prolog with features for interval-based representation. Because several ideas for interval-based representation exist the tool provides the possibility to switch between different representation and propagation techniques. This is done by representing different interval graphs. So one application may consist of several graphs with different attributes for propagation and one interval may exist in different graphs. The application domain is technical expert systems.

1 Temporal Reasoning

In temporal reasoning different approaches for representation and propagation of constraints exist. The expressiveness of an interval-based representation stands in contrast to the complexity of the propagation. So the model of Allen [1] that supports full interval logic has the disadvantage that global consistency is an exponential problem. On the other hand in approaches based on time points some interval relations like {<, m, mi, >} are missing [2]. In scheduling and planning [3] these temporal constraints are often necessary.

In technical applications also a quantitative representation of time is needed. This may result in a system with duration constraints [4]. Through these additional constraints the interval graph may be more restrictive and the amount for propagation less. But this depends heavily on the number of duration constraints. So it depends on the application whether the use of duration constraints is advantageous.

We have described a new method of propagation using sequence graphs [5]. This technique has advantages if the application lasts for a longer time and the most frequently used interval relation is a sequence. In this case the amount of edges and with that the complexity of propagation is reduced significantly.

TIMEX is a tool based on Prolog that gives the possibility to use these different propagation techniques in one application. In TIMEX an interval graph is a structure that has attributes that indicate how propagation is to be done. So for one application a user can define different graphs with different features. Intervals of the application can be in different graphs simultaneous.

TIMEX is intended for expert systems in technical applications. We use an interval-based representation for a scheduling expert system in a steelmaking plant. The scheduling is done hierarchically and on the different levels of hierarchy different graphs are used. On an upper level of hierarchy qualitative constraints and no durations are used. On the lowest level also quantitative time values are necessary.

2 Basic Representation

Our knowledge representation technique is based on intervals. The user of TIMEX can choose whether an interval is only a symbol resp. a name or a structure consisting of three attributes: start, end and duration. To formulate temporal assertions in Prolog the “@”-operator is used. So the validity of an assertion is restricted in TIMEX to an interval. Different propositions are distinguished in [6] but we do not want to discuss this here.

temporal-assertion @ interval.

2.1 Granulation Intervals

To define intervals with attributes we introduce granulation intervals (G). Start and end of an interval are granulation intervals. The ending granulation interval is the first after the interval.
If we use a clock to measure time in an application we will get a discrete quantity. After every increment of this quantity there has been gone the time of one granulation interval. Granulation intervals are the smallest measurable time period.

Because TIMEX is intended for technical applications it must be accepted that the real interval in an application will deviate from the interval in the representation. The largest deviation of an interval in the representation from the real interval in the technical process will be one granulation at both sides of an interval. In many technical applications the granulation will be typical one millisecond. But in the steelmaking plant the granulation on the lowest level of hierarchy is 15 minutes. On higher levels of hierarchy a granulation of an hour is appropriate.

We model granulation intervals with a time axis that is build up like the natural numbers. All variables describing granulation intervals are represented thru the letter "G" and an index. The difference to natural numbers is only the interpretation as an interval.

We introduce a constant granulation interval "∞" called infinity which is element of the time axis. This element indicates that there exists a granulation interval that lies far in the future and no exact value is known. All other granulation intervals are smaller than "∞". The successor of this element is the element itself.

\[
\begin{align*}
A1: & \quad \forall G_1 (G_1 \in \text{time-axis} \rightarrow \\
& \quad \exists G_2 (G_2 = \text{suc}(G_1) \land G_2 \in \text{time-axis})) \\
A2: & \quad 0 \in \text{time-axis} \\
A3: & \quad \forall G (G \in \text{time-axis} \rightarrow 0 \neq \text{suc}(G)) \\
A4: & \quad \forall G_1, G_2 (G_1 \in \text{time-axis} \land G_2 \in \text{time-axis} \land \text{suc}(G_1) = \text{suc}(G_2) \rightarrow G_1 = G_2) \\
A5: & \quad \forall p (p(0) \land \forall G (G \in \text{time-axis} \land p(G) \rightarrow p(\text{suc}(G))) \rightarrow \forall G (G \in \text{time-axis} \rightarrow p(G))) \\
D1: & \quad \forall G (G < \infty \land \text{suc}(\infty) = \infty)
\end{align*}
\]

Fig 1: Axiomatic Base of Granulation Intervals

Addition, subtraction, and ordering relations are defined on granulation intervals. To simplify definitions and theorems two functions are defined that give the minimal resp. the maximal value of two granulation intervals.

\[
\begin{align*}
D2: & \quad 0 + G_1 = G_1 \land G_1 + \text{suc}(G_2) = \text{suc}(G_1 + G_2) \\
D3: & \quad G_1 \leq G_2 \leftrightarrow \exists G_3 (G_1 + G_3 = G_2) \\
& \quad G_1 < G_2 \leftrightarrow \exists G_3 (G_3 \neq 0 \land G_1 + G_3 = G_2) \\
D4: & \quad G_3 - G_1 = X \leftrightarrow (G_1 \leq G_3 \land G_1 + X = G_3) \\
D5: & \quad \text{min}(G_1, G_2) = \begin{cases} \\
G_1 & \text{for } G_1 \leq G_2 \\
G_2 & \text{for } G_1 > G_2 \\
\end{cases} \\
& \quad \text{max}(G_1, G_2) = \begin{cases} \\
G_1 & \text{for } G_1 \geq G_2 \\
G_2 & \text{for } G_1 < G_2 \\
\end{cases}
\end{align*}
\]

Fig 2: Definitions on Granulation Intervals

2.2 Time Bounds

We have mentioned that endpoints of intervals are granulation intervals. Because these attributes are often uncertain or incomplete they may be constrained by bounds. We speak of time bounds (B). An attribute of an interval can be constrained by a granulation interval to the upper or to the lower bound. The definition of a domain is a pair \( G_1 .. G_2 \) describing a range of possible granulation intervals. During propagation process the domain may be reduced further. An attribute is constrained to the entire time axis if the time bound is uncertain. We describe this by: 0 .. ∞. If the range consists only of one value the upper and the lower bound will be equal: \( G_1 .. G_1 \).

We deduce Addition, subtraction and the ordering of time bounds from the corresponding definitions of the granulation intervals. The intersection of two time bounds is the set of all granulation intervals that are in both time bounds. Intersections of time bounds are used to generate stronger constraints on absolute time specifications.

\[
\begin{align*}
T1: & \quad G_1 .. G_2 + G_3 .. G_4 = G_1 + G_3 .. G_2 + G_4 \\
T2: & \quad G_1 .. G_2 - G_3 .. G_4 = \\
& \quad G_1 - G_3 .. G_2 - G_4 \land G_3 \leq G_1 \land G_4 \leq G_1 \\
T3: & \quad G_1 .. G_2 < G_3 .. G_4 = G_1 < G_3 \land G_2 < G_4 \\
& \quad G_1 < G_2 \land G_3 \leq G_4 \land G_2 \leq G_4 \\
D6: & \quad \forall G_2, G_3 (G_2 < G_3) \rightarrow \forall G_1 (G_1 .. G_2 \cap G_3 .. G_4 = \text{max}(G_1, G_3) .. \text{min}(G_2, G_4))
\end{align*}
\]

Fig 3: Definitions for Time Bounds

2.3 Intervals

Intervals are like granulation intervals and time bounds objects in our logic calculus. By means of classification of objects we get a representation in which we can show simply which propositions are syntactically correct. An interval is an object consisting only of a name or of a name and three time bounds.
The *begin* of an interval is a time bound that indicates how many granulation intervals since the time 0 must have gone until the interval starts earliest resp. latest. The *end* of an interval is a time bound that indicates how many granulation intervals since the time 0 must have gone until the interval starts earliest resp. latest. The last granulation interval is not inside the interval. This convention is due to simplifications of subsequent definitions. The *duration* is a time bound that indicates how many granulation intervals the interval minimal resp. maximal contains.

From begin and end of an interval the duration can be deduced. The specification of a third attribute is always redundant. But we suppose that interval specifications are often incomplete. Perhaps we know about a process how long it continues but not when it will begin or end. If the process begins actually the end can be determined with the duration.

\[ \text{A6: begin}(I) + \text{duration}(I) = \text{begin}(I) \]

**Fig 4:** Duration Constraint

Using time bounds an interval specification is a set of possible intervals.

### 2.4 Time Constraints

A *time constraint* is a set that describes the possible relation between two time bounds. Between time bounds two orderings are defined “<”, “≤”. If we add the two inverse relations “≥”, “>” and the equality “=” we get a set of five possible relations. The set \{<, >\} is inconsistent and no possible time constraint. In [7] it is shown that without this restriction the propagation needs 4-consistency. We need only 3-consistency. This time-point based representation is called *continuous* in [7]. If no proposition about the relation is known the time constraint is *unknown* “–”.

To check consistency operations are needed that generate intersection “∩” and union “∪” of two given time constraints. The set of possible time constraints form a lattice with an unit element “–” and a zero element “Ø”.

**Fig 5:** Hasse-Diagrams

Intersection of time constraints is defined by the supremum and the union by the infimum in the Hasse-diagram above. The zero element “Ø” does not belong to the set of possible time constraints. The zero element is used for contradictions. If it is deduced there is an inconsistency in the knowledge base. This lattice is illustrated by the first Hasse-diagram in figure 5. The transitive time constraint is defined by the supremum in the second Hasse-diagram.

### 2.5 Interval Constraints

There are 13 disjunctive simple relations between intervals possible [1]. Through disjunction of simple relations more complex relations can be formulated. In TIMEX two possibilities exist for the definition of interval constraints. The relations could be defined per se like Allen has done or they will be defined on base of the constraints between granulation intervals.

The operations on them depend on this definition. We show here only the definition of *interval relations* on base of time constraints between the time bounds that constrain the interval. An interval relation is an object consisting of two interval names and an *interval constraint*.

An interval constraint is an object consisting of four elements (four time constraints) that describes the relations between the four endings.

**Fig 6:** Representation of Interval Constraints

We generate disjunctions of interval constraints thru disjunctions of time constraints. If one of the four components of an interval constraint is unknown this yields in a disjunction of basic interval constraints.

So the first element of an interval constraint describes the relation of the two corresponding beginning granulation intervals. The following interval relation \(I_1 \{=} <, \leq, \geq, –\} I_2\) states e.g. that the beginning endpoint of interval \(I_1\) is before that of \(I_2\). If the other time constraints are unknown the interval constraint describes Allen’s \{=, s, si\}.

Every component of an interval constraint is an element of \{<, =, >, ≤, ≥, –\}. To determine transitive interval constraints, set operations for interval constraints are introduced. These Operations are based on time constraints.

We define a special subset of interval constraints. An *interval sequence constraint*, states that two intervals must occur in sequence.
**Fig 8: Definition of Sequence Constraint**

The transitive interval constraint is explained by transitive time constraints.

**Fig 7: Transitive Interval Relation**

To improve the readability some predicates are introduced. The following table shows all predicates we have defined for our application. The inverse constraints are not necessary on the representation level.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Allen</th>
<th>Time Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>meets(I₁, I₂)</td>
<td>m</td>
<td>(&lt;/, &lt;, =)</td>
</tr>
<tr>
<td>before(I₁, I₂)</td>
<td>&lt;</td>
<td>(&lt;/, &lt;, &lt;)</td>
</tr>
<tr>
<td>sequence(I₁, I₂)</td>
<td>{&lt;, m}</td>
<td>(&lt;/, &lt;, &lt;)</td>
</tr>
<tr>
<td>in(I₁, I₂)</td>
<td>{=}</td>
<td>(≥, ≤, &lt;)</td>
</tr>
<tr>
<td>equal(I₁, I₂)</td>
<td>=</td>
<td>(=, =, &lt;)</td>
</tr>
<tr>
<td>during(I₁, I₂)</td>
<td>d</td>
<td>(&gt;, &lt;, &lt;)</td>
</tr>
<tr>
<td>starts(I₁, I₂)</td>
<td>s</td>
<td>(&gt;, &lt;, &lt;)</td>
</tr>
<tr>
<td>finishes(I₁, I₂)</td>
<td>f</td>
<td>(&gt;, =, &lt;)</td>
</tr>
<tr>
<td>overlaps(I₁, I₂)</td>
<td>o</td>
<td>(&gt;, =, &lt;)</td>
</tr>
<tr>
<td>initiate(I₁, I₂)</td>
<td>m</td>
<td>(&lt;/, &lt;, =)</td>
</tr>
<tr>
<td>follows(I₁, I₂)</td>
<td>m, o</td>
<td>(&lt;/, &lt;, =)</td>
</tr>
<tr>
<td>intersects(I₁, I₂)</td>
<td>o, d</td>
<td>(&lt;/, &lt;, ≥)</td>
</tr>
</tbody>
</table>

**Fig 9: Definition of Interval Constraints**

3 Interval Graphs

An interval graph is an ordered set with four elements consisting of a name, a finite set of intervals, a finite set of interval relations, and a set of attributes. The attributes decide the propagation in the graph.

All intervals of an interval graph are connected with each other. Two intervals are connected if a path between both points in the interval graph exists.

A :ib:graph!:sequence -!:graph is an interval graph that is not complete because the properties of a sequence chain are used to reduce the number of edges in the graph. A sequence chain is a path in an interval graph where all constraints are sequence constraints.

The special property of a sequence chain is: “If an explicit constraint between two intervals exists then there is no knowledge about an third interval that occurs between them.

**Fig 10: Property of Sequence Graphs**

Since sequence graphs are used the query for interval constraints is not so easy as in a complete interval graph. If there is a query for a constraint between two intervals and these intervals are connected by an edge of the sequence graph there will be a path in the graph between the two intervals. If a constraint is looked for that is not represented explicitly by an edge in the graph there are two directions to look for – into the future or into the past.

Suppose we are looking for the constraint between the intervals I₁ and I₂. If there is no explicit edge between the two corresponding points in the graph we have two possibilities – I₁ is before I₂ or vice versa. If the domains of the endpoints are strongly restricted we can decide it through a comparison of the ranges. Otherwise we have to search. Starting from one point we can search in two directions – forward or backward in the sequence graph. If we have decided one direction and this direction is right we will never have to backtrack. We have to search in the chain until we have found an interval between the two intervals.

If we use sequence chains some propagations are not needed because we know that there will be a stronger constraint. This property of a sequence chain is called monotocity. The following theorem describes this property.

**Fig 11: Monotocity of Sequence Constraints**

\[
\forall C_1, C_2, C_3 \ [\text{sequenceConstraint}(C_1) \land \text{sequenceConstraint}(C_2) \rightarrow \\
\text{transitive(transitive(C_1, C_2), C_3)} \subseteq \\
\text{transitive(C_1, transitive(C_2, C_3))} \land \\
\text{transitive(C_3, transitive(C_1, C_2))} \subseteq \\
\text{transitive(transitive(C_3, C_1), C_2))}]
\]
4 Propagation of Constraints

Propagation of constraints is a widely used technique. In principle we use a technique described by Aho [8] for the computation of transitive closure in graphs.

Also Allen has used this algorithm. We will show in the following two sections the special techniques for quantitative constraints and for sequence graphs.

4.1 Quantitative Constraints

Sometimes no quantitative values for intervals are known. Then it is better not to use the quantitative representation feature. If this feature is switched off for a graph intervals are only represented by their names.

But if we use the quantitative representation the attributes of an interval are constrained by the equation begin(I) + duration(I) = end(I). Each attribute is described by a domain. If an interval is added first the duration constraint will be examined. Perhaps the domains will be reduced. The constraint resolution is done with the theorems of addition and subtraction for time bounds.

If now an interval constraint is added into a graph the attributes of both intervals may be reduced further. So if the two intervals shall meet the meeting granulation interval must be same. We can decide this with the four time constraints in an interval constraint.

The propagation is rather simply, but we have to decide, if we want to propagate further. If an interval attribute is constrained to one definitive value we could look for other intervals with attributes with the same definitive value. Now we could conclude a time constraint between the two intervals.

This search for an interval is a difficult task with much overhead. So we give an user the possibility to switch off this further propagation separately for a graph.

4.2 Qualitative Constraints

The basic algorithm for propagation is similar to the algorithm for transitive closure in networks of Aho [8]. An edge (interval relation) between two nodes N1 and N2 (two intervals) shall be added to a graph. This pair is added first as a task to an agenda. If the execution of the task is successful, the edge is inserted in the graph. In the process of execution this task other tasks may be produced. These are added to the agenda.

All tasks of an agenda are performed in a loop. For a task (an edge in the graph) a set of all explicit connected nodes (intervals) are generated. For every such node N3 the transitivity rule is applied.

If the new computed edge resp. the interval constraint is stronger than the old one, the new constraint is added to the graph. This results also in a new task on the agenda. The propagation algorithm terminates if no further tasks on the agenda exist. Because tasks are only added if stronger constraints are added, the termination is safe.

The propagation algorithm with sequence graphs is mostly the same as with complete interval graphs. If a stronger constraint is added to the graph we distinguish whether the new constraint is a sequence constraint or not. If it is not, normal propagation is done. If the new constraint is a sequence constraint, we generate two interval sets. The “BeforeSet” includes all intervals connected explicitly by a sequence constraint with I1 and the “AfterSet” includes all intervals connected explicitly by a sequence constraint.

\[
\text{newConstraint} = \text{sequence}(N_1, N_2) \Rightarrow
\]
\[
\text{BeforeSet} := \{ N_3 | \text{sequence}(N_3, N_1) \} \land
\]
\[
\text{AfterSet} := \{ N_4 | \text{sequence}(N_2, N_4) \}
\]

Fig 12: “BeforeSet” and “AfterSet”

The following figures illustrate this. Suppose we have the following constellation:

\[
\begin{align*}
\text{Fig 13: Example} \\
\text{The constraints between these intervals are described in the following graph.}
\end{align*}
\]

Fig 14: Sequence Graph

Suppose we constrain I3 to be before I4. Then the “BeforeSet” consists of I1 and I2 and the “AfterSet” of I5. Now all edges between members of the “BeforeSet” and members of the “AfterSet” can be deleted. But there are further edges to be deleted. All edges between mem-
bers of the “BeforeSet” and $I_4$ are deleted. Also all edges between members of the “AfterSet” and $I_3$ may be omitted.

![Reduction Sequence Graph](image)

**Fig 15:** Reduced Sequence Graph

If no sequence constraint is added to the graph only the relation has to be added, a new task has to be generated and the propagation procedure is to be called.

## 5 Implementation

A first version of TIMEX was implemented on base of AAIS-Prolog [9] and was used by a planning system described in [6]. A second version is implemented in Quintus Prolog [10] on a Sparc Station. Tests with different graph propagation techniques have shown that the propagation with sequence graphs results in significantly faster propagation for most examples. Typical examples are applications in the steelmaking plant with ten heats and with each heat consisting of about eight intervals. A full description of the implementation is found in [11]. Now it is intended to develop a version in C.

### References

“Maintaining Knowledge about Temporal Intervals”.  

“Constraint Propagation Algorithms for Temporal Reasoning”.  

“Planning Using a Temporal World Model”.  

“Plan Generation in a Temporal Frame”.  

“Propagation of Intervals in Sequence Graphs”  

“Knowledge-Based Real-Time Planning”  
Ph. D. Technical University Berlin, 1989


“The Design and Analysis of Computer Algorithms”.  

Advanced A. I. Systems, Inc.

Version 10, QuintusProlog Release 2.5  
Quintus Computer Systems

“TIMEX”  