FORMALIZING AND USING PERSISTENCY

Thomas Guckenbiehl
Fraunhofer-Institute of Information and Data Processing (IITB)
Fraunhoferstr. 1
D-7500 Karlsruhe
FRG
guc@iitb.fhg.de

Abstract

In many formalizations of a changing world things do not change all the time but are persistent thoughout a time interval. Often this persistency is represented by facts refering to intervals int which are still valid if int is replaced by any subinterval int' Approaches like episode propagation or Penberthy's temporal unification try to employ this property for efficient reasoning. However these approaches lack formality. In this paper their way of reasoning about persistency is reconstructed as inference rules that combine appropriate time-boxes with standard resolution. In many cases Burckert's Constrained Resolution may be used. More complex examples may be handled by a new inference rule, called Persistency Resolution. An analysis of this rule leads to a more general notion of persistency.

1 Introduction

During the last decade temporal reasoning has turned out to be crucial for intelligent systems acting in the real world. This resulted in a lot of work on how to represent temporal knowledge and how to reason with it. With respect to reasoning one may distinguish at least four levels on which work has been done:

- *Reasoning about the temporal structure* is concerned with the handling of constraints between time points or intervals by so called *time-boxes* (e. g. [Allen, 1983; van Beek, 1989])
- *Temporal Data Base Management* explores efficient techniques for storage of facts like "She was at home between 2 and 4 o'clock" and answering questions like "Was she at home between 3 and 3.30?" (e. g. [Kahn and Gorry, 1977]; Dean and McDermott, 1987]).
- *Temporal Elaboration* tries to infer monotonically new facts by applying general knowledge about a domain to facts from a temporal data base (e g. [Decker 88]).
- *Temporal Extension* uses additional heuristics like the Closed World Assumption or the Persistency-by-default Assumption to infer facts that are not implied monotonically by the knowledge base (e g. [Dean and McDermott, 1987; Shoham, 1987], traditional discrete-event simulation).

This paper is concerned with temporal elaboration of statements that incorporate something like TRUE (O,int) to express that some property Oholds throughout some time interval int. I will use TRUE(O,o), if int is the single element set {to}. The semantics of such facts, which have been popular in AI since the work of McDermott [1982] and Allen [1983], has been clarified e. g. by [Galton, 1990]. To distinguish them from general facts in a knowledge base they will be called *persistence facts* in the sequel.

Although this representation is adequate and intuitive in many domains, it cannot be handled efficiently by standard inference procedures. The combination of a time-box and resolution is more adequate, as demonstrated in the next section.

2 Problems with Persistent Facts

Consider e. g. the following rule from the description of an AND-gate:

(1) TRUE (in_1 1 ,T) A TRUE(in_2 = 1 ,T)
=> TRUE (out = I ,T)

If we know the persistence facts

(2) TRUE (in_1= 1 ,[0, 100])

(3) TRUE(in_2= 1 ,[50, 200])

about the inputs, we would like to infer

(4) TRUE (out = 1 ,[50, 100]).

Hyperresolution using standard unification does not work, since the matchers {T<-[0,100]} and {T<- [50 , 200]} arising from matching the two conditions to (2) and (3) resp. are inconsistent.

A first order approach might handle the problem representing persistency by some axiom

(PA) int' *Q* int A TRUE (o , int) => TRUE (O , int')

Then knowledge about set-inclusion may be used to apply this axiom to (2) and (3), infering TRUE (in_1 = 1 , [50 , 100]) and TRUE (in_2 = 1 , [50 , 100]). Finally the reasoning system may apply (1) to these new facts to infer (4). But how does it get to know that inf in (PA) should be instantiated to [50,100] and not to some other interval? In particular, there is an infinite set of possible instantiations which satisfy both conditions of (1).

It is not clear to us if Time Map Management System like that of Dean can handle such problems. Following the examples given in [Dean and McDermott, 1987], Dean's TMM seems to proceed as follows: while matching the first condition to (2), it unifies T with [0 ,100]. Then it looks for a fact that satisfies the second condition throughout this interval. Hence it would not be able to use (3) and infer (4).

There have been more informal approaches capable to do the desired reasoning, e. g Episode Propagation (c. f. [Williams, 1986; Decker, 1988]) and Temporal Unification (c. f. [Penberthy, 1987]).

An episode propagator is a constraint propagator where the values of constraint variables are interpreted as sets of episodes from the history of an attribute. An attribute a is a function on time points, and an episode may be represented as a persistence fact TRUE(a= v,int). Single episodes are propagated between histories and constraints which, like (1), relate different episode patterns.

[Penberthy, 1987] proposed *Temporal Unification,* a special purpose unification procedure for atoms containing intervals as arguments. If the atom is persistent with respect to this argument, Penberthy calls it a fact interval and the atom a fact. Otherwise it is called an event interval and the atom an event. Temporal Unification of intervals int_1 and int_2 yields their intersection if both are fact intervals or the same event interval. If only one of them, say int_L, is a fact interval and int_2 c int_1, the unifier is int_2. In all other cases the intervals are ununifiable

This paper extracts the key ideas behind these ad hoc approaches and casts them into formal inference rules that combine standard resolution with appropriate constraint reasoners For instance Constrained Resolution [Burckert, 1990] may be used in the first example. More complex examples may be handled by a new inference rule called Persistency Resolution. A closer look at this rule reveals that it is not restricted to persistence facts but may also be used with statements about processes like "The car is moving" and even nontemporal statements like "It rained throughout the country".

After giving some notation, section 4 presents a special formalization of Persistence facts, on which sections 5 and 6 base Constrained Resolution and Persistency Resolution. A discussion of a more general notion of persistency closes the paper.

3 Notation

In contrast to tense logical languages (c. f. [Rescher and Urquhart, 1971]), which contain modal operators like G' or 'H', variable and constant symbols will be used to denote time-points and intervals as well as functions and relations on them explicitly. Only such interpretations are allowed that give these symbols their natural extension. As syntactical variables for expressions of our language 'trm' represents an arbitrary term, symbols starting with capital letters denote variables, while constants start with lower case letters. In particular 'T'denotes a time-point, 'x' may denote an interval or a time-point, Y symbolizes a time-point constant and 'int' an interval constant. Small greek letters E_i are used for atoms with a nontemporal topsymbol and large greek letters for arbitrary formulae. Small greek letters o represent substitutions, expr[Xj ,..., x_n] symbolizes that expr contains at most the pairwise different temporal variables $x_1 .... x_n$ (though there may be other nontemporal ones). All variables are assumed to be universally quantified, and renaming of variables will be avoided if possible.

4 Qualified Formulae

A closer look at the example from section 2 reveals that standard inference techniques *are* inefficient because standard unification is unable to process the information implicit in statements TRUE (O, int): that O holds in all time points T contained in int and therefore TRUE (O, int') is also true for *every* subinterval inf of int. Matching the first condition of (1) to (2) constrains the valid assignments to T to elements of [0,100], while matching the second condition to (3) further constrains them to elements of [50,200]. Our informal reasoning joins both constraints to derive (4).

Therefore I propose to make the information implicit in TRUE(O,trm) explicit by formalizing this expression as

(5) V T: (T 6 trm => O[T]) , if trm denotes an interval

(6) {T<-trm} *[T] , if trm denotes a time point

where T is a new variable and ⊹ modifies *' to mirror its dependency on time (e. g. by additional temporal arguments).

For instance we may formalize our example from section 2 as

(V) -^(in_l(T)=l) V -»<in_2(T)=l) v<out(T)=l)

(2') T'6[0,100] => in_1(T')=l

(3^1) T'€[50,200] => in_2(T')=l

More generally, trm may be characterized as the solution-set of a constraint $\Psi_C[T]$, e. g. $(T \in int_1 \wedge T \in int_2)$ or $(T = lb(int))$ where lb denotes the greatest lower bound. Furthermore we are not restricted to a single temporal variable. Hence (5) is an instance of

(7) $\Psi_C[X_1, \ldots, X_n] \Rightarrow \Phi[X_1, \ldots, X_m]$, with $m \leq n$.

In the sequel a formula that contains the TRUE-operator will be called a *persistence formula*. Furthermore a formula like (7) will be called a ***qualified formula*** (or *q-formula*) with ***qualification*** Ψ_C and ***kernel*** Φ. A ***negative q-formula*** is a formula $\neg(\Psi_C \Rightarrow \Phi)$ and a ***qualified literal*** (or *q-literal*) is either a positive or a negative q-formula. If the kernel is an atom we talk about *qualified atoms* (or *q-atoms*). If it does not contain any nontemporal variables, it will be called a *qualified fact* or *q-fact*.

Although in modified form, qualified facts have been around in AI for quite some time. For instance the propositions in an ATMS (c. f. [deKleer, 1986]) may be viewed as qualified by their label, i. e. a disjunction of sets of assumptions. Another example are clauses of Constraint Logic Programs (c. f. [Jaffar and Lassez, 1986]). Recently they have been generalized by [Bürckert, 1990] in the form of Constrained Clauses, which correspond to positive q-formulae.

5 Constrained Resolution

Seemingly, previous work has only dealt with positive q-formulae $\Psi_C \Rightarrow \Phi$. One reason is probably that such formulae can be rewritten as $\neg\Psi_C \vee \Phi$. This expression may be regarded as a single clause, if Φ is in clause-form and inference is done using combinations of special purpose reasoners for the qualification and general techniques for the kernel. [Hrycej, 1988] for instance integrates a time box for Allen's interval calculus with Prolog along these lines. [Bürckert, 1990] has generalized such combinations by an inference rule called *Constrained Resolution*:

(CR)
$$\frac{\begin{array}{lll} \Psi_C & \Rightarrow & [\ \xi \vee \Phi\ ] \\ \Psi_C' & \Rightarrow & [\neg\xi' \vee \Phi'\ ] \end{array}}{\begin{array}{l} \text{if } \sigma(\Psi_C \wedge \Psi_C' \wedge trm_1 = trm_1' \wedge \ldots \wedge trm_n = trm_n') \\ \text{is satisfiable:} \\ \sigma(\Psi_C \wedge \Psi_C' \wedge trm_1 = trm_1' \wedge \ldots \wedge trm_n = trm_n') \\ \qquad \Rightarrow \sigma[\Phi \vee \Phi'], \end{array}}$$

where σ is the most general unifier of ξ and ξ' and trm_i, trm_i' are corresponding temporal terms in ξ and ξ'.

Bürckert proves that, given some appropriate constraint theory, this rule is sound and complete. One easily checks that applying this rule to (1'), (2') and (3') yields

(4') $(T \in [0, 100]) \wedge (T \in [50, 200]) \Rightarrow out(T) = 1$.

Final simplification of the qualification by the constraint reasoner results in the desired q-fact

(4'') $(T \in [50, 100]) \Rightarrow out(T) = 1$

In many situations Constrained Resolution is a good formalization of reasoning about persistency. For instance if we include some delay of 30 ns into our model of the AND-gate, replacing (1) by

(8) $TRUE(in_1 = 1, T) \wedge TRUE(in_2 = 1, T)$
$\Rightarrow TRUE(out = 1, T+30)$,

Constrained Resolution still works.

However it fails for persistence formulae whose clause-form contains some negated persistence atom $\neg TRUE(\Phi, int)$. Consider e. g. the set-up times for data inputs to a flip/flop. This is the minimal time span prior to a clock pulse for which an input has to stay constant. It prevents spurious changes of the inputs during a clock pulse from influencing the output. Formulating this for a D-flip/flop with 200 ns set-up time and 50 ns delay yields

(9) $TRUE(d = V, [T-200, T]) \wedge TRUE(pulse = occuring, T)$
$\Rightarrow TRUE(q = V, T+50)$

with the clause-form

(9') $\neg TRUE(d = V, [T-200, T])$
$\vee \neg TRUE(pulse = occuring, T)$
$\vee TRUE(q = V, T+50)$.

Considering this rule and the facts

(10) $TRUE(d = 0, [0, 400])$
(11) $TRUE(pulse = occuring, 100)$
(12) $TRUE(pulse = occuring, 300)$

we should be able to derive that the rule is not applicable to (10) and (11) but to (10) and (12), yielding

(13) $TRUE(q = 0, 350)$.

The problems come in as we try to change these formulas into constrained clauses by introducing explicit qualifications:

(9'') $\neg \forall T_1 : (T_1 \in [T-200, T] \Rightarrow d(T_1) = V)$
$\vee \neg(pulse(T) = occuring)$
$\vee q(T+50) = V$

(10') $T' \in [0, 400] \Rightarrow d(T') = 0$
(11') $pulse(100) = occuring$
(12') $pulse(300) = occuring$.

Obviously, the first disjunct of (9''), which resulted from the first negated persistence literal in (9'), corresponds to two clauses. To apply Constrained Resolution, we have to introduce the new skolem-function f and transform (9'') into clause-form. This results in two constrained clauses

(9''a) $f(T, V) \in [T-200, T]$
$\vee \neg(pulse(T) = occuring) \quad \vee q(T+50) = V$

$$(9''b)\quad \neg(d(f(T,V))=V) \vee \neg(\mathrm{pulse}(T)=\mathrm{occuring}) \vee q(T+50)=V.$$

Applying (CR) to (9''b) and (10') yields

$$(14)\quad f(T,0)\in[0,400] \Rightarrow [\neg(\mathrm{pulse}(T)=\mathrm{occuring}) \vee q(T+50)=0].$$

Further application of (CR) to (14) and (12') gives after simplification by the time-box:

$$(15)\quad f(300,0)\in[0,400] \Rightarrow [q(350)=0].$$

On the other hand applying (CR) to (9''a) and (12') results in

$$(16)\quad \neg(f(300,V)\in[100,300]) \Rightarrow [q(350)=V].$$

To derive $q(350)=0$, as we intuitively did, we have to combine the qualifications of (15) and the instance of (16) under $\{V\leftarrow 0\}$ and recognize that every interpretation of $f(300,0)$ satisfies at least one of them. However this is beyond the scope of Constrained Resolution.

To handle such examples we have to look for another inference rule.

6 Qualified Resolution and Persistency Resolution

The problem in applying (CR) to our last example arises because the splitting of (9'') results in spreading the constraints relevant to our informal reasoning over the two clauses (15) and (16). The reason for splitting (9'') was that Constrained Resolution resolves standard literals in constrained clauses, but cannot deal with negative qualified literals. So the basic idea for a more suited inference rule is to allow for the resolution of such negative q-literals to positive ones. This leads to *Qualified Resolution*:

$$\textbf{(QR)}\quad \begin{array}{l} \neg(\Psi_C \Rightarrow \xi) \vee \Phi \\ (\Psi_C' \Rightarrow \xi') \\ \hline \text{if } \neg\sigma[(\Psi_C \Rightarrow \Psi_C') \wedge \mathrm{trm}_1=\mathrm{trm}_1' \wedge \ldots \wedge \mathrm{trm}_n=\mathrm{trm}_n'] \text{ is satisfiable:} \\ \sigma[(\Psi_C \Rightarrow \Psi_C') \wedge \mathrm{trm}_1=\mathrm{trm}_1' \wedge \ldots \wedge \mathrm{trm}_n=\mathrm{trm}_n'] \Rightarrow \sigma\Phi, \end{array}$$

where σ is a most general unifier of ξ and ξ' and trm_i, trm_i' are the corresponding temporal terms of ξ and ξ', resp. (A correctness proof for this inference rule may be found in [Guckenbiehl, 1990].)

Since the residuum constraint does neither correspond to a single clause nor to a q-literal, it cannot be handled by standard or Qualified Resolution. Hence like Constrained Resolution this inference rule requires an additional constraint reasoner. I do not know yet under which conditions provision of a (semi-) decision algorithm for constraints guarantees (refutation-) completeness of the whole inference rule.

Persistency Resolution

Even if constraint-satisfiability is decidable, it will in general be computationally expensive, since the residuum constraint is not atomic and therefore the constraint resulting from multiple applications of (QR) is not a single clause. Fortunately there is a specialization of Qualified Resolution, which I call *Persistency Resolution*, that yields such a residuum.

In the first place it requires that q-literals have the form

$$[T_1 \in \mathrm{trm}[X_1,\ldots,X_n] \wedge \ldots \wedge T_m \in \mathrm{trm}[X_1,\ldots,X_n] \Rightarrow \xi[T_1,\ldots,T_m],$$

where $T_1,\ldots,T_m$ do not appear outside the q-literal. We may restrict the discussion to $m=1$, since the generalization to $m>1$ is just a combination of the methods explored in the sequel. Then we have:

$$\textbf{(PR)}\quad \begin{array}{l} \neg(T_1 \in \mathrm{trm}[X_1,\ldots,X_n] \Rightarrow \xi[T_1]) \vee \Phi[X_1,\ldots,X_n] \\ (T_1' \in \mathrm{trm}' \Rightarrow \xi'[T_1']) \\ \hline \text{if } \sigma[(\mathrm{trm}[X_1,\ldots,X_n] \subseteq \mathrm{trm}') \wedge \mathrm{trm}_1=\mathrm{trm}_1' \wedge \ldots \wedge \mathrm{trm}_n=\mathrm{trm}_n'] \text{ is satisfiable:} \\ \sigma[(\mathrm{trm}[X_1,\ldots,X_n] \subseteq \mathrm{trm}') \wedge \mathrm{trm}_1=\mathrm{trm}_1' \wedge \ldots \wedge \mathrm{trm}_n=\mathrm{trm}_n'] \\ \qquad \Rightarrow \sigma\Phi[X_1,\ldots,X_n], \end{array}$$

where σ is a most general unifier of ξ and ξ' which unifies T_1 and T_1', and trm_i, trm_i' are the corresponding temporal terms of ξ and ξ', resp. The simple correctness proof is given in [Guckenbiehl, 1990], but as with the general case I have no completeness results.

Persistency Resolution in the examples

Since (1'), (2') and (3') from our first example may be transformed into the appropriate form, we may use Persistency Resolution instead of Constrained Resolution to derive (4). In the second example we use (PR) to resolve the first q-literal of (9'') and (10'):

$$\begin{array}{lll} (17) & & \neg([T\text{-}200,T] \subseteq [0,400]) \\ & \vee & \neg((T_2 \in \{T\}) \Rightarrow (\mathrm{pulse}(T_2)=\mathrm{occuring})) \\ & \vee & ((T_3 \in \{T+50\}) \Rightarrow q(T_3)=0) \end{array}$$

Further application of (PR) to (17) and (11') together with syntactic transformation gives

$$(18)\quad [([T\text{-}200,T] \subseteq [0,400]) \wedge (\{T\} \subseteq \{100\}) \wedge (T_3 \in \{T+50\})] \Rightarrow q(T_3)=0,$$

while application to (17) and (12') results in

$$(19)\quad [([T\text{-}200,T] \subseteq [0,400]) \wedge (\{T\} \subseteq \{300\}) \wedge (T_3 \in \{T+50\})] \Rightarrow q(T_3)=0.$$

As in Constraint Logical Programming appropriate constraint handlers may now simplify the qualifications to false and $(T_3=350)$, resp.

7 Generalizing Persistency

Up to now propositions were related to time points which had been collected into intervals. However consider the following sentences:

- "The car is moving all the time."

(Notice that a predicate like moving basically refers to intervals instead of time-points and hence corresponds to a process instead of a persistence fact. The reason is that we cannot decide a proposition like "The car is moving" by observation of the world at a single time point, e. g. represented by a photography. We have to look at a film, that represents the world during a time interval. We may however define predicates like in-a-move refering to time points and derive in-a-move(car , t_0) from moving (car, int) and $t_0 \in$ int.)

"We always go swimming on mondays." (such propositions have been studied by [Ladkin, 1986])

"It rains throughout the country".

"Birds are studied by ornithologists".

By analyzing these examples and comparing them to a persistence fact like TRUE (O, int), we may recognize the following similarities:

Firstly, all of them talk about collections of entities: e. g. intervals (viewed as sets of time-points or collection of their subintervals), sets of days, areas (viewed as sets of their subareas), classes (viewed as sets of their subclasses).

Secondly, all propositions remain true, if this collection is replaced by a subcollection: if we always go swimming on mondays, we always go swimming on every first monday of a month; if it rains throughout the country, it also rains throughout the north or south of the country; if ornithologists study birds, they also study robins (although perhaps not every particular one).

And finally, if the propositions are true for two collections, then they *are* also true for their union.

Notice that we only talked about the concept of membership in the collection, but not about any other properties, like order, density or whatever. And looking at (PR) we see that Persistency Resolution as well only requires the concepts membership and subset. It is the time box that needs additional knowledge about the structure of time to decide on the satisfiability of constraints. Hence we may use (PR) for reasoning about areas in space, about classes or about other kinds of collections, if we can provide for an appropriate constraint reasoner. This leads to the following definitions:

A *persistency structure* S is a triple ((POS U COL), PFUNC , ({C , €} U PREL)), with

POS is a set of objects, called positions;

- COL is a set of objects, called collections, containing elements C0 (the empty collection) and Cpos (the collection of all positions).

PFUNC is a set of n-ary functions on (POS U COL);

- '$\in$' is a relation on POS x COL with $p \in c_{POS}$ for all positions p and $p \in c_{\emptyset}$ for no position p.
- '$\subseteq$' is the subset relation on POS x POS, induced by '$\in$'.
- PREL is a set of n-ary relations on (POS U COL);

Notice that we do not require POS and COL to be disjoint. Hence every interval may be used as a collection of its subintervals.

As with time we include the symbols of S into our logical language, together with Variable symbols C for collections and P for positions. Therefore we may define *Generalized Persistency Resolution* as

$$\text{(GPR)}\quad \frac{\begin{array}{lll} \neg\,(P_1 \in trm[X_1, \ldots, X_n] & \Rightarrow \xi[P_1]) & \vee\ \Phi[X_1, \ldots, X_n] \\ (P_1' \in trm' & \Rightarrow \xi'[P_1']) & \end{array}}{\begin{array}{l} \text{if } [(trm[X_1, \ldots, X_n] \subseteq trm') \wedge trm_1 = trm_1' \wedge \ldots \wedge trm_n = trm_n'] \\ \text{is satisfiable:} \\ \sigma[(trm[X_1, \ldots, X_n] \subseteq trm') \wedge trm_1 = trm_1' \wedge \ldots \wedge trm_n = trm_n'] \\ \qquad \Rightarrow \quad \sigma\Phi[X_1, \ldots, X_n]\ , \end{array}}$$

where σ is a most general unifier of ξ and ξ' which unifies P_1 and P_1', provided that P_1 does not appear in $\Phi[X_1, \ldots, X_n]$, and trm_i , trm_i' are those corresponding terms of ξ and ξ', resp., that are interpreted over S.

To decide on the satisfiability of the resolution residuum, (GPR) needs special inference procedures that can handle constraints on the particular persistency structure.

8 Conclusion

Reasoning with persistence facts like TRUE (O,int) is crucial for temporal reasoning, but difficult for standard first order inference procedures. The problem is that they do not account for the implicit information about subintervals of int. Williams' episode propagation and Penberthy's Temporal unification attempt to implement our intuitive style of reasoning about persistency. They use special inference procedures whose integration with more general techniques appears rather ad hoc. To analyze the prerequisites and potential of these approaches a more formal discussion of how to combine time-boxes with general resolution is necessary.

It turned out that for many examples of informal reasoning about persistency Burckert's Constrained Resolution is an adequate model. However if some formula contains multiple literals TRUE(O , trm), of which at least one is negative and refers to an interval, Constrained Resolution breaks down. Such formulae can be resolved by a new inference rule, called Persistency Resolution. Analysis of the basic elements of this rule finally led to a more general notion of persistency that is not restricted to time-points and intervals.

[Guckenbiehl, 1990] characterizes the functionality of a constraint reasoner which enables the integration of Persistency Resolution into a forward chaining production system. Furthermore, it describes the implementation of this functionality for certain forms of constraints on points and open, closed and halfopen intervals on the reals. This constraint-reasoner has been used in the Extended Episode Propagator (c. f. [Guckenbiehl, 1991]).

I am not aware of any other formal treatment of how to combine temporal and general inference techniques for efficient elaboration of persistence facts, although a lot is currently done on the more ambitious task of using additional assumptions like persistency-by-default or Closed-world. Furthermore ther is some work on exploiting theory unification for temporal reasoning (e. g. [Ohlbach, 1989] that may be important for reasoning with persistence facts.

Persistency Resolution is just a first step towards formal reasoning with persistence facts, and as always much remains to be done. On the formal level the properties of Qualification and Persistency Resolution, particularly completeness, have to be elaborated in more detail. On the conceptual level other interpretations of persistency, e. g. persistency in space, should be explored On the implementational level this requires the design of appropriate constraint-reasoners.

Acknowledgements

I would like to thank my colleagues at IITB and fellow researchers from SFB 314 for valuable comments on an earlier draft of this paper, in particular I have benefited from discussions with Gisela Schafer-Richter, Andreas Nonnengart and Hans-Jurgen Burckert. This work has been supported by the Deutsche Forschungs Gemeinschaft (DFG) in its Special Collaboration Program on AI and Knowledge Based Systems (SFB 314), project X2

References

[Allen, 1983] J. F. Allen: "Maintaining Knowledge about Temporal Intervals", *Comm. ACM,* 26, No. 11 (November 1983), pp. 832 - 843.

[Allen, 1984] J. F. Allen: "Towards a general theory of Action and Time", *Artificial Intelligence,* 23 (1984), pp. 123-154.

[Burckert, 1990] H.-J. Burckert: "A resolution principle for clauses with constraints", *Proc. 10. International Conference on Automated Deduction,* Springer LNAI 449, pp. 187-192, 1990.

[Dean and McDermott, 1987) T. L. Dean and D. McDermott: "Temporal Data Base Management", *Artificial Intelligence,* 32 (1987), pp. 1-55.

[Decker, 1988] R. Decker: "Modelling the temporal behavior of technical systems", Proc. *German Workshop on Artificial Intelligence GWAI,* 1988, pp. 41-50.

[deKleer, 1986] J. deKleer: "An Assumption Based Truth Maintenance System", *Artificial Intelligence,* 28(1986), pp. 127-162.

[Galton, 1990] A. Galton: "A critical examination of Allen's Theory of Action and Time", *Artificial Intelligence,* 42 (1990), pp. 159-188.

[Guckenbiehl, 1990] T. Guckenbiehl: *Reasoning with persistent facts,* internal report, FhG-IITB, 1990.

[Guckenbiehl, 1991] T. Guckenbiehl: *The Extended Episode Propagator,* Internal Report, FhG-IITB, in preparation.

[Hrycej, 1988] T. Hrycej: "Temporal Prolog", *Proc. European Conference on Artificial Intelligence,* Munich, 1988, pp. 296-301.

[Kahn and Gorry, 1977] K. Kahn and A. Gorry: "Mechanizing Temporal Knowledge", *Artificial Intelligence* 9 (1977), pp. 87-108.

[Ladkin, 1986] P. Ladkin: "Time Representation: A Taxomony of interval relations", Proc. *AAAI National Conference on Artificial Intelligence,* 1986, pp. 360-366.

[McDermott, 1982] D McDermott: "A temporal logic for Reasoning about Processes and Plans", *Cognitive Science,* 6(1982), pp. 101 -155.

[Ohlbach, 1989] H.-J. Ohlbach: *Context Logic,* Universitat Kaiserslautern, SEKI-report SR-89-08, 1989.

[Penberthy, 1987] J. S. Penberthy: "Temporal Unification and the Temporal Partial Order", Proc. *5^{th} IEEE-Conf. on AI applications,* San Diego, 1987, pp. 223-228.

[Rescher and Urquhart, 1971] N. Rescher and A Urquhart: *Temporal Logic,* Berlin, 1971.

[Shoham, 1987] Y. Shoham: *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence,* Ph. D. Thesis, Yale University, CompSc. Dept.,May 1987.

[van Beek, 1989] P. van Beek: "Approximation Algorithms for Temporal Reasoning", Proc. *International Joint Conference on Artificial Intelligence,* 1989, pp. 1291-1296.

[Williams, 1986] B. Williams: "Doing Time: Putting Qualitative Reasoning on firmer ground", *Proc. AAAI National Conference on Artificial Intelligence,* 1986, pp. 105-112.