Satisfying First-Order Constraints About Time Intervals

Peter B. Ladkin
Kestrel Institutc
1801 Page Mill Road
Palo Alto, Ca 94304-1216

Abstract

James Allen defined a calculus of time intervals in *[All83]*, as a representation of temporal knowledge that could be used in AI. We shall call this the *Interval Calculus*. In his paper, Allen investigated specification and constraint satisfaction in the Interval Calculus. Other constraint-satisfaction algorithms for intervals have considered subclasses of Boolean formulas only. The methods herein extend consistency-checking and constraint-satisfaction procedures to finitely many arbitrary quantified formulas in the Interval Calculus. We use a first-order theory from *[LadMad87.1, LadMad88.1]*, that precisely corresponds to Allen's calculus. We show that every first-order constraint expressible in this theory is equivalent to a Boolean constraint of a particular restricted form. We use this result to obtain a procedure for detecting consistency of arbitrary quantified formulas, and finding intervals that satisfy arbitrary consistent formulas of the Interval Calculus.

1 Introduction

We are concerned here with constraint satisfaction in the Interval Calculus defined in *[All83]*. Allen's calculus encompassed an approach to temporal specification and theorem proving new to AI, though his thirteen basic relations had been used elsewhere (e.g. *[vBen83]*). Allen contributed a constraint satisfaction algorithm, which used the composition table for the relations to infer path-inconsistencies in interval constraint graphs. Interval representations were subsequently used for representing time for automated planning e.g. *[All84, AllKau85, PelAll87]*.

In *[LadMad87.1, LadMad88.1]* we obtained results which show that the calculus is the complete theory of intervals over the rational numbers, and has only this one countable model, up to isomorphism. Thus we may use semantic techniques to satisfy constraints in the calculus.

In this paper, we present procedures for quantifier-elimination, consistency checking (and therefore deciding), and providing satisfying assignments for consistent formulas in the full first-order theory of the Interval Calculus. Thus arbitrary quantified formulas in the Interval Calculus may be handled with the methods described here. Previous constraint-satisfaction procedures have only considered subclasses of the formulas without quantifiers *[All83, MacFre85, Val87, Bel87]*.

Briefly, by the results of *[LadMad88.1]* we can translate a sentence in the interval theory into a sentence in the theory of unbounded dense linear order that expresses the 'same' constraint. We may now eliminate quantifiers in the formula in the theory of unbounded dense linear order, and translate the formula back into an interval formula. The translation from atomic formulae in the theory of unbounded dense linear order does not introduce any quantifiers, so the resulting interval formula is quantifier-free, and equivalent to the original formula. The resulting quantifier-free formula has a certain restricted form, and may be checked for consistency by a variety of known techniques that operate directly on restricted quantifier-free interval formulae. However, an intuitively better method for checking consistency stops with the quantifier-free formula in the theory of dense unbounded linear order and checks this formula directly for consistency, by a simple test, without translating back to intervals.

There is practical interest not just in checking constraints for consistency alone, but in trying to satisfy the constraints, i.e. find an interpretation of the free variables in the constraining formula that will render this formula true in the intended model. We show that we may combine quantifier-elimination with a simple assignment procedure to obtain an assignment to free variables of a consistent interval formula that satisfies the formula. We call such a procedure a *satisfaction procedure*. The satisfaction procedure utilises the translation into the rational ordering and the quantifier-elimination there. If the formula is consistent, a collection of rationals is found which satisfies the quantifier-free formula. These rationals are then used as endpoints for intervals which satisfy the original interval formula.

It is a matter for further research to decide between the various available techniques for satisfying Boolean formulas in the interval theory. Our contribution to this research is to present not just a consistency algorithm but a satisfaction algorithm for a much larger class of constraints than has been considered before in the context of interval theories. Proofs of our results may be found in *[Lad87.5]*.

1.1 Definitions

A *structure* is a set of objects, along with with relations on those objects and total functions on the set, with all

arguments and values in the set, and distinguished objects called *constants*. We denote structures in the usual way, using angle brackets. The rational numbers Q with the relation of *less-than* on the rationals is denoted by $\langle Q, < \rangle$. We call this structure RAT.

$INT(Q)$ is the structure in a language with equality with thirteen binary relation primitives, introduced in *[LadMad88.1]*, and below. (We shall sometimes refer to $INT(Q)$ as INT.) The domain of $INT(Q)$ is

$$L = \{\langle x, y\rangle : x < y, x, y \in Q\}$$

and the primitive relations are the relations defined by Allen *[All83]*. The binary relations are thus sets of pairs of pairs of rationals. The following definitions of the relations are from *[LadMad88.1]*, (Allen defined them over R, not Q, but it doesn't matter *[LadMad87.1, LadMad88.1]*).

$$Id(L) = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x = x' < y = y' \in Q\}$$

$Id(L)$ is the identity relation on the domain L. The following six relations are primitives:

$$P = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < y < x' < y' \in Q\}$$

$$D = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x' < x < y < y' \in Q\}$$

$$O = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < x' < y < y' \in Q\}$$

$$M = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < y = x' < y' \in Q\}$$

$$S = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x = x' < y < y' \in Q\}$$

$$F = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x' < x < y = y' \in Q\}$$

The *converse* of a binary relation R is $R^\smile = \{(y, x) : (x, y) \in R\}$. The remaining six relations are the *converse* relations of the above six, so $INT(Q)$ has domain L and relations $Id(L)$, P, D, O, M, S, F, $P^\smile$, $D^\smile$, $O^\smile$, $M^\smile$, $S^\smile$, $F^\smile$.

We use the notation

$$M \models \phi\{x_1 \leftarrow a_1, ..., x_n \leftarrow a_n\}$$

where the free variables of ϕ are included in the list $x_1, .., x_n$, and $a_1, ..., a_n$ are elements of the domain of M, to mean that the formula ϕ is true in structure M under all assignments that assign a_1 to x_1, ... , a_n to x_n. We shall implicitly assume that the variables $x_1, ..., x_n$ are all distinct.

The *theory of the model M*, denoted by $Th(M)$, is the set of all sentences in the language of M that are true in M. *Th(M)* is complete (by definition!), and of course M is a model for *Th(M)*.

Given an interval $i = (x, y)$ in the domain of *INT(Q)*, we define $i_L = x$ and $i_R = y$ to be the projections of i onto its left and right endpoints in Q.

The Axiomatisation of $Th(INT(Q))$

In *[LadMad87.1, LadMad88.1]* we gave a collection of axioms for Allen's calculus, in a first-order language with equality and 13 binary relation symbols. We showed that the theory T defined by the axioms is countably categorical, hence complete and decidable, and 'the' countable model is *INT(Q)* (i.e. all models are isomorphic to this model). From this is follows that $T = Th(INT(Q))$, so we may use semantic techniques from the theory of dense unbounded linear order to derive constraint satisfaction algorithms for the interval calculus.

2 The Translations

We introduce a translation $(-^*)$ from formulae of the language of INT to formulae of the language of RAT, and another translation $(-^\dagger)$ from Boolean formulae of the language of RAT to Boolean formulae of the language of INT that preserves satisfiability, and furthermore such that objects satisfying the image of a formula ϕ under one of these mappings are easily computable from objects satisfying ϕ.

2.1 From INT to RAT

We define a translation $(-^*)$ from a formula ϕ in the language of INT to a formula ϕ^* in the language of Q such that

$$INT \models \phi\{z_1 \leftarrow i_1, ..., z_n \leftarrow i_n\}$$

iff

$$RAT \models \phi^*\{x_1 \leftarrow (i_1)_L, ..., x_n \leftarrow (i_n)_L, y_1 \leftarrow (i_1)_R, ..., y_n \leftarrow (i_n)_R\}$$

where the x_i and y_i are new variables not occurring in ϕ.

For convenience, we use three infinite sequences of distinct variables $e_1, .., e_n, .., f_1, .., f_n, .., g_1, .., g_n, ..$, We shall consider formulas in the language of INT to contain variables only from the list of e_n, and formulas in the language of RAT only to contain variables from the two other lists. The intuitive reason for the three different lists of variables is that we shall be translating assertions about intervals into assertions about their left and right endpoints, and vice versa, and so we shall associate each interval variable e_n with corresponding variables f_n for its left endpoint, and g_n for its right endpoint. This is a useful piece of bookkeeping. We use the metavariables z, w to range over the list of e_n's, metavariables x, x' to range over the list of f_n's, and metavariables y, y' to range over the list of g_n's.

Roughly speaking, we are translating an assertion about $i_1, ..., i_n$ in INT into an assertion about the endpoints $(i_1)_L, ..., (i_n)_L, (i_1)_R, ...(i_n)_R$ in RAT. The translation is given by looking at the definition of the relations above. The defining formula in each of the relation definitions is a formula $\phi_R(x, y, x', y')$ in the language of RAT, for each of the thirteen primitive relations R of **IA**.

Hence two intervals i, j in $INT(Q)$ are in the relation R iff the predicate $\phi_R(x, y, x', y')$ is true for the endpoints of i and j. We use this fact to define the translation $(-^*)$ for the atomic formulas.

- If ϕ is an atomic formula $R(z, w)$, then ϕ^* is $\phi_R(x, y, x', y')$, (where if $z = e_n$, then $x = f_n$ and $y = g_n$, and if $w = e_m$, then $x' = f_m$ and $y' = g_m$)
- If $\phi = (\neg\psi)$ then $\phi^* = (\neg\psi^*)$.
- If $\phi = (\psi \wedge \rho)$ then $\phi^* = (\psi^* \wedge \rho^*)$.
- If $\phi = (\psi \vee \rho)$ then $\phi^* = (\psi^* \vee \rho^*)$.
- If $\phi = (\psi \rightarrow \rho)$ then $\phi^* = (\psi^* \rightarrow \rho^*)$.
- If $\phi = (\forall z)\psi$ and $z = e_n$ then $\phi^* = (\forall x \forall y)(x < y \rightarrow \psi^*)$ where $x = f_n$ and $y = g_n$
- If $\phi = \exists z\psi$ and $z = e_n$ then $\phi^* = (\exists x \exists y)(x < y \wedge \psi^*)$ where $x = f_n$ and $y = g_n$

Lemma 1 *Let ϕ be a formula in the language of INT. Then*

$$INT \models \phi\{e_1 \leftarrow i_1, ..., e_n \leftarrow i_n\}$$

$$\textit{iff}$$

$$RAT \models \phi^*\{f_1 \leftarrow (i_1)_L, ..., f_n \leftarrow (i_n)_L, g_1 \leftarrow (i_1)_R, ..., g_n \leftarrow (i_n)_R\}$$

2.2 From RAT to INT

First, we state a normal form theorem from *[ChaKei73]* for formulas in the language of RAT. Define an *order relation* to be a formula of the form

$$u_{i_1} < u_{i_2} < ..... < u_{i_k}$$

where the iterated $<$ is shorthand for the conjunction of atomic formulas involving adjacent variables, and each u_{i_j} is either some f_m or some g_n. Say a formula is in *rational order normal form* (*RONF* for short) if it is a disjunction of order relations.

Theorem 1 (standard:) *Every formula ϕ in the language of RAT is equivalent in $Th(RAT)$ to a formula $\phi^\natural$ in RONF; furthermore there is an algorithm for obtaining such a RONF formula $\phi^\natural$ from an arbitrary formula ϕ in the language of RAT, and the free variables of $\phi^\natural$ are a subset of those of ϕ, and possibly a proper subset.*

We now consider the translation of atomic formulas in the language of RAT into Boolean formulas of the language of INT. We use the notation $i(R_1 + R_2 + ... + R_p)j$ to assert that the interval i is in one of the relations R_q to j.

The following statements about RAT and INT are easy to check:

- $i_L < j_L \Leftrightarrow i\,(P + M + O + F^\smile + D^\smile)\,j$
- $i_R < j_R \Leftrightarrow i\,(P + O + S + D)\,j$
- $i_L < j_R \Leftrightarrow i\,(P + M + O + Id(L) + S + F + D)\,j$
- $i_R < j_L \Leftrightarrow i\,P\,j$
- $i_L = j_L \Leftrightarrow i\,(S + Id(L) + S^\smile)\,j$
- $i_R = j_R \Leftrightarrow i\,(F + Id(L) + F^\smile)\,j$
- $i_L = j_R \Leftrightarrow i\,M^\smile\,j$
- $i_R = j_L \Leftrightarrow i\,M\,j$

We use these truths to define a translation $(-^\dagger)$ from atomic formulas ϕ in the language of RAT into formulas $\phi^\dagger$ in the language of INT. We use the small roman letters, some with superscripts,

$$p, d, o, m, s, f, p^\smile, d^\smile, o^\smile, m^\smile, s^\smile, f^\smile$$

along with $=$, to denote the thirteen primitive predicate symbols in the language of INT.

- If $\phi = (f_m < f_n)$ then $\phi^\dagger = (p(e_m, e_n) \vee m(e_m, e_n) \vee o(e_m, e_n) \vee f^\smile(e_m, e_n) \vee d^\smile(e_m, e_n))$
- If $\phi = (g_m < g_n)$ then $\phi^\dagger = (p(e_m, e_n) \vee o(e_m, e_n) \vee s(e_m, e_n) \vee d(e_m, e_n))$
- If $\phi = (f_m < g_n)$ then $\phi^\dagger = (p(e_m, e_n) \vee m(e_m, e_n) \vee o(e_m, e_n) \vee s(e_m, e_n) \vee e_m = e_n \vee f(e_m, e_n) \vee d(e_m, e_n))$
- If $\phi = (g_m < f_n)$ then $\phi^\dagger = p(e_m, e_n)$
- If $\phi = (f_m = f_n)$ then $\phi^\dagger = (s(e_m, e_n) \vee e_m = e_n \vee s^\smile(e_m, e_n))$
- If $\phi = (g_m = g_n)$ then $\phi^\dagger = (f(e_m, e_n) \vee e_m = e_n \vee f^\smile(e_m, e_n))$
- If $\phi = (f_m = g_n)$ then $\phi^\dagger = m^\smile(e_m, e_n)$
- If $\phi = (g_m = f_n)$ then $\phi^\dagger = m(e_m, e_n)$

We give an example of how this translation works.

$$RAT \models (g_m < g_n)\{g_m \leftarrow i_R, g_n \leftarrow j_R\}$$

$$\Leftrightarrow$$

$$i_R < j_R$$

$$\Leftrightarrow$$

$$i\,(P + O + S + D)\,j$$

$$\Leftrightarrow$$

$$INT \models (p(e_m, e_n) \vee o(e_m, e_n) \vee s(e_m, e_n) \vee d(e_m, e_n))\{e_m \leftarrow i, e_n \leftarrow j\}$$

$$\Leftrightarrow$$

$$INT \models \phi^\dagger\{e_m \leftarrow i, e_n \leftarrow j\}$$

We extend the translation to all of the Boolean formulae of the language of RAT:

- If $\phi = (\neg\psi)$ then $\phi^\dagger = (\neg\psi^\dagger)$.
- If $\phi = (\psi \wedge \rho)$ then $\phi^\dagger = (\psi^\dagger \wedge \rho^\dagger)$.
- If $\phi = (\psi \vee \rho)$ then $\phi^\dagger = (\psi^\dagger \vee \rho^\dagger)$.
- If $\phi = (\psi \rightarrow \rho)$ then $\phi^\dagger = (\psi^\dagger \rightarrow \rho^\dagger)$.

If the assignments of i_L, i_R, j_L, and j_R to f_m, f_n, g_m, and g_n, and i, j to e_m, e_n are made in accordance with our convention, we call such a pair of assignments *mutually acceptable*. If α is a mutually acceptable pair of assigments, let α_{RAT} be the assignment in the language of RAT, and α_{INT} be the corresponding assignment in the language of INT. We have the following lemma:

Lemma 2 *For every Boolean formula ϕ of the language of RAT, for every mutually acceptable pair of assignments α,*

$$RAT \models \phi\{\alpha_{RAT}\} \Leftrightarrow INT \models \phi^\dagger\{\alpha_{INT}\}$$

We shall not need to extend the translation to quantified formulae of RAT.

We define a *uniform disjunction* of atomic formulae in the language of INT to be a disjunction of atomic formulae of the language of INT involving the same two variables e_m, e_n occurring in the same order in each subformula. Examples of uniform disjunctions are the formulae $\phi^\dagger$ corresponding to the atomic formulae ϕ in the language of RAT. In the case that ϕ is an order relation, $\phi^\dagger$ will be a conjunction of uniform disjunctions, and thus if ϕ is a disjunction of order relations, $\phi^\dagger$ will be a disjunction of conjunctions of uniform disjunctions.

3 The Algorithms

3.1 Quantifier-Elimination in $Th(INT)$

We have defined two translations, $(-^*)$ from formulas in the language of INT to formulas in the language of RAT, and $(-^\dagger)$ from Boolean formulas in the language of RAT to Boolean formulas in the language of INT, which preserve satisfiability - in fact, which preserve satisfaction by mutually acceptable assignments. From these two translations, along with the quantifier-elimination procedure $\psi \longmapsto (\psi^\sharp)$ from $Th(RAT)$, we may define a quantifier-elimination procedure for $Th(INT)$.

The variables in our translations from INT to RAT and vice versa, and the assignments in the satisfaction relation, have been fairly carefully controlled to ensure the preservation of satisfaction as we move back and forth from INT to RAT. We need to ensure that the translation $-^\sharp$ in $Th(RAT)$ is equally careful. We may choose $-^\sharp$ so that the free variables of $\phi^\sharp$ are a subset of the free variables of its input formula ϕ. This ensures that the satisfaction relation is preserved.

The Quantifier-Elimination Algorithm:
Given a formula ϕ in the language of INT, compute $((\phi^*)^\sharp)^\dagger$.

End of Algorithm.

Lemma 3 *For any formula ϕ in the language of INT, $((\phi^*)^\sharp)^\dagger$ is a disjunction of conjunctions of uniform disjunctions, and $INT \models \phi \leftrightarrow ((\phi^*)^\sharp)^\dagger$ and thus $Th(INT) \vdash \phi \leftrightarrow ((\phi^*)^\sharp)^\dagger$*

The lemma guarantees the correctness of the algorithm, which obviously always terminates.

3.2 Consistency Algorithms

Allen's algorithm *[All83]* checked conjunctions of uniform disjunctions for consistency. We call a conjunction of uniform disjunctions an *Allen formula*. Let us call an algorithm for checking consistency of Allen formulae a *complete Allen algorithm*. Our first consistency algorithm will use a complete Allen algorithm for checking arbitrary first-order constraints in $Th(INT)$ by combining it with the quantifier-elimination procedure.

The quantifier-elimination method takes as input a formula ϕ of the language of INT, and produces an equivalent quantifier-free formula $\psi_\phi = ((\phi^*)^\sharp)^\dagger$, which is a disjunction of Allen formulae. Given that Allen formulae may be checked for consistency, a disjunction of Allen formulae may be checked for consistency by checking each of them in parallel.

Consistency Algorithm I:
Given a formula ϕ of the language of INT
1) Find $\psi_\phi (= ((\phi^*)^\sharp)^\dagger)$.
2) ψ_ϕ is a disjunction of Allen formulae. Apply a complete Allen algorithm to each disjunct of ψ_ϕ.
End of Algorithm.

The algorithm terminates and is correct modulo an appropriate choice of complete Allen algorithm. This algorithm is not necessarily the easiest way to check consistency. Given a formula ϕ in $Th(INT)$, the formula $(\phi^*)^\sharp)$ is a disjunction of order relations. Since the translation $\phi \longmapsto (\phi^*)^\sharp$ preserves satisfaction, we may check consistency of ϕ by checking the consistency of $(\phi^*)^\sharp)$ in $Th(RAT)$. $(\phi^*)^\sharp$ is a disjunction of order relations. We shall say an order relation O is *in* $(\phi^*)^\sharp$ if and only if O is one of the disjuncts of $(\phi^*)^\sharp$. $(\phi^*)^\sharp$ is consistent if and

only if some order relation in $(\phi^*)^\sharp$ is satisfiable. We may thus use the following lemma *[ChaKei73]* for an improved algorithm:

Lemma 4 *An order relation $u_{i_1} < u_{i_2} < \ldots\ldots < u_{i_k}$ is satisfiable if and only if it contains only a single occurrence of each free variable u_{i_j} $(1 \leq j \leq k)$.*

Consistency Algorithm II:
Given a formula ϕ of $Th(INT)$,
1) Compute $(\phi^*)^\sharp$);
2) Check each order relation in $(\phi^*)^\sharp$) for consistency (this may be accomplished in parallel);
3) Return **consistent** if one order relation is consistent, **inconsistent** if they are all inconsistent.
End of Algorithm.

Algorithm II omits the final translation $(-^\dagger)$ back into the language of INT, and substitutes a double-occurrence check on each order relation in the reduced formula in $Th(RAT)$, so it should be clear that *Algorithm II* is at least as efficient as *Algorithm I*, and it doesn't use a complete Allen algorithm, but a simple check instead.

3.3 A Satisfaction Procedure

The techniques used in *Consistency Algorithm I* and *Consistency Algorithm II* may be used to yield more information about the formula ϕ. A consistent order relation $u_{i_1} < u_{i_2} < \ldots\ldots < u_{i_k}$ may be satisfied by any increasing sequence of rational numbers, say the integers $1, \ldots\ldots, k$. Let an *integer assignment* be an assignment of increasing consecutive integers, starting with 1, to each variable in some order relation in $(\phi^*)^\sharp$. These integers will then represent assignments of integers to variables representing the endpoints of interval variables in ϕ. Thus, intervals with these endpoint assignments can be assigned to the interval variables of $((\phi^*)^\sharp)^\dagger)$, and by the mutual satisfiability property of the translation $(-^\dagger)$, the conjunction of uniform disjunctions that corresponds to the satisfied order relation will be satisfied by these intervals, so $((\phi^*)^\sharp)^\dagger)$ will be satisfied, so ϕ will be satisfied by this assignment also. We can summarise this in the following satisfaction algorithm:

The Satisfaction Algorithm
Given ϕ in the language of INT,
1) Compute $(\phi^*)^\sharp)$.
2) Make an integer assignment to the first (either in terms of order, or in terms of time) consistent order relation in $(\phi^*)^\sharp)$. If there is no such relation, return **inconsistent**. Suppose this order relation is O.
3) Suppose f_n has been assigned integer p, and g_n does not appear in O. Assign $p+1$ to g_n. Similarly, suppose g_n has been assigned integer $p+1$, and f_n does not appear in O. Assign p to f_n. Do this for all such f_n and g_n.
4) For each interval variable e_n such that f_n or g_n appear in O, suppose f_n and g_n have been assigned integers p and q respectively. Assign the interval (p, q) to e_n.
5) Assign an arbitrary interval, say $(0, 1)$, to each free variable e_m of ϕ such that neither f_m nor g_m appear in O.
End of Algorithm.

The correctness of the algorithm is straightforward. We have noted already the purpose of steps **1** and **2**. Suppose ϕ is consistent with $Th(INT)$. Steps **3** and **4** construct a pair of mutually acceptable assignments. The purpose of step **3** is to make an assignment to the other endpoint of intervals which have had only one endpoint assigned. The choice of this other endpoint is arbitrary, consistent with the requirement that $i_L < i_R$ for each interval i. Step **4** makes the assignment of the appropriate pair to an interval variable. Finally, step **5** completes the assignment to all the free variables of ϕ. Since O and thus $((\phi^*)^\sharp)^\dagger)$ is satisfied by a subassignment of the appropriate assignment of this mutually acceptable pair, it follows that ϕ is satisfied by the assignment constructed by the satisfaction procedure.

4 Summary

We have introduced two translations $-^*$ from the language of the interval theory to the language of the theory of unbounded dense linear order, and $-^\dagger$ in the other direction, which preserve satisfiability of formulas. By composing these with the quantifier-elimination method for the theory of unbounded dense linear order, we obtained a quantifier-elimination method for $Th(INT)$. We gave consistency algorithms for formulas in $Th(INT)$. Finally, we showed how a procedure for constructing a satisfying assignment for a consistent interval formula ϕ could be designed by combining the quantifier-elimination algorithm with a simple assignment procedure for consistent order relations in $Th(RAT)$, and finding a mutually acceptable pair of assignments from this.

We have noted that the contribution of this work is to extend satisfaction procedures to arbitrary quantified formulas, and hence finite collections of such, in the Interval Calculus.

Bibliography

All83 : Allen, J.F., *Maintaining Knowledge about Temporal Intervals*, Comm. A.C.M. 26 (11), November 1983, 832-843.

All84 : Allen, J.F., *Towards a General Theory of Action and Time*, Artificial Intelligence 23 (2), July 1984, 123-154.

AllKau85 : Allen, J.F. and Kautz, H., *A Model of Naive Temporal Reasoning*, in Hobbs, J.R. and Moore, R.C., editors, *Formal Theories of the Commonsense World*, Ablex 1985.

Bel87 : Bell, C.E., *Representing And Reasoning With Disjunctive Temporal Constraints In A Point-Based Model*, preprint, University of Iowa, Department of Management Sciences.

ChaKei73 : Chang, C.C., and Keisler, H.J., *Model Theory*, North-Holland, 1973.

Lad87.5 : Ladkin, P.B., *Constraint Satisfaction in Time Interval Structures I: Convex Intervals*, Kestrel Institute Technical Report KES.U.87.11, 1987.

LadMad87.1 : Ladkin, P.B. and Maddux, R.D., *The Algebra of Convex Time Intervals*, Kestrel Institute Technical Report KES.U.87.2.

LadMad88.1 : Ladkin, P.B. and Maddux, R.D., *Representation and Reasoning with Convex Time Intervals*, Kestrel Institute Technical Report KES.U.88.2.

MacFre85 : Mackworth, A.K., and Freuder, E.C., *The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems*, Artificial Intelligence 25, 65-74, 1985.

PelAll87 : Pelavin, R., and Allen, J.F., *A Model For Concurrent Actions Having Temporal Extent*, Proceedings of AAAI-87, the Sixth National Conference on Artificial Intelligence, Morgan Kaufmann 1987, 246-250.

Val87 : Valdés-Pérez, R.E., *The Satisfiability of Temporal Constraint Networks*, Proceedings of AAAI-87, the Sixth National Conference on Artificial Intelligence, pp256-260, Morgan Kaufmann 1987.

vBen83 : van Benthem, J.F.A.K., *The Logic of Time*, Reidel 1983.

VilKau86 : Vilain, M., and Kautz, H., *Constraint Propagation Algorithms for Temporal Reasoning*, Proceedings of AAAI-86, 377-382, Morgan Kaufmann, 1986.