Journal of Universal Computer Science, vol. 16, no. 11 (2010), 1425-1438
submitted: 5/1/10, accepted: 28/5/10, appeared: 1/6/10 © J.UCS

Finding a Consistent Scenario to an Interval Algebra Network Containing Possibly Infinite Intervals

André Trudel
(Acadia University, Wolfville, Canada
andre.trudel@acadiau.ca)

Abstract: Interval algebra networks are traditionally defined over finite intervals. In this paper, we relax this restriction by allowing one or more of the intervals involved to be infinite. Intervals in the network can be finite, left-infinite, right-infinite, or infinite in both directions. The network's intervals can all be of the same type, or different. We present algorithms for finding a consistent scenario.

Keywords: Allen's Interval Algebra, Infinite Temporal Intervals, Temporal Knowledge Representation and Reasoning
Categories: I.2.4

1 Introduction

Allen [84] defines a temporal reasoning approach based on intervals and the 13 possible binary relations between them. The relations are before (b), meets (m), overlaps (o), during (d), starts (s), finishes (f), and equals (=) (see Table 1). Each relation has an inverse. The inverse symbol for b is bi and similarly for the others: mi, oi, di, si, and fi. The inverse of equals is equals.

A relation between two intervals is restricted to a disjunction of the basic relations, which is represented as a set. For example, (A m B) V (A o B) is written as A {m,o} B. The relation between two intervals is allowed to be any subset of I = {b,bi,m,mi,o,oi,d,di,s,si,f,fi,=} including I itself.

An IA (Interval Algebra) network is a graph where each node represents an interval. Directed edges in the network are labelled with subsets of I. By convention, edges labelled with I are not shown. An IA network is consistent (or satisfiable) if each interval in the network can be mapped to a real interval such that all the constraints on the edges hold (i.e., one disjunct on each edge is true).

A scenario of an IA network is a singleton labelling of the network (i.e., each edge only has one of its original labels). A consistent scenario is a scenario where each constraint on each edge is true. An IA network is consistent if and only if it has a consistent scenario.

Intervals in Allen's interval algebra are finite and convex. In this paper, the finiteness condition is relaxed. In addition to finite intervals, intervals that are half infinite towards negative infinity, half infinite towards positive infinity, and infinite in both directions are allowed. Note that all intervals are convex.

In this paper, we present and solve IA^{inf} networks. An IA^{inf} network is similar in structure to an IA network. In an IA^{inf} network, intervals are allowed to be non-finite. The intervals can all be of the same type, or mixed. For example, an IA^{inf} network can

contain finite, left-infinite, right-infinite, and infinite intervals. Note that an IA network is an IA^{inf} network.

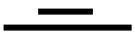
Relation	Symbol	Example
X before Y	b	
X meets Y	m	
X overlaps Y	o	
X starts Y	s	
X during Y	d	
X finishes Y	f	
X equals Y	=	

Table 1: Possible relationships between two finite intervals

In the next section, we catalogue the different possible relationships among finite and non-finite intervals. We then present two algorithms for finding a consistent scenario of an IA^{inf} network.

2 Possible Relationships Involving Non-finite Intervals

The following graphical notation is used for convex intervals:

- Finite length interval (i.e., *finite*):
- Fixed endpoint on the left and infinite on the right (i.e., *right-infinite*):
- Fixed endpoint on the right and infinite on the left (i.e., *left-infinite*):
- Infinite in both directions (i.e., *infinite*):

All the possible relationships involving non-finite intervals are shown in Table 2 and Table 3. Table 2 summarizes the non-finite possibilities by relation. Note that each entry in Table 2 has an inverse. For example, in the first row finite interval X is before (b) right-infinite interval Y. It is also the case that Y is after (bi) X. Table 3 summarizes the possible relations by interval type. The entry in row r and column c is all the possible relations between an interval of type r and c. For example, if X is left-infinite and Y is right-infinite, the only possibilities are X {o, m, b} Y. It is interesting to note that each entry in Table 3 also appears as entries in Allen's [83] full composition table.

An interesting property of Allen's relations is that if a relation r holds between two possibly non-finite intervals, then it also holds between finite versions of the intervals (the intervals are made finite by chopping the infinite ends at large positive and/or negative values). For example, let X and Y be intervals of type finite, right-infinite, left-infinite, or infinite. Note that X and Y may or may not be of the same type, and if

they are of the same type they may not be equal. Assume there is a valid relation r which holds between X and Y (i.e., X {r} Y). Now map both X and Y to real intervals. If X and/or Y extend towards negative infinity, chop off these intervals at a value called LE such that if X and Y contain any finite endpoints, they are all larger than LE. Do the same on the right hand side and chop all intervals going towards positive infinity at a large positive point RE. Both X and Y are now finite. One can verify manually that it is the case that regardless of the original intervals and relation chosen, it is still the case that X {r} Y. An example of the construction is shown in Figure 1. X is right-infinite, Y is infinite, and X {f} Y. Both intervals are made finite by chopping the infinite ends. X's left endpoint is not modified. It remains the case that X {f} Y for the finite versions of the intervals.

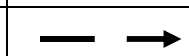
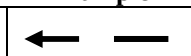
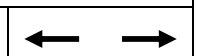
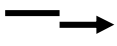
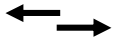
Symbol	**Example**		
b			
m			
o			
s			
d			
f			
=			

Table 2: Summary by relation

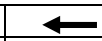
				
	b, bi, m, mi, o, oi, d, di, s, si, f, fi, =	b, m, o, s, d	bi, mi, oi, f, d	d
	bi, mi, oi, si, di	fi, =, f	oi, mi, bi	f
	b, m, o, fi, di	o, m, b	si, =, s	s
	di	fi	si	=

Table 3: Summary by interval type

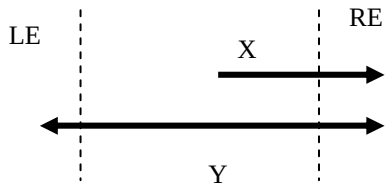


Figure 1: X {f} Y

3 Properly Labelling Missing Edges

An IA^{inf} network is shown in Figure 2 where interval A is finite, B is left-infinite, C is right-infinite, and D is infinite. If the network in Figure 2 is used as input to a standard IA network finite interval algorithm (e.g., [van Beek, 96]), we could generate the solution shown in Figure 3. This solution is correct if all the intervals are finite. It is incorrect in our case. It is impossible for the interval B to be before (b) the infinite interval D.

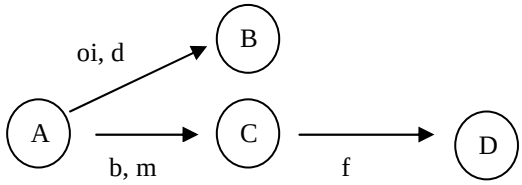


Figure 2: IA^{inf} network

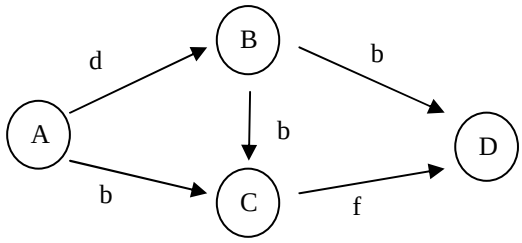


Figure 3: Consistent finite interval scenario

The problem is not with the algorithm. It should be used with a proper model of time. Figure 2 has three hidden edges (i.e., AD, BC, and BD) which are each assumed by the algorithm to have a label of I. This is causing the problem. The label I on each

of these edges is incorrect because not all the labels within I are possible. For example, B cannot be before (b) the infinite interval D. To rectify this situation, instead of labeling missing edges with I as is the standard practice, we must label missing edges with the appropriate entry from Table 3. The correctly labeled network from Figure 2 is shown in Figure 4. Note that the label "oi,mi,bi" in the center of the figure belongs to the edge CB. This label is the entry in the right-infinite row and left-infinite column in Table 3.

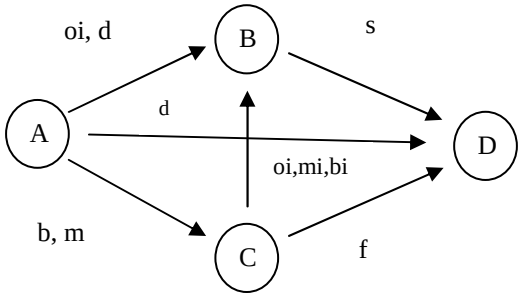


Figure 4: IA^{inf} network with missing edges properly labeled

If we now input the network in Figure 4 to IA network software (e.g., [van Beek, 96]), we could generate the correct scenario shown in Figure 5. The scenario in Figure 5 represents the relative arrangement of intervals shown in Figure 6.

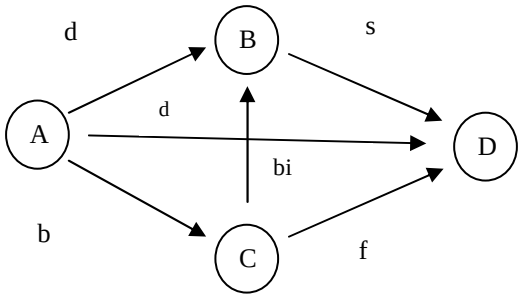


Figure 5: Correct non-finite interval scenario

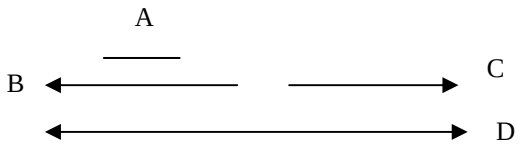


Figure 6: Solution involving non-finite intervals

4 Algorithm 1: IA networks

A consistent scenario to an IA^{inf} network is found as follows. First add missing edges labeled with the appropriate entry from Table 3. Then, apply finite interval IA network solution software to the network. The software will either report that the original network is inconsistent, or return a consistent scenario.

The algorithm is simple, and correct. Let NF-IAN be an IA^{inf} network. If NF-IAN has any missing edges, add them to NF-IAN along with the appropriate label from Table 3. Let another IA network F-IAN have the same nodes, edges, and labels as NF-IAN. The only difference between the two is that all the intervals in F-IAN are finite. The following theorem proves the correctness of the proposed algorithm.

Theorem: S is a consistent scenario of NF-IAN if and only if it is a consistent scenario of F-IAN.

Proof: Assume S is a consistent scenario of NF-IAN. We can map all the intervals in S to intervals over the real numbers such that each relation in S holds. As we did in Section 2, we chop off all the left-infinite and infinite intervals on the left hand side at some arbitrary number LE which is smaller in value than any of the finite endpoints which may occur in any of the other intervals. We do the same with right-infinite and infinite intervals on the right hand side. They all get chopped at the same point RE. As observed in Section 2, this chopping does not affect the individual relations between pairs of intervals. All the edge labels in S will still hold and all the intervals are finite. We therefore have a consistent scenario for F-IAN. For example, assume NF-IAN is the network in Figure 4, F-IAN is the same network where all the intervals are finite, and S is the scenario represented by Figure 6. The non-finite intervals of S are chopped in Figure 7, and the resulting finite F-IAN scenario is shown in Figure 8.

Now consider the case where S is a consistent scenario of F-IAN. Map all the intervals in S to finite intervals over the real numbers such that each relation in S holds. Let LI be the set of left-infinite and infinite intervals in NF-IAN. If LI is non-empty, the following will be the case:

1. The finite intervals in S corresponding to the intervals in LI will all have the same left endpoint. No other intervals will have this left endpoint. This follows from the possible relations in Table 3 involving left-infinite and infinite intervals.
2. It also follows from Table 3 that no interval endpoint will be to the left of the left endpoints of the intervals in S which correspond to the intervals in LI.

Based on the above properties, we can shift the left endpoints of the intervals in S which correspond to the intervals in LI any arbitrary distance towards negative infinity without violating any relations in S. Push these endpoints all the way to negative infinity. Now perform the same operations in the opposite direction with the right endpoints of the intervals which correspond to the right-infinite and infinite intervals, if there are any. None of the relations in S have been violated. S has been transformed into a consistent scenario for NF-IAN. For example, let F-IAN be the network in Figure 4 and S be the scenario

represented in Figure 8. Intervals B and D can be extended on the left towards infinity and, C and D can be extended on the right towards infinity without violating any of the relations in S. The resulting scenario, shown in Figure 7, is a scenario for the NF-IAN version of Figure 4.

Therefore, S is a consistent scenario of NF-IAN if and only if it is a consistent scenario of F-IAN. Q.E.D.

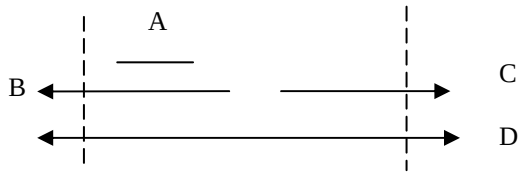


Figure 7: Chopping the non-finite intervals

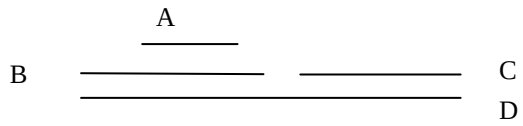


Figure 8: Finite scenario S

5 Algorithm 2: Finite Domain CSP

An IA network is traditionally defined to be a binary CSP with infinite domains. The intervals are the variables. The domain of each variable is the set of pairs of reals of the form (x,y) where x<y. The constraint between two variables i and j is the label on the edge (i,j) in the IA network.

Thornton et al. [04] show how to convert an IA network into an equivalent non-binary CSP with finite integer domains. They observe that the relative positions of the interval endpoints in an IA network can be used to determine consistency: each interval in an IA network with n intervals can be mapped to a real interval such that all the constraints on the edges hold if and only if each interval in the IA network can be mapped to an interval with integer end-points in the range 1,…,2n such that all the constraints on the edges hold.

An equivalent result was derived earlier by Ligozat [90]. In [Ligozat, 90], a purely algebraic proof in terms of p-intervals is provided for the equivalence between general IA networks and networks with integer end-points in the range of 1 to 2n.

An IA network can also be converted to a binary CSP with finite integer domains [Trudel, 03; Trudel, 05].

Although intervals can be infinite, an IA^{inf} network can be converted to a finite domain CSP. Given an n interval IA^{inf} network, we define the domains of the endpoints as follows. Each left endpoint of left-infinite and infinite intervals is equal

to 0. Each right endpoint of right-infinite and infinite endpoint is equal to 2n+1. The domain of all other endpoints is from 1 to 2n. For example in

Figure 13, the domain for the left endpoints of intervals A and D is zero, and the domain of all the other endpoints is 1..8 (in this case n=4). The result is that all the originally finite endpoints are bounded from below and above by the infinite endpoints. The remainder of the transformation to a finite domain CSP is unchanged. See Thornton et al. [04] for details.

Off the shelf finite domain CSP software can be used to solve IA^{inf} networks.

6 Algorithm 3: Under Specified IA^{inf} Networks

The algorithms presented in the previous sections assume each interval in the network is known a priori to be either finite, left-infinite, right-infinite, or infinite. This information is required to properly label missing edges. What if the type of one, many or all the intervals in the network are unknown?

We show how to label unknown nodes using an example. Consider the IA^{inf} network in Figure 9. Interval A is left-infinite, and B is finite. The types of C and D are unknown. The first step is to label the unknown intervals with all four possible types as in

Figure 10. Now view the network as a CSP problem where the nodes are the variables and their domain is its type. The next step is to make the network arc-consistent. Arc-consistency is achieved by repeatedly comparing each edge in the network with Table 3. Note that missing edges are ignored. For example, for edge AD we look at the left-infinite row of Table 3 and look for entries containing the relation "s". This constrains D to be either left-infinite or infinite. The edge BD does not further constrain D. For edge BC, the entries b, m, and o only appear in the first row of Table 3 in the finite and right-infinite columns. No processing is required for edge AB. The arc-consistent network is shown in Figure 11. We must now choose between the two types for intervals C and D. Looking at Table 3, the finite interval row contains the most relations, and the infinite row the fewest. The left-infinite and right-infinite rows contain an equal number of relations. In order to maximize the number of relations on missing edges which hopefully increases the probability of the existence of a solution, we prefer finite intervals, equally less preferable are left and right-infinite intervals, and the least preferred type is infinite. The final network is shown in Figure 12. This IA^{inf} network can then be solved using the algorithm presented in the previous section.

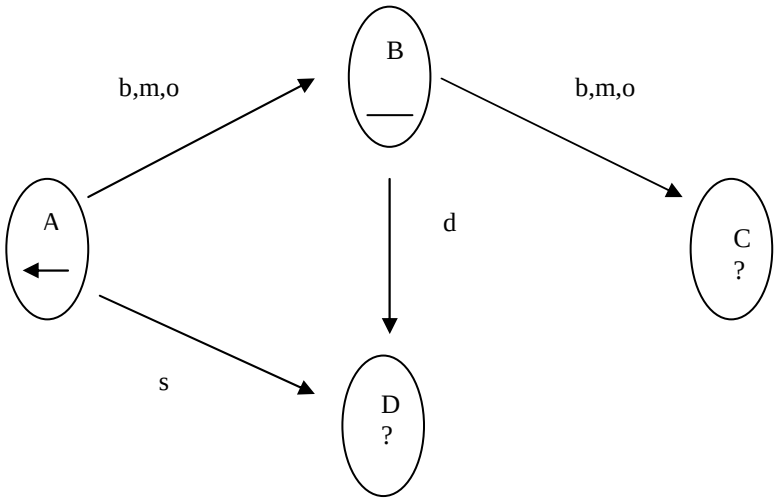


Figure 9: Intervals C and D are of Unknown Type

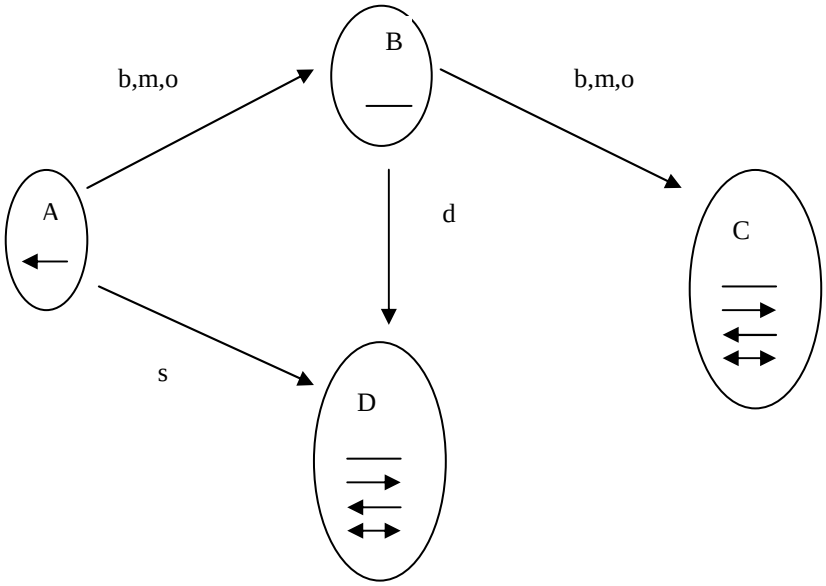


Figure 10: Missing Types are Replaced with all Possibilities

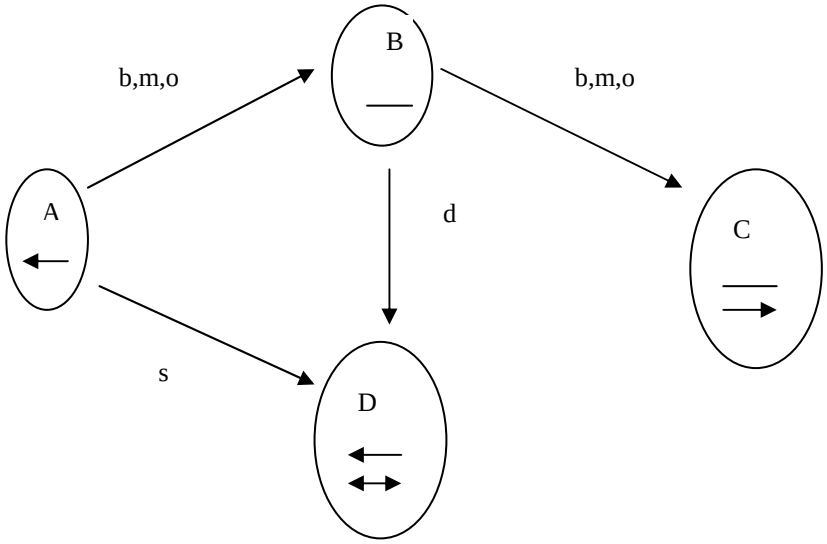


Figure 11: Arc-consistent Network

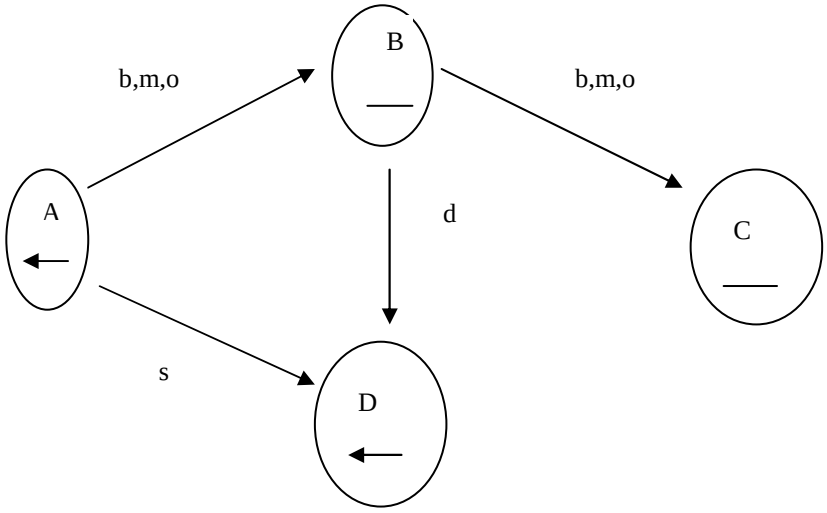


Figure 12: Final Network

To summarize, given an IA^{inf} network, we first assign all four possible types to each interval of unknown type. We then apply an arc-consistency algorithm (e.g., use AC-3). If the network is not arc-consistent, then it does not have a consistent scenario.

Otherwise, for each node which has more than one type assigned, order the types using the following preferences:

1. Finite.
2. Left or right-infinite. If both are present arbitrarily choose one.
3. Infinite.

For each node with multiple types, choose the highest ranked type. We now have a standard IA^{inf} network and the algorithms from the previous sections can be used to solve it. If the network does not have a solution, backtrack over the multiple node types one at a time until a solution is found. If no solution is found after trying all the possible combinations of node types, the network cannot be solved.

Note that if all the interval types are unknown, the type finite will be assigned to each, resulting in a standard IA network. If a solution is not found with all nodes of type finite, no backtracking over alternate node types is required. In this case, the network has no solution.

It is sometimes possible to change the type of an originally unknown interval after a solution is found. For example, a solution to the network in Figure 12 is shown in

Figure 13. The relative position of the intervals is shown in Figure 14. If there are no right-infinite or infinite intervals, and the right-most finite points belong to intervals that were originally of unknown type, then these intervals can all be allowed to extend to positive infinity without affecting the consistency of the solution. For example, the scenario depicted in Figure 15 is a solution to the network in Figure 11. A similar construction is sometimes also possible in the direction of negative infinity.

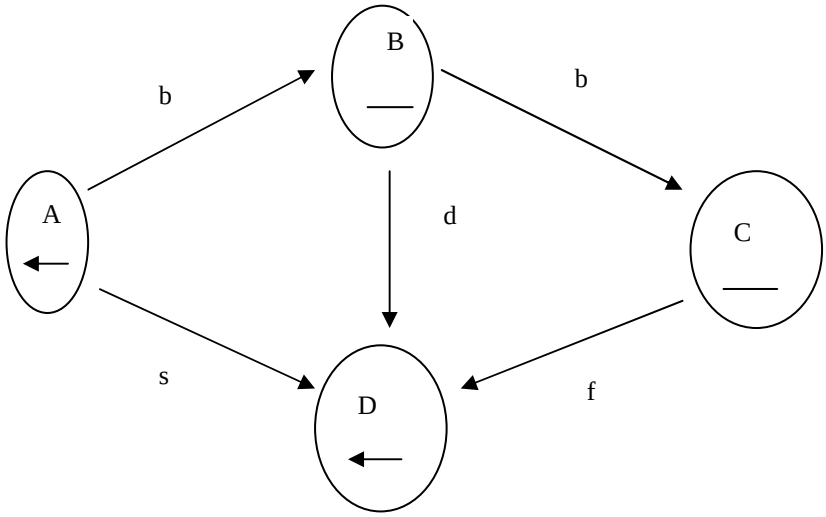


Figure 13: Consistent scenario

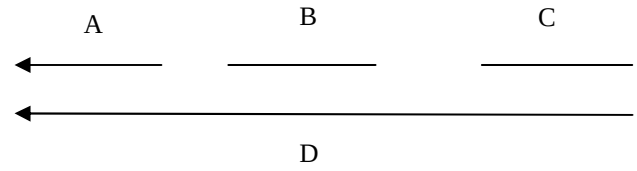


Figure 14: Scenario Drawing

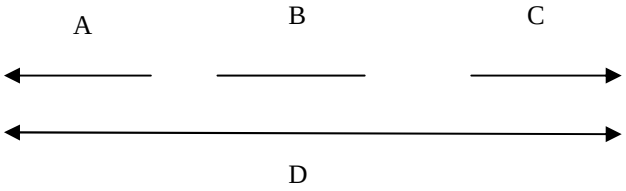


Figure 15: Infinite Version

7 Previous Work

Details of the IA network based algorithm for solving IA^{inf} networks have been first presented in [Trudel, 09]. I am not aware of any other previously published algorithms for solving IA networks containing non-finite intervals.

The concern with infinity is not limited to AI in computer science. For example, the IEEE standard 754-1985 for binary floating-point arithmetic has special values assigned to positive and negative infinity. The default in IEEE arithmetic is to round overflowed numbers to infinity [Goldberg, 91].

The problem of representing non-finite intervals was not addressed in this paper. Hobbs [02], and in a later collaboration with Pan [Hobbs, 04] present a succinct and elegant first order axiomatization of Allen's relations for non-finite intervals. Although they make minimal ontological commitments, the axiomatization contains instants, intervals can have endpoints, and relies on the fact that a left-infinite interval has no left endpoint (similarly for right-infinite and infinite intervals). This seems to imply a point based model of time. Note that a model does not necessarily need to contain an infinite number of points to represent the non-finite intervals.

Another elegant and formal axiomatization of Allen's intervals for non-finite intervals appears in [Cukierman, 04]. This axiomatization includes predicates to distinguish between the various types of non-finite intervals. This capability is missing in the logics presented in [Hobbs, 02] and [Hobbs, 04], where the user must extend the logic.

Bouzid and Ladkin [02] define temporal intervals as the union of convex finite intervals. They also define operations over these sets. One operation is union. Since

their underlying structure is the rationals, they require this infinite interval in their system since the union of a set and its complement is the set of all the rationals. Positive and negative infinity are represented by adding two points at infinity to the set of rationals. They do not consider the possible relationships between infinite sets.

Infinitely periodic temporal data has been studied in the temporal database area. Baudinet et. al. [91] and Kabanza et. al. [95] consider infinite sequences of finite intervals over the integers. Note that the individual intervals are finite; not infinite.

Another paper from the temporal database area is by Koubarakis [94] which uses the rationals as the underlying temporal structure. He does not consider infinitely periodic temporal data. But, since he is using an underlying dense temporal structure, he considers the fact that temporal information can be true at infinitely many points over a finite interval to be infinite temporal information. He does not allow non-finite intervals.

8 Future Work

Future work will include implementing and testing all the algorithms described in this paper. Also, the algorithms need to be applied to a real world example.

9 Conclusion

Allen's interval algebra axiomatization has been extended to include non-finite intervals (e.g., [Cukierman, 04] and [Hobbs, 04]). However, no one has considered the problem of finding a consistent scenario to an IA^{inf} network.

In this paper, we presented two algorithms for finding a consistent scenario of an IA^{inf} network. Both algorithms re-use algorithms and techniques developed for IA networks. The algorithms assume that each interval duration type is known a priori. We showed how to fill in missing interval types so that the algorithms can then be applied.

In a third algorithm, we showed how to solve an IA^{inf} network with missing or unknown interval types.

Acknowledgements

Author is supported by an NSERC Discovery Grant.

References

[Allen, 83] Allen, J.F.: Maintaining knowledge about temporal intervals, Communications of the ACM, 26, 1983, 832-843.

[Allen, 84] Allen, J.F.: Towards a general model of action and time, Artificial Intelligence, 23(2), 1984, 123-154.

[Baudinet, 91] Baudinet, M., Niezette, M., Wolper, P.: On the representation of infinite temporal data and queries, 10th ACM Symposium on Principles of Database Systems, Denver, May 1991.

[Bouzid, 02] Bouzid, M. Ladkin, P.: Simple reasoning with time-dependent propositions, Logic Journal of the IGPL, Vol. 10, No. 4, 2002, 379-399.

[Cukierman, 04] Cukierman, D.R., Delgrande, J.P.: A theory for convex interval relations including unbounded intervals, Proceedings of the 17th International FLAIRS Conference, Miami Beach, Florida, May 2004.

[Goldberg, 91] Goldberg, D.: What every computer scientist should know about floating-point arithmetic, ACM Computing Surveys, Vol. 23, No. 1, March 1991, 5-48.

[Hobbs, 02] Hobbs, J.R.: A DAML Ontology of Time, November 2002, retrieved on April 8 2009 from: http://www.cs.rochester.edu/~ferguson/daml/daml-time-nov2002.txt

[Hobbs, 04] Hobbs, J.R., Pan, F.: An ontology of time for the semantic Web, ACM Transactions on Asian Language Information Processing, Vol. 3, No. 1, March 2004, 66-85.

[Kabanza, 95] Kabanza, F., Stevenne, J-M., Wolper, P.: Handling infinite temporal data, Journal of Computer and System Sciences, Vol. 51, No. 1, August 1995, 3-17.

[Koubarakis, 94] Koubarakis, M.: Database models for infinite and indefinite temporal information, Information Systems, Vol. 19, No. 2., 1994, 141-173.

[Ligozat, 90] Ligozat, G.: Weak representations of interval algebras, 8th National Conference on Artificial Intelligence (AAAI-90), 1990, 715-720, Boston, USA.

[Thornton, 04] Thornton, J., Beaumont, M., Sattar, A., Maher, M.: A local search approach to modeling and solving interval algebra problems, Journal of logic and computation, 4(1), 2004, 93-112.

[Trudel, 03] Trudel, A.: How to convert a qualitative temporal CSP into a finite domain binary CSP, Spatial and Temporal Reasoning workshop held during the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003, 121-124, Acapulco, Mexico.

[Trudel, 05] Trudel, A., Zhang, H.: Exploiting the relationship between IA networks and finite domain CSPs, 12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 2005, 177-179, Vermont, USA.

[Trudel, 09] Trudel, A.: Interval algebra networks with infinite intervals, 16th International symposium on temporal representation and reasoning (TIME09), 2009, 141-146, Brixen, Italy. Also appears in the IJCAI-09 Workshop on Spatial and Temporal Reasoning, 2009 Pasadena, California, USA.

[van Beek, 96] van Beek, P., Manchak, D.W.: The design and experimental analysis of algorithms for temporal reasoning, Journal of Artificial Intelligence Research, 4, 1996, 1-18.