International Journal of Hybrid Information Technology
Vol.6, No.6 (2013), pp.195-202
http://dx.doi.org/10.14257/ijhit.2013.6.6.17

Implementing a Consistency Checker for Uncertain or Incomplete Temporal System

Yue Wang, Jixin Ma and Brian Knight

School of Computing and Mathematical Sciences
University of Greenwich, London, The United Kingdom

Abstract

Briefly presenting a generalization of Allen's interval-based approach to temporal reasoning, this paper will see point & typed-based structure of time intervals as an intended model of point & interval-based time theory to illustrate a Consistency Checker for Uncertain or Incomplete Temporal System which can be used to check whether there are circuit(s) among the temporal intervals and whether the temporal intervals are consistent or not, and this paper also succinctly discourses the future work about how to find the best solution of this checker.

Keywords: *point & typed-based structure; Consistency Checker; temporal intervals*

1. Background

To illustrate the temporal references in daily life, it is a truth universally acknowledged that temporal references play an important role in common universal references, which can be expressed with points or intervals that can be defined in temporal language such as that 'A before B' or 'A during B' and so on. Maintaining knowledge about temporal intervals, James Allen introduces a temporal logic based on intervals and their qualitative relationships in time [1]. Paper [2], detailed describing the temporal intervals, specifically indicates the time theory of thirteen relationships. Paper[3] see point&typed-based structure of time intervals as an intended model of point&interval-based time theory which will be used as basic theory in this paper.This paper will introduce a Consistency Checker for Uncertain or Incomplete Temporal System whose relative statistics can be found among [1, 2].

Analogous to the 13 relations introduced by Allen, accordingly, 30 exclusive temporal relations over time elements including both time points and time intervals can be concluded, which can be derived from the single Meets order relation and classified into the following 4 groups:

- Relations relating a point to a point:

  {Equal, Before, After}

- Relations relating a point to an interval:

  {Before, Meets, Starts, During, Finishes, Met-by, After}

- Relations relating an interval to a point:

  {Before, Meets, Started-by, Contains, Finished-by, Met-by, After}

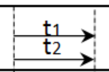
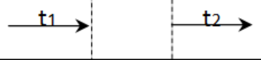
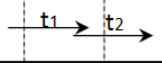
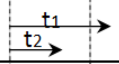
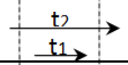
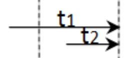
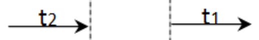
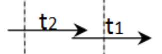
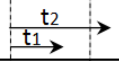
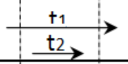
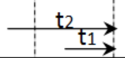
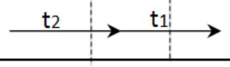
- Relations relating an interval to an interval:

  {Equal, Before, Meets, Overlaps, Starts, During, Finishes, Finished-by, Contains, Started-by, Overlapped-by, Met-by, After}

After paper [1], Allen extended his theory and showed that they applied to any fully ordered discrete or continuous relation. Then, for the relation definition capability of conceptual graphs can be used to define a wide range of temporal intervals in terms of base relations or the 'MEETS' relation, as desired. Here, T = {t_1, …, t_n} was defined as is a finite set of time elements, expressing the knowledge of what time elements are involved. Diagram 1 where t1 and t2 denote temporal intervals shows the 12 possible exclusive order relations and their simply characterized by a single axiom.

After that, in paper [4], a new theory was proposed to adopt the general time which takes a nonempty set, *T*, of primitive time elements, with an *immediate predecessor* relation, Meets, over time elements, and a *duration assignment* function, Dur, from time elements to non-negative real numbers. If Dur(t) = 0, then t is called a point; otherwise, that is Dur(t) >0, t is called an interval. The basic set of axioms concerning the triad (T, Meets, Dur) can be found in paper[4].

Table 1. Allen's Relations on Intervals and Characterized relationships

Relation	Characterized (axiom)	Graph
Equal(t_1, t_2)	$\exists t', t'' \in \mathbf{T}(\mathrm{Meets}(t', t_1) \wedge \mathrm{Meets}(t', t_2) \wedge \mathrm{Meets}(t_1, t'') \wedge \mathrm{Meets}(t_2, t''))$	
Before(t_1, t_2)	$\exists t \in \mathbf{T}(\mathrm{Meets}(t_1, t) \wedge \mathrm{Meets}(t, t_2))$	
Overlaps(t_1, t_2)	$\exists t, t_3, t_4 \in \mathbf{T}(t_1 = t_3 \oplus t \wedge t_2 = t \oplus t_4)$	
Starts(t_1, t_2)	$\exists t \in \mathbf{T}(t_2 = t_1 \oplus t)$	
During(t_1, t_2)	$\exists t_3, t_4 \in \mathbf{T}(t_2 = t_3 \oplus t_1 \oplus t_4)$	
Finishes(t_1, t_2)	$\exists t \in \mathbf{T}(t_2 = t \oplus t_1)$	
After(t_1, t_2)	Before(t_2, t_1)	
Overlapped-by(t_1,t_2)	Overlaps(t_2, t_1)	
Started-by(t_1, t_2)	Starts(t_2, t_1)	
Contains(t_1, t_2)	During(t_2, t_1)	
Finished-by(t_1,t_2)	Finishes(t_2,t_1)	
Met-by(t_1,t_2)	Meets(t_2,t_1)	

2. Implementing of the Consistency Checker

2.1 Database designing

The Figure 1 below indicates the intervals, inputted at Intervals-table by user, which were processed into a Graph-table that contains their name and the locations that will be shown on the screen.

As can be seen from the Intervals-table, user need to enter two elements, their relationships, their rights and whether the relationships is certain or not; if uncertain, then the uncertain element will be stored doubly in the Intervals-table for the other possible intervals; after that all the intervals will be characterized into the "MEETS" structure which will be delivered to Meets-table where new relationships will be added into; finally an visualizing algorithm will be programmed to convert these data into an picture which will be picked by the software that shows it on the screen.

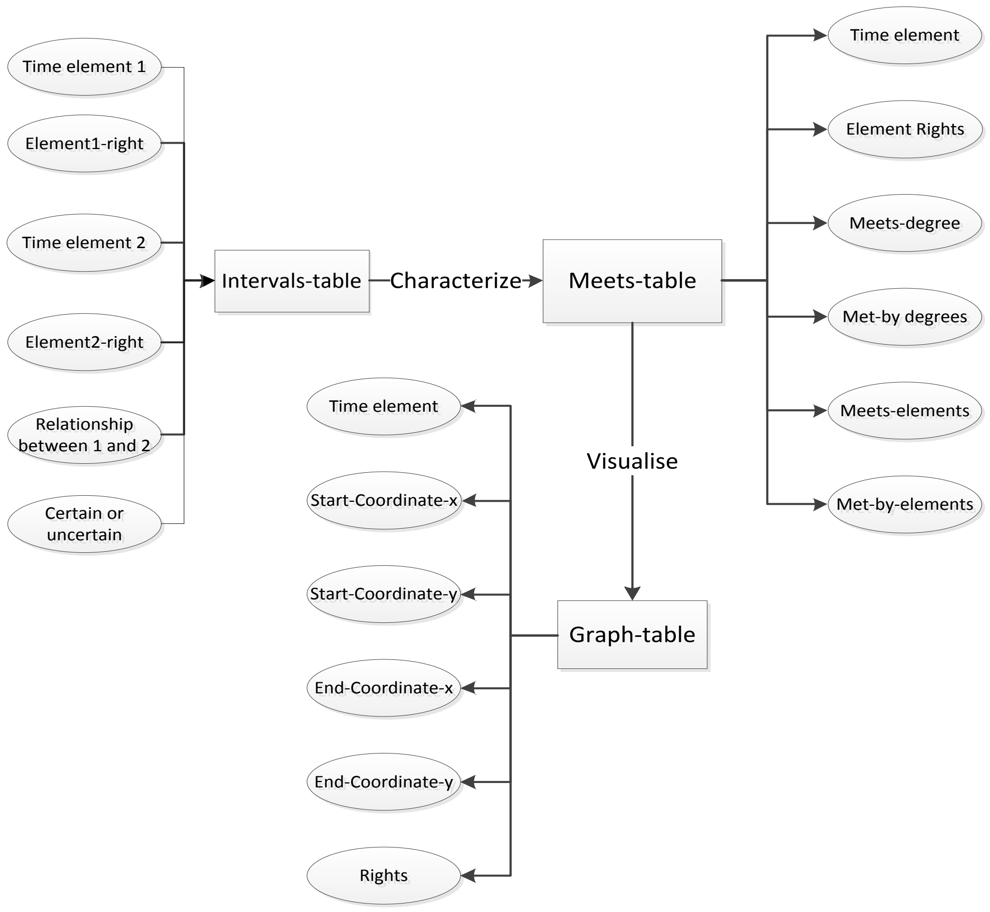


Figure 1. Database table

2.2 Algorithm designing

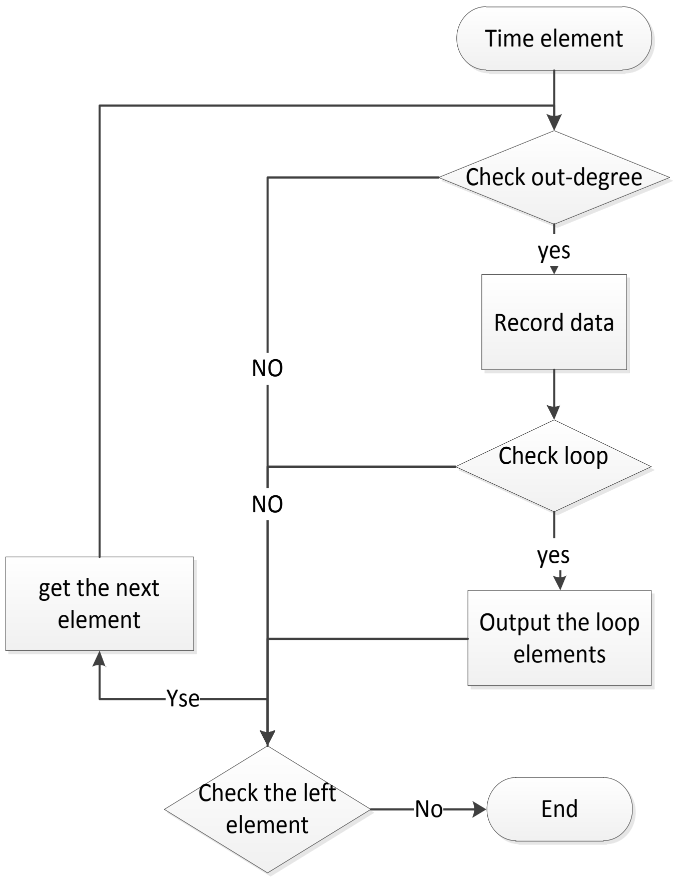


Figure 2. Loop checking flow chart

What can we get from Figure 2 is that this algorithm was design to find the loop from the structure. Picked from the database table, the data will be processed by a recursive algorithm until all the elements were checked.

One more thing that has to be emphasized is that when arriving the step 'Check the left element', the first element needs to be checked is the one which met by the original one.

For example:

$T_1 = \{t_1, t_2, t_3\}$

$M_1 = \{Meets (t_1, t_2), (Meets (t_2, t_3)\}$

$T_2 = \{t_4, t_5, t_6\}$

$M_2 = \{Meets (t_4, t_5), (Meets (t_4, t_6)\}$

While t_1 was picked from the database, t_2 will be checked first at the step of 'Check the left element'. And if t_4 was chosen, which one (t_5 *or* t_6) will be selected after is relying on the ID stored in database.

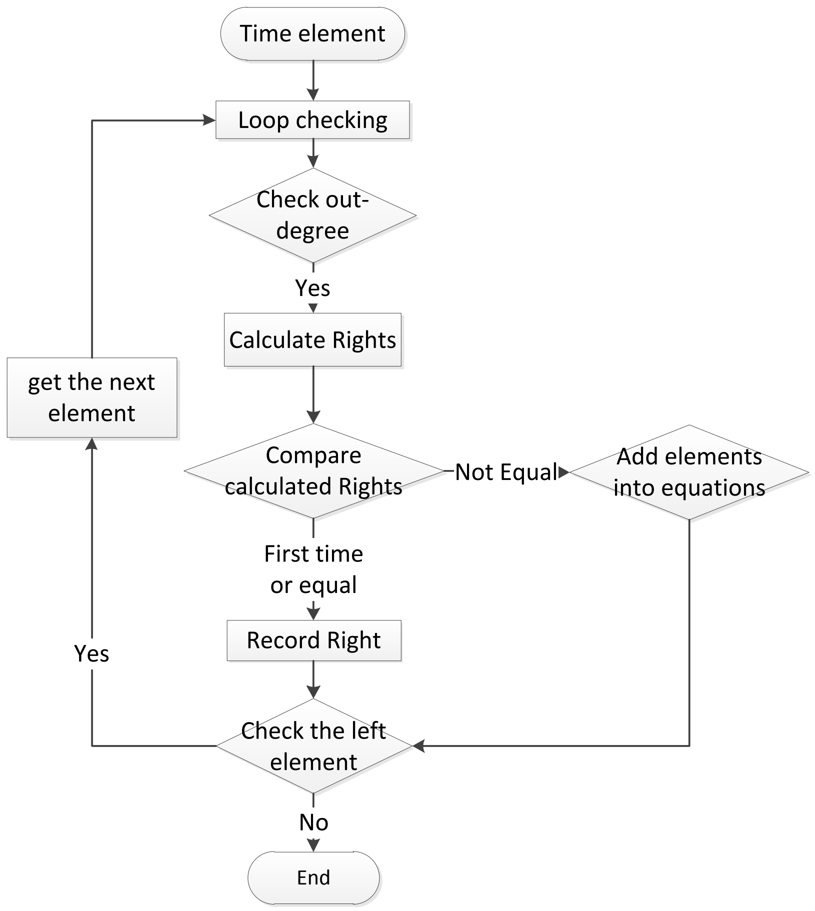


Figure 3. Consistency checking flow chart

It can be found from Figure 3 that the main step of this recursive algorithm is 'compare calculated rights' and 'Add elements into equations', which play an important role for this Consistency Checker.

Firstly, at 'compare calculated rights' stage, if it is the first time of the rights to be calculated, or if the calculated rights equal the recorded number before, then that working flow just goes to the next step will be OK.

Secondly, however, if it is a 'Not Equal', that means we can get different numbers from different paths. Therefore, this inconsistent occasion needs to be recorded and added into the equations which will be used to get the best model while the minimum action will be taken in the future.

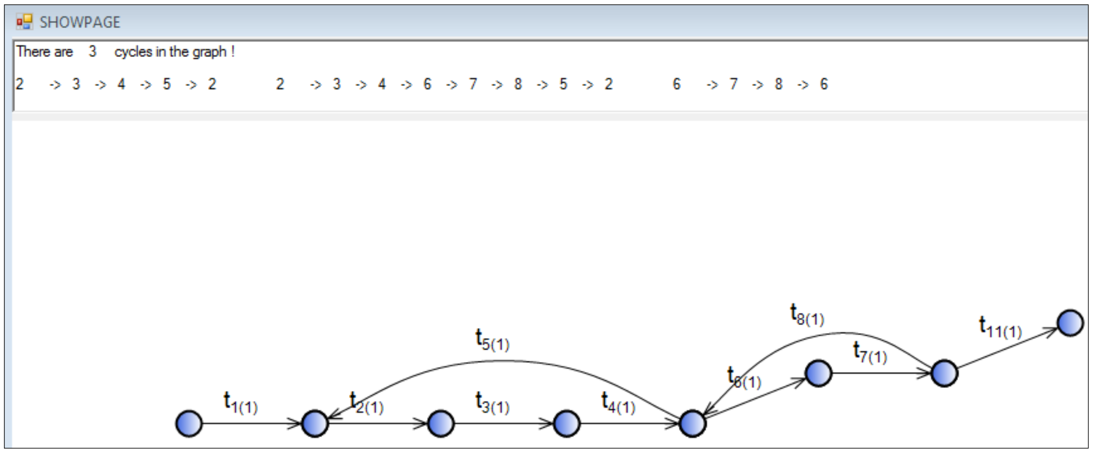
2.3 Performance evaluation

In order to examine the effectiveness of our consistency checker, the performance was measured in testing a specification of two examples below:

$T_1 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{11}\}$
$M_1 = \{$Meets (t_1, t_2), Meets (t_2, t_3), Meets (t_3, t_4), Meets (t_4, t_5), Meets (t_4, t_6), Meets (t_5, t_2) , Meets (t_6, t_7) , Meets (t_7, t_8) , Meets (t_8, t_6) ,Meets $(t_7, t_{11})\}$

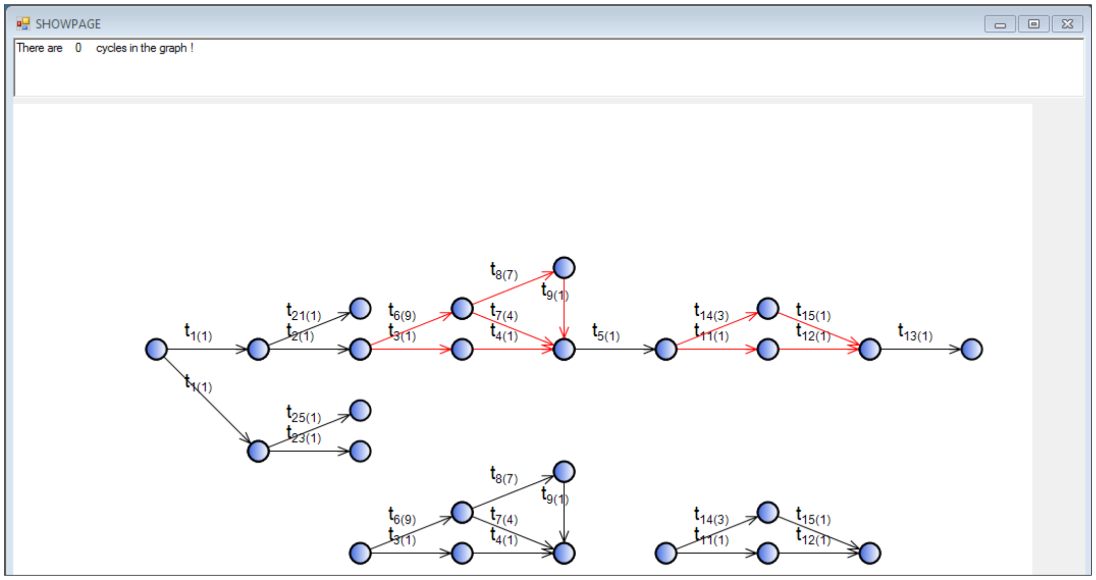
D_1 = {Dur(t_1) = 1, Dur(t_2) = 1, Dur(t_3) = 1,Dur(t_4) = 1, Dur(t_5) = 1, Dur(t_6) = 1, Dur(t_7) = 1, Dur(t_8) = 1, Dur(t_{11}) = 1}
T_2 = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{21}, t_{23}, t_{25}}
M_2 = {((Meets (t_1, t_2), Meets (t_1, t_{21})) ∨ Meets (t_1, t_{23})), Starts (t_{23}, t_{25}), Meets (t_2, t_3), Meets (t_2, t_6), Meets (t_3, t_4), Meets (t_4, t_5), Meets (t_6, t_7), starts (t_8, t_7), Meets (t_7, t_5), Meets (t_8, t_9), Meets (t_9, t_5), Meets (t_5, t_{11}) , Meets (t_5, t_{14}), Meets (t_{11}, t_{12}) , Meets (t_{12}, t_{13}), Meets (t_{14}, t_{15}), Ends(t_{15}, t_{12})}
D_2 = {Dur(t_1) = 1, Dur(t_2) = 1, Dur(t_3) = 1,Dur(t_4) = 1, Dur(t_5) = 1, Dur(t_6) = 9, Dur(t_7) = 4, Dur(t_8) = 7, Dur(t_9) = 1, Dur(t_{11}) = 1, Dur(t_{12}) = 1, Dur(t_{13}) = 1, Dur(t_{14}) = 3, Dur(t_{15}) = 1, Dur(t_{21}) = 1, Dur(t_{13}) = 1, Dur(t_{25}) = 1}

As for illustration, consider the following temporal reference (T_1, M_1, D_1) and (T_2, M_2, D_2), where, for the reason of simple expression, comma "," in M and D stands for logical connective " ".



```
There are  3   cycles in the graph !
2  -> 3  -> 4  -> 5  -> 2        2  -> 3  -> 4  -> 6  -> 7  -> 8  -> 5  -> 2        6  -> 7  -> 8  -> 6
```

Figure 4. Checking result of (T_1, M_1, D_1)



```
There are  0   cycles in the graph !
```

Figure 5. Checking result of (T_2, M_2, D_2)

The result is shown in Figure 4 about temporal graph (T_1, M_1, D_1) that there are three circuits all of which were found thoroughly. And the three cycles' pathways were illustrated with many "→" to indicate the elements and their directions.

However, in Figure 5, although there is no any loop in it, but when the checker is calculating the rights, inconsistent intervals were turned to red color and repainted below. Take t_{11}, t_{12}, t_{14} and t_{15} for example, the sum of t_{11} and t_{12} is larger than the sum of t_{14} and t_{15}, which means that it is inconsistent.

3. Concluding Remarks and Future Work

In this paper, a new consistency checker is introduced and examined for uncertain or incomplete temporal system, which can be used to find and show the circuit(s) and consistency. However, because of technical reasons, there are still two problems that need to be solved. Firstly, when the rights of the time elements are unknown, it will be impossible for the software to calculate and find. Secondly, if "X_1, X_2, X_3.... X_n" were defined to stand for the unknown numbers, an equation can be constructed to get the best solutions, but how to define the "best" solution to get the best model is still a big problem.

References

[1] J. Allen, “Maintaining knowledge about temporal intervals”, Communications of the ACM, vol. 26, no. 11, (**1983**), pp. 832-843.

[2] J. Allen and P. Hayes, “A common-sense theory of time”, Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, California, (**1985**) August 18-23, pp. 528-531, Morgan Kaufmann Publishers, San Francisco.

[3] J. Ma and P. Hayes, “Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies?”, the computer journal, vol. 49, no. 1, (**2006**), pp. 32-41.

[4] J. Ma and B. Knight, “A General Temporal Theory”, the Computer Journal, vol. 37, no. 2, (**1994)**, pp. 114-123.

[5] J. Ma, B. Knight, M. Petridis and X. Bai, “A Graphical Representation for Uncertain and Incomplete Temporal Knowledge”, gcis, vol. 1, **(2010)**, pp.117-120, 2010 Second WRI Global Congress on Intelligent Systems.

[6] J. Allen, “Towards a General Theory of Action and Time”, Artificial Intelligence, vol. 23, (**1984**), pp. 123-154.

[7] J. Allen and P. Hayes, “Moments and Points in an Interval-based Temporal-based Logic”, Computational Intelligence, vol. 5, (**1989**), pp. 225-238.

[8] J. van Benthem, “The Logic of Time”, Kluwer Academic, Dordrech, (**1983**).

[9] A. Galton, “Critical Examination of Allen's Theory of Action and Time”, Artificial Intelligence, vol. 42, (**1990**), pp. 159-188.

[10] J. Jensen, J. Clifford, S. Gadia, A. Segev and R. Snodgrass, “A Glossary of Temporal Database Concepts”, SIGMOD RECORD, vol. 21, no. 3, (**1992**), pp. 35-43.

[11] B. Knight and J. Ma, “A General Temporal Model Supporting Duration Reasoning”, Artificial Intelligence Communication, vol. 5, no. 2, (**1992**), pp. 75-84.

[12] J. Ma and B. Knight, “A General Temporal Theory”, the Computer Journal, vol. 37, pp. 2, (**1994**), pp. 114-123.

[13] J. Ma and B Knight, “Representing the Dividing Instant”, the Computer Journal, vol. 46, no. 2, (**2003**), pp. 213-222.

[14] L. Vila, “A Survey on Temporal Reasoning in Artificial Intelligence”, AI Communication, vol. 7, no. 1, (**1994**), pp. 4-28.