Contextual Conflict Determination Among Sensory Events Using Cooperating Agents

V.K.Murthy[1] , E.V.Krishnamurthy[2]

***Abstract* - A cooperating-agents based algorithm is described to detect temporal consistency among sensory events and for constraint processing. We describe the algorithm using an example. Also we describe the cooperative aspects of the agent -based algorithm using an UML activity diagram.**

***Index Terms*-Agents, Conflicts, Cooperation, Sensory agents , Temporal consistency, UML activity diagram**

I. INTRODUCTION

Temporal consistency and conflict detection among a number of related sensory events perceived by a network of agents is very important for many practical applications, Ribaric [10], including contextual-conflict determination, Murthy [8], Murthy and Krishnamurthy [9], criminology and forensic sciences. The main goal in temporal reasoning is to validate the different propositions involving time and evaluate the consistency among these statements using the various local constraints that can be either qualitative or quantitative, and arriving at a globally consistent picture of the whole scenario. This is a computationally hard problem, in general. However, if we restrict ourselves to a problem, in which any two events are related by a single temporal constraint involving positive integer inequalities (by a suitable choice of time units), then the problem is solvable in polynomial time. If there are many constraints among events, the problem needs to be broken into pieces involving the possible combinations among the various constraints. This leads to an exponential growth in complexity. In this paper, we will only deal with the simple temporal consistency problem among a set of agents. For this purpose we need to assume that the agents are using the same accurate clock and the constraints are exact.

[1] Colleges of Applied Sciences, Ministry of Higher Education, Sultanate of Oman, and Australian National University, Canberra, ACT 0200, Australia edayathuk@gmail.com and ariyalurk@gmail.com.
[2]Australian National University, Canberra, ACT 0200 Australia, Evk.Krishnamurthy@anu.edu.au

We first describe the qualitative and quantitative temporal consistency problem and its representation as a Directed graph. Then we describe the basic directed graph representation of temporal constraints among agents, where each agent corresponds to a node representing an event. Following this, we describe the multi-agent paradigm and its applications to temporal consistency problem and evaluate the global consistency based on a simple yet powerful theorem by Shostak [11]. Also we describe the agent-based detection of temporal consistency through an UML activity diagram. The last section contains the conclusion.

II. TEMPORAL CONSISTENCY PROBLEM

Given a set of sensory events, satisfying certain time constraints, the temporal consistency problem deals with reasoning of this set of events and evaluate whether the set is consistent or conflict-free. We assume that we are dealing with real-life macro-events and not cosmic or subatomic events and time has a beginning and has an arrow towards the future. We consider a simple set of constraints in which any two events are related by a single constraint in a well-defined range.

Events

Events are uniquely labelled and ordered thus: E(0), E(1), E(2) ,... E(i), E(i+1),...., E(j),....,where we assume that E(0) is the standard reference time point or the beginning of the world we consider .E(i) is before E(i+1) is denoted by E(i) < E(i+1). Accordingly we assume that the events belong to a well-founded set , defined below .

Well-founded sets

A binary relation < is *well-founded* over a class of objects, if it satisfies the no-decreasing condition; that is, there is no infinite sequence of objects decreasing with respect to that relation. In other words, there are no infinite sequences {x(0), x(1), x(2), x(3),....} of objects such that: x(0) > x(1) > x(2) > x(3)>... where > is the inverse binary relation of <. For a detailed study of well founded sets and relations, and related temporal logic, see Manna and Waldinger [7].A set in which the elements are related through a well-founded relation is called a *well-founded set*..

III. QUALITATIVE TEMPORAL RELATIONS

We use the terminology of Apt [1] and Dechter [3] to describe the qualitative temporal consistency problem and then extend it to the quantitative temporal consistency problem.

We assume that every event A has a trigger-beginning Ab and a trigger-end Ae, and the duration of an event is the time interval in integral units, between these two well-defined points. This means also for real-world events Ab occurs before Ae, or Ab<Ae and Ab=Ae, if and only if, the event is a trigger event. The trigger events are like impulse (delta) functions having no width. This approach helps us to extend the qualitative temporal consistency problem to quantitative temporal consistency.

We denote the following seven temporal relations, Apt [1], Dechter [3] in the above notation; note that we are permitted to use only the relation < to preserve the well-founded relation; however, we will write x = y, if and only, if x< y and y< x meaning that x and y are coincident.

Also the " inverse relations" – such as after, met by, started by, overlapped by, finished by, contains, as in Apt[1], Dechter [3] which are grammatical transformations from the active to passive voice in English, can equally well be represented by the following relations:

0. For every event: Ab<Ae, and Ab =Ae when A is a trigger event.
1. A before B : Ae<Bb.
2. A meets B: Ae=Bb and Ab< Bb
3. A overlaps B: Ab<Bb and Bb<Ae and Ae <Be
4. A starts B : Ab=Bb and Ae <Be
5. A during B: Bb<Ab and Ae<Be
6. A finishes B: Ae=Be and Bb<Ab
7. A equals B (coincident): Ab=Bb and Ae =Be

The above relations are qualitative, since no numerical quantity is assigned to measure the time differences. Since we have defined < and = , we can use conventional type of arithmetic to convert them to quantitative reasoning by using addition and subtraction of integers representing time steps. For example if x = y - k where k is a positive integer representing basic time units, then we can write x < y ; i.e x is k units of time before y or x+k = y. We say that two events are coincident, if they use the same clock and the events x and y occur simultaneously. This means the shortest time difference between these two events x and y is Zero or x < y and y<x; that is x and y are coincident. In fact , the composition, and disjunction among the relations (including ternary and n-ary relations) can be dealt with using the arithmetic of inequalities and the directed graph approach, to be described below.

Examples

(i) Consider the qualitative temporal logic statement: A during B and B overlaps C:

This means Bb<Ab and Ae<Be; further, since A is a real event: Ab<Ae.

Also B overlaps C means: Bb< Cb, Cb<Be and Be <Ce.

Thus Bb<Cb< Be <Ce; Bb <Ab<Ae< Be<Ce.

Although we know Bb <Cb and Bb<Ab ; further we know that Ae<Ce; but we cannnot compare Ab and Cb;

However, Ae <Ce; this means A does not finish C and A is not equal to C.

Thus A can occur in any of the five following modes:

A before C, A meets C, A overlaps C,A starts C,A can be during C.

Hence, the statements provided are therefore incomplete to resolve these cases uniquely.

(ii) Consider the qualitative temporal logic statement: A starts B and B finishes C

This means Ab=Bb and Ae <Be; Be =Ce and Cb<Bb; that is Cb <Ab and Ae <Ce

Thus Cb <Ab<Ae<Ce ; or A is during C (or C contains A).

We can also say that if A starts B and B finishes C, then A before C is inconsistent, since Cb <Ae or C before A.

(iii) Consider the qualitative temporal logic statement: Ameets B and B overlaps C:

This means Ae =Bb, Ab< Bb, Bb<Cb and Cb<Be and Be <Ce; Thus we have Ab<Ae(=Bb)<Cb <Be <Ce.

That is Ae<Cb or A before C.

(iv) Consider the qualitative temporal logic statement: A meets B and A starts B:

This means Ae=Bb and Ab=Bb or Ab=Ae. Thus A is just a trigger event.

(v) Consider the qualitative temporal logic statement : A before B and A during B.

That is Ab<Ae and Ae <Bb; and Bb<Ab; Ae<Be.

This means, Ab<Bb <Ab or Ab <Ab ; this is inconsistent.

We can now use this example to move on to quantitative temporal consistency through this example. If we assume that the duration of A is x positive units of time, and B starts after y positive units of time and ends after z positive units of time then Ab+x=Ae; Ae+y =Bb . This means Ab+x+y =Bb; A during B means Bb <Ab or Bb occurs at least one unit of time earlier than Ab. That is Bb = Ab- 1;. Combining the equations we find that Ab+x+y =Ab-1 or x+y = -1; this is impossible since x and y are positive. Hence the statement is inconsistent.

IV. QUANTITATIVE TEMPORAL CONSISTENCY

We now introduce numerical measures in the qualitative temporal relations to convert them to quantitative temporal relations. We assume that all the events described below are trigger events and time difference between the occurrence of these two events E(i) and E(j) satisfy the inequality of the form: $a \leq E(i)-E(j) \leq b$, where and b are non-negative integers. We use the following notation to map the positive integer inequalities among the events into a directed graph. Note that a and b can be zero for coincident events, and a= b, if there is no slackness in the constraint or the events occur with exact time difference.

Given any two events there is only one constraint, namely: $L(i,j) \leq E(j)-E(i) \leq U(i,j)$.

Here L(i,j) and U(i,j) are non-negative integers denoting lower and upper limits. The subtraction sign indicates obviously, E(j) is later than E(i) or equivalently, E(i) is earlier than E(j), thus satisfying the requirement that the time flows forward.

The above inequality can be split into two one-sided inequalities:

$E(j)-E(i) \leq U(i,j)$ and $E(i)-E(j) \leq - L(i,j)$

We now map the inequality to a directed graph (digraph)thus: the lower limit L(i,j) appears as an edge with

a negative weight along the j-i direction and the upper limit U(i,j) appears as a directed edge with a positive weight along the i-j direction, as shown in Figure 1. Note that U(i,j)-L(i,j)≥ 0. Also, U(i,j)= L(i,j) if the events have exactly specified time differences and U(i,j)= L(i,j) = 0 when the events are coincident.

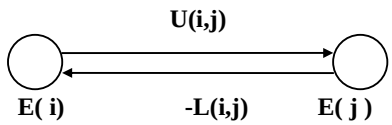

Figure 1: Mapping Inequality to a Digraph

Hence, if there are n events [i =0,1,...,(n-1)] , then we will have n(n-1)/2 directed edges in the graph each representing one side of an inequality. The adjacency matrix of this directed graph that represents the simple constraint problem will have [n + n(n-1)/2] or n(n+1)/2 entries in which n of the entries corresponding to E(i,i)= 0.

Further, note that the least time path between two connected nodes i and j representing event E(i) and E(j) respectively, is given by: Time (0,j) ≤ Time (0,i) + U(i,j).

We now state and prove a variant of a theorem, proved by Shostak [11], in the context of linear inequalities..

Theorem 1

A directed graph that describes a single temporal constraint at each edge represents a temporally consistent problem, if and only if, the sum of distances across a cycle is non-negative.

Proof:

If part: Since the elements of the set are well founded, if the problem is consistent then it implies that, they satisfy the no-decreasing condition across a cycle from the starting node of the cycle; or E(k)-E(k) ≥ 0

Only if part: If the cycle sum is negative it implies that across a cycle E(k)-E(k) < 0 ; this implies E(k)<E(k) which is not true since E(k) is non-decreasing with respect to itself and at the worst E(k)= E(k) and the cycle sum is zero. Further, if there are no negative cycles, it is a consistent problem and the least-time path between any two nodes is well-defined . This can be obtained using the shortest path algorithm for spatial distances. This is because, for any pair of connected nodes i and j, the least time path satisfies: Time (0,j) ≤ Time (0,i) + U(i,j).

Remark: If U(i,j)=L(i,j) for all i,j then all the inequalities become equalities and the cycle sums are zero in both directions, for consistent problems.

Example

A simple illustration of this theorem is in the creation of the International date- line for setting up consistency of the local time of clocks having a well-defined time difference, at different places across the globe. To the left of the date-line , in clockwise direction we add +24 hours to denote next day, and to the right of the international date line in the anticlockwise direction we subtract 24 hours to denote the previous day to the travelling clock so that the cycle sum is zero either way, and the travelling clock is synchronized with the local clock , as shown below in Figure 2. Note that the time differences are exact here and we have equalities between different nodes.

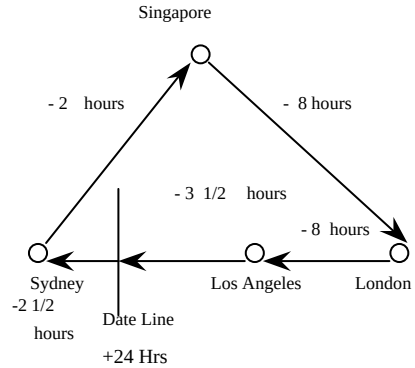

Figure 2: Consistency of International Clock

V. DETECTING TEMPORAL CONSISTENCY

A cooperating multi-agent system can be defined as a loosely coupled network of agents that interact among them and through the environment to solve a problem [6]. Operationally, the multiagent system carries out distributed computation by sending, receiving, handshaking and acknowledging messages and performing some local computations and has the following features:

1. An agent can carry out elementary computations and it knows its neighbour's names and other neighbourhood connectivity information.
2. There is a seeding (initial) agent which initiates the solution process.
3. Each agent can be active or inactive.
4. Initially all agents are inactive except for a specified seeding agent that initiates the computation.
5. An active agent can do local computation, send and receive messages and can spontaneously become inactive.
6. An inactive agent becomes active, if and only if, it receives a message.
7. Each agent may retain its current belief or revise its belief as a result of receiving a new message by performing a local computation. If it revises its belief, it communicates its revised state of belief to other concerned agents; else it does not revise its solution.

Thus:

1.Each agent offers only a partial solution to a problem and holds only a partial information.
2. Control and Data are decentralized
4. Computation is not necessarily synchronous.
5. The computation terminates when the solution is reached collectively and the multi-agent system is said to be self-stabilizing.

We now describe how several agents, Krishnamurthy and Murthy [6] can cooperate to check global temporal consistency of a temporal constraint problem. For this purpose, we consider the problem of finding a least time path between any two vertices in a directed graph whose edges have a certain assigned positive or negative costs. The vertices of this directed graph correspond to events represented by agents. Here the agents communicate directly through messages (Message passing method). Here we use the terminology similar to the Agent language KQML (Knowledge Query and Manipulation Language), designed by Finn et al. [4], consisting of the following *Six primitives*:

1. *A Performative*: A single word describing the purpose of the message, e.g.,Tell, Reply
2. *Identity of Sender*
3. *Identity of Receiver*
4. *Language used in Content*
5 *Ontology-Vocabulary*: Context within which the message content is to be interpreted
6. *Message content*

Example

Let E0,E1,E2,E3 and E4 denote sensory events constrained by the following inequalities:
(1) $10 \leq E1-E0 \leq 20$
(2) $30 \leq E2-E1 \leq 40$ or (2') $60 \leq E2-E1 \leq 70$.
(3) $20 \leq E4-E3 \leq 30$ or (3') $40 \leq E4-E3 \leq 50$.
(4) $10 \leq E2-E3 \leq 20$
(5) $60 \leq E4-E0 \leq 70$

The above inequalities are mapped into the directed graphs to describe the relationship among the events. Since there are two possible inequalities (2) &(2') and (3) & (3') , we need to consider four different possibilities. Thus the temporal constraints among the events can be represented by four different directed graphs.
Figures 3 and 4 represent the two out of four possible constraints, where the former represents a consistent situation, while the latter represents an inconsistent situation.
We will now describe the use of cooperating agents to solve this problem.

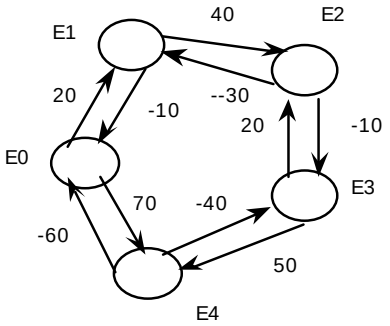

Figure 3

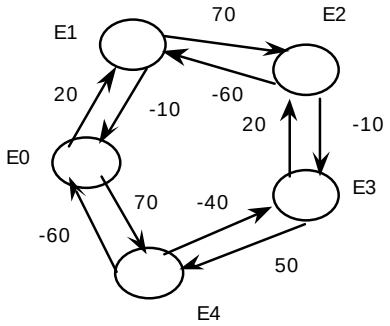

Figure 4

A. Detecting Temporal Consistency

To detect the temporal consistency , we assume that there are N agents with names identical to the nodes in the graph and each agent is connected to other agents in an isomorphic manner to the given graph. Such an assumption on the topology of the network simplifies the organizational knowledge. Also each agent knows the identity of its neighbours, the direction and cost of connection of the outgoing edges. Note that the outdegree of each node is the number of sending channels and the indegree is the number of receiving channels.The production rules for multi-agent computation are as follows:
a.Initialization and update of beliefs: Agent X (root) sends to all its neighbours Y the tuple: (X, s= 0,Y, t=0) describing the name X of the root, and the distance s= 0 from its source neighbour, its distance t= 0 from root; all the neighbours of the root handshake, receive, and store it. This corresponds to the initialization of beliefs.
Each agent Y at a distance u from X then sends its neighbour Z at a distance w from it, the tuple (Y, u, Z, u+t) describing its name, its distance u from the source neighbour, its distance u+t from the initial node. This is the initial set of beliefs of the agents.
b. Halting: Finally, the initial node checks its cycle lengths; if any one cycle is negative, it declares that the problem is inconsistent.

Example
Consider the directed graph in Figure 3, in which the edge costs are as shown; we denote the graph by the triplet , a pair of nodes (X,Y) followed by the cost s of the edge ,thus: (X,Y,c). The graph in Figure 3 is then given by, (To denote nodes, we omit the letter E and use only the suffix i of Ei as the labels):
(0, 1, 20); (1, 2, 40); (2, 3, -10); (3, 4, 50); (4, 0, -60) ; (0, 4, 70); (4, 3, -40);
(3, 2, 20); (2, 1, -30); (1, 0, -10).
We choose the vertex 0 as the root. The graph is encoded and assigned to the agents as shown in Figure 5. Here we indicate by arrows the direction of communication , and names of the communicating agents and inside the box we indicate by an ordered pair the distance of neighbour and distance of root. We apply the rules systematically.
The collective agent communication protocol and computational tree of Figure 5 is obtained from Figure 3 , using the rules described. At initiation, the node labelled E0 is the root and the seeding agent. In Figure 5, we indicate by arrows the direction of communication
among the agents, and an ordered quadruplet indicates: node name (the distance from neighbour , its distance to root) credit retained. The root E0 contains the ordered pair : (0,0).E1 contains the pair (20,20) indicating that its distance from the neighbour is 20 and from the root is 20. Note that the constraints in Figure 5 are consistent since the cycles are nonnegative, the constraints in Figure 6 are inconsistent since one of the cycles is negative.

B. UML ACTIVITY DIAGRAM

It is convenient to visualize the above interaction among the agents using the Unified modelling language (UML) diagram , Figure 7. UML 2.0 has 13 behavioural diagrams meant specially for improved understanding Booch et al [2], Holt [5] . Among these diagrams the "*Activity Diagram*" permits a very low level modelling and is suitable for our needs, since the agent actions are at a low-level. Also it specifies the dynamic behaviour of the agents, their message

and control flow and how they cooperate. Activity diagrams are made up of three basic elements: (i) Activity Node, (ii) Activity Edge, and (iii) Region.

(i) Activity Node: There are three types of activity nodes: Activity invocation, Object and control node. The Activity invocation permits us to establish traceability to the rest of the model, via operations, activities and actions.

(ii) Activity edge: This can be of two types: control flow and message or object flow.

(iii) Region: This has two main types: Interruptible activity region ands Activity partition. The former allows us to put a boundary on the diagram where activities can be interrupted; while the latter mechanism allows us to group together the different activity invocations.

VI. CONCLUSION

We described a cooperative agent-based algorithm to solve the simple temporal consistency problem among a set of events, using a directed graph representation. In this representation of temporal constraints among agents, each agent corresponds to an event. This algorithm for detecting the global consistency is based on a simple yet powerful theorem by Shostak [11]. Also we illustrated the agent-based cooperative process using an example and the UML activity diagram.

REFERENCES

[1]K.R.Apt, Principles of Constraint Programming, Cambridge University Press, Cambridge, U.K, 2004.

[2]G.Booch, J.Rumbaugh and I.Jacobson, The unified Modelling language User guide, Addison Wesley, Massachusetts, 1999.

[3] R.Dechter, *Constraint Processing*, Morgan Kaufmann Publishers, San Francisco, 2003

[4]T.Finn et al., *KQML:An Agent communication Language*, in J.M.Bradshaw (Ed),Software Agents, M.I.T.Press, Cambridge, Mass,1997, pp.291-316.

[5] J.Holt, UML for Systems Engineering, IEE, London 2004

[6]E.V.Krishnamurthy and V.K.Murthy, *Distributed agent paradigm for soft and hard computation*, Journal of Network and Computer Applications, **29**, ,pp.124-146. 2006

[7]Z.Manna and R.Waldinger, The Deductive foundations of Computer programming, Addison Wesley, Reading, Mass, 1993

[8]V.K.Murthy , Contextual-knowledge management in peer to peer computing, International Journal of Knowledge-Based & Intelligent Engineering Systems, Vol. 9, 303-314, 2005.

[9]V.K.Murthy. and E.V.Krishnamurthy, *Contextual information Management using Contract-based workflow*, Proc.ACM Computing Frontiers, CF'05, Iscia, Italy, 2005

[10] S.Ribaric, *Temporal knowledge representation and reasoning model for temporally rich domains*, Lecture notes in Artificial Intelligence,Vol. 3682**,** Springer Verlag, New York, pp.430-436,2005.

[11] R. Shostak, *Deciding linear inequalities by computing loop residues*, Journal of the ACM,**.8,** 1981, pp.769-779.

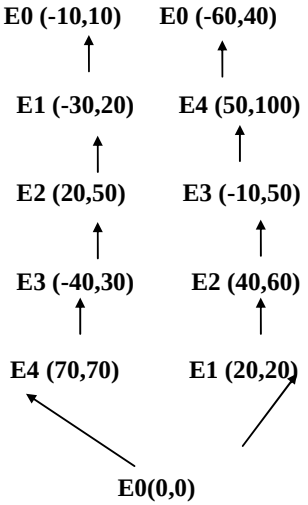

Figure 5. Agent Computation Tree

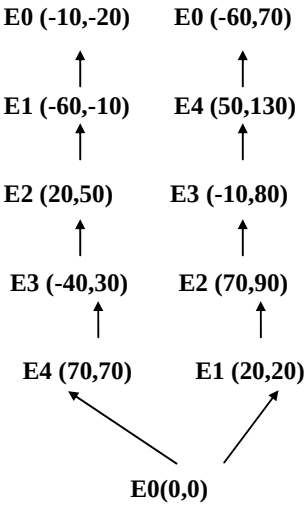

Figure.6 Agent Computation Tree

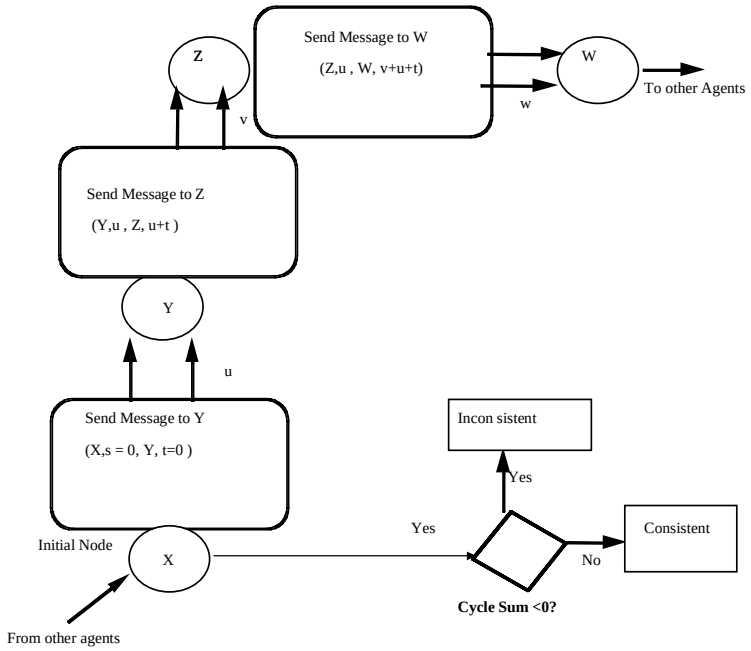

Figure 7. UML Activity Diagram