EVENT CALCULUS

Gary C. Borchardt*

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA

Abstract

This paper presents Event Calculus**, a model for representing the identifying characteristics of physical events in terms of changes in a scene and time-related combinations of other physical events. The model is used to construct a knowledge-based system for event recognition which forms a high-level description of changes in a scene, given a low-level description as input.

Time-varying information is represented in the form of "GRAPHs", data structures which plot the elements of various domains against time. Several varieties of operations are presented which map GRAPHs into GRAPHs, and representations of physical events are formed as symbolic expressions involving these operations. The paper concludes with an overview of the event recognition system, as implemented in INTERLISP on a VAX 11/780, and an example of a session with this system.

I Introduction

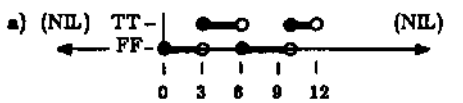
In recent years, much attention has been given to the issue of representing temporal knowledge. Several systems, including those by Vilain [1], Allen [2], Kandrashina [3] and Malik and Binford [4] have been presented for modeling temporal information and reasoning in time, following earlier works by Bruce [5] and Kahn and Gorry [0]. In addition, extensive general models of time, including framework* for causality, belief and continuous change in quantities have been proposed by Allen [7] and McDermott [8].

Somewhat removed from this thrust, however, have been a number of efforts in "event recognition," including the works of Neumann and Novak [9], Tsotsos [10], Nagel [11], Tsuji, Moriiono and Kuroda [12] and Okada [13]. The representational models employed by such systems have been quite diverse, and naturally so, as physical events themselves span an entire range from simple state changes to complex interactions of subevents and alternate possibilities. The model presented here attempts to capture a good portion of this range in the complexity of physical events.

The motivation for this research originated largely in the Event Shape Diagrams of Walts [14] and the extensive treatment of time in Miller and Johnson-Laird's *Language and Perception* [15]. The presentation of Event Calculus in this paper is necessarily brief; a more detailed account of the model may be found in [16].

II GRAPHs and SYMBOLS

Processing of temporal information in the Event Calculus model revolves around the manipulation of data structures called "GRAPHs", which correspond to piecewise-constant partial functions mapping time into the elements of particular domains. This representation captures both the notion that we may not know the behavior of a quantity or conditbn over all time (thus, a partial function) and the notion that we may not desire to record the behavior of that quantity or condition beyond the limits of some given granularity (thus, a piecewise-constant function). Figure 1 illustrates such a construction, mapping time into boolean values, and its corresponding representation as a GRAPH in LISP. Note that values are continuous to the *right* of the time points at which changes occur. "Time" is taken as the set of real numbers, with endpoints "BEGINNING" and "END" attached.



b) ((BEGINNING NIL)(0 FF)(3 TT)(6 FF)(10 TT)(12 NIL))

Figure 1. a) Piecewise-constant partial function mapping time into boolean values; b) corresponding LISP representation (GRAPH).

A quantity or condition which does not vary in time is represented by a single value called a "SYMBOL", which is generally treated in the same manner as a GRAPH mapping all time to that value.

HI Functions Taking GRAPHs as Arguments

Given the GRAPH as a fundamental data entity, it is possible to create a class of functions, each of which takes a number of GRAPHs or SYMBOLs as arguments and returns a GRAPH or SYMBOL as its result (call this a "FUNCTION"). One important restriction is placed on FUNCTIONS — the resulting value at any point in time may be a function *only* of the values of its arguments taken at that same point in time. That is to say, a certain value at a certain time in the result may not be a function of a value at some other time in one of the arguments (this facility is reserved for OPERATORS, described below).

The Event Calculus model provides three modes for specifying an optional time argument in the application of a FUNCTION or related construct. Evaluation "at" or "just before" a point in time causes a SYMBOL to be returned, while evaluation "from" one point in time "to" a second point in time causes a GRAPH to be returned. The following examples illustrate this evaluation process, involving application of the "ADD" FUNCTION in conjunction with various arguments and time specifications.

```
(ADD ((BEGINNING NIL)(0 200)(2 700)(4 NIL))
     ((BEGINNING NIL)(0 100)(3 300)(4 NIL))
     AT 3)
```

1000

```
(ADD ((BEGINNING 100)(0 200)(10 300))  500
     JUSTBEFORE 10)
```

700

• Author's current address: Jet Propulsion Laboratory, M/S 168-514, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109.

** This research was supported in part by the National Science Foundation under Grant No. NSF 81-17238.

```
(ADD 200 300 FROM 3 TO 7)

((BEGINNING NIL)(3 500)(7 NIL))

(ADD ((BEGINNING NIL)(0 280)(10 140)(15 240)(20 NIL))
     ((BEGINNING NIL)(5 200)(15 100)(25 NIL)))

((BEGINNING NIL)(5 480)(10 340)(20 NIL))
```

Note that the intervals from 10 to 15 and 15 to 20 are automatically joined in the result of the last example, aa both intervals map to the value "340". When the optional time argument is omitted, as in the last example, the time argument at the next outer level is used: in thii case, a default of "FROM BEGINNING TO END".

IV CONDITIONALs, QUANTIFIERS and OPERATORs

In addition to FUNCTIONS, Event Calculus provides three related constructs, "CONDITIONALs", "QUANTIFIERS" and "OPERATORS". *CONDITIONALS* differ from FUNCTIONS in that one or more of their arguments may remain unevaluated throughout the course of their application, dependent on the value of some other argument. This is similar to the operation of the "and" and "or" functions in LISP. CONDITIONALs are included primarily for reasons of time-efficiency in the evaluation of Event Calculus expressions. *QUANTIFIERS* repeatedly evaluate a given Event Calculus expression while binding a specified variable successively to each element of a domain, accumulating a result in some manner. *OPERATORS* are released from the "strict time mapping" restriction placed on FUNCTIONS, as described above, and thus the value at a particular time in the result of an OPERATOR may be dependent on the values of its arguments at *other* times.

OPERATORS are the most interesting of these varieties in the context of event recognition, as they provide a mechanism for expressing both cumulative properties of quantities and conditions, and interrelations of quantities and conditions in time. Following is a list describing a few of the OPERATORS provided by the implemented Event Calculus interpreter. (Examples involving application the OPERATORS "STOP" and "NEXT" may be found in Section V.)

- DDT,IDT: Discrete approximations of differentiation and integration with respect to time.
- NEXT,PREVIOUS: Shift the values of a GRAPH over one time interval to the left (NEXT) or right (PREVIOUS).
- SETPOINT: For each interval in a GRAPH, replace the value for that interval with the starting time point of that interval.
- START,STOP: Search a boolean GRAPH for "FF" to "TT" ("TT" to "FF") transitions. Return for each time interval the time of the last such transition, or "BEGINNING" if no such time exists.
- EVER,ALWAYS: For each time interval in a boolean GRAPH, return "TT" if the value of the GRAPH has ever (always) been "TT" between a specified starting time and the time of the interval in question. Otherwise, return "FF".
- FLOOR,CEILING: Search a numerical GRAPH for minimum (maximum) values. Return for each time interval the accumulated minimum (maximum) following a specified time.

V Describing Physical Events

In the implementation of the knowledge-based system for event recognition, events are modeled as FUNCTIONS, defined by Event Calculus expressions of varying complexity. The application is a "robot arm" context, with blocks-world type objects and robot "hands." One of the simpler events in this context is that of a hand holding an object. The definition of the FUNCTION "HOLD" if as follows (this if somewhat simplified - see [16]). Variables begin with a pound sign.

```
(HOLD#A#B) ::=

  (AND (SURROUND #A #B)
       (AND (DTOUOH (PART #A FINGERA) #B)
            (DTOUCH (PART #A FINGERB) #B)))
```

"AND" is a CONDITIONAL: its second argument if not evaluated for any time intenral in which its first argument is "FF" ("FF" is simply returned for such time intervals). "SURROUND", "DTOUCH" and "PART" are FUNCTIONS. In Englbh, this definition specifies that hand #A b holding object #B iff hand #A's (two) fingers surround block #B, and each finger directly touches #B. Note that no OPERATORs are required in the definition of "HOLD", as it b possible to determine whether or not a hand b holding an object at a particular time using the values of other FUNCTIONs taken only at that tame point in time.

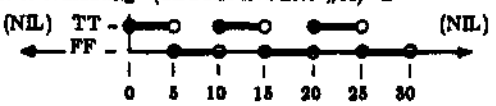
A more complex event which does require the use of OPERATORs is "GRASP", meaning "take hold of" in thb context.

```
(GRASP #A#B) ::=

  (AND (CLOSEFINGERS #A)
       (HOLD #A #B
             AT (NEXT (STOP (CLOSEFINGERS #A)))))
```

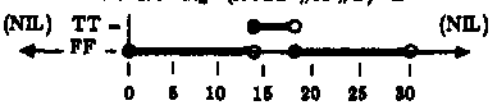
Thb definition b taken to mean that hand #A b in the process of grasping object #B iff hand #A b closing its fingers and it b also true that hand #A will be holding object #B at the end of its closing operation. The following example examines thb definition in greater detail. (Visual representations of the GRAPHs are added for clarity.)

Suppose that, for some particular hand #A and object #B, the result of evaluating "(CLOSEFINGERS #A)" is



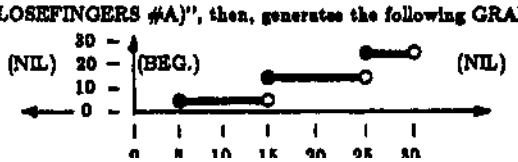
((BEGINNING NIL)(0 TT)(5 FF)(10 TT)(15 FF)(20 TT)(25 FF) (30 NIL)),

and the result of evaluating "(HOLD #A #B)" is



((BEGINNING NIL)(0 FF)(14 TT)(18 FF)(30 NIL)).

Application of the OPERATOR "STOP" to the result of "(CLOSEFINGERS #A)", then, generates the following GRAPH.



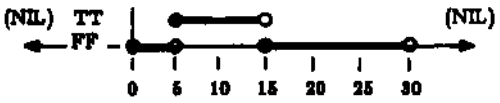
((BEGINNING NIL)(0 BEGINNING)(5 5)(15 15)(25 25)(30 NIL))

Thb GRAPH maps time to time and specifies for each interval the most recent time that the GRAPH for "(CLOSEFINGERS #A)" made a transition from "TT" to "FF" (i.e. hand #A stopped closing its fingers). A default value of "BEGINNING" b used over the interval from time 0 to time 5, as no such transition has yet occurred. When thb GRAPH b passed to the OPERATOR "NEXT", the values are shifted over one time interval to the left, specifying for each interval in time the *next* point in time at which hand #A will have stopped closing its fingers.* The result of evaluating "(NEXT (STOP (CLOSEFINGERS #A)))" b thus

((BEGINNING NIL)(0 5)(5 15)(15 25)(30 NIL)).

* This is not entirely correct, as a fool-proof version of "GRASP" would need to check that the value for each point in time in thb GRAPH b never less than the time itself.

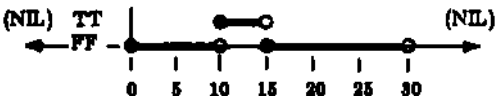
Before proceeding, it if necessary to reconsider the three modes for specifying a time argument. The previous examples employing time specifications (Section HI) used only SYMBOLs in this capacity. Whan a GRAPH appears as a time specification, this is interpreted as a request to evaluate over all time, substituting time specifications indicated by *the GRAPH* for actual intervals in time. In the case of "(HOLD #A #B AT (NEXT (STOP (CLOSEFINGERS #A))))" in the definition of "GRASP", evaluation in this manner produces the following GRAPH.



((BEGINNING NIL)(0 FF)(5 TT)(15 FF)(30 NIL))

For each interral in this GRAPH, the value indicated represents the result of evaluating "(HOLD #A #B)" at the point in time when, with respect to *that particular interval,* hand #A will next stop closing its fingers.

As a final step in the eraluation of "(GRASP #A #B)", the conjunction of the original "(CLOSEFINGERS #A)" GRAPH and the above GRAPH is taken, resulting in the following.



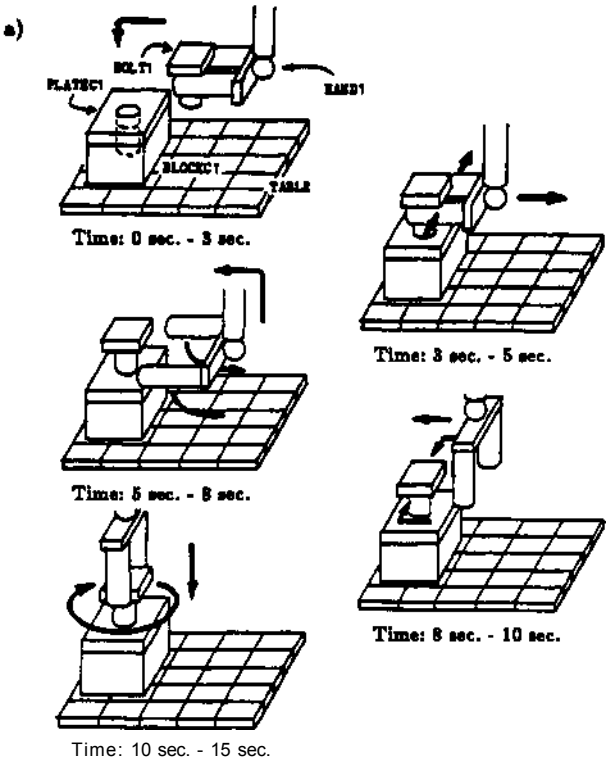
((BEGINNING NIL)(0 FF)(10 TT)(15 FF)(30 NIL))

Hand #A is seen to be grasping object #B from time 10 to time 15. This is because hand #A is closing its fingers orer this interral, and it is true orer the entire interral that hand #A will be holding object #B at the point in time in which it next stops closing its fingers. Orer the intenrals from time 0 to time 5 and from time 20 to time 25, hand #A is closing its fingers, but it is not true that it will be holding object #B at the end of this action. Likewise, from time 15 to time 18, hand #A is holding object #B, but no longer is it closing its fingers.

VI An Event Recognition Session

The representation scheme described above has been used to construct a knowledge-based system for recognising the occurrences of physical events and forming a high-level description of changes in a scene given a relatively lowlevel description of those changes as input. The system consists of the Event Calculus expression interpreter, a knowledge base of defined FUNCTIONS for 28 events in the "robot arm" context, along with approximately 100 other "supporting" FUNCTIONS, and finally, a control mechanism which proceeds In a mixed bottom-up/top-down manner in identifying the occurrences of events. Events are organised in a hierarchical fashion from those relating to positions and orientations of objects at the bottom to those concerning the fastening and unfastening of objects to other objects with a bolt, stacking and unstacking objects and so forth at the top. It is envisioned that such a system could be used as one component in the construction of a "teachable robot," serving to codify the actions taking place in the demonstration of a given task.

The input to this system is assumed to be provided by a computer vision system which tracks three-dimensional objects in space, returning coordinates of these objects over time, plus a few other useful items such as contact between objects and support of one object by another. The example given below was run using a simulated data set of this sort, amounting to approximately 1200 lines of LISP-coded input and specifying the low-level changes occurring over a fifteen second interval in a particular scene, with one second spacing between measurements. Figure 2 depicts the sequence analysed and includes excerpts from the actual input file for the session. "BOLTI", "HANDI", "PLATEC1" and "BLOCKCI", as indicated in Figure 2, are "constructions" containing several parts each, these parts corresponding to a simple geometrical shapes.



b) (ENTER 'POSITION
... (FINGERA1 X) ((BEGINNING NIL)(0 8.5)(I 4.5)(2 3)
(5 5)(6 7.5)(8 5)(8 2)(16 NIL))
... (FINGERA1 Y) ((BEGINNING NIL)(0 4)(4 3)(10 3.5)
(11 6.5)(12 3.5)(13 6.5)(14 3.5)
(15 8.5)(16 NIL))
... (FINGERA1 Z) ...

(ENTER 'DTOUCH
... (FINGERA1 SHAFT1) ((BEGINNING NIL)(0 TT)
(4 FF)(16 NIL))

Figure 2. a) Sequence of actions used in the event recognition session; b) Excerpts from the input file for the session, indicating the position of one finger of the hand and contact between that finger and the shaft of the bolt.

The resultant description as produced by the event recognition system consists of the following.

(HAND1 IS STACKING BOLT1 ON BLOCKC1
FROM TIME 0 TO TIME 3)
(HAND1 IS STACKING BOLT1 ON PLATEC1
FROM TIME 0 TO TIME 3)
(HAND1 IS INSERTING BOLT1 INTO PLATEC1
FROM TIME 0 TO TIME 3)
(HAND1 IS UNGRASPING BOLT1 FROM TIME 3 TO TIME 4)
(HAND1 IS PICKING BOLT1 UP FROM TIME 4 TO TIME 10)
(FINGERA1 IS ROTATING FROM TIME 5 TO TIME 6)
(FINGERB1 IS ROTATING FROM TIME 5 TO TIME 6)
(PALM1 IS ROTATING FROM TIME 5 TO TIME 6)
(HAND1 IS GRASPING BOLT1 FROM TIME 8 TO TIME 10)
(HAND1 IS TURNING BOLT1 FROM TIME 10 TO TIME 15)
(HAND1 IS FASTENING PLATEC1 TO BLOCKC1
USING BOLT1 FROM TIME 10 TO TIME 15)
(HAND1 IS HOLDING BOLT1 FROM TIME 15 TO TIME 16)

It should be noted that the goal of the knowledge-based system was not only to identify the events which had occurred in the changing scene, but also to provide a description of these changes in terms of the highest-level events, with lower-level events "peeking through" only where no higher-level events had occurred. This goal was achieved by a process of "suggestions and explanations," with lower-level events "suggesting" the possible occurrences of related higher-level events, and higher-level events "explaining" occurrences of lower-level events. In this manner, the "HOLD" event listed as occurring from time 15 to time 18 in the output of the above example is the last remaining piece of an event originally identified as occurring from time *10* to time 16. The segment from time 10 to time 15 has been removed from the description following identification of the higher-level events "TURN" and "FASTEN" which have provided explanations for the "HOLD" event. As no explanations for the "HOLD" event have been generated over the interval from time 15 to time 16, it remains in the final description covering this interval.

It may also be noted that some degree of seeming redundancy was left in the above description by the system in cases where one event was part of another event but not a necessary component in that event. For instance, it would not be necessary for "HANDI" to turn "BOLTI" in order to accomplish the fastening operation, as this could be aided by another hand turning the other objects.

VII Concluding Remarks

The Event Calculus model is still not ironed out completely. Problems with edge effects in OPERATORS have not been handled wholly to satisfaction, and SYMBOLs cannot always be replaced by GRAPHs mapping all time to a single value. Nevertheless, the Event Calculus formalism provides a powerful tool for the manipulation of time-related information. It may be possible as well to extend the GRAPH structure in a simple manner to include variables as time coordinates. With such a scheme, a time reasoning system such as proposed by Vilain or Allen could be incorporated together with the model for maintaining the integrity of such GRAPHs and producing alternate GRAPH possibilities when no exact ordering can be determined, yet calculations are desired. Another approach would be to extend a conventional semantic network by this formalism, incorporating GRAPHs into the "aspect" mechanism of various attributes and including a suitable set of FUNCTIONS and OPERATORS for the manipulation of these GRAPHs.

Acknowledgements

The author wishes to thank Dr. David L. Walts and the members of the Advanced Automation Research Group at the Coordinated Science Laboratory for their comments and advice offered during the course of this research.

References)

[1] Vilain, M.B., "A System for Reasoning About Time." In *Proc. AAAI-82,* Pittsburgh, PA, 1982, 197-201.

[2] Allen, J.F., *Maintaining Knowledge About Temporal Intervals,* TR 86, Computer Science Dept., Univ. Rochester, 1981.

[3] Kandrashina, E.Yu., "Representation of Temporal Knowledge." In *Proc. IJCAI-88,* Karlsruhe, W. Germany, 1983, 346-348.

[4] Malik, J., and Binford, T.O., "Reasoning in Time and Space." In *Proc. IJCAI-8S,* Karlsruhe, W. Germany, 1983, 343-345.

[5] Bruce, B.C., "A Model for Temporal References and Its Application in a Question Answering Program." *Artificial Intelligence 8,* 1972, 1-25.

[6] Kahn, K., and Gorry, GA., "Mechanising Temporal Knowledge." *Artificial Intelligence 9,* 1977, 87-108.

[7] Allen, J.F., "Towards a General Theory of Action and Time." *Artificial Intelligence 28,* 1984, 123-154.

[8] McDermott, D., "A Temporal Logic for Reasoning About Processes and Plans." *Cognitive Science 6,* 1982, 101-155.

[9] Neumann, B., and Novak, H.-J., "Event Models for Recognition and Natural Language Description of Events in Real-World Image Sequences." In *Proc. IJCAI-88,* Karlsruhe, W. Germany, 1983, 724-726.

[10] Tsotsos, J.K., "Temporal Event Recognition: An Application to Left Ventricular Performance." In *Proc. IJCAI-81,* Vancouver, B.C., 1981, 900-907.

[11] Nagel, H.-H., "Analysing Sequences of TV-Frames." In *Proc. IJCAI-77,* Cambridge, MA, 1977, 626.

[12] Tsuji, S., Morisono, A., and Kuroda, S., "Understanding a Simple Cartoon Film by a Computer Vision System." In *Proc. IJCAI-77,* Cambridge, MA, 1977, 609-610.

[13] Okada, N., "SUPP: Understanding Moving Picture Patterns Based on Linguistic Knowledge." In *Proc. IJCAI-79,* Tokyo, Japan, 1979, 690-692.

[14] Walts, D.L., "Event Shape Diagrams." In *Proc. AAAI-82,* Pittsburgh, PA, 1982, 84-87.

[15] Miller, GA., and Johnson-Laird, P.N., *Language and Perception,* Harvard University Press, Cambridge, MA, 1976.

[16] Borchardt, G.C., *A Computer Model for the Representation and Identification of Physical Events,* Master's Thesis, Report T-142, Coordinated Science Laboratory, Univ. Illinois at Urbana-Champaign, 1984.