Interval Temporal Logic in General-Purpose Situational Simulations

Eddy M. Rojas[1] and Amlan Mukherjee[2]

Abstract: The need for contextually rich educational environments in construction engineering and management calls for the development of situational simulations. Situational simulations emulate real processes and provide temporally dynamic clinical exercises that expose participants to rapidly unfolding events and the pressures of decision making. A survey of simulations of construction management processes and construction operations shows that commonly used discrete event simulation paradigms are unsuitable for representing actions and events in interactive general purpose situational simulations for the construction domain. Instead, this paper argues that a definition of the situational environment using the semantics of constraint satisfaction and an interval representation of time is more appropriate for representing activities, events, actions, and situations relevant to the construction domain. This paper also illustrates how this new paradigm facilitates the implementation of a reasoning mechanism that can be used by a software agent to perceive present actions and predict the future evolution of a simulated environment.

DOI: XXXX

CE Database subject headings: Simulation models; Construction management; Engineering education.

Introduction

Traditional construction education follows the Cartesian view of mind–matter dualism where the learner and the learning context are detached. As a result, concepts are presented as fixed, well-structured, independent entities and classroom activities are disconnected from authentic context resulting in fragmentation and specialization of courses and educational experiences. This fragmentation of knowledge has been identified in the construction domain (Chinowsky and Vanegas 1996; Fruchter 1997) and is partially responsible for the polarization of learner and learning context.

This understanding has led researchers to explore alternatives in construction education using gaming and simulation environments such as *Superbid* (AbouRizk 1993), *STRATEGY* (McCabe et al. 2000), *ICMLS* (Sawhney et al. 2001), and *VIRCON* (Jaafari et al. 2001). Some of these programs have been inspired by earlier research projects such as *CONSTRUCTO* (Halpin and Woodhead 1970) and *AROUSAL* (Ndekugri and Lansley 1992). These efforts have provided a stepping stone towards creating participatory, contextually rich educational environments.

Rojas and Mukherjee (2003b) argue that situational simulations can be effectively used in developing contextually rich educational environments to train decision makers in construction. They can emulate real construction management processes and provide temporally dynamic clinical exercises that expose participants to rapidly unfolding events and the pressures of decision making. As the participant reacts to critical simulated situations, the simulated environment responds to their manipulations by challenging them to use their knowledge and skills to experiment and solve problems in a dynamic setting where conditions constantly change in response to their actions.

Rojas and Mukherjee (2003b) have further gone on to describe the conceptual framework, which provides the foundation for the *Virtual Coach*. The *Virtual Coach* is a general purpose situational simulation for the construction engineering and management domain. The conceptual framework consists of a process model, a product model, and an information model. The product model of the simulation represents the constructs and the visualized information in the simulated environment (Rojas and Mukherjee 2003c). The information model encompasses information about the project that is being simulated. The information is coded into databases and knowledge bases. While the database has information about the "As-Planned" execution of the project, the knowledge base has knowledge about actions and events specific to the context of the construction project being simulated. Rojas and Mukherjee (2003a) explain the flow of information in the situational simulation. The process model is defined by constraints, dependencies, attributes, and events. Rojas and Mukherjee (2003b) further go on to develop a set of equations, which can be used to model construction management processes. In this paper, the writers introduce semantics to represent and reason about actions, events, and situations in situational simulations such as the *Virtual Coach*.

In Rojas and Mukherjee (2003b) the authors have also argued, that in comparison to other situational simulation platforms like *AROUSAL* (Ndekugri and Lansley 1992) and *CONSTRUCTO* (Halpin and Woodhead 1970), the *Virtual Coach* platform not only simulates construction management processes but also has the capability to simulate a very diverse set of events, dealing

[1]Assistant Professor, Dept. of Construction Management, Univ. of Washington, 116 Architecture Hall, Seattle, WA 98195-1610. E-mail: er@u.washington.edu

[2]Graduate Student, Dept. of Civil Engineering, Univ. of Washington, 116 Architecture Hall, Seattle, WA 98195-1610. E-mail: amlan@u.washington.edu

Note. Discussion open until June 1, 2005. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on October 10, 2003; approved on April 2, 2004. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 19, No. 1, January 1, 2005. ©ASCE, ISSN 0887-3801/2005/1-1–XXXX/$18.00.

with bad weather, labor management, space management, material allocation, and equipment management, within a general purpose framework, and an interactive user interface; challenging the participants to complete the project within limitations of budget and time in the face of critical events.

In order to come up with a situational simulation environment, which is general purpose in nature, we chose the semantics of temporal representation and constraint satisfaction. The following sections explore the reasons for this choice in more detail. A high level intuition is as follows: First, almost any problem that can be quantified can be expressed as a constraint satisfaction problem or a problem that can be solved by searching for an appropriate solution from within a given finite sample space. Second, the axioms of time being universal and verifiable, temporal reasoning can apply to formal systems across a very wide range of problems. In this paper, formal semantics are introduced for representing and reasoning about actions, events, and activities in a general purpose situational simulation using concepts of constraint satisfaction and interval temporal reasoning (Allen and Ferguson 1994). While this research was motivated by the development of the *Virtual Coach* in specific, it has implications in the future development of general purpose situational simulation environments for the construction domain in general.

Simulations in Construction

A survey of simulations in construction engineering and management suggests that these can be classified using three different approaches. The first approach classifies simulations based on whether they are simulating construction management processes or construction operations. The second approach to classifying simulations is based on whether they are of a special purpose or a general purpose in nature. Special purpose simulations are restricted in scope to particular operations like tunneling or a particular management process like bidding. General purpose simulations allow for greater flexibility of scope since they are programmable.

The third approach to classifying simulations can be based on how interactive they are. Situational simulations are temporally dynamic and interactive in nature. In their simplest form, simulations of construction processes use a set of initial conditions and parameters, and a well-defined model to project outcomes regarding a simulated operation. For example, given information regarding the availability of trucks and loaders, their unit costs and the amount of earth to be moved, a process simulation would be able to project the total time and cost for an excavation operation. Situational simulations also have a well-defined model and a set of initial conditions, but as the simulation proceeds the system generates random events and expects the user to react to such events. How the simulation evolves is completely dependent on the underlying process model (Rojas and Mukherjee 2003b) used, the way the events are generated and user interaction.

Such an interactive simulation is very useful in developing learning environments in which the participant is capable of exploring "what-if" scenarios involving construction management processes. For instance, in a situational simulation of an earth moving operation participants might be in a situation where they have to deal with finishing an operation within time and budget constraints, under the influence of bad weather and a labor strike. Most of the surveyed construction simulations have very limited user interactivity.

The *Virtual Coach* is classified as a general purpose simulation of construction management processes. The relationship of the

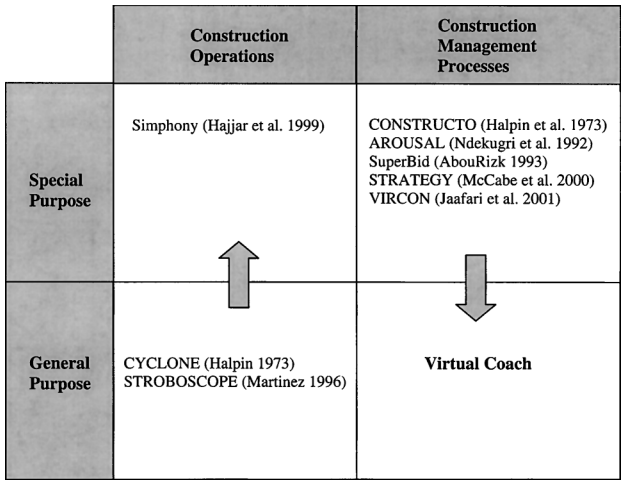

Fig. 1. Relationship between virtual coach and other construction simulation environments

Virtual Coach to other construction simulation environments is illustrated in Fig. 1. It is also important to notice that research regarding simulations of construction operations have been moving from the general purpose to the special purpose, while research regarding simulations of the construction management process has progressed in the opposite direction. In order to develop a general purpose simulation environment, the authors looked at existing simulation paradigms that have been used in general purpose construction operation simulations. In the next section a review of the existing simulation paradigms in construction is presented and the case is made regarding why these cannot be used for the *Virtual Coach*.

Simulation Paradigms in Construction

Martinez and Ioannou (1999) explain in detail the essence of construction simulation systems and justify the use of discrete event simulations for modeling construction operations. They go on to study the applicability of the activity scanning and process interaction simulation strategies to construction operations. Gil and Tommelein (2001) have also discussed the Event Scheduling paradigm.

Simulation languages such as *STROBOSCOPE* and *CYCLONE* have for a long time provided a general and special purpose framework for simulating construction operations and construction management processes, with absent or limited interactivity. They are based on the *Activity Scanning* simulation paradigm, which treats activities and events as time points, and do not include any temporal reasoning.

Activity scanning simulation models are based on a set of "activities" each of which has a set of defined conditions and outcomes. Using a simple example, the "activity" in this context typically represents a single construction task and a construction operation can be simulated by a sequence of such activities. Hence, an earth moving operation can be represented by the activities: *PushLoad*, *BackTrack*, *Haul*, *DumpAndSpread*, and *Return*, each of which has a condition and an outcome (Martinez and Ioannou 1999). An activity cannot occur if the condition is not fulfilled and when it occurs it always produces the predicted outcome. This scheme provides a way of representing simple networks, which describe the relationships between the activities, conditions, and outcomes using directional arcs. The direction of

the arcs goes from condition to activity to outcome. Such networks are referred to as activity cycle diagrams (ACDs). The major languages used for modeling construction simulation, namely *CYCLONE* and *STROBOSCOPE*, both use ACDs.

In discrete event models, like the ACD, discrete items change state as events occur in the simulation. The state of the model changes only when those events occur. The passing of time has no direct consequences. The advance of simulated time is hence a progression from event to event and often the temporal distances between these events are unequal. This representation paradigm creates problems in effectively representing multiple parallel events, which span over intervals of time.

Typically, events in the construction domain span over intervals of time and often more than one event occurs concurrently. If in the middle of a project there were n parallel construction activities in progress, the discrete event simulation paradigms would be limited in their ability to express more than one event at the same time; especially if the events have different durations. Reasoning about activities and events, which overlap in time and have temporal dependencies is also very difficult to express using the time-point approach in discrete event simulation paradigms. Using an interval representation of time allows for a successful representation of parallel events and the temporal relationships between activities and events, beyond start-to-finish relationships. In a subsequent section a formal comparison of time point representation and time interval representations in the construction domain is explored.

The *Virtual Coach* is a cycle-based simulation. Time, in the simulation, advances in equal contiguous increments. Events are represented by intervals, which are bounded by time points. Assertions about the simulation hold true over intervals of time, and the intervals of time are related to each other by semantics of interval temporal logic as propounded by Allen and Ferguson (1994). Finally, in developing a general purpose, simulation paradigm it is necessary to be able to represent and reason about broad classes of problems. The following section broadly classifies problems in the construction domain into constraint satisfaction and planning.

Construction Management Problem

Simulations are based on theoretical abstractions of the real life processes they emulate. The understanding of complex real life problems at the highest level starts with such abstractions. Dissociating detail from the underlying thread of reasoning can theoretically derive abstractions for processes. Similarly, the abstraction presented in this section is a first step at understanding the common threads of reasoning underlying the general class of construction management processes. It allows for the theoretical classification of problems in the construction domain, and is a stepping stone to developing a general purpose framework, which can handle a diverse set of problems within the construction domain. In the authors' quest for an abstraction of processes in the construction management domain, a hypothetical problem solver and a hypothetical agent are used.

An agent can perceive its environment through sensors and can act upon that environment through effectors (Russell and Norvig 2002). Agents are attributed a notion of intelligence. They can reason logically and act autonomously (free of human control) towards a goal. They are aware of the repercussions of their actions on the environment and dynamically integrate their experiences into existing reasoning mechanisms. In the computer mediated simulation domain there can be two kinds of agents: software agents (programs) and humans (interacting with a computer mediated environment). The human agent is henceforth referred to as the Participant.

A problem solver is a component of an agent (Talukdar, private communication 1998). Problem solvers perceive problems in the environment and solve them using a set of defined tools. While the problem solver allows the abstraction of classes of problems involved, the notion of intelligence in the agent allows the grasp of the underlying threads of reasoning in the world of construction. The problems in the preplanning and the implementation phases of a construction project are investigated next.

A state is a discrete structured representation of a problem. The set of all possible states defines the state space. The state space is finite. Search based problem solvers return a sequence of state transformations, which will allow an agent to transform an initial state to a goal state. The sequence of state transformations is arrived at using a goal test function that returns an index, which indicates how close a particular state is to the goal state. Such a sequence is generated by algorithms, which search the state space at every step to locate the state, which takes it closest to the goal state.

In context, to solving a problem, which deals with creating a construction plan, the state space consists of all possible partial and complete activity schedules. The initial state is an empty schedule and the goal state is the resource-loaded schedule. Successor functions are programs that allocate resources to activities from a set of available resources, to generate subsequent state descriptions. Successive states are partial schedules. In a schedule the activities take time and the resources are associated with costs. Allocation of resources to the activities is governed by a set of constraints. The same resources cannot be allocated to more than one activity simultaneously and only specific activities can be assigned specific resources. Labor and equipment allocation should be distributed to achieve optimal cost. Constraints defined on the ordering of activities are defined as precedence constraints. The problem formulation, in this case, is to assign resources to all activities, in keeping with resource and precedence constraints. This reduces the problem to a constraint satisfaction problem CSP.

Constraint satisfaction problems deal with the state in greater detail. The state is defined as a set of variables each of which is associated with a finite domain. The set of constraints define all the allowable combinations of values that the variables defining the state can take up. In the construction domain a state is defined by a schedule, which is a set of activities. Each activity is defined by variables, which describe the resources that the activity has been allocated. The goal test is the satisfaction of a set of constraints over a set of variables. The set of constraints are classified as the precedence constraints and the resource constraints.

The above analysis allows us to conclude that during the preconstruction phase the problem of creating a resource loaded activity schedule, also referred to as the "As-Planned" schedule, is a constraint satisfaction problem. A search based constraint solver can solve it. A number of research efforts support this claim. Succur and Grobler (1996) suggest a CSP formulation for the construction project planning. They represent a single project scheduling problem S by the tuple $S_s=(\Gamma, P, \Lambda)$ where $\Gamma=R_l$ is a set of resources, P is a representation of the project as a set of tasks which are represented using time intervals, and Λ is a set of agents which go about solving the constraint satisfaction problem. Such a structure essentially represents a set of precedence constraints (which they refer to as temporal constraints) and implicit

resource constraints. They further go on to illustrate a solution by using forward-checking constraint propagation algorithms using pruning and conflict resolution. Hammond et al. (2000) suggest the use of a partitioned dependency structure matrix to represent information about a schedule. A closer analysis shows that the partitioned matrix is a state representation of precedence and resource dependencies in a schedule. There exists a finite set of such matrices for any given project. By reordering the sequence of design tasks to maximize the number of design tasks below the diagonal, the *DSM* software essentially deals with precedence constraint satisfaction by making state transformations. *WorkPlan* (Choo et al. 1999) also uses resource and precedence constraint satisfaction in the *WorkPlan* implementation.

During the project implementation phase, project managers like to stick to the "As-Planned" schedule as it already encodes the budget and time restrictions that they need to meet. However, in reality, circumstances seldom permit the "As-Built" schedule to be identical to the "As-Planned" schedule. Projects get derailed from the "As-Planned" implementation because of violations in resource and precedence constraints that define the constraint satisfaction problem in the preconstruction phase.

The next point of investigation is the violation of constraints. In this context, an action is defined as a trigger that creates a set of conditions in the construction environment resulting in constraint violations. The set of conditions triggered is defined as an event. This hints at the causal nature of events in the construction environment. According to Suchman (1967), causality can be applied when one input appears to imply the occurrence of an output, and a causal relationship can be inferred by analyzing how input and output are related or associated. Soibelman and Pena-Mora (2000) claim that Suchman's model can be applied to a construction project for intervening in a chain of events in three different ways: by apprehending and preventing the event from preconditions (primary intervention), reduction of postconditions (secondary intervention), and rehabilitation or reduction of future consequences (tertiary intervention). During the construction process, precedence and resource constraints are violated by various events, which occur in the construction environment. They directly result in disturbances or violations in precedence or resource constraints. An expert construction manager can apprehend constraint violations and/or take corrective measures every time there is one. While the number of possible events that can occur in the construction environment is probably infinite, it can be argued that all of them can be expressed as violations of a closed and finite set of constraints.

An example illustrating the causal nature of events may be expressed as: "a day of bad weather may create delays in outdoor activities." In the construction environment the occurrence of the action "bad weather" causes a related event defined by a delay. The result of the delay has cascading effects on the whole activity network further delaying future activities bound by precedence constraints. Similarly, there is a direct causal link between the delay in material delivery and the unavailability of material for specific activities. Unavailability of material translates to a resource constraint violation. The causal links between the effects of events and their occurrences, and the conditions, which lead to their occurrences, are formally described in detail in a later section in this paper.

The authors argue that the construction management domain is governed by a set of precedence and resource constraints the violation of which, during the construction phase, triggers events. Such events, though specific to the construction environment, are causal in nature. This suggests that given a language to formally represent actions, events, and their relation to constraint violations we can reason about information in a temporally dynamic construction environment. A hypothetical agent can use such a representation to autonomously reason about future events and possible repercussions. This takes the issue to the domain of planning.

Planning problems make use of problem structure to generate relevant plans. Unlike search based problem solvers which are dependent on a specific set of successor functions to affect the environment, planners have a greater degree of autonomy and can create plans which are sensitive to context specific information. To describe the structure, which defines the context, a state description of the environment is not enough. State descriptions are discrete and do not allow the expression of multiple relationships changing simultaneously across time. A formal language is necessary to describe such changes in the environment. The job of an agent attempting to implement a construction project is to dynamically update existing schedules by satisfying constraints, while being sympathetic to the context in the environment, participant interactions, and possible future repercussions. Meanwhile, the participant is in charge of managing the implementation of the project by looking out for constraint violations, and taking corrective measures to make sure that the project is completed despite the constraint violations. The semantics developed in the next section provide a language, which can be used by the agent to represent and reason about construction domain events within a general purpose situational simulation environment.

Formalization of Interval Temporal Logic Model

The first part of this section explores the representation of time using intervals bounded by time points. The basic axioms of interval temporal logic have been borrowed from the work done by Allen and Ferguson (1994). We have added some more axioms to suit representation of activities and events particular to the construction domain. The second part of this section formally defines the situational simulation environment in terms of variables and predicates in a way suitable for representation of actions and events. Finally, the section concludes by formally explaining why a time interval based representation is more suitable than a time point based representation.

Notion of Time

Time has no beginning or end. Time is an infinite interval and any simulation is effectively some finite subset of this infinite interval. All assertions can be expressed in terms of time. In a temporally dynamic world, the validity of an assertion is defined by intervals of time in which it holds true. A universal truth holds over the infinite interval of time, while an assertion about a spell of low productivity will only hold over a finite interval of time. This allows the description of aspects of the environment within a structure of time. A premature falsification of the assertion is a shortening of the interval, while extending the associated time interval can extend the validity of an assertion. A truth assertion about the availability or requirement of a resource over an interval of time can be used to represent resource constraints. Ending and starting time points define time intervals. In fact, all intervals are sets of time points bounded by starting and ending time points. This structured notion of time allows us to appropriately represent precedence constraints.

PROOF COPY [CP/2003/022472] 007501QCP

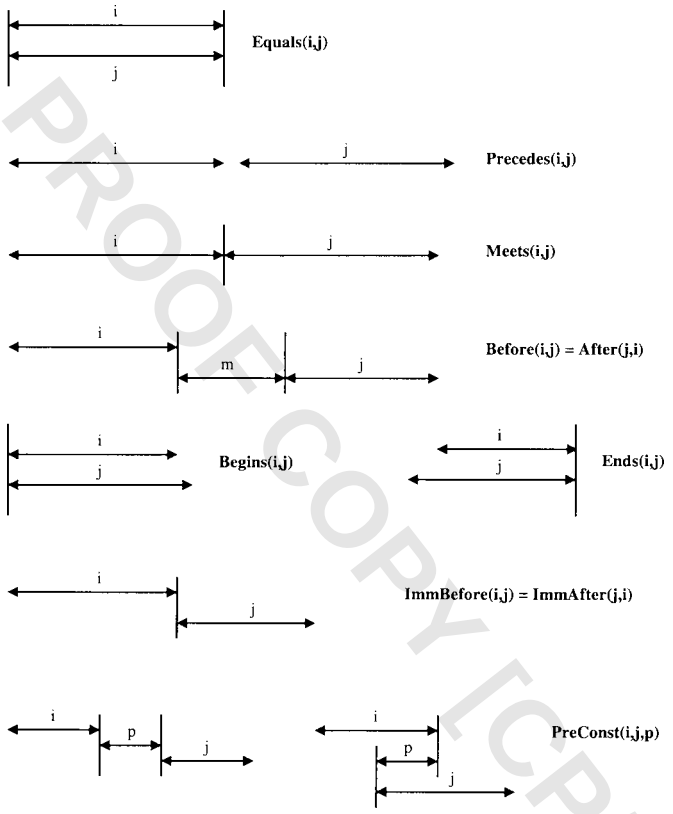

Fig. 2. Axioms defined on intervals

Time Points and Time Intervals

Time points are always represented by positive integers and represent instants in the continuum of time. In the situational simulation environment, a time point is defined as the smallest discrete interval of time within the scope of the simulation and is referred to as the discrete granularity "**p**."

A series of consecutive time points make up a time interval. Every time interval is associated with an ordered pair of integers. The integers define the start and end time points of the time interval. Hence, a time interval i, which stretches from the third day to the fifth day of the simulation, will map onto the ordered pair {3,5} where the discrete granularity of the simulation is 1 day. When a time interval represents an activity, the time points represent the early start and early finish points for the activity

$$\forall i\ .\ \ \exists J,K\ .\ J<K \tag{1}$$

The interval duration is given by $K-J+1$

$$i{:}\{J,K\};\ i\ .\ start=J,\ i\ .\ end=K \tag{2}$$

The convention followed in this paper is that all time points are represented in the upper case, while all time intervals are represented in the lower case.

This model uses an interval representation of time as proposed by Allen and Ferguson (1994). On the basis of the axiomatic relations *Meets*, *Before*, and *After* defined on intervals by Allen and Ferguson, precedence relations between intervals have been axiomatized to aid representation in the situational simulation domain. A diagrammatic representation of the axiomatized time intervals can be seen in Fig. 2. The precedence relations developed include consequence, coincidence, precedence and concurrence. A brief review of the relations *Meets( )*, *Before( )*, and *After( )* follows.

Meets Two time intervals i and j are said to meet if and only if i precedes j, and yet there is no time between i and j, and i and j do not overlap. In terms of discrete time points this can be axiomatized as

$$\forall i,j\ .\ \exists I,J,K,L\ .\ (i=\{I,J\}) \wedge (j=\{K,L\}) \wedge (J=K-1) \Rightarrow Meets(i,j)\equiv i{:}j \tag{3}$$

It can be proved from this axiom that a time interval cannot meet itself. This rules out possibilities of circular models of time.

Before A truth value for $Before(i,j)$, implies that interval i starts before interval j and can be expressed as

$$Before(i,j)\equiv\ \exists m\ .\ \ Meets(i,m)\wedge Meets(m,j)\equiv i<j \tag{4}$$

After The inverse of Before is After, which is defined as

$$After(i,j)\equiv\ \exists m\ .\ Meets(j,m)\wedge Meets(m,i)\equiv i>j \tag{5}$$

Precedence constraints between time intervals representing activities, actions and events are based on the definitions of the following axioms.

Consequence Axiom 1: *An interval i is said to be immediately before another interval j if i precedes j and meets it. Inversely, the interval j is said to be immediately after the interval i*

$$\forall i,j\ .\ \ ImmBefore(i,j)\Rightarrow (i<j)\wedge(i{:}j)\equiv ImmAfter(j,i) \Rightarrow (j>i)\wedge(j{:}i) \tag{6}$$

Coincidence Axiom 2: *Intervals i and j are said to have a coincident point of beginning if there exists a time interval t which comes immediately before both i and j. Similarly, if there is a time interval which comes immediately after two time intervals i and j then they are said to have a coincident point of ending*

$$\forall i,j\ .\ \ Begins(i,j)\Rightarrow\ \exists t\ .\ \ ImmBefore(t,i)\wedge ImmBefore(t,j) \tag{7}$$

$$\forall i,j\ Ends(i,j)\Rightarrow\ \exists t\ .\ \ ImmAfter(t,i)\wedge ImmAfter(t,j) \tag{8}$$

Precedence Axiom 3: *Two time intervals i and j will always have a finish to start precedence relationship defined by the time interval p.*

Combining Eqs. (6)–(8) we get

$$\forall i,j,p\ .\ PrecConst(i,j,p)\Rightarrow(ImmBefore(i,p)\wedge ImmAfter(j,p)) \vee(Begins(j,p)\wedge Ends(i,p))$$

$$\forall i,j,p\ .\ \ PrecConst(i,j,0)\Rightarrow i<{:}j \tag{9}$$

Concurrence Axiom 4: *Given that p_{now} is the discrete granularity which represents the current time in the simulation, all time intervals which span across it are said to be concurrent*

$$\forall t\ ConCurrent(t)\Rightarrow p_{now}\subset t \tag{10}$$

Definitions

Time–Activity Plane

An activity is an emulation of a real life construction operation and is represented by an interval, which has the same length as its duration. Activities take time from start to completion. Activity intervals are dynamic in nature, as activity durations may change

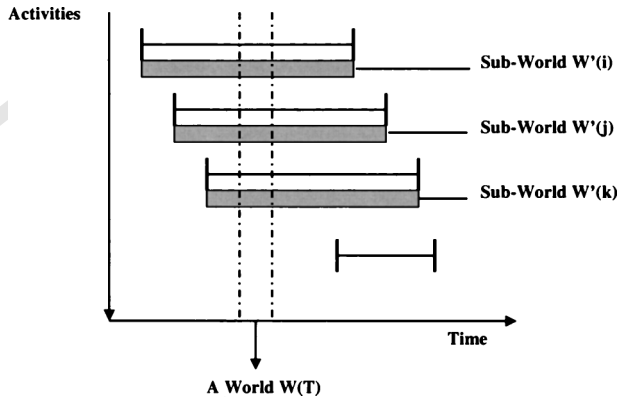

Fig. 3. Worlds and subworlds in activity–time plane

during the construction process. A construction project being simulated can be divided into an integral number of discrete time points of length equal to the discrete granularity p of the simulation. The total number of time points multiplied by the length of a time point gives the total time of the project.

The time–activity plane (Fig. 3) is a discrete plane defined by the positive time axis and the positive activity axis. The time axis increases in the positive direction and represents discrete time points. Each activity is represented by a discrete unit on the activity axis, as an interval in time spanning a set of time points bounded by the starting and ending time points on the time–activity plane. Each activity interval is a set of n time points of length p. The activity–time element is thus the unique ordered pair (i,T) on the time–activity plane representing the activity i at the time point T. The activity–time element is the basic unit of all information representation in the simulation. The simulation proceeds from time point to time point. Each element (i,T) has associated with it minimum material, labor, and equipment requirements for the activity i at the time point T. An activity i is constrained not to start at the time point T if the minimum requirements associated with the element (i,T) are not fulfilled.

Environment

The environment sets the scene for the situational simulation. It is the participant's perception of the simulated construction project. It is interactive, temporally dynamic and virtual in nature. The environment emulates activities, events and processes pertaining to construction projects. It is characterized by a set of variables, each of which describes a unique aspect of the environment.

Variable

A Variable is a symbolic representation of an aspect of the simulation environment. There are two types of variables in the environment. The first type represents aspects that are defined over discrete domains. Such variables are used to logically reason about the environment. The second type of variables is defined over continuous domains and is used to model the dynamical behavior of the system.

The discrete variables can only take up values from a discrete finite domain $S=[s_1,s_2,s_3,\ldots]$, which is the set of all possible attributes of the aspect described by the variable. Every aspect in the environment is completely defined by the set of attributes S. Hence, every variable maps on to a unique domain. A set of variables completely characterizes the environment. Precedence and resource constraints are relationships between the variables and determine the values taken up by the variables. The environment (E) being a composition of the entities is expressed as a set of the variables defining its entities

$$E=[v_1,v_2,v_3,\ldots,v_n] \tag{11}$$

where domain of

$$v_i=S(v_i)=[s_1,s_2,s_3,\ldots,s_k] \tag{12}$$

In Eqs. (11) and (12), the symbol v_i=a variable which describes the aspect v_i in the environment. There is a closure on the set of discrete variables in the environment. Hence, as the simulation proceeds, the variables may take up different attributes from the domain, but new variables may not be added to the set.

A combination of n such variables completely defines the simulation environment, as expressed in Eq. (12). As the environment changes, variables take up different values from their domains to reflect the change. This information is expressed through Boolean predicates, which state the truth regarding time intervals over which variables hold particular values. The truth that the entity represented by the variable v_i has the attribute value s_1 over the time interval t is represented by the predicate

$$v_i(s_1,t) \tag{13}$$

For example, the aspect Weather may be represented by the variable weather, which could take values from the domain [sunny, rainy, snowy]. The predicate weather(sunny,t) signifies that the weather in the environment will hold sunny over the time interval t. Reasoning in the environment uses conditions, which are conjunctive clauses (a conjunctive clause is a first-order logically ANDed expression of simple logical predicates). Hence, the condition representing snowy weather and null productivity over the interval t is represented by the sentence

$$weather(snowy,t)\wedge prod(null,t) \tag{14}$$

Variables are homogeneous over the time intervals in which they hold. Unless otherwise altered, it may be assumed in the above example that the weather will hold snowy for the entire interval t.

The set of variables, which are defined over continuous domains, encode information about the quantities (number of units) of resources that activity intervals use/require. These variables can also be expressed as predicates that hold representative values over intervals of time. They are not used for logical reasoning, but as parameters, which pass numerical information to the mathematical model. These variables are used to calculate derived variables such as remaining duration, direct costs and production rate.

Variable Classification

Entities in the environment, as represented by variables, are often specific to the context of certain activities. This calls for a classification of variables in the environment. All variables that describe aspects of particular activities are *activity specific variables*; all variables that describe aspects relevant to the whole time–activity plane are *global variables*. For example, weather is a global variable since its effects can be felt across activities. However, the availability of an exclusive material or a certain piece of equipment is specific to that particular activity. For example, the unavailability of earth-moving equipment will not affect a concrete pouring operation, even though it will delay a concurrent earth moving operation. All discrete variables are collectively referred to as environmental variables. All of them can also be classified as activity specific variables or global variables.

It is possible for more than one concurrent activity to have instances of the same variable with differing values at the same point of time. For example, the labor efficiency for an activity might be 100%, while that of a concurrent activity may be 80%. Thus, the activity specific variable representing labor efficiency can take up two different values in two different *contexts* at the same time. We define contexts as dynamic time intervals identical to the activity intervals. For all intents and purposes, the contexts are set by activities. Activity specific variables are always in context to the activity, which they define, and at any point of time there can be multiple instances of the same activity specific variable, each in context to a different activity. However, within the same context no variable can have multiple instances, as that would mean an entity having more than one attribute at the same time. This understanding is the basis for the formal definition of *activity specific variables* as variables, which can have multiple instances across contexts at the same time, and *global variables* as variables, which can only have a single instance across all contexts at any point of time.

The variables, which are defined over continuous domains, are all context specific variables. With respect to the mathematical model (Rojas and Mukherjee 2003b), they are independent variables, which are passed as parameters to functions, which compute the derived, or dependent variables (Rojas and Mukherjee 2003c). Continuous variables are used to model the dynamical behavior of the system and are referred to as *independent variables*. A more detailed description of their behavior is beyond the scope of this paper.

World

A world is a snap shot of the environment at a specific time point t, as shown in Fig. 3. The time point is the granularity of the simulation and is usually represented as a day. Progress from day to day in the real world translates to progress from time point to time point in the simulation. The simulation thus moves from one world to the next. Symbolically, the world at the time point T is denoted by $W(T)$ which is given by the set of all variables defining the environment at that time point

$$W(T) = E|_T \tag{15}$$

Subworld

The set of all variables in the environment, which belong to the same context, is defined as a subworld. The subworld is therefore a subset of the world where all the variables are specific to a particular activity that defines the context, as shown in Fig. 3. For the context defined by the activity interval i the subworld at the time point t is the set of m variables which describe entities in the activity and is denoted by

$$W'(i) = \{v_{i1}, v_{i2}, \ldots, v_{\text{im}}\} \in E|_T \tag{16}$$

At any time point t, if the ongoing activity intervals defining the concurrent contexts are i, j, k, then the set of environment variables is given by

$$W(T) = [W'(i) \cup W'(j) \cup W'(k)] + W'(\Phi) \tag{17}$$

and $W'(\Phi)$ is a uniquely defined pseudocontext for global variables

$$W'(\Phi) = Set\_of\_Global\_Variables \tag{18}$$

Actions, Events, and Situations in Situational Simulations

Temporal logic is also used to expressively represent actions, events, and situations. Actions are triggers, which create events and situations. Some examples of outcomes of actions are bad weather, material delivery, reallocation of resources, labor strikes, etc. In the simulation environment actions occur instantaneously in time, at the starting time point of the interval of the event they trigger.

Events reflect the effects of real life episodes on resource and precedence constraints within the construction domain. All events span over time intervals. Each event is associated with three sets of variables: the *Pre-Condition* set, the *Event Condition set*, and the *Consequence* set and is triggered by a unique action. Member variables of the event condition and precondition sets are identical. However, the variables in the two sets must have different attribute values. The change in attribute values is triggered by actions. The event is reflected by the event condition set of variables. Future effects of the event are captured in the consequence set, which is a set of assertions about values of variables in the future. The compound predicates $Pre\_Cond(t)$, $Event\_Cond(t)$, and $Consequence(t)$ are conjunctive clauses of simple predicates which assert attributes of the member variables over the time interval t during which the conditions specified by the precondition, event condition, and consequence sets hold, respectively. They are also homogeneous over the time intervals in which they hold. For example, in the event of a labor strike that lasts for the time interval t, productivity (represented by the variable prod) for all activities is reduced to 0 due to a 0% availability of labor (represented by the variable labor). In this case, the event condition set is $\{labor(null, t), prod(null, t)\}$ across all activity contexts. The precondition set is $\{labor(\langle\rangle 0\%, t'), prod(\langle\rangle 0\%, t')\}$ and the predicate $Meets(t', t)$ is true. This event represents a violation in a resource constraint. The action to create the labor strike event can only be taken if the precondition set is fulfilled in the immediately previous time point. This is expressed as

$$\forall t\ .\ Act\_Labor\_Strike(t\ .\ start) \Rightarrow \exists\, t'\ .\ labor(\langle\rangle 0\%, t') \wedge prod(\langle\rangle 0\%, t) \wedge Event(Labor\_Strike, t) \wedge ImmBefore(t', t) \tag{19}$$

The fact that the precondition set is a necessary condition for an action to occur and that every action triggers an event is used to axiomatize the definition of an action as

$$\forall t\ .\ Action(t\ .\ start) \Rightarrow \exists\, e, t'\ .\ Pre\_Cond(t') \wedge Event(e, t) \wedge ImmBefore(t', t) \tag{20}$$

The converse of the above axiom does not hold, however, if the precondition set holds, then it may be concluded that the action may potentially be taken. This is axiomatized as

Axiom 5: *A specific action can be predicted to potentially occur at $T+1$ if for all contexts concurrent at both T and $T+1$ there exists at least one context i such that $W'(i)$ at t has a subset of variables which satisfy the precondition set of the action.*

Consequences of an event are assertions about the future that are direct outcomes of the event. The consequence set is a set of variables that assert attributes of entities in a future time interval, which is directly affected by the occurrence of the event. For example, in the case of the event Labor_Strike, the productivity of all activities may take a while to recover, and continue to be at 50% for the time interval t'' immediately after the Labor_Strike event is over. The time intervals t and t'' are tied by the precedence constraint predicate. Hence, the labor strike event may be defined in first order predicate logic as:

$$\forall t\ .\ \text{Event}(\text{Labor\_Strike}\ .\ t) \Rightarrow \exists\, t'\ .\ \text{labor}(\text{null},t) \wedge \text{prod}(\text{null},t) \wedge \text{prod}(0.5,t'') \wedge \text{PreConst}(t,t'',0) \tag{21}$$

This allows the generalization of the definition for an event as

$$\forall e,t\ \textit{Event}(e,t) \Rightarrow \exists\, t',p\ \textit{Event\_Cond}(t) \wedge \textit{Consequence}(t') \wedge \textit{PrecConst}(t,t',p) \tag{22}$$

Information about actions and events is stored in a knowledge base and is based on the event and action definitions discussed here. If the precondition and event condition sets of variables for an event belong to the same context, then the event is specific to a particular context and is effectively an activity or context dependent event. However, if the precondition and event condition sets of variables are global variables only, then the event is a global or independent event. For example, because weather is a global variable, an event related to it will be a global event. By definition variables in the event condition and precondition sets have to be either global or context specific. They cannot be mixed. The consequence set may, in both cases, still have variables from across activity contexts. Situations are events that result in immediate constraint violations and demand immediate user intervention to carry on with the simulation. All events may not create immediate constraint violations, and hence may not create situations.

Participant Interactions and Agent Actions

Participants interact with the environment by changing values of the variables. However, participants can interact only with variables within their jurisdiction. By changing the contexts of resource variables, participants can reallocate resources between activities. Global variables are beyond their control (e.g., the participant cannot change the weather). Access is limited to context specific variables, which describe the resource requirements of the activities.

Software agents have greater access to variables than the participants do. The agents can access all global and context specific variables. However, in taking actions that affect the environment, agents are not allowed to change eternal truths about the environment (e.g., an agent cannot change the attribute of an excavation activity from outdoor to indoor). All agent actions are essentially operators, which transform a set of preconditions to a set of event conditions. Because participant interactions are limited within the environment, they cannot directly create events in the environment. However, they can do so indirectly. For example, a reallocation of resources might result in resource constraint violations which, when perceived by the agent, will indirectly create events. Hence participant interactions can only create the *Pre_Cond( )* set, but only agent actions can transform a *Pre_Cond( )* set to a *Post_Cond( )* set.

Interval versus Time Point Representation

The axioms and definitions described so far have been based on a representation, which uses time intervals rather than time points. Time intervals can be represented as a series of time points and the simulation itself traverses from time point to time points. This section uses the constructs of finite state machines to justify the use of intervals over time points.

The situational simulation can be looked upon as a finite state machine (FSM), which is defined as a model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to successive states. It can be described by the tuple

$$\mathbf{M} = \langle \mathbf{S}, \mathbf{I}, \mathbf{R}, \mathbf{L} \rangle \tag{23}$$

where $\mathbf{S}$=finite set of states; $\mathbf{I} \subseteq \mathbf{S}$=set of initial states; $\mathbf{R} \subseteq \mathbf{S} \times \mathbf{S}$=transition relation, specifying the possible transitions from state to state; and $\mathbf{L}$=function that labels states with the atomic propositions from a given language. Such a tuple is called a Kripke structure.

States in the situational simulation environment are expressed using sets of variables. The two sets of variables defined so far are the world set $W(T)$, which characterizes the environment at each time point, and the subworld set $W'(i)$, which characterizes the environment in terms of intervals, over the activity context i. Closure on the set of variables and the attribute domains limits the number of possible worlds and subworlds to a finite number. $\mathbf{W}$ denotes the set of all worlds. If there are n variables, each of which can take up at most m attributes, then the cardinality of the set $\mathbf{W}$ is at most n^m. In reality the number is lower, because there will be many worlds which are mathematically accountable but absurd in reality.

The set of all possible subworlds specific to the activity context i can be expresses as $\mathbf{W}_i$. This set does not completely define the activity environment, because it leaves out the global variables. However, if we augment it with the set of global variables, then we get a set of states that completely defines the context. This is denoted by $\mathbf{W}_i^+$

$$W^+(i) = W'(i) + W(\mathbf{\Phi});\ W'(i) \in \mathbf{W}_i \tag{24}$$

$$\mathbf{W}_i^+ = \{W^+(i)\} \tag{25}$$

$W^+(i)$=augmented subworld; $\mathbf{W}$ and $\mathbf{W}_i^+$ are equivalent to $\mathbf{S}$ in the Kripke structure; and $W(T)$ $(W(T) \subseteq \mathbf{W})$ corresponds to an initial state defined on a time point. Similarly, $W^+(i)$=initial state defined on the context interval i.

An action in the environment creates transitions in state. Differences between attributes of variables in the precondition and event condition sets of an event triggered by an action indicate a transition in state. The critical question is, do the actions create changes in states defined in $\mathbf{W}$ or in $\mathbf{W}_i^+$?

Actions creating dependent events operate on variables specific to the context specific subworld. Actions creating independent events operate on the set $W(\mathbf{\Phi})$. Since the set of variables in

the subworlds have been augmented with the global variables, independent events are essentially multiple instances of the same action across all contexts. Hence, actions creating context dependent and independent events can create state transitions in $\mathbf{W}_i^+$. This representation allows multiple state changes to occur in the environment, each in a different context, at any point of time. In other words, an interval representation allows simultaneous context dependent events across different activity contexts without breaking the semantic structure.

It is very difficult to define unique state change actions as operators on $\mathbf{W}$, without breaking the semantic structure. Only actions triggering independent events can be expressed as state transitions. Simultaneous context specific dependent events cannot be expressed within $\mathbf{W}$ using the defined action-event semantics. Hence, state transitions defined by actions are best suited for the set of states $\mathbf{W}_i^+$ defined on Subworlds that are based on an interval representation of time.

The Kripke structure that can be defined for the context i in the simulation environment is

$$\mathbf{M} = \langle \mathbf{W}_i^+, \mathbf{I}, \mathbf{R}, \mathbf{L} \rangle \tag{26}$$

where

$$\mathbf{I} = \mathbf{W}^+(i)$$

$$\mathbf{R} \subseteq \mathbf{W}_i^+ \times \mathbf{W}_i^+ \tag{27}$$

$$\mathbf{L} \in Set\_of\_all\_Events \tag{28}$$

In effect we have a FSM for each context allowing simultaneous activities and events; actions serving the purpose of state transition operators and events providing a language to express changes in state. This is the rational behind using an interval representation of time.

Tambe et al. (1995) argue that FSM languages are too restrictive to represent human like intelligence. Similarly, time-point representation based FSM languages are inadequate for reasoning about the real life nature of the parallel events in the construction domain. Even though such an approach was used for developing situational simulations in the air-combat domain (Tambe et al. 1995), parallelism and simulation of multiple fighter planes was achieved through DIS technology by running multiple copies of pilot agents participating in battlefield simulations (*ModSAF*: Calder et al. 1993). In essence they were running multiple FSMs in parallel.

Agent Reasoning

In this section a reasoning mechanism is introduced. This mechanism can be used by a software agent to perceive present actions and predict the future evolution of the simulation environment based on the definition of the simulation environment and the interval temporal reasoning.

The agent comes into play between every consecutive discrete time point in the simulation (between any two consecutive "days" during the simulation of the project). The agent infers after the time point T and before the time point $(T+1)$. Given the knowledge of the environment in terms of variables at the end of the time point T and a set of assertions about the environment, the agent can identify the events, which were triggered due to user interaction during the time point T and predict the outcomes of such present events in the future. Furthermore, by identifying the existing conditions, the agent can suggest a list of actions to the event generator to appropriately create situations in the environment.

In its inference environment, the agent has complete access to information encoded in terms of sets of variables $W(T)$ and $W(T-1)$. Since subworlds cannot change state during the inference process, the agent's inference environment is static. It is also discrete, since there are a finite number of possible states that a subworld can take. The environment is also nonepisodic, because the agent needs information from $W(T)$ and $W(T-1)$. Finally, from the agent's point of view, the environment is nondeterministic, as it cannot predict all user interactions or event generator decisions in the immediate future.

It may be noted here that for every event, the set of event conditions may be referred to as the postcondition set for the action triggering the event. The precondition set of the action and event are identical. The following assumptions of closure can also be made:

1. Event closure: An occurrence of an event implies that an action occurred. This is expressed as

$$\forall e,t\ .\ Event(e,t) \Rightarrow \exists\ Action(t\ .\ start) \tag{29}$$

2. Attribute closure: This reflects a closure on the attributes and variables and implies that any change in attributes of variables implies that an event has occurred. This is expressed as

$$\forall t,t'\ .\ Pre\_Cond(t') \wedge Post\_Cond(t) \wedge Meets(t',t)$$
$$\Rightarrow \exists\ e\ Event(e,t) \tag{30}$$

On the basis of the definition of an action [Eq. (20)], definition of an event [Eq. (22)], variable unification (variables unify when they both describe the same entity and take up the same attribute value, two sets of variables unify when there is a one-to-one unification between the members of the sets) and the assumptions of closure [Eqs. (29) and (30)], the following theorem can be stated:

Theorem: *A specific action can be inferred to have occurred in between time points* $(T-1)$ *and* T *if for all contexts concurrent at both* $(T-1)$ *and* T *there exists at least one context* i *such that* $W'(i)$ *at* $(T-1)$ *has a subset of variables which unify with the precondition set of the action and* $W'(i)$ *at* T *has a subset of variables which unify with the postcondition set of the same action.*

Given a context i for which there exists sets S and S' which unify with the precondition and postconditions sets of the same action at $(T-1)$ and T, respectively, it is required to prove that there was an $Action(T)$.

Proof: We know that $S \subset W'(i)$ at $(T-1)$ and $S' \subset W'(i)$ at t. By definition, the member variables of S and S' are identical but have different attribute values. Let us assume that S and S' are homogeneous over time intervals m and n. A change in the values of variables occurs at $(T-1)$ and at T. Therefore, $m.end=(T-1)$ and $n.start=T$. Hence by Eq. (3) we can say $Meets(m,n)$.

Now by Eq. (30)

$$S(m) \wedge S'(n) \wedge\ Meets(m,n) \Rightarrow Event(e,n) \tag{31}$$

by Eq. (29)

$$Event(e,n) \Rightarrow Action(n\ .\ start) \tag{32}$$

Hence, there was an $Action(T)$.

Corollary: *For every action taken there is a set of assertions, which predict the future of the simulation environment.*

```
Agent (Assertions, KB)
{Perceive W(T), W(T-1) from Assertions
        { For each context i concurrent in W(T) and W(T-1)
                { if W'(i) at T!= W'(i) at (T-1)
                        { For each event e in KB
                                {Check for unification of Pre_Cond
                                   and Post_Cond set of e with W'(i);
                                Infer actions taken in T;
                                Infer Consequences of actions in (T+1);
                                Add inferences to set of Assertions;
                                Update entire schedule; }
                        }
                }
        }
}
```

Fig. 4. Agent algorithm

This is proved because of the fact that every action implies an event which, in turn, implies a consequence set [Eqs. (20) and (22)]. Prediction by the agent for possible future actions follows Axiom 4.

Validation

A theorem prover was implemented for the stated theorem using the Forward Chaining algorithm (Russell and Norvig 2002). Given the perceptions of the world in terms of $W(T)$ and $W(T-1)$, the agent classifies the variables into subworlds specific to contexts that are concurrent in both T and $T-1$. It then isolates all contexts in which variables have registered changes in attribute values. Then for each of these contexts, it unifies the variables with action and event definitions in the knowledge base. All inferences are added as facts to the Assertion set, thus allowing reasoning in future worlds to be based on perceptions and outcomes of the past. Fig. 4 illustrates the algorithm.

The implementation was done in Sun Java. The knowledge base encodes information about defined events using directed acyclic graphs. The variables and actions represent nodes, while the edges are represented dependencies between the nodes and are labeled with the states of the variable. There are no edges between variable nodes. All edges originate from variable nodes and lead to action nodes, or originate from the action nodes and lead to variable nodes. The labels on the edges define the state of the variable, which shares an edge with an action node. The set of all variables, which have edges *into* an action node, form the precondition set of the event triggered by the action. Similarly the set of all variables, which have edges *out* of an action node comprise the postcondition and event condition sets of the event triggered by the action. Whenever there is a match in the precondition set of variables of an action with $W'(i) \subset W(T)$ (a *match* entails, that the variables should also have the same states as the state labels on the edges into the action node), the event triggered by the action can either be thrown by the agent or predicted for the future. If there is a match in the precondition set of variables of an action with $W'(i) \subset W(T-1)$ and a match of the postcondition set of variables of an action with $W'(i) \subset W(T)$, then the event can be inferred to have occurred at time point T of the simulation for the context i.

The implementation of the software agent was done to specifically test the representation. For a simulation spanning over 20 days or time points, a given set of event definitions and a random event generator, the implementation was able to detect and predict events perfectly. Events arising out of consequences of previous events could also be detected perfectly. That is, whenever there was an event it was detected. Also, whenever the preconditions were fulfilled an event was predicted. Finally, more than one event could be detected and reasoned about on any particular day, even when the events spanned over different periods of time.

Given the theoretical model these results are not surprising. All agent inference and reasoning is done on the basis of assertions about actions and events in a knowledge base of facts. The inference mechanism is sound and complete within definitions of actions and events defined in the knowledge base. Hence, if an action is defined in the knowledge base, then it will always be predicted every time its precondition set is fulfilled. Also, an event defined in the knowledge base will always be inferred if it has occurred. However, if there is a combination of variable attributes, which is not documented in the knowledge base, then they will simply go unnoticed. Hence, an efficient implementation of this agent lies in developing an accurate knowledge base of facts, and creating appropriate closures on participant interaction.

The reasoning introduced in this paper is limited to dealing with simple conjunctive clauses only. Disjunctions are difficult to deal with because they often make the problems computationally intractable. Also, the success of the reasoning mechanism depends upon how accurately the action and event definitions capture the causal nature of events in the real world. Research is being conducted to identify appropriate precondition, event condition, and consequence sets so that the environment can appropriately simulate the reality of events.

Conclusions

Rojas and Mukherjee (2003b) cite the usefulness of situational simulation environments as learning environments in other domains and argue that the same can be used in the construction management domain. In this paper the writers have gone on to establish that the currently used discrete event simulations paradigms are unsuitable for developing a general purpose situational simulation environment for the construction engineering and management domain. Such paradigms are not appropriate for representing and reasoning about multiple parallel events overlapping in time. A survey of construction simulation shows that there is limited work done in the field of general purpose interactive simulations.

The contribution of this paper lies in introducing semantics based on constraint satisfaction and interval temporal reasoning, which allow the representation and reasoning about actions, events and situations within a general purpose framework. The writers have also theoretically validated the representation to be sound and complete. Events are very important in situational simulations as they form the crux of the educational experience that situational simulations are supposed to deliver. Hence, this work is a stepping stone in aiding the development and application of situational simulation environments in construction.

Notation

The following symbols are used in this paper:

$\forall$ = for all;
$\exists$ = there exists;
$\Rightarrow$ = implies;
$\neg$ = negation;
$\wedge$ = logical AND; and

$\vee$ = logical OR.

Further information on First-Order Logic syntax and semantics can be found at Russell and Norvig (2002).

References

AbouRizk, S. (1993). "Stochastic simulations of construction bidding and project management." *Microcomput. Civ. Eng.*, 8(4), 343–353.

Allen, J. F., and Ferguson, G. (1994). "Actions and events in interval temporal logic." *Microcomput. Civ. Eng.*, 4(5), 531–579.

Calder, R. B., Smith, J. E., Courtemanche, A. J., Mar, J. M.F., and Ceranowicz, A. Z. (1993). "ModSAF behavior simulation and control." *Proc., 3rd Conf. on Computer Generated Forces and Behavioral Representation*, Univ. of Central Florida Institute for Simulation and Training, Orlando, Fla., 347–356.

Chinowsly, P., and Vanegas, J. (1996). "Combining practice and theory in construction education curricula." *ASEE Annual Conf. Proc.*

Choo, H.J., and Tommelein, I.D. (1999). "Parade of trades: a computer game for understanding variability and dependence." *Technical Report No. 99-1*, Construction Engineering and Management Program, Civil and Environmental Engineering Dept., University of California, Berkeley, Calif.

Choo, H. J., Tommelein, I. D., Ballard, G., and Zabelle, T. R. (1999). "WorkPlan: Constraint-based database for work package scheduling." *J. Constr. Eng. Manage.*, 125(3), 151–160.

Dudziak, W., and Hendrickson, C. (1988). "Simulation game for contract negotiations." *J. Manage. Eng.*, 4(2), 113–121.

Fruchter, R. (1997). "The A/E/C virtual atelier: experience and future directions." *Proc., 4th Congress of Computing in Civil Engineering*, ASCE, Reston, Va., 395–402.

Gil, N., and Tommelein, I. (2001). "Comparison of simulation modeling techniques that use preemption to capture design uncertainty." *Proc., 2001 Winter Simulation Conf.*

Hajjar, D., and AbouRizk, S. (1999). "Simphony: an environment for building special purpose construction simulation tools." *Proc., 1999 Winter Simulation Conf.*

Halpin, D., and Woodhead, R. (1970). "A computerized construction management game." Dept. of Civil Engineering, Univ. of Illinois, Urbana Champagne, Ill.

Halpin, D., and Woodhead, R. (1973). *CONSTRUCTO—a heuristic game for construction management*, Univ. of Illinois Press, Champaign, Ill.

Hammond, J., Jeong Choo, H., Austin, S., Tommelein, I, and Ballard, G. (2000). "Integrated design planning, scheduling, and control with desplan." *Proc., 8th Annual Conf. of the Int. Group for Lean Construction*, IGLC-6, Brighton, United Kingdom.

Jaafari, A., Manivong, K., and Chaaya, M. (2001). "VIRCON: interactive system for teaching construction management." *J. Constr. Eng. Manage.*, 127(1), 66–75.

Martinez, J. (2001). "EZSTROBE—general purpose simulation system based on activity cycle diagrams." *Proc., 2001 Winter Simulation Conf.*

Martinez, J., and Ioannou, P. (1999). "General-purpose systems for effective construction simulation." *J. Constr. Eng. Manage.*, 125(4), 265–276.

Martinez, J. C. (1996). "STROBOSCOPE: state and resource based simulation of construction processes." PhD dissertation, Univ. of Michigan, Ann Arbor, Mich.

McCabe, B., Ching, K. S., and Savio, R. (2000). "STRATEGY: a construction simulation environment." *Proc., Construction Congress VI*, 115–120.

Ndekugri, I., and Lansley, P. (1992). "Role of simulation in construction management." *Build Res. Inf.*, 20(2), 109–115.

Rojas, E., and Mukherjee, A. (2003a). "Implementing situational simulations through distributed databases." *Proc., Construction Research Congress in Construction—Wind of Change: Integration and Innovation*, ASCE, Reston, Va.

Rojas, E., and Mukherjee, A. (2003b). "Modeling the construction management process to support situational simulations." *J. Comput. Civ. Eng.*, 17(4), 273–280.

Rojas, E., and Mukherjee, A. (2003c). "Visualizing situational simulation information." *Proc., Construction Research Congress in Construction—Wind of Change: Integration and Innovation*, ASCE, Reston, Va.

Russell, S. J., and Norvig, P. (2002). *Artificial intelligence: A modern approach*, 2nd Ed., Prentice-Hall, Upper Saddle River, N.J.

Sawhney, A., Mund, A., and Koczenasz, J. (2001). "Internet-based interactive construction management learning system." *J. Constr. Education*, 6(3), 124–138.

Soibelman, L., and Kim, H. (2002). "Data preparation process for construction kowledge generation through knowledge discovery in databases." *J. Comput. Civ. Eng.*, 16(1), 39–48.

Soibelman, L., and Pena-Mora, F. (2000). "Distributed multi-reasoning mechanism to support conceptual structural design." *J. Struct. Eng.*, 126(6), 733–742.

Sterman, J. D. (1992). "Systems dynamics modeling for project management." Systems Dynamics Group, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Mass.

Suchman, L. (1967). *Causal analysis*, U.S. General Accounting Office, 6–17.

Sucur, M., and Grobler, F. (1996). "Construction planning through multiagent constraint satisfaction." *Proc., 3rd Congress of Computing in Civil Engineering*, Anaheim, Calif.

Talukdar, S. (1998). "Rules for collaborations among cyber-agents" Invited talk: Improving Decision Trees Using Tabu Search, Informs National Meeting, Montreal, Quebec, April.

Tambe, M., Johnson, W., Lewis, J., Randolph, M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K. (1995). "Intelligent agent for interactive simulation environments." *Constr. Law J.* Spring, ■■–■■.

Veshosky, D., and Egbers, J. (1991). "Civil engineering project management game: teaching with simulation." *J. Prof. Issues Eng. Educ. Pract.*, 117(3), 203–213.