Situation based control for cyber-physical environments

Vivek K. Singh
University of California, Irvine
singhv@uci.edu

Ramesh Jain
University of California, Irvine
jain@ics.uci.edu

ABSTRACT

The use of event-based paradigms to handle control problems in cyber-physical environments is of critical research importance. Most current works however provide partial solutions by only considering individual aspects like event detection, temporal calculi or signal based control. There is thus a need to define a new problem of situation based control which supports symbolic reasoning, strong temporal support and explicit inclusion of domain knowledge to undertake intelligent control in dynamic environments. We describe the problem from a traditional control theoretic perspective and show how it can be handled in practice using the semantics of 'Situation Calculus'. The motivations for future research as well as the research challenges have been identified. The use of the proposed approach to support emerging cyber-physical applications is demonstrated through the example of a multimodal tele-presence application involving selection of appropriate sensor and actuator parameters based on exogenous user actions.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, theory

Keywords

Situation based control, cyber-phyical control systems, Situation calculus

1. INTRODUCTION

Recent advances in *cyber-physical systems* - systems in which computing, broadly construed, interacts with the physical world - have led to multiple advancements in the areas of aerospace, automotive, chemical processes, healthcare, manufacturing, entertainment, and consumer appliances. However, most cyber-physical systems of today shy away from dealing with inherent dynamic nature of such environments and either ignore the temporal complexities or address them in an ad hoc manner [6]. In real world however, we cannot expect the same set of conditions to hold forever, and need explicit mechanisms to model world changes and responding to them.

We feel the lack of progress is not due to lack of relevant research interest. Multiple works in computer vision, multimedia and related communities [1, 8] have focused on event detection and identification. They however do not consider what to do next with these events and typically do not use formal temporal logic to represent these events or handle actuator response. Work in areas like event calculus and situation calculus[9, 13] on the other hand, provide valuable formal models for representing events and situations based on their impact on the real world. They typically though, ignore the issues of event detection and or actuator control in generic settings. Lastly, research in discrete event control [4] has provided some very useful tools for handling event based input and output in generic cyber-physical settings. They however focus on signal based systems and do not consider symbolic data or domain semantics.

Solving control theoretic problems for powerful cyber-physical applications though requires an effective combination of the strengths of each of these areas. We need to create general purpose tools which consider event detection, formally study their impact on world models and provide robust tools for deciding control actions.

Hence, we formulate a new problem of generic situation-based control in cyber-physical environments and describe how this problem can be solved using the semantics of situation calculus. We argue that such a control approach should support the following critical requirements:

1. The ability of the controller to reason based on symbols (rather than just signals)
2. Inherent support for temporal reasoning, and
3. Explicit inclusion of domain semantics.

We model the problem of effectively handling events and creating appropriate response actions in cyber-physical (CP) environments as that of 'intelligent control' (as shown in figure 1). We divide the dynamic environment being considered into 2 parts viz. system-controllable (where the actions are totally controlled by the system) and system-uncontrollable (where the actions are exogenous to the sys-

tem e.g. user actions). We model the world in terms of relevant *fluents*[1]. Each user action impacts the fluent values and can potentially cause the system to move away from its equilibrium state. The system then undertakes the control actions to move back towards equilibrium. The control actions thus undertaken can further change the system state, resulting in respective control actions. Such a process continues until the system return to equilibrium state, at which it waits for next user action. The user actions are captured by the system using sensors, and the control actions are undertaken using available actuators.

An important point to note in this discussion is that the control action is determined by the 'state' variable, which elegantly combines the user-action input with the previous state value. This resultant state which is the *necessary and sufficient world descriptor to decide the control output* is defined as *situation* in our system.

The contributions of this paper are to:
1. Introduce and formally define the problem of situation-based control for cyber-phyical systems.
2. Describe the mapping of this control problem into a Situation Calculus based approach for supporting relevant practical applications.

The organization of the remainder of this paper is as follows. Section 2 describes the related work. Section 3 defines the problem of Situation based control and implementation of an intelligent controller. Section 4 discusses the advantages of a situation based approach. Section 5 describes the use of the proposed ideas into a real world application of appropriate sensor and actuator parameter selection in a tele-presence scenario. We wind off with a discussion on future work and conclusions.

2. RELATED WORK

Discrete Event based control systems[4] have made significant progress in using event based control systems for different applications. However, they typically focus on signal-based systems and do not exploit or use the real world semantics. From computer vision and multimedia based systems[1, 8], a large volume of work has been done on event detection and correlation. However, it does not focus on effects of these events on world states or any inference.

Knowledge based systems [17] on the other hand focus on inferring truth values about the world state based on certain user inputs and rules. They however, do not focus on dynamic (time-varying) aspects. Very few works from this area look into dynamic control systems or deal with sensing inputs.

A related problem with a similar name 'Situational Control' was studied in Russia by Pospelov [11]. Based on citations in [16] and [5], this situational control theory was based on semiotic models of the domain developed in linguistics and language psychology. Semiotics as a science of signs explores the syntactic, semantic and pragmatic aspects of signs. Pospelov considered situations as states of the relations between objects referred to, at some point in time. More details of this work are not easily available in English language. However it is clear that their focus was very different from our work.

Multiple advances have been made by the Situation awareness[2] and Situation modeling[5] community in terms of defining situation awareness and its impact on various mission and control tasks[14]. However there focus has not been from a control-theoretic perspective for dealing with cyber-physical systems.

Our methodology and terminologies build largely on the work in the area of situation calculus pioneered by McCarthy[9] and subsequently by Reiter[10, 13]. While this calculus has been successfully employed to logic, robotics[7] and is starting to be used in databases, it has not been applied to multi-modal or cyber-physical systems.

Rather than focusing on any particular application area, we build this work onto the broad domain of cyber-physical systems which have wide-spread applicability and proven impact in multiple areas like aerospace, automotive, chemical processes, healthcare, manufacturing, entertainment, and consumer appliances. In fact any phyical environment which contains computing-enabled devices can be considered as a cyber-physical environment [6].

3. DEFINITION AND FORMALISMS

3.1 Preliminaries: Situation Calculus

The Situation Calculus is a logic formalism designed for representing and reasoning about dynamical domains. It builds upon traditional predicate, 1st and 2nd order calculus, but is different because it allows for truth values to change over time. The main use of Situation Calculus (on similar lines to traditional calculus), is in using a set of truth values (facts) about a closed world and some reasoning mechanisms (predicates or functions), to infer new truth values which have not explicitly been provided earlier. $L_{sitCalc}$ is a second order language with equality. It has 4 disjoint components.

1. Actions (A) for actions i.e. those which change the 'state' of the world,
2. Situations (S) for 'history of events' [2]
3. Objects (O) as the default sort for everything else.
4. The most important component of Situation Calculus is the fluent (F) sort, which defines truth statements which are dependent on the situation. Fluents can be relational (typically give True/False answers) or functional (return any value as computed).

Thus the language for event calculus can be defined as:

$$\Sigma = \{A, S, O, F\} \quad (1)$$

[1]Fluents are first order predicates having an argument that depends on time. For example the predicate *On(box,table)* can not be always true (or false). We must *reify* the temporal aspect of this predicate as a variable to make it more useful. e.g. using situation 's' to get *On(box,table, **s**)*

[2]This is the definition as per the Reiter formulation. While conceptually we prefer to think of situations as 'snap-shots of the world at a given instant' based on McCarthy[9], the mathematical formulations discussed here are conveniently built upon the formalisms of Reiter formulation[10, 13]. What we define as 'situation' can be thought of as a 'state' variable in the Reiter formulation.

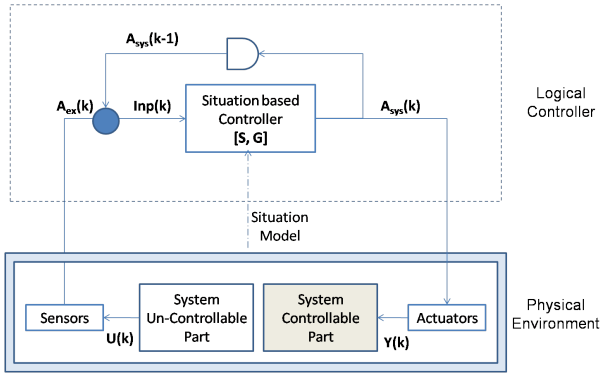


Figure 1: Overall structure for CPE control systems

Within Σ, we can formulate action theories that describe how the world changes as a result of the available actions. We focus on a variant of the basic action theories defined in [13]. An action theory D has the following form:

$$D = D_{fnd} \cup D_{una} \cup \epsilon \cup D_{ap} \cup D_{ss} \cup D_0 \quad (2)$$

1. D_{fnd} consists of the axioms for equality and a set of domain independent foundational axioms which formally define legal situations including $(do(a,s) = do(a',s')) \subset (a = a' \wedge s = s')$
2. D_{una} is the set of unique-names axioms for actions. $A(x) = A'(y) \wedge A(x) = A(y) \supset x = y$.
3. ϵ is a set of unique-names axioms for constants along with a domain closure axiom for sort object.
4. D_{ap} is a set of action precondition axioms, one per action symbol A, of the form $Poss(A(y), s) \equiv \Pi A(y, s)$.
5. D_{ss} is a set of successor state axioms (SSAs), one for each fluent symbol F, which characterizes all the ways the value of a particular fluent can be changed. $Poss(a,s) \rightarrow [F(\overline{x}, do(a,s)) \leftrightarrow \gamma_F^+(\overline{x}, a, s) \vee (F(\overline{x}, s) \wedge \neg\gamma_F^-(\overline{x}, a, s))]$
6. D_0 is a set of axioms describing the initial situation S_0.

The central idea of situation calculus is to define preconditions D_{ap} for each event to happen, and define the impact of these events on the world fluents D_{ss}, once they occur. The primitive operator in situation calculus is $do(action, situation)$, which links up actions and situations. Thus:

$$A \times S \rightarrow S \quad (3)$$

The simple *do* operator can be iteratively employed together with the various axioms, to undertake more sophisticated operations like *progression* and *regression*, which can be used for *planning* and *projection*. Note that D_{ss} also provides a way for handling the frame-problem[3]. A fluent F is true in a situation resulting from applying event a, to situation s, only if, either event/action a caused it to be true (listed under $\gamma_F^+(\overline{x}, a, s)$), or it was already true in situation s, and action a, did not cancel it (listed under $\gamma_F^-(\overline{x}, a, s)$).

3.2 Situation based control

The basic representation for event based control of a CP environment is shown in figure 1. The exogenous actions (which can not be controlled by the system) undertaken by the users at time instance k are shown as $U(k)$. These actions are captured by the sensors and represented as $A_{ex}(k)$.

$$A_{ex}(k) = f_1(U(k)) \quad (4)$$

The state of the dynamic environment at cycle k, can be affected by either the user-driven actions $A_{ex}(k)$ or by system's own control actions as computed during the previous cycle $(A_{sys}(k-1))$. The net input going to the controller can thus be defined as:

$$Inp(k) = f_2(A_{ex}(k), A_{sys}(k-1)) \quad (5)$$

The state of the system in cycle k, is defined as:

$$S(k) = f_3(Inp(k), S(k-1)) \quad (6)$$

The control action or the output of the system in a situation based control system is dependent on the state/situation and the Goal G.

$$A_{sys}(k) = f_4(S(k), G) \quad (7)$$

Which, depending on the representational requirements can also be expressed as:

$$A_{sys}(k) = f_4(f_3(Inp(k), S(k-1)), G) \quad (8)$$

or:

$$A_{sys}(k) = f_4(f_3(f_2(A_{ex}(k), A_{sys}(k-1)), S(k-1)), G) \quad (9)$$

and so on.

Finally, the translation of the controller output actions from a decision level to their physical implementation is undertaken via the Actuators:

$$Y(k) = f_5(A_{out}(k)) \quad (10)$$

Assuming reasonable ways for exogenous action/ event detection using sensors and action-performance using actuators, we will restrict our focus here to appropriate conversion from $A_{ex}(k)$ to $A_{sys}(k)$ based on the presented situation $(S(k))$, and its evolution.

For temporal symbolic control systems, the functions f_3 and f_4 cannot be formulated as standard numerical functions, as we want to handle symbolic data. We need an explicit mechanism which allows us to define the state or situation $S(k)$, and create inference mechanisms which would undertake the translations described by function f_4.

3.3 Implementing the Controller

In this section we discuss how the semantics of the functions f_3 (Eq. 6) and f_4 (Eq.7) can be undertaken using a temporal symbolic framework which allows domain specification. The semantics discussed here build upon the RGOLOG (Reactive Golog) [13] which is a variant of Prolog and Golog that allows concurrent processing and reactivity which are not handled adequately in standard Golog[7].

Let us start by considering the Eq. 6 which needs mechanisms for translation from S(k-1) to S(k), based on input actions Inp(k). Given, that the basic axioms D_{ap} and D_{ss} (pre-condition and post-condition axioms), are well defined, this translation is quite intuitively handled by situation calculus through it's primitive operation $Do(A, S) \rightarrow$

[3]Handling the *non-effects* of actions. For a detailed review of the frame problem and how D_{ss} handles it, see [12]

S'. Hence:

$$S(k) = Do(Inp(k), S(k-1)) \tag{11}$$

Note that this formulation also handles elegantly a sequence of actions, which can simply be executed one after the other.

Now we consider the semantics for Eq. 7, and look at the definition of G i.e. the goal or the equilibrium state which the system is trying to achieve. Let us consider a set of *condition-action* rules with possible conditions which can move the system away from the equilibrium. The *conditions* listed become true in different situations($S(k)$) based on the inputs Inp(k), acting upon $S(k-1)$ as shown in Eq. 11. The input actions can be both exogenous or system controlled. The control actions[4] required to bring back the system to the equilibrium state are $A_{out}(k)$. In general, there can be up to n condition-action pairs which can be represented as:

$$\begin{aligned} \phi_1(X_1) &\rightarrow \alpha_1(X_1), \\ &\vdots \\ \phi_1(X_n) &\rightarrow \alpha_1(X_n) \end{aligned} \tag{12}$$

where X_i contains all the free variables present in the definition. The goal state can hence be defined (in CNF form) as one where none of the condition-action pairs gets violated i.e.

$$S_{Goal} \models (\neg\phi_1(X_1) \vee \alpha_1(X_1)) \wedge \cdots \wedge (\neg\phi_n(X_n) \vee \alpha_n(X_n)) \tag{13}$$

As can be noticed, $\alpha_i(X_i)$ contains multiple free variables and hence may require multiple steps or physical actions to attain the goal state. Further, the i^{th} control action, may potentially lead to the j^{th} violation condition, and hence multiple iterations of control action may be required. The problem of finding the appropriate control action can be defined as the *planning* operation of Situation Calculus. Such a process is handled in Situation Calculus literature based on *proving a theorem* [3] that:

$$\exists A_{out}(k) : Do(A_{out}(k), S(k)) \rightarrow S_{Goal} \tag{14}$$

where $A_{out}(k)$ is a sequence of control actions which are undertaken by the system one in each cycle. Note that this equation links the situation $S(k)$ and goal state S_{Goal} with the output action $A_{out}(k)$, just as was required for the realization of f_4(Eq. 7). Thus equations 11 and 14, provide the necessary translations from the control theoretic problems to the Situation Calculus semantics.

3.4 Controller: Discussion

The theorem proving method shown in previous section has been shown to work well in theoretic settings [3]. However, for creating real time concurrent systems, RGOLOG provides the option of using a procedural definition of the control actions to be undertaken[13]. Thus the system will no longer recursively find the 'plan' to reach the goal based on a declarative structure, but rather follow a set of well-defined procedures to move towards goal state. To handle this we can consider the Inference engine of the controller to be two Golog programs running concurrently, one which controls the 'normal' operation of the systems to move towards a Goal state and other which handles high-priority *interrupts*, which change the world state in any iteration. Hence new control action sequence will be undertaken from that point onwards with an aim to bring system once again to the goal state. Note that this mechanism is not guaranteed to be optimal as the current step undertaken might become useless (or even detrimental) after the occurrence of a new exogenous action. However this approach performs reasonably well for an *on-line* control action mechanism.

[4]Certain conditions might need external control actions too. But they are 'uncontrollable' in control theoretic terms then. We focus only on 'controllable' systems.

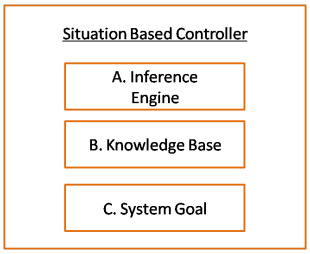


Figure 2: Components of the Situation Based Controller

As might be clear by now, the precise decisions for the next action or next state depend on the rules of the application domain. To include the semantics of the condition-action pairs we add a new class to the action theory D (Eq. 2):

$$D' = D \cup D_{ca} \tag{15}$$

where D_{ca} refers to the condition action rules.

D' consists of both generic and application specific components. The components $D_{fnd}, D_{una}, \epsilon$ are generic and common across all domains and can be considered part of the 'Inference Engine' component of the controller (see fig. 2). The terms D_{ap}, D_{ss}, D_0 and D_{ca} on the other hand need to be defined separately for each domain by a system designer and together form the 'Knowledge Base' component. The procedure for handling the goal state has already been described above.

3.4.1 Situation Modeling

We now describe here a generic procedure which can be used by system designer for creating a situation model, i.e. describing the domain semantics as relevant for the application. This will cover the Knowledge base and the System goal aspects of the controller.

1. Examine the (closed) world to be considered for the application. Identify the subset which will influence the application outcomes in terms of:
   - Objects and their properties to be considered.
   - Actions allowed in the model.
   - Fluents considered in the model. Defined fluents[5], if used should be mapped to primitive fluents.
2. For each action, list down its pre-conditions (D_{ap}) in terms of fluents.

[5]Those which use a combination of primitive fluents

3. To describe the after-effects of each action, use the *successor state axioms* (D_{ss}), clearly listing the actions and conditions for each fluent to be made true or false.

4. Describe the initial situation S_0 in terms of the fluents identified. This covers the D_0 component of D.

5. Define the Goal or equilibrium state for the system in terms of conditions (fluents) and their control actions. This covers the D_{ca} component of D.

Note that identifying the correct set of actions and fluents which will be relevant and sufficient for the application at hand, is application-specific. Hence, while we provide generic tools and guidelines, it is the responsibility of the application designer to create an appropriate situation model. This is analogous to E/R modeling in databases, where the application relevant domain details are modeled by a system designer using some generic tools. Obviously, while E/R models deal with static snapshots of the world, Situation Modeling involves temporal dynamics and has very different end aim.

4. MOTIVATIONS FOR SITUATION BASED CONTROL AND CHALLENGES

4.1 Generic adaptability

Situation modeling approach provides a generic approach to allow different application designers to describe the semantics of their own application. Thus they can plug the necessary details into a developed architecture, and start using inference and intelligent control rather than starting from scratch for each application they develop. A situation model completely specifies all that is necessary and sufficient to characterize a dynamic environment. Hence, it allows for *abstraction* of details across systems. Larger systems can be built which use smaller systems as building blocks.

4.2 Enhanced sensing based on feedback from situation controller

The allocation of sensing resources can be optimized based on world *states* to be detected, rather than lower level data attributes. In fact we can combine top-down and bottom-up approaches, and sensing to be undertaken can be function of the current system state.

Situational approaches can sit on top of the standard sensors which tend to be *noisy*. The situation calculus approach can identify if a detected sequence of exogenous events is *valid* and admissible. Invalid event patterns (e.g. 'wearing socks', *after* 'wearing shoes') can be detected and filtered. Note that this validation now can be at an event-based/semantic level rather than signal level. Alternatively, invalid event patterns can be put into an abnormal classification and used to raise necessary alarms /control actions.

4.3 Reasoning and Analysis

Situation Calculus has an in-built notion of which actions affect which fluents. This can be very useful for goal-based minimal event-set selection. For example, given an initial state S_0 (e.g. User is 'away' i.e. $\neg$ isPresent(P1)) and a goal state S_{goal} which entails that (User is 'busy' i.e. isPresent(P1) $\wedge$ isBusy(P1)); we can compute the minimal number of events which can lead from S_0 to S_{goal}. Conversely, from a corpus of occured events we can find the optimal sub-situation representation i.e. the minimal subset which has directly affected the state transition from S_0 to S_{goal}.

The use of situation calculus allows for creation of tools for 'observability' of black box processes. Through *abductive* reasoning, we can infer what was the initial state S_0, if a sequence of event E, and the final state S_{goal} are available.

4.4 Using Predictive Analysis for control action

In our discussion so far we have assumed no knowledge about exogenous (user) actions. However, if we assume human actors to be acting in a goal-centric manner i.e. using an optimal plan to go from initial State S_0 to desired state S_{goal}, the system can be used to predict the next few user steps. This can allow for faster control action. Tools like Model Predictive Control can be applied to decide the most optimal control action based on current actions and the expected future trajectory.

5. PRACTICAL APPLICATION

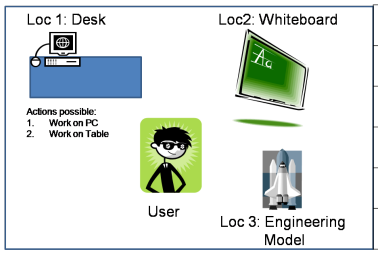
5.1 Background: E2E communication

E2E (Environment-to-Environment) communication is a new paradigm for supporting multi-modal tele-presence. Traditional video-conferencing systems have focused on connecting fixed sensors and actuators (camera and display device) to support communication leading to restricted user movement and interaction. E2E paradigm works on instrumenting the cyber-physical environments with as many sensors and actuators as required and using an event based sentient system which understands user actions to select the best sensors (for outward transmission) and actuators (for rendering incoming information) at each time cycle. This motivates our research on situation based control systems wherein exogenous user actions, need to be handled in the most appropriate manner by a temporal, symbolic, domain-expertise based control system.

Our current implementation involves connecting three environments spread across two different buildings using 7 cameras, 4 microphones, 4 speakers and 5 display devices (1 multi-tiled display, 2 projectors and 2 PC monitors). A typical problem in such a system is that of choosing the best (audio/ video output) data feed to be sent to other users, and identifying the most appropriate settings for actuators and rendering devices to present the incoming audio video information. Readers are pointed to [15] for more details on this paradigm and implementation.

5.2 Automatic camera selection and audio control example

To ground the ideas discussed about modeling situation, user actions and control control action, let us consider a simple problem encountered in our E2E communication systems. The system needs to undertake appropriate camera selection for sharing (video-out) with external environments and adjust the speaker volume (audio-in) control based on



Conditions		Actions	
Move to location	Activity	Selected Cam	Desired Volume
Desk	WorkOn PC	1	1
Desk	WorkOn Table	2	2
Whitebo ard	-	3	3
Model	-	4	4

Figure 3: The CP environment being considered.

user actions in an environment. The relevant system actions include *camera selection* and *setting the audio volume* to high/low values (enumerated between 1 to 4). The exogenous user actions include changing its position (moveToLoc(x)) from being near a 'desk' to a 'whiteboard' to an 'engineering model'. Also, if user location is at his desk he can be working on either the PC or his table. The dynamic cyber-physical world being handled has been summarized in fig. 3. The condition-action pairs have also been shown informally for easy understanding.

Clearly, the exogenous actions require control action from the system. While the camera selection task is assumed to be a one-step process which can be undertaken instantly, setting the volume to the desired level may require multiple control action iterations as we consider the case where the volume can be adjusted by only one unit in each cycle.

Based on the description in section 3.4.1, we now proceed to describe the various axioms which together yield the desired control outcome.

Step 1: Identify the relevant Objects, Actions and Fluents.

Objects
• Desk, Whiteboard and Engineering Model.

Control Actions (System controllable)
• SelectCam(n). To select camera number n to be sent out.
• IncreaseVolume. To increase speaker volume by one unit.
• DecreaseVolume. To decrease speaker volume by one unit.
• SetDesVolume. To set the *desired* speaker volume based on user actions. The actual change of state to this desired level may require multiple steps.
• wait. A *no-op*, created to for next exogenous action to interrupt.

Exogenous Actions
• MoveToLoc(x). User can move to any location between 'desk', 'whiteboard' and 'model'
• StartWorkOnPC. User starts to work on his PC. Also, that he is no longer working on the desk.
• StartWorkOnTable. User starts to work on his Table.

Relational Fluents
• isWorkingOnPC(s)
• isWorkingOnTable(s)

Functional Fluents
• atLoc(s)
• CamSelected(s)
• CurrVol(s)
• DesVol(s)

Step 2: Identify the preconditions for each action

Action Precondition Axioms
• Poss(SelectCam(n), s) ↔ ¬ CamSelected(s) = n
• Poss (IncreaseVolume, s) ↔ (CurrVol,s) ¡ 4
• Poss (DecreaseVolume, s) ↔ (CurrVol,s) ¿ 1
• Poss(wait)= True.
• Poss(MoveToLoc(x), s) ↔ ¬ atLoc(x, s)
• Poss(StartWorkOnPC, s) ↔ atLoc(s)=Desk ∧ ¬ isWorkingOnPC(s)
• Poss(StartWorkOnTable, s) ↔ atLoc(s)=Desk ∧ ¬ isWorkingOnTable(s)

Step 3: Identify the after-effects of each action

Successor State Axioms
• isWorkingOnPC(Do(a,s)) ↔ a=StartWorkOnPC ∨ isWorkingOnPC(s) ∧ ¬ a = StartWorkOnTable
• isWorkingOnTable(Do(a,s)) ↔ a=StartWorkOnTable ∨ isWorkingOnTable(s) ∧ ¬ a=StartWorkOnPC
• atLoc(Do(a,s))=x ↔ a=MoveToLoc(x) ∨ atLoc(s)=x ∧ ¬ a=MoveToLoc(y) ∧ x ≠ y
• CamSelected(Do(a,s))=n ↔ a=SelectCam(n) ∨ CamSelected(s)=n ∧ ¬ a=SelectCam(y) ∧ x ≠ y
• CurrVol(Do(a,s))=n ↔ a= IncreaseVolume ∧ CurrVol(s)=n-1 ∨ a= DecreaseVolume ∧ CurrVol(s)=n+1 ∨ CurrVol(s)=n ∧ a=¬ IncreaseVolume ∧ a=¬ DecreaseVolume

Step 4: Describe the initial situation

Initial Situation
• isWorkingOnPC(S_0)
• ¬ isWorkingOnTable(S_0)
• atLoc(S_0) = Desk
• CamSelected(S_0) = 1
• CurrVol(S_0) = 4
• DesVol(S_0) = 1

Step 5: Identify the goal state using action-condition constraints

Condition-Action (Goal) Constraints
• atLoc(s)= 'Desk' ∧ isWorkingOnPC(s) → Do= Seq ∧ CamSelected(Seq,s) = 1 ∧ DesVol(Seq, s)= 1
• atLoc(s)= 'Desk' ∧ isWorkingOnTable(s) → Do= Seq ∧ CamSelected(Seq,s) = 2 ∧ DesVol(Seq, s)= 2
• atLoc(s)= 'WhiteBoard' → Do= Seq ∧ CamSelected(Seq,s) = 3 ∧ DesVol(Seq, s)= 3
• atLoc(s)= 'Model' → Do= Seq ∧ CamSelected(Seq,s) = 4 ∧ DesVol(Seq, s)= 4

We will adopt a procedural as opposed to declarative/ theorem proving based method for finding the appropriate control actions.
Procedures

```
proc control( ) % The main control loop
wait;
while (CurrVol ≠ DesVol) do
IF CurrVol > DesVol THEN DecreaseVolume
ELSE IncreaseVolume
endWhile
endProc

proc rules( ) % Loop to check for exogenous actions and take counter-action
IF atLoc(s)= 'Desk' ∧ isWorkingOnPC(s) THEN SelectCam(1) ∧ SetDesVolume(1)
IF atLoc(s)= 'Desk' ∧ isWorkingOnTable(s) THEN SelectCam(2) ∧ SetDesVolume(2)
IF atLoc(s)= 'Whiteboard' THEN SelectCam(3) ∧ SetDesVolume(3)
IF atLoc(s)= 'Model' THEN SelectCam(4) ∧ SetDesVolume(4)
endProc
```

The use of procedural control allows for easier adoption of concurrency. For example, it can be noted that in the initial state S_0, the DesVol=4 but the CurVol=1. Thus the output of the controller to move to S_{Goal}, without any exogenous actions is the (3 step) control action sequence:

$DecreaseVolume, DecreaseVolume, DecreaseVolume, S_0$

However if we cause an exogenous action $MoveToLoc('Model')$

at the end of second cycle, this high-priority interrupt changes the desired volume level to 4. Thus the system does not continue working towards the plan created earlier (involving DecreaseVolume). Rather it immediately starts increasing the volume to reach towards its new goal. Hence the obtained action execution order is:

$IncreaseVolume, IncreaseVolume, SelectCam(4)$, ***MoveToLoc('Model')***, $DecreaseVolume, DecreaseVolume, S_0$

This is indeed representative (and appropriate) for a real world online scenario, where we do not want a control system, to keep working towards an old plan, if a new exogenous action has changed the desired outcome.

It might be important at this point to re-look at the control problem being solved and why it could not be solved using other prevalent approaches. Firstly, discrete event based system like [4] do not deal with symbols and hence will not be able to handle symbolic activities like 'isWorkingOnPC' or 'isWorkingOnTable'. Knowledge based control systems, would have also not worked as they do not study transition of states across time and concurrency. The example of control sequence reacting to the exogenous action, could not be handled by a traditional knowledge based control system. Similarly from our initial motivation perspective, we noticed that this system does satisfy the native temporal suppport requirement. Similarly it allows for symbolic inference and lastly the inclusion of domain semantic via axioms D_{ap}, D_{ss}, D_0 and D_{ca}, as shown in the example.

Our progress so far has been in terms of problem definition and defining the appropriate axioms and control procedures. We have successfully tested out the above scenario at a logic level in software (using Prolog). We are currently working towards linking the control with the physical sensors in the environment.

6. CONCLUSIONS

The Primary contribution of this paper lies is in defining the problem of situation based control and describing the process of handling the situation based control by building upon the tenets of Situation Calculus. We have defined situation as a collection of world state descriptors which is sufficient to convert the input (system or user) actions into appropriate control actions. The motivations for future research as well as the research challenges have been identified. We have shown through a practical cyber-application how the proposed ideas can be used in practice. We are currently working towards integrating the logical control structure with the actual physical sensors and actuators.

7. REFERENCES

[1] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45, 1998.

[2] M. Endsley. Situation awareness global assessment technique (sagat). In *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, volume 3, pages 789–795, May 1988.

[3] A. Finzi, F. Pirri, Ray, and R. Reiter. Open world planning in the situation calculus. In *In Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00*, pages 754–760. AAAI Press, 1999.

[4] Y.-C. Ho. Introduction to special issue on dynamics of discrete event systems. *Proceedings of the IEEE*, 77(1):3–6, Jan 1989.

[5] G. Jakobson, J. Buford, and L. Lewis. A framework of cognitive situation modeling and recognition. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–7, Oct. 2006.

[6] E. A. Lee. Cyber physical systems: Design challenges. In *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May 2008. Invited Paper.

[7] H. J. Levesque, R. Reiter, Y. Lesprance, F. Lin, R. B. Scherl, and R. B. Golog: A logic programming language for dynamic domains, 1994.

[8] G. Luo, R. Yan, and P. S. Yu. Real-time new event detection for video streams. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 379–388, 2008.

[9] J. Mccarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

[10] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *J. ACM*, 46(3):325–361, 1999.

[11] D. A. Pospelov. *Situational Control: Theory and Practice(in Russian))*. Nauka, 1986.

[12] R. Reiter. The frame problem in situation the calculus: a simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, pages 359–380, 1991.

[13] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, 2001.

[14] P. M. Salmon, G. H. Walker, D. Ladva, N. A. Stanton, D. P. Jenkins, and L. Rafferty. Measuring situation awareness in command and control: comparison of methods study. In *ECCE '07: Proceedings of the 14th European conference on Cognitive ergonomics*, pages 27–34, 2007.

[15] V. K. Singh, H. Pirsiavash, I. Rishabh, and R. Jain. Towards environment-to-environment (e2e) multimedia communication systems. In *SAME '08: Proceeding of the 1st ACM international workshop on Semantic ambient media experiences*, pages 31–40, 2008.

[16] V. Stefanuk. In search for hidden meaning: Pospelov's work on applied semiotics. *Integration of Knowledge Intensive Multi-Agent Systems, 2003. International Conference on*, pages 575–578, Sept.-4 Oct. 2003.

[17] G. A. Sullivan. A knowledge-based control architecture with interactive reasoning functions. *IEEE Trans. on Knowl. and Data Eng.*, 8(1):179–183, 1996.