Appears in *Modeling and Using Context: Proceedings of the Tenth International and Interdisciplinary Conference* (P. Brézillon, R.M. Turner, & C. Penco, eds.), volume 1025 of *Lecture Notes in Artificial Intelligence,* Springer, 2017. Copyright Springer and the authors, all rights reserved.

# Context and the Virtual Human

Chris Wilson and Roy M. Turner

School of Computing and Information Science
University of Maine
Orono, ME 04469 USA
`chris.wilson@maine.edu, rturner@maine.edu`

**Abstract.** Artificial agents that simulate aspects of human behavior are expected to behave much like a real human in a similar situation would. Human behavior however, is largely context-dependent. Ignoring the effect context can have on an agent's behavior can hinder the quality and plausibility of agent's behavior. In this paper, we present a context-driven approach to modeling plausible aspects of human behavior which focuses on endowing an agent with the ability to recognize and deal effectively with anticipated contextual changes.

**Keywords:** context-mediated behavior, human behavior, simulation

## 1 Introduction

There can be no serious objection to the statement that for humans in general, behavior is context-dependent. For example, a person who hopes to become gainfully employed would not consider text messaging a friend to discuss weekend plans while attending a job interview. Why not? Because most people know (either through social inheritance or from prior experience) that this type of behavior is not appropriate while they are in the context "attending an interview".

For artificial agents, context-appropriate behavior is just as important as it is for humans. After all, an agent that is unable to behave appropriately for the contexts it will operate in is essentially useless.

This is especially true for "virtual humans" (i.e., agents used to simulate aspects of human behavior). Virtual humans can be found in a variety of applications ranging from military and social simulations to commercially available video games. The quality of a simulation or the entertainment value of a video game depends on the plausibility of a virtual human's observable actions. For example, a virtual human that always performs its goals in the same order or an enemy soldier that aims and fires at an opponent with 100% accuracy may not be considered realistic.

Ignoring the effects of context can also cause a virtual human to exhibit unrealistic behaviors. For example, in the Fallout 3 video game, a village populated with friendly townsfolk is attacked by mutants, whom the player must repel. After the attack, surviving villagers still greet the player in the same friendly fashion, even though they are surrounded by the corpses of their neighbors [3].

In this paper, we propose an approach to improving the quality of virtual human behavior using context. Unlike our previous work on context-mediated behavior (CMB) [5], which uses knowledge of known contexts to mediate an agent's behavior while in them, our current approach couples knowledge of known contexts with knowledge of how contexts can change to *drive* an agent's behavior, thus allowing it to avoid situations where context-inappropriate behavior can occur.

Our approach is part of a larger body of work aimed at developing technologies to support people with various forms of cognitive impairment. Part of that work involves simulating behaviors (both "healthy" and erroneous) associated with cognitive impairment to conduct simulated tests of prototype technologies [7]. Consequently, the examples and discussion in this paper are tailored to the simulation of humans performing basic activities of daily living.

### 1.1 Overview of Our Approach

During plan execution, the actions performed by an agent can cause features of its situation to change. These situational changes can in turn lead to changes in the agent's context. As we will discuss below, an agent's context does not change as a whole, and some features of the current context may not be appropriate for the contexts that will arise. These features can be problematic and should be remedied before the agent enters those contexts.

In our approach, problematic features of an agent's current context are used to impose constraints on the contexts the agent can enter as long as those features exist. These constraints are then used to influence the agent's choice of plan for achieving its goals. Our approach incorporates a form of plan projection that helps an agent determine how features of its context will evolve during plan execution. The resulting features are then analyzed to determine if the chosen plan will cause the agent to exhibit behaviors that may be inappropriate for the contexts that will arise, then take steps to modify any problematic features so they are appropriate for accomplishing its goals.

The remainder of this paper is organized as follows. In Section 2, we present a summary of related work. Section 3 provides a discussion of context as it relates to our work and in Section 4, we provide a discussion of our approach to contextual reasoning. Finally, in Section 5 we provide a high-level summary of other applications that can benefit from our approach.

## 2 Related Work

All agents use some form of contextual knowledge, whether implicit or explicit. In rule-based agents, contextual knowledge is encoded in rule antecedents describing when a rule is applicable. For planners, contextual knowledge can be contained in preconditions or filter conditions that are part of the operator schema description.

A problem with contextual knowledge being local to rules or operators, however, is that unless care is taken, it can cause an agent to exhibit the same behavior regardless of the current context. For example, a precondition to performing the step `(exit-house agent)` might be `(is-dressed agent)`. Alone, this rule will cause an agent to get dressed regardless if it is exiting the house to run errands or because the house is on fire. While this type of behavior could be remedied by adding additional constraints that indicate when a particular behavior is appropriate, this approach could lead to an explosion of such qualifiers.

As an alternative, some researchers have proposed the use of "smart objects" and situations to make an agent's behavior more context-appropriate [3]. For example, a smart object might, depending on the current context, inform an agent how to hold or gaze at it. These approaches do not, however, aid an agent in planning to achieve a specific goal and they may not lend themselves to applications where agents operate in uncontrolled or unstructured environments. In fact, since the agent cannot know a priori what the object will tell it, this may actually hinder an agent's ability to plan.

In our earlier work, we argued for the benefits of explicitly representing contexts and contextual knowledge with respect to acquiring, learning, reasoning about, and using such knowledge [5]. In our approach to context-mediated behavior (CMB), known contexts are represented as contextual schemas (c-schemas), which both describe the contexts as well as prescribe how to behave while in them. For the purpose of our current work, however, we are not only interested in using contextual knowledge to mediate aspects of an agent's current behavior, but also aspects of its future behavior. To this end, we treat an agent's context as a dynamic object that continuously evolves as an agent works to achieve its goals.

Similar to other approaches, we do not treat context as a monolithic object that evolves over time, but rather, as a collection of smaller contextual objects that evolve at different rates. For example, Brézillon and colleagues [1] propose three types of contextual knowledge, namely *contextualized, contextual* and *external* knowledge, that can be applied to a problem-solving step. Contextualized knowledge describes any knowledge that is used by an agent during a problem-solving step, whereas contextual knowledge is any knowledge that is not explicitly used during a step, but that constrains it. External knowledge is all other knowledge that has nothing to do with the problem-solving step.

This approach allows an agent's contextual knowledge to evolve during problem solving. For example, while pursuing a goal, a piece of contextualized knowledge might become either contextual or external knowledge. In our approach, the use of c-schemas allows us to explicitly represent both contextualized and contextual knowledge and also provides, in the current work, a way to reason about future contexts in order to help create plans.

## 3 Context

Nearly all AI research on contextual awareness provides its own definition of context. Some definitions are tailored to a specific application [2], whereas others are more general [4]. We too need a definition that is suitable for our current work. Because we are interested in using context to simulate plausible aspects of human behavior, our definition should allow us to represent context as it pertains to real-world situations. Our definition should also allow us to represent contextual changes.

Previously, we defined the term *context* to mean any identifiable configuration of situational (e.g., environmental, goal-related and agent-related) features that has predictive power for an agent's behavior [5]. This definition is very general, which makes it suitable for a variety of applications. We are able to exploit this generality for the purpose of our current work.

First, defining a context in terms of situational features allows us to define contextual changes in terms of changes to the features that make up those contexts. For example, the contexts "at home" and "away from home" can be defined in terms of a single spatial feature such as an agent's location. As a result, an agent's context can change from "at home" to "away from home" when the agent's location changes from `(at-home)` to `(not(at-home))`.

Second, our definition also allows an agent to be in multiple contexts simultaneously. This is highly desirable when simulating aspects of human behavior because real humans can find themselves in multiple contexts at any given moment. While running errands for example, a person might be in the contexts "dressed", "morning", "away from home", "hungry" and "at grocery store".

Being in multiple contexts can have implications for how an agent achieves its goals, because behaviors that are appropriate for one context may not be appropriate for the other contexts. Consider for example, a person who is in the context "hungry" and "at home". In this case, it is perfectly normal for the person to satisfy his or her hunger by preparing and eating a meal. This plan for achieving the goal satisfy hunger may not be appropriate if for example, the latter context were "at friend's house"

### 3.1 Types of Contexts

At a given moment, an agent can be in numerous contexts. These contexts define what we will refer to as an agent's *current* context. As it pursues its goals however, those contexts will change. For example, while pursuing the goal `satisfy hunger`, an agent's context might change from "preparing meal" to "cooking a meal" to "eating a meal". At the same time, its temporal context might change from "morning" to "afternoon". Given this, we do not treat an agent's current context as a monolithic object. Instead, we partition this context into dynamic collections of smaller contexts each having its own implications for an agent's behaviors. Our partitioning scheme is based on the manner in which the individual contexts change.

***Existential contexts***. Some features of an agent's situation are always present but continuously changing. For example, the time of day, day of week and the weather outside are features of any person's situation as long as he or she is alive. We refer to contexts defined in terms of these features as *existential contexts* (ECs). Trivial examples of ECs include the temporal contexts "morning", "afternoon" and "night" as well as environmental contexts such as "sunny weather".

ECs are rigid in that there are generally no actions an agent can take that will affect these contexts. ECs place constraints on which goals an agent will pursue while it is in those contexts. For example, existential contexts can prevent erroneous behaviors such as visiting a store after closing time, cooking breakfast in the middle of the night or placing phone calls at 3 am.

***Problem-solving contexts***. Some features of an agent's situation exist as a result of the goal(s) currently being pursued. For example, a plan for cooking a grilled cheese sandwich can introduce into an agent's situation features such as the ingredients and cooking utensils required to cook the sandwich. The state of objects in the agent's environment can also be affected. For example, cooking the cheese sandwich requires that the agent turn the stove on and place the cheese sandwich in a pan on the stove. These features define what we will refer to as *problem-solving contexts* (PSCs). PSCs, are ephemeral and usually removed from the agent's current context by "cleanup" actions that are either part of the plan or specified by the agent's prior knowledge about the context (e.g., remove pan from stove, turn off stove).

PSCs can have implications for how an agent achieves goals while in those contexts. Unlike ECs, PSCs can be modified to allow for goal accomplishment. As a motivating example, suppose that while cooking a grilled cheese sandwich, an agent is presented with the goal `check the mail`. Suppose further that the agent's mailbox is next to a road and far from the agent's house. Due to the inherent risk of burning the sandwich or, worse, starting a fire, the sandwich being on the stove and the stove being powered on are not appropriate for the context "away from house". Therefore, the agent should avoid committing to the goal `check the mail` as long as those features exist. To allow for the achievement of the goal `check the mail`, the agent could remove the pan from the stove and turn the stove off temporarily.

***Persistent contexts***. Other situational features are longer-lived and can persist across successive changes to an agent's context. These features define what we refer to as *persistent contexts* (PCs). In some cases, a PC is a direct consequence of achieving a goal. For example, by achieving the goal `get dressed`, an agent will be in the context "wearing clothes" or "dressed". PCs can also be a side-effect of achieving a goal. Suppose for example that while buying groceries, an agent spends all of its money. As a result it will be in the context "out of money" until it acquires more.

## 3.2 Representing and Using Contextual Knowledge

Explicitly representing contextual knowledge allows an agent to reason about the context itself, which is critical to assessing the context and behaving appropriately while in it. In our earlier work [5, 6], we used contextual schemas (c-schemas) to explicitly represented an agent's a priori contextual knowledge. C-schemas are knowledge structures that contain descriptive and prescriptive knowledge about a particular context. This can include physical and temporal features of the context as well as information related to other agents and objects that are relevant during problem solving. We continue to use c-schemas in the current work. However, we have augmented them with additional knowledge to support reasoning about contextual changes.

***Contextual constraints.*** In some instances, features of an agent's current context can be considered problematic and may not be appropriate for subsequent contexts an agent can enter into. For example, due to the inherent risk of burning food or starting a fire, contextual features such the presence of food on the stove and the stove being powered on should be used to mediate an agent's behavior while it is in the context "cooking a meal". We are particularly interested in using problematic features to influence which goals (if any) an agent should commit to while it is in a particular context. For example, we would not want our agent checking the mail, running errands or committing to any other goal which distracts its attention from the food on the stove.

One way to achieve this type of behavior is to explicitly define the goals that an agent should avoid while it is in a particular context. Much like encoding contextual knowledge in rule antecedents, this approach can lead to an explosion of qualifiers, as there could be a significant number of goals that an agent should not pursue while in a particular context. We face a similar situation if we were to explicitly define which goals are appropriate for a particular context.

As an alternative, we use problematic features themselves to define which contexts an agent can enter into. This is accomplished by augmenting c-schemas with what we will refer to as *contextual constraints*. Contextual constraints specify contextual features that must be part of any context an agent will enter into while a problematic feature exists. Figure 1 shows a partial example of a c-schema for the context "cooking a meal". From this figure, we see that problematic features related to the cooking process are used to impose constraints on the agent's location (e.g., `(at-home agent)`, `(at-kitchen agent)` and `(at-stove agent)`).

We also note that each contextual constraint carries a preference value. Preference values indicate whether a constraint should be strictly adhered to or if there are situations where it can be ignored. For example, constraints on the agent's location (e.g., `(at-home agent)` and `(at-kitchen agent)`) carry strong preference values, which means they should not be ignored and that any context the agent enters into should intersect the current context on the features of that constraint or "clean-up" actions will be required.

Weak preference values allow for greater flexibility in an agent's behavior by allowing an agent to pursue multiple goals simultaneously when appropriate.

```
Actors:
    self
    *others ;;zero or more intelligent agents
Setting:
    Day: *
    Time: *
    Location: (and (at-home self)
                   (at-kitchen self)
                   (at-counter self))
    Environment: (and (is-on stove)
                      (in-pan food)
                      (on-stove food)
    Agent: (is-hungry self)
Contextual constraints:
    Constraint: (in-house self)
        Constrained-by: (and (is-on stove) (on-stove pan))
        Constraint preference: strong
    Constraint: (at-kitchen self)
        Constrained-by: (and (is-on stove) (on-stove pan))
        Constraint preference: strong
    Constraint: (at-stove self)
        Constrained-by: (and (is-on stove) (on-stove pan))
        Constraint preference: weak
```

**Fig. 1.** C-Schema - *Cooking a Meal*

Suppose for example that while cooking a meal, the kitchen phone rings or that the agent wants to get a drink from the refrigerator. By allowing weak constraints (e.g., `(at-stove agent)`) to be violated, the agent could commit to these goals or any other goal that takes place in the kitchen while it is still in the context "cooking a meal". We might expect a real human to behave similarly when he or she is in a comparable situation.

## 4 Using Context to Drive Behavior

During plan execution, an agent can transition through a variety of contexts. For example, while pursuing the goal `go to work`, an agent's context might transition from "at home" to "driving a car" to "at coffee shop" and finally "at work". Our approach to contextual reasoning is focused on endowing an agent with the ability to identify context transitions related to plan execution as well as any inappropriate behaviors stemming from those transitions. More important, our approach also allows for an agent to modify its plan to allow appropriate goal accomplishment if inappropriate behavior is detected. As opposed to simply replanning however, our approach allows an agent to make context-specific modifications to a plan, thus avoiding the need to completely replan.

### 4.1 Focusing Attention

Much like a real human, a virtual human does not have unlimited problem-solving resources, and must therefore decide which goals it should focus its attention on. In our approach, goals are assigned a numeric priority value, with the agent giving preference to goals with higher priority values. We note that a goal's priority does not indicate whether or not a goal is appropriate for a particular context, it simply informs the agent that the goal should be considered. For example, suppose an agent is presented with the goals `watch television` and `use bathroom` and that the priority assigned to these goals are 1 and 3 respectively. In this case, the agent would consider using the bathroom over watching television.

### 4.2 Predicting Inappropriate Behavior

Before an agent commits to a goal, it must construct or choose a plan for achieving that goal, then determine if the plan allows context-appropriate goal accomplishment.

Contextual constraints provide a simple heuristic for predicting when context-inappropriate behavior can occur. By comparing constrained features of its current context with features of a new goal's c-schema, an agent can determine if transitioning into that context will violate any constraints that are currently imposed.

In our approach, when a constraint violation is detected, the agent constructs a more detailed representation of its future context, because constraint violations alone do not account for contextual changes that might occur during plan execution. For example, it is possible for a plan's steps to remove problematic features and associated constraints from an agent's current context. At the same time, it is possible for a plan to introduce problematic features which may violate constraints imposed by future contexts.

To construct a future representation of an agent's context, we use a form of plan projection to determine how an agent's context will evolve during plan execution. For the purpose of our current work, this involves traversing a plan's steps and applying knowledge of each step's effects to existing knowledge about the agent's current context. This process is used to determine which features of the agent's current context will persist and which will be removed. Persistent features are then merged with features from a pending goal's c-schema to form what we will refer to as a *predicted context.*

Predicted contexts form a partial representation of an agent's future context and can be used to make decisions about the appropriateness of an agent's actions. A predicted context is a partial representation of a future context because a predicted context does not represent contextual changes such as those related to the agent's environment which are beyond the agent's control.

To demonstrate our approach, suppose that is is 12:30 pm on Saturday. It is sunny outside and our agent has committed to the goal `satisfy hunger`. Assume that the agent's plan for achieving this goal consists of the subgoals `prepare cheese sandwich`, `cook cheese sandwich`, `eat sandwich`, `clean up`, and the agent is currently committed to the subgoal `cook sandwich`. A partial example of the agent's PSC is shown in Figure 2.

```
Actors:
    self
    *others ;;zero or more intelligent agents
Setting:
    Day: Saturday
    Time: 12:30PM
    Location: (and (at-home self)
                   (at-kitchen self)
                   (at-counter self))
    Environment: (and (is-on stove)
                      (in-pan sandwich)
                      (on-stove pan))
    Agent: (is-hungry self)
Contextual constraints:
    Constraint - location: (at-home self)
        Constrained-by: (and (is-on stove) (on-stove pan))
        Constraint preference: strong
    Constraint - location: (at-kitchen self)
        Constrained-by: (and (is-on stove) (on-stove pan))
        Constraint preference: strong
    Constraint - location: (at-stove self)
        Constrained-by: (and (is-on stove) (on-stove pan))
        Constraint preference: weak
```

**Fig. 2.** Problem-solving context - *Cook Sandwich*

Suppose that while cooking the cheese sandwich, the agent is presented with the goal `check mail` and that its plan for achieving this goal consists of the high-level steps shown in Figure 3. By comparing the constrained features of the PSC against the corresponding features of the new goal's c-schema (see Figure 4) we see that by entering this context, the agent will violate the strong constraint `(at-home self)` which is imposed on its current location.

```
Step-1: (:action exit :parameters (house agent)
          :preconditions (in-house agent)
          :effect (not (in-house agent)))
Step-2: (:action goto :parameters (mailbox agent)
          :preconditions (not (in-house agent))
          :effect (at-mailbox agent))
Step-3: (:action open :parameters (mailbox agent)
          :preconditions (at-mailbox agent)
          :effect (is-open mailbox))
Step-4: (:action get :parameters (mail agent)
          :preconditions (and (is-open mailbox)
                              (have-mail mailbox))
          :effect (have-mail agent))
```

**Fig. 3.** Sample plan: `check mail`

```
Actors:
    self
    *others ;;zero or more intelligent agents
Setting:
    Day: *
    Time: *
    Location: (and (not (in-house self))
                   (at-road self)
                   (at-mailbox self))
    Environment: *
    Agent: *
Contextual constraints:
    Constraint: (dressed self)
        Constrained-by: (not (in-house self))
        Constraint preference: strong
```

**Fig. 4.** C-Schema - `Check Mail`

Projecting the agent's plan and merging the resulting features with the c-schema for checking the mail (see Figure 4) results in the predicted context shown in Figure 5. Analyzing this predicted context reveals that the agent's future location will violate the constraints imposed on the agent's current location. Therefore, to avoid inappropriate behaviors, the agent should not commit to the goal `check mail` given its current context and the plan for achieving this goal.

```
Actors:
    self
    *others ;;zero or more intelligent agents
Setting:
    Day: Saturday
    Time: 12:30PM
    Location: (and (not (in-house self))
                   (at-road self)
                   (at-mailbox self))
State:
    Environmental:
        (and (is-on stove)
             (on-stove pan)
             (is-sunny weather))
    Agent:
        (and (is-dressed self) (is-hungry self))
Constraints:
    Constraint: (is-dressed agent)
        Constrained-by: (not (in-house agent))
        Constraint preference: strong
```

**Fig. 5.** Predicted context - `check mail`

## 4.3 Choosing a Course of Action

When an agent predicts context-inappropriate behavior, it must decide how to proceed. One possible course of action is to simply defer the pending goal until the agent's context allows for appropriate goal accomplishment (e.g., when problematic features have been removed and contextual constraints have been lifted). While this solution is guaranteed to avoid any inappropriate behaviors, it can also lead to unwanted patterns behavior because the agent might always defer a pending goal (regardless of priority or urgency) if inappropriate behavior is predicted.

***Modifying a plan.*** Before deferring a goal, our agent attempts to modify its current context to allow for appropriate goal accomplishment. Contextual constraints provide insight on how to achieve this. For example, each constraint references two features, namely the problematic and constrained feature. To determine how to modify the chosen plan, an agent can reference its procedural knowledge to identify the actions required to remove a problematic feature from the current context if such an action exists, then augment the chosen plan with those actions.

Continuing our previous example, an agent might determine that clean-up actions (e.g., removing the pan from the stove and turning off the stove) associated with the subgoal `cook sandwich` will remove problematic features from its current context, thus lifting the imposed constraints. Adding these steps to the chosen plan results in the modified plan shown in Figure 6.

```
Step-1: (:action remove :parameters (pan stove)
          :preconditions (on-stove pan)
          :effect (not (on-stove pan)))
Step-2: (:action turn-off :parameters (stove agent)
          :preconditions (is-on stove)
          :effect (not (powered-on stove))
Step-3: (:action exit :parameters (home agent)
          :preconditions (at-home agent)
          :effect (not (at-home agent)))
Step-4: (:action goto :parameters (mailbox agent)
          :preconditions (not (at-home agent))
          :effect (at-mailbox agent))
Step-5: (:action open :parameters (mailbox agent)
          :preconditions (at-mailbox agent)
          :effect (is-open mailbox))
Step-6: (:action get :parameters (mail agent)
          :preconditions (and (is-open mailbox)
                              (have-mail mailbox))
          :effect (have-mail agent))
```

**Fig. 6.** Modified plan: *check mail*

***Choosing a different plan.*** In some instances, it may not be possible to modify a plan to allow appropriate goal accomplishment. For example, suppose that an agent is bedridden yet needs to achieve the goal `check mail`. In this context, the agent is unable to achieve this goal using the plan in Figure 3 and there may not be any way to modify this plan to allow for goal accomplishment in this context. As a result, the agent might choose to satisfy this goal by invoking an alternative goal such as asking another agent to check the mail on its behalf.

***Deferring the pending goal.*** If an agent is unable to formulate an appropriate course of action for achieving a goal, it might be forced to defer the goal until a later time. For example, suppose we are simulating a person who at 10 pm, recalls the goal `schedule doctor's appointment`. In this case, the agent should defer this goal because there is no way to modify a plan to allow for its accomplishment given the agent's current context (e.g., "night time").

## 5 Other Applications

Our approach to context-driven behavior will also be useful for real-world agents, such as robots and autonomous vehicles. In our prior work in the domain of autonomous underwater vehicles (AUVs), we have used c-schemas for understanding the agent's situation and selecting and modulating behavior appropriate for it. Missing from past work was any consideration on the agent's part of intentionally changing its context intelligently or of contexts themselves having primacy over goals in some instances. The current work addresses both of these gaps in CMB.

As an example of the need for a real-world agent to reason about and intentionally change its context, consider the example we have used elsewhere of an AUV entering a mined harbor. In previous work, we emphasized that the AUV would need to recognize the context, which would bring to mind a c-schema that tells it to modulate its behavior: e.g., go slowly, turn off obstacle avoidance sonar that could trigger a mine, etc.

This is reasonable if the AUV suddenly and unexpectedly encounters a minefield. However, in many cases, operating in a minefield is the primary purpose of the AUV (e.g., to map the mined harbor or to detect and destroy the mines). In such cases, the approach outlined in this paper makes much more sense. While en route to the work site, the AUV could reason about its *future* (i.e., predicted) context of being in a mined harbor and realize that *before* entering the harbor, it needs to slow down, turn off its sonar, etc. This allows the vehicle to cleanly enter the new context without risking catastrophe while it reasons about how to behave in it.

As an example of a context being the primary consideration rather than a goal, suppose a scientist wants to deploy an AUV to remain on station in an area, looking for interesting or anomalous events. It is somewhat artificial to give the AUV the nebulous goal of looking for such things: what actions would the goal entail? what predictions could be made about it? Instead, it would be much more reasonable for the agent to commit to (or be told to commit to) being in the *context* of looking for such events. The corresponding c-schema would provide guidance to the AUV about what constitutes "interesting" or "anomalous" event, perhaps in relation to other c-schemas having to do with the particular location, without the need for there to be a vague goal. For example, the c-schema might contain the knowledge that any sensor data that is two standard deviations from what is expected in the (locational) context is anomalous and should be reported.

## 6 Conclusions

In this paper, we have presented a brief discussion of our approach to improving the plausibility of virtual human behavior using a novel form of contextual reasoning. This approach allows an agent to predict inappropriate behavior that can occur during plan execution and take appropriate steps to avoid such behaviors. One advantage to this approach is that it highlights specific steps that may result in inappropriate behaviors. This allows an agent to make context-specific modifications to a plan when it predicts context-inappropriate behavior might occur, thus avoiding the need to completely replan. We intend to continue our work on the approach developed so far, using it to address limitations associated with our earlier approach to CMB. This includes using this approach both for virtual humans and for intelligent real-world agents.

# Bibliography

[1] Patrick Brézillon, J.-Ch. Pomerol, and I Saker. Contextual and contextualized knowledge: An application in subway control. *International Journal of Human-Computer Studies*, 48(3):357–373, 1998.

[2] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[3] Colm Sloan, John D Kelleher, and Brian Mac Namee. Feeling the ambiance: Using smart ambiance to increase contextual awareness in game agents. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 298–300. ACM, 2011.

[4] Brian S Stensrud, Gilbert C Barrett, and Avelino J Gonzalez. Context-based reasoning: A revised specification. In *FLAIRS Conference*, pages 603–610, 2004.

[5] Roy M. Turner. *Adaptive Reasoning For Real-World Problems: A Schema-Based Approach.* Psychology Press, 1994.

[6] Roy M. Turner. Context-mediated behavior. In Patrick Brézillon and Avelino Gonzalez, editors, *Context in Computing: A Cross-Disciplinary Approach for Modeling the Real World Through Contextual Reasoning*, chapter 32, pages 523–540. Springer, 2014.

[7] Chris Wilson and Roy M Turner. Modeling erroneous human behavior: A context-driven approach. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 538–543. Springer, 2015.