Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.068533

ARTICLE

Semantic Knowledge Based Reinforcement Learning Formalism for Smart Learning Environments

Taimoor Hassan[1], Ibrar Hussain[1,*], Hafiz Mahfooz Ul Haque[2], Hamid Turab Mirza[3], Muhammad Nadeem Ali[4] and Byung-Seo Kim[4,*]

[1]Department of Software Engineering, Faculty of Information Technology, University of Lahore, Lahore, 54000, Pakistan
[2]Department of Software Engineering, Faculty of IT & CS, University of Central Punjab, Lahore, 54000, Pakistan
[3]Department of Computer Science, Faculty of Information Science & Technology, COMSATS University Islamabad, Lahore Campus, Lahore, 54000, Pakistan
[4]Department of Software Convergence and Communication Engineering, Sejong Campus, Hongik University, Sejong-si, Seoul, 30016, Republic of Korea

*Corresponding Authors: Ibrar Hussain. Email: ibrar.hussain@cs.uol.edu.pk; Byung-Seo Kim. Email: jsnbs@hongik.ac.kr

Received: 31 May 2025; Accepted: 23 July 2025; Published: 29 August 2025

ABSTRACT: Smart learning environments have been considered as vital sources and essential needs in modern digital education systems. With the rapid proliferation of smart and assistive technologies, smart learning processes have become quite convenient, comfortable, and financially affordable. This shift has led to the emergence of pervasive computing environments, where user's intelligent behavior is supported by smart gadgets; however, it is becoming more challenging due to inconsistent behavior of Artificial intelligence (AI) assistive technologies in terms of networking issues, slow user responses to technologies and limited computational resources. This paper presents a context-aware predictive reasoning based formalism for smart learning environments that facilitates students in managing their academic as well as extra-curricular activities autonomously with limited human intervention. This system consists of a three-tier architecture including the acquisition of the contextualized information from the environment autonomously, modeling the system using Web Ontology Rule Language (OWL 2 RL) and Semantic Web Rule Language (SWRL), and perform reasoning to infer the desired goals whenever and wherever needed. For contextual reasoning, we develop a non-monotonic reasoning based formalism to reason with contextual information using rule-based reasoning. The focus is on distributed problem solving, where context-aware agents exchange information using rule-based reasoning and specify constraints to accomplish desired goals. To formally model-check and simulate the system behavior, we model the case study of a smart learning environment in the UPPAAL model checker and verify the desired properties in the model, such as safety, liveness and robust properties to reflect the overall correctness behavior of the system with achieving the minimum analysis time of 0.002 s and 34,712 KB memory utilization.

KEYWORDS: Context-awareness; reinforcement learning; multi-agent systems; non-monotonic reasoning; formal verification

1 Introduction

The context-aware computing paradigm has gained significant attention in the last two decades, and this trend has been rapidly evolving toward ubiquitous computing with the incorporation of ambient intelligence and assisted living using smart devices/gadgets [1]. Context-aware systems often consist of interconnected heterogeneous devices and/or sensors that are deployed in the systems to autonomously acquire an unprecedented amount of data to/from different sensors/devices and perform computations to assist the user in their daily routine life tasks [2]. These systems are typically installed at homes, offices, and educational institutes to facilitate users with their maximum capacity comfort level. These systems are usually smart and intelligent having lightweight and/or embedded sensors attached to comfortable wearable/connected devices that follow certain protocols for communication and ensure the availability of real-time right information at the right time in the right place. The literature has revealed numerous smart systems and applications in different domains such as smart classrooms, smart healthcare systems, smart transport systems, etc. Among others, smart learning environments have been considered as one of the most emerging research areas due to their significant impact in urban cities and in developed countries where the dependency on such smart devices and systems has been observed at a large scale [3]. Such applications can be very useful for smart gadget-addicted people who are mostly connected to their families and friends using smart systems.

Recent years have witnessed that smart classrooms have been considered a new paradigm and a driving force towards the development of smart learning environments with the incorporation of interconnected network and context-awareness. With the rapid growth of today's digital native generation, students are becoming more addicted to using AI assistive technologies, and their dependencies are exponentially increasing on such smart gadgets/devices which made their lives much more comfortable and relaxed but device dependent. With the advent of digital learning technologies, smart learning environments have gained significant recognition for grabbing students' attention in innovative ways of learning, performing their assessments remotely, and fulfilling their academic activities using smart systems and applications. The literature has revealed a significant number of autonomous systems in different facets of smart learning environments, including academic as well as non-academic activities autonomously. Academic activities are tailored to fulfill the activities of smart classrooms autonomously with minimal human intervention, whereas non-academic activities schedule updates and notify students automatically after certain intervals of time if any change is applied in the system like change in the university transport schedule, cafeteria food menu, lecture timings, etc.

In the literature, a significant effort has been made on the development of smart learning systems and applications [4,5]. In [5], Martin et al. have proposed a framework for designing and managing adaptive mobile learning systems. This system can be used by both teachers and students to carry out one-on-one or group learning tasks. The system's main goal is to suggest the learning activities that are most suited for each user to fulfill their ultimate need. Researchers are putting great effort into improving learning systems and their numerous applications [6–8]. Zeeshan et al. [9] have presented a comprehensive review considering UN sustainable development goals on how AI assistive technologies can become an effective educational toolkit to facilitate educational managers, teachers, and students. They have showcased the cutting-edge technology utilization for the educational systems in different perspectives such as the student attendance management system, their assessment tasks along with feedback systems, and interactive tools for teaching and managing their assessment tasks. This work has the following contributions in this paper: 1) We propose the ontology-driven context-aware predictive reasoning formalism for smart learning environment based on the Markov decision process. 2) This work has two-fold segregation as academic and administrative. The academic system focuses on academic as well as non-academic activities as a whole, whereas the administrative system particularly emphasizes the smart classroom environment. The system consists of a three-tier architecture where non-monotonic reasoning-based context-aware agents acquire the contextualized information from the ontology, perform prediction-based analysis to take appropriate actions, and then adapt the system's behavior accordingly. 3) We model the case study of a smart learning environment using the UPPAAL model checker and verify the correctness properties of the system [10].

The rest of the paper is structured as follows. In Section 2, we briefly discuss the related work on smart learning environments. In Section 3, we model the domain of the smart learning environment in Protégé ontology editor and develop complex rules of the system. Section 4 presents a reinforcement learning based context-aware multi-agent reasoning model. Section 5 presents the formal modeling and verification of the case study and finally, we conclude in Section 6.

2 Related Work

Literature has revealed a renewed research interest in probing different approaches to the development of smart learning environments using the notions of context-awareness with reinforcement learning techniques. Much of this work concentrate on contextual reasoning based predictive decision support mechanism. In context-aware deployment settings [4,6,7], the essential need for developing formal context models [11–13] has been significantly focused by the research community [14]. In [4], authors have developed a context-aware mobile organizer which autonomously sends updates to students about their queries. In this work, a context-aware mobile organizer has been developed that assists students with their daily routine tasks such as bus schedules, cafe locations, room locations, lecture times, and access to the learning management system while they are on university premises. The context-aware mobile organizer autonomously sends updates to corresponding users in case of changing contextual information. In [11], the authors have presented a context-aware smart classroom architecture for the execution of classes in smart classrooms. They have suggested a three-tier architecture; (a) a context-aware smart classroom prototype, (b) a technology integration model, and (c) measures to support smart classroom operations. Using the suggested architecture and Raspberry Pi-based application interfaces, a context-aware energy-saving smart classroom application was developed. The findings show that the architecture can be used to create context-aware smart classrooms on smart campuses. The study emphasizes the importance of the suggested context-aware smart classroom architecture, which comprises critical components for smart classroom design and administration on university campuses.

Paudel et al. [12] describe a technique to provide sustainable and clean energy to a university campus by utilizing the Internet of Things (IoT) and deep learning for action recognition. The system monitors and manages energy usage in real time by integrating IoT sensors with video action recognition in classrooms, resulting in a considerable reduction in power load on the electric grid. The suggested context-aware architecture incorporates a variety of sensors, including temperature, humidity, brightness, and video sensors, to offer a multi-modal approach to energy conservation [13]. The research successfully trains and executes a neural network for human action recognition, allowing for more efficient energy management. The results reveal that the proposed architecture can save 25–30 percent of energy in buildings, which is a satisfactory degree of accuracy. The system's ability to automate energy management tasks has the potential to further minimize human labor and pave the path for broader community applications for overall energy savings and cost-effectiveness.

In [7], Ilhan et al. presented a teacher-student architecture for cooperative decentralized MARL (Multi-agent Reinforcement Learning) in which agents assess their understanding in various states utilizing the Random Network Distillation (RND) technique to launch the advising dynamics without pre-assumed roles [15]. The experimental results in a grid world environment reveal results indicating that this strategy could be effective in MARL.

The study proposed by Liu et al. [14] offers a cyber-physical-social system that monitors students' learning progress in a smart classroom using various sensors. Based on multi-modal sensor data, it employs reinforcement learning techniques to provide personalized learning advice. To detect students' learning states, the suggested system analyzes their heartbeats, quiz scores, blinks, and facial expressions. It suggests efficient learning activities matched to their current situations through reinforcement learning. A Markov decision process is used to model this interactive learning recommendation process. Initial simulation results demonstrate the system's ability to provide smart learning recommendations.

In [6], Torrey et al. present a reinforcement learning approach to education in which one agent teacher counsels another agent student during a sequential task. The teacher has a limited amount of advice "budget". The student and the teacher are both represented as reinforcement learning agents. Experiments in the mountain automobile area illustrate the superiority of the provided strategy over existing heuristics.

In addition, we conducted a comparative analysis of state-of-the-art approaches based on an evaluation of their methodological frameworks, operation potential, and the usefulness of the approaches in real-life smart learning environments. It determines the notable strengths and limitations of each approach, clarifying their performance trade-offs, scalability, adaptability, and the effectiveness of their reasoning. Table 1 provides a summary of the findings and forms the foundation for acknowledging essential research gaps, as well as informing the development of intelligent systems more contextually and semantically enriched. The overview explains the advances as well as the constraints of the existing methodologies. Although the literature under review has had a substantial impact on various approaches and models within the field of smart-learning environments, its application towards semantic-based knowledge modeling context and inferencing mechanisms have not been explored adequately. This shortcoming points to a decisive direction for future development, especially in those systems that rely on a deep understanding of contextual dynamics.

Table 1: Comparative analysis of used approaches

Paper	Used approach	Strengths	Weakness
[4]	Context-aware university mobile organizer	Relevance & practical application, context-awareness, requirement-driven design, structured development	Lack of implementation details, no evaluation metrics, user adoption issue not addressed, no mention of comparative analysis
[6]	RL agents as teachers and teaching algorithms	Novel teaching framework, improves learning efficiency, broad applicability	Limited scope, no multi-agent teaching, lack of real-world testing, state importance not fully explored
[7]	Heuristic-based teacher-student action advising in MARL and RND	Handles heterogeneous agents, innovative use of RND	State importance metric issues, delayed advice influence, limited agent scalability
[12]	Self-controlled smart energy saving using deep learning, transfer learning, and context awareness with IoT	Efficient energy management, IoT-driven automation, multi-modal data fusion	Limited dataset, unverified cost savings

(Continued)

Table 1 (continued)

Paper	Used approach	Strengths	Weakness
[11]	Develops a technology integration model using Raspberry Pi, Python, Flask, PHP, MariaDB, RFID	Scalable architecture, technology integration, practical demonstration	Lacks security mechanisms, incomplete data exchange model
[14]	Constructs a cyber-physical-social system using multi-modal sensors, RL and MDP	Personalized learning, multi-modal data fusion, AI-driven optimization	Limited real-world testing, privacy concerns
[13]	Regression models and genetic programming	Water conservation, automation & remote monitoring, integration of multi-modal data	Limited scope, no yield improvement
[15]	Random Network Distillation (RND) as an exploration method for Atari games with sparse rewards	Effective for local exploration, scalability, handles sparse rewards	Limited global exploration, fails in complex tasks, needs further optimization
[16]	Introduces PyTSC, a MARL-based traffic signal control (TSC) library	Versatile & extensible, optimized performance, real-world impact	Limited generalization, complex integration, scalability challenges
[17]	Mixed vehicle platoon formation method for CAVs and HDVs	Efficient platoon formation, traffic & energy efficiency	Limited collision avoidance, HDV behavior complexity, urban traffic challenges

Among all the techniques studied, ontology-based context modeling promises to be the most successful due to its structured but flexible character, its use of formal semantics, and its capability to provide strong support for high-level reasoning. These attributes enable effective and understandable representation of contexts and can thus improve the adaptivity, scalability, and robustness of the multi-agent intelligent systems that are based on the performance in heterogeneous and dynamic environments. Thus, a great focus on ontology-based applications can significantly increase situation awareness and decision making in the smart learning environment.

3 Context Modeling of Smart Learning Environment

In this work, the term "Smart" refers to autonomous adaptive learning environments that use semantic technologies and rule-based reasoning to enable intelligent self-directed decision making with minimal human intervention. Within such environments, smart devices, ontological modeling, and inference engines dynamically adapt to learners' evolving needs, offering proactive support for both academic and non-academic activities. The literature shows substantial progress in modeling heterogeneous knowledge sources to design context-aware multi-agent systems across various domains such as health care, smart parking, and smart learning systems, which further strengthens the foundation for developing truly intelligent and adaptive learning systems [18,19]. A context model (also known as context modeling) specifies how contextual information is managed in a structured and organized manner. The contextual information is intended to be produced as a formal or semi-formal description. Context modeling has gained significant attention in developing context-aware systems and applications, and researchers opt for different context modeling techniques such as ontology-based modeling, graphical modeling, logic-based modeling, markup scheme modeling, and key-value pair modeling [20]. We choose the ontology-based approach which is considered to be the most optimistic modeling technique due to its efficient and simplistic reasoning power and expressivity. Ontologies have gained much attention in knowledge sharing among different knowledge sources. In an ontology, a domain can be modeled in terms of classes, data, and object properties, and individuals in an OWL/RDF ontology format and represent association among classes using properties. OWL is based on Description Logic (DL), which can be used to conceptualize the domain, and it is a straightforward approach to transform DL axioms into OWL axioms. DL is a fragment of First-Order Logic (FOL) through which logic-based systems can be formalized using semantic networks. DL knowledge base consists of a finite set of terminological and assertional sentences, namely TBOX, ABOX, and RBOX. TBOX (Terminological Box) defines atomic concepts and it can be mapped to the corresponding class in the ontology, ABOX (Assertional Box) represents association among concepts and roles. It is further categorized as concept assertion and role assertion. RBox (Relational Box) shows inter-dependencies among different roles.

In this work, we develop a semantic knowledge model based on non-monotonic reasoning for the smart learning environment. We construct an ontology of the case study of smart learning environments. The core aim is to facilitate students in managing their daily routine academic activities efficiently and effectively and provide them with a safe, secure, and comfortable learning environment while the student is staying on campus. The proposed system autonomously assists students in their class schedules, cancellation/makeup classes, relocating classes, generating alerts of their assessments deadline, getting alerts on library and foods, etc. Users of this system will get a daily work schedule every morning and receive notifications with every change in the situation. We assume the system consists of a set of sensors/embedded sensors that autonomously monitor the scheduled classes as per the semester calendar and notify the class status to the teachers and admin staff. In addition, the system can be equipped with domotic features such as lighting and cooling system. For example; the sensors should have the capability to check the presence of students and teachers in the class and turn on/off lighting and cooling system. The system will automatically be set on the multimedia and sound system for lecture purposes. To model the case study in Protégé ontology editor, we consider eight agents as shown in Table 2. The ontology is divided into two parts: academic level and administrative level. Academic-level ontology captures the features to fulfill all academic-related activities whereas administrative-level ontology is designed in a way that it can provide associative support. We initially model the scenario in the ontology. The first step is to acquire data from the sensors and keep the ontology in an organized pattern. Then based on the existing knowledge base, rules are triggered by context-aware agents using semantic knowledge translator [21], and then translated rules are applied to achieve the desired goals.

Table 2: A set of few execution rules for smart learning environment

Agent 1: Smart Classroom Agent
Initial Facts: $Classroom(C-307), ClassActivity('InProgress), QuizInProgress('Yes)$
$R11: Classroom(?c), hasAudience(?pres,''Sufficient''), ProjectorStatus(?proj,''On''),$ $ClassActivity(?ca) \rightarrow hasClassStatus(?ca,''InProgress'')$ $R12: Classroom(?c), hasAudience(?pres,''Sufficient''), ProjectorStatus(?proj,''On''),$ $\neg ClassActivity(?ca), QuizInProgress(?qa) \rightarrow hasClassStatus(?ca,''InProgress'')$

(Continued)

Table 2 (continued)

$R13: Classroom(?c), hasAudience(?pres,''Sufficient''), ProjectorStatus(?proj,''Off''), \neg\, ClassActivity(?ca), QuizInProgress(?qa) \rightarrow hasClassStatus(?ca,''InProgress'')$ $R14: Classroom(?c), hasAudience(?pres,''Insufficient'') \rightarrow \neg\, hasClassStatus(?ca,''InProgress'')$ $R15: Tell(2,1,hasAudience(?pres,''Sufficient'')) \rightarrow hasAudience(?pres,''Sufficient'')$ $R16: Tell(2,1,hasAudience(?pres,''InSufficient'')) \rightarrow hasAudience(?pres,''Insufficient'')$ $R17: Tell(5,1,ProjectorStatus(?proj,''On'')) \rightarrow ProjectorStatus(?proj,''On'')$ $R18: Tell(5,1,ProjectorStatus(?proj,''Off'')) \rightarrow ProjectorStatus(?proj,''Off'')$
Agent 2: PresenceDetector Agent
Initial Facts: $PresenceDetector('Yes), headCount(6)$
$R21: PresenceDetector(?pres), headCount(?hc), greaterThan(?hc,5) \rightarrow hasAudience(?pres,''Sufficient'')$ $R22: PresenceDetector(?pres), headCount(?hc), lessThan(?hc,6) \rightarrow hasAudience(?pres,''Insufficient'')$ $R23: hasAudience(?pres,''Sufficient'') \rightarrow Tell(2,1,hasAudience(?pres,''Sufficient''))$ $R24: hasAudience(?pres,''Insufficient'') \rightarrow Tell(2,1,hasAudience(?pres,''InSufficient''))$
Agent 3: Temperature Agent
Initial Facts: $Classroom(C-307), Temperature(19)$
$R31: Temperature(?temp), greaterThan(?temp,18), lessThan(?temp,25) \rightarrow hasTemperature(?temp,''Normal'')$ $R32: Temperature(?temp), greaterThan(?temp,25) \rightarrow hasTemperature(?temp,''High'')$ $R33: Temperature(?temp), lessThan(?temp,16) \rightarrow hasTemperature(?temp,''Low'')$ $R34: hasTemperature(?temp,''Low''), Classroom(?c) \rightarrow Tell(3,4,hasTemperature(?temp,''Low''))$ $R35: hasTemperature(?temp,''High''), Classroom(?c) \rightarrow Tell(3,4,hasTemperature(?temp,''High''))$
Agent 4: CoolingSystem Agent
Initial Facts: $Classroom(C-307)$
$R41: Tell(3,4,hasTemperature(?temp,''High'')) \rightarrow hasTemperature(?temp,''High'')$ $R42: Tell(3,4,hasTemperature(?temp,''Low'')) \rightarrow hasTemperature(?temp,''Low'')$ $R43: hasTemperature(?temp,''Low'') \rightarrow TurnCoolingSystem(?c,''Off'')$ $R44: hasTemperature(?temp,''High'') \rightarrow TurnCoolingSystem(?c,''On'')$
Agent 5: Projector Agent
Initial Facts: $Projector('On)$
$R51: Tell(2,5,hasAudience(?pres,''Sufficient'')) \rightarrow hasAudience(?pres,''Sufficient'')$ $R52: Tell(2,5,hasAudience(?pres,''Insufficient'')) \rightarrow hasAudience(?pres,''Insufficient'')$ $R53: hasAudience(?pres,''Insufficient''), Projector(?proj) \rightarrow ProjectorStatus(?proj,''Off'')$ $R54: hasAudience(?pres,''Sufficient''), Projector(?proj) \rightarrow ProjectorStatus(?proj,''On'')$

(Continued)

Table 2 (continued)

$R55: ProjectorStatus(?proj,''On'') \rightarrow Tell(5,1,ProjectorStatus(?proj,''On''))$ $R56: ProjectorStatus(?proj,''Off'') \rightarrow Tell(5,1,ProjectorStatus(?proj,''Off''))$
Agent 6: Camera Agent
Initial Facts: $Camera(cam-101), CamZone(?cam-101,'Courtyard), Location('Courtyard), Image('Courtyard)$
$R61: Camera(?cam), Location(?loc), CamZone(?cam,?loc), Image(?img) \rightarrow CaptureCam(?img,?loc)$ $R62: CaptureCam(?img,?loc) \rightarrow Tell(6,8,CaptureCam(?img,?loc))$
Agent 7: Security Agent
Initial Facts: $SecurityOfficer('Alan), LocateCulprit('Jimmy,'Courtyard)$
$R71: Tell(8,7,SuspeciousActvityDetectedAtLocation(?img,?loc)) \rightarrow SuspeciousActvityDetectedAtLocation(?img,?loc)$ $R72: SecurityOfficer(?officer), SuspeciousActvityDetectedAtLocation(?img,?loc), LocateCulprit(?culp,?loc) \rightarrow CatchCulpritBy(?culp,?Officer)$
Agent 8: Controller Agent
Initial Facts: $Controller(Tim), NoticeSuspeciousActivity(Image,''Yes'')$
$R81: Tell(6,8,CaptureCam(?img,?loc)) \rightarrow CaptureCam(?img,?loc)$ $R82: Controller(?cont), CaptureCam(?img,?loc), NoticeSuspeciousActivity(?img,''Yes'') \rightarrow SuspeciousActvityDetectedAtLocation(?img,?loc)$ $R83: SuspeciousActvityDetectedAtLocation(?img,?loc) \rightarrow Tell(8,7,SuspeciousActvityDetectedAtLocation(?img,?loc))$

Fig. 1 depicts a fragment of the smart learning environment, whereas a class hierarchy of the smart learning environment can be seen in Fig. 2. We have a two set of rules to model the systems. In ontology, we construct straightforward rules in OWL 2 RL (OWL 2 Rule language) because it requires scalable reasoning with very less expressivity and it allows rule-based reasoning capability. In addition, OWL 2 RL accommodates rules both in RDF and OWL 2 ontologies format. We develop complex rules using SWRL (Semantic web rule language). SWRL rules are formed from a set class atoms and property atoms (object and data). We construct SWRL rules for eight agents to represent the working flow of the system, however, we omit the rest of the rules due to space constraints. Fig. 3 shows object properties and data properties and Fig. 4 shows complex rules using OWL 2 RL and SWRL.

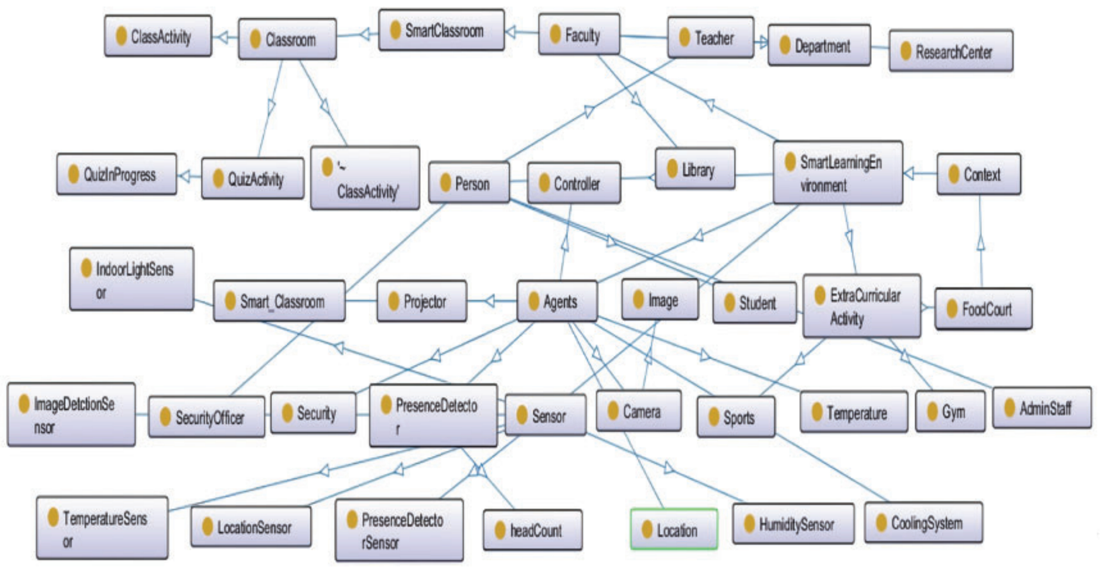
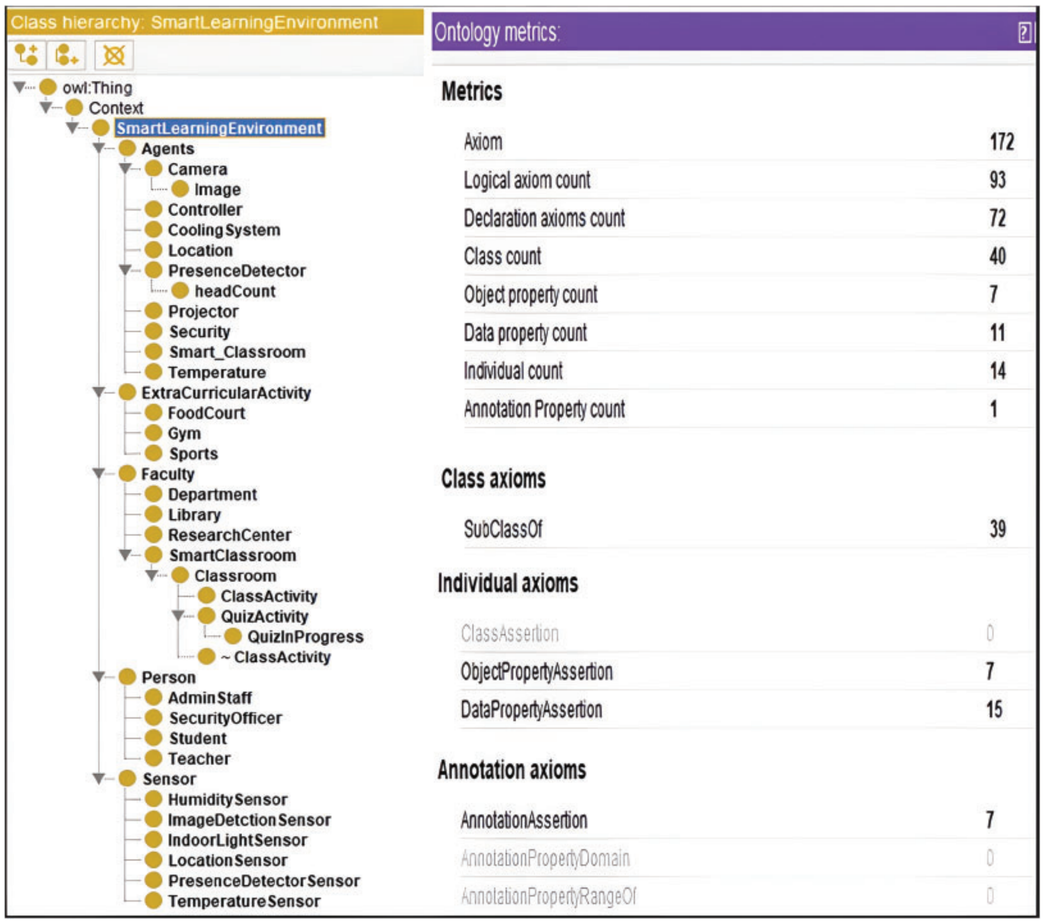


Figure 1: A fragment of smart learning environment ontology


Class hierarchy: SmartLearningEnvironment

- owl:Thing
  - Context
    - SmartLearningEnvironment
      - Agents
        - Camera
          - Image
        - Controller
        - CoolingSystem
        - Location
        - PresenceDetector
          - headCount
        - Projector
        - Security
        - Smart_Classroom
        - Temperature
      - ExtraCurricularActivity
        - FoodCourt
        - Gym
        - Sports
      - Faculty
        - Department
        - Library
        - ResearchCenter
        - SmartClassroom
          - Classroom
            - ClassActivity
            - QuizActivity
              - QuizInProgress
            - ~ ClassActivity
      - Person
        - AdminStaff
        - SecurityOfficer
        - Student
        - Teacher
      - Sensor
        - HumiditySensor
        - ImageDetctionSensor
        - IndoorLightSensor
        - LocationSensor
        - PresenceDetectorSensor
        - TemperatureSensor

Ontology metrics:

Metrics

Axiom	172
Logical axiom count	93
Declaration axioms count	72
Class count	40
Object property count	7
Data property count	11
Individual count	14
Annotation Property count	1

Class axioms

SubClassOf	39

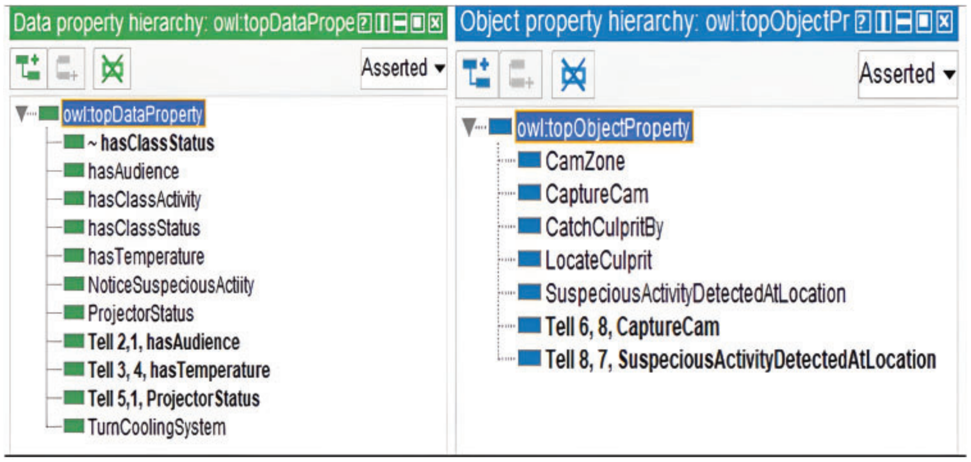
Individual axioms

ClassAssertion	0
ObjectPropertyAssertion	7
DataPropertyAssertion	15

Annotation axioms

AnnotationAssertion	7
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0

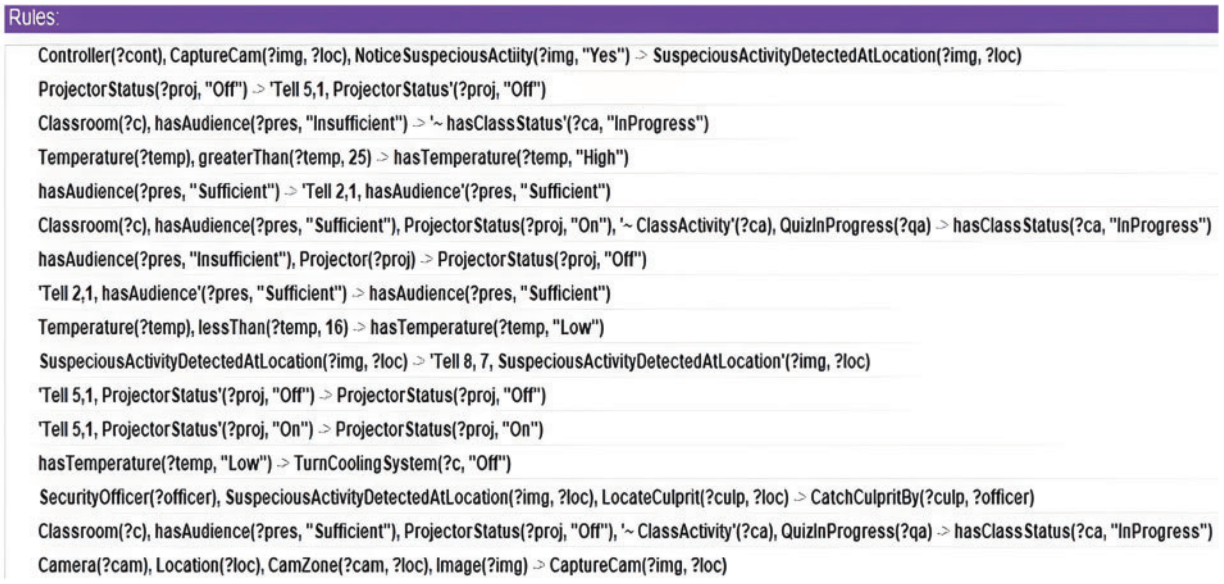
Figure 2: Smart learning environment ontology hierarchy and metrics


Data property hierarchy: owl:topDataPrope
Asserted
Object property hierarchy: owl:topObjectPr
Asserted

```
owl:topDataProperty
    ~ hasClassStatus
    hasAudience
    hasClassActivity
    hasClassStatus
    hasTemperature
    NoticeSuspeciousActiity
    ProjectorStatus
    Tell 2,1, hasAudience
    Tell 3, 4, hasTemperature
    Tell 5,1, ProjectorStatus
    TurnCoolingSystem

owl:topObjectProperty
    CamZone
    CaptureCam
    CatchCulpritBy
    LocateCulprit
    SuspeciousActivityDetectedAtLocation
    Tell 6, 8, CaptureCam
    Tell 8, 7, SuspeciousActivityDetectedAtLocation
```

Figure 3: Data and object properties



```
Rules:
Controller(?cont), CaptureCam(?img, ?loc), NoticeSuspeciousActiity(?img, "Yes") -> SuspeciousActivityDetectedAtLocation(?img, ?loc)
ProjectorStatus(?proj, "Off") -> 'Tell 5,1, ProjectorStatus'(?proj, "Off")
Classroom(?c), hasAudience(?pres, "Insufficient") -> '~ hasClassStatus'(?ca, "InProgress")
Temperature(?temp), greaterThan(?temp, 25) -> hasTemperature(?temp, "High")
hasAudience(?pres, "Sufficient") -> 'Tell 2,1, hasAudience'(?pres, "Sufficient")
Classroom(?c), hasAudience(?pres, "Sufficient"), ProjectorStatus(?proj, "On"), '~ ClassActivity'(?ca), QuizInProgress(?qa) -> hasClassStatus(?ca, "InProgress")
hasAudience(?pres, "Insufficient"), Projector(?proj) -> ProjectorStatus(?proj, "Off")
'Tell 2,1, hasAudience'(?pres, "Sufficient") -> hasAudience(?pres, "Sufficient")
Temperature(?temp), lessThan(?temp, 16) -> hasTemperature(?temp, "Low")
SuspeciousActivityDetectedAtLocation(?img, ?loc) -> 'Tell 8, 7, SuspeciousActivityDetectedAtLocation'(?img, ?loc)
'Tell 5,1, ProjectorStatus'(?proj, "Off") -> ProjectorStatus(?proj, "Off")
'Tell 5,1, ProjectorStatus'(?proj, "On") -> ProjectorStatus(?proj, "On")
hasTemperature(?temp, "Low") -> TurnCoolingSystem(?c, "Off")
SecurityOfficer(?officer), SuspeciousActivityDetectedAtLocation(?img, ?loc), LocateCulprit(?culp, ?loc) -> CatchCulpritBy(?culp, ?officer)
Classroom(?c), hasAudience(?pres, "Sufficient"), ProjectorStatus(?proj, "Off"), '~ ClassActivity'(?ca), QuizInProgress(?qa) -> hasClassStatus(?ca, "InProgress")
Camera(?cam), Location(?loc), CamZone(?cam, ?loc), Image(?img) -> CaptureCam(?img, ?loc)
```

Figure 4: OWL 2 RL and SWRL rules

4 Context-Aware Predictive Formalism for Smart Learning Environment

This section presents a reinforcement learning based context-aware multi-agent reasoning model using contextual defeasible reasoning. Contextual defeasible logic (CDL) is a non-monotonic reasoning technique to handle conflicting and/or incomplete information. In a context-aware deployment setting, there are *n* number of agents deployed in the system where each agent is connected to its corresponding sensor or another agent to acquire/exchange contextual information. Each agent in the system is programmed to solve a specific problem using its own knowledge base which consists of a set of strict rules and defeasible rules, and a reasoning strategy. Each agent is developed to perform dedicated tasks only. Whenever an agent needs to acquire contextualized information, the system initiates a request to extract the corresponding agent's rules from the ontology. Agents in the system may use two types of rules; deduction rules and communication rules. Deduction rules are of the forms of antecedents and consequent patterns and these rules are further classified into strict rules and defeasible rules. Strict rules can be executed in a conventional way whereas defeasible rules are used to defeat conflicting information. In case of conflicting information, CDL provides a feature superiority relation to prioritize the rules and resolve conflicting information.

4.1 Reinforcement Learning Agents' Behavior and Reasoning Strategy

Reinforcement learning (RL) provides a mechanism of reward and punishment where each agent in the system is trained to specify the set of rules for the execution and set priorities based on the agent's rewards. The rules with higher rewards should get higher priorities for the execution of contextualized information. A key challenge is aligning the adaptive behavior of RL with the static structure of the ontology, particularly during rapid shifts in decision-making priorities. However, in the system, RL often solves sequential decision-support problems where agents acquire contextualized information from the environment, perform reasoning based on the reward/punishment policy, and then take appropriate actions accordingly. RL based context-aware agents autonomously learn the reasoning policy and select the set of rules to be triggered to fulfill the desired goals. Fig. 5 depicts the working flow of the proposed context-aware predictive smart learning environment. As mentioned above, the system has three-tier architecture: Context acquisition process, context modeling using semantic knowledge, and contextual predictive reasoning formalism. To suitably model the system, we develop n number of RL based context-aware multi-agent systems which can be represented as $A_g = \{1, 2, 3, \ldots, n\}$. As the system starts its execution, each agent includes a set of rules along with a set of facts and superiority relations, represented by a tuple as $(\Omega, >)$ where Ω represents the set of all rules (strict and defeasible) and a set of facts. In the system, the set of deduction and communication rules is expressed as $\mathfrak{R} = \{\mathfrak{R}^{Ded}, \mathfrak{R}^{Com}\}$ and the set of deduction rules are further classified as $\mathfrak{R}^{Ded} = \{\mathfrak{R}^{s}, \mathfrak{R}^{d}\}$. $\mathfrak{R}^{s}$ represents the set of strict rules and $\mathfrak{R}^{d}$ represents the set of defeasible rules. Rules are of the form of antecedents and consequent patterns, for example, $A_1, A_2, \ldots, A_n \rightarrow A$. The antecedents are written in conjunction forms as $A_1, A_2, \ldots, A_n$ and if all facts of antecedents are matched then the rule is executed and the consequent is developed. To specify the defeasible rules, we use the double arrow symbol ($\Rightarrow$) and for the rest of all rules, we use the single arrow symbol ($\rightarrow$). The important point is to note that rules are executed with other rules, however, these rules can be fired to resolve inconsistent or incomplete contextual information. As the system starts execution, agents in the system can perform three core actions: *Rule*, *Copy*, and *Idle*. *Rule* action is triggered whenever an agent fires any type of rule to perform a certain action. *Copy* action is executed whenever agents exchange information among themselves and *Idle* action is executed when an agent transits to the next state without any action. To exchange information, agents may use communication primitives as "*Tell(8,7, SuspeciousActvityDetectedAtLocation(?img,?loc))*" which means that agent 8 (controller agent) notifies agent 7 (security agent) about the location of the incident so that culprits can be captured on immediate basis. In RL based multi-agent systems, an agent learns how to carry out a task in an environment. The agent gets feedback in the form of real-valued incentives as it acts. Using this data, reinforcement learning algorithms progressively enhance an agent's control strategy to optimize the agent's overall long-term predicted reward. Nevertheless, contemporary RL methods face challenges related to sample inefficiency and prolonged training times, limiting their scalability and applicability in real-world scenarios. Furthermore, more complications and factors need to be taken into account when handling situations in which several reinforcement learning agents interact in a common space setting known as Multi-Agent Reinforcement Learning. In the domain of reinforcement learning, an agent undergoes a learning process by experimenting with actions to accomplish a task within a given environment. With each action, the agent receives feedback in the form of real-valued rewards. At every step, the agent observes the state (s) of its environment. Employing its policy (π), the agent chooses and executes an action (a), influencing the environment state to change to s'. The agent then observes this updated state, along with a corresponding reward (r), and employs this information to refine its policy. Considering the existing state, the agent's policy dictates agents about the action plan. Each agent in the system has its own policy for the execution of its dedicated tasks, however, multiple agents may use the same policy as well. The policy of an agent in a state can be represented as:

$$a = \pi(s)$$

where s is the current state and a is the action selected by the policy. In the system's environment dynamics, the set of actions triggered by agents processes a transition flow from state s to s'. This transition can be represented as a function $P(s'|s, a)$, which gives the probability (Pro) of transitioning to state (s'), given that action a is taken in state s. As the agents in the system follow the reinforcement learning paradigm, so each action taken by an agent either gets a reward or punishment. In case of successful action, the agent receives a reward r from the environment after taking action a in the state s. The reward function can be denoted as follows in which $R\ (s, a, s')$ is the reward received after transitioning from state s to s' by taking action a.

$$r = R(s, a, s')$$

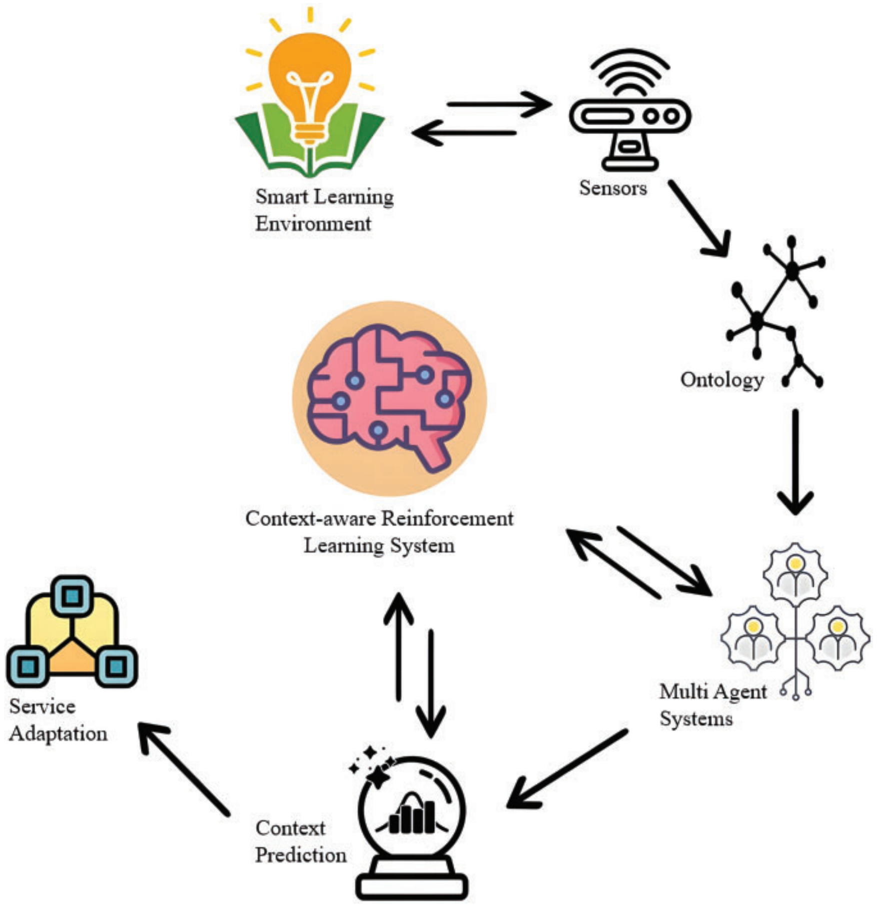


Figure 5: Context-aware RL system framework

In case if agents performs idle transitions, then the system may not get any reward. This can be set at the design time in the policy of the system. The policy may be improved to optimize the decision making process.

The whole learning process can be summed up as a cyclic process: the agent interacts with the environment repeatedly, chooses actions based on its policy, monitors rewards and states, and updates its policy. This procedure is repeated over several episodes until the agent reaches the near-optimal or optimal policy that maximizes the total expected reward over the long run, which is expressed as follows:

$$s \xrightarrow{\pi(s)} a \xrightarrow{P(s'|s,a)} s' \xrightarrow{R(s,a,s')} r \xrightarrow{PolicyUpdate} \pi'$$

The reinforcement learning paradigm is used to solve the ranking problem, which is formalized as a Markov Decision Process (MDP). The reinforcement learning can be modelled by a Markov decision process (MDP) and is represented by a tuple $\mathbb{M} = (A_g, S, \Omega, V, A_{act}, T, R_a)$ where A_g is the set of agents, S represents a set of states, Ω is a set of ground atomic facts and rules, $V : \Omega \rightarrow \wp(S)$ is a mapping which assigns the truth of ground atomic facts to the subset of states S, A_{act} is a non-empty set of actions including idle action, R_a is the reward function having a set of rewards $R_a = \{1, 2, \ldots, r\}$, and $T : S \times A_g \rightarrow \wp(A_{act}) \backslash \{\varnothing\}$ is a transition function which shows the number of available actions (moves) for each agent i at state $T(s, i)$ where $i \in A_g$ and $s \in S$. In the model, the system transits from one state to another state as a result of performing actions by agents. To model the system, we denote the set of agents $A_g = \{1, 2, \ldots, n\}$ and a set of preference order to set the rewards $R_a = \{1, 2, \ldots, r\}$ and required by the agents to perform a set of actions $A_{act} = \bigcup_{act \in \Omega} A_{act}$ where A_{act} exclusively represents the set of actions performed by the agent i. In the system, the agent learns to maximize its long-term benefit by receiving feedback in the form of rewards or penalties based on its behaviors.

4.2 Multi-Agent Advising Strategy

Considering a case scenario of smart class room where agents autonomously detects the policy plan for the execution and advise agents to perform specified tasks. Each agent in the system works based on its predefined policy π according to its customized requirements.

Agents' Advising Strategy: When the execution of multiple tasks is equally important, then the system prioritizes the agent's tasks based on a strategy known as the Advising Strategy. In Algorithm 1, the system advises the projector agent with the policy to switch on the projector if the count reaches 5 and above. We call this early advising technique where a presence detector agent continuously tracks the number of people in a classroom location. After getting advice, the projector turns on when the head count reaches a threshold of five, and switch off when its count reaches below 5.

Algorithm 1: Early advising

```
Input: Head Count → $H_C$
Initialization: State → s, Presence Detector → $P_D$
Output: Projector Status → $P_S$
for $P_D$ (s) do
    if $H_C \geq 5$ then
        Advice $(\pi(s), Ag_{proj})$
        $P_S$ ← ON
    else
```

(Continued)

Algorithm 1 (continued)

```
        $P_S \leftarrow$ OFF
    end if
end for
```

Priority Advising Strategy: When each stage of a task is equally important, it seems reasonable to start early counseling. However, the system detects a conflicting context in a specific state, the system then sets preferences on the concurrent tasks in the hierarchy of states and prioritize the conflicting context to be executed earlier. In such situations, the system generate conflict set to prioritize contextual information. Considering a situation in the case study, Algorithm 2 depicts the prioritized advising strategy where a surveillance system takes over priority advising when hazardous situation occur. In this case, this algorithm can be helpful in security or surveillance systems where it's important to spot suspicious activity and assists to catch the culprit.

Algorithm 2: Prioritized advising

```
Input: Notice suspicious activity → $notice_{SuspAct}$, Capture Camera → $cap_{cam}$, location image → $loc_{img}$
Initialization: Controller agent → $Ctrl_{agent}$, location → loc
Output: Catch the Culprit → $catch_{culprit}$
for $Ctrl_{agent}(s)$ do
    if $notice_{SuspAct}(loc)$ & & $cap_{cam}(loc_{img})$ then
        $catch_{culprit}$
    end if
end for
```

Predictive Advising Strategy: Based on many aspects of the smart classroom, such as available resources, projector status, class activity, and ongoing quizzes, the predictive advising Algorithm 3 determines if a class session is in process. During the class session in our case, teachers and students can utilize this forecast to get immediate advice or support. To evaluate the overall agents' learning and execution behavior, a set of few rules for smart learning environment are presented in Table 2. We define acronyms for the terms in Table 3. Formally, a system's policy is developed at the design time of the system and it is updated with a set of actions performed by agents in terms of transitions. In case of Smart Classroom, $[SCR]$ is the policy probability $Pro_{[SCR]}$ which perform action (a) at specific state (s) and transit to the next state (s'). The system assigns a reward based on specific action triggered at the right time and in the right place which is expressed as $(s', r|s, a)$ and increase the value in reward (r) in terms of time R_{t+1}. In case if the system does not get rewarding action, then the policy probability $Pro_{[SCR]}$, is represented by the expression (s', $P_u|s$, a) and increase the value in punishment (P_u) in terms of time P_{t+1}. We execute a few set of rules in antecedent and consequent pattern as shown in Table 4. The Fig. 6 depicts a comprehensive overview of the policy (both reward and punishment) based agents' communication mechanism to achieve the desired goal. The description of few agent's task are listed below:

- Presence detector agent notifies the projector agent to turn on the projector when the headcount is 5 or above and switches off when its count limit reaches below 5.
- Temperature agent notifies the cooling system agent to switch to the normal mode when the temperature remains between 18 and 25 degree Celsius, otherwise convert it into low and high mode according to the condition.

- The Security agent notifies the camera agent to capture the suspicious activity at the location and sends the details to the controller agent through the security agent to catch the culprit.

```
**Algorithm 3:** Predictive advising
**Input:** Class Sufficient → $cl_{suff}$, Projector Status → $Proj_{st}$, Class Activity → $cl_{act}$, Quiz in progress → $qz_{iprog}$
**Initialization:** Smart classroom → $S_{CR}$,
**Output:** Class in progress → $cl_{iprog}$
**for** $S_{CR}(s)$ **do**
    **if** $cl_{suff} \,||\, Proj_{st} \rightarrow$ ON $||\, cl_{act} \rightarrow$ ON $||\, qz_{iprog}$ **then**
        $cl_{iprog}$
    **end if**
**end for**
```

Table 3: Symbol definitions

Symbol	Definition
`[ant]`	Antecedent
`[cons]`	Consequent
`[scr]`	SmartClassRoom
`[t]`	Time
`[R]`	Reward
`[Pro]`	Probability
`[S]`	State
`[Pu]`	Punishment
`[nip]`	NotInProgress
`[ip]`	InProgress
`[cap]`	Capture
`[cl]`	Class
`[Pt]`	Punishment (Time)
`[Rt]`	Reward (Time)
`[NoticeSuspAct]`	NoticeSuspiciousActivity
`[SusActDetatLoc]`	SuspiciousActivity DetectedatLocation
`[CapCam]`	CaptureCamera
`[LocateCul]`	LocateCulprit

Table 4: RL based set of antecedent and consequent pattern

Case-1
$r_{1[ant]}$ = classroom(c-307), hasAud(sufficient), ProjStat(on),ClAct(yes)
$r_{1[cons]}$ = hasClassStat(ip) $\Rightarrow P_{ro[SCR]}$ (s', r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{1[ant]}$, $R_t = r_{11}$ — $s_t = r_{1[cons]}$, R_{t+1}]
$r_{2[ant]}$ = classroom(c-307), hasAud(sufficient), ProjStat(on),ClAct(no), QuizIP(yes)
$r_{2[cons]}$ = hasClassStat(ip) $\Rightarrow P_{ro[SCR]}$ (s', r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{2[ant]}$, $R_t = r_{12}$ — $s_t = r_{2[cons]}$, R_{t+2}]
$r_{3[ant]}$ = classroom(c-307), hasAud(insufficient)

(Continued)

Table 4 (continued)

$r_{3[cons]}$ = hasClassStat(nip) $\Rightarrow P_{ro[SCR]}$ (s', P_u — s, a) = $P_{ro[SCR]}$ [$s_t = r_{3[ant]}$, $P_t = r_{14}$ — $s_t = r_{3[cons]}$, P_{t+1}]
$r_{4[ant]}$ = classroom(c-307), hasAud(sufficient), ProjStat(off),ClAct(no), QuizIP(yes)
$r_{4[cons]}$ = hasClassStat(ip) $\Rightarrow P_{ro[SCR]}$ (s',r — s,a) = $P_{ro[SCR]}$ [$s_t = r_{4[ant]}$, $R_t = r_{13}$ — $s_t = r_{4[cons]}$, R_{t+3}]
Case-2
$r_{5[ant]}$ = PresenceDet(yes), HeadCo(6), GreaterThan(5)
$r_{5[cons]}$ hasAud(sufficient) $\Rightarrow P_{ro[SCR]}$ (s', r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{5[ant]}$, $R_t = r_{21}$ — $s_t = r_{5[cons]}$, R_{t+4}]
$r_{6[ant]}$ = PresenceDet(yes), HeadCo(5orless), LessThan(6)
$r_{6[cons]}$ hasAud(insufficient) $\Rightarrow P_{ro[SCR]}$ (s', P_u — s, a) = $P_{ro[SCR]}$ [$s_t = r_{6[ant]}$, $P_t = r_{22}$ — $s_t = r_{6[cons]}$, P_{t+2}]
$r_{7[ant]}$ = PresenceDet(yes), HeadCo(6), EqualTo(6)
$r_{7[cons]}$ hasAud(sufficient) $\Rightarrow P_{ro[SCR]}$ (s', r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{7[ant]}$, $R_t = r_{22}$ — $s_t = r_{7[cons]}$, R_{t+5}]
$r_{8[ant]}$ = PresenceDet(yes), HeadCo(0), EqualTo(0)
$r_{8[cons]}$ hasAud(insufficient) $\Rightarrow P_{ro[SCR]}$ (s', P_u — s, a) = $P_{ro[SCR]}$ [$s_t = r_{8[ant]}$, $P_t = r_{22}$ — $s_t = r_{8[cons]}$, P_{t+3}]
Case-3
$r_{9[ant]}$ = clTemp(c-307), GreaterThan(18), LessThan(25)
$r_{9[cons]}$ = hasclTemp(normal) $\Rightarrow P_{ro[SCR]}$ (s', r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{9[ant]}$, $R_t = r_{31}$ — $s_t = r_{9[cons]}$, R_{t+6}]
$r_{10[ant]}$ = clTemp(c-307), GreaterThan(25)
$r_{10[cons]}$ = hasclTemp(high) $\Rightarrow P_{ro[SCR]}$ (s', P_u — s, a) = $P_{ro[SCR]}$ [$s_t = r_{10[ant]}$, $P_t = r_{32}$ — $s_t = r_{10[cons]}$, P_{t+4}]
$r_{11[ant]}$ = clTemp(c-307), LessThan(16)
$r_{11[cons]}$ = hasclTemp(low) $\Rightarrow P_{ro[SCR]}$ (s', P_u — s, a) = $P_{ro[SCR]}$ [$s_t = r_{11[ant]}$, $P_t = r_{33}$ — $s_t = r_{11[cons]}$, P_{t+5}]
Case-4
$r_{12[ant]}$ = hasclTemp(low)
$r_{12[ant]}$ = CoolingSystem(off) $\Rightarrow P_{ro[SCR]}$ (s',r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{12[ant]}$, $R_t = r_{43}$ — $s_t = r_{12[cons]}$, R_{t+7}]
$r_{13[ant]}$ = hasclTemp(high)
$r_{13[ant]}$ = CoolingSystem(on) $\Rightarrow P_{ro[SCR]}$ (s',r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{13[ant]}$, $R_t = r_{43}$ — $s_t = r_{13[cons]}$, R_{t+8}]
Case-5
$r_{14[ant]}$ = hasAud(insufficient), clProjector(off)
$r_{14[cons]}$ = clProjectStat(off) $\Rightarrow P_{ro[SCR]}$ (s',P_u — s, a) = $P_{ro[SCR]}$ [$s_t = r_{14[ant]}$, $P_t = r_{53}$ — $s_t = r_{14[cons]}$, P_{t+6}]
$r_{15[ant]}$ = hasAud(sufficient), clProjector(on)
$r_{15[cons]}$ = clProjectStat(on) $\Rightarrow P_{ro[SCR]}$ (s', r — s, a) = $P_{ro[SCR]}$ [$s_t = r_{15[ant]}$, $R_t = r_{54}$ — $s_t = r_{15[cons]}$, R_{t+9}]

(Continued)

Table 4 (continued)

Case-6
$r_{16[ant]}$ = Cam(101), Location(courtyard), CamZone(cam-101), Image(courtyard) $r_{16[cons]}$ = CapCam(courtyard) $\Rightarrow P_{ro[SCR]}$ $(s'$, r — s, a) = $P_{ro[SCR]}$ $[s_t = r_{16[ant]}$, $R_t = r_{61}$ — $s_t = r_{16[cons]}$, $R_{t+10}]$
Case-7
$r_{17[ant]}$ = SecurityOff(alan), SusActDetAtLoc(yes), LocateCul(jimmy, courtyard) $r_{17[cons]}$ = CatchCul(jimmy) $\Rightarrow P_{ro[SCR]}$ $(s'$, r — s, a) = $P_{ro[SCR]}$ $[s_t = r_{17[ant]}$, $R_t = r_{72}$ — $s_t = r_{17[cons]}$, $R_{t+11}]$
Case-8
$r_{18[ant]}$ = Contro(tim), CapCam(courtyard), NoticeSuspAct(yes) $r_{18[cons]}$ = SuspActDetAtLoc(courtyard) $\Rightarrow P_{ro[SCR]}$ $(s'$, r — s, a) = $P_{ro[SCR]}$ $[s_t = r_{18[ant]}$, $R_t = r_{82}$ — $s_t = r_{18[cons]}$, $R_{t+12}]$

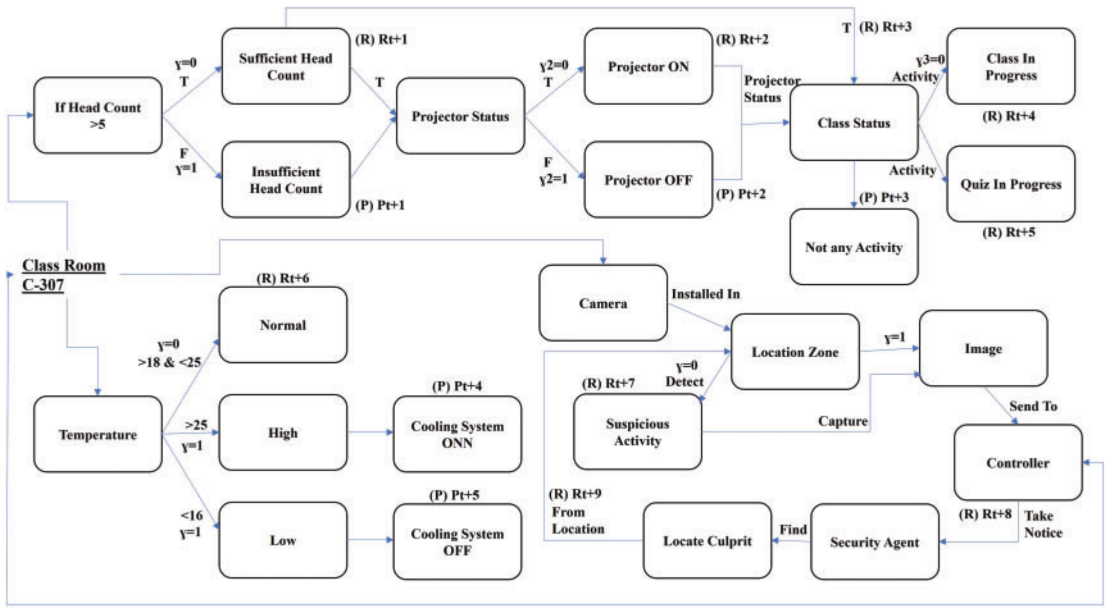


Figure 6: Agents interaction

We use *EDOH* [22] to extract the ontology axioms in the form of horn-clause rules. Once the rules are transformed from the ontology, then the system defines agents' policy based on the rewards/punishment function and sets priorities of the rules in order to perform actions. As the Markov decision process is a stochastic control process to model decision-making situations, actions and rewards have a very close association. Each time an action performed by the agent allows the system to reset the policy and assign priorities to the rules. Agents in the system trigger rules based on newly defined policies. In this way, agents update the rule priorities dynamically to handle incomplete and inconsistent contextual information. The system designer develops all rules at the design time of the system according to the provided case study, and the rule priorities are set at initial execution time. Then priorities of the rules may change based on the policies of agents' actions. We assume each rule $r_i \in \mathfrak{R}$ has a priority value. More specifically, the priority determines the preference order in the execution of the rules. Context-aware agents have the ability to reason non-monotonically and cooperate non-deterministically to accomplish the desired objectives.

5 Formal Verification of the Proposed System

Literature has witnessed that researchers have been using model-checking techniques to formally model the system and verify its correctness properties before the actual implementation of the system. Model checking is widely usable by the industry in various domains such as safety-critical systems, e-healthcare, e-government, etc. to check the correctness properties whether these are hardware or software or a combination of both. However, in theoretical models, it is considered to be the most appropriate approach. It is an automated or semi-automated modeling technique where the system can be encoded in the model checker and properties can be specified with the intention to verify the system's correctness properties. Model checking is applied to formalize the system specifications and property specifications. System specification specifies the concurrent transitions of the system whereas property specification verifies whether the required properties hold in that model or not. However, if we talk about previous studies [21,23,24], our results as described in Section 5 are better in comparison. In the literature, various model-checking tools have been used according to the required specification of the system such as UPPAAL [10], Maude [25], Petri nets [26], etc. Among others, we opt UPPAAL model checker due to its graphical modeling and simulation behavior and it is more convenient to model, specify and verify real-time systems. UPPAL model checker has three modules: (a) editor can be used to model the domain along with its declaration and configurations. (b) The simulator provides a simulation zone to simulate the process executions, and (c) the verifier analyzes the system behavior and verifies correctness properties. Model checking is a formal verification method that analytically discovers all possible states of a system to confirm its correctness. Significantly, like a computer chess program that assesses each possible move, a model checker tool such as UPPAAL observes each possible situation the system could encounter. This comprehensive investigation permits verification of whether a given system model satisfies specific requirements or properties. Different outdated methods such as testing, emulation, or simulation, model checking can uncover even delicate and hard-to-detect errors, providing an advanced level of assurance in system reliability.

We model the domain of the smart learning environment in the editor zone and simulate the system processes. In Table 2, we have developed a set of rules to model a non-monotonic reasoning based context-aware predictive decision support system. To suitably simulate the system behavior, we need to verify the system's correctness properties such as safety, liveness, and robustness properties. Fig. 7 shows the simulation of class status. As per defined specifications, the system should check the status of the class at the specified time and monitors whether a class is held at the right time and in the right place. In addition, the system specification will also monitor the class teaching duration and quiz time duration of the class. In Fig. 8, the system shows the configuration of the vigilance system. Let's assume a camera detects some suspicious activities at the specified location, the system should immediately notify to the controller room to take action immediately. In the system, channels can be used to enable communication among different processes. Moreover, Fig. 9 elaborates the system encoding using model checking and formal verification with the help of UPPAAL tool.

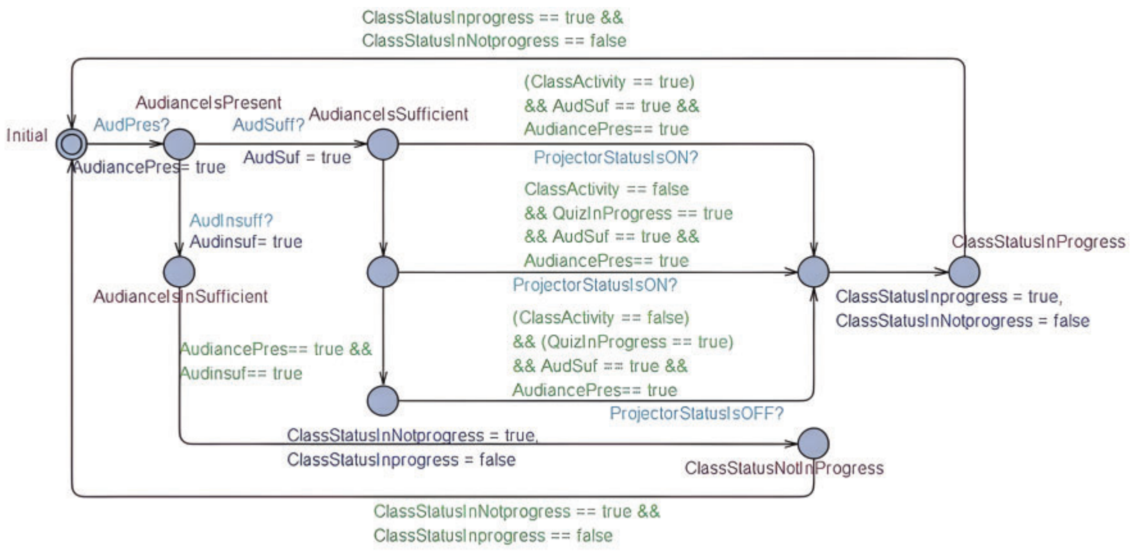


Figure 7: Class status simualtion

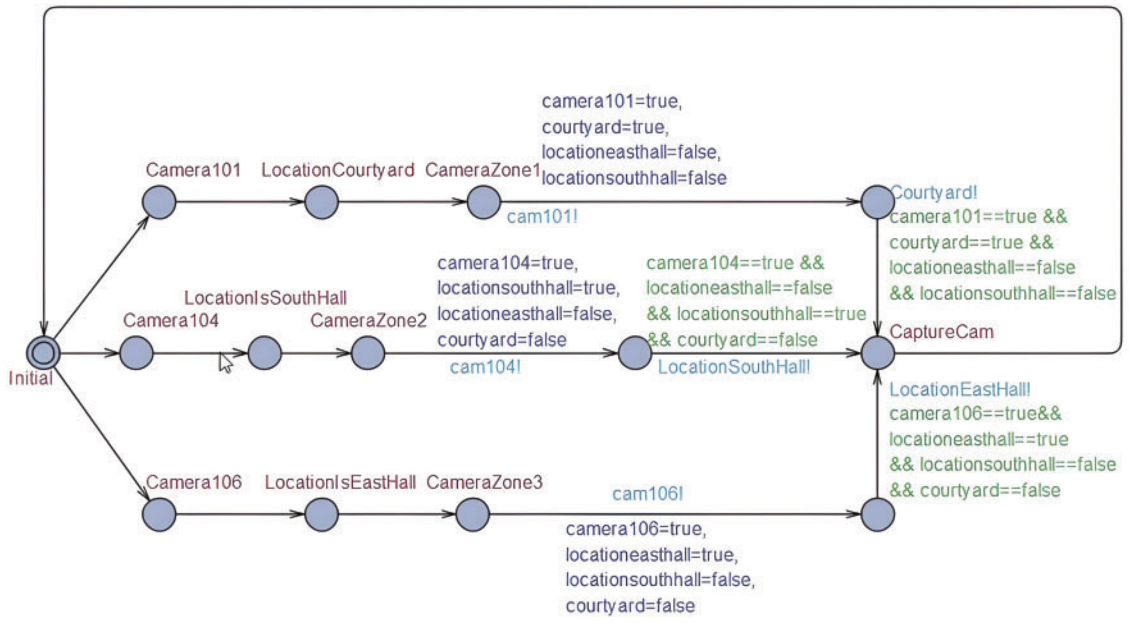


Figure 8: Capture camera zone simulation

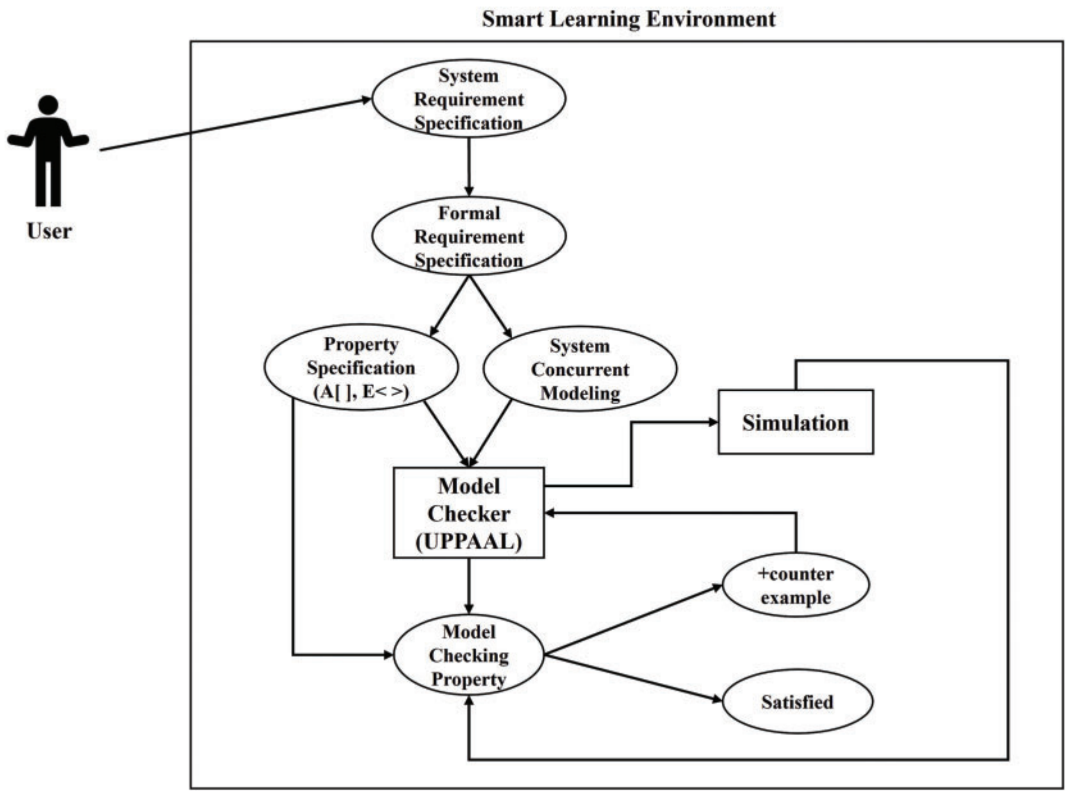
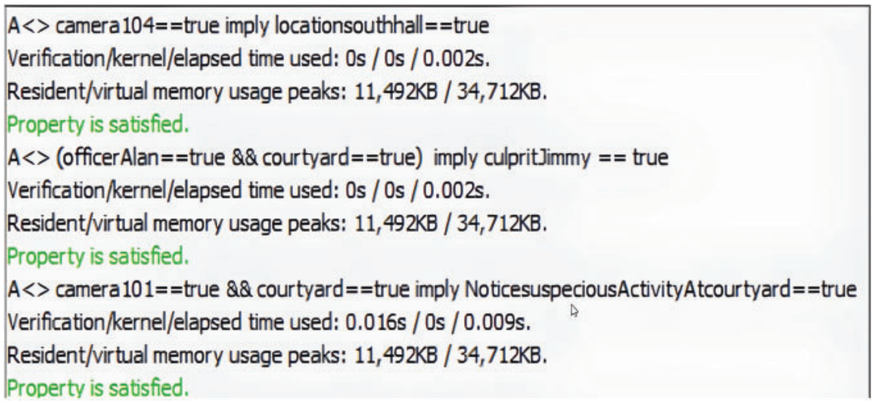

Smart Learning Environment
User
System Requirement Specification
Formal Requirement Specification
Property Specification (A[ ], E< >)
System Concurrent Modeling
Simulation
Model Checker (UPPAAL)
+counter example
Model Checking Property
Satisfied

Figure 9: System encoding using model checking & formal verification

We need to check and verify the liveness property which means that the system should ensure that desirable property eventually holds in the model. Liveness property is satisfied when a process enters its critical section and then eventually would become successful. For example; as per system specification, whenever suspicious activity is performed, the system must have to be vigilant and activated to generate alarms to catch the culprit. Fig. 10 depicts how the system ensures the execution of this property. Moreover, the verification time of each property is represented in graph form in Fig. 11. In which the X-axis represents the each property and the Y-axis represents the time span in seconds. A safety property ensures that something bad must not happen. The main goal of a safety property is to prevent the risk of getting loss or absence of the properties when the system is operational. Deadlock property is a critical property related to synchronization and resource allocation in concurrent systems. A deadlock situation occurs when two or more processes are unable to move and wait for the actions to be triggered. This property must hold to satisfy the requirements. Reachability property is used to analyze the behavior of a system as shown in Fig. 12. Additionally, the verification result for reachability property is shown in graph form in Fig. 13. In which the X-axis shown the property being verified, the left Y-axis shown the time in seconds and the right Y-axis shown the memory usage in KBs. It specifies the state in the system can be reached following a sequence of transitions or actions. This property is very useful to verify the correctness properties of the program. We executed various queries using the UPPAAL tool, and their results, execution time (in seconds), and memory usage (in KBs) can be seen in Table 5. If we compare our results with existing approaches [27–29] to evaluate how much better our outcomes are, we can say that even without ensuring full scalability and compatibility, and only for the sake of comparison at a generic level. Our verification results, in terms of time (in seconds) and memory usage (in KBs) are better than those of the existing approaches [30–32].



```
A<> camera104==true imply locationsouthhall==true
Verification/kernel/elapsed time used: 0s / 0s / 0.002s.
Resident/virtual memory usage peaks: 11,492KB / 34,712KB.
Property is satisfied.
A<> (officerAlan==true && courtyard==true)  imply culpritJimmy == true
Verification/kernel/elapsed time used: 0s / 0s / 0.002s.
Resident/virtual memory usage peaks: 11,492KB / 34,712KB.
Property is satisfied.
A<> camera101==true && courtyard==true imply NoticesuspeciousActivityAtcourtyard==true
Verification/kernel/elapsed time used: 0.016s / 0s / 0.009s.
Resident/virtual memory usage peaks: 11,492KB / 34,712KB.
Property is satisfied.
```

Figure 10: Properties to check suspicious activity and subsequent action

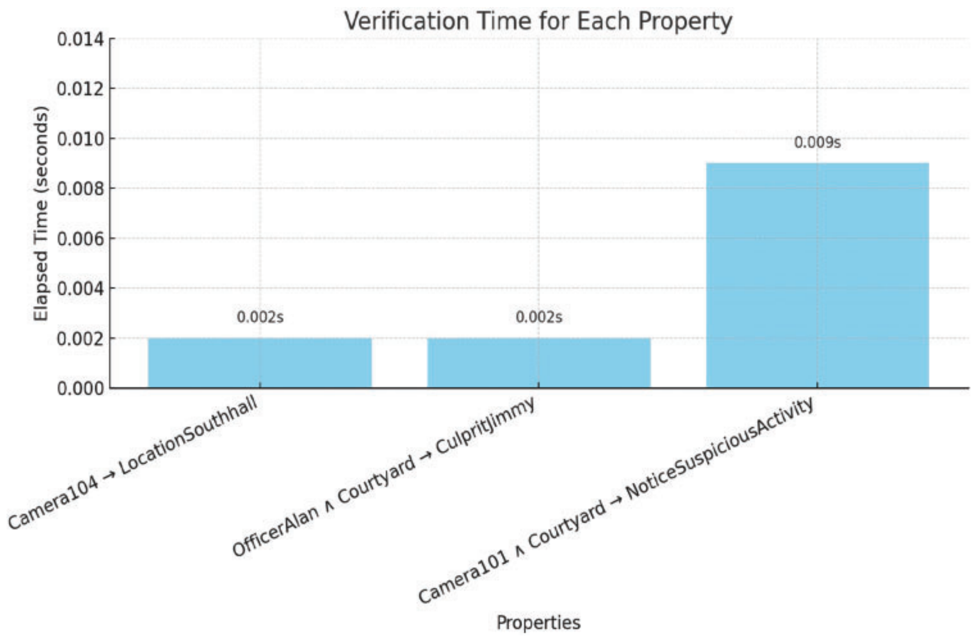
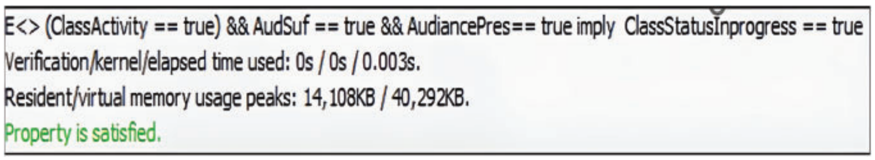


Figure 11: Verification time for each property



```
E<> (ClassActivity == true) && AudSuf == true && AudiancePres== true imply  ClassStatusInprogress == true
Verification/kernel/elapsed time used: 0s / 0s / 0.003s.
Resident/virtual memory usage peaks: 14,108KB / 40,292KB.
Property is satisfied.
```

Figure 12: Property to check class status

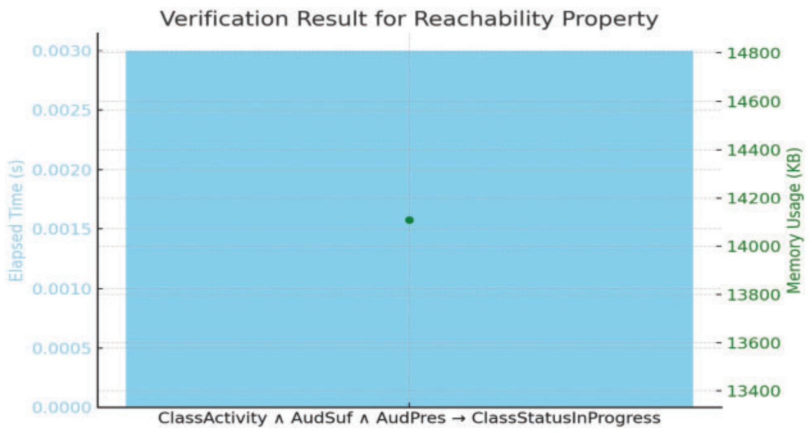


Figure 13: Verification result for reachability property

Table 5: UPPAAL verification results

Property ID	Query	Results	Time (Seconds)	Memory (KB)
R72	A<> (officerAlan = = true && courtyard = = true) imply culpritJimmy = = true	Satisfied	0.002	34,712
R82	A<> camera101 = = true && courtyard = = true imply Notice-suspeciousActivityAtcourtyard = = true	Satisfied	0.009	34,712
R11	E<> (ClassActivity = = true) && AudSuf = = true && AudiancePres = = true imply ClassStatusInprogress = = true	Satisfied	0.003	40,292

6 Conclusion

In this paper, we have proposed a semantic knowledge-based context-aware predictive reasoning formalism for smart learning environments. This model is based on the Markov decision process where the students and academicians can manage their academic as well as administrative activities autonomously. We have deployed contextual defeasible reasoning techniques to handle conflicting and/or inconsistent contextual information, which may cause context exchanges to/from semantic knowledge sources and developed agents advising strategies for agents to gain maximal rewards by continuous learning and then adapt their behavior accordingly. We model the case study in UPPAAL model checker to declare the processes, simulate the system's behavior and verify correctness properties. In future work, we will propose a reinforcement learning framework incorporating heterogeneous knowledge sources based on non-monotonic reasoning, and develop state-of-the-art applications of the proposed framework in a real-time environment. This system would provide artificial intelligence (AI) assistive facilities to users in different domains.

Acknowledgement: The authors would like to thank all individuals and institutions who contributed to the successful completion of this research. Their support and valuable insights were greatly appreciated. Special thanks to the National Research Foundation (NRF), Republic of Korea, under project BK21 FOUR (4299990213939) for supporting this work.

Funding Statement: This work was supported by the National Research Foundation (NRF), Republic of Korea, under project BK21 FOUR (4299990213939).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Methodology, Writing—Original Draft Preparation: Taimoor Hassan, Hafiz Mahfooz Ul Haque; Literature Review, Visualization: Muhammad Nadeem Ali, Byung-Seo Kim, Hamid Turab Mirza; Reviewing and Editing: Ibrar Hussain. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Ben Hafaiedh I, Elaoud A, Maddouri A. A formal model-based approach to design failure-aware Internet of Things architectures. J Reliab Intell Environ. 2024;10(4):413–30. doi:10.1007/s40860-024-00225-z.
2. Gholami H, Chang CK, Aduri P, Ma A, Rekabdar B. A data-driven situation-aware framework for predictive analysis in smart environments. J Intell Inform Syst. 2022;59(3):679–704. doi:10.1007/s10844-022-00721-9.
3. Hashem IA, Siddiqa A, Alaba FA, Bilal M, Alhashmi SM. Distributed intelligence for IoT-based smart cities: a survey. Neural Comput Appl. 2024;36(27):16621–56. doi:10.1007/s00521-024-10136-y.
4. Mirisaee SH, Zin AM. A context-aware mobile organizer for university students. J Comput Sci. 2009;5(12):898. doi:10.3844/jcssp.2009.898.904.
5. Martín E, Carro RM, Rodríguez P. A mechanism to support context-based adaptation in m-learning. In: Innovative Approaches for Learning and Knowledge Sharing: First European Conference on Technology Enhanced Learning, EC-TEL 2006; 2006 Oct 1–4; Crete, Greece: Springer; 2006. p. 302–15.
6. Torrey L, Taylor M. Teaching on a budget: agents advising agents in reinforcement learning. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems. Saint Paul, MN, USA; 2013. p. 1053–60.
7. Ilhan E, Gow J, Perez-Liebana D. Teaching on a budget in multi-agent deep reinforcement learning. In: 2019 IEEE Conference on Games (CoG). London, UK: IEEE; 2019. p. 1–8.
8. Magdich R, Jemal H, Ben Ayed M. Context-awareness trust management model for trustworthy communications in the social internet of things. Neural Comput Appl. 2022;34(24):21961–86. doi:10.1007/s00521-022-07656-w.
9. Zeeshan K, Hämäläinen T, Neittaanmäki P. Internet of things for sustainable smart education: an overview. Sustainability. 2022;14(7):4293. doi:10.3390/su14074293.
10. Larsen KG, Pettersson P, Yi W. UPPAAL in a nutshell. Int J Softw Tools Technol Trans. 1997;1(1–2):134–52. doi:10.1007/s100090050010.
11. Huang LS, Su JY, Pao TL. A context aware smart classroom architecture for smart campuses. Appl Sci. 2019;9(9):1837. doi:10.3390/app9091837.
12. Paudel P, Kim S, Park S, Choi KH. A context-aware IoT and deep-learning-based smart classroom for controlling demand and supply of power load. Electronics. 2020;9(6):1039. doi:10.3390/electronics9061039.
13. Behzadipour F, Ghasemi Nezhad Raeini M, Abdanan Mehdizadeh S, Taki M, Khalil Moghadam B, Zare Bavani MR, et al. A smart IoT-based irrigation system design using AI and prediction model. Neural Comput Appl. 2023;35(35):24843–57. doi:10.1007/s00521-023-08987-y.

14. Liu S, Chen Y, Huang H, Xiao L, Hei X. Towards smart educational recommendations with reinforcement learning in classroom. In: 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). Wollongong, NSW, Australia: IEEE; 2018. p. 1079–84.
15. Burda Y, Edwards H, Storkey A, Klimov O. Exploration by random network distillation. arXiv:1810.12894. 2018.
16. Bokade R, Jin X. Pytsc: a unified platform for multi-agent reinforcement learning in traffic signal control. Sensors. 2025;25(5):1302. doi:10.3390/s25051302.
17. Shi Y, Dong H, He CR, Chen Y, Song Z. Mixed vehicle platoon forming: a multi-agent reinforcement learning approach. IEEE Int Things J. 2025;12(11):16886–98. doi:10.1109/jiot.2025.3535732.
18. Zulfiqar H, Ul Haque HM, Tariq F, Khan RM. A survey on smart parking systems in urban cities. Concurr Comput. 2023;35(15):e6511. doi:10.1002/cpe.6511.
19. Yousaf S, Haque HMU, Atif M, Hashmi MA, Khalid A, Vinh PC. A context-aware multi-agent reasoning based intelligent assistive formalism. Internet Things. 2023;23(2):100857. doi:10.1016/j.iot.2023.100857.
20. Baldauf M, Dustdar S, Rosenberg F. A survey on context-aware systems. Int J Ad Hoc Ubiquit Comput. 2007;2(4):263–77.
21. Haque HMU, Khan SU. A context-aware reasoning framework for heterogeneous systems. In: 2018 International Conference on Advancements in Computational Sciences (ICACS). Lahore, Pakistan: IEEE; 2018. p. 1–9.
22. Haque HMU, Khan SU, Hussain I. Semantic knowledge transformation for context-aware heterogeneous formalisms. Int J Adv Comput Sci Appl. 2019;10(12): 664–70. doi:10.14569/IJACSA.2019.0101285.
23. Hwang GJ. Definition, framework and research issues of smart learning environments-a context-aware ubiquitous learning perspective. Smart Learn Environ. 2014;1(1):1–14. doi:10.1186/s40561-014-0004-5.
24. Augusto JC. Contexts and context-awareness revisited from an intelligent environments perspective. Appl Artif Intell. 2022;36(1):2008644. doi:10.1080/08839514.2021.2008644.
25. Eker S, Meseguer J, Sridharanarayanan A. The Maude LTL model checker and its implementation. In: Model Checking Software: 10th International SPIN Workshop; 2003 May 9–10; Portland, OR, USA: Springer; 2003. p. 230–4.
26. Reisig W. Understanding petri nets: modeling techniques, analysis methods, case studies. 1st ed. Berlin/Heidelberg, Germany: Springer; 2013.
27. Al-Bataineh O, Reynolds M, Rosenblum D. A comparative study of decision diagrams for real-time model checking. In: Model Checking Software: 25th International Symposium, SPIN 2018; 2018 Jun 20–22; Malaga, Spain: Springer; 2018. p. 216–34.
28. Kunnappilly A, Backeman P, Seceleanu C. From UML modeling to UPPAAL model checking of 5G dynamic service orchestration. In: Proceedings of the 7th Conference on the Engineering of Computer Based Systems (ECBS). Novi Sad, Serbia; 2021. p. 1–10.
29. Tiwari S, Iyer K, Enoiu EP. Combining model-based testing and automated analysis of behavioural models using GraphWalker and UPPAAL. In: 2022 29th Asia-Pacific Software Engineering Conference (APSEC). Tokyo, Japan: IEEE; 2022. p. 452–6.
30. Kim JH, Larsen KG, Nielsen B, Mikučionis M, Olsen P. Formal analysis and testing of real-time automotive systems using UPPAAL tools. In: Cimatti A, Sirjani M, editors. Formal Methods for Industrial Critical Systems: 20th International Workshop, FMICS 2015; 2015 Jun 22–23; Oslo, Norway: Springer International Publishing; 2015. p. 47–61.
31. Yang Z, Qiu Z, Zhou Y, Huang Z, Bodeveix JP, Filali M. C2AADL Reverse: a model-driven reverse engineering approach to development and verification of safety-critical software. J Syst Archit. 2021;118(3):102202. doi:10.1016/j.sysarc.2021.102202.
32. Vogel T, Carwehl M, Rodrigues GN, Grunske L. A property specification pattern catalog for real-time system verification with UPPAAL. Inf Softw Tech. 2023;154(1):107100. doi:10.1016/j.infsof.2022.107100.