A New Real-Time Algorithm to Extend DL Assertional Formalism to Represent and Deduce Entities in Robotic Soccer

Saminda Abeyruwan and Ubbo Visser

University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146 USA
{saminda,visser}@cs.miami.edu

Abstract. Creating, maintaining, and deducing accurate world knowledge in a dynamic, complex, adversarial, and stochastic environment such as the RoboCup environment is a demanding task. Knowledge should be represented in real-time (i.e., within ms) and deduction from knowledge should be inferred within the same time constraints. We propose an extended assertional formalism for an expressive $\mathcal{SROIQ}(\mathcal{D})$ Description Logic to represent asserted entities in a lattice structure. This structure can represent temporal-like information. Since the computational complexity of the classes of description logic increases with its expressivity, the problem demands either a restriction in the expressivity or an empirical upper bound on the maximum number of axioms in the knowledge base. We assume that the terminological/relational knowledge changes significantly slower than the assertional knowledge. Henceforth, using a fixed terminological and relational formalisms and the proposed lattice structure, we empirically bound the size of the knowledge bases to find the best trade-off in order to achieve deduction capabilities of an existing description logic reasoner in real-time. The queries deduce instances using the equivalent class expressions defined in the terminology. We have conducted all our experiments in the RoboCup 3D Soccer Simulation League environment and provide justifications of the usefulness of the proposed assertional extension. We show the feasibility of our new approach under real-time constraints and conclude that a modified FaCT++ reasoner empirically outperforms other reasoners within the given class of complexity.

Keywords: $\mathcal{SROIQ}(\mathcal{D})$ Logic, Symbol Grounding, RoboCup agents

1 Introduction

The OWL 2 Web Ontology Language[1], recommended by the World Wide Web Consortium (W3C) as part of the existing "Semantic Web" technologies, provides an explicit specification of a conceptualization that allows computers to intelligently search, combine, and process "data" (e.g., visual percepts, sensorimotor streams) on the basis of its meaning, i.e., the semantics. Therefore, similar

[1] http://www.w3.org/TR/owl2-overview/

to humans, computers can interpret data and deduce conclusion from them in its day-to-day operations. The conceptualization provides an abstract, simplified view of the world being modeled [5], the ability to use reasoning subsystems to draw meaningful conclusions from these models, and to exchange complex information or conclusions among multi-agent systems unambiguously. Complex robotic systems such as soccer playing robots in RoboCup [8] environments or self-driving cars (e.g., [18]) need substantial awareness of its surroundings. There is usually a substantial gap between the information that a robotic system actually collects via its modeling mechanisms and the high-level knowledge that could be used in order to obtain the appropriate decisions. Generally, high-level knowledge varies at a slower time scale than the modeling information, and most of the systems are bound to a faster duty cycle. We have investigated an ontological methodology to reduce this gap, ground information with respect to a domain of discourse, and reason in real-time.

The OWL 2 specification is based on $\mathcal{SROIQ}(\mathcal{D})$ Description Logic [7]. It is a less expressive, but highly structured, decidable fragment of the first-order predicate logic. Efficient reasoning engines exist that use DL constructs to infer about the domain of discourse. Though DL is decidable, its worst-case complexity is exponential, which demands upper bounds on the size of the knowledge base. It is common that OWL 2 ontologies are reasoned off-line and use the deduced axioms later in the process to scale for practical problems [11]. Even though DL has exponential worst case complexity, it provides constructs that can be used in real-time robotic systems to model data, derive conclusions from them, and exchange the now semantically grounded data among similar robotic systems. Here, we empirically investigate the ability to use DL in a real-time system setting. The proposed method uses a fixed terminological (TBox) and a fixed relational (RBox) formalism, and provides the justification of using "an extended assertional formalism (ABox)" to represent modeling information in a lattice structure, that has temporal-like structures, without explicitly adding new constructs to $\mathcal{SROIQ}(\mathcal{D})$ DL. This gives us the opportunity to use existing DL reasoners in a real-time setting. Since the general reasoning problem is N2ExpTime-complete, we maintain upper bounds on the number of axioms in each formalism, and empirically study the behavior of the extended ABox.

The RoboCup 3D Soccer Simulation environment provides a dynamic, real-time, complex, adversarial, and stochastic multi-agent environment for simulated agents. The agents formalize their goals in two layers: (1) the physical layers – controls related to walking, kicking, and so forth are conducted; and (2) the decision layers – high-level actions are taken to emerge behaviors. Our proposed method resides in the decision layer to assert modeling information and deduce soccer domain specifications. We have conducted all our experiments in the RoboCup 3D Soccer Simulation League Server[2]. Since, every simulation cycle is limited to 20 ms, we consider the upper-bound of the real-time reasoning within 5 ms, 10 ms, or 15 ms. We also consider and discuss situation in which multiple duty cycles, e.g., five cycles amounts to 100 ms, can be used with a

[2] http://simspark.sourceforge.net/wiki/

threading architecture to harness the idle processing time of the CPU. The fixed TBox contains class expressions to deduce individuals. An example would be the definition of a pass between two players or intercept a moving ball and so forth. We can compose queries to the system and use several heuristics to control the axiom count. The heuristics are activated based on pre-defined criteria such as active region surrounding the ball.

DL provides an appropriate trade-off between expressivity and scalability in practice (the reader is referred to [2,3] a comprehensive discussion on DL syntax[3], semantics, and model construction). In the proposed extension, the TBox and RBox is stationary, while the ABox changes over time. Therefore, complexity is dominated by the data complexity, which is NP-hard for $\mathcal{SROIQ}(\mathcal{D})$ DL ABoxes and N2ExpTime-complete for the combined TBox, RBox, and ABox. Thus, the real-time systems need an upper bound for the size of the ABox, while retaining as much as logical consequences as possible. Modern $\mathcal{SROIQ}(\mathcal{D})$ DL reasoners such as the (1) tableau-based FaCT++ [17] and Pellet [15] reasoners; and the (2) hyper-tableau HermiT [14] reasoner, use intelligent heuristics and optimization methods to perform inferencing as efficiently as possible. We investigate the real-time performances of tableau-based reasoners with respect to the proposed ABox extension.

2 Related Work

In AI, an ontology defines a formal specification of a conceptualization [5]. The conceptualization is defined using concepts, individuals, and relations among them. The formal specification allows agents in a multi-agent system to share information, and it provides a base to agents to act rationally to achieve common goals. The knowledge an agent possesses has the distinct feature of time dependence. But instead of committing to a temporal architecture, we are extending an ABox to a variable lattice structure that captures temporal-like information within the constructs given in DL. Therefore, we explicitly fixed the conceptualization encoded in the TBox and RBox, and change the ABox conceptualization. Similar to our approach, the $\mathcal{TL}$-$\mathcal{ALCF}$ DL extends static $\mathcal{ALCF}$ to represent interval-based temporal networks using Allen's interval-based temporal logic [9]. Our approach differs from this work in that we use $\mathcal{SROIQ}(\mathcal{D})$ DL and we encode the temporal-like information (only in the ABox) in a lattice structure that captures the dynamics of the changing knowledge. Therefore, we can directly use existing $\mathcal{SROIQ}(\mathcal{D})$ DL reasoners without substantial modifications. OWL-Time [6] allows representing temporal concepts and temporal relations in $\mathcal{SHOIN}(\mathcal{D})$ DL to represent new languages such as tOWL [10] to conceptualize concrete-domains. Our work significantly differs from these approaches as we directly represent the temporal-like assertions in a lattice structure and constrain the size of the ABox to support real-time requirements.

Allen's temporal interval algebra [1] captures the ability to represent intervals and temporal properties, and their evolution over those intervals. There are

[3] As a convention, we indicate entities in a conceptualization using sans serif letters (e.g., ∀hasParticipant.Thing, ∃hasID.nonNegativeInteger[>0], PassBall, HoldBall).

many instances where these constructs are presented in OWL DL ontologies (c.f., [12]) and we use an approach similar to that of Open Biological and Biomedical Ontologies Relation Ontology (OBO RO) [16] to represent temporal-like constructs within the ABox lattice structure.

OWL DL provides resources to represent entities in Semantic Web ontologies. These ontologies are large in nature (T/R/A/Box) and the main focus of many of the research approaches is to investigate: (1) the inference characteristics in expressivity, correctness, worst-case computational complexities of DL languages, incremental classification, rules, justification abilities, and large ABox reasoning; and (2) empirical performance indicators with respect to classification, satisfiability, subsumption, consistency, performance, and heap space and time [4]. These ontologies generally require minutes or hours to finish the reasoning tasks, while we consider the tasks that finish within a few milliseconds (e.g.,$\sim$10 ms), yet using all of the functionalities of the reasoner. This is a conflicting objective that needs compromises in different degrees.

A perdurantist (four-dimensionalist) approach has been introduced in [20] to represent entities that change information over time. Instead of depending on time directly, we have used the concepts of continuants and occurrents to represent entities on our domain of discourse. A continuant represents an entity that exists in whole at any time in which it exists at all, and persists through time while maintaining its identity. It has no temporal parts. An example would be the team of an agent. An occurrent is an entity that has temporal parts, and if it occurs, unfolds or develops through time [16]. An example would be the orientation, and two-dimensional location of an agent. Our method uses a combination of continuant and occurrent concepts to create assertions in the extended ABox.

An approach presented in [13] recognizes and predicts spatio-temporal patterns within games of the RoboCup 3D Soccer Simulation League. The method recognizes situations in real-time, and has the the ability to learn and predict the opponent behavior. Recognition, learning, and prediction is performed using Bayesian Networks, and the method requires on average $\sim$40 ms to compute inferences. The work most closely related to our work is presented in [19]. This method introduces a knowledge processing pipeline to detect complex events and action sequences as a spatio-temporal pattern sequence generated from qualitative scene descriptions. The method has been tested under tournament conditions with 5 Hz resulting in precise and also incomplete perceptions.

3 DL Assertional Formalism Extension

In this section, we present the syntax and semantics of the assertional formalism (ABox) extension to $\mathcal{SROIQ}(\mathcal{D})$ DL to represent entities in RoboCup 3D soccer simulation league. Firstly, we provide the definition, secondly, we describe the extension with respect an illustrative examples, thirdly, we describe a few real world examples from our knowledge base, and finally, we describe the extended ABox algorithm. Our extended ABox definition goes as follows:

Definition 1. ***(ABox Extension)*** *Given a fixed TBox and an RBox as defined in section 1, the extended ABox is defined as follows*[4]*:*

(1) There exists sampling points, $\mathbf{t_i} \in \mathbb{Z}_{>0}$*, such that, when pre-defined criteria are matched, a set of individual assertions are generated.*
(2) These assertions are of the form $[C(a_{\mathbf{t_i}})]_{\mathbf{t_j}}$ *for class expressions, and*
(3) $[R(a_{\mathbf{t_i}}, b_{\mathbf{t_j}})]_{\mathbf{t_j}}$*,* $\mathbf{t_i} \leq \mathbf{t_j}$ *for relations with given individuals* $a_{\mathbf{t_i}}$ *and* $b_{\mathbf{t_j}}$ *at the sampling point* $\mathbf{t_j}$*.*
(4) The individual assertions are realized with a $\mathsf{timeToLive} \in \mathbb{Z}_{\geq 0}$ *data type property, and they will be active for* $\mathsf{timeToLive} > 0$*.*
(5) The assertions are active for maximum sampling points of $\mathsf{latticeLength} \in \mathbb{Z}_{>0}$*, and they are first created with* $\mathsf{timeToLive} = \mathsf{latticeLength}$*.*
(6) At each sampling points, the timeToLive *data value of all individuals except the individuals with* $\mathsf{timeToLive} \neq \mathsf{latticeLength}$ *is decremented by one, and the assertions are purged when* $\mathsf{timeToLive} = 0$*.*
(7) Lattice structure query expression $[C(\mathsf{refinement})]$ *for an equivalent class expression* C *and an optional user defined refinement for which the individuals of* C *should bind to.*

The semantics of the Definition 1 is given by same constructs used in section 1. The extended ABox does not include additional constructs, yet provides an efficient framework to manage the number of asserted axioms. Each individual in the extended ABox is annotated with timeToLive data property. The individuals are generated with $\mathsf{timeToLive} = \mathsf{latticeLength}$, and they are purged when $\mathsf{timeToLive} = 0$.

3.1 An Illustrative Example

Figure 1 (a) shows an illustrative example of an extended ABox with the lattice structure with latticeLength four. In this example, time increases from left-to-right. The sampling points are $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$, and $\mathbf{t}_4$ such that, $\mathbf{t}_1 < \mathbf{t}_2 < \mathbf{t}_3 < \mathbf{t}_4$, and $\mathbf{t_i} \in \mathbb{Z}_{\geq 0}, \forall_{\mathbf{i}} \in \mathbb{Z}_{\geq 0}$. The symbol "●" shows an individual in the extended ABox (we will call the extended ABox as ABox at this point forward, and distinguishing the difference, if ambiguity occurs), and the Internationalized Resource Identifier (IRI) is shown to the right. Let's assume that before the sampling point $\mathbf{t}_1$, the ABox is empty. Let's assume that at $\mathbf{t}_1$ four individuals, **1**, **2**, **3**, and **4**, are added to the ABox. Therefore, $\text{ABox}_{\mathbf{t}_1}$ contains these four individuals. If they are asserted with types, they will be of the form $C(1)_{\mathbf{t}_1}, \ldots$ for some class expressions in TBox. These individuals are also asserted with the timeToLive concrete property with value four. It means that the individuals, that are created at this sampling point, will be lasted for latticeLength - 1 sampling points in the future. In this example, they will last for three more sampling points. All individuals in $\text{ABox}_{\mathbf{t}_1}$ will have the same timeToLive value. In addition, we also add other abstract and concrete properties to the individuals in $\text{ABox}_{\mathbf{t}_1}$ that match any pre-defined criteria. All these assertions are represented in a vertical line at $\mathbf{t}_1$.

[4] $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, $\mathbb{Z}_{>0} = \{n \in \mathbb{Z}; n > 0\}$, and $\mathbb{Z}_{\geq 0} = \{n \in \mathbb{Z}; n \geq 0\}$.

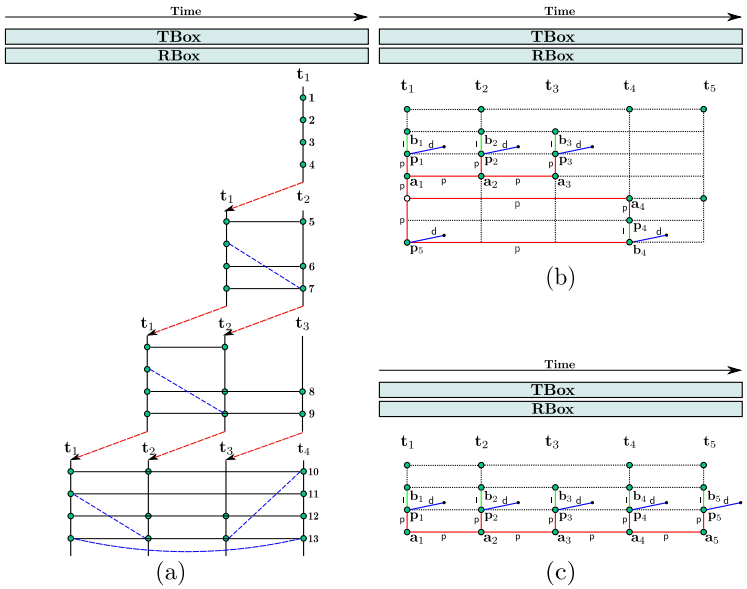

Fig. 1: (a) An extended ABox with the lattice structure, (b) An extended ABox that matches an instance of the equivalent class expression PassBall, and (c) An extended ABox that matches an instance of the equivalent class expression HoldBall.

3.2 Real World Examples

In the next sampling point, $\mathbf{t}_2$, we create $\text{ABox}_{\mathbf{t}_2}$ and add this to the extended ABox. At this point, all the timeToLive concrete properties in the $\text{ABox}_{\mathbf{t}_1}$ individuals are decremented by one. Let's assume that the individuals **5**, **6**, and **7** belong to $\text{ABox}_{\mathbf{t}_2}$. The timeToLive value is set to latticeLength. The horizontal lines show all existing abstract relations between the individuals from $\text{ABox}_{\mathbf{t}_1}$ to $\text{ABox}_{\mathbf{t}_2}$. At sampling point $\mathbf{t}_2$, a situation could occur that there exists some individuals that may not have a corresponding individual from the previous ABox. e.g., individual **2** does not have a corresponding individual from $\text{ABox}_{\mathbf{t}_2}$. The abstract properties are from the RBox, and they could be of the form atomic roles or generalized role inclusion axioms. In addition, individuals in $\text{ABox}_{\mathbf{t}_1}$ and $\text{ABox}_{\mathbf{t}_2}$ may add additional abstract roles as shown from the dashed line in Figure 1 (a), hence, initiating a lattice structure.

We create an $\text{ABox}_{\mathbf{t}_3}$ at the next sampling point, $\mathbf{t}_3$, with individuals $\mathbf{t}_8$ and $\mathbf{t}_9$, and their timeToLive value is set to four. The timeToLive values of the individuals in $\text{ABox}_{\mathbf{t}_1}$ and $\text{ABox}_{\mathbf{t}_2}$ are decremented by one. The abstract relations that exist among the individuals in $\text{ABox}_{\mathbf{t}_1}$ and $\text{ABox}_{\mathbf{t}_2}$ do not change, while new abstract relations are formed among individuals in $\text{ABox}_{\mathbf{t}_2}$ and $\text{ABox}_{\mathbf{t}_3}$. Say there are no such abstract relations formed among the individuals. The same procedure continues at the sampling point $\mathbf{t}_4$. In addition, individuals could participate in longer relations. The individual **4** in $\text{ABox}_{\mathbf{t}_1}$ and the individual **13** in $\text{ABox}_{\mathbf{t}_4}$ have abstract relationships in the extended ABox in this example (cf. Figure 1

(a) bottom). At sampling $\mathbf{t}_5$, the timeToLive value of the individuals in $\text{ABox}_{\mathbf{t}_1}$ becomes zero, and those individuals are purged from the extended ABox with all related axioms. At $\mathbf{t}_5$, the $\text{ABox}_{\mathbf{t}_1}$ is purged and a new $\text{ABox}_{\mathbf{t}_5}$ will be created. Henceforth, the process continues as mentioned above.

Our knowledge base, K, consists of a fixed TBox, RBox, and an extended ABox that is created according to Definition 1. The consistency of the knowledge base is checked with an $\mathcal{SROIQ}(\mathcal{D})$ DL reasoner. We start with a satisfiable knowledge base with a TBox and an RBox, and the extended ABox changes the knowledge as the dynamics of the system changes. It is the responsibility of the reasoner to decide the satisfiability of the knowledge base by adding $\text{ABox}_{\mathbf{t_i}}$, $\mathbf{i} = 1, \ldots$, to the extended ABox. If the knowledge base is unsatisfiable, then the $\text{ABox}_{\mathbf{t_i}}$ will be removed from the knowledge base. We query for assertions using equivalent class expressions and user defined refinements.

In this subsection we provide a few examples from our knowledge base to understand the process. We have developed an ontology to represent entities in the RoboCup 3D Soccer Simulation environment based on prior knowledge.

(1) The equivalent class, $\mathsf{Object} \equiv \exists \mathsf{timeToLive.nonNegativeInteger}$, defines an object in our domain of discourse as any entity that has a positive time-to-live value.
(2) We define an agent using equivalent class expression;
$\mathsf{Agent} \equiv \mathsf{Object} \sqcap \exists \mathsf{hasID.nonNegativeInteger}[>0]$, and $\mathsf{hasID} \in \mathbb{Z}_{>0}$, as any object in the domain of discourse that has a strictly positive identification number.
(3) Therefore, we define a home agent and an opponent agent; $\mathsf{HomeAgent} \sqsubseteq \mathsf{Agent}$ and $\mathsf{OpponentAgent} \sqsubseteq \mathsf{Agent}$. The agents are disjoint: $\mathsf{HomeAgent} \sqcap \mathsf{OpponentAgent} \sqsubseteq \bot$.
(4) Most entities in the RoboCup 3D soccer simulation have poses (rotation and two dimensional position on the field). We define a pose: $\mathsf{Pose2D} \equiv (\exists \mathsf{rotation.int} \sqcap \exists \mathsf{xcoord.int} \sqcap \exists \mathsf{ycoord.int})$, with $\mathsf{rotation}, \mathsf{xcoord}, \mathsf{ycoord} \in \mathbb{Z}$. Any *thing* in the domain of discourse which has an orientation and (x, y) coordinates in a two-dimensional plane. The distances are annotated with millimeters (mm), while the angles in radians are subjected to the mapping function $f : [-\pi, \pi] \mapsto [0, 2048]$.
(5) Ball GCI axioms are: $\mathsf{Ball} \sqsubseteq \exists \mathsf{locatedIn.Pose2D}$ and $\mathsf{Ball} \sqsubseteq \mathsf{Object}$.
(6) Using axioms 1, 2, 3, 4, and 5, we can query for all objects potentially close to the ball from the extended ABox as $\mathsf{ObjectsWithBallContact} \equiv \mathsf{Object} \sqcap \exists \mathsf{hasParticipant.Ball} \sqcap \exists \mathsf{hasParticipant.}(\mathsf{Participant} \sqcap \exists \mathsf{distance.int}[<500])$, and $\mathsf{distance} \in \mathbb{Z}$, any object in the domain of discourse which has a ball participant and the ball participant is *close* to the object.
hasParticipant is a transitive object property. We use N-ary relationship representations[5] to state the connection between agents, participants, and soccer ball representations. We use a distance threshold, which is given as prior knowledge from the domain experts. This class expression uses a 500 mm distance threshold to quantify the closeness property.

[5] http://www.w3.org/TR/swbp-n-aryRelations/

(7) Let's define a class expression for the HoldBall skill, which queries for agents that control the ball. We define a refinement such that the individuals should have different timeToLive values and there must exists at at least five individuals in the class expression. We have defined the equivalent class expression: HoldBall $\equiv$ Agent $\sqcap$ $\exists$hasParticipant.(BallParticipant $\sqcap$ $\exists$distance.int[<150]), demands classification of agents that have some ball participants within *a close proximity*. [HoldBall(refinement)] query expression uses several ABox parameter choices and the prior knowledge of duration in which an agent should be in close proximity to the ball. An instance of the ABox that matches the query expression is given in Figure 1 (c). The refinements are executed after the deduction process is finished.

Our assumptions are: (A1) latticeLength is five; (A2) user define refinements; and (A3) given sampling points $\mathbf{t_i}$, $\mathbf{i} = 1, \ldots, 5$. The individuals are: (I1) $\mathbf{b_j}$, $\mathbf{j} = 1, \ldots, 5$, represents an instance of class expression Ball; (I2) $\mathbf{p_j}$, $\mathbf{j} = 1, \ldots, 5$, represents an instance of class expression Participant; and (I3) $\mathbf{a_j}$, $\mathbf{j} = 1, \ldots, 5$, represents an instance of class expression Agent, and it is a realization of the same agent over the extended ABox sampling points. The relations are: (R1) $\mathbf{p}$ and $\mathbf{l}$ represent the transitive abstract properties hasParticipant and locatedIn respectively; (R2) $\mathbf{d}$ represents the concrete property distance; and (R3) there exists diverse variety of relations among individuals that does not influence the given outcome. Using these criteria, and if the ball is within a close proximity (e.g., $\leq$500 mm), a DL reasoner can deduce that $\mathbf{a_1}$ is the only instance of the class HoldBall within the parameters of the given ABox. Using our TBox, we can determine the type of the agent, and using RBox and ABox we can determine the identification and other assertions. If the definition of the class expression HoldBall needs to be more specific, such as whether the agent needs to remain stationary or stay as far away from the opponent as possible, these conditions should be explicitly stated in the class expression. These additional constraints increase the number of axioms, and there will be a trade-off between computational complexity and the number of queries we can define.

The interpretation of the statement "in close proximity" is based on prior knowledge and design parameters. We can define multiple subclasses of HoldBall with different refinements that meets our criteria. According to the query expression, the result set contains either home agents or opponent agents. In addition, a DL reasoner deduces the fact that HoldBall $\sqsubseteq$ WithBallContact.

(8) PassBall equivalent class expression queries for home agents that pass the ball to its teammates. It is defined as:
PassBall $\equiv$ HomeAgent $\sqcap$ $\exists$hasParticipant.(BallParticipant $\sqcap$ $\exists$distance.int[<100]) $\sqcap$ $\exists$hasParticipant.(HomeAgent $\sqcap$ $\exists$hasParticipant.(BallParticipant $\sqcap$ $\exists$distance.int[<150]) $\sqcap$ $\exists$ hasParticipant.(BallParticipant $\sqcap$ $\exists$distance.int[>1000]).

PassBall class subsumes individuals in the extended ABox close to the ball, and within the same ABox, locate another agent of the same team that is close to the ball. In order to conduct a pass, the ball is required to be relatively close to an agent (e.g., <150 mm) and the ball should travel some

distance (e.g., >1000 mm), and the receiving ball should be also close to an agent.

There are some limitations in the given definition. First, we neither can write the requirement that the passing agent and the receiving agent should be different in the class expression nor a refinement for PassBall using DL expressivity. Second, there could be situations where external forces or disturbances from another agent or environment could cause the ball to move from the close-by-agent to another agent in the same team. PassBall definition does not capture these special cases. In order to capture the degree of PassBall confidence, extra systems with probabilistic interpretation must be used.

In RoboCup 3D Soccer Simulation, there are no constructs to define a *kick* directly. Therefore, we have defined the pass without explicitly committing to a notion of a kick. Similarly, we define a pass ball behavior to opponent agents. Hence, we generalize PassBall class expression to deduce either a home or an opponent agent as the passing agent using logical union conjunction. An instance of the ABox that matches the class expression is given in Figure 1 (b). The symbol ∘ in Figure 1 (b) represents a *black node* that connects relevant individuals. We have used the same assumption and parameters that are defined in Example 7. We have made a slight modification to the individuals such that, the sets $\{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}\}$ and $\{\mathbf{a_4}\}$ are disjoint realizations of physically different agents (we have used the agent identification number to make this distinction). A DL reasoner can deduce that $\mathbf{a_1}$ is an instance of the class PassBall for this particular example.

4 Experiments

In order to establish a baseline, we have compared the distributions of deduction times of four reasoners[6]. We have modified the FaCT++ implementation to use hash tables instead of binary tree implementations, when it is necessary, and slightly changed the caching mechanisms. This modification has positive effects on TBox and RBox reasoning. We expect improvements in ABox reasoning, when there are individuals with many data type axioms. We have used the modified version of FaCT++ in baseline establishment and it is labeled as FaCT++ (Modified). Table 1 shows the average reasoning times and the 95% confidence intervals. We have used 80−logical axioms (62−entities) from the ontology (TBox and RBox) for this experiment. This calculation uses 500 independent trials from each reasoner. We observe that the modified FaCT++ DL reasoner shows a statistically better performance over the other reasoners[7]. Our modifications to FaCT++ DL reasoner have improved 22% over the original FaCT++ implementation. Henceforth, we have selected the modified FaCT++

[6] (1) HermiT (1.3.8): http://hermit-reasoner.com/; (2) Pellet (2.3.1): http://clarkparsia.com/pellet/; and (3) FaCT++ (1.6.2): http://code.google.com/p/factplusplus/.

[7] We have used 2.2GHz Core−i7 (4GB) laptop for all experiments. Authors can provide the modified FaCT++ implementation and the ontology upon request.

DL reasoner to be used with the simulated agents, and we have used the reasoners in non-incremental mode. The baseline establishment has set the empirical

Table 1: Average reasoning times (95% confidence).

Reasoner	Time in milliseconds
FaCT++ (Modified)	$\boldsymbol{1.092 \pm 0.002}$
FaCT++ (1.6.2)	1.403 ± 0.009
HermiT (1.3.8)	2.289 ± 2.654
Pellet (2.3.1)	11.634 ± 2.465

lower bound to zero individuals and ∼150 axioms. This corresponds to an empty ABox. In order to estimate the empirical upper bound, we have conducted the following experiment: Firstly, we have added the examples mentioned in subsection 3.2 to the ontology. Secondly, we have created a hypothetical world model for an agent. This world model changes its believes randomly about teammates, opponents, and ball poses. We have used a uniform distribution to sample entities in the belief model. All poses are randomly generated inside the simulated soccer field. Thirdly, we have chosen a value from $[2, 20]$ for latticeLength to generate axioms. Finally, we ran 10 sets of 100 sampling points for every latticeLength setting. It corresponds to a set of $19,000$ data points. Figures 2 and 3 show the concluding results (the error bars show one-standard deviation of bins of ten).

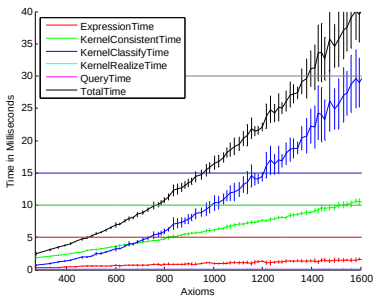

Fig. 2: Establishment of empirical upper bound.

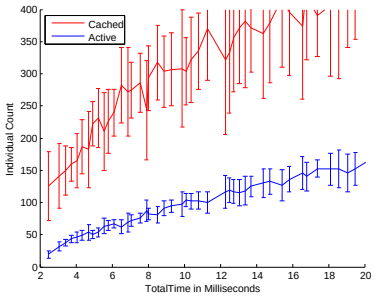

Fig. 3: Relationship between reasoning time and number of individuals.

In Figure 2, the KernelConsistentTime plot shows that the consistency checking scales linearly with the number of axioms. The classification (KernelClassifyTime plot) produces an exponential growth. Its contribution immensely affects the reasoning time and the empirical upper bound. It also affects the variability of timing. After $1,000$ axioms, there is significant variance, which is undesirable for real-time systems. The realization (KernelRealizeTime plot) and expression (ExpressionTime plot) times are relatively negligible. The expression time exhibits our operations for refinements and to track the evolution of the extended ABox. It shows linear time complexity and it is justifiable for real-time operations.

Figure 3 shows the number of individuals (active and cached) with respect to total deduction time. The extended ABox algorithm uses the caching mechanism to reuse individuals, which improves the expression time. We can conclude from these figures that in order to operate within 5 ms, we can keep ∼500 axioms ∼50 individuals. Similarly, we can use ∼800 axioms and ∼100 individuals for 10 ms, and ∼1000 axioms and ∼150 individuals for 15 ms. This suggests that with a given latticeLength, as long as we bound the size of the axioms and individuals, our system can operate in real-time. We have investigated the requirement whether DL-based constructs are suitable for real-time operations. Figures 2 and 3 emphasize the facts that: (1) there is an upper bound where DL boxes are effective to deduce conclusions in real-time; and (2) the total deduction time exponentially increase with the number of axioms. Therefore, it suggests that we can use multiple cycles (e.g., 100 ms corresponds to five cycles in our domain) to execute the reasoning process. This requires an agent equipped with a low-priority thread that uses extra clock cycles of the processing units.

In order to bound the size of the axioms and individuals, we have developed explicit heuristics for which the agents should react to. Agents keep assertions about Ball in all sampling points to produce concise decisions. All agents maintain assertions about themselves and w.r.t. the Ball. We have considered only two players close to the ball from each team to participate with Ball individuals. Each agent keeps track of two close players from each team. The agents participate with other agents through Participant objects. This provides a clean and simple mechanism in which an agent could include Allen's temporal constructs to be used within the ABox. In our on-line setting, we ran 11 vs 11 games with the given heuristics. We set the latticeLength to five for this experiment. An agent tracks 49.18 ± 4.86 individuals and 529.44 ± 105.31 axioms in 6.57 ± 2.81 ms. Therefore, we can justify that our algorithm is real-time compatible on a RoboCup 3D simulated robot.

5 Conclusion

We presented a new approach of using an extended ABox structure to represent temporal-like information and deducing conclusions in real-time. Our approach has extended the $\mathcal{SROIQ}(\mathcal{D})$ DL ABox with a lattice structure and it provides flexibility to use existing DL reasoners. We have tested and validated our approach in an off-line and on-line settings for the RoboCup 3D soccer domain. The approach enables autonomous agents to successfully interpret its believes about the world. We have showed that the deduction complexity and the computation complexity produce a conflicting objective. Therefore, our approach has empirically bounded the size of DL boxes and modified the FaCT++ DL reasoner to be compatible with real-time operations. We intend to use our approach with incremental reasoning on a physical robot to model believes and interpret entities in uncertain environments in the near future.

References

1. Allen, J.F., Ferguson, G.: Actions and Events in Interval Temporal Logic. Journal of Logic and Computation 4(5), 531–579 (1994)

2. Baader, F., Nutt, W.: Basic Description Logics. In: Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.) Description Logic Handbook. pp. 43–95. Cambridge University Press (2003)
3. Brown, D., Chumakina, M., Corbett, G.G. (eds.): Canonical Morphology and Syntax. Oxford University Press, New York (2013)
4. Dentler, K., Cornet, R., ten Teije, A., de Keizer, N.: Comparison of Reasoners for Large Ontologies in the OWL 2 EL Profile. Semantic Web 2(2), 71–87 (April 2011)
5. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: Guarino, N., Poli, R. (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, Deventer, The Netherlands (1993)
6. Hobbs, J.R., Pan, F.: An Ontology of Time for the Semantic Web. ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing 3(1), 66–85 (March 2004)
7. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible SROIQ. In: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006). pp. 57–67. AAAI Press (June 2006)
8. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. In: Proceedings of the First International Conference on Autonomous Agents. pp. 340–347. AGENTS '97, ACM, New York, NY, USA (1997)
9. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal Description Logics: A Survey. In: Demri, S., Jensen, C.S. (eds.) TIME. pp. 3–14. IEEE Computer Society (2008)
10. Milea, V., Frasincar, F., Kaymak, U.: tOWL: A Temporal Web Ontology Language. IEEE Transactions on Systems, Man, and Cybernetics, Part B 42(1), 268–281 (2012)
11. Motik, B., Horrocks, I., Kim, S.M.: Delta-Reasoner: A Semantic Web Reasoner for an Intelligent Mobile Platform. In: Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab, S. (eds.) WWW (Companion Volume). pp. 63–72. ACM (2012)
12. Petnga, L., Austin, M.: Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems. In: Paredis, C.J.J., Bishop, C., Bodner, D.A. (eds.) CSER. Procedia Computer Science, vol. 16, pp. 403–412. Elsevier (2013)
13. Rachuy, C., Visser, U.: Behavior-Analysis and -Prediction for Agents in Real-Time and Dynamic Adversarial Environments. In: Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence. pp. 979–980. IOS Press, Amsterdam, The Netherlands, The Netherlands (2010)
14. Shearer, R., Motik, B., Horrocks, I.: HermiT: A Highly-Efficient OWL Reasoner. In: Ruttenberg, A., Sattler, U., Dolbear, C. (eds.) Proceedings of the 5th International Workshop on OWL: Experiences and Directions. Karlsruhe, Germany (October 26–27 2008)
15. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Web Semantics: Science,Services and Agents on the World Wide Web 5(2), 51–53 (June 2007)
16. Smith, B., Ceusters, W., Klagges, B., Kohler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A., Rosse, C.: Relations in Biomedical Ontologies. Genome Biology 6(5), R46+ (2005)
17. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR 2006). Lecture Notes in Artificial Intelligence, vol. 4130, pp. 292–297. Springer (2006)
18. Wang, C.C., Thorpe, C., Thrun, S.: Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Taipei, Taiwan (September 2003)
19. Warden, T., Lattner, A.D., Visser, U.: Real-Time Spatio-Temporal Analysis of Dynamic Scenes in 3D Soccer Simulation. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008: Robot Soccer World Cup XII, pp. 366–378. Springer-Verlag, Berlin, Heidelberg (2009)
20. Welty, C.A., Fikes, R.: A Reusable Ontology for Fluents in OWL. In: Bennett, B., Fellbaum, C. (eds.) Formal Ontology in Information Systems. Frontiers in Artificial Intel. and Apps., vol. 150, pp. 226–236. IOS (2006)