2015-07-16

Learnable Knowledge for Autonomous Agents

Saminda W. Abeyruwan
University of Miami, samindaa@gmail.com

UNIVERSITY OF MIAMI

LEARNABLE KNOWLEDGE FOR AUTONOMOUS AGENTS

By

Saminda Abeyruwan

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

August 2015

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

LEARNABLE KNOWLEDGE FOR AUTONOMOUS AGENTS

Saminda Abeyruwan

Approved:

Ubbo Visser, Ph.D. Associate Professor of Computer Science	Dilip Sarkar, Ph.D. Associate Professor of Computer Science
Geoff Sutcliffe, Ph.D. Professor of Computer Science	Stephan Schürer, Ph.D. Associate Professor of Molecular and Cellular Pharmacology
Vance Lemmon, Ph.D. Professor of Neurological Surgery	Dean of the Graduate School

ABEYRUWAN, SAMINDA (Ph.D., Computer Science)

Learnable Knowledge for Autonomous Agents (August 2015)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Ubbo Visser.
No. of pages in text. (152)

While computation power has increased and the statistical machine learning methods have made substantial advancement, many problems that would benefit from real-time interpretation have not exploited their combined strengths. For instance, the problem of gathering data from the environment and transforming it into knowledge as well as updating the knowledge as new data become available. Currently, with substantial expressivity and moderate computational cost, high-level languages or first-order predicate logic or model-based machine learning are used for *static* representation of knowledge, that is used for reasoning and inferring. In this dissertation, we address how an entity *dynamically gather* knowledge from environmental data and use that for *inferring evolving events* and *dynamically update* the current knowledge. We develop theoretical and empirical solutions using Description Logic representation and reasoning, and General Value Functions in Reinforcement Learning. The proposed solutions dynamically extract low-level knowledge from available data and update the high-level knowledge, which is used to predict the evolving future events. We show its applications in three real world domains: 1) RoboCup 3D Soccer Simulation environment, 2) High-throughput screening, and 3) Axon regeneration.

Acknowledgements

My dissertation is developed over time through the collaboration with several great scientists and research groups. First, I would like to express my sincere gratitude to my advisor and mentor Dr. Ubbo Visser. Without Ubbo's persistent guidance, support, advice, and freedom explore new ideas, this dissertation would have not been possible. Second, I owe so much to my informal co-advisor Dilip Sarkar, who has helped me for the past seven years in different aspects of my life, and who has considerably pushed me towards organizing this dissertation. Third, I am so fortunate to collaborate with BioAssay and RegenBase ontology research groups with Dr. Stephan Schürer, Prof. Vance Lemmon, and Prof. John Bixby. Our collaborations not only solved problems that have expanded my research horizons and analytical skills, but also financially supported me to venture all my endeavors. Fourth, I would like to thank my colleagues, Andreas Seekircher, Dr. Justin Stoecker, and Dr. Uma Vempati, whom I have collaborated with research publications and international competitions. Finally, I would like to thank my committee members and the chairmen of the Department of Computer Science, Prof. Geoff Sutcliffe, providing feedback on my dissertation and helping me to improve the presentation.

Above all, I am thankful for my parents and brothers. I am blessed to have their love, guidance, and support. And praise and thanks goes to my savior Jesus Christ for the many blessing bestowed upon me through strength, courage, wisdom, and hope throughout my life. Thank You!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge representation and reasoning is ubiquitous! It directly leads to intelligence and being able to use that knowledge to achieve personal goals, whether the entity in this question is a "human being" or an "artificial software program". Therefore, it is clear that knowledge representation and reasoning is central to intelligence. Though we have studied the exact interpretation of knowledge, the question remains whether it can be efficiently acquired and accurately used. The former leads to well known "knowledge acquisition bottleneck" [4], while the latter questions whether the important aspects can be searched within a reasonable amount of time with respect to a given domain.

The basic questions of this dissertation are: 1) What is knowledge? 2) Is it learnable? 3) What constituents keep the knowledge correct? 4) Where does an entity get the ground truth from to verify knowledge? and 5) Can the knowledge be used with a reasonable amount of time? Knowledge representation and reasoning comes in different forms and we have investigated the above basic questions with respect to Description Logic and forms of other knowledge representation formalisms.

1.1 Project Description and Scope

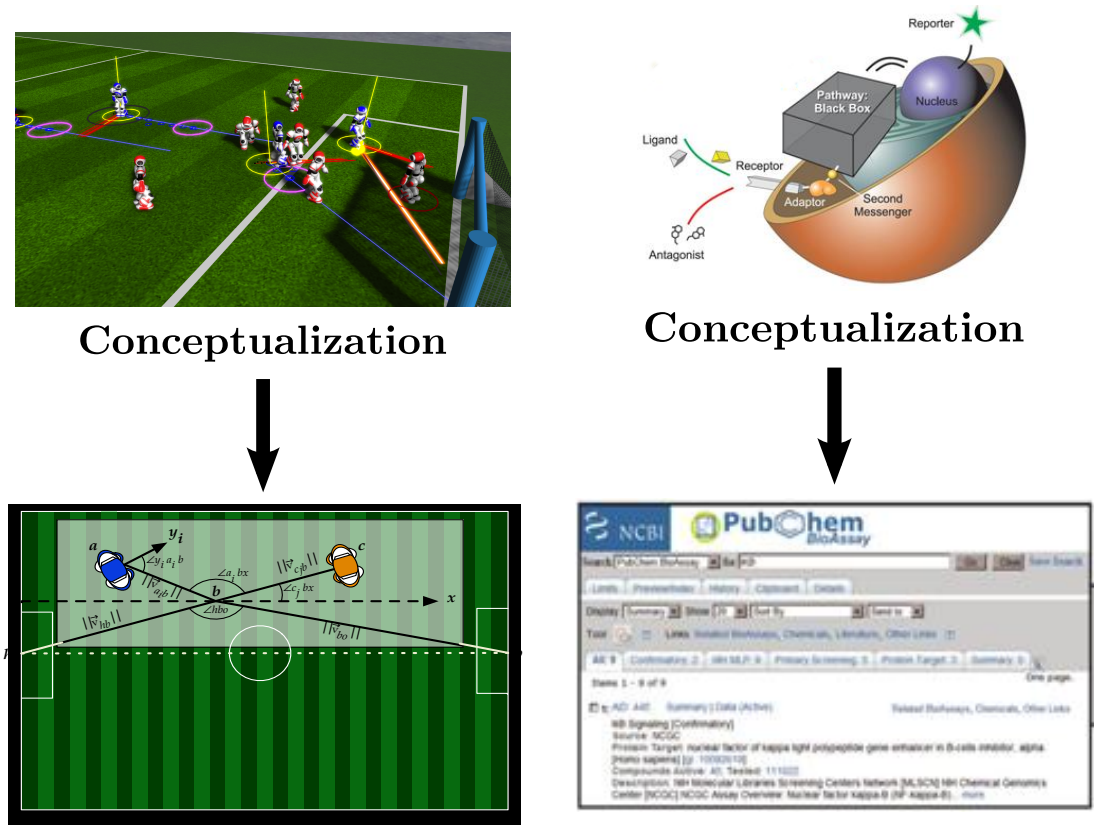
The dissertation is based on work that I have contributed to three different projects over the past five years:

1) **Autonomous agents**: describes a sub-domain of RoboCup, where entities (3D simulated robots and physical NAO humanoid robots) co-exist for the purpose of acting and reacting in dynamic, uncertain, real-time, and adversarial environment using soccer as a testbed.

2) **BioAssay ontology**: describes biological screening assays and their results including high-throughput screening data for the purpose of categorizing assays and data analysis.

3) **RegenBase ontology**: describes an information framework and knowledge base to facilitate research about nervous system regeneration.

Figure 1.1 shows the contemporary flow of knowledge for two illustrative domains of discourse. The micro-domain example (Figure 1.1(a)) shows an instance of simulated 3D soccer playing agents. In this setting, the autonomy of the agents are designed with respect to the conceptualization that the programmer processes. Similarly in the macro-domain (Figure 1.1(b)), the specification of assays or experiments are described in collection of corpora with structured on semi-structured formats. This form of flow can be considered as "public knowledge", in which: 1) human experts understand the knowledge and ensure that it matches their beliefs; 2) internal



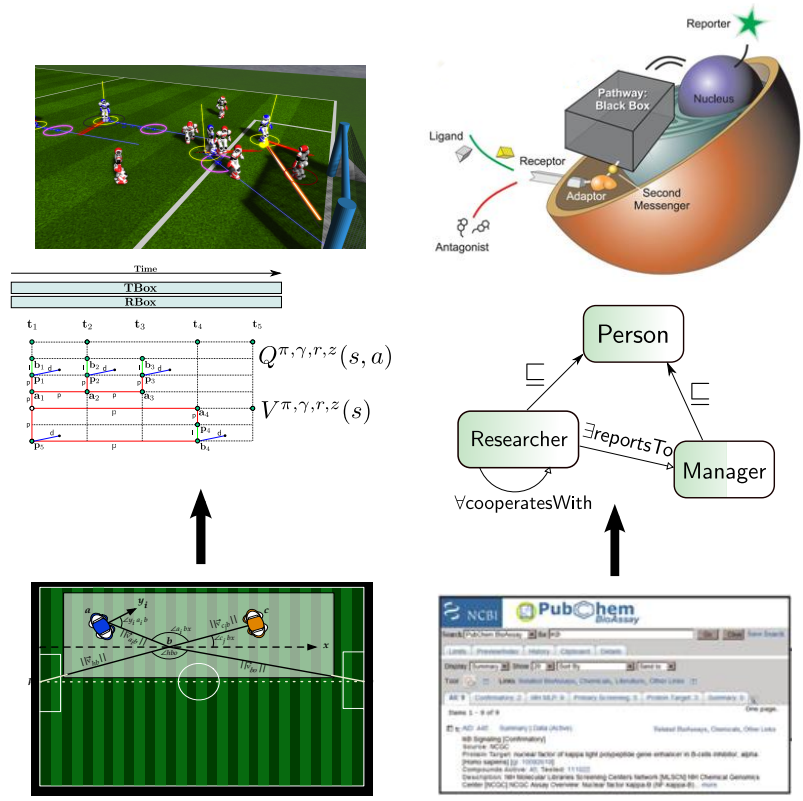
(a) Micro example domain.

(b) Macro example domain.

Figure 1.1: Example domains of discourse.

system checks that this knowledge coheres, and removes inconsistencies; and 3) it compares with external data in some way and changes it as needed to match the data. These are valid forms to maintain correct knowledge. In this dissertation, we are interested in the ability to infer "some aspect" of the conceptualization or dichotomy from the underlying data stream.

Figure 1.2 shows our point-of-view. Therefore, we contribute to solve problems presented in three domains with: 1) Extending $\mathcal{SROIQV}^{\mathcal{D}}$ Description Logic; 2) General Value Functions in Reinforcement Learning as an alternative knowledge representation formalism; and 3) Efficient implementation and framework design.



(a) Micro example domain. (b) Macro example domain.

Figure 1.2: Proposed solutions to example domains of discourse.

Therefore, the dissertation presents knowledge learning and reasoning as the unified theme to solve the problems presented in given three domains.

1.2 Contributions

The focus of this dissertation is to learn and infer knowledge in "real-time". The notion of real-time depends on the constrains of the domain. For a domain such as RoboCup 3D Soccer Simulation, the real-time constraints are in the range of milliseconds. This includes knowledge acquisition, modeling, reasoning, and querying times. Therefore, computational cost is important, while maintaining the satisfiability

guarantees. Hence, we have studied the real-time aspect of Description Logic and an alternative representation mechanism by which knowledge can be encoded to build scalable autonomous agents.

For larger domains such as BioAssay and RegenBase ontologies, space constraints, i.e., a constraint on the size of the physical memory (random-access memory) to reason the knowledge base with the given complexity of Description Logic, also compliment with time constraints. Here, the notion of real-time to reason may vary from minutes to hours. Therefore, these physical constraints need to be addressed in-order to fully utilize the powers of Description Logic we have used in our modeling. We have addressed the following four research topics in this dissertation:

1. **Real-time reasoning**: in this topic, we have investigated the problem of representing and deducing knowledge in real-time (i.e., within ms) given a set of constraints. We propose an extended assertion box for an expressive $\mathcal{SROIQV}^{\mathcal{D}}$ Description Logic to represent asserted entities in a lattice structure, which can naturally 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 have investigated these points and empirically validated our methods on the RoboCup 3D soccer simulation environment [5].

2. **Knowledge learning using General Value Functions**: in this topic, we have used recently introduced Off-Policy Gradient Descent algorithms in Reinforcement Learning to learn knowledge with its representation formalisms for dynamic role assignments. Subsets of agents have been used to identify the dynamics and the semantics for which the agents learn to maximize their performance measures, and to gather knowledge about different objectives, so that all agents participate effectively and efficiently within the group. We have used the RoboCup 3D soccer simulation environment as our test-bed to validate the methods presented in this topic [6].

3. **Representation of and reasoning with large-scale knowledge base**: in this topic, we have provided our design, development, and implementation of the first ontology to describe high-throughput screening experiments and screening results using Description Logic. Our ontology serves as a foundation for the standardization of high-throughput screening assays and data and as a semantic knowledge model. We have shown important examples and the advantages of this approach [7], [8], and [9].

4. **Ontology modularization for large-scale knowledge base**: in this topic, we have investigated one of the critical questions evolving ontologies: how can we provide a way to efficiently reuse and share among various research projects specific parts of our ontologies without violating the integrity of the ontology and without creating redundancies [10].

1.3 Overview of the Chapters

The remainder of this dissertation is organized as follows. Chapter 2 provides a set of preliminaries, in which the relevant theory about Description Logic and test domains are explained. In Chapter 3, real-time reasoning of sensorimotor data streams are introduced. Chapter 4 discusses an alternative perspective of knowledge representation and reasoning using the concepts from Reinforcement Learning. In Chapter 5, large scale ontology reasoning and representation are presented, while Chapter 6 discusses our ontology modularization methodology that scales for large ontologies. Finally, in Chapter 7, the dissertation is concluded with a summary and discusses promising future avenues.

Chapter 2

Preliminaries

The main contributions presented in the dissertation are based on real-time reasoning of knowledge bases of different modalities. We have used *Description Logic* as the main representation of the knowledge bases. In order to evaluate our main contributions, we have used *the RoboCup 3D Soccer Simulation Environment* as our testbed. This chapter provides a brief introduction to these technologies, which will be described succinctly in the latter chapters.

2.1 Description Logic (Web Ontology Language)

The Web Ontology Language (OWL 2) [11] 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" on the basis of its meaning, i.e., the semantics. Therefore, similar to humans, computers can interpret and deduce conclusions from data in its day-to-day operations. The conceptualization provides mechanisms to express complex ideas in simpler models, 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. The Description Logic (DL) contains a set of decidable constructs from the first-order predicate logic, and it is the corner stone for the development of the OWL 2 DL ontologies in knowledge representation [12]. The computational complexity of a given DL depends on the constructs that are used, and they are traditionally represented with different complexity classes.

Attribute Language with Complement ($\mathcal{ALC}$) provides the basic DL constructs with classes, roles, and individuals. The formal syntax of $\mathcal{ALC}$ is defined as follows. Let A be a named atomic class, and, without loss of generality, let R be an abstract role. The class expressions (concepts or concept expressions) C, D are recursively constructed by: $\mathsf{C}, \mathsf{D} \leftarrow \mathsf{A} \mid \top \mid \bot \mid \neg\mathsf{C} \mid \mathsf{C} \sqcap \mathsf{D} \mid \mathsf{C} \sqcup \mathsf{D} \mid \forall\mathsf{R}.\mathsf{C} \mid \exists\mathsf{R}.\mathsf{C}$, where, $\top$ is the top concept, $\bot$ is the bottom concept, the symbols for conjunction, disjunction, and negation are given by $\sqcap$, $\sqcup$, and $\neg$ respectively, and $\forall$ and $\exists$ represent the universal and existential quantifiers. $\mathcal{ALC}$ DL knowledge bases consist of two groups: (1) the TBox provides statements about the terminological knowledge; and (2) the ABox provides statements about the assertional knowledge about individuals. These statements are also known as axioms in description logic. For class expressions C and D, the TBox statements are of the form $\mathsf{C} \equiv \mathsf{D}$ or $\mathsf{C} \sqsubseteq \mathsf{D}$, where $\equiv$ denotes the equivalences among classes and $\sqsubseteq$ constructs the subsumption or general class inclusion (GCI) axioms. On the other hand, an ABox consists of axioms of the form $\mathsf{C}(\mathsf{a})$ and $\mathsf{R}(\mathsf{a},\mathsf{b})$, where R is a role, and, a, b are individuals. $\mathcal{ALC}$ DL has

been extended to $\mathcal{SROIQV}^{\mathcal{D}}$ DL with the following syntactic constructs: $\{\mathsf{a}\}$ | $\{x\}$ | $\exists \mathsf{R}.\mathsf{Self}$ | $\leq \mathsf{n}\mathsf{R}.\mathsf{C}$ | $\geq \mathsf{n}\mathsf{S}.\mathsf{C}$, where, concept $\{\mathsf{a}\}$ represents nominals, concept $\{x\}$ with x is a variable called nominal schemas, $\exists \mathsf{R}.\mathsf{Self}$ relates an individual to itself, and $\mathrm{n} \in \mathbb{Z}^+$ with $\leq \mathsf{n}\mathsf{R}.\mathsf{C}$ and $\geq \mathsf{n}\mathsf{S}.\mathsf{C}$ provide the qualified cardinality restrictions. $\mathcal{SROIQV}^{\mathcal{D}}$ DL introduces an RBox with general role inclusion axioms of the form $\mathsf{R}_1 \circ \ldots \circ \mathsf{R}_2 \sqsubseteq \mathsf{R}$, which provides the meaning that concatenation of $\mathsf{R}_1, \ldots, \mathsf{R}_2$ is a subrole of R. In addition, there are constructs to represent transitive, symmetric, asymmetric, reflexive, irreflexive, functional, inverse functional, and disjoint roles and concepts. It is to be noted that roles can either be abstract or concrete.

The interpretation of DL is given by the direct model-theoretic semantics. The classes, roles, and individuals are given symbols from mutually disjoint sets of $\mathbf{I}$, $\mathbf{C}$, and $\mathbf{R}$ respectively. There exists another set called the domain of interpretation, Δ, which contains entities for resources, individuals, or single objects. Using the domain of interpretation, the individuals, classes, and roles are interpreted by functions $f_{\mathbf{I}} : \mathbf{I} \mapsto \Delta$, $f_{\mathbf{C}} : \mathbf{C} \mapsto 2^{\Delta}$, and $f_{\mathbf{R}} : \mathbf{I} \mapsto 2^{\Delta \times \Delta}$ respectively. The complex classes and role expressions are interpreted by an extended interpretation function, $.^{\mathcal{I}}$, such that the interpretation faithfully captures the structure of the knowledge base. If a model exists, then the knowledge base is satisfiable, and the implicit knowledge (logical consequence) is entailed though an inference procedure. DL logic uses efficient tableau algorithms to infer subsumption, class equivalence, class disjointness, global consistency, class consistency, instance checking, and instance retrieval. The reader is referred to [1, 13, 14] for a comprehensive discussion on $\mathcal{SROIQV}^{\mathcal{D}}$ DL syntax,

semantics, and model construction. The extended interpretation function has the following constituents: (1) $\top^{\mathcal{I}} = \Delta$ and $\bot^{\mathcal{I}} = \emptyset$; (2) $(\neg \mathsf{C})^{\mathcal{I}} = \Delta \backslash \mathsf{C}^{\mathcal{I}}$ such that, $\neg \mathsf{C}$ describes things which are not in C; (3) $(\mathsf{C} \sqcap \mathsf{D})^{\mathcal{I}} = \mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$ such that, $\mathsf{C} \sqcap \mathsf{D}$ describes things which are both in C and in D; (4) $(\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}} = \mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$ such that, $\mathsf{C} \sqcap \mathsf{D}$ describes things which are both in C or in D; (5) $(\exists \mathsf{R}.\mathsf{C})^{\mathcal{I}} = \{x |$ there is some y with $(x, y) \in \mathsf{R}^{\mathcal{I}} \cap y \in \mathsf{C}^{\mathcal{I}}\}$ such that, $\exists \mathsf{R}.\mathsf{C}$ describes those things which are connected via R with something in C; (6) $(\forall \mathsf{R}.\mathsf{C})^{\mathcal{I}} = \{x |$ for all y with $(x, y) \in \mathsf{R}^{\mathcal{I}} \Rightarrow y \in \mathsf{C}^{\mathcal{I}}\}$ such that, $\forall \mathsf{R}.\mathsf{C}$ describes those things x for which every y which connects from x via role R is in the class C; (7) $(\leq \mathsf{n}\mathsf{R}.\mathsf{C})^{\mathcal{I}} = \{x | \#\{(x, y) \in \mathsf{R}^{\mathcal{I}} \mid y \in \mathsf{C}^{\mathcal{I}}\} \leq n\}$ such that, $\leq \mathsf{n}\mathsf{R}.\mathsf{C}$ describes those things which are connected via R to at most n things in C; (8) $(\geq \mathsf{n}\mathsf{R}.\mathsf{C})^{\mathcal{I}} = \{x | \#\{(x, y) \in \mathsf{R}^{\mathcal{I}} \mid y \in \mathsf{C}^{\mathcal{I}}\} \geq n\}$ such that, $\geq \mathsf{n}\mathsf{R}.\mathsf{C}$ describes those things which are connected via R to at least n things in C; (9) $\{\mathsf{a}\}^{\mathcal{I}} = \{\mathsf{a}^{\mathcal{I}}\}$ such that, $\{\mathsf{a}\}$ describes the class containing only a; (10) $(\exists \mathsf{S}.\mathsf{Self})^{\mathcal{I}} = \{x \mid (x, x) \in \mathsf{S}^{\mathcal{I}}\}$ such that, $\exists \mathsf{S}.\mathsf{Self}$ describes those things which are connected to themselves via S; (11) $(\mathsf{R}^{-})^{\mathcal{I}} = \{(\mathsf{b}, \mathsf{a}) \mid (\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}}\}$ is for all $\mathsf{R} \in \mathbf{R}$; and (12) $\mathsf{U}^{\mathcal{I}} = \Delta \times \Delta$ is for the universal role U.

The extended interpretation is a model of the knowledge base, K, if the axioms of the knowledge base further satisfies the following constraints: (1) if $\mathsf{C}(\mathsf{a}) \in K$, then $\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$; (2) if $\mathsf{R}(\mathsf{a},\mathsf{b}) \in K$, then $(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$; (3) if $\neg \mathsf{R}(\mathsf{a},\mathsf{b}) \in K$, then $(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \notin \mathsf{R}^{\mathcal{I}}$; (4) if $\mathsf{C} \sqsubseteq \mathsf{D} \in K$, then $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$; (5) if $\mathsf{S} \sqsubseteq \mathsf{R} \in K$, then $\mathsf{S}^{\mathcal{I}} \subseteq \mathsf{R}^{\mathcal{I}}$; (6) if $\mathsf{S}_1 \circ \ldots \circ \mathsf{S}_n \sqsubseteq \mathsf{R} \in K$, then $\{(\mathsf{a}_1, \mathsf{a}_{n+1}) \in \Delta \times \Delta \mid$ there are $\mathsf{a}_1, \ldots, \mathsf{a}_n \in \Delta$ such that, $(\mathsf{a}_i, \mathsf{a}_{i+1}) \in \mathsf{S}_i^{\mathcal{I}}$ for all $i = 1, \ldots, n\} \in \mathsf{R}^{\mathcal{I}}$; (7) if symmetric$(\mathsf{R}) \in K$, then

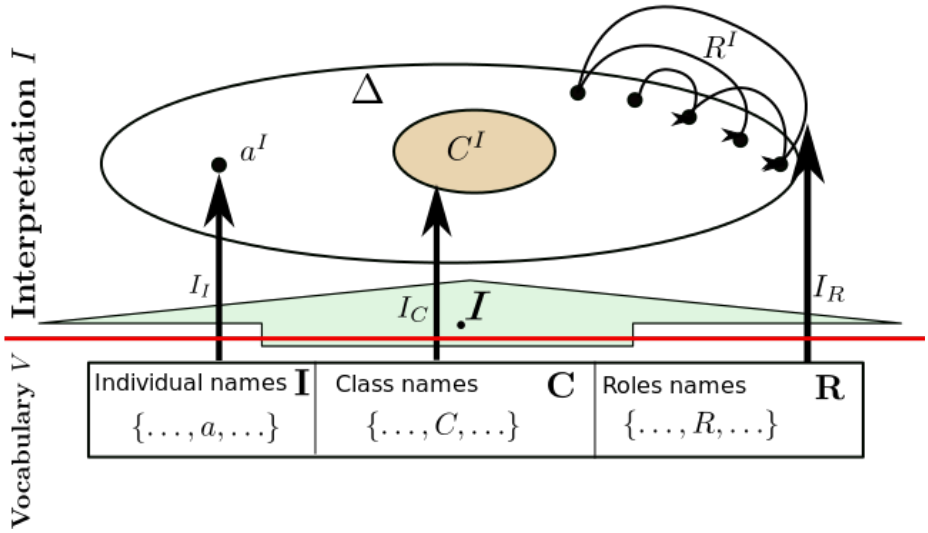


Figure 2.1: Graphical representation of direct model-theoretic interpretation of a DL (redrawn based on the original figure available in [1]).

$\{(x, x) \mid x \in \Delta\} \in \mathsf{R}^{\mathcal{I}}$; (8) if asymmetric($\mathsf{R}$) $\in K$, then $(x, y) \notin \mathsf{R}^{\mathcal{I}}$, whenever $(y, x) \in \mathsf{R}^{\mathcal{I}}$; and (9) if disjoint($\mathsf{R}$, S) $\in K$, then $\mathsf{R}^{\mathcal{I}} \cap \mathsf{S}^{\mathcal{I}} = \emptyset$.

The OWL 2 specification is based on $\mathcal{SROIQV}^{\mathcal{D}}$ Description Logic, and it is the decidable fragment of the first-order predicate logic ($\mathcal{SROIQV}^{\mathcal{D}}$ used in OWL 2 is called DL henceforth). Efficient reasoning engines exist that use the DL constructs in the knowledge base to infer about the domain of discourse. Modern DL reasoners such as the (1) tableau-based Konclude [15], FaCT++ [16], and Pellet [17] reasoners; and the (2) hyper-tableau HermiT [18] reasoner, use intelligent heuristics and optimization methods to perform inferencing as efficiently as possible.

2.2 RoboCup Soccer

The Robot World Cup Initiative (RoboCup) promotes and fosters AI and intelligent research by providing standard problems where a wide range of technologies can be integrated and examined [19]. Within the sub-leagues of RoboCup, we have focused on the 3D Soccer Simulation League to theorize, design, and develop new methods, and to empirically validate their potential usages.

The RoboCup 3D soccer simulation environment is based on the general purpose multi-agent simulator SimSpark [20]. The robot agents in the simulation are modeled based on Aldebaran NAO humanoid robots [2].

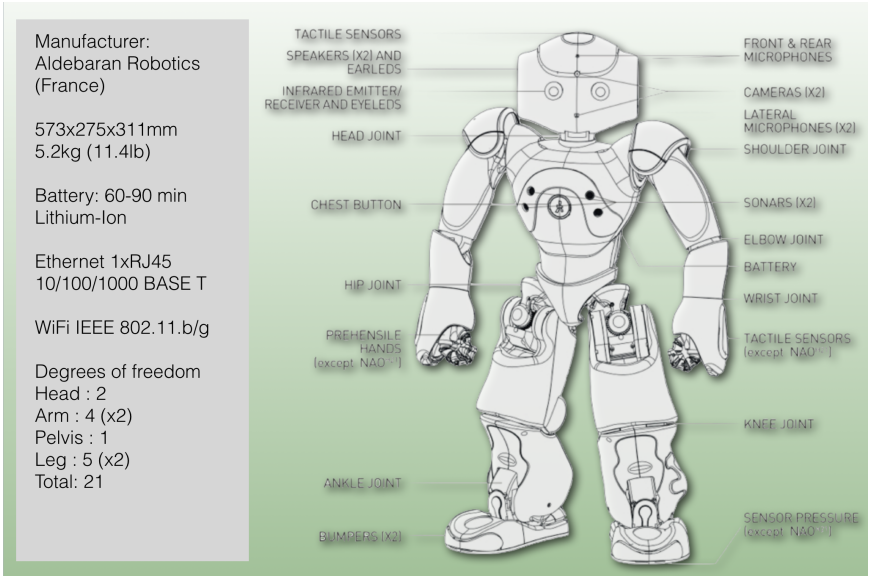


Figure 2.2: Aldebaran NAO humanoid robot specification [2].

Each robot has 22 degrees of freedom. Figure 2.2 shows the specification of NAO humanoid robot. It has one less degree of freedom compared to the simulation model due to the two hip joints are physically connected to a single motor. The agents communicate with the server through message passing and each agent is equipped

with noise free joint perceptors and effectors. In addition to this, each agent has a noisy restricted vision cone of 120^o. Every simulation cycle is limited to 20 ms, where agents perceive noise free angular measurements of each joint and the agents stimulate the necessary joints by sending torque values to the simulation server. The vision information from the server is available every third cycle (60 ms), which provides spherical coordinates of the perceived objects. The agents also have the option of communicating with each other every other simulation cycle (40 ms) by broadcasting a 20 $bytes$ message. The simulation league competitions are currently conducted with 11 robots on each side (22 total).

Chapter 3

Real-time Reasoning

Maintaining and deducing accurate world knowledge in a real-time, complex, adversarial, and stochastic environment such as the RoboCup Soccer (see Chapter 2.2) is a challenging task [21]. Knowledge needs be represented in real-time (i.e., within ms) and deductions from knowledge should be inferred within the same time constraints. We propose an extended assertion box for an expressive $\mathcal{SROIQV}^{\mathcal{D}}$ Description Logic (see Chapter 2.1) 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. In the given domain of discourse, we assume that the terminological/relational knowledge changes significantly slower than the asserted knowledge. Henceforth, (1) using the defined lattice structure; (2) a fixed terminological box; and (3) a fixed relation box, we empirically bound the size of the boxes to find the best trade-off 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 terminological box. 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.

3.1 Introductory Remarks

Complex robotic systems such as soccer playing robots in RoboCup environments [22] or self-driving cars (e.g., [23, 24]) need substantial awareness of their surroundings. In Artificial Intelligence (AI), there is a substantial gap between the information that a system collects via its modeling mechanisms and the high-level knowledge. 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. Here, we are investigating an ontological methodology to reduce this gap, ground information with respect to a domain of discourse, and reason in real-time.

DL provides an appropriate trade-off between expressivity and scalability in practice. The computational complexity of DL is primarily dominated by the data, which is NP-hard for $\mathcal{SROIQV}^{\mathcal{D}}$ DL ABoxes and N2ExpTime-complete for the combined TBox, RBox, and ABox. Thus, real-time systems need an upper bound for the size of the ABox, while retaining as much as logical consequences as possible. Therefore, we investigated the real-time performance of tableau-based reasoners with respect to the proposed ABox extension with the assumption that the TBox and RBox is stationary compared to the changing ABoxes over time.

Though DL is decidable, its worst-case complexity is 2NExpTime-complete, which enforces upper bounds on the size of TBox, RBox, and ABox. It is common practice that OWL 2 ontologies are reasoned off-line and use the deduced axioms later in the process to be scalable for practical problems [25]. Even though $\mathcal{SROIQV}^{\mathcal{D}}$ DL has high worst case complexity, it provides constructs that can be used in real-time robotic systems to model data, derive conclusions from this data, and exchange the now semantically grounded data among similar robotic systems. In this chapter, we empirically investigate the ability to use DL in a real-time system setting. The proposed method uses a fixed TBox and RBox, and provides the justification of using "an extended ABox" to represent modeling information in a lattice structure, that has temporal-like structures, without explicitly adding new constructs to $\mathcal{SROIQV}^{\mathcal{D}}$ DL. This gives us the opportunity to use existing DL reasoners in a real-time setting. Since the general reasoning problem is NP-hard, we maintain upper bounds on the number of axioms in each box, and empirically study the behavior of the extended ABox.

We hypothesis that agents formalize their goals in two layers: (1) the physical layers – controls related to walking, kicking etc. are conducted; and (2) the decision layers – high-level actions are taken to determine behaviors. Our proposed method resides in the decision layer to assert modeling information and deduce soccer domain specifications. One goal of multi-agent systems research is the investigation of the prospects of efficient cooperation among a set of agents in real-time environments. Our method can also be used to exchange information among agents with the given

conceptualization, thus, exchange the *semantically grounded* information rather that raw data. We have conducted all our experiments in the RoboCup 3D soccer simulation environment. Since the duty cycle is 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 situations in which multiple duty cycles, e.g., five cycles amounts to 100 ms, can be used with a 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 etc. 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 an active region surrounding the ball.

3.2 Related Work

In AI, an ontology is a computational artifact that defines a formal specification of a conceptualization [26]. 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. 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 [27, 28]. Our approach differs from this work in that we use $\mathcal{SROIQV}^{\mathcal{D}}$ 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 $\mathcal{SROIQV}^{\mathcal{D}}$ DL reasoners without substantial modifications. OWL-Time [29] allows representing temporal concepts and temporal relations in $\mathcal{SHOIN}^{\mathcal{D}}$ DL and paved the way to represent new languages such as tOWL [30] to represent 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 [31] has the ability to represent intervals and temporal properties, and their evolution over those intervals. There are many instances where these constructs are presented in OWL DL ontologies (e.g., [32, 33]), and we use an approach similar to that of Open Biological and Biomedical Ontologies Relation Ontology (OBO RO) [34] to represent temporal-like constructs within the ABox lattice structure.

OWL DL provides resources to represent entities in ontologies. These ontologies are large in nature (T/R/ABox) 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 [35]. 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., ∼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 [36] 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 [34]. 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 [37] recognizes and predicts spatio-temporal patterns the RoboCup 3D Simulation League domain. 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 ∼40 ms to compute inferences. The work most closely related to our work is presented in [38]. 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.

The assertions are generated as a direct consequence of the agent-environment interactions. These interactions constitute the agent knowledge, and it is represented in a formalized form. The knowledge representational forms show different degrees of computational complexities and expressiveness. The computational requirements increase with the extension of expressiveness of the representational forms. Therefore, there are alternative forms to represent knowledge, which are scalable for on-line learning, while preserving expressivity. Horde [6, 39] is a real-time learning architecture to express knowledge using General Value Functions in Reinforcement Learning. This method needs additional knowledge in representing entities with function approximation, question functions, answer functions, and step size tuning; yet provides an opportunity to learn knowledge from agent-environment interaction experiences.

3.3 ABox Extension

In this section, we present the syntax and semantics of the ABox extension to $\mathcal{SROIQV}^{\mathcal{D}}$ DL to represent entities in RoboCup 3D soccer simulation league. Our method is also applicable to other leagues in RoboCup initiatives with supervision. Firstly, we provide the definition, secondly, we describe the extension with an illus-

trative 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 3.1 *(**ABox Extension**) Given a fixed TBox and an RBox as defined in Chapter 2.1, the extended ABox is defined as follows:*

(1) There exists sampling points, $\mathbf{t_i} \in \mathbb{Z}^+$*, 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* $[R(a_{\mathbf{t_i}}, b_{\mathbf{t_j}})]_{\mathbf{t_j}}$*,* $\mathbf{t_i} \leq \mathbf{t_j}$ *for relations with given individuals* $a_{\mathbf{t_i}}$ *and* $b_{\mathbf{t_j}}$ *at the sampling point* $\mathbf{t_j}$*.*

(3) The individual assertions are realized with a $\mathsf{timeToLive} \in \mathbb{Z}_0^+$ *data type property, and they will be active for* $\mathsf{timeToLive} > 0$*.*

(4) The assertions are active for maximum sampling points of $\mathsf{latticeLength} \in \mathbb{Z}^+$*, and they are first created with* $\mathsf{timeToLive} = \mathsf{latticeLength}$*.*

(5) 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* $\mathsf{timeToLive} = 0$*.*

(6) 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 3.1 is given by the same constructs used in Chapter 2.1. 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 timeToLive = latticeLength, and they are purged when timeToLive = 0.

3.3.1 Algorithm

Algorithm 1 EXTENDED_ABOX

Require: $\mathcal{SROIQV}^{\mathcal{D}}$) TBox, RBox, latticeLength, refinements and Blackboard.
Ensure: Satisfiable $\mathcal{K}$ with an extended ABox.

Initialize ABox=$\emptyset$. {At $\mathbf{t} = 0$ an empty ABox.}
Sampling times $\forall_{\mathbf{t}}\ \mathbf{t} \in \mathbb{Z}_0^+$,
Find $\mathsf{ABox}_{\mathbf{i}} \in \mathsf{ABox}$ such that timeToLive=0 for $\mathbf{i} = 1, \ldots, \mathbf{t} - 1$.
if $\mathsf{ABox}_{\mathbf{i}} \neq \emptyset$ **then**
    $\mathsf{ABox} = \mathsf{ABox}$ - $\mathsf{ABox}_{\mathbf{i}}$.
end if
Decrease the timeToLive values of $\mathsf{ABox}_{\mathbf{i}} \in \mathsf{ABox}$.
Generate the new $\mathsf{ABox}_{\mathbf{t}}$ using the Blackboard (timeToLive=latticeLength).
Add axioms to $\mathsf{ABox}_{\mathbf{i}} \in \mathsf{ABox}$ from $\mathsf{ABox}_{\mathbf{t}}$ using active heuristics and prior knowledge.
$\mathsf{ABox} = \mathsf{ABox} + \mathsf{ABox}_{\mathbf{t}}$.
Check consistency of $\mathcal{K}$.
if $\mathcal{K} \models$ TRUE **then**
    Classify and realize $\mathcal{K}$.
else
    $\mathsf{ABox} = \mathsf{ABox}$ - $\mathsf{ABox}_{\mathbf{t}}$. Classify and realize $\mathcal{K}$.
end if
Execute query expressions using refinements.

Algorithm 1 shows the main operations to manipulation the extended ABox. It requires a fixed TBox, RBox, latticeLength, refinements, and Blackboard as inputs. refinements contain user defined conditions for query expressions. Blackboard contains

the information from agent sensors and filters at the given sampling point $\mathbf{t} \in \mathbb{Z}_0^+$. We generate the assertions using Blackboard information. Algorithm 1 ensures that the knowledge base, $\mathcal{K}$, is satisfiable with the extended ABox. For all $\mathbf{t}$, Algorithm 1 finds $\mathsf{ABox_i}$ with timeToLive=0 at line 3. If such an ABox exits then it is removed. Line 7 decreases timeToLive values of the existing ABoxes. Line 8 generates the new ABox. Line 9 adds the new ABox to the extended ABox; while adding axioms to existing ABoxes that matches the active heuristics. Finally, Algorithm 1 checks the consistency of $\mathcal{K}$. If the knowledge base is globally consistent, we query $\mathcal{K}$ for individuals. Otherwise, we remove $\mathsf{ABox_t}$ from the extended ABox and proceed to the next sampling instance.

3.3.2 An Illustrative Example

Figure 3.1 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 $\mathbf{t_i} \in \mathbb{Z}_0^+, \forall \mathbf{i} \in \mathbb{Z}_0^+$. 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, $\mathbf{1},\ \mathbf{2},\ \mathbf{3}$, and $\mathbf{4}$, are added to the ABox. Therefore, $\mathrm{ABox}_{\mathbf{t}_1}$ 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

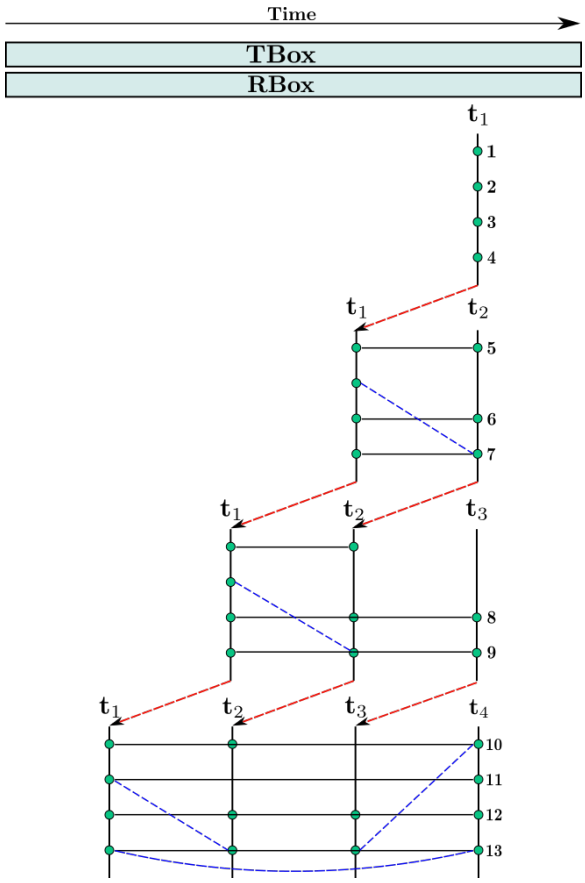


Figure 3.1: Extended ABox with the lattice structure.

also asserted with 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 $\mathbf{t}_1$.

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 shows all the abstract relations exists 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 shows from the dashed line in figure 3.1, 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 exists 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 3.1 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 3.1. The consistency of the knowledge base is checked with an $\mathcal{SROIQV}^{\mathcal{D}}$ 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 equivalence class expressions and user defined refinements.

3.3.3 Real World Examples

A few examples from our knowledge base might help to understand the process. We have developed an ontology to represent entities in the RoboCup 3D soccer simulation environment based on the conditions provided in Sections 2.1 and 3.2.

(1) We have defined an object in our domain of discourse;

$$\mathsf{Object} \equiv \exists \mathsf{timeToLive.nonNegativeInteger},$$

as any entity that has a positive time-to-live value.

(2) We have defined an agent using equivalence class expression;

$$\mathsf{Agent} \equiv \mathsf{Object} \sqcap \exists \mathsf{hasID}.\mathsf{nonNegativeInteger}[>0],$$

and $\mathsf{hasID} \in \mathbb{Z}^+$, 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;

$$\mathsf{HomeAgent} \sqsubseteq \mathsf{Agent} \text{ and } \mathsf{OpponentAgent} \sqsubseteq \mathsf{Agent}.$$

The agents are disjoint:

$$\mathsf{HomeAgent} \sqcap \mathsf{OpponentAgent} \sqsubseteq \bot.$$

(4) Most entities in the RoboCup 3D soccer simulation have a pose, i.e., an orientation/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:

$$\mathsf{Ball} \sqsubseteq \exists \mathsf{locatedIn.Pose2D} \text{ and } \mathsf{Ball} \sqsubseteq \mathsf{Object}.$$

(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:

$$\mathsf{ObjectsWithBallContact} \equiv \mathsf{Object} \sqcap \exists \mathsf{hasParticipant.Ball} \sqcap \exists \mathsf{hasParticipant.(Participant} \sqcap \exists \mathsf{distance.int}[<500]),$$

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 [40] 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 expert. We use distance threshold of 500 mm.

(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 least five individuals in the class expression. We have defined the equivalence class expression:

$$\mathsf{HoldBall} \equiv \mathsf{Agent} \sqcap \exists \mathsf{hasParticipant.(BallParticipant} \sqcap \exists \mathsf{distance.int}[<150]),$$

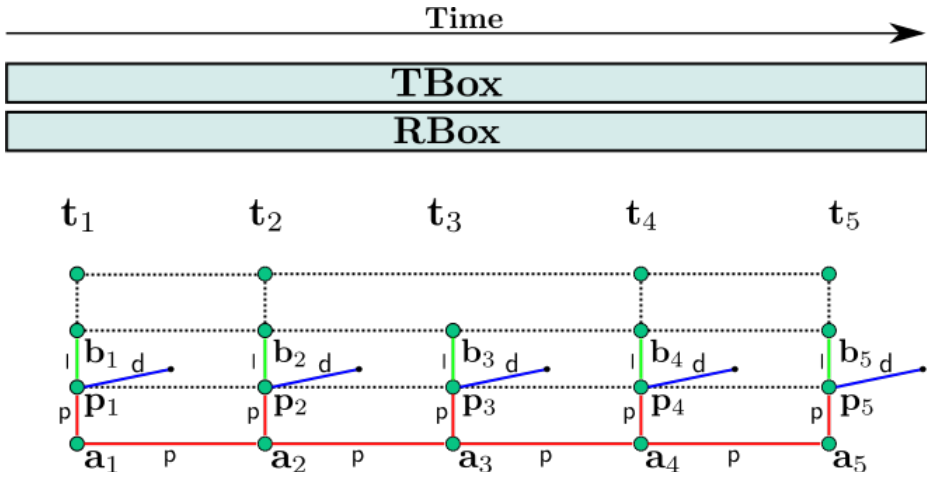


Figure 3.2: Extended ABox that matches an instance of the equivalent class expression HoldBall.

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 3.2. 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 equivalence class expression queries for the home agents that pass the ball to its teammates. It is defined as:

PassBall ≡ HomeAgent

⊓ ∃hasParticipant.(BallParticipant ⊓ ∃distance.int[<100])

⊓ ∃hasParticipant.(

HomeAgent ⊓ ∃hasParticipant.(BallParticipant ⊓ ∃distance.int[<150])

) ⊓ ∃ hasParticipant.(BallParticipant ⊓ ∃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.

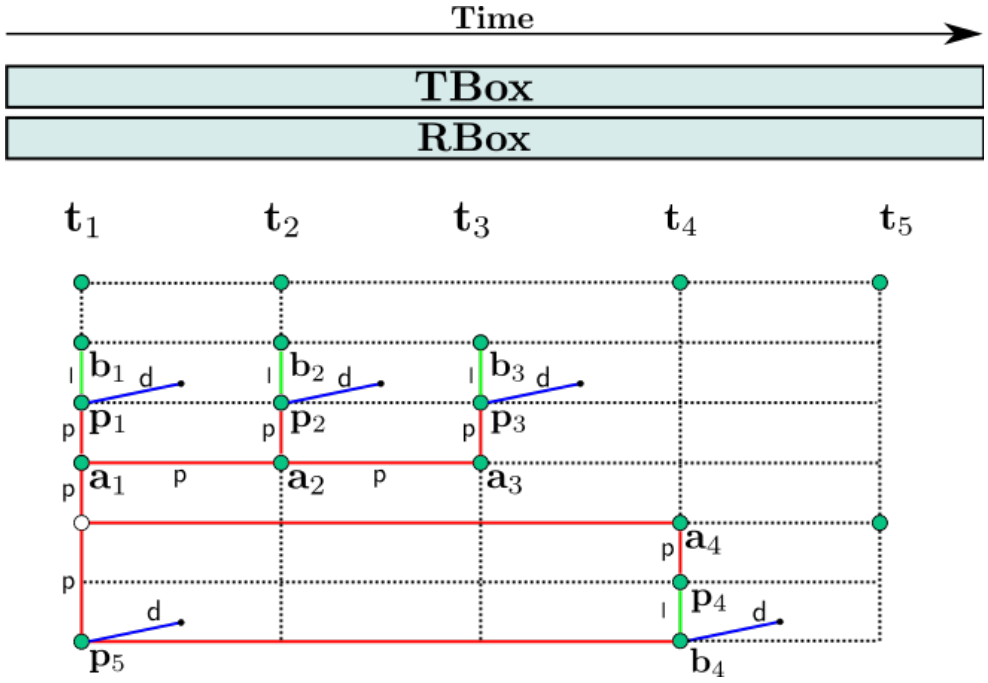


Figure 3.3: Extended ABox that matches an instance of the equivalent class expression PassBall.

There are some drawbacks 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 $\mathcal{SROIQV}^{\mathcal{D}}$ 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. Third, we commit to the Open World Assumption. 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 3.3. The symbol $\circ$ in Figure 3.3 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.

3.4 Experiments

In order to establish the baseline (average reasoning time), we have compared the distributions of the deduction times of the three reasoners:

(1) HermiT (1.3.8): `http://hermit-reasoner.com/`;

(1) Pellet (2.3.1): `http://clarkparsia.com/pellet/`; and

(1) FaCT++ (1.6.2): `http://code.google.com/p/factplusplus/`.

We have used 80−logical axioms (62−entities) from the ontology (TBox and RBox) for this experiment. Figure 3.4 shows the distribution of the reasoning times in milliseconds. The times are compared without incremental reasoning. 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).

We have used a 2.2GHz Core−i7 (4GB) laptop for all experiments. We have observed that the modified FaCT++ DL reasoner shows a statistically better performance over the other reasoners. Table 3.1 shows the average reasoning times and the 95% confidence intervals. This calculation uses 500 independent trials from each reasoner. Our modifications to FaCT++ DL reasoner have improved 22% over the original FaCT++ implementation. Henceforth, we have selected the modified

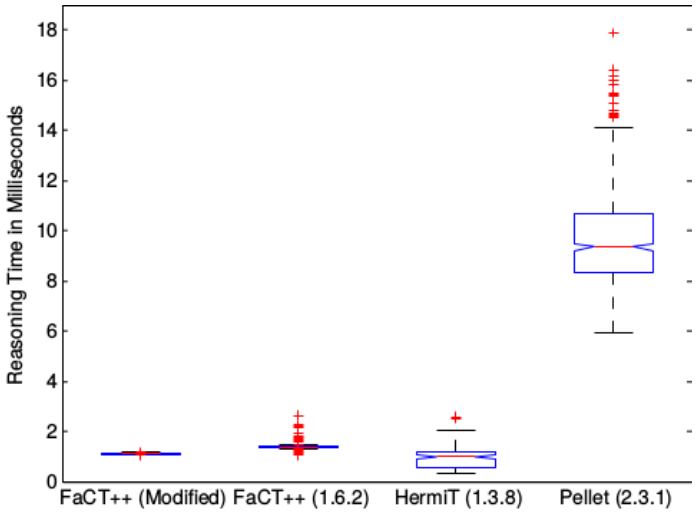


Figure 3.4: Distribution of the reasoning times in ms.

FaCT++ DL reasoner to be used with the simulated agents, and we have used the reasoner in non-incremental mode.

The baseline establishment has set the empirical lower bound to zero individuals and $\sim$ 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 3.3.3 to the ontology. Secondly, we have created a hypothetical world model for an agent. This world model changes its believes about teammates, opponents, and ball randomly. We have used a uniform distribution to sample entities in the belief model. All poses are randomly generated

Table 3.1: Average reasoning times (95% confidence).

Reasoner	Time in milliseconds (95% confidence)
FaCT++ (Modified)	$\mathbf{1.092 \pm 0.002}$
FaCT++ (1.6.2)	1.403 ± 0.009
HermiT (1.3.8)	2.289 ± 2.654
Pellet (2.3.1)	11.634 ± 2.465

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 3.5 and 3.6 shows the concluding results (the error bars show one-standard deviation of bins of ten).

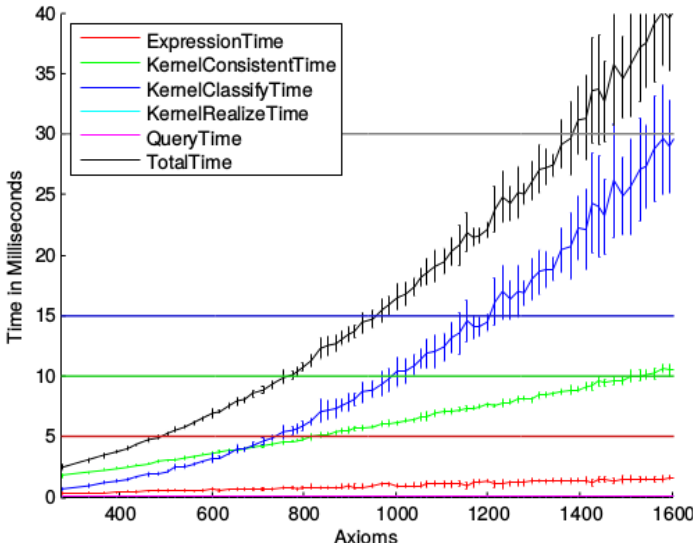


Figure 3.5: Establishment of empirical upper bound.

Figure 3.5 shows that the consistency checking scales linearly with axioms. The classification 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 and expression times are relatively negligible. The expression time exhibits our operations for refinements and to tracks the evolution of the extended ABox. It shows linear time complexity and it is justifiable for real-time operations.

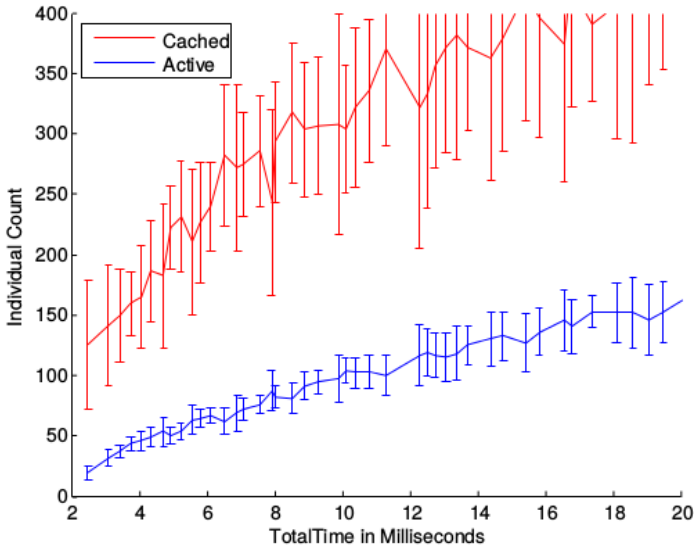


Figure 3.6: Relationship between reasoning time and number of individuals.

Figure 3.6 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 and this amounts to ∼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 3.5 and 3.6 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 wrt. 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. We set the latticeLength to five. In our on-line setting, we ran 11 vs 11 games with the given heuristics. An agent tracks 49.18 ± 4.86 individuals and 529.44 ± 105.31 axioms in 6.57 ± 2.81 ms. Therefore, we justify that our algorithm is real-time compatible on a RoboCup 3D simulated robot.

3.5 Conclusions

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 $\mathcal{SROIQV}^{\mathcal{D}}$ DL ABox with a lattice structure and it provides flexibility to use existing DL reasoners. We have 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.

Chapter 4

Knowledge Learning using General Value Functions

We use recently introduced Off-Policy Gradient Descent algorithms within Reinforcement Learning that illustrate learnable knowledge representations for dynamic role assignments. The results show that the agents have learned competitive policies against the top teams from the RoboCup 2012 competitions for three vs three, five vs five, and seven vs seven agents. We have explicitly used subsets of agents to identify the dynamics and the semantics for which the agents learn to maximize their performance measures, and to gather knowledge about different objectives, so that all agents participate effectively and efficiently within the group.

4.1 Introductory Remarks

Here, we investigate a mechanism suitable for decision layers to use recently introduced Off-Policy Gradient Decent Algorithms in Reinforcement Leaning (RL) that illustrate learnable knowledge representations to learn *a dynamic role assignment function*.

In order to design an effective dynamic role assignment functions, the agents need to consider the dynamics of agent-environment interactions. We consider these interactions as the agent's knowledge. If this knowledge is represented in a formalized form (e.g., in first-order predicate logic or DL (see Chapter 3)) an agent could infer many aspects about its interactions consistent with that knowledge. The knowledge representational forms show different degrees of computational complexities and expressiveness [41].

The computational requirements increase with the extension of expressiveness of the representational forms. Therefore, we need to identify and commit to a representational form, which is scalable for on-line learning while preserving expressivity. A *human* soccer player knows a lot of information about the game before (s)he enters onto the field and this prior knowledge influences the outcome of the game to a great extent. In addition, human soccer players dynamically change their knowledge during games in order to achieve maximum rewards. Therefore, the knowledge of the human soccer player is to a certain extent either *predictive* or *goal-oriented*. Can a *robotic* soccer player collect and maintain predictive and goal-oriented knowledge? This is a challenging problem for agents with time constraints and limited computational resources.

We learn the role assignment function using a framework that is developed based on the concepts of Horde, the real-time learning methodology, to express knowledge using General Value Functions (GVFs) [41]. Similar to Horde's sub-agents, the agents in a team are treated as independent RL sub-agents, but the agents take actions based

on their belief of the world model. The agents may have different world models due to noisy perceptions and communication delays. The GVFs are constituted within the RL framework. They are predictions or off-policy controls that are answers to questions. For example, in order to make a prediction a question must be asked of the form "If I move in this formation, would I be in a position to score a goal?", or "What set of actions do I need to block the progress of the opponent agent with the number 3?". The question defines what to learn. Thus, the problem of prediction or control can be addressed by learning value functions. An agent obtains its knowledge from information communicated back and forth between the agents and the agent-environment interaction experiences.

There are primarily two algorithms to learn the GVFs, and these algorithms are based on Off-Policy Gradient Temporal Difference (OP-GTD) learning: 1. with action-value methods, a prediction question uses GQ(λ) algorithm [42], and a control or a goal-oriented question uses Greedy-GQ(λ) algorithm [43]. These algorithms learn a deterministic target policy and the control algorithm finds the greedy action with respect to the action-value function; and 2. with policy-gradient methods, a goal-oriented question can be answered using an Off-Policy Actor-Critic algorithm [44], with an extended state-value function, GTD(λ) [45], for GVFs. The policy gradient methods are favorable for problems having stochastic optimal policies, adversarial environments, and problems with large action spaces. The OP-GTD algorithms possesses a number of properties that are desirable for on-line learning within the RoboCup 3D Soccer Simulation environment: 1. off-policy updates; 2. linear function

approximation; 3. no restrictions on the features used; 4. temporal-difference learning; 5. on-line and incremental; 6. linear in memory and per-time-step computation costs; and 7. convergent to a local optimum or equilibrium point [46, 43].

In this chapter, we present a methodology and an implementation to learn a dynamic role assignment function considering the dynamics of agent-environment interactions based on GVFs. The agents ask questions and approximate value functions answer to those questions. The agents independently learn the role assignment functions in the presence of an adversary team. Based on the interactions, the agents may have to change their roles in order to continue in the formation and to maximize rewards. There is a finite number of roles that an agent can commit to, and the GVFs learn about the role assignment function. We have conducted all our experiments in the RoboCup 3D Soccer Simulation League Environment(see Chapter 3).

4.2 Related Work

One goal of multi-agent systems research is the investigation of the prospects of efficient cooperation among a set of agents in real-time environments. In our research, we focus on the cooperation of a set of agents in a real-time robotic soccer simulation environment, where the agents learn about an optimal or a near-optimal role assignment function within a given formation using GVFs. This subtask is particularly challenging compared to other simulation leagues considering the limitations of the environment, i.e. the limited locomotion capabilities, limited communication band-

width, or crowd management rules. The role assignment is a part of the hierarchical machine learning paradigm [47, 48], where a formation defines the role space. Homogeneous agents can change roles flexibly within a formation to maximize a given reward function.

The RL framework offers a set of tools to design sophisticated and hard-to-engineer behaviors in many different robotic domains [49]. Within the domain of *robotic soccer*, RL has been successfully applied in learning the keep-away subtask in the RoboCup 2D [50] and 3D [51] Soccer Simulation Leagues. Also, in other RoboCup leagues, such as the Middle Size League, RL has been applied successfully to acquire competitive behaviors [52]. One of the noticeable impact on RL is reported by the Brainstormers team, the RoboCup 2D Simulation League team, on learning different subtasks [53]. A comprehensive analysis of a general batch RL framework for learning challenging and complex behaviors in robot soccer is reported in [54]. Despite convergence guarantees, Q(λ) [55] with linear function approximation has been used in role assignment in robot soccer [56] and faster learning is observed with the introduction of heuristically accelerated methods [57]. The dynamic role allocation framework based on dynamic programming is described in [58] for real-time soccer environments. The role assignment with this method is tightly coupled with the agent's low-level abilities and does not take the opponents into consideration. On the other hand, the proposed framework uses the knowledge of the opponent positions as well as other dynamics for the role assignment function.

Sutton et al. [41] have introduced a real-time learning architecture, Horde, for expressing knowledge using General Value Functions (GVFs). Our research is built on Horde to ask a set of questions such that the agents assign optimal or near-optimal roles within formations. In addition, following researches describe methods and components to build strategic agents: [59] describes a methodology to build a cognizant robot that possesses vast amount of situated, reversible and expressive knowledge. [60] presents a methodology to "next" in real time predicting thousands of features of the world state, and [61] presents methods predict about temporally extended consequences of a robot's behaviors in general forms of knowledge. The GVFs are successfully used (e.g., [62, 63]) for switching and prediction tasks in assistive biomedical robots.

4.3 Learnable knowledge representation for Robotic Soccer

Recently, within the context of the RL framework [55], a knowledge representation language has been introduced, that is expressive and learnable from sensorimotor data. This representation is directly usable for robotic soccer as agent-environment interactions are conducted through perceptors and actuators. In this approach, knowledge is represented as a large number of *approximate value functions* each with its 1. *own policy*; 2. *pseudo-reward function*; 3. *pseudo-termination function*; and 4. *pseudo-terminal-reward function* [41]. In continuous state spaces, approximate

value functions are learned using function approximation and using more efficient off-policy learning algorithms. First, we briefly introduce some of the important concepts related to the GVFs. The complete information about the GVFs are available in [41, 42, 43, 45]. Second, we show its direct application to simulated robotic soccer.

4.3.1 Interpretation

The interpretation of the approximate value function as a knowledge representation language grounded on information from perceptors and actuators is defined as:

Definition 4.1 *The knowledge expressed as an approximate value function is true or accurate, if its numerical values matches those of the mathematically defined value function it is approximating.*

Therefore, according to the Definition (4.1), a value function asks a *question*, and an approximate value function is the *answer* to that question. Based on prior interpretation, the standard RL framework extends to represent learnable knowledge as follows. In the standard RL framework [55], let the agent and the world interact in discrete time steps $t = 1, 2, 3, \ldots$. The agent senses the state at each time step $S_t \in \mathcal{S}$, and selects an action $A_t \in \mathcal{A}$. One time step later the agent receives a scalar reward $R_{t+1} \in \mathbb{R}$, and senses the state $S_{t+1} \in \mathcal{S}$. The rewards are generated according to the *reward function* $r : S_{t+1} \rightarrow \mathbb{R}$. The objective of the standard RL framework is to learn the stochastic action-selection *policy* $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, that gives the probability of selecting each action in each state, $\pi(s, a) = \pi(s|a) = \mathcal{P}(A_t = a|S_t = s)$, such

that the agent maximizes rewards summed over the time steps. The standard RL framework extends to include a *terminal-reward-function*, $z : \mathcal{S} \to \mathbb{R}$, where $z(s)$ is the terminal reward received when the termination occurs in state s. In the RL framework, $\gamma \in [0, 1)$ is used to discount delayed rewards. Another interpretation of the discounting factor is a constant probability of $1 - \gamma$ termination of arrival to a state with zero terminal-reward. This factor is generalized to a *termination function* $\gamma : \mathcal{S} \to [0, 1]$, where $1 - \gamma(s)$ is the probability of termination at state s, and a terminal reward $z(s)$ is generated.

4.3.2 Off-Policy Action-Value Methods for GVFs

The first method to learn about GVFs, from off-policy experiences, is to use action-value functions. Let G_t be the complete return from state S_t at time t, then the sum of the rewards (transient plus terminal) until termination at time T is:

$$G_t = \sum_{k=t+1}^{T} r(S_k) + z(S_T).$$

The action-value function is:

$$Q^\pi(s, a) = \mathbb{E}(G_t | S_t = s, A_t = a, A_{t+1:T-1} \sim \pi, T \sim \gamma),$$

where, $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. This is the expected return for a trajectory started from state s, and action a, and selecting actions according to the policy π, until termination occurs with γ. We approximate the action-value function with $\hat{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Therefore, the action-value function is a precise grounded question, while the approximate action-value function offers the numerical answer. The complete algorithm for Greedy-GQ(λ) with linear function approximation for GVFs learning is as shown in Algorithm 2.

Algorithm 2 Greedy-GQ(λ) with linear function approximation for GVFs learning [45].

Initialize w_0 to 0, and θ_0 arbitrary.
Choose proper (small) positive values for α_θ, α_w, and set values for $\gamma(.) \in (0, 1]$, $\lambda(.) \in [0, 1]$.
repeat
  Initialize $e = 0$.
  Take A_t from S_t according to π_b, and arrive at S_{t+1}.
  Observe sample, $(S_t, A_t, r(S_{t+1}), z(S_{t+1}), S_{t+1},)$ at time step t (with their corresponding state-action feature vectors), where $\hat{\phi}_{t+1} = \phi(S_{t+1}, A^*_{t+1}), A^*_{t+1} = \text{argmax}_b\, \theta_t^{\mathrm{T}} \phi(S_{t+1}, b)$.
  for each observed sample **do**
    $\delta_t \leftarrow r(S_{t+1}) + (1 - \gamma(S_{t+1}))z(S_{t+1}) + \gamma(S_{t+1})\theta_t^{\mathrm{T}}\hat{\phi}_{t+1} - \theta_t^{\mathrm{T}}\phi_t$.
    If $A_t = A^*_t$, **then** $\rho_t \leftarrow \frac{1}{\pi_b(A^*_t|S_t)}$; **otherwise** $\rho_t \rightarrow 0$.
    $e_t \leftarrow I_t\phi_t + \gamma(S_t)\lambda(S_t)\rho_t e_{t-1}$.
    $\theta_{t+1} \leftarrow \theta_t + \alpha_\theta[\delta_t e_t - \gamma(S_{t+1})(1 - \lambda(S_{t+1}))(w_t^{\mathrm{T}} e_t)\hat{\phi}_{t+1}]$.
    $w_{t+1} \leftarrow w_t + \alpha_w[\delta_t e_t - (w_t^{\mathrm{T}}\phi_t)\phi_t)]$.
  end for
until each episode.

The GVFs are defined over four functions: $\pi, \gamma, r,$ and z. The functions r and z act as pseudo-reward and pseudo-terminal-reward functions respectively. Function γ is also in pseudo form as well. However, the γ function is more substantive than reward functions as the termination interrupts the normal flow of state transitions. In pseudo termination, the standard termination is omitted. In robotic soccer, the base problem can be defined as the time until a goal is scored by either the home or the opponent team.

We can consider a pseudo-termination has occurred when the striker is changed. The GVF with respect to a state-action function is defined as:

$$Q^{\pi,\gamma,r,z}(s,a) = \mathbb{E}(G_t|S_t = s, A_t = a, A_{t+1:T-1} \sim \pi, T \sim \gamma).$$

The four functions, $\pi, \gamma, r,$ and z, are the *question functions* to GVFs, which in return defines the general value function's semantics. The RL agent learns an approximate action-value function, $\hat{Q}$, using the four auxiliary functions π, γ, r and z. We assume that the state space is continuous and the action space is discrete. We approximate the action-value function using a linear function approximator. We use a feature extractor $\phi : S_t \times A_t \rightarrow \{0,1\}^N, N \in \mathbb{N}$, built on tile coding [55] to generate feature vectors from state variables and actions. This is a sparse vector with a constant number of "1" features, hence, a constant norm. In addition, tile coding has the key advantage of real-time learning and to implement computationally efficient algorithms to learn approximate value functions. In linear function approximation, there exists a weight vector, $\theta \in \mathbb{R}^N, N \in \mathbb{N}$, to be learned. Therefore, the approximate GVFs are defined as:

$$\hat{Q}(s,a,\theta) = \theta^{\mathrm{T}}\phi(s,a),$$

such that, $\hat{Q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^N \rightarrow \mathbb{R}$. Weights are learned using the gradient-descent temporal-difference Algorithm 2 [45]. The Algorithm learns stably and efficiently using linear function approximation from *off-policy* experiences. Off-policy experiences are generated from a *behavior policy*, π_b, that is different from the policy being learned

about named as *target policy*, π. Therefore, one could learn multiple target policies from the same behavior policy.

4.3.3 Off-Policy Policy Gradient Methods for GVFs

The second method to learn about GVFs is using the off-policy policy gradient methods with actor-critic architectures that use a state-value function suitable for learning GVFs. It is defined as:

$$V^{\pi,\gamma,r,z}(s) = \mathbb{E}(G_t|S_t = s, A_{t:T-1} \sim \pi, T \sim \gamma),$$

where, $V^{\pi,\gamma,r,z}(s)$ is the true state-value function, and the approximate GVF is defined as:

$$\hat{V}(s, v) = v^{\mathrm{T}}\phi(s),$$

where, the functions $\pi, \gamma, r,$ and z are defined as in the subsection (4.3.2). Since our target policy π is discrete stochastic, we use a Gibbs distribution of the form:

$$\pi(a|s) = \frac{e^{u^{\mathrm{T}}\phi(s,a)}}{\sum_b e^{u^{\mathrm{T}}\phi(s,b)}},$$

where, $\phi(s, a)$ are state-action features for state s, and action a, which are in general unrelated to state features $\phi(s)$ that are used in state-value function approximation. $u \in \mathbb{R}^{N_u}, N_u \in \mathbb{N}$, is a weight vector, which is modified by the actor to learn about the stochastic target policy. The log-gradient of the policy at state s, and

action a, is $\frac{\nabla_u \pi(a|s)}{\pi(a|s)} = \phi(s,a) - \sum_b \pi(b|s)\phi(s,b)$. The complete algorithm for Off-PAC with linear function approximation for GVFs learning is shown in Algorithm 3.

We are interested in finding optimal policies for the dynamic role assignment, and henceforth, we use Algorithms 2 and 3 for control purposes (Appendix A provides the detail description of the implementation of the Algorithms 2 and 3 using C++). We use linear function approximation for continuous state spaces, and discrete actions are used within options. Lastly, to summarize, the definitions of the question functions and the answer functions are given as:

Definition 4.2 *The question functions are defined by:*

1. $\pi : S_t \times A_t \to [0,1]$ *(target policy is greedy w.r.t. learned value function);*

2. $\gamma : S_t \to [0,1]$ *(termination function);*

3. $r : S_{t+1} \to \mathbb{R}$ *(transient reward function); and*

4. $z : S_{t+1} \to \mathbb{R}$ *(terminal reward function).*

Definition 4.3 *The answer functions are defined by:*

1. $\pi_b : S_t \times A_t \to [0, 1]$ *(behavior policy);*

2. $I_t : S_t \times A_t \to [0, 1]$ *(interest function);*

3. $\phi : S_t \times A_t \to \mathbb{R}^N$ *(feature-vector function); and*

4. $\lambda : S_t \to [0, 1]$ *(eligibility-trace decay-rate function).*

Algorithm 3 Off-PAC with linear function approximation for GVFs learning [45, 44].

Initialize w_0 to 0, and v_0 and u_0 arbitrary.
Choose proper (small) positive values for α_v, α_w, α_u, and set values for $\gamma(.) \in (0, 1]$, $\lambda(.) \in [0, 1]$.
repeat
  Initialize $e^v = 0$, and $e^u = 0$.
  Take A_t from S_t according to π_b, and arrive at S_{t+1}.
  Observe sample, $(S_t, A_t, r(S_{t+1}), z(S_{t+1}), S_{t+1})$ at time step t (with their corresponding state (ϕ_t, ϕ_{t+1}) feature vectors, where $\phi_t = \phi(S_t)$).
  for each observed sample **do**
    $\delta_t \leftarrow r(S_{t+1}) + (1 - \gamma(S_{t+1}))z(S_{t+1}) + \gamma(S_{t+1})v_t^{\mathrm{T}}\phi_{t+1} - v_t^{\mathrm{T}}\phi_t$.
    $\rho_t \leftarrow \frac{\pi(A_t|S_t)}{\pi_b(A_t|S_t)}$.
    Update the critic (GTD(λ) algorithm for GVFs).
      $e_t^v \leftarrow \rho_t(\phi_t + \gamma(S_t)\lambda(S_t)e_{t-1}^v)$.
      $v_{t+1} \leftarrow v_t + \alpha_v[\delta_t e_t^v - \gamma(S_{t+1})(1 - \lambda(S_{t+1}))(e_t^{v\mathrm{T}} w_t)\phi_{t+1}]$.
      $w_{t+1} \leftarrow w_t + \alpha_w[\delta_t e_t - (w_t^{\mathrm{T}}\phi_t)\phi_t)]$.
    Update the actor.
      $e_t^u \leftarrow \rho_t \left[\frac{\nabla_u \pi(A_t|S_t)}{\pi(A_t|S_t)} + \gamma(S_t)\lambda(S_{t+1})e_{t-1}^u\right]$.
      $u_{t+1} \leftarrow u_t + \alpha_u \delta_t e_t^u$.
  end for
until each episode.

4.4 Dynamic Role Assignment

A *role* is a specification of an internal or an external behavior of an agent. In our soccer domain, roles select behaviors of agents based on different reference criteria:

the agent close to the ball becomes the striker. Given a role space, $\mathcal{R} = \{r_1, \ldots, r_n\}$, of size n, the collaboration among $m \leq n$ agents, $\mathcal{A} = \{a_1, \ldots, a_m\}$, is obtained through *formations*. The role space consists of active and reactive roles. For example, the striker is an active role and the defender could be a reactive role. Given a reactive role, there is a function, $R \mapsto T$, that maps roles to target positions, T, on the field. These target positions are calculated with respect to a reference pose (e.g., ball position) and other auxiliary criteria such as crowd management rules. A role assignment function, $R \mapsto A$, provides a mapping from role space to agent space, while maximizing some reward function. The role assignment function can be static or dynamic. Static role assignments often provide inferior performance in robot soccer [58]. Therefore, we learn a dynamic role assignment function within the RL framework using off-policy control.

Figure 4.1: Primary formation, [3]

4.4.1 Target Positions with the Primary Formation

Within our framework, an agent can choose one role among thirteen roles. These roles are part of a primary formation, and an agent calculates the respective target positions according to its belief of the absolute ball position and the rules imposed by the 3D soccer simulation server. We have labeled the role space in order to describe the behaviors associated with them. Figure 4.1 shows the target positions for the role space before the kickoff state. The agent closest to the ball takes the striker role (SK), which is the only active role.

Let us assume that the agent's belief of the absolute ball position is given by (x_b, y_b). Forward left (FL) and forward right (FR) target positions are offset by $(x_b, y_b) \pm (0, 2)$. The extended forward left (EX1L) and extended forward right ((EX1R)) target positions are offset by $(x_b, y_b) \pm (0, 4)$. The stopper (ST) position is given by $(x_b - 2.0, y_b)$. The extended middle (EX1M) position is used as a blocking position and it is calculated based on the closest opponent to the current agent. The other target positions, wing left (WL), wing right (WR), wing middle (WM), back left (BL), back right (BR), and back middle (BM) are calculated with respect to the vector from the middle of the home goal to the ball and offset by a factor which increases close to the home goal. When the ball is within the reach of goal keeper, the (GK) role is changed to goal keeper striker (GKSK) role.

We slightly change the positions when the ball is near the side lines, home goal, and opponent goal. These adjustments are made in order to keep the target positions inside the field. We allow target positions to be overlapping. The dynamic role

assignment function may assign the same role during the learning period. In order to avoid position conflicts an offset is added; the feedback provides negative rewards for such situations.

4.4.2 Roles to RL Action Mapping

The agent closest to the ball becomes the striker, and only one agent is allowed to become the striker. The other agents except the goalie are allowed to choose from twelve roles. We map the available roles to discrete actions of the RL algorithm. In order to use Algorithm 2, an agent must formulate a question function using a value function, and the answer function provides the solution as an approximate value function. All the agents formulate the same question: *What is my role in this formation in order to maximize future rewards?* All agents learn independently according to the question, while collaboratively aiding each other to maximize their future reward. We make the assumption that the agents do not communicate their current role. Therefore, at a specific step, multiple agents may commit to the same role. We discourage this condition by modifying the question: *What is my role in this formation in order to maximize future rewards, while maintaining a completely different role from all teammates in all time steps?*

4.4.3 State Variables Representation

Figure 4.2 shows the schematic diagram of the state variable representation. All points and vectors in Figure 4.2 are defined with respect to a global coordinate system. h

is the middle point of the home goal, while o is the middle point of the opponent goal. b is the ball position. $\|.\|$ represents the vector length, while $\angle pqr$ represents the angle among three points p, q, and r pivoted at q. a_i represents the self-localized point of the $i = 1, \ldots, 11$ teammate agent. y_i is some point in the direction of the robot orientation of teammate agents. c_j, $j = 1, \ldots, 11$, represents the mid-point of the tracked opponent agent. x represents a point on a vector parallel to unit vector e_x. Using these labels, we define the state variables as:

$$\{\| \vec{v}_{hb} \|, \| \vec{v}_{bo} \|, \angle hbo, \{\| \vec{v}_{a_i b} \|, \angle y_i a_i b, \angle a_i bx\}_{i=n_{start}}^{n_{end}}, \{\| \vec{v}_{c_j b} \|, \angle c_j bx, \}_{j=1}^{m_{max}}\}.$$

n_{start} is the teammate starting id and n_{end} the ending id. m_{max} is the number of opponents considered. Angles are normalized to $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

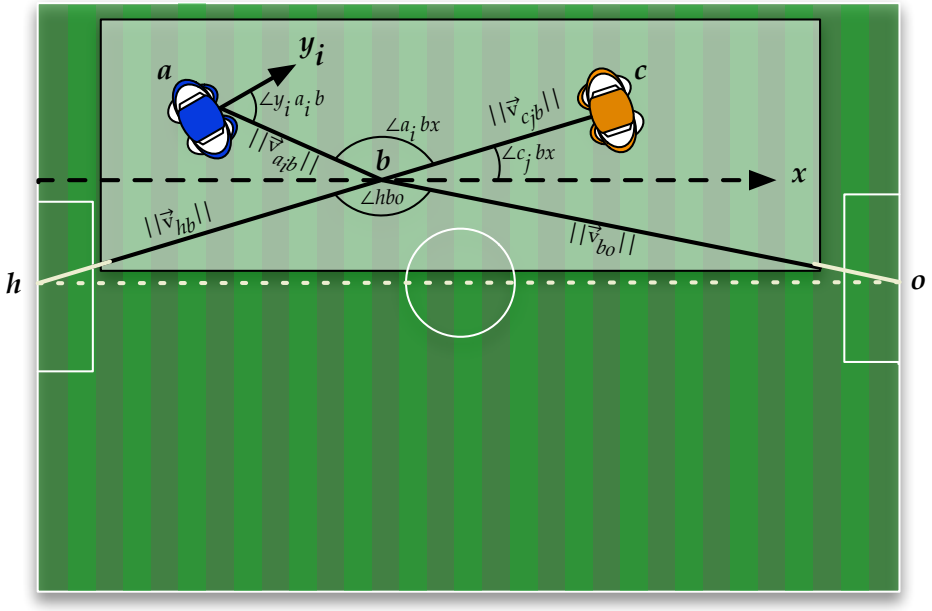


Figure 4.2: State variable representation and the primary function. Some field lines are omitted due to clarity.

Table 4.1: GVF Definitions for State-Action Functions: Question functions.

Function	**Values**
π	Greedy w.r.t. $\hat{Q}$.
$\gamma(.)$	0.8
t	2 seconds.
$r(.)$	The change of x value of the absolute ball position. A small negative reward of 0.01 for each cycle. A negative reward of 5 is given to all agents within a radius of 1.5 meters.
$z(.)$	$+100$ for scoring against opponent. -100 for opponent scoring.

4.5 Question and Answer Functions

There are twelve actions available in each state. We have left out the striker role from the action set. The agent nearest to the ball becomes the striker. All agents communicate their belief to other agents. Based on their belief, all agents calculate a cost function and assign the closest agent as the striker. We have formulated a cost function based on relative distance to the ball, angle of the agent, number of teammates and opponents within a region near the ball, and whether the agents are active. In our formulation, there is a natural termination condition; scoring goals. With respect to the striker role assignment procedure, we define a pseudo-termination condition. When an agent becomes a striker, a pseudo-termination occurs, and the striker agent does not participate in the learning process unless it chooses another role. We define the question and answer functions as shown in Tables 4.1, 4.2, 4.3, and 4.4.

Table 4.2: GVF Definitions for State-Action Functions: Answer functions.

Function	**Values**
π_b	ϵ-greedy w.r.t. target state-action function
ϵ	0.05
$I_t(.)$	1
$\lambda(.)$	0.8
$\parallel \theta \parallel = \parallel \mathbf{w} \parallel$	$10^6 + 1$
$\parallel \mathbf{e} \parallel$	2000 (efficient trace implementation)
α_θ	$\frac{0.01}{289}, \frac{0.01}{449}, \frac{0.01}{609}$
α_w	$0.001 \times \alpha_\theta$
$\phi(.,.)$	Tile coding to formulate the feature vector. $n_{start} = 2$ and $n_{end} = 3, 5, 7$. $m_{max} = 3, 5, 7$. Therefore, there are $18, 28$, and 30 state variables. A state variable is independently tiled with 16 tilings with approximately each with $\frac{1}{16}$ generalization. Therefore, there are $288+1, 448+1, 608+1$ active tiles (i.e., tiles with feature 1) hashed to a binary vector dimension $10^6 + 1$. The bias feature is always active.

Table 4.3: GVF for Gradient Descent Functions: Question functions.

Function	**Values**
π	Gibbs distribution.
$\gamma(.)$	0.9
t	2 seconds.
$r(.)$	The change of x value of the absolute ball position. A small negative reward of 0.01 for each cycle. A negative reward of 5 is given to all agents within a radius of 1.5 meters.
$z(.)$	$+100$ for scoring against opponent. -100 for opponent scoring.

Table 4.4: GVF for Gradient Descent Functions: Answer functions.

Function	**Values**
π_b	The learned Gibbs distribution is used with a small perturbation. In order to provide exploration, with probability 0.01, Gibbs distribution is perturbed using some β value. In our experiments, we use $\beta = 0.5$. Therefore, we use a behavior policy: $\frac{e^{u^{\mathrm{T}}\phi(s,a)+\beta}}{\sum_b e^{u^{\mathrm{T}}\phi(s,b)+\beta}}$.
$\lambda_{\text{critic}}(.) = \lambda_{\text{actor}}(.)$	0.3
$\parallel \mathbf{u} \parallel$	$10^6 + 1$
$\parallel \theta \parallel = \parallel \mathbf{w} \parallel$	$10^6 + 1$
$\parallel \mathbf{e} \parallel$	2000 (efficient trace implementation).
$\parallel \mathbf{e^v} \parallel = \parallel \mathbf{e^u} \parallel$	2000 (efficient trace implementation).
α_v	$\frac{0.01}{289}, \frac{0.01}{449}, \frac{0.01}{609}$
α_w	$0.0001 \times \alpha_v$
α_v	$\frac{0.001}{289}, \frac{0.001}{449}, \frac{0.001}{609}$
$\phi(.)$	The representations for the state-value function, we use tile coding to formulate the feature vector. $n_{start} = 2$ and $n_{end} = 3, 5, 7$. $m_{max} = 3, 5, 7$. Therefore, there are $18, 28, 30$ state variables. State variable is independently tiled with 16 tilings with approximately each with $\frac{1}{16}$ generalization. Therefore, there are $288 + 1, 448 + 1, 608 + 1$ active tiles (i.e., tiles with feature 1) hashed to a binary vector dimension $10^6 + 1$. The bias feature is always active.
$\phi(.,.)$	The representations for the Gibbs distribution, we use tile coding to formulate the feature vector. $n_{start} = 2$ and $n_{end} = 3, 5, 7$. $m_{max} = 3, 5, 7$. Therefore, there are $18, 28, 30$ state variables. A state variable is independently tiled with 16 tilings with approximately each with $\frac{1}{16}$ generalization. Therefore, there are $288 + 1, 448 + 1, 608 + 1$ active tiles (i.e., tiles with feature 1) hashed to a binary vector dimension $10^6 + 1$. The hashing has also considered the given action. The bias feature is always set to active.

4.6 Experiments

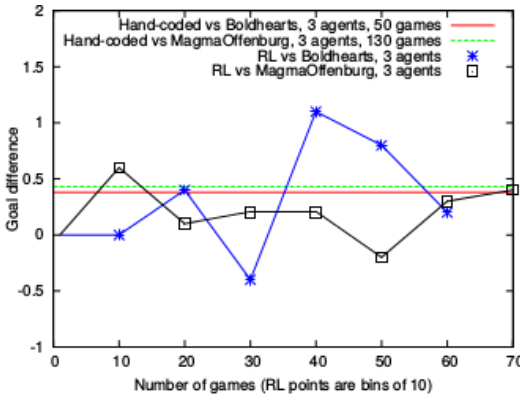
We conducted experiments against the teams Boldhearts and MagmaOffenburg, both semi-finalists of the RoboCup 3D Soccer Simulation competition in Mexico 2012 (the published binary of the team UTAustinVilla showed unexpected behaviors in our tests and is therefore omitted). We conducted knowledge learning according to the configuration given in Section 4.5. Subsection 4.6.1 describes the performance of the Algorithm 2, and Subsection 4.6.2 describes the performance of the Algorithm 3 for the experiment setup.

4.6.1 GVFs with Greedy-GQ(λ)

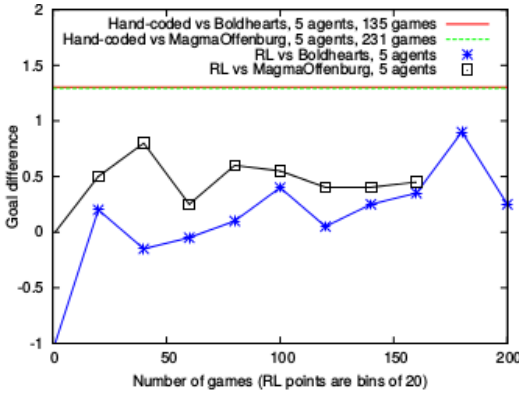
The first experiments were done using a team size of five with the RL agents against Boldhearts. After 140 games our RL agent increased the chance to win from 30% to 50%. This number does not increase more in the next games, but after 260 games the number of lost games (initially 35%) is reduced to 15%. In the further experiments we used the goal difference to compare the performance of the RL agent. Figure 4.3 shows the average goal differences that the hand-tuned role assignment and the RL agents archive in games against Boldhearts and MagmaOffenburg using different team sizes. With only three agents per team the RL agent only needs 40 games to learn a policy that outperforms the hand-coded role selection (Figure 4.3(a)). Also with five agents per team, the learning agent is able to increase the goal difference against both opponents (Figure 4.3(b)). However, it does not reach the performance of the

manually tuned role selection. Nevertheless considering the amount of time spent for fine-tuning the hand-coded role selection, these results are promising. Furthermore, the outcome of the games depends a lot on the underlying skills of the agents, such as walking or dribbling. These skills are noisy, thus the results need to be averaged over many games (std. deviations in Figure 4.3 are between 0.5 and 1.3).

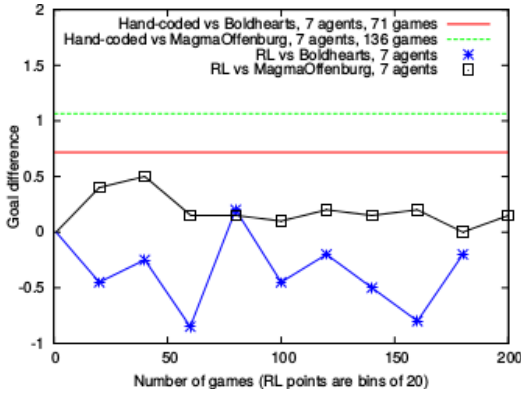
The results in Figure 4.3(c) show a bigger gap between RL and the hand-coded agent. However, using seven agents the goal difference is generally decreased, since the defense is easily improved by increasing the number of agents. Also the hand-



(a) Three vs three agents.



(b) Five vs five agents.



(c) Seven vs seven agents.

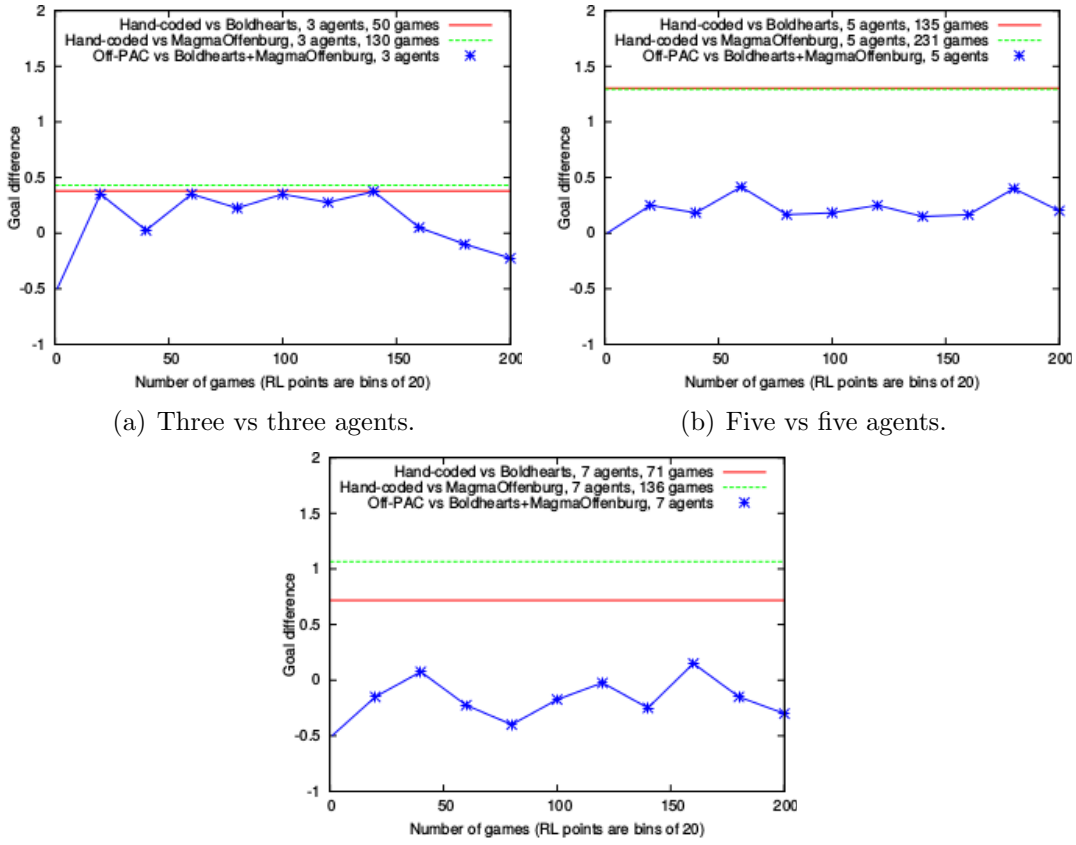
Figure 4.3: Goal difference in games with (a) three; (b) five; and (c) seven agents per team using Greedy-GQ(λ) algorithm.

coded role selection results in a smaller goal difference. Furthermore, considering seven agents in each team the state space is already increased significantly. Only 200 games seem to be not sufficient to learn a good policy. Sometimes the RL agents reach a positive goal difference, but it stays below the hand-coded role selection. In Section 4.7, we discuss some of the reasons for this inferior performances for the team size seven. Even though the RL agent did not perform well considering only the goal difference, it has learned a moderately satisfactory policy. After 180 games the amount of games won is increased slightly from initially 10% to approximately 20%.

4.6.2 GVFs with Off-PAC

With Off-PAC, we used a similar environment to that of subsection (4.6.1), but with a different learning setup. Instead of learning individual policies for teams separately, we learned a single policy for both teams. We ran the opponent teams in a round robin fashion for 200 games and repeated complete runs for multiple times. The first experiments were done using a team size of three with RL agents against both teams. Figure 4.4(a) shows the results of bins of 20 games averaged between two trials. After 20 games, the RL agents have learned a stable policy compared to the hand-tuned policy, but the learned policy bounded above the hand-tuned role assignment function. The second experiments were done using a team size of five with the RL agents against opponent teams. Figure 4.4(b) shows the results of bins of 20 games averaged among three trials. After 100 games, our RL agent increased the chance of winning to 50%. This number does not increase more in the next games. As Figures

4.4(a) and 4.4(b) show, the three and five agents per team are able to increase the goal difference against both opponents. However, it does not reach the performance of the manually tuned role selection. Similar to Subsection 4.6.1, the amount of time spent for fine-tuning the hand-coded role selection, these results are promising, and the outcome of the experiment heavily depends on the underlying skills of the agents.



(a) Three vs three agents.

(b) Five vs five agents.

(c) Seven vs seven agents.

Figure 4.4: Goal difference in games with (a) three; (b) five; and (c) seven agents per team using Off-PAC algorithm.

The final experiments were done using a team size of seven with the RL agents against opponent teams. Figure 4.4(c) shows the results of bins of 20 games averaged among two trials. Similar to subsection 4.6.1, with seven agents per team, the results

in Figure 4.4(c) show a bigger gap between RL and the hand-tuned agent. However, using seven agents the goal difference is generally decreased, since the defense is easily improved by increasing the number of agents. Also the hand-tuned role selection results in a smaller goal difference. Figure 4.4(c) shows an increase in the trend of winning games. As mentioned earlier, only 200 games seem to be not sufficient to learn a good policy. Even though the RL agents reach a positive goal difference, but it stays below the hand-tuned role selection method. Within the given setting, the RL agents have learned a moderately satisfactory policy. Whether the learned policy is satisfactory for other teams needs to be further investigated.

The RoboCup 3D soccer simulation is inherently a dynamic, and a stochastic environment. There is an infinitesimal chance that a given situation (state) may occur for many games. Therefore, it is important that the learning algorithms extract as much information as possible from the training examples. We use the algorithms in the on-line incremental setting, and once the experience is consumed it is discarded. Since we learned from off-policy experiences, we can save the tuples, $(S_t, A_t, S_{t+1}, r(S_{t+1}), \rho_t, z(S_{t+1}))$, and learn the policy off-line. The Greedy-GQ(λ) learns a deterministic greedy policy. This may not be suitable for complex and dynamic environments such as the RoboCup 3D soccer simulation environment. The Off-PAC algorithm is designed for stochastic environment. The experiment shows that this algorithm needs careful tuning of learning rates and feature selection, as evident from Figure 4.4(a) after 160 games.

4.7 Conclusions

We have designed and experimented RL agents that learn to assign roles in order to maximize expected future rewards. All the agents in the team ask the question "*What is my role in this formation in order to maximize future rewards, while maintaining a completely different role from all teammates in all time steps?*" This is a goal-oriented question. We use Greedy-GQ(λ) and Off-PAC to learn experientially grounded knowledge encoded in GVFs. Dynamic role assignment function is abstracted from all other low-level components such as walking engine, obstacle avoidance, object tracking etc. If the role assignment function selects a passive role and assigns a target location, the lower-layers handle this request. If the lower-layers fail to comply to this request, for example being reactive, this feedback is not provided to the role assignment function. If this information needs to be included; it should become a part of the state representation, and the reward signal should be modified accordingly. The target positions for passive roles are created w.r.t. the absolute ball location and the rules imposed by the 3D soccer simulation league. When the ball moves relatively quickly, the target locations change more quickly. We have given positive rewards only for the forward ball movements. In order to reinforce more agents within an area close to the ball, we need to provide appropriate rewards. These are part of reward shaping [64]. Reward shaping should be handled carefully as the agents may learn sub-optimal policies not contributing to the overall goal.

The experimental evidences show that agents are learning competitive role assignment functions for defending and attacking. We have to emphasize that the behavior policy is ϵ-greedy with a relatively small exploration or slightly perturbed around the target policy. It is not a uniformly distributed policy as used in [41]. The main reason for this decision is that when an adversary is present with the intention of maximizing its objectives, practically the learning agent may have to run for a long period to observe positive samples. Therefore, we have used the an off-policy Greedy-GQ(λ) and Off-PAC algorithms for learning goal-oriented GVFs within on-policy control setting. Our hypothesis is that with the improvements of the functionalities of lower-layers, the role assignment function would find better policies for the given question and answer functions. Our next step is to let the RL agent learn policies against other RoboCup 3D soccer simulation league teams. Beside the role assignment, we also contributed with testing off-policy learning in high-dimensional state spaces in a competitive adversarial environment. We have conducted experiments with three, five, and seven agents per team. The full game consists of eleven agents. The next step is to extend learning to consider all agents, and to include methods that select informative state variables and features.

Chapter 5

Representation of and Reasoning with Large-Scale Knowledge Base

High-throughput screening (HTS) is one of the main strategies to identify novel entry points for the development of small molecule chemical probes and drugs and is now commonly accessible to public sector research. Large amounts of data generated in HTS campaigns are submitted to public repositories such as PubChem, which is growing at an exponential rate. The diversity and quantity of available HTS assays and screening results pose enormous challenges to organizing, standardizing, integrating, and analyzing the datasets and thus to maximize the scientific and ultimately the public health impact of the huge investments made to implement public sector HTS capabilities. Novel approaches to organize, standardize and access HTS data are required to address these challenges. We developed the first ontology to describe HTS experiments and screening results using expressive description logic. The BioAssay Ontology (BAO) serves as a foundation for the standardization of HTS assays and data and as a semantic knowledge model. In this chapter we show important examples of formalizing HTS domain knowledge and the advantages of this approach.

5.1 Introductory Remarks

High-throughput screening (HTS) has evolved into an industrialized process and HTS of small molecules is one of the most important strategies to identify novel entry points for drug discovery projects [65]. Until about half a decade ago, HTS and ultra-high throughput screening (uHTS) have been primarily in the realm of the pharmaceutical industry where huge amounts of data have been generated using these technologies. In 2003, NIH started to make HTS and uHTS capabilities accessible to public sector research via the Molecular Libraries Initiative [66] to advance translational research and specifically the Molecular Libraries Program (MLP) [67]. MLP projects leverage innovative assay technologies to develop compounds effective at modulating biological processes or disease states via novel targets. The program has established publicly funded screening centers along with a common screening library (the MLSMR, Molecular Libraries Small Molecule Repository) and data repository, PubChem [68]. Following a pilot phase, the Molecular Libraries Probe Production Centers Network (MLPCN), which consists of four comprehensive and three specialized centers, has been running numerous screening campaigns and has produced a wide range of chemical probes [69]. Since 2004, the MLPCN centers have deposited over two thousand HTS assays testing the effects of several hundred thousand compounds. More recently a European effort, EU Openscreen [70], to establish small molecule screening capabilities is being developed. Besides PubChem there are other data repositories including ChEMBL [71], which includes data curated from the medicinal chemistry

literature, and the Psychoactive Drug Screening Program (PDSP) with mostly receptor and ion channel binding assay results. The MLP is currently the largest public screening effort. The pace with which novel biological assay and HTS results are being submitted suggests that we have only begun to explore the scope of possible assay formats and technologies to interrogate complex biological systems.

Similar to the HTS datasets produced in the pharmaceutical industry, the public sector screening data represent an invaluable resource, which has received wide-spread attention (including from the pharmaceutical companies). However, their diversity and quantity also present enormous challenges to organizing, standardizing, and integrating the data with the goal to maximize their scientific and ultimately their public health impact as the screening results are carried forward into drug development programs.

Despite calls for HTS standards [72], there have been no public initiatives defining minimum specifications, data exchange formats, or a controlled terminology. This situation lies in contrast to other fields such as microarray experimentation, where minimum information specifications (Minimum information about a Microarray Experiment or MIAME 2.0), multiple data models (MicroArray Gene Expression Object Model or MAGE-OM) and the MGED (Microarray and Gene Expression Data) ontology [73] have been developed and incorporated into Web Services such as the Gene Expression Omnibus [74] to facilitate data exchange.

PubChem was set up with flexibility in mind and is able to collect almost any type of assay results. Screened compounds and substances are represented seamlessly

by chemical structure files and pertinent assay data are interlinked to other NCBI resources. However, PubChem has limitations that burden data retrieval and meta-analysis. Foremost is an unstructured/semi-structured data representation format that is largely determined by the submitter. Information regarding assay formats (e.g., cell-based vs. biochemical), readout technologies, reagents employed, and details of the biological system interrogated are represented as free text. This makes it impossible to query PubChem by simple, yet relevant concepts, such as "luciferase reporter gene assays" or "GPCR agonist assays".

To describe compound activities, PubChem uses two terms, Outcome and Score, that have different connotations depending on the submitter. This discrepancy effectively renders quantitative comparisons between assays impossible. Additional terms describing assay results (referred to as assay endpoints in this chapter), such as the half maximal inhibitory concentration (IC50), have different nomenclatures. For example, "JAK2V617F Inhibition (IC50)" (AID 2165), "mutant luminescence Mean IC50" (AID 792), "Best-Fit Value IC50 (uM)" (AID 1916), "IC50_Mean" (AID 2784), "Mean IC50" (AID 1695) are all equivalent endpoints for the purpose of analysis, but are distinct in the repository. This system has led to the accumulation of over 17,000 unique endpoints that cannot be compared without large-scale annotation efforts. In addition to inconsistent naming, there is no semantic description of screening endpoints. In this chapter, we show how the definition of endpoints (such as "IC50") in an ontology with formal semantics facilitates the retrieval of data that are relevant to a search query, but not explicitly defined by the query terms (inferred results).

Ontologies have traditionally been used in biology to organize information within a domain and, to a lesser extent, to annotate experimental data. A successful and highly-used biomedical ontology is the Gene Ontology (GO) [75], which consists of a taxonomy of terms describing gene product localization and function. Several hundred ontologies are hosted by the Open Biological and Biomedical Ontologies (OBO) Foundry [76] as well as the National Center for Biomedical Ontologies , centered on domains ranging from African traditional medicine to Zebrafish anatomy. A closer look reveals that the majority of these ontologies are actually taxonomies or "enriched taxonomies" (with comments for understanding). It has been suggested that the general utility of many of these ontologies is likely overestimated, because terms lack clear semantics and multiple conventions are used to describe overlapping information [77]. In addition, many of the biomedical ontologies so far have not made use of available description logic (DL) features of the Web Ontology Language (OWL), the official ontology language recommended by the World Wide Web Consortium (W3C).

In this chapter we describe a novel approach to standardize, organize and semantically define biological assays and screening results such as those in PubChem, and which addresses many of the challenges raised above. We briefly discuss the main components required to describe important details of bioassay experiments and screening results. We illustrate the architecture of the BioAssay Ontology (BAO) and show examples of how some of the concepts are implemented in BAO to serve as a standard and as a knowledge model. BAO is organized by several main concepts, which describe important characteristics of assays and by which assays can be mean-

ingfully categorized. One of the goals of BAO is to enable the classification of assays by relevant categories so that related assays can quickly be identified, for the purpose of data analysis or assay development. These main categories relate to questions like: i) What type of perturbing agent (perturbagen) was screened? ii) What was the main biological / chemical category (format) of the assay? iii) How was the perturbation converted into a detectable signal? iv) What was the physicochemical method of signal detection? v) What was the biological context (meta target) of the assay? vi) How were the results reported/quantified?

A novel feature of BAO is that it supports inferences within the functionality of OWL 2.0, raising the possibility of automated knowledge acquisition from existing datasets. We present a number of semantic query scenarios that are enabled by BAO. In addition to identifying assays and data by concepts in the ontology, we show how our approach can retrieve inferred results that are highly relevant to a query, but would not match the search term explicitly and therefore could not be easily identified by a classical (relational) search. These type of queries are made possible by the standardization that is provided via BAO and the reasoning/inference capabilities.

5.2 Main Concepts of the BioAssay Ontology and Curation of PubChem Assays

BAO describes biological screening assays, in which the perturbation of a biological system or a component thereof (relative to a reference state) by a perturbagen is

detected and in many cases quantified. An example for a simple assay is the inhibition of an enzyme by a small molecule, which would be detectable by quantifying the product of the enzymatic reaction. For example inhibition of a kinase could be detected via an antibody specific to the phosphorylated substrate (a kinase catalyzes the phosphorylation of a substrate by ATP). In one assay design, the antibody is linked with a fluorescence resonance energy transfer (FRET) donor and the (kinase) substrate with a FRET acceptor. A fluorescence signal of the FRET acceptor is only generated if donor and acceptor are in proximity, i.e. if the substrate is phosphorylated. If the kinase is inhibited by a small molecule perturbagen, the signal decreases. An implementation as homogeneous time resolved FRET (HTRF) assay is applicable to high throughput screening. Countless sophisticated biological screening assays to interrogate simple to complex biological systems have been developed.

With BAO we aim to develop an open standard for the description of HTS and microscopy-based high-content screening (HCS) assays and data for the purpose of classification and analysis. To describe biological screening experiments such as those deposited in PubChem, we first identified the main categories that need to be captured in order to meaningfully compare data from different biological screening experiment. These components are perturbagen, format, design, detection technology, meta target, endpoint, which are described here:

Perturbagen

Assay "perturbagen" refers to the agent that directly interacts or indirectly affects the meta target of a bioassay. PubChem assays predominantly have small molecules as perturbagens; however the concept perturbagen in BAO includes various other perturbing agents, including, nucleic acid (e.g., siRNA, cDNA), lipid, or proteins. Perturbagen specifications include perturbagen source and details on its delivery.

Assay Format

The assay "format" is a higher-level assay category that relates to the biological and chemical features that are common to each test condition in the assay. Assay format includes several broad categories. "Biochemical format" describes assays that are performed with a purified protein, such as the example above. "Cell-based format" relates to assays that are performed with living cells. "Organism-based format" refers to assays with a living organism. Other common formats include "cell-free format", "tissue-based format", and "physicochemical format". Additional format specifications are captured that describe, for example, whether the assay is homogeneous or heterogeneous in nature.

Assay Design and Detection Technology

The assay "design" describes the methodology to report the action of the perturbagen on the target; i.e. how the perturbation is converted into a detectable signal. In BAO, assay design is broadly classified into one of eight categories: "binding re-

porter", "enzyme reporter", "inducible reporter", "morphology reporter", "viability reporter", "redistribution reporter", "conformation reporter", and "membrane potential reporter". We further annotated the readout "detection technology" used in the assays. These annotations fall into one of several categories, including "spectrophotometry", "fluorescence", "luminescence", "label free technology", "scintillation counting", and "microscopy". Further specifications of assay design and detection technology can include the assay kit or detected wavelength.

Assay Meta Target

Assay "meta target" is a description of the component(s) of the biological system that interact with the perturbagen. Meta target can be directly described as a molecular entity (e.g., a purified protein or a protein complex), or indirectly by a biological process or event (e.g., phosphorylation), or a signaling pathway. An important aspect of our meta target annotations is that they are embedded with semantic information (e.g., *"is target of"* only "measure group"; disjointness with classes such as "perturbagen" or "endpoint"). Meta target may be further linked to additional terms and external content, such as a pathway database. One of the goals of describing meta targets is to infer possible molecular targets or perturbagen mechanisms of action based on the analysis of results of many related assays. Meta target specifications include protein modifications, cell lines, or details about the mechanism of ligand-protein interaction.

Assay Endpoint

An assay "endpoint" describes a quantitative or qualitative outcome of the bioassay. The main classes that we identified are "perturbagen concentration"- and "response"-type endpoints. Simple examples include IC50, EC50, CC50 and percent inhibition, percent activation, percent viability, respectively. We conducted two stages of endpoint formalization, the first of which was to standardize the endpoint names in PubChem by manual curation. This reduces the number of different representations of each endpoint concept. In the examples illustrated below we have reduced 85 unique PubChem endpoint representations to 18 standardized endpoints. However, it is not possible (by manual curation) to uniquely describe each endpoint by exactly one representation, because the endpoint concept depends on other assay concepts and can even vary among different perturbagens of the same assay. In BAO, we therefore defined the endpoint concepts semantically using description logic to specify relationships among the endpoint types and other BAO concepts (see below, Ontology-facilitated query examples, example 3). This enables us to retrieve inferred results, which could otherwise not be obtained or would require complex Boolean endpoint queries. An excerpt of BAO around the class assay endpoint is shown in Figure 5.1. For the purpose of demonstrating the semantic querying capabilities facilitated by BAO (which are described below) we curated over 300 bioassays from PubChem and standardized the endpoints using BAO.

5.3 Ontology Outline, Development and Implementation

BAO was designed to describe biological screening experiments and their outcomes by the six main components outlined above, in addition to general assay attributes that do not fall into any of these categories. Each BAO component includes multiple levels of sub-classes and specification classes which are linked via object property relationships to form a knowledge representation. The current version of the ontology is available on our website and at the NCBO bioportal.

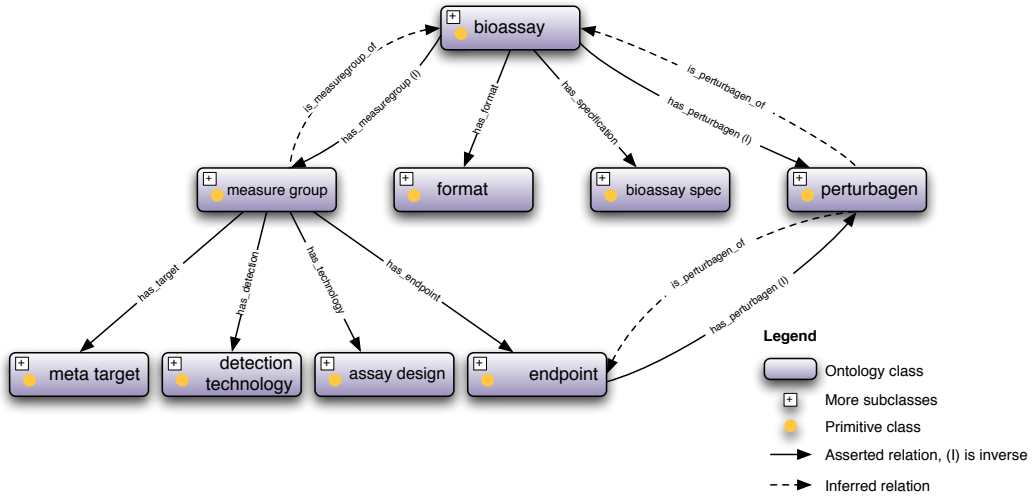


Figure 5.1: BAO excerpt showing the root-level classes and some of their relationships.

Our development approach follows established ontology engineering methodologies using a combination of top-down, domain expert-driven and bottom-up, data-driven approaches [78]. Several external ontologies contain partial information of some of the components of biological assays described by BAO. To leverage these efforts, we have imported into BAO relevant sections from Gene Ontology (GO) [75], Cell Line

Ontology (CLO) [79], Unit Ontology (UO) [80] and others. GO biological process terms and CLO cell line names and additional parameters are used in BAO meta target and meta target specifications. Organism names associated with targets were imported from NCBI taxonomy. Protein target names and IDs were referenced from UniProt. From UO we imported concentration unit and time unit terms. We are currently working on mapping BAO to other OBO ontologies. For example, OBI includes relevant information to describe biological assays.

We have mapped some of the BAO relationships to the OBO Relationship Ontology (RO) and we aim to make more use of RO relationships in the future. Additionally, we may be able to use RO to map BAO concepts to other ontologies, in particular OBI. BAO is rich with a DL expressivity of ALCHOIQ(D). This means that the ontology has the basic S (ALC) expressivity [81] with role hierarchies (H), nominals (O), inverse properties (I), qualified cardinality restrictions (Q), and the use of datatype properties, data values or data types (D). It should be noted that three major bioinformatic terminology bases: SNOMED [82], Galen [83], and GO [75] have the expressivity of EL, with additional role properties. In EL, only intersections between concepts and full existential quantification are possible. In comparison, BAO is a significant improvement in expressivity.

Figure 5.1 illustrates the high-level outline of BAO. It shows the root-level classes, which are described above and general bioassay specifications, and some of their relationships. Some concepts (format, perturbagen and bioassay specifications) are linked directly to bioassay while others (endpoint, meta target, design, detection technology)

are linked via a measure group to accommodate multiplexed and multi-parametric assays. It is also important to note that the assay components are not modeled as sub-classes of bioassay, because they do not have a formal *"is a"* relationship to bioassay. The bioassay component specification classes are not shown. Figure 5.2 shows an excerpt of the BAO classes (and their subsumption hierarchies) that are related to the concept "endpoint". For example Figure 5.2 illustrates the different type of endpoints, such as concentration- and response-type and also the relationships to the specification class, which includes (among others) "endpoint mode of action" with various sub-classes. These concepts are relevant for the semantic querying and reasoning capabilities described in the examples below.

The complete specification in OWL 2.0 can be visually explored and downloaded from our web page (http://www.bioassayontology.org/visualize/). To illustrate how each of these classes is embedded with semantic information, the following example depicts a detailed specification for the class "IC50", defined as the concentration of the perturbagen that results in 50% inhibition.

Equivalent classes

ic50 ≡ (∃ "has has mode of action".inhibition) ⊓

(∀ "has mode of action".inhibition) ⊓

("has percent response" ."50 percent inhibition individual")

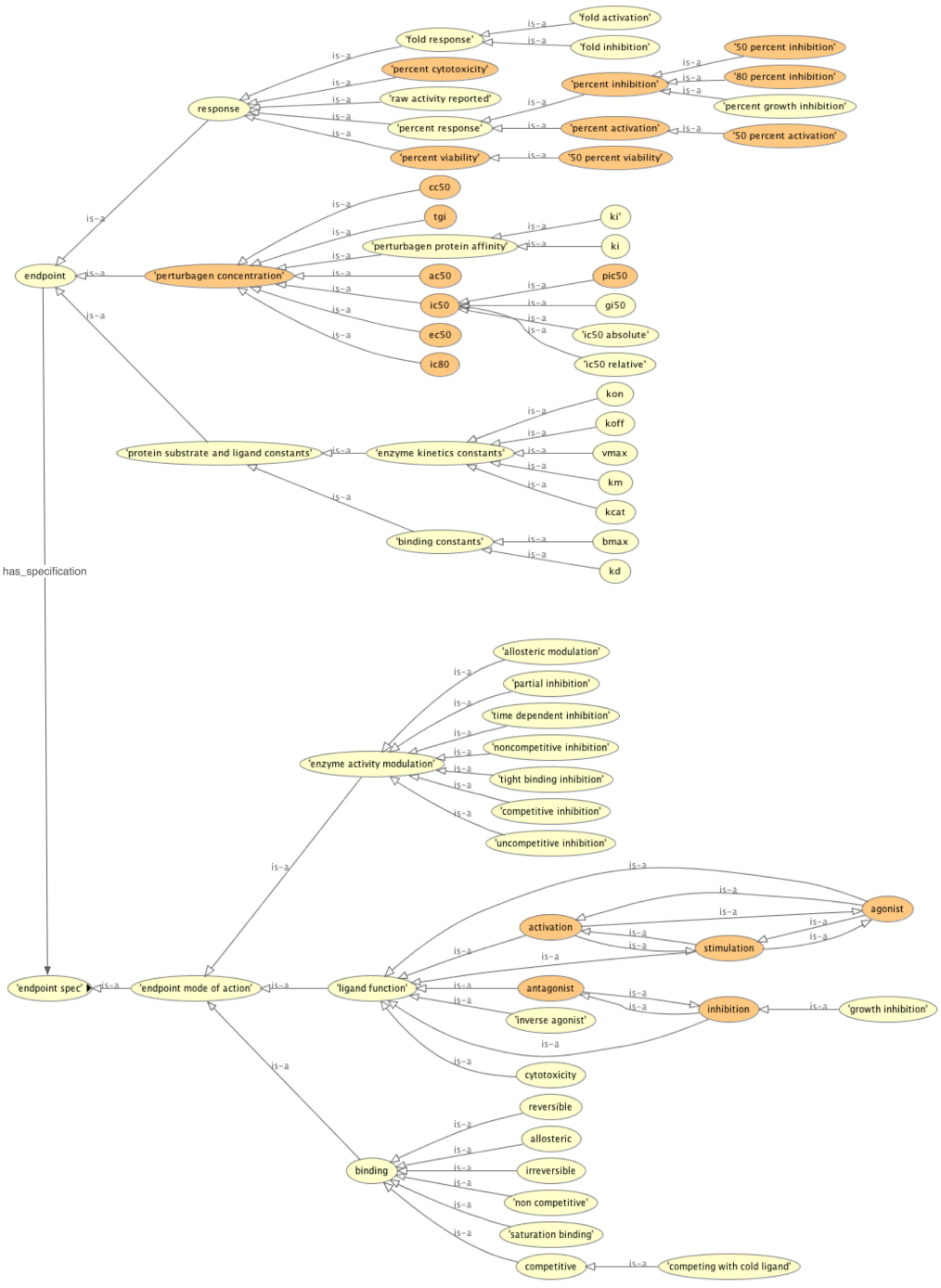


Figure 5.2: A view on some of BAO's concepts, defined as either primitive (light gray/yellow) or defined classes (dark gray/orange).

Inherited anonymous classes

ic50 $\sqsubseteq$ ($\exists$ "has perturbagen concentration unit"."concentration unit") $\sqcap$

($\forall$ "has perturbagen concentration unit"."concentration unit") $\sqcap$

(=1 "has perturbagen concentration value".xsd:float) $\sqcap$

($\forall$ "has specification"."endpoint spec") $\sqcap$

($\exists$ "has perturbagen".perturbagen) $\sqcap$

(=1 "has perturbagen"."$\top$")

Superclasses

ic50 $\sqsubseteq$ ($\forall$ "has curvefit spec"."curvefit spec")

ic50 $\equiv$ "perturbagen concentration"

It is important to note that in OWL 2.0, there are only definitions for equivalent classes (necessary & sufficient conditions), and superclasses (necessary conditions). Necessary and sufficient conditions are used to classify individuals; for example we might be able to infer that an individual endpoint must be an IC50 because the mode of action is inhibition (among other criteria). With only necessary conditions, the definition is logically different, saying that if an individual is a member of the class IC50, it is necessarily a sub-class of "perturbagen concentration". The equivalent class IC50 specifies *"has mode of action"* only "inhibition". "Only" here denotes universal quantification, describing all the individuals whose *"has mode of action"* relationships refer to members of the class inhibition; or conversely, the individuals that do not have *"has mode of action"* relationships to individuals that are not members of the class

"inhibition". There are also existential restrictions that can be seen as "among other things", and are used to close a given property, which is necessary for the reasoning process. The keyword "some" denotes existential restrictions. An example in our ontology is *"has mode of action"* some "inhibition". This specifies the existence of at least one relationship along a given property to an individual, which is a member of the class IC50.

Certain specifications are inherited from classes that are higher up in the class hierarchy. An example of this is the inherited anonymous class definition of individuals having the object property *"has perturbagen concentration value"*. There is also the relationship *"has perturbagen"*, describing that every individual of the IC50 class must have at least one perturbagen.

5.4 Ontology Implementation and Application

The workflow for applying the ontology to real data from PubChem is illustrated in Figure 5.3. First, we have summarized a set of attributes about the assays that needed to be annotated. We have considered >120 attributes (e.g.,, "EndpointStandardized", which takes values of IC50, percent inhibition, fold activation, etc.). These attributes are populated row-by-row in a spreadsheet for the relevant assays using a local mirror of the PubChem data source. A major portion of the spreadsheet is curated manually. In order to compensate for the errors that may have been introduced

during the manual work, we have written a software module to cross-reference each entry in the spreadsheet with the PubChem data source. There were some redundant information among the annotation spreadsheet and data in PubChem, for example screening concentration reported in the assay description (which was manually curated) and the screening concentration deposited to PubChem (which was available in the mirror data source). Some information in the annotation template was explicitly repeated from PubChem in order to correctly map annotated (standardized) terms to data in PubChem, for example to standardize endpoints. This redundancy can be seen as a quality control step to uncover any discrepancies between original and curated data. This step has revealed some inconsistencies in the PubChem database, such as PubChem table entries that are not atomic, incorrect or missing screening concentrations or units; and it has also helped to minimize the errors that had been made throughout the cumbersome curation process.

Second, we have developed a core software module, described as Loader/Bootstrap in Figure 5.3, which reads the curated and quality-checked data and then uses the ontology as well as necessary PubChem data to create a *logical* model of the domain. The reasoning engine Pellet was used, both to create and query the domain model. We also experimented with other DL reasoners, such as HermiT and FaCT++, but used Pellet because of its existing API (Application Programming Interface) that allows interfacing to other software components that we use.

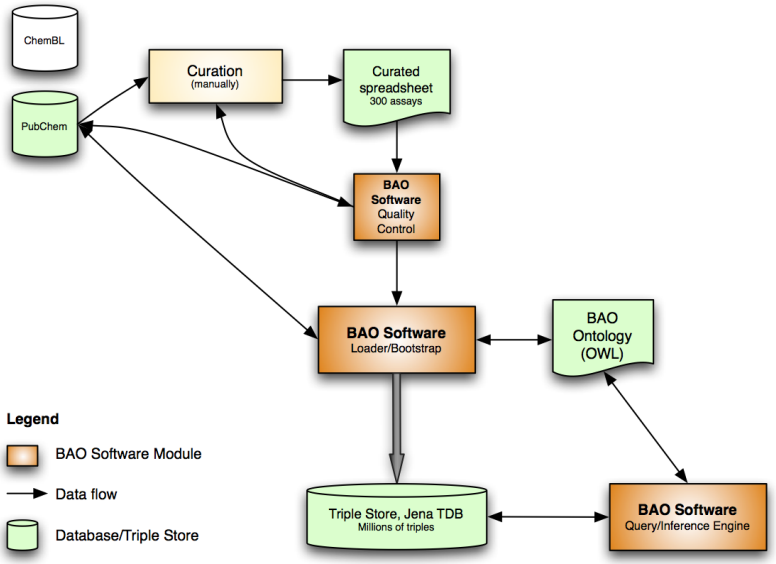


Figure 5.3: BAO software modules (orange/dark gray), documents and databases (light green/light gray).

5.4.1 Curation of Assay Data

In an effort to make the PubChem data amenable to large-scale computational analysis, we manually curated the bioassays. Detailed information were captured from each individual assay based on BAO classes, which fall into the main categories format, meta target, design, detection technology, perturbagen (at this point we only consider small molecule compounds), endpoint, and general assays characteristics. The annotations from each assay were populated in a spreadsheet, cross checked, and then loaded onto a triple store after merging with the relevant PubChem endpoint data using the ontology as described above. In addition to the bioassays run at the MLPCN, PubChem houses data from other sources. We are in the process of incorporating these datasets into BAO.

5.4.2 Searching Facade

The ontology was used to facilitate the featured search queries. Our "BAOSearch" is an application for querying, viewing, browsing and downloading diverse high-throughput screening (HTS) data for drug discovery and related life science research. BAOSearch is a multi-tier, web-based, AJAX-enabled application written primarily in Java and built following a Restful [84] web services paradigm. The service-based aspect of the architecture allows the user interface (UI) to be separated from storage and manipulation of the data, and provides well-defined interfaces for UI components to access and manipulate application data. This separation of application components creates the potential of developing multiple UIs that access the same service, but which render the data differently, or run on different platforms (e.g., browsers, mobile applications). This architecture also creates an opportunity for other software applications (not only UIs) to access the system to query and retrieve data. The browser-based UI was built using JSP and JavaScript, with components from several JavaScript libraries including jQuery. All data were stored in a MySQL database. SDB was used as the triple-store. Other data required by the application was stored in a relational schema accessible using Hibernate. Figure 5.4 shows the high-level architecture of the BAOSearch project.

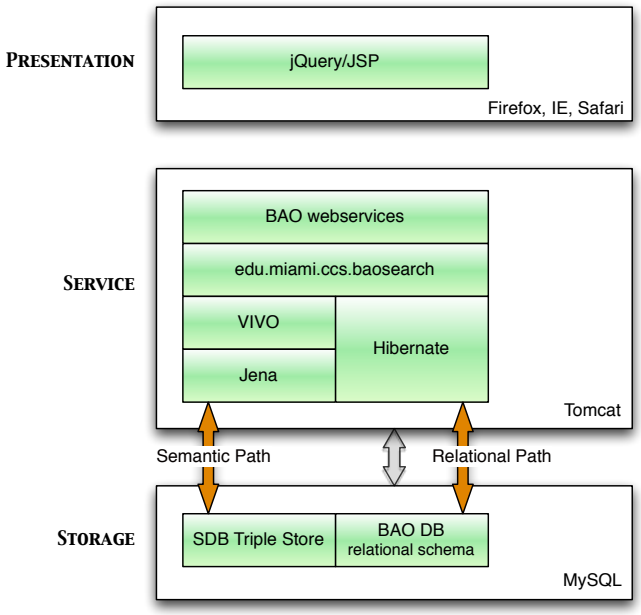


Figure 5.4: High-level architecture of BAOSearch

5.5 Ontology-facilitated Query Examples

We performed a series of experiments based on 194 out of the 300 curated PubChem bioassays that had the (standardized) endpoint terms IC50, EC50, AC50, percent activation, percent stimulation and percent inhibition. Since the entire set of assays and endpoints would have required >17 GB worth of RDF triples, we decided to limit the amount of considered endpoints to 20 for performance reasons. Future versions of the software will focus on optimization and the use of additional annotations. With 20 endpoints, the software generated 45,075 triples (asserted ontology + triple database) in the Jena store. All example queries can be found and tested online at http://baoquery.ccs.miami.edu/joseki/query.html. The reasoner classifies the individuals and SPARQL allows an efficient search through this inferred graph.

Example 1

This example illustrates a common query for compounds with an IC50 value of less than a certain cutoff (here $\leq$10 μM). Such a query should also return results of differently named IC50 endpoints (e.g., AC50), which a user may not know exist. A user querying the database may also be interested in returning other relevant endpoints, such as IC80 values $\leq$10 μM (if they existed in the repository) or other result types such as potent inhibitors screened at less than the IC50 concentration. With the semantic definition of IC50 above, we can achieve both. Query: return all compounds from assays with an inhibitory mode of action and that have a percentage response of 50% or greater at $\leq$10 μM screening concentration.

The SPARQL query was the following:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX bao: <http://www.bioassayontology.org/bao#>

# results
SELECT DISTINCT ?compound ?endpoint ?type ?responseValue
                ?screeningConc ?assay
WHERE
{
  # from endpoints
  ?endpoint rdf:type bao:BAO_0000179 .
  ?endpoint bao:BAO_0000196 ?inhibition .
  # has a mode of action inhibition
  ?inhibition rdf:type bao:BAO_0000091 .
```

```
  # perturbagen concentration endpoint
  ?endpoint bao:BAO_0000336 ?screeningConc .
  # has concentration unit micro molar
  ?endpoint bao:BAO_0000183 bao:BAO_0000107 .
  # has percent response
  ?endpoint bao:BAO_0000337 ?percentResponse .
  ?percentResponse bao:BAO_0000195 ?responseValue .
  ?endpoint rdf:type ?type .
  ?type rdfs:subClassOf bao:BAO_0000180 .
  # response endpoint
  UNION
  {
    ?endpoint bao:BAO_0000196 ?inhibition .
    ?inhibition rdf:type bao:BAO_0000091 .
    ?endpoint bao:BAO_0000338 ?pert .
    ?pert bao:BAO_0000183 bao:BAO_0000107 .
    ?pert bao:BAO_0000336 ?screeningConc .
    ?pert bao:BAO_0000183 bao:BAO_0000107 .
    ?endpoint bao:BAO_0000195 ?responseValue .
    ?endpoint rdf:type ?type .
    ?type rdfs:subClassOf bao:BAO_0000181 .
  }
  ?endpoint bao:BAO_0000185 ?compound .
  ?endpoint rdf:type ?type .
  ?assay bao:BAO_0000209  ?measureGroup .
  ?measureGroup bao:BAO_0000208  ?endpoint .
  # screening concentration <= 10 micro molar
  #   && percent response >= 50%
  FILTER(?screeningConc <= 10 && ?responseValue >= 50)
}
```

The BAO software returns 2,741 SPARQL endpoint results from the inferred model residing in the triple store, 4 of which are shown below for illustrative purposes. All results are individuals with a working internal resource identifier (IRI), which corresponds to a URI, but is valid only internally. IRIs are abbreviated due to space limitations, but all complete IRIs are available via `http://baoquery.ccs.miami.edu/joseki/query.html`

```
   (5)(?compound=<bao#individual_BAO_0000021_2858522>)
      (?endpoint=<bao#individual_BAO_0000190_2_2357>)
      (?type=<bao#BAO_0000190>)
      (?responseValue="50"^^xsd:float)
      (?screeningConc="4.0"^^xsd:float)
      (?assay=<bao\#individual_BAO_0000015_1293>)
  (17)(?compound=<bao#individual_BAO_0000021_133407>)
      (?endpoint=<bao#individual_BAO_0000190_2_2533>)
      (?type=<bao#BAO_0000190>)
      (?responseValue="50"^^xsd:float)
      (?screeningConc="8.59"^^xsd:float)
      (?assay=<bao#individual_BAO_0000015_2409>)
  (24)(?compound=<bao#individual_BAO_0000021_11057>)
      (?endpoint=<bao#individual_BAO_0000190_2_4122>)
      (?type=<bao#BAO_0000186>)
      (?responseValue="50"^^xsd:float)
      (?screeningConc="6.3096"^^xsd:float)
      (?assay=<bao#individual_BAO_0000015_948>)
(2690)(?compound=<bao#individual_BAO_0000021_657680>)
      (?endpoint=<bao#individual_BAO_0000201_1_1670>)
      (?type=<bao#BAO_0000201>)
      (?responseValue="63.48"^^xsd:float)
      (?screeningConc="4.0"^^xsd:float)
      (?assay=<bao#individual_BAO_0000015_1834>)
```

Results are shown by their unique IRIs, e.g., the first result contains the compound ID (CID) 2858522 of an individual of the class perturbagen (BAO_0000021). The SPARQL query also selects for the endpoints of the perturbagens that fulfill the activity criteria. The query retrieves results that classify as specific types of endpoints (subsumption reasoning). Result (5) (CID 2858522, AID 1293) was found because IC50 (note, that in PubChem AID 1293 this endpoint has been incorrectly reported as EC50; we corrected this during the curation process) (BAO_0000190) *is_a* perturbagen concentration-type endpoint (as defined above). Result (17) (CID 133407,

AID 2409) also returns IC50. Result (18) (not shown) returns the same data as AC50 concordant with the (inferred) subsumption hierarchy (compare Figure 5.5b). Querying AC50 (instead of IC50) thus would also retrieve this result. Result (24) (CID 11057, AID 948) is an AC50 endpoint (named "potency" in PubChem); result (23) returns the same data as IC50 (not shown) - again consistent with the inferred class hierarchy. Result (2690) (CID 657680, AID 1834) is a percentage inhibition endpoint (63.5 %) and the screening concentration is 4 μM (i.e. less than the query 10 μM). These different types of results can be retrieved because of the subsumption reasoning of the DL engine using formally defined endpoints. This example illustrates that with the endpoint definition in BAO, we can identify and return relevant query results, which are not restricted to a specific endpoint type or endpoint representation (that is specified by the query), as it would typically be the case in a relational system.

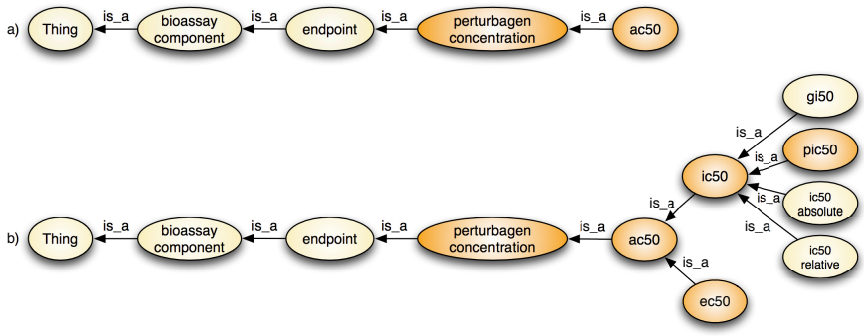


Figure 5.5: a) Asserted logical taxonomy for AC50 (above) and b) Inferred logical taxonomy, where IC50 is classified as a sub-class of AC50.

Example 2

We illustrate an example of constructive reasoning in identifying compounds of a particular pharmacological action. Query: return all assays with compounds that have a mode of action "activation" and show a percentage response of $\geq$ 50 % at $\leq$10 μM screening concentration.

The query syntax was the following (we are omitting the PREFIX section this time):

```
SELECT DISTINCT ?compound ?endpoint ?type ?moaType
                ?responseValue ?screeningConc ?assay
WHERE
{
  # from endpoints
  ?endpoint rdf:type bao:BAO_0000179 .
  ?endpoint bao:BAO_0000196 ?activation .
  # has a mode of action activation
  ?activation rdf:type bao:BAO_0000087 .
  ?activation rdf:type ?moaType .
  ?moaType rdfs:subClassOf bao:BAO_0000084 .
  # perturbagen concentration endpoint
  ?endpoint bao:BAO_0000336 ?screeningConc .
  # has concentration unit micro molar
  ?endpoint bao:BAO_0000183 bao:BAO_0000107 .
  # has percent response
  ?endpoint bao:BAO_0000337 ?percentResponse .
  ?percentResponse bao:BAO_0000195 ?responseValue .
  ?endpoint rdf:type ?type .
  ?type rdfs:subClassOf bao:BAO_0000180 .
  # response endpoint
  UNION
  {
    ?endpoint bao:BAO_0000196 ?activation .
    ?activation rdf:type bao:BAO_0000087 .
    ?activation rdf:type ?moaType .
    ?moaType rdfs:subClassOf bao:BAO_0000084 .
```

```
    ?endpoint bao:BAO_0000338 ?pert .
    ?pert bao:BAO_0000183 bao:BAO_0000107 .
    ?pert bao:BAO_0000336 ?screeningConc .
    ?pert bao:BAO_0000183 bao:BAO_0000107 .
    ?endpoint bao:BAO_0000195 ?responseValue .
    ?endpoint rdf:type ?type .
    ?type rdfs:subClassOf bao:BAO_0000181 .
  }
  ?endpoint bao:BAO_0000185 ?compound .
  ?endpoint rdf:type ?type .
  ?assay bao:BAO_0000209  ?measureGroup .
  ?measureGroup bao:BAO_0000208  ?endpoint .
  # screening concentration <= 10 micro molar &&
  # percent response >= 50%
  FILTER(?screeningConc <= 10 && ?responseValue >= 50)
}
```

Similar to example 1, the system returns different types of relevant results. In addition to assays with compounds that have an endpoint "percent activation" of 50% at <10 μM, this query also returns assays with an EC50 or an AC50 value of <10 μM. Moreover, this example demonstrates one of the constructive reasoning mechanisms in BAO where "activation" was defined as *equivalent* to "stimulation" (among other equivalent classes, e.g., agonist). As the reasoning system returns results that satisfy the original query and the inferred query, searching "activation" (BAO_0000087) returns exactly the same results as querying for "stimulation" (BAO_0000093) independent from the specific term used to describe the pharmacological action. Selected results are:

```
   (1)(?compound=<bao#individual_BAO_0000021_653469>)
      (?endpoint=<bao#individual_BAO_0000188_2_5524>)
      (?type=<bao#BAO_0000180>)
      (?moaType=<bao#BAO_0000087>)
```

```
      (?responseValue="50"^^xsd:float)
      (?screeningConc="2.154"^^xsd:float)
      (?assay=<bao#individual_BAO_0000015_695>)
   (5)(?compound=<bao#individual_BAO_0000021_653469>)
      (?endpoint=<bao#individual_BAO_0000188_2_5524>)
      (?type=<bao#BAO_0000188>)
      (?moaType=<bao#BAO_0000093>)
      (?responseValue="50"^^xsd:float)
      (?screeningConc="2.154"^^xsd:float)
      (?assay=<bao#individual_BAO_0000015_695>)
(5130)(?compound=<bao#individual_BAO_0000021_645132>)
      (?endpoint=<bao#individual_BAO_0000200_1_464>)
      (?type=<bao#BAO_0000181>)
      (?moaType=<bao#BAO_0000087>)
      (?responseValue="132.52"^^xsd:float)
      (?screeningConc="5.7"^^xsd:float)
      (?assay=<bao#individual_BAO_0000015_1318>)
(5131)(...)
```

The first result (1) refers to AID 695. As before, the formal definition of "mode of action" in the ontology and the reasoning system make it possible to retrieve relevant results by inference, which could not be returned from a relational database system (e.g., agonist if one searched for activation).

Example 3

We demonstrate a specific case concerning three concepts: endpoint, bioassay, and perturbagen. Figure 5.6 shows the relevant relationships between these concepts (note: the concept "measure group" exists to accommodate multiplexed assays; it is not used in this example); it is a more detailed representation of some of the concepts in Figure 5.1. Of particular interest is the relation *"has perturbagen"* that holds between endpoint and perturbagen as well as bioassay and perturbagen. The ontology specifies that this property has an *inverse* relationship with *"is perturbagen of"*. Here we show how these relationships (with their characteristics) are used to retrieve eligible instances (individuals) by inference. This reasoning mechanism thus makes it possible to retrieve perturbagens based on more complex concepts, for example a class of promiscuous compounds (compounds that are active in several assays - see below).

To illustrate this, we queried for all perturbagens that have a percentage response of ≥50 % in at least three assays. The SPARQL query was as follows:

```
SELECT ?pert
WHERE
{
  ?pert rdf:type bao:BAO_0000021 .
  ?pert bao:BAO_0000361 ?assay .
  ?assay bao:BAO_0000209 ?measureGroup .
  ?measureGroup bao:BAO_0000208 ?endpoint .
  ?endpoint bao:BAO_0000195 ?percentResponseValue .
  UNION
  {
```

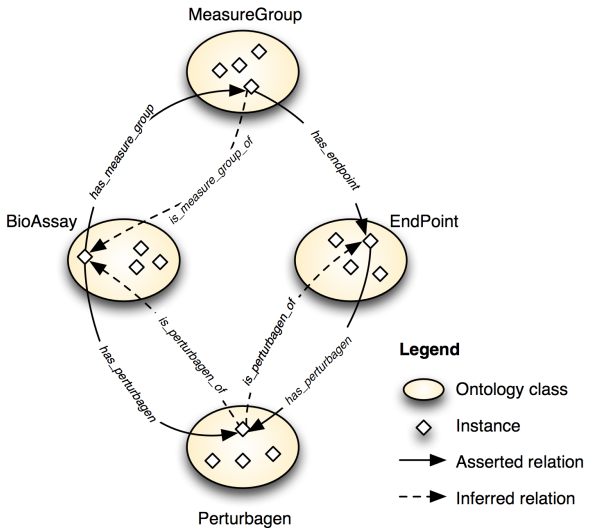


Figure 5.6: Relationships between BioAssay, EndPoint, and Perturbagen.

```
    ?pert rdf:type bao:BAO_0000021 .
    ?pert bao:BAO_0000361 ?assay .
    ?assay bao:BAO_0000209 ?measureGroup .
    ?measureGroup bao:BAO_0000208 ?endpoint .
    ?endpoint bao:BAO_0000337 ?percentResponse .
    ?percentResponse bao:BAO_0000195
                          ?percentResponseValue .
  }
  FILTER (?percentResponseValue >= 50)
}
GROUP BY ?pert
HAVING (count(distinct ?assay) >= 3)
```

In this query, we used the inferred relation *"is perturbagen of"*, which points to either an endpoint or a bioassay. The query separately checked for bioassay instances and endpoint instances. The syntax allowed for the expression of the notion of "at least" in a simple way. Specifically, we used the syntactic extensions available in the ARQ SPARQL implementation. The "GROUP BY" extended clause grouped the unique "?pert" result set (?pert is a variable here) in a row-by-row basis. The

"HAVING" clause applied the filter "count(distinct ?assay))" to the result set after grouping. The results of the query were as follows. First, we queried for the compound and obtained:

```
(1)(?pert=<bao#individual_BAO_0000021_646704>)
```

We then used this result (bao:individual_BAO_0000021_646704) for the next query:

```
SELECT ?assay ?percentResponseValue
WHERE
{
  bao:individual_BAO_0000021_646704 bao:BAO_0000361
                                                    ?assay .
  ?assay bao:BAO_0000209 ?mg .
  ?mg bao:BAO_0000208 ?endpoint .
  bao:individual_BAO_0000021_646704 bao:BAO_0000361
                                                  ?endpoint .
  ?endpoint bao:BAO_0000195 ?percentResponseValue .
  UNION
  {
    bao:individual_BAO_0000021_646704 bao:BAO_0000361
                                                    ?assay .
    ?assay bao:BAO_0000209 ?mg .
    bao:individual_BAO_0000021_646704 bao:BAO_0000361
                                                   ?endpoint.
    ?endpoint bao:BAO_0000337 ?percentResponse .
    ?percentResponse bao:BAO_0000195
                                       ?percentResponseValue .
  }
  FILTER (?percentResponseValue >= 50)
}
```

```
(1)(?assay=<bao#individual_BAO_0000015_1262>)
   (?percentResponseValue="116.84"^^xsd:float)
(2)(?assay=<bao#individual_BAO_0000015_1306>)
   (?percentResponseValue="106.48"^^xsd:float)
(3)(?assay=<bao#individual_BAO_0000015_1316>)
   (?percentResponseValue="99.42"^^xsd:float)
```

Example 3 is a simple illustration to identify compounds with a specific profile (here, active in three assays). The query actually retrieved inferred information, facilitated by the *inverse* relationship *"is perturbagen of"*. Further specification of this query, e.g., by BAO meta target or design sub-classes, would allow to quickly identify individuals based on more complex concepts, for example compounds that are promiscuously active in assays of a specific design and which are therefore likely artifacts.

The three query examples illustrate some of the features that can be used in complex search queries with an underlying DL-based ontology. Other features such as role hierarchies, quantifiers, nominals etc. were also used in our ontology.

5.6 Conclusion

We have developed an ontology to describe biological assay and screening results. The BioAssay Ontology (BAO) provides a foundation for standardizing assay descriptions and endpoints and serves as a knowledge model by describing screening experiments and results semantically using DL. BAO facilitates semantic search capabilities enabling the retrieval of data that are relevant to a query and that could

not be readily obtained otherwise. 300 PubChem assays were curated and 194 were loaded into a triple store to demonstrate various search scenarios. This is the first ontology to describe this domain, and certainly the first time that bioassay and HTS data have been represented using expressive description logic. There are numerous advantages to this approach; most importantly it opens new functionality for querying and analyzing HTS datasets and the potential for discovering knowledge that is not explicitly represented, by inference. We demonstrated these novel capabilities and their benefits by three simple examples of how specific features of our approach can be implemented. One of the examples illustrated a query for (inferred) perturbagens with a defined activity profile. As BAO includes class hierarchies for target, design, detection technology, etc., perturbagen sub-classes of interest may be directly defined in the ontology using the same approach; e.g., "compounds promiscuously active in luciferase reporter gene assays". Using a reasoning engine, the individuals that are members of such a class could be automatically inferred among the currently annotated assays. We are continuing to refine and extend the BAO and supporting software. We have already created a web portal with an easy-to-use querying interface that incorporates some of the described functionality. A user can query PubChem data using BAO terminology and collect sets of results for further analysis. It also allows end users to formulate their own queries via a graphical user interface. Future developments will include an annotation tool for domain experts that will aid in the curation process and the incorporation of additional data sources.

Chapter 6

Ontology Modularization for Large-Scale Knowledge Base

The lack of established standards to describe and annotate biological assays and screening outcomes in the domain of drug and chemical probe discovery is a severe limitation to utilize public and proprietary drug screening data to their maximum potential. We have created the BioAssay Ontology (BAO) in Chapter 5 to develop common reference metadata terms and definitions required for describing relevant information of low- and high- throughput drug and probe screening assays and results. The main objectives of BAO are to enable effective integration, aggregation, retrieval, and analyses of drug screening data. Since we first released BAO on the BioPortal in 2010 we have considerably expanded and enhanced BAO and we have applied the ontology in several internal and external collaborative projects, for example the BioAssay Research Database (BARD). We describe the evolution of BAO with a design that enables modeling complex assays including profile and panel assays such as those in the Library of Integrated Network-based Cellular Signatures (LINCS). One of the critical questions in evolving BAO is the following: how can we provide

a way to efficiently reuse and share among various research projects specific parts of our ontologies without violating the integrity of the ontology and without creating redundancies. This chapter provides a comprehensive answer to this question with a description of a methodology for ontology modularization using a layered architecture. Our modularization approach defines several distinct BAO components and separates internal from external modules and domain-level from structural components. This approach facilitates the generation/extraction of derived ontologies (or perspectives) that can suit particular use cases or software applications. We describe the evolution of BAO related to its formal structures, engineering approaches, and content to enable modeling of complex assays and integration with other ontologies and datasets.

6.1 Introductory Remarks

The new BAO 2.0 formally describes perturbation bioassays in the domain of drug and probe discovery, such as small molecule HTS assays and screening results for the purpose of categorizing the assays and outcomes by concepts that relate to the screening model system (format), assay method, the biology interrogated in the assay (such as a protein target or biological process), the detection method (how does the assay work), and types of results (endpoints). BAO 2.0 is organized into several major sections, which include multiple levels of subcategories of subsumption class hierarchies. A number of specific object property relationships were created to connect the classes and develop a knowledge representation.

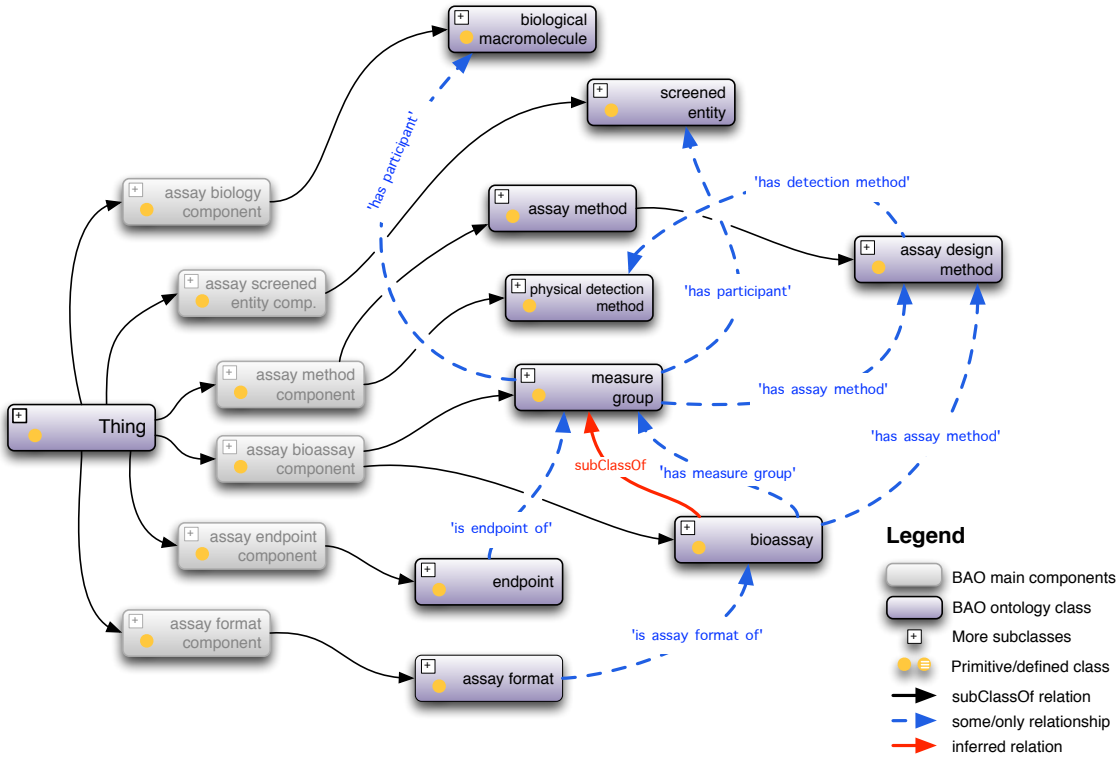


Figure 6.1: BAO 2.0 main classes with some relationships between them. Six main components (shaded classes) are used to formally describe bioassays by terms related to bioassay, biology, screened entity, assay method, format, and endpoint. The most important classes and their relations as shown including bioassay, measure group, biological macromolecule, screened entity, assay method (specifically assay design method and physical detection method), assay format, and endpoint. There exists complex interactions among these entities. OWL DL 2 ($\mathcal{SROIQV}^D$), the decidable subset of the first-order-predicate logic provides the interpretations, models, and logical consequences.

The main categories in BAO 2.0, titled components, include bioassay, assay biology, assay method, assay format, assay endpoint, assay screened entity (Figure 6.1). Each of these component classes includes the subsumption trees of terms corresponding to the category and additional trees of related terms to describe each of the main components properly and formally. In BAO 2.0, we incorporated a slightly different pattern from BAO 1.6, since we were interested in making BAO 2.0 compatible with the existing

upper-level and other domain-level ontologies. The BAO 2.0 categories also lend BAO to its native structures that is most useful to users, for example to annotate assays or to implement a user interface in a software application. We describe briefly the main class hierarchies of BAO 2.0 corresponding to the above components (Figure 6.1):

- BAO assay bioassay component includes the bioassay subsumption tree, and several other classes to describe assays, including assay kit, bioassay type, and bioassay specification, which contains terminology trees to describe various details about a bioassay and its context. The class hierarchy bioassay includes the list of the bioassays and their formal description, e.g., cell cycle assay, enzyme activity assay. Bioassays are organized roughly by their application (what the assay is used for). The class hierarchy assay kit includes the reagents and their cocktails that are commercially available to perform the different chemical reactions that encompass an assay (i.e., out of the box, ready to run assays). The information in bioassay specification is similar to BAO 1.6.

- BAO assay format component includes the assay format subsumption tree to describe the biological model system; a conceptualization of assays based on the biological and/or chemical features of the experimental system.

- BAO assay method component includes terminologies to describe how the assay is performed, most importantly assay method and physical detection method. It also includes computational method, instrument, and relevant other material entity "assay ingredients". The class hierarchy assay method includes assay de-

sign method and assay supporting method; assay design method describes how a biological perturbation of the model system is translated into a detectable signal. The class hierarchy computational method contains various methods that are based on the application of information technology to chemistry and biology. The physical detection method hierarchy includes the method (technology) used to detect the signal that corresponds to the perturbagen in the assay environment and enabled by the assay design method. Class instrument consists of instruments used for detection/readout from an assay and their components, e.g., FLIPR, ViewLux plate reader, PHERAstar, etc; software lists the types of software that are used in the various instruments, e.g., image analysis software, which is a component of the high content screening (HCS) platforms.

- BAO assay biology component includes various class hierarchies to describe the biology of the assay including biological process, biological macromolecule, cell line cell, cellular component, cell phenotype, anatomical entity, disease, function, organism. Many of these are mapped to external sources (*vide infra*). To describe the biology of a simple binding assay for example, a biological macromolecule protein would have the biological role target. Many other role classes exist (*vide infra*). The class function includes the physiological function of biological macromolecules, e.g., protein binding, kinase activity. This module was imported from the Gene Ontology (GO). The class cellular phenotype encompasses both the molecular characteristics of a cell and the (morphological) shape and structure of a cell and its parts.

- BAO assay screened entity component includes screened entity, which is the chemical or biological entity that is tested/screened in the assay. The screened entity typically modulates the function of the (known or unknown) biological macromolecule with the role of a target. The most important screened entity for BAO is the class small molecule, that contains compounds that are tested in the process of developing chemical probes and drugs, which is the primary domain of BAO.

- BAO assay endpoint component includes subsumption trees to describe the assay result or endpoint and other required information to quantitatively or qualitatively express the biological perturbation measured in a bioassay, such as units of measurement (imported from UO), and other details to interpret the results in the context of the assay methodology and the biology, such as as the mode of action of the perturbagen that the endpoint characterizes, or the signal direction and endpoint action correlation of the assay. More details about the class endpoint are described below.

- Additional classes that were not assigned to any one of the main BAO components are organization, people, role, and quality: Organization includes, for example manufactures of assay kits, instruments, etc., or screening center where assays are performed. People include the individuals who are involved in performing scientific research, such as assay development, compound screening, chemical synthesis, etc. Role describes the action that an entity performs in a

given context; an entity can have more than one role, e.g., target, perturbagen. BAO 2.0 has imported roles from the Chemical Entities of Biological Interest (ChEBI) ontology and we have added some missing classes. Quality lists the characteristics that inhere in an entity of biological origin, namely, organism, cell, and molecule or a physical entity, e.g., intensity, optical quality. Most of the terms in this class were imported from the Phenotypic Quality Ontology (PATO); missing ones were added to BAO.

- BAO properties include both the object and data types that are required to create relationships among the different concepts in BAO 2.0. These properties were either imported from the Relationship Ontology (RO), where available or created in BAO 2.0.

6.2 Upper Level Ontology Structure and Aligning External Ontologies

Since, there are several advantages of using upper level ontologies (ULOs), BAO 2.0 makes use of the Basic Formal Ontology (BFO) and OBO Relations Ontology (OBO-RO) as its upper level ontologies. We have used the current release of BFO 2.0 ontology (http://purl.obolibrary.org/obo/bfo.owl), which is also tightly coupled with OBO-RO ontology (http://purl.obolibrary.org/obo/ro.owl). Figure 6.2 shows the main categories of BFO and examples of corresponding BAO 2.0 classes. BFO conceptualization abstractly represents objects, entities, and relations in our domain

of discourse, and it is substantially used in biomedical ontologies compared to other OWL version of ULOs such as SUMO (http://www.ontologyportal.org/SUMO.owl) or DOLCE (http://www.loa.istc.cnr.it/ontologies/DLP_397.owl). The advantage of using an ULO is that it allows integration of existing domain ontologies, by grounding them on a formally rigid ontological framework [85, 86]. We make available a development instance of the BAO BFO version. Figure 6.2 also illustrates external ontologies, components of which we currently use in BAO. Their alignment with BAO is facilitated in part by the BFO structure [87, 88]. One important mid level ontology is the Ontology for Biomedical Investigations (OBI) [89]. We have previously outlined the different focus of BAO vs. OBI [90]. However, this is not to say that they are incompatible; alignment is one of the future tasks required to evolve BAO further. We have created a version of BAO 2.0 that contains BFO and OBO-RO as ULOs (bao_complete_bfo_dev, which is a development version) and another one without them (bao_complete, released). bao_complete_bfo_dev simplifies alignments to external ontologies and is targeted to the ontology development community while bao_complete is targeted to the drug and probe screening community and developers of software applications (such as our BAOSearch application). We emphases the fact that the BAO-to-BFO alignment is based on our knowledge and understanding of BFO and OBO-RO structures, and BAO mechanisms. The alignment is an ongoing process, and we have a community wide bug reporting system to uses to provide feedback to provide semantically better alignments. bao_complete_bfo_dev and bao_complete are targeted towards different users groups, the latter is more amenable to perform on

large-scale analysis of the chemical biology data without the additional constraints imposed by BFO and OBO-RO.

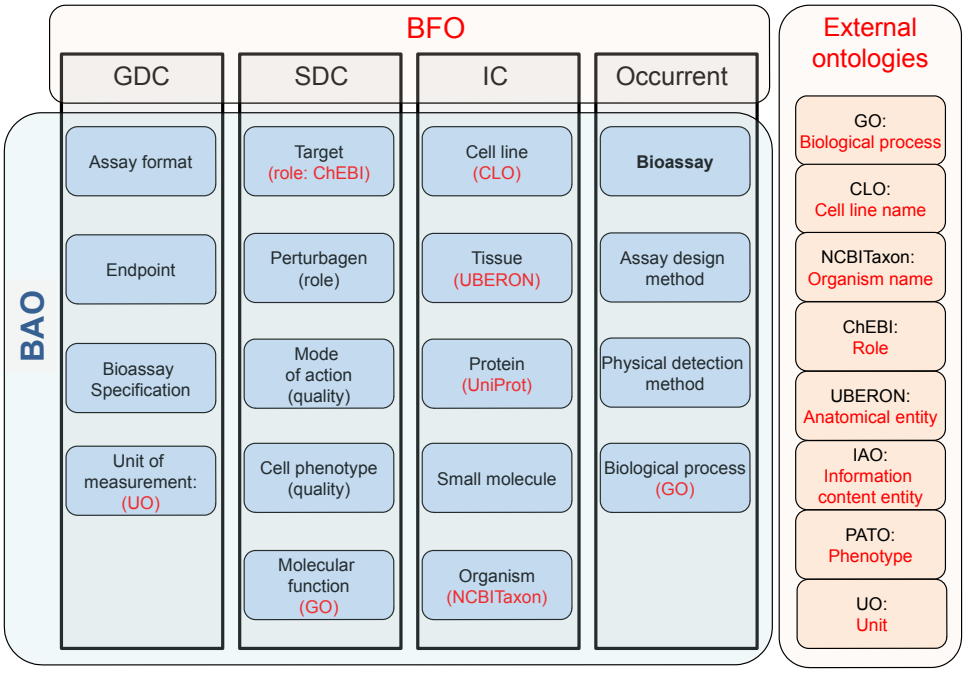


Figure 6.2: BAO 2.0 makes use of BFO as the upper-level ontology and incorporates several external ontology modules. BAO classes were mapped under appropriate BFO concepts. The BFO framework also facilitates alignment to external ontology modules. Blue boxes are examples of BAO classes categorized by BFO (black rectangles). External ontologies used in BAO are shown as red boxes and labels.

6.3 Modular Architecture and Implementation

Our modularization approach is illustrated in Figure 6.4. The modularization framework uses a layered architecture and uses the modeling primitives, vocabularies, modules and axioms. Vocabularies only contain terms (classes with subsumption only).

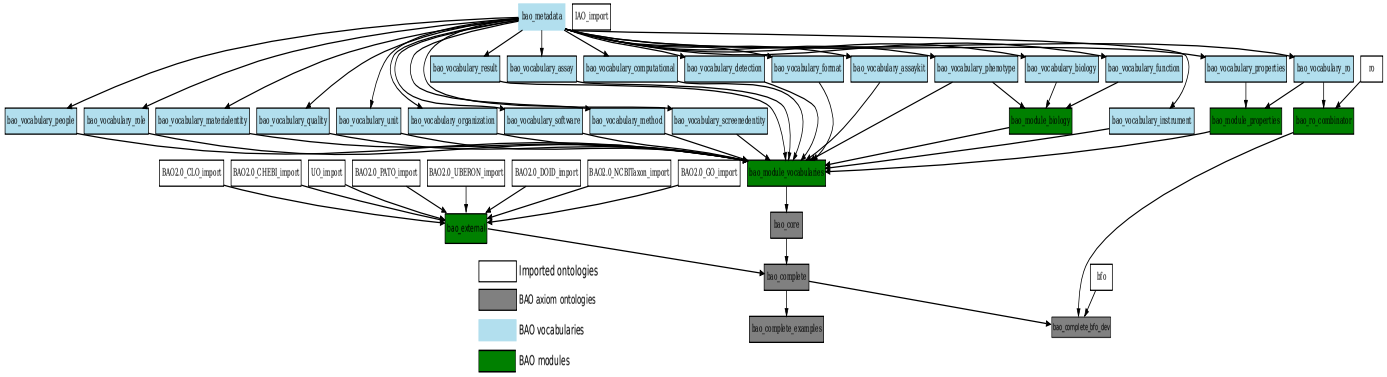


Figure 6.3: BAO 2.0 ontology modularization framework implementation. BAO 2.0 modularization framework provides effective software engineering methods to build complex ontologies. Shown are the current vocabularies, modules, and axiom files also indicating internal vs. external sources.

Module layers enable combining vocabularies in flexible ways to create desired ontology structures or subsets. Axioms are separate files that do not contain any classes or properties. Classes and relationships are imported (directly or indirectly) from module and/or vocabulary files. The above mentioned classes in BAO 2.0 were created as separate vocabulary files. They were then imported into the bao_core file. BAO core only contains axioms incorporating BAO classes and BAO properties.

In our modularization approach we separate external and internal sources. Overlap among external and internal classes and properties (i.e., those required in BAO core) are resolved using combinator modules, that is, external classes and properties are mapped (equivalence or subsumption) to corresponding BAO classes and properties. This approach assures that BAO core remains stable and independent from external sources that may change. The complete BAO includes external axioms and imports BAO core (indirectly importing all vocabularies and properties) and external modules (bao_complete file). Using this approach we also generated the BFO version of BAO. All internal and external vocabulary, module and axiom files are available via the BAO website. Figure 6.3 shows the modularization illustrating vocabularies, intermediate modules, ontology axioms, and module sources with mappings.

6.3.1 Development Approach

BAO 2.0 was developed from BAO 1.6. It was performed in the following steps: First, upper level classes were created to include the various entities that participate in a bioassay, and their roles and qualities. Second, vocabulary files were created by

moving the individual upper classes to the respective files. Third, all the vocabulary files were imported into a single file, called bao_core (see Modularization below). Fourth, the upper level ontology, BFO was imported into bao_complete and the various vocabulary files (either intact or separated as required) were moved to the respective upper level ontology classes. The process of importing external ontology modules, including object properties are described in detail below.

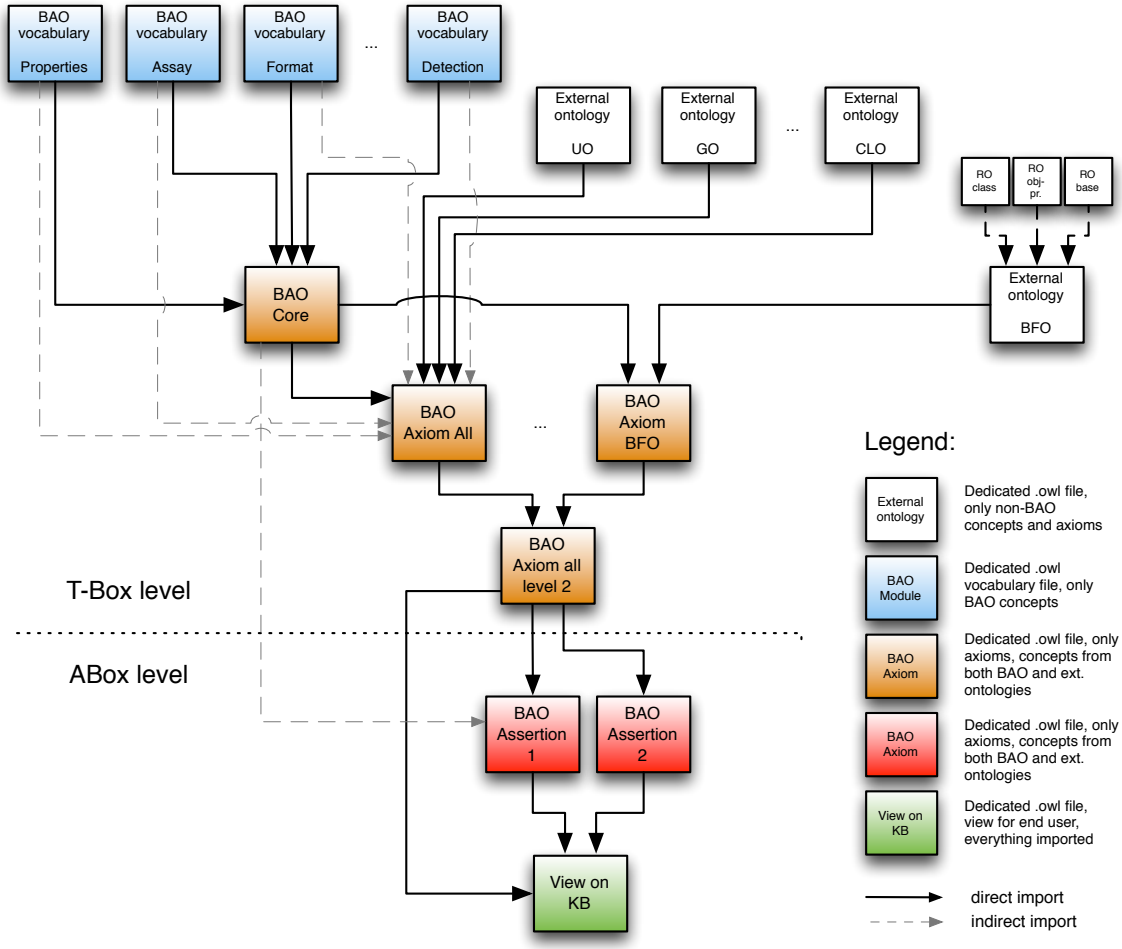


Figure 6.4: BAO 2.0 ontology modularization framework. The framework uses a layered architecture to abstract complexities from different sources. It provides modeling primitives of vocabularies, modules, axioms, and perspectives to develop heterogeneous ontologies.

6.3.2 Generating and Processing External Ontology Modules

BAO is currently using excerpts from eleven external ontologies (including BFO): (1) Gene Ontology (GO); (2) Cell Line Ontology (CLO); (3) Unit Ontology (UO); (4) NCBI Taxonomy (NCBITaxon); (5) Human Disease Ontology (DOID); (6) Chemical Entities of Biological Interest (ChEBI); (7) UBERON (a comparative anatomy ontology); (8) Phenotypic Quality Ontology (PATO); (9) Information Artifact Ontology (IAO); (10) Relationship Ontology (RO); and (11) Basic Formal Ontology (BFO). The workflow for extracting external ontologies is as follows: Domain experts provide the list of concepts of interest and their ontology IDs. Based on these lists and the expression level of the external ontologies, we either use Java programs with OWL API to extract modules from the external ontologies of interest or we use the online tool OntoFox [91] to extract the concepts of interest. Several of the ontologies listed above are taxonomies, where we use OntoFox to avoid overlapping efforts and/or redundant code. Currently we use OntoFox for the ontologies listed below: (1) GO; (2) CLO; (3) NCBITaxon; (4) DOID; (5) ChEBI; (6) PATO; and (7) UBERON.

6.3.3 BAO Modularization Implementation

The Web Ontology Language (OWL) [11] - Description Logic (DL) - provides a rich set of constructors to model a domain of discourse. The DL expressivity comes with a substantial computational cost, as the state-of-the-art DL reasoners costs 2NExp-complete (e.g., [1]). When the size of the ontology increases (number of axioms), the computational cost increases exponentially. In order to manage the size complexity,

we provide a modularization methodology that preserves the required expressivity, yet being able to scale with the size of the ontology. Figure 6.4 shows the basic structure of the methodology.

The proposed modularization framework uses concepts from Directed Acyclic Graphs (DAG)s [92]. First, we determine the abstract horizon between TBox and ABox. TBox contains modules, which define the conceptualization without dependencies. These modules are self contained and well defined with respect to the domain of discourse. In these modules we provide concepts, relations, and individuals. The individuals are restricted to nominals, therefore they only act to close the class expressions. Figure 6.4 shows the main components of the framework: the top left boxes are physical files, i.e., each of them is an .owl file. They contain parts of our ontology, e.g., the top left file may contain everything of the domain of discourse that we think is necessary and important. We can have n of these modules.

Second, once the n modules are defined and if those modules have interdependent axioms, they are provided with another ontology (or module), which imports the necessary modules. At this level one could create any number of gluing modules, which import other modules without dependencies or with dependencies. At this level, the modules depends only on the modules of discourse. So, they all are combined in another physical .owl file, which we may call bao_core (c.f., 6.4). The purpose of this core file is that it not only combines all of the submodules together (by referring to concepts from other physical files), it also is self-contained. This means that there is no outside term or relationship in this file.

Third, at this level we can design modules that import modules from our domain of discourse, and also from third party ontologies. Third party ontologies could be large, therefore a suitable module extraction method (e.g., OWL API) can be used to extract only part of those ontologies (*vide supra*). An example would be using a BFO term or a RO relationships. We would model this in the bao_axiom level. We can have one bao_axiom file or multiple files, each may be modeled for a different purpose, e.g., tailored for various research groups. Thus, bao_Axiom $1 \ldots n$ as seen in Figure 6.4. Once these ontologies are imported, the alignment takes place. The alignments are defined for concepts and relations using equivalence or subsumption DL constructs. The alignment depends on the domain experts' best guesses.

Forth, release the TBox based on the modules created from the third phase. Depending on the end-users, the modules are combined without loss of generality. With this methodology we make sure that we only send out physical files that contain our (and the absolute necessary) knowledge.

Fifth, at this level, the necessary modules ABoxes (again $1 \ldots n$ ABoxes) are created. ABoxes can be loaded to a triple store or to a distributed file system (Hadoop DFS [93]) in a way that one could achieve pseudo-parallel reasoning. Finally, using modules, we define *views on the knowledge base*. These are files that contain imports (both direct and indirect) from various TBoxes and ABoxes modules for the end-user. It can be seen as a *view*, using database terminology. In essence, we will be able to tailor these views based on the modules that we need. We expect that this methodology will speed-up the loading process, since only the necessary modules are loaded rather

than every file that imports thousands of unnecessary and possibly redundant terms (e.g., due to potential loops in the imports). Therefore, BAO modules: (1) modify, expand, and maintain BAO independently; (2) use BAO in related efforts, such as knowledge reporting, more efficiently; (3) expand and synchronize BAO concepts in related efforts (e.g., BARD, LIFE, RegenBase, etc.); (4) reuse parts of BAO for different projects; (5) use other ontologies easily and without effecting BAO in order to support community efforts; and (6) provide transparent mapping of BAO to upper ontologies. In summary, our methodology is as follows: (1) different files for different modules should be created and each module should contain all the concepts as a taxonomy file; (2) after the n modules are created as taxonomies, the core owl file should combine these modules; (3) the axioms related to the core ontology terms should be added to the core owl file; (4) once the core owl file is created that has nothing but the ontology's native concepts and axioms created by the native concepts, the third level file that has external ontologies should be created. The external ontologies can be added by using different combinations and related axioms can be added to the ontology at this level; (5) after creation of the one or more owl files that link different external ontologies and contain related axioms, individuals related with the ontology are added/loaded in the next level; (6) the view file that contains imports (both direct and indirect) from various TBoxes and ABoxes is created from user specifications. Based on this framework and methodology we have modeled the BAO 2.0. Figure 6.3 provides a complete description of the current vocabularies, modules, and axioms files and their connections developed for the ontology.

Our modularization framework differs significantly from existing methodologies: In decision making, a state represents a situation in which decisions should be made. An ontology provides a basic framework to represent situations in which decisions are made. Depending on the layer in which the decisions needs to be made, a state can represent from low-level signals to high-level mental abstractions. Therefore, state abstraction provides the basis in which layer-wise decisions are made. OWL ontologies provides mechanisms such as "owl:import" to represent state abstractions. But this has not been explicitly studied in large scale ontologies. Our modularization framework assesses the capabilities of OWL ontologies to represent state abstractions in different complexities.

Ontology interpretation provides mechanism to represent vocabularies for classes, roles, and individuals. Without any other assumption, the interpretations of these entities provides the state of the system. These entities are analogues to low-level sensations from perceptions. Our modularization framework captures these representations in the vocabulary layer. One can use these representations for tasks such as to populate drop-down menus in a web-application etc. These representations are at its basic levels and the system does not assume any constraints. Having provided constraints leads to OWL ontologies to represent state abstractions.

In order to provide additional information related to basic entities, the next step is to enrich the state with constraints. In first-order-predicate logic, constraints are provided by axioms. Therefore, in OWL ontologies, we use axioms as the method to provide the constraints, hence, the state abstractions. The "modules" in the modular-

ization framework provides different constraints. The modules are connected though the "owl:imports" mechanism, and the constraints are provided by OWL constructs available in $\mathcal{SROIQV}^D$ description logic. Therefore, at each layer, domain experts provide axioms for the best of their knowledge. The modularization framework provides hard boundaries in which, a domain user can extract constraints. This partially addresses one of the problems in ontologies: axioms extraction, which has NP-hard complexity. The hard boundaries provides decision points where abstract state of another system should have been extracted, for example, one can extract whole BAO abstract state without reference to a upper-level ontology such as BFO.

The proposed modularization framework provides basic steps to implement our knowledge base reporting (KBR) application. It needs to infer knowledge from massive ABoxes with parallel reasoning using frameworks such as Map-Reduce. KBR needs decision layers in which to knowledge to be reported, and our modularization framework provides those decision points. In artificial intelligence, state representations and state abstractions are an open-problems. OWL ontologies, with respect to first-order-predicate logic, provides methods to represent knowledge, but, to our knowledge the state abstraction is not discussed widely. Our modularization framework addresses these problems and possibilities in which state abstraction can be generalized.

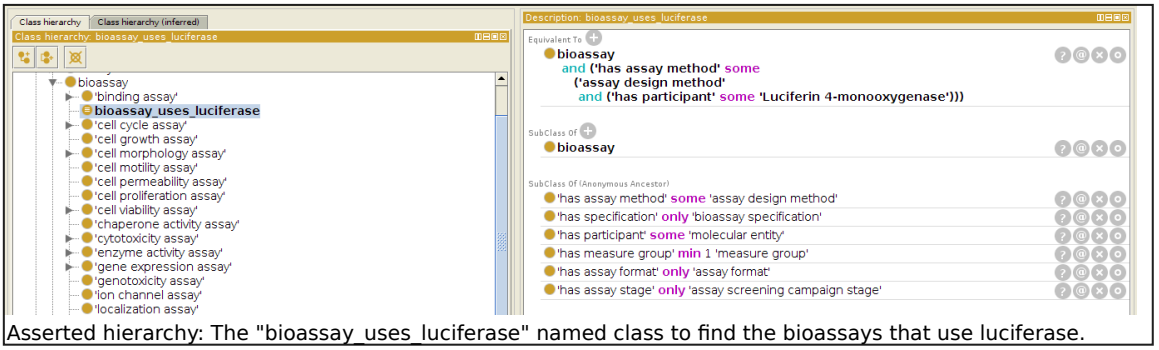
6.3.4 Assay Annotations: Terminology alignment, Reformatting, and Processing

As described in our earlier publication [90], assays from PubChem were annotated using BAO 1.6 terminology. These existing annotations were mapped to corresponding BAO 2.0 classes and annotations were expanded including cell culture conditions, DNA construct, quality, role and function of molecular entities. In addition, the object properties and data properties were refined; many were imported from the RO. Cell line, gene and protein names were standardized by importing the nomenclature from CLO or specific repository, NCBI or HUGO, and UniProt, respectively. In total, 1,000 assays in the PubChem database were annotated using BAO 2.0. These are leveraged in BARD; however, BARD includes all assays and results generated by the MLP screening centers ($> 6,000$) and organizes them by probe projects (> 600). In the process of annotating assays, new terms were collected and subsequently mapped or added to BAO manually (after expert review). In addition, we incorporated terms from some of the Novartis ontology modules and terms requested by other collaborating group (e.g., Astra Zeneca). Assay annotations were captured in a spread sheet with column headers that correspond to BAO classes or relations. For the luciferase assays, we translated the columns headers for the most important annotations and their contents into triples (by mapping column headers to corresponding relations) and loaded them into a RDF triple store as previously described. Figure 6.5 shows an example where we have used BAO 2.0 to infer all bioassays that use a method

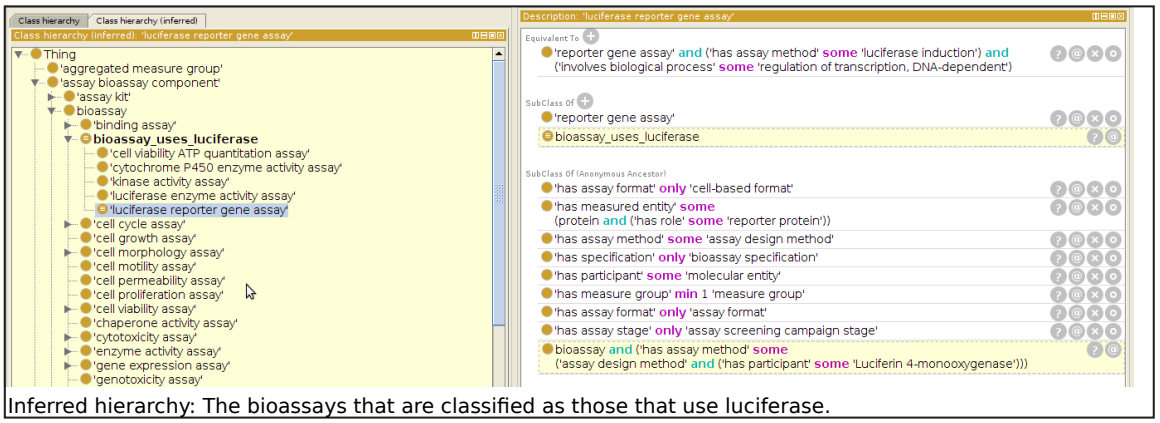
in which Luciferin 4-monooxygenase is a participant. We have defined the equivalent class bioassay_uses_luciferase as bioassay ⊓ ∃'has assay method' ('assay design method' ⊓ ∃'has participant' 'Luciferin 4-monooxygenase'). OWL DL reasoners infer that cell viability ATP quantitation assay, cytochrome P450 enzyme activity assay, kinase activity assay, luciferase enzyme activity assay, and luciferase reporter gene assay are indeed luciferase assays. Figure 6.5 provides justification for luciferase reporter gene assay being a subclass of bioassay_uses_luciferase. This allows us to identify assays that are annotated with any of these assay design methods as assays that use luciferase, which is relevant to identify assay artifacts across various different bioassays.

6.4 Modeling Assays and Results

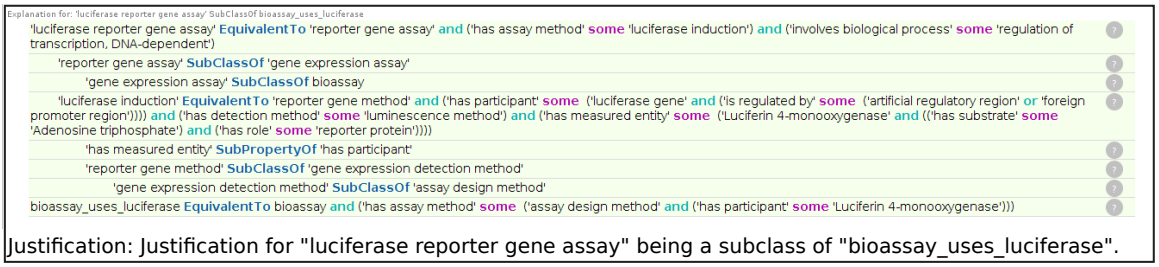
In addition to the BAO modularized design and systematic construction, we also tried to make the definitions of concepts in BAO consistent. We especially defined bioassays with their essential components such as assay design method, endpoints, measure groups, and molecular participants. Figure 6.1 illustrates how assays are modeled by specifying information related to the biology (such as target and/or biological process), assay format, assay method (including assays design method and physical detection method, screened entity and endpoint (result) as described above. The BAO 2.0 architecture allows a more flexible definition of bioassays, for example the same biomolecule can participate in assays in different roles and functions. Important classes include:



Asserted hierarchy: The "bioassay_uses_luciferase" named class to find the bioassays that use luciferase.



Inferred hierarchy: The bioassays that are classified as those that use luciferase.



Justification: Justification for "luciferase reporter gene assay" being a subclass of "bioassay_uses_luciferase".

Figure 6.5: An example that infers all bioassays in the ontology that use luciferase. This example provides asserted and inferred hierarchies for bioassays that use luciferase as a participant. It also provides justification for luciferase reporter gene assay being a subclass of bioassay_uses_luciferase.

- target: The target concept is defined by using the relationships has participant and has role. That is because targets are biological entities (i.e., participants) of assays that are playing the role target. Assays may have single or multiple targets depending on the assay type.

- biological process: A large number of assays are designed to measure outcomes of biological processes. Thus, based on the assay in study, we have written axioms for these information in the assay definitions.

- screened entity: This concept refers to a molecular entity with the role screened entity role.

- participants: Every assay has at least one participant, usually more. While axiomizing the assays, we try to define the particular roles that these participants play in the different assays. However, when we are not certain about the roles, we choose not to put axioms in order to avoid false reasoning cases.

- assay design method: Every assay has an assay design as the underlying method to generate a detectable signal and could correlate with the strength of the perturbation of the biological model system by the screened entity.

- physical detection method: An assay design method, generating a type of signal is linked to a corresponding detection method (the physical principle of detecting the signal), which is typically performed by a detection /instrument.

The concepts listed above along with various other classes are used while modeling the concepts bioassay, measure group, and endpoint. We had previously introduced the concept measure group to link multiple endpoints to the same bioassay [94]. We have now generalized this model so that measure group can be derived from one or more measure groups. This allows the formal and iterative construction of more

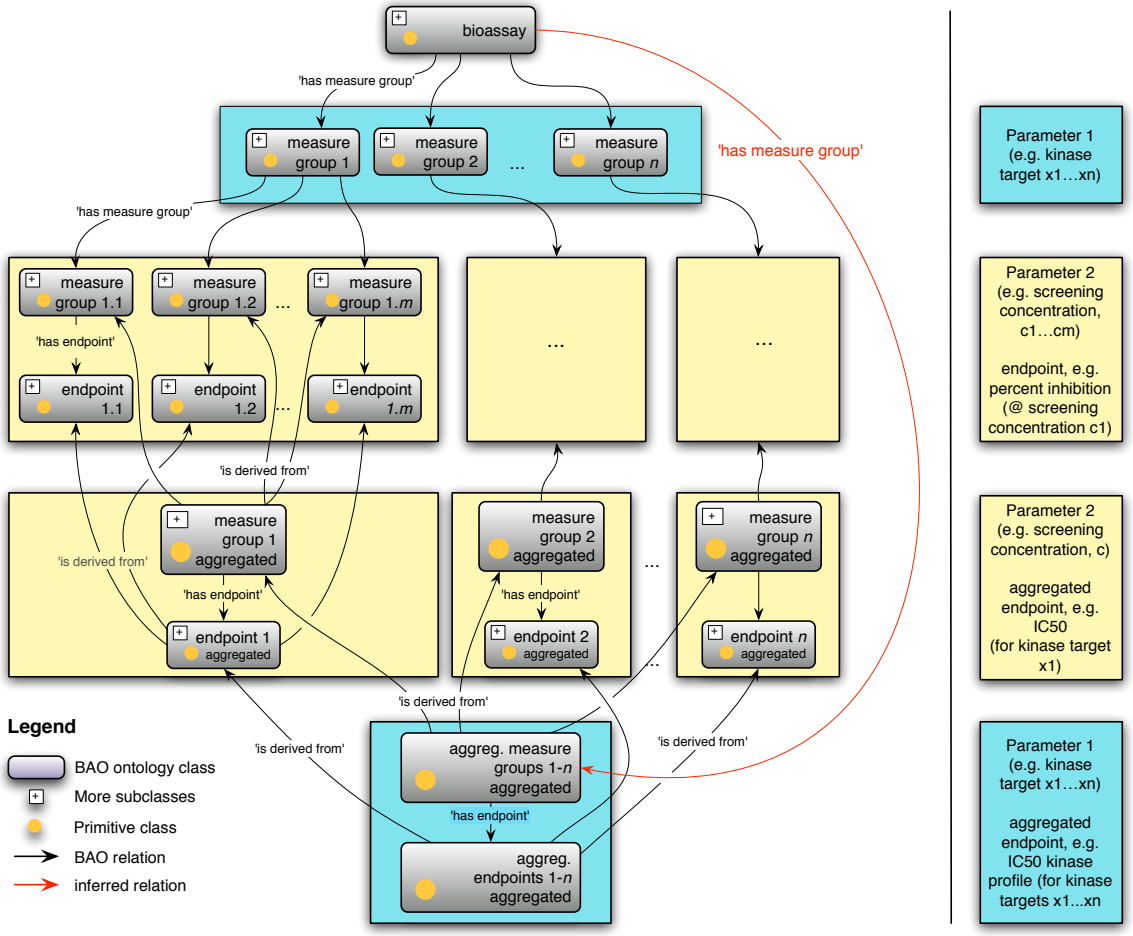


Figure 6.6: Graphical illustration of BAO 2.0 measure group class definition. The class measure group is used to group and link one or more sets of experimental results to one bioassay. By definition one assay can have multiple measure groups. The measure group contains overlapping axioms with the bioassay, which allows the reasoner to infer that the measure group is acting like an equivalent class of the bioassay; i.e., measure group is inferred as subclass of bioassay. Shown is an example of kinase concentration-response profiling panel assay, in which compounds are tested at m concentrations against n kinase targets.

complex assays and endpoints that are derived from multiple measurements (Figure 6.6). The axiomatization was done in a way that infers measure group as a subclass of bioassay (compare Figure 6.1). The axiomatization was motivated by pragmatic considerations for the workflows and perspectives for organizing and analyzing the

assay results, which is the core focus of BAO. It may be argued, that operationally it is not formally an assay; however that is not in conflict with the BAO perspective. It should be noted that BAO measure groups and results remain associated with their corresponding subclasses of bioassay, whose instances are procedurally, methodologically, and materially real. To understand better the relations between the concepts measure group and endpoint we explore them in more depth:

- The class measure group is a concept to group and link one or more (different) sets of experimental results to one bioassay. By definition one assay can have multiple measure groups. However, if the assay has a single measure group, the measure group contains overlapping axioms with the bioassay, which then allows the reasoner to infer that the measure group is acting like an equivalent class of bioassay. However, the measure group and bioassay equivalence cannot be asserted. Because the measure group, in addition to holding the assay component metadata for each reported endpoint, also provides flexibility to generate different derived endpoints, e.g., IC50 (generated from several response values at different concentrations, i.e., concentration-response), or profile endpoints (e.g., a kinase panel assay). In cases where we have multiple measure groups, we then can have derived measure groups, which may contain subclasses of the multiple measure groups.

- The class endpoint, alternatively called result, is a quantitive or qualitative representation of a perturbation (change from a defined reference state of the model

system) that is measured by the bioassay. An endpoint consists of a series of data points, one for each perturbing agent employed by the assay. Every endpoint is obtained by using at least one measure group. For each endpoint, there exists a unit and a value, which is a number (e.g., float, which makes this concept a data property, and the concept is axiomized using a data property as opposed to an object property). For example, for a concentration endpoint, there exists a concentration unit and a concentration value, which is a float number (data property, not functional). However, the endpoint itself is a concentration, but it is derived (e.g., IC50). Assays could have single or multiple endpoints depending on the assay type.

Endpoints are not used to handle the different measurements in the same assay. That is axiomized through the measure group concept. They may vary due to parameters such as time, concentration, target, and so on, or combinations. The formal definitions allow us to create individuals for different endpoints that might be using the same measure groups, i.e., results are measured once and different methods are applied on these measurements to find different derived endpoints. We can group different measure groups to define "intermediate" results. We can create profile endpoints and we can define profiles of intermediate aggregated measure groups (Figure 6.6). An endpoint individual is associated with a specific measure group and a specific compound combination and has a specific value and unit.

In BAO 2.0, endpoints are classified into several categories; the most important ones are concentration endpoint (which includes concentration response endpoint), re-

sponse endpoint, protein substrate and ligand constant, and physical property endpoint. The class mode of action defines the functional effect and physical binding characteristics of the screened entity on the target using the subclasses ligand function mode of action (inhibition, activation, etc.) and ligand binding mode of action (reversible, irreversible, competitive, etc). Each endpoint is associated with a mode of action, e.g., IC50 and percent activation have inhibition and activation as the functional mode of action, respectively. The class signal direction defines how the functional effect of the perturbation corresponds to the intensity of the detected signal, i.e., increase or decrease with activation or inhibition. This is important to identify suitable counter screens; for example if the detected perturbation results in signal decrease in a cell-based assay, cytotoxic compounds may be detected as actives. The class endpoint action correlation defines if the endpoint value corresponds to increased or decreased functional effect (inhibition, activation). Both signal direction and endpoint action correlation are required to formally interpret the results, because the same perturbation (e.g., inhibition of substrate-protein binding by a competing ligand) may be measured via a different molecular entity with the role measured entity (e.g., substrate-bound protein or ligand-bound protein) and the effect can be expressed in different ways (e.g., normalized as remaining percent activity or percent inhibition). Further, depending on the assay design method the same perturbation in the same model system may result in increased or decreased signal.

6.5 Application to Model LINCS Profiling and Panel Assays and Results

The concepts bioassay, measure group, and endpoint as described above enable the formal definition of panel and profiling assays such as those routinely run in the LINCS program. An effective modeling solution is relevant, because of the emphasis of LINCS to operate on result profiles and signatures, in contrast to individual endpoints. We define a panel assay as the parallel, spatially separate implementation of several identical assays, but that vary in one parameter (other than the screened entity), typically the target. A popular example is a kinase panel, for example the DisoveRx KINOMEscan assay that is also run at LINCS and in which compounds are screened against over 450 kinases in parallel. Similar to a panel assay, a profiling assay can generate a large number of readouts for any given tested compound, but all results are obtained from the same physical experiment, i.e., the same well. Such assays are also called multiplexed assays and rely on sophisticated assay methods and/or detection technologies that enable the detection of many signals in parallel, such as flow cytometry, mass spectrometry or imaging. One example also run at LINCS is the L1000 transcriptional profiling assay (*vide supra*). As illustrated in Figure 6.6, our approach would also allow to define concentration response (e.g., IC50) kinase profiling assays via iterative aggregation of sets of measure groups corresponding to two parameters, namely m screening concentration (values) and n kinase targets. The first aggregation by screening concentration (e.g., via curve fitting) defines the IC50

endpoint for each kinase and the second aggregation defines an IC50 kinase profile endpoint. An actual example of such a assay is the ActivX Biosciences KiNative assay, which is also run in the LINCS program. We have modeled several LINCS assays including KINOMEScan assay, transcriptional response profiling assay, cell cycle state assay. The specific instances of these assays including hundreds of kinase targets, transcribed genes, cell lines, etc was implemented in an application ontology and these assays are screening results are available in our LIFE software system [95].

An example of a phenotypic cell-based LINCS assay is the cell cycle state assay. It is also described in BAO 2.0. In the LINCS project, several small molecules that are known to function as kinase inhibitors were tested on cancer cell lines for their ability to arrest the mitotic cell cycle. This assay was modeled in BAO as follows: the assay design method is S phase assessment or M phase assessment method. The presence of the markers, namely, EdU and anti-MPM-2 antibody, indicates that cells have entered/completed S phase and M phase, respectively. Hoechst 33342 was used to stain nuclei from all cells to obtain the total cell count in the assay. The detection method is fluorescence microscopy and the measured entity is DNA. The assay readout parameters are intensity parameter and counting parameter. The intensity of EdU and MPM2 were measured in the nucleus and cytoplasm, respectively. The counts of Hoechst 33342, EdU and MPM2 positive cells were reported after the threshold to signal intensity of each marker was applied. The endpoint was derived from the assay readout parameters after normalizing with the assay controls. The endpoint for this assay is percent apoptotic cells, percent mitotic cells, percent interphase cells, percent

DNA replicated cells, percent G2 arrested cells, and/or percent mitotic arrested cells. The cellular phenotype or its disposition is obtained by quantifying cells which are positive for each of these markers.

6.6 Categorizing Mechanistically Related Assays by Inference

BAO 2.0 contains detailed description of a range of common HTS assay, including the categories: binding assay, cell cycle assay, cell viability assay, cytotoxicity assay, enzyme activity assay, gene expression assay, redistribution assay, and signal transduction assay. The essential information that was described for each assay type includes format, method (including assay design), detection, endpoint, and molecular and cellular entities and their roles, qualities and functions describing the biology of the system or which are key components involved in the assay design or detection methods. We have previously shown how promiscuous frequent hitter compounds (undesired assay artifacts) can be deconvoluted and categorized mechanistically based on detailed knowledge about the assays and their related design and detection methods [96]. However, using the previous version of BAO (1.6) these assays were not yet defined in a way that formalizes all necessary knowledge about their commonalities. This means that previously, in order to perform mechanism-based cross assay analysis, some human expert knowledge was required to identify and categorize related assays beyond their asserted annotations.

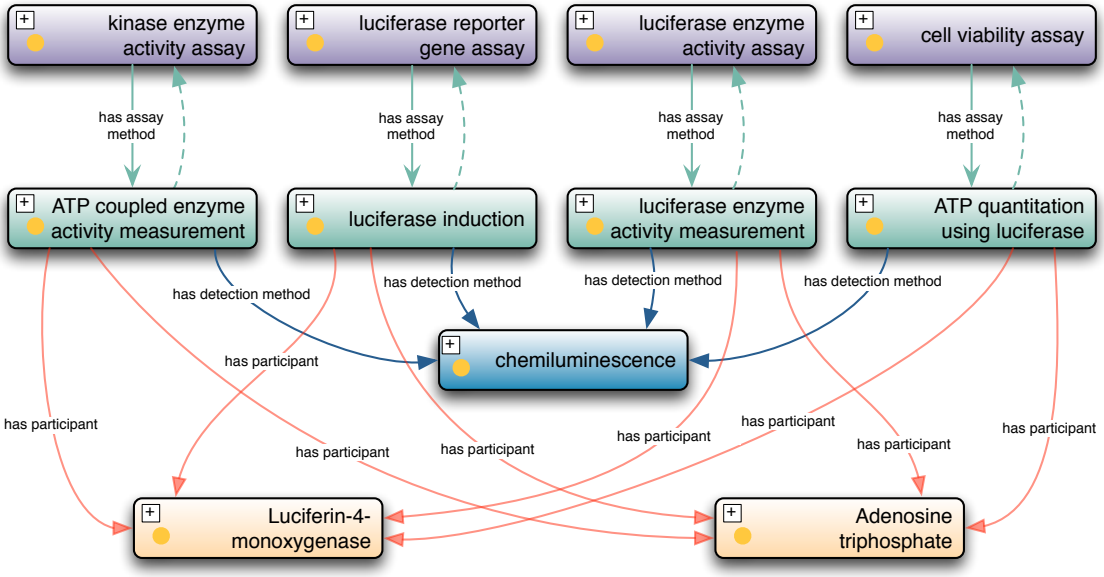


Figure 6.7: Conceptual modeling of different luciferase assays. Shown are bioassay, assay design method, physical detection method and participants (molecular entities with a specified role in the assay).

BAO 2.0 provides a framework that enables automated classification of assays into meaningful categories of interest, for example to aid in identifying common assay artifacts and their likely mechanism of action. We illustrate this using several related assays: luciferase reporter gene assay, cell viability ATP quantitation assay, cytochrome P450 enzyme activity assay, kinase activity assay, and luciferase enzyme activity assay. Of these, the reporter gene and cell viability assays are cell-based, while the others are biochemical assays. The modeling of these assays is illustrated in Figure 6.7. All assays use a different assay design method. Therefore they cannot be identified as mechanistically related based on that annotation alone. The physical detection method chemiluminescence is the same for all assays, but it is too generic to classify the assays by mechanisms that underlie artifacts, because luminescence can be generated by many methods. However, among these examples, all assays perform (in different ways)

the luciferase-catalyzed chemical reaction of luciferin and ATP forming oxyluciferin and light (luminescence) and thus luciferase and ATP participate in all these assays, although in different roles. For example in the reporter gene assay the amount of expressed luciferase is quantified by the intensity of light (luminescence) produced in the presence of substrates, ATP, and luciferin. In the viability assays the proportion of living cells is quantified by measuring ATP content, again by the same reaction (with ATP as the limiting reagent in the role measured entity). Similarity ATP-coupled assays measure the residual amount of ATP (e.g., after a kinase reaction) by a coupled luciferase reaction. The P450 luciferin-coupled assay mentioned above measures the amount of luciferin generated after detoxification by cytochrome P450 enzyme activity. Luciferase enzyme activity assays quantify the biochemical luciferase enzyme activity by the intensity of light, again using the same chemical reaction. In BAO2.0 we modeled these assays with the necessary formalism to enable the reasoning engine to categorize the assays as mechanistically related. As an example, Figure 6.8 shows the asserted TBox of the assay design method ATP quantization using luciferase and ATP coupled enzyme activity measurement method and the inferred TBox in which the latter is classified as a subclass of the former. For illustrative purposes we defined a class of all assays with an assay design method in which luciferase participates (in any role). The axioms and the asserted and inferred hierarchies are shown in Figure 6.5. All assays mentioned above are inferred as assays that use luciferase, thus illustrating how BAO formal assay definitions enable a classification based on the mechanistic principle of the assay (assay design method). This in turn classifies the assay based

on likely common artifacts (e.g. compounds that stabilize or inhibit luciferase) [96]. Figure 6.5 also shows the justification for classifying the assays mentioned above under this category.

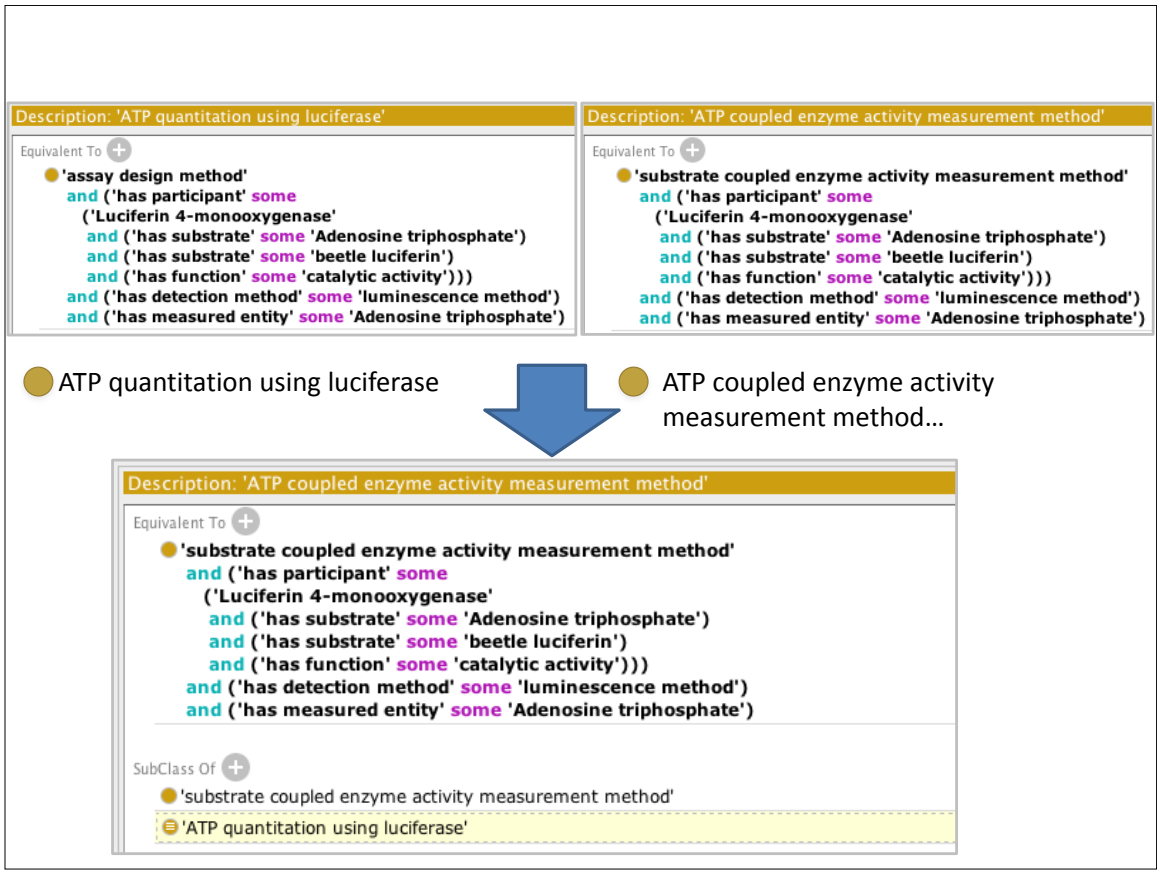


Figure 6.8: Examples of luciferase assay design methods. Shown are the asserted TBox of ATP quantitation using luciferase and ATP coupled enzyme activity measurement method and the inferred TBox in which the latter is classified as a subclass of the former.

6.7 Conclusions

We have developed BAO 2.0 as a reference for standard metadata terms and definitions required to describe relevant information of low and high-throughput drug and probe screening assays and results to enable effective data integration, aggregation, retrieval, and analyses. BAO 2.0 has been developed collaboratively to provide wider scope in describing and modeling diverse and complex assays. BAO is extended significantly with regard to the previous version using domain knowledge and data annotated in BARD and by other collaborators. We have described a flexible layered architecture to develop and integrate plethora of modules from established biomedical ontologies and upper level ontologies. Our modularization approach defines several integral distinct BAO 2.0 modules and separates internal from external modules and domain-level from structural components. This approach facilitates the generation/extraction of derived ontologies (or perspectives) that suit a particular use case or software application. We have generalized BAO to enable modeling of result profiles (signatures) generated in panel and profiling assays, for example those in the LINCS project. BAO leverages OWL DL ($\mathcal{SROIQV}^D$) to capture and formalize knowledge about assays and screening results and to enable computational systems to utilize knowledge. This enables the classification of assays and screening results into categories that relate to the assay model system, the biology (e.g., protein target or process), how a signal is generated and how it is detected, and screening results. We demonstrated inference reasoning capabilities of BAO to classify assays

into categories that relate to how the assay works. This offers the potential to identify common promiscuous frequent hitters and their possible mechanism of action. We continue to develop and expand BAO further with the goal to establish a standard to report chemical biology assays and their results. For example, to better describe the pharmacology of GPCRs, we recently developed a GPCR ontology framework [97]. We are also expanding BAO further to describe high-content phenotypic assays. BAO is currently used in several public and private screening projects and evaluated by a number of organizations and projects. The participation of groups from industry and academia to develop and use BAO illustrates the utility of the product as well as increasing public-private collaboration in pre-competitive areas, such as the development of standards and ontologies.

Chapter 7

Conclusion and Future Work

In this dissertation we have presented solutions to the problem of real-time knowledge representation, learning, and reasoning for three modalities: 1) RoboCup soccer; 2) High-throughput Screening; and 3) Axon regeneration. The notion of "real-time" depends on the constraints of the domain and the frequency with which the contents need to be queried. We have shown that within the given computational complexity of DL, real-time knowledge representation and learning can be conducted for dynamically varying assertions. We have shown an extension to the assertional formalism to represent *time* naturally within DL semantics. The proposed method have been successfully used in the RoboCup 3D soccer simulation environment and we have shown that the reasoning and querying is achievable within 5 ms, 10 ms, and 15 ms.

We have investigated potential alternatives to knowledge representation and learning to a sub-task of RoboCup 3D soccer simulation league. This involves the identification of role assignments to the soccer playing agents. We have used the General Value Functions in Reinforcement Learning as an alternative to Description Logic to represent and learn the underlying dynamics and semantics of the problem. The method has learned policies that outperform hand coded knowledge on smaller tasks,

while, for larger tasks, the method is par with the hand coded policies. We have used DL as the basis to represent knowledge in large corpora such as high-throughput screening and axon regeneration. First, we have investigated whether it is possible to encode the use cases presented in these domain using the constructs of DL. Second, once the knowledge base is formalized, we have investigated, whether it is possible to reason about the knowledge within a reasonable amount of time. Third, a modularization methodology to integrate a myriad of existing ontologies to a given knowledge base without compromising the interpretation. Our methods have introduced solutions to segregate ontologies for a given user perspective.

One of the ways in which we envision that the work presented in this dissertation can be further extended with merging the DL conceptualization with GVF semantics. This can be achieved by treating GVFs as part of an ABox. We learn domain specific GVFs and treat them as symbols that needs to be interpreted by the ontology. Therefore, one can provide complex concept descriptions and binary relationships to GFVs. Therefore, an existing ontology reasoner can infer GVFs as individuals of the knowledge base. The methods presented in Chapter 3 can be used to represent dynamically changing GVFs. Hence, one can learn and infer about entities in the domain within the given real-time bounds.

Appendix A

RLLib

RLLib (`http://rllib.samminda.org`) is a lightweight C++ template library that implements incremental, standard, and gradient temporal-difference learning algorithms in reinforcement learning. It is an optimized library for robotic applications and embedded devices that operates under fast duty cycles (e.g., $\leq$ 30 ms). RLLib has been tested and evaluated on RoboCup 3D soccer simulation agents, NAO V4 humanoid robots, and Tiva C series launchpad microcontrollers to predict, control, learn behavior, and represent learnable knowledge. We have used RLLib to empirically evaluate the ideas presented in Chapter 4.

A.1 Features

RLLib implements and features: 1) off-policy prediction algorithms: (GTD(λ) and GQ(λ)) [45]; 2) off-policy control algorithms: Greedy-GQ(λ) [45] and (Softmax GQ(λ) and Off-PAC) [44]; 3) on-policy algorithms: (TD(λ), SARSA(λ), Expected SARSA(λ), and Actor-Critic (continuous and discrete actions, discounted, averaged reward settings, so forth)) [55], (Alpha Bound TD(λ) and SARSA(λ)) [98], and (True

TD(λ) and SARSA(λ)) [99]; 4) incremental supervised learning algorithms: Adaline [100], (IDBD and Semi-Linear IDBD) [101], and Auto-Step [102]; 5) discrete and continuous policies: (Random, Random X percent bias, Greedy, ϵ-greedy, Boltzmann, Normal, and Softmax); 6) sparse feature extractors (e.g., Tile Coding) with pluggable hash functions [55]; 7) an efficient implementation of the dot product for sparse coder based feature representations; 8) benchmark environments: (Mountain Car, Mountain Car 3D, Swinging Pendulum, Helicopter, and Continuous Grid World) [55]; 9) optimization for very fast duty cycles (e.g., using culling traces, RLLib is tested on the RoboCup 3D simulator agents, physical NAO V4 humanoid robots, and Tiva C series launchpad microcontrollers); 10) a framework to design complex behaviors, predictors, controllers, and represent highly expressive learnable knowledge representations in RL using GVFs; 11) a framework to visualize benchmark problems; and 12) a plethora of examples demonstrating on-policy and off-policy control experiments.

A.2 Platform

RLLib closely follows the design principles and recommendations presented in [103] and [104]. The development of the library has taken significant efforts to minimize memory footprint as well as computational requirements that are requested by RL problems. Since RLLib specifically emphasizes on learnable knowledge representation and reasoning, it has been modularized based on on-policy and off-policy RL methods.

Listings 1 and 2 provide the minimal pseudo-code to setup on/off-policy RL agents. The controlling, behavior learning, and learnable knowledge representation problems should use "ControlAlgorithm.h" header file. The algorithms related to predictions and supervised learning problems are implemented in "PredictorAlgorithm.h" and "SupervisedAlgorithm.h" header files. All C++ templates implemented in the library are under the namespace RLLib, and use a single parameter T. This parameter could be a C++ primitive type as shown in Listings 1 and 2 or a complex object defined by the user. It is our experience that majority of RL problems can be defined using a primitive type. Devices such as Tiva C launchpad microcontrollers supports single precision floating point representations. Therefore, the templates should be initialized with float primitive type.

```
// -------------------------------------------------
1  include "ControlAlgorithm.h"
2  include "RL.h"
3  using namespace RLLib;
// RL Problem -------------------------------------
4  RLProblem<double>*  problem = ...;
// Projector --------------------------------------
5  Hashing<double>* hashing = ...;
6  Projector<double>*  projector = ...;
7  StateToStateAction<double>*  toStateAction = ...;
// Predictor --------------------------------------
8  Trace<double>* e = ...;
9  GQ<double>* gq = ...;
// Policies π and π_b -----------------------------
10 Policy<double>*  target = ...;
11 Policy<double>*  behavior =...;
// -------------------------------------------------
12
13
// Controller -------------------------------------
14 OffPolicyControlLearner<double>*  control = ...;
// Runner -----------------------------------------
15 RLAgent<double>*  agent = ...;
16 Simulator<double>*  sim = ...;
17 sim->run(); // OR sim->step();
// -------------------------------------------------
```

Listing 1: Pseudo-code for action-value methods.

```
// -------------------------------------------------
1  include "ControlAlgorithm.h"
2  include "RL.h"
3  using namespace RLLib;
// RL Problem -------------------------------------
4  RLProblem<double>*  problem = ...;
// Projectors -------------------------------------
5  Hashing<double>* hashing = ...;
6  Projector<double>*  projector = ...;
7  StateToStateAction<double>*  toStateAction = ...;
// Critic -----------------------------------------
8  Trace<double>* critice = ...;
9  GTDLambda<double>* critic = ...;
// Policies π and π_b -----------------------------
10 PolicyDistribution<double>*  target = ...;
11 Policy<double>* behavior =...;
// Actor ------------------------------------------
12 Traces<double>*  actoreTraces = ...;
13 ActorOffPolicy<double>*  actor = ...;
// Controller -------------------------------------
14 OffPolicyControlLearner<double>*  control = ...;
// Runner -----------------------------------------
15 RLAgent<double>*  agent = ...;
16 Simulator<double>*  sim = ...;
17 sim->run(); // OR sim->step();
// -------------------------------------------------
```

Listing 2: Pseudo-code for policy gradient methods .

In line 4, we define the specification of the RL problem using an instance of the template RLProblem<T>. This template as well as RLAgent<T> and Simulator<T>

(lines 15–16) templates are defined in "RL.h" header file. An instance of the template Projector<T> (line 6) extracts features from the state variables. These features are part of a function approximation architecture (linear or non-linear), e.g., tile based sparse features [55, Section 8.3.2] or compact features [105], suitable for the problem. Some feature extractors require a hashing function, that is defined in line 5. For action-value functions, the features could also include actions (or options), that is defined in StateToStateAction<T> (line 7). In Listing 1, lines 8–9 define the predictor, which is used in off-policy controller (line 14), while Listing 2 defines the critic that is used in the actor-critic controller. Lines 12–13 in Listing 2 define the actor for the prior controller. Lines 10–11 define the target policy (the smooth policy distributions in Listing 2) and the behavior policy. Line 15 defines the RL agent that is used in the simulator (line 16) simultaneously with the RL problem.

In simulations (e.g., [106]), specifically when the simulator has the control over the perception-actuation cycles, the runner is executed with run() in line 17. In practical problems (e.g., [51]), where the agents and the environments run on disjoint processes, the runner will wait for the percepts, then updates the agent, which in return transmits the actions to the environment. In such situations, the runner is executed with step() in line 17. It is to be noted that either in simulations or in practical problems the runner will execute the same update steps, such that, the user will experience the same set of execution steps. A practitioner can construct the C++ objects in lines 4–16 in an initialization subroutine, and execute line 17 in a subroutine that calls in every duty cycle.

There are complex combination of RL algorithms used in practice. RLLib allows many combination of these algorithms by changing a few lines of code in Listings 1 and 2. For example, in oder to implement on-policy action-value methods, a practitioner can change the predictor in line 9 to `Sarsa<T>` and the controller in line 14 to `OnPolicyControlLearner<T>` in Listing 1. Similarly, different combination of RL algorithms can be included in Listing 2 for actor-critic, and parameterized policies.

A.3 Experiments

This section provides experiments: 1) to validate the effectiveness of the library across multiple hardware platforms; and 2) the ability to answer a subjectively posed predictive question using the conceptualization of GVFs.

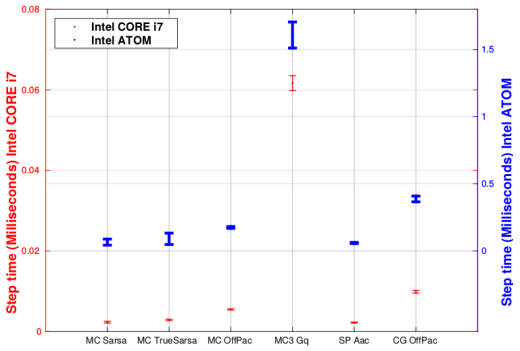


Figure A.1: Step update times in milliseconds. The thick error bars (blue) show the step time for Intel ATOM, while the thin error bars (red) show the step update time for Intel CORE–i7.

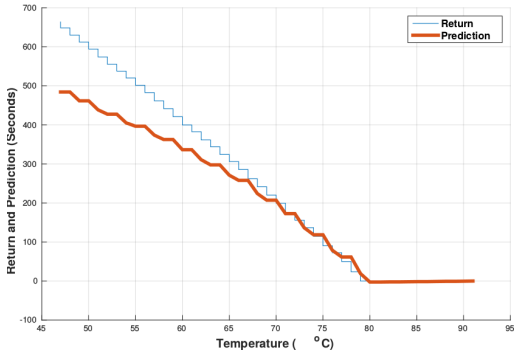


Figure A.2: Predicting the time to shutdown in seconds. The bold line (red) shows the prediction to shutdown from a given temperature of NAO left knee. The thin line (blue) shows the actual return.

Figure A.1 shows the step time in milliseconds, i.e., the time an algorithm requires to update its parameters from the observations, for a set of benchmark problems pop-

ular in RL literature. We have considered two hardware platforms: 1) Intel CORE–i7 2.2GHz laptop; and 2) Intel ATOM 1.6GHz CPU available on NAO humanoid robot. MC Sarsa, MC TrueSarsa, and MC OffPac represent the two dimensional mountain car benchmark problem solved using sarsa, true sarsa, and off-policy actor-critic algorithms. The reader is referred to [106], [99], and [44] that describe the problems, feature extractions, and parameter settings respectively. MC3 Gq describes the three dimensional mountain car [107] solved using greedy-GQ algorithm [45]. SP Aac and CG OffPac represent swinging pendulum and continuous grid-world problems solved using average [108] and off-policy [44] actor-critic algorithms. Even though the step update on NAO platform is on average 30.64 times slower than the laptop, the step update is suitable for real-time operations (worst case time is approximately 1.5ms).

Our second experiment poses a question of the form "How much time remaining on NAO before the left knee temperature reaches above 80^oC?". We have configured a NAO robot to walk in the standard platform league soccer field. We have started the robot in resting temperature, and stopped the experiment when at least one of the joints reached the critical temperature ($> 80^o$C). We have used GQ(λ) [44] with the GVF question encoding: π(s,a) = 1, $r(s) = 0.01, z(s) = 0$ seconds (we have queried the robot sensor values at 100Hz), $\forall s \in \mathcal{S}$, and $\gamma(s) = 0$ if the left knee temperature is greater than 80^oC, otherwise $\gamma(s) = 1$.

We have used the answer encoding: $\lambda(s) = 0.4$, $\forall s \in \mathcal{S}$ and single tiling with 28 regions of the effective temperature range (47^oC, 91^oC) with 512 memory (the size of the feature, eligibility, and primary and secondary parameter vectors). We set the

two step size parameters to $\alpha_v = 0.2$ and $\alpha_w = 0.000001$. As shown in Figure A.2, the agent has learned accurate predictions to shutdown from returns.

Bibliography

[1] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman and Hall (CRC), 2009.

[2] Aldebaran Robotics (NAO Hardware). `http://www.aldebaran-robotics.com/documentation/nao/hardware/`. Last visited on 07/06/2015.

[3] Justin Stoecker and Ubbo Visser. Roboviz: Programmable Visualization for Simulated Soccer. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluc Saranli, editors, *RoboCup*, Lecture Notes in Computer Science, pages 282–293. Springer, 2011.

[4] J Cullen and A Bryman. The Knowledge Acquisition Bottleneck: Time for Reassessment? *Expert Systems*, 5(3):216–225, 1988.

[5] Saminda Abeyruwan and Ubbo Visser. A New Real-Time Algorithm to Extend DL Assertional Formalism to Represent and Deduce Entities in Robotic Soccer. In Reinaldo A. C. Bianchi, H. Levent Akin, Subramanian Ramamoorthy, and Komei Sugiura, editors, *RoboCup 2014: Robot World Cup XVIII*, volume 8992 of *Lecture Notes in Computer Science*, pages 270–282. Springer International Publishing, 2015.

[6] Saminda Abeyruwan, Andreas Seekircher, and Ubbo Visser. Dynamic Role Assignment using General Value Functions. In *Proceedings of Autonomous Agents and Multi-Agent Systems, Workshop on Adaptive Learning Agents*, 2013.

[7] Ubbo Visser, Saminda Abeyruwan, Uma Vempati, Robin P Smith, Vance Lemmon, and Stephan C Schürer. BioAssay Ontology (BAO): A Semantic Description of Bioassays and High-throughput Screening Results. *BMC Bioinformatics*, 12(1):257, 2011.

[8] Saminda Abeyruwan, Caty Chung, Nakul Datar, Felimon Gayanilo, Amar Koleti, Vance Lemmon, Christopher Mader, Mitsunori Ogihara, Deepthi Puram, Kunie Sakurai, Robin Smith, Uma Vempati, Sreeharsha Venkatapuram, Ubbo Visser, and Stephan Schürer. BAOSearch: A Semantic Web Application for Biological Screening and Drug Discovery Research. *Semantic Web Challenge*, 2010.

[9] Vance P Lemmon, Saminda Abeyruwan, Ubbo Visser, and John L Bixby. Facilitating Transparency in Spinal Cord Injury Studies using Data Standards and Ontologies. *Neural Regeneration Research*, 9(1):6, 2014.

[10] Saminda Abeyruwan, Uma D Vempati, Hande Küçük-McGinty, Ubbo Visser, Amar Koleti, Ahsan Mir, Kunie Sakurai, Caty Chung, Joshua A Bittker, Paul A Clemons, et al. Evolving BioAssay Ontology (BAO): modularization, integration and applications. *Journal of Biomedical Semantics*, 5(Suppl 1):S5, 2014.

[11] The OWL 2 Web Ontology Language Document Overview. `http://www.w3.org/TR/owl2-overview/`. Last visited on 07/06/2015.

[12] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, New York, NY, USA, 2nd edition, 2010.

[13] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible SROIQ. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 57–67. AAAI Press, June 2006.

[14] Markus Krötzsch, Frederick Maier, Adila Krisnadhi, and Pascal Hitzler. A better uncle for OWL: nominal schemas for integrating rules and ontologies. In *Proc. 20th International Conference on World Wide Web (WWW'11)*, pages 645–654. ACM, March 2011.

[15] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System Description. *Journal of Web Semantics (JWS)*, 27:78–85, 2014.

[16] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[17] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A Practical OWL-DL Reasoner. *Web Semantics: Science,Services and Agents on the World Wide Web*, 5(2):51–53, June 2007.

[18] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In Alan Ruttenberg, Ulrile Sattler, and Cathy Dolbear, editors, *Proceedings of the 5th International Workshop on OWL: Experiences and Directions*, Karlsruhe, Germany, October 26–27 2008.

[19] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347, New York, NY, USA, 1997. ACM.

[20] Spark Generic Physical Multiagent Simulator (SimSpark). `http://simspark.sourceforge.net/wiki/index.php/Main_Page`. Last visited on 07/06/2015.

[21] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

[22] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The Robot World Cup Initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347, New York, NY, USA, 1997. ACM.

[23] Michelle Birdsall. Google and ITE: The Road Ahead for Self-Driving Cars. *ITE Journal*, 84(5), 2014.

[24] Chieh Chih Wang, Charles Thorpe, and Sebastian Thrun. 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.

[25] Boris Motik, Ian Horrocks, and Su Myeon Kim. Delta-Reasoner: A Semantic Web Reasoner for an Intelligent Mobile Platform. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *WWW (Companion Volume)*, pages 63–72. ACM, 2012.

[26] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[27] Alessandro Artale and Enrico Franconi. A Temporal Description Logic for Reasoning about Actions and Plans. *Journal of Artificial Intelligence Research (JAIR)*, 9:463–506, 1998.

[28] Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporal Description Logics: A Survey. In Stphane Demri and Christian S. Jensen, editors, *TIME*, pages 3–14. IEEE Computer Society, 2008.

[29] Jerry R. Hobbs and Feng Pan. 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.

[30] Viorel Milea, Flavius Frasincar, and Uzay Kaymak. tOWL: A Temporal Web Ontology Language. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(1):268–281, 2012.

[31] James F. Allen and George Ferguson. Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[32] Nikos Papadakis, Kostas Stravoskoufos, Evdoxios Baratis, Euripides G. M. Petrakis, and Dimitris Plexousakis. PROTON: A Prolog Reasoner for Temporal ONtologies in OWL. *Expert Systems with Applications*, 38(12):14660–14667, 2011.

[33] Leonard Petnga and Mark Austin. Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems. In Christiaan J. J. Paredis, Carlee Bishop, and Douglas A. Bodner, editors, *CSER*, volume 16 of *Procedia Computer Science*, pages 403–412. Elsevier, 2013.

[34] Barry Smith, Werner Ceusters, Bert Klagges, Jacob Kohler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan Rector, and Cornelius Rosse. Relations in Biomedical Ontologies. *Genome Biology*, 6(5):R46+, 2005.

[35] Kathrin Dentler, Ronald Cornet, Annette ten Teije, and Nicolette de Keizer. Comparison of Reasoners for Large Ontologies in the OWL 2 EL Profile. *Semantic Web*, 2(2):71–87, April 2011.

[36] Christopher A. Welty and Richard Fikes. A Reusable Ontology for Fluents in OWL. In Brandon Bennett and Christiane Fellbaum, editors, *Formal Ontology in Information Systems*, volume 150 of *Frontiers in Artificial Intel. and Apps.*, pages 226–236. IOS, 2006.

[37] Carsten Rachuy and Ubbo Visser. 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*, pages 979–980, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[38] Tobias Warden, Andreas D. Lattner, and Ubbo Visser. Real-Time Spatio-Temporal Analysis of Dynamic Scenes in 3D Soccer Simulation. In Luca Iocchi, Hitoshi Matsubara, Alfredo Weitzenfeld, and Changjiu Zhou, editors, *RoboCup 2008: Robot Soccer World Cup XII*, pages 366–378. Springer-Verlag, Berlin, Heidelberg, 2009.

[39] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '11, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[40] Natasha Noy and Alan Rector. Defining N-ary Relations on the Semantic Web. Technical report, W3C Working Group, 2006.

[41] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '11, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[42] Hamid Reza Maei and Richard S Sutton. GQ(λ): A General Gradient Algorithm for Temporal-Difference Prediction Learning with Eligibility Traces. *Proceedings of the 3rd Conference on Artificial General Intelligence (AGI-10)*, pages 1–6, 2010.

[43] Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S. Sutton. Toward Off-Policy Learning Control with Function Approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pages 719–726, 2010.

[44] Richard S. Sutton Thomas Degris, Martha White. Off-Policy Actor-Critic. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML)*, 2012.

[45] Hamid Reza Maei. *Gradient Temporal-Difference Learning Algorithms*. PhD Thesis, University of Alberta., 2011. PhD Thesis.

[46] Richard S. Sutton, Csaba Szepesvári, and Hamid Reza Maei. A Convergent O(N) Algorithm for Off-Policy Temporal-Difference Learning with Linear Function Approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1609–1616. MIT Press, 2008.

[47] Peter Stone and Manuela Veloso. Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[48] Peter Stone and Manuela Veloso. Layered Learning. In *Proceedings of the Eleventh European Conference on Machine Learning*, pages 369–381. Springer Verlag, 1999.

[49] J. Kober, J. Andrew (Drew) Bagnell, and J. Peters. Reinforcement Learning in Robotics: A Survey. *International Journal of Robotics Research*, July 2013.

[50] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement Learning for RoboCup-Soccer Keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[51] Andreas Seekircher, Saminda Abeyruwan, and Ubbo Visser. Accurate Ball Tracking with Extended Kalman Filters as a Prerequisite for a High-Level Behavior with Reinforcement Learning. In *The 6th Workshop on Humanoid Soccer Robots at Humanoid Conference, Bled (Slovenia)*, 2011.

[52] Thomas Gabel, Sascha Lange, Martin Lauer, and Martin Riedmiller. Bridging the Gap: Learning in the Robocup Simulation and Midsize League. In *Proceedings of the 7th Portuguese Conference on Automatic Control (Controlo)*, 2006.

[53] Martin Riedmiller and Thomas Gabel. On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup. In *Third IEEE Symposium on Computational Intelligence and Games*, pages 17–23. IEEE, 2007.

[54] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement Learning for Robot Soccer. *Autonomous Robots*, 27:55–73, July 2009.

[55] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[56] Hatice Köse, Utku Tatladede, Cetin Mericli, Kemal Kaplan, and H. Levent Akan. Q-Learning Based Market-Driven Multi-Agent Collaboration in Robot Soccer. In *Proceedings of the Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN)*, pages 219–228, 2004.

[57] José Angelo Gurzoni, Jr., Flavio Tonidandel, and Reinaldo A. C. Bianchi. Market-Based Dynamic Task Allocation using Heuristically Accelerated Reinforcement Learning. In *Proceedings of the 15th Portugese Conference on Progress in Artificial Intelligence*, EPIA'11, pages 365–376, Berlin, Heidelberg, 2011. Springer-Verlag.

[58] Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan, Francisco Barrera, Adrian Lopez-Mobilia, Nicolae Ştiurcă, Victor Vu, and Peter Stone. UT Austin Villa 2011: A Champion Agent in the RoboCup 3D Soccer Simulation Competition. In *Proceedings of 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, June 2012.

[59] Thomas Degris and Joseph Modayily. Scaling-up Knowledge for a Cognizant Robot. *In Notes of the AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*, 2012.

[60] Joseph Modayil, Adam White, and Richard S. Sutton. Multi-timescale Nexting in a Reinforcement Learning Robot. In *From Animals to Animats 12 - 12th International Conference on Simulation of Adaptive Behavior (SAB)*, pages 299–309, 2012.

[61] Joseph Modayil, Adam White, Patrick M. Pilarski, and Richard S. Sutton. Acquiring a Broad Range of Empirical Knowledge in Real Time by Temporal-Difference Learning. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1903–1910. IEEE, 2012.

[62] P.M. Pilarski, M.R. Dawson, T. Degris, J.P. Carey, and R.S. Sutton. Dynamic Switching and Real-Time Machine Learning for Improved Human Control of Assistive Biomedical Robots. In *4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 296 –302, June 2012.

[63] A. White, J. Modayil, and R.S. Sutton. Scaling Life-Long Off-Policy Learning. In *International Conference on Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE*, pages 1–6, 2012.

[64] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[65] BA Posner. High-Throughput Screening-Driven Lead Discovery: Meeting the Challenges of Finding new Therapeutics. *Current Opinion in Drug Discovery and Development*, 8(4):487–494, 2005.

[66] Christopher P. Austin, Linda S. Brady, Thomas R. Insel, and Francis S. Collins. NIH Molecular Libraries Initiative. *Science*, 306(5699):1138–1139, November 2004.

[67] Alice McCarthy. The {NIH} Molecular Libraries Program: Identifying Chemical Probes for New Medicines. *Chemistry & Biology*, 17(6):549 – 550, 2010.

[68] Yanli Wang, Evan Bolton, Svetlana Dracheva, Karen Karapetyan, Benjamin A Shoemaker, Tugba O Suzek, Jiyao Wang, Jewen Xiao, Jian Zhang, and Stephen H Bryant. An Overview of the PubChem BioAssay Resource. *Nucleic Acids Research*, 38:D255–D266, 2010.

[69] Anuradha Roy, Peter R McDonald, Sitta Sittampalam, and Rathnam Chaguturu. Open Access High-Throughput Drug Discovery in the Public Domain: A Mount Everest in the Making. *Current Pharmaceutical Biotechnology*, 11(7):764, 2010.

[70] Torsten Meiners, Bahne Stechmann, and Ronald Frank. EU OPENSCREEN: Chemical Tools for the Study of Plant Biology and Resistance Mechanisms. *Journal of Chemical Biology*, 7(4):113–118, 2014.

[71] Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. ChEMBL: A Large-Scale Bioactivity Database for Drug Discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, jan 2012.

[72] James Inglese, Caroline E Shamu, and R Kiplin Guy. Reporting Data from High-Throughput Screening of Small-Molecule Libraries. *Nature Chemical Biology*, 3(8):438–441, 2007.

[73] Patricia L Whetzel, Helen Parkinson, Helen C Causton, Liju Fan, Jennifer Fostel, Gilberto Fragoso, Laurence Game, Mervi Heiskanen, Norman Morrison, Philippe Rocca-Serra, et al. The MGED Ontology: A Resource for Semantics-Based Description of Microarray Experiments. *Bioinformatics*, 22(7):866–873, 2006.

[74] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository. *Nucleic Acids Research*, 30(1):207–210, 2002.

[75] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene Ontology: Tool for the Unification of Biology. *Nature Genetics*, 25(1):25–29, 2000.

[76] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The OBO Foundry: Coordinated Evolution of Ontologies to Support Biomedical Data Integration. *Nature Biotechnology*, 25(11):1251–1255, 2007.

[77] Daniel Schober, Barry Smith, Suzanna E Lewis, Waclaw Kusnierczyk, Jane Lomax, Chris Mungall, Chris F Taylor, Philippe Rocca-Serra, and Susanna-Assunta Sansone. Survey-Based Naming Conventions for use in OBO Foundry Ontology Development. *BMC Bioinformatics*, 10(1):125, 2009.

[78] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an Ontology? In *Handbook on Ontologies*, pages 1–17. Springer, 2009.

[79] Sirarat Sarntivijai, Yu Lin, Zuoshuang Xiang, Terrence Meehan, Alexander Diehl, Uma Vempati, Stephan Schurer, Chao Pang, James Malone, Helen Parkinson, Yue Liu, Terue Takatsuki, Kaoru Saijo, Hiroshi Masuya, Yukio Nakamura, Matthew Brush, Melissa Haendel, Jie Zheng, Christian Stoeckert, Bjoern Peters, Christopher Mungall, Thomas Carey, David States, Brian Athey, and Yongqun He. CLO: The Cell Line Ontology. *Journal of Biomedical Semantics*, 5(1):37, 2014.

[80] Hajo Rijgersberg, Mark van Assem, and Jan Top. Ontology of Units of Measure and Related Concepts. *Semantic Web*, 4(1):3–13, January 2013.

[81] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[82] Kent A Spackman, Keith E Campbell, and Roger A Côté. SNOMED RT: a reference terminology for health care. In *Proceedings of the AMIA annual fall symposium*, pages 640–644. American Medical Informatics Association, 1997.

[83] Jeremy Rogers and Alan Rector. Galen's model of parts and wholes: experience and comparisons. In *Proceedings of the AMIA symposium*, pages 714–718. American Medical Informatics Association, 2000.

[84] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[85] Basic Formal Ontology (BFO) Project. Last visited on 07/06/2015.

[86] Holger Stenzhorn, Elena Beisswanger, and Stefan Schulz. Towards a Top-Domain Ontology for Linking Biomedical Ontologies. In Kuhn, Klaus A. and Warren, James R. and Leong, Tze-Yun, editor, *MedInfo*, volume 129 of *Studies in Health Technology and Informatics*, pages 1225–1229, , 2007. IOS Press.

[87] W. Ceusters and B. Smith. A realism-based approach to the evolution of biomedical ontologies. *Annual Symposium proceedings/AMIA Symposium. AMIA Symposium*, pages 121–125, 2006.

[88] Michel Dumontier and Robert Hoehndorf. Realism for scientific ontologies. In *Proceedings of the 2010 conference on Formal Ontology in Information Systems: Proceedings of the Sixth International Conference (FOIS 2010)*, pages 387–399, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[89] R R Brinkman, M Courtot, D Derom, J M Fostel, Y He, P Lord, J Malone, H Parkinson, B Peters, P Rocca-Serra, A Ruttenberg, S A Sansone, L N Soldatova, C J Stoeckert Jr., J A Turner, and J Zheng. Modeling biomedical experimental processes with OBI. *J Biomed Semantics*, 1(Suppl 1):S7, 2010.

[90] Uma D. Vempati, Magdalena J. Przydzial, Caty Chung, Saminda Abeyruwan, Ahsan Mir, Kunie Sakurai, Ubbo Visser, Vance P. Lemmon, and Stephan C. Schürer. Formalization, Annotation and Analysis of Diverse Drug and Probe Screening Assay Datasets Using the BioAssay Ontology (BAO). *PLoS ONE*, 7(11), November 2012.

[91] Ontology tool that Fetches ontology terms and axioms (OntoFox). `http://ontofox.hegroup.org/`. Last visited on 07/06/2015.

[92] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[93] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[94] Ubbo Visser, Saminda Abeyruwan, Uma Vempati, Robin Smith, Vance Lemmon, and Stephan Schurer. BioAssay Ontology (BAO): A Semantic Description of Bioassays and High-Throughput Screening Results. *BMC Bioinformatics*, 12(1):257+, 2011.

[95] Library of Integrated Network-based Cellular Signatures (LINCS) Information FramEwork. `http://life.ccs.miami.edu/life/`. Last visited on 07/06/2015.

[96] Stephan C Schürer, Uma Vempati, Robin Smith, Mark Southern, and Vance Lemmon. BioAssay Ontology Annotations Facilitate Aross-Analysis of Diverse High-Throughput Screening Data Sets. *Journal of Biomolecular Screening*, 16(4):415–426, 2011.

[97] Magdalena J. Przydzial, Barun Bhhatarai, Amar Koleti, Uma Vempati, and Stephan C. Schürer. GPCR Ontology: Development and Application of a G Protein-Coupled Receptor Pharmacology Knowledge Framework. *Bioinformatics*, 29:3211–3219, 2013.

[98] William Dabney and Andrew G Barto. Adaptive Step-Size for Online Temporal Difference Learning. In *AAAI Conference on Artificial Intelligence*, 2012.

[99] Harm V. Seijen and Rich Sutton. True Online TD(λ). In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 692–700. JMLR Workshop and Conference Proceedings, 2014.

[100] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[101] Richard S Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pages 871–878. ACM, 2007.

[102] Ashique Rupam Mahmood, Richard S Sutton, Thomas Degris, and Patrick M Pilarski. Tuning-free step-size adaptation. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 2121–2124. IEEE, 2012.

[103] Tim Kovacs and Robert Egginton. On the Analysis and Design of Software for Reinforcement Learning, with a Survey of Existing Systems. *Machine Learning*, 84(1-2):7–49, 2011.

[104] Richard S. Sutton. A Standard Interface for Reinforcement Learning Software in C++. `http://webdocs.cs.ualberta.ca/~sutton/RLinterface/RLI-Cplusplus.html`. Last visited on 07/06/2015.

[105] G.D. Konidaris, S. Osentoski, and P.S. Thomas. Value Function Approximation in Reinforcement Learning using the Fourier Basis. In *Proceedings of the 25th Conference on Artificial Intelligence*, pages 380–385, 2011.

[106] Richard S Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

[107] Matthew E. Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous Transfer for Reinforcement Learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 283–290, 2008.

[108] Thomas Degris, Patrick M Pilarski, and Richard S Sutton. Model-free reinforcement learning with continuous action in practice. In *American Control Conference (ACC)*, pages 2177–2182. IEEE, 2012.