

Advanced Reasoning about Dynamical Systems

Yilan Gu
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, ON, Canada M5S 3G4
yilan@cs.toronto.edu

ABSTRACT

My research focuses on advanced reasoning about dynamical systems. We propose a modified fragment of the situation calculus that can assure decidable reasoning and has potential applications to Semantic Web services. Recently, we also develop a framework to handle large taxonomies of actions compactly, which allow us to gain computational advantages. Now, we consider another improvement to the situation calculus using order-sorted logic. In the future, we will further consider their possible applications to other AI research areas.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous; D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures*

General Terms

Reasoning about Action and Change

1. INTRODUCTION

One of the most interesting and important problems in the research area of Artificial Intelligence (AI) is how to model actions and changes in the dynamic world efficiently. For example, problems involved in robotics, planning problems, software agents, database management and compositions of Web services all can be considered as events or actions involved in dynamical systems. It is not trivial to fully address the problem, as suggested by researchers in the AI community [11], a comprehensive theory of dynamical systems must at least accommodate some of the following aspects: the causal laws relating actions to their effects, the preconditions under which an action can be performed, exogenous and natural events, determining what to do and when to do, complex actions and procedures, discrete and continuous time, concurrency, and planning a course of actions, etc.

During the past two decades, a number of researchers in AI have developing mathematical and computational foundations for dynamical systems, among which the situation calculus [11] is a logical approach to provide designers a way to model and reason about dynamical systems with the high-level knowledge representations. It was first proposed by John McCarthy in 1963 [9], and later was evolved over a number of years in response to the needs of the University of Toronto Cognitive Robotics Project [11]. It provides a theoretical and implementation framework for autonomous robotic or software that reason, act and perceive in changing environments. It also serves as a foundation for many research projects in different areas. For instance, it is used to provide a well-defined semantics for Web services [10, 2] and a foundation for a high-level programming language Golog [7].

My current research focuses primarily on the following aspects of advanced reasoning in the situation calculus. First, we proposed a modified fragment of the situation calculus that can assure decidable reasoning and has potential applications to the Semantic Webs. This work has been published in [5] and some other proceedings of workshops. Recently, we developed a framework to handle large taxonomies of actions in a more compact way comparing to Reiter's situation calculus. It allows us to gain computational advantages. The overall accomplishment will appear in [6], a major AI conference this year. Currently, we consider another improvement to the situation calculus by using order-sorted logic, so that we can make the reasoning mechanism in the situation calculus more efficient. Other future work will also consider the combinations of the above techniques and their possible applications to other AI research areas.

2. BACKGROUND: THE SITUATION CALCULUS

The situation calculus is a first-order logic language for reasoning about dynamical systems [11]. All dialects of the situation calculus include three disjoint sorts (*actions, situations and objects*).

Actions are first-order terms consisting of an action function symbol and its arguments. Actions change the world. For example, action function $pickUp(x)$ represent an action of picking up object x . **Situations** are first-order terms which denote possible world histories. A distinguished constant S_0 is used to denote the *initial situation*, and function $do(a, s)$ denotes the situation that results from performing action a in situation s . Every situation corresponds

uniquely to a sequence of actions. For example, $do(pickUp(B_0), S_0)$ denotes the situation that results from picking up box B_0 in the situation S_0 . **Objects** are first-order terms other than actions and situations that depend on the domain of an application. For instance, B_0 is an object.

Fluents are relations or functions whose values may vary from one situation to the next. Normally, a fluent is denoted by a predicate or function symbol whose last argument has the sort situation. For example, $holding(x, do(pickUp(x), S_0))$ represents that the agent is holding x from execution of actions $pickUp(x)$ in S_0 .

A *basic action theory* (BAT) \mathcal{D} in the situation calculus is a set of axioms written in \mathcal{L}_{sc} with the following five classes of axioms to model actions and their effects [11].

Action precondition axioms \mathcal{D}_{ap} : The situation calculus includes the distinguished predicate $Poss(a, s)$ to characterize actions a that are possible to execute in s . For each action function $A(\vec{x})$, there is one axiom of the form $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$. For example, $Poss(pickUp(x), s) \equiv \neg \exists y holding(y, s)$, which means that it is possible to pick up object x in any situation s iff the agent is not holding anything in situation s .

Successor state axioms \mathcal{D}_{ss} : For each relational fluent $F(\vec{x}, s)$, there is one axiom that completely characterizes the truth value of F in the next situation $do(a, s)$ in terms of values that fluents have in the current situation s . For example, $holding(x, do(a, s)) \equiv a = pickUp(x) \vee holding(x, s) \wedge a \neq putDown(x)$, which means that the agent holds x after doing action a in situation s iff either a is action $pickUp(x)$ or it holds x in situation s and a is not action $putDown(x)$.

Initial theory \mathcal{D}_{S_0} : A set of first-order formulas whose only situation term is S_0 . It specifies the values of all fluents in the initial state. It also describes all the facts that are not changeable by any actions in the domain. For example, $\forall y. \neg holding(y, S_0)$ represents that initially the agent does not holding anything.

Unique name axioms for actions \mathcal{D}_{una} : Includes axioms specifying that two actions are different if their action names are different, and that identical actions have identical arguments.

Foundational axioms for situations Σ : The axioms for situations which characterize the basic properties of situations. These axioms are domain independent. They are included in the axiomatization of any dynamical system in the situation calculus (see [11] for details).

One of the most important reasoning tasks in the situation calculus is the projection problem, that is, to determine whether a given goal can be hold after executing a sequence of actions starting from the initial situation. Another basic reasoning task is the executability problem, that is, to determine whether or not a sequence of actions is executable starting from the initial situation. Planning and high-level program execution are two important settings where the executability and projection problems arise naturally.

Regression is a central computational mechanism that forms the

basis for automated reasoning in the situation calculus [11]. A recursive definition of the regression operator \mathcal{R} on any *regressible formula* ϕ is given in [11]. We use notation $\mathcal{R}[\phi]$ to denote the formula that results from eliminating $Poss$ atoms in favor of their definitions as given by action precondition axioms and replacing fluent atoms about $do(\alpha, s)$ by logically equivalent expressions about s as given by successor state axioms repeatedly until it cannot make such replacements any further. For instance, to determine whether $holding(O, do(A, S_0))$ for some object O and ground action A is the same as to determine whether $A = pickUp(O) \vee holding(O, S_0) \wedge A \neq putDown(O)$ by using the successor state axiom.

The regression theorem [11] shows that one can reduce the evaluation of a regressible formula W to a first-order theorem proving task in the initial theory together with unique name axioms for actions. This fact is the key result in the situation calculus. It demonstrates that an executability or a projection task can be reduced to a theorem proving task that does not use precondition, successor state, and foundational axioms. This is one of the reasons why the situation calculus provides a natural and easy way to represent and reason about dynamical systems.

3. DECIDABLE REASONING IN A MODIFIED SITUATION CALCULUS

3.1 Motivation

The motivation of this part includes two aspects. Firstly, the situation calculus is formulated in a general first-order logic, therefore the reasoning about effects of sequences of actions is generally undecidable under the open world assumption (OWA). We propose to use C^2 , a fragment of first-order logic, as a foundation for a modified situation calculus so that the reasoning about the executability and the projection problem (under the OWA) will be decidable in such language. Secondly, such modified language has a natural connection with description logics (DLs), which have been mostly used to describe static knowledge bases (KBs) in the Semantic Web. We use such modified language to formalize Semantic Web services.

C^2 is the two-variable first-order logic FO^2 extended with counting quantifiers $\exists^{\geq n}$ and $\exists^{\leq n}$ for any natural number $n \geq 1$. FO^2 is the fragment of the ordinary first-order logic (with equality), in which function symbols are not allowed and the formulas only use no more than two variables x and y (free and bound). The satisfiability problem for C^2 is decidable and recently Pratt-Hartmann proves that this problem is in NEXPTIME even when counting quantifiers are coded succinctly.

Description Logics (DLs) [1] are a well-known family of knowledge representation formalisms, which play an important role in providing the formal foundations of several widely used Web ontology languages including OWL in the area of the Semantic Web. However, DLs are mostly used to describe static KBs. The research of formalizing and reasoning about web services (which can be considered as actions) in the Semantic Web environment is still at the beginning stage (e.g., [10]).

3.2 Our approach

Based on the above observation, we propose the following modeling and reasoning approach.

Formalizing a modified situation calculus \mathcal{L}_{sc}^{DL} In [5], we restrict the syntax of the object terms used in the situation calculus to be either variable x , variable y or constants. All fluents and action functions considered in the modified language have no more than two object arguments. We require that the syntax of the axioms in a basic action theory (BAT) should follow certain restrictions. For example, the right hand-side formula for each precondition axiom is a C^2 formula if the situation argument is suppressed and each sentence in the initial theory should be a C^2 formula. We also introduce new classes of axioms similar to TBox and ABox in DLs to achieve integration of taxonomic reasoning and reasoning about actions.

A modified regression operator Based on the above modified formalization of the situation calculus, a modified regression operator \mathcal{R} is defined (see [5]) in \mathcal{L}_{sc}^{DL} . It is defined recursively to (1) allow *dynamic* TBox axioms are expanded during regression using *lazy unfolding technique* [1] and (2) assure the regression result of a regressable C^2 formula (whose situation term is ground) is a C^2 formula uniform in S_0 . Similar to the regression theorem in [11], given a BAT D and a regressable formula W in \mathcal{L}_{sc}^{DL} , we prove that $D \models W$ iff the initial theory D_{S_0} entails the regression result of W using the modified regression. We therefore prove that the projection problem and the executability problem (under the OWA) in \mathcal{L}_{sc}^{DL} is decidable. (Details are provided in [4, 5].) We also give some simple examples to illustrate the representation power of \mathcal{L}_{sc}^{DL} and the modified regression.

Progression problem in the language \mathcal{L}_{sc}^{DL} After doing longer and longer sequences of actions, solving projection problems and/or executability problems becomes increasingly more difficult. Hence, it is beneficial to progress the initial incomplete knowledge base (KB) to represent the current state of the world. Then, the subsequent projection problems can be solved with respect to a new progressed KB. The task of computing a progressed KB is called the progression problem. Therefore, we are also interested in studying the progression problem in \mathcal{L}_{sc}^{DL} .

In [5], given a *local-effect* BAT¹ in \mathcal{L}_{sc}^{DL} , we consider the progression of a certain type of incomplete KB, which is called as *CNF-based KB* and is defined as a set of disjunction of *equality-based formulas*². We provide a progression algorithm (called *modified progression below*) and show that the resulting KB is still CNF-based, and it is (*classically*) *sound*, i.e., any model of the *classical progression* (defined in [11]) of the current KB is a model of the modified progression of the current KB. In general, the modified progression of a CNF-based KB in \mathcal{L}_{sc}^{DL} (wrt a local-effect BAT) is not

¹The detailed definition of a local-effect BAT can be found in [11, 8, 5].

²An equality-based formula is a formula of the form $\forall \vec{x} e(\vec{x}) \supset l(\vec{x}, S)$ where $e(\vec{x})$ is a quantifier-free boolean formula with inequality and equality only and $l(\vec{x}, S)$ is a fluent literal [8].

(*classically*) *complete*, i.e., any model of the modified progression of the current KB is a model of the classical progression of the current KB.

3.3 Possible Future Directions

In the future, we will try to identify whether the modified progression of the current CNF-based KB (wrt a given BAT) can be (*classically*) complete with some additional conditions, for example, the context-complete KBs defined in [8]. We will also try to determine whether the progression of a *finite* theory in \mathcal{L}_{sc}^{DL} is first-order definable or not.

4. MODULAR BASIC ACTION THEORIES

4.1 Motivation

The second part of my research is to propose a possibly more compact representation of BATs. The motivation is twofold. The first comes from the intention to design a representation that allows writing more compact and modular BAT than it is currently possible, and also BAT that will be easier to elaborate and expand to new domains. BAT have not been designed to support other forms of reasoning, e.g., taxonomic reasoning about objects and actions. Essentially, these theories are “flat” and do not provide representation for hierarchies of actions or objects. This can lead to potential difficulties if one intends to use BAT for the purposes of large scale formalization of reasoning about actions on the common-sense level, when potentially arbitrary actions and objects have to be represented. Intuitively, many events and actions have different degrees of generality: the action of driving a car from home to an office is specialization of the action of transportation using a vehicle, that is in its turn specialization of the action of moving a thing from one location to another. We intend to represent hierarchies among actions and use them in new hierarchical BAT. Our second motivation comes from OpenCyc [3]: the open source version of the Cyc technology, the world’s largest general knowledge base and common-sense reasoning engine. OpenCyc has extensive taxonomies for things, and elaborated, but unsystematic taxonomies for events and actions. However, there is no formal semantics for solving the frame problem in OpenCyc using BATs, for solving executability and projection problems using regression, and to the best of our knowledge, there is no formal specification for implementing reasoning about actions in OpenCyc. Hence, we want to provide a possible formal semantics for reasoning about actions in OpenCyc by using Reiter’s BAT.

4.2 Our Approach

The proposed research includes the following aspects, and details can be found in [6].

Representing actions hierarchically We propose to classify events and actions hierarchically similar to the representations in Cyc. We introduce a finite set of axioms using a newly introduced predicate *specification*(a_1, a_2) which represents that action a_1 is a *direct specialization* of action a_2 (action a_2 is a *direct generalization* of a_1). Such relationships between actions specified by the axioms can be illustrated using an acyclic digraph, hence the set of axioms represents an action hierarchy. We consider an acyclic diagram instead of

a tree, because we want to allow multiple inheritances between actions in a general case. The (*distant*) *specification* $isA(a_1, a_2)$ is a reflexive-transitive closure of *specification*, which can be illustrated by a path from a_2 to a_1 in the action hierarchy digraph. For any action term α , $\forall a.isA(a, \alpha)$ generally represents a group of actions which are specifications of α .

A more compact modular representation of BATs Based on the specified action hierarchy, we modify the syntax of BATs so that we can possibly represent dynamical systems in a more compact way. For instance, instead of giving one precondition axiom for each action function, we specify the precondition axiom for each group of actions classified using predicate isA . When writing the successor state axiom for each fluent, instead of specifying positive or negative conditions for each action individually, we provide positive or negative conditions for each group of actions classified using predicate isA .

Correctness and advantages of the new representation We show that our new modular BATs can be potentially expanded into “flat” Reiter’s BATs; consequently, they are conservative extensions of Reiter’s BATs: every entailment from Reiter’s BAT remains entailment in hierarchical BAT. Moreover, we show that our axioms can be more succinct, but an extended Reiter’s regression operator can still be used to solve the projection problem. We also show that our representation has significant computational advantages. For taxonomies of actions that can be represented as finitely branching trees, the regression operator can sometimes work exponentially faster with our theories than it works with Reiter’s BATs.

Finally, we propose general guidelines on how a taxonomy of actions can be constructed from the given set of effect axioms if these axioms characterize completely properties of actions in a domain.

4.3 Possible Future Directions

We may consider if it is possible to reuse the modular BAT for a dynamical system in constructing the BAT for another system, if the former one is a sub-system of the later one. We will explore logical conditions that make this transfer from one domain to another possible.

5. AN ORDER-SORTED SITUATION CALCULUS

5.1 Motivation

My current ongoing research involves considering further improvement of the efficiency of the regression, the central reasoning mechanism in the situation calculus. The current situation calculus is based on a many-sorted logic with only three disjoint sorts, which does not consider the distinction and taxonomic reasoning about objects. However, in reality, autonomous agents need to perform actions based on suitable objects. For example, it is impossible to fry fish in a paper cup. Hence, the dynamical systems need to include taxonomical information and assertional information about

classes of objects, and the agents should be able to perform reasoning about actions and their effects given such additional information.

Moreover, studies show that the extension of first-order logic with sorts has several advantages over the standard first-order logic. For instance, it supports typed computation, allows for more compact and more natural formalizations of AI problems, and the exploitation of the sort information by sorted unification can result in drastic search space reductions [12, 13].

5.2 Our Approach

Our general idea is to extend Reiter’s situation calculus with the *order-sorted logic*. In particular, we propose the following work.

A basic action theory with sort theory First, we distinct objects as different classes, called *sorts*. For each dynamical system, we axiomatize a basic action theory based on order-sorted logic. Such basic action theory includes a sort theory that has subsort declarations and term declarations, particularly for action functions and fluents. For example, the argument x of an action function $pickUp(x)$ should be a class of objects that are movable, sort *man* needs to be declared as a subsort of *human* if the considered dynamical system involves both sorts. The first argument and the second argument of fluent $cooked(x, s)$ should be of sort *food* and *situation*. Moreover, the precondition axioms and the successor state axioms in the basic action theory are specified for actions and fluents with suitable sorts using well-sorted formulas.

A modified regression operator Second, we define investigate what is a regressable formula under such framework. We consider extending the current regression operator with well-sortedness checking and unification techniques. With the modified regression operator, we expect to gain computational efficiency by terminating the regression earlier when reasoning about actions or effects for unreasonable sorts. For instance, when asking whether or not a fork is cooked after executing a long sequence of actions, agents will answer “*false*” and terminate the reasoning in the first step of regression by using our regression operator because of the unsuitable sort. However, with Reiter’s regression operator, the regression will continue step by step for the sequence of actions until reaching the initial situation, and then perform reasoning on the regressed formula using the initial theory and give the answer.

6. OTHER RESEARCH DIRECTIONS

Building on our current work, many other future research directions can be explored. Depending on considerations of different possible application areas, such as web service compositions with large taxonomies of actions, planning recognitions, and common-sense taxonomic reasoning about objects and events, we may consider the possibilities of combining either two or three themes of the decidable fragment of the situation calculus, the hierarchical representation of actions and the order-sorted situation calculus. Another future research direction is the efficient implementation of our work in practical scenarios. For instance, it would be interesting to

see how our modified situation calculus can be used in real applications along the lines of SNAP, an e-commerce ontology developed in IBM for an automated system for recommending products and services in the domains of banking, insurance and telephony.

Intelligence, 18, 1996.

7. REFERENCES

- [1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [2] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella. e-service composition by description logics based reasoning. In D. Calvanese, G. de Giacomo, and E. Franconi, editors, *Proceedings of the 2003 International Workshop in Description Logics (DL-2003)*, Rome, Italy, 2003.
- [3] Cycorp. Opencyc 1.0: www.opencyc.org, 2002?
- [4] Y. Gu and M. Soutchanski. A logic for decidable reasoning about services. In *Proc. of AAAI-06 workshop on AI-Driven Technologies for Services-Oriented Computing*, Boston, USA, July 2006. <http://www.cs.toronto.edu/~yilan/publications/papers/aaai06.pdf>.
- [5] Y. Gu and M. Soutchanski. Decidable reasoning in a modified situation calculus. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 1891–1897, Hyderabad, India, January 2007.
- [6] Y. Gu and M. Soutchanski. Reasoning about large taxonomies of actions. In *Proc. of the twenty-third AAAI Conference on Artificial Intelligence (AAAI-08)*, Chicago, July 2008. AAAI Press.
- [7] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.
- [8] Y. Liu and H. J. Levesque. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. IJCAI-05*, Edinburgh, Scotland, August 2005.
- [9] J. McCarthy. Situations, actions and causal laws. Technical Report Technical Report Memo 2, Stanford University Artificial Intelligence Laboratory, Stanford, CA, 1963. Reprinted in Marvin Minsky, editor, *Semantic information processing*, MIT Press, 1968.
- [10] S. McIlraith and T. Son. Adapting Golog for composition of semantic web services. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, pages 482–493, Toulouse, France, April 22-25 2002. Morgan Kaufmann.
- [11] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.
- [12] M. Schmidt-Schauß. *Computational aspects of an order-sorted logic with term declarations*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.
- [13] C. Weidenbach. Unification in sort theories and its applications. *Annals of Mathematics and Artificial*