ADVANCED REASONING ABOUT DYNAMICAL SYSTEMS

by

Yilan Gu

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 2010 by Yilan Gu
Abstract

Advanced Reasoning about Dynamical Systems

Yilan Gu

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2010

In this thesis, we study advanced reasoning about dynamical systems in a logical framework – the situation calculus. In particular, we consider promoting the efficiency of reasoning about action in the situation calculus from three different aspects.

First, we propose a modified situation calculus based on the two-variable predicate logic with counting quantifiers. We show that solving the projection and executability problems via regression in such language are decidable. We prove that generally these two problems are co-NExpTime-complete in the modified language. We also consider restricting the format of regressable formulas and basic action theories (BATs) further to gain better computational complexity for reasoning about action via regression. We mention possible applications to formalization of Semantic Web services.

Then, we propose a hierarchical representation of actions based on the situation calculus to facilitate development, maintenance and elaboration of very large taxonomies of actions. We show that our axioms can be more succinct, while still using an extended regression operator to solve the projection problem. Moreover, such 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 the BATs current situation calculus. We also propose a general guideline on how a taxonomy of actions can be constructed from the given set of effect axioms.
Finally, we extend the current situation calculus with the order-sorted logic. In the new formalism, we add sort theories to the usual initial theories to describe taxonomies of objects. We then investigate what is the well-sortness for BATs under such framework. We consider extending the current regression operator with well-sortness checking and unification techniques. With the modified regression, we gain computational efficiency by terminating the regression earlier when reasoning tasks are ill-sorted and by reducing the search spaces for well-sorted objects. We also study that the connection between the order-sorted situation calculus and the current situation calculus.
Acknowledgements

It is a pleasure to thank all people who have helped and inspired me during my doctoral study.

First and foremost I would like to express my sincerest gratitude to my thesis advisor, Prof. Mikhail Soutchanski, who has been extremely supportive throughout my doctoral program with his knowledge and patience whilst allowing me the room to work in my own way. Dr. Soutchanski gave me both academic and emotional support to help me get through difficult times. Without his encouragement, sound advice and good company, this thesis would not have been completed or written. I also would like to thank him and his family for providing me with a friendly environment, which helped to make my research life become more rewarding.

It is difficult to overstate my gratitude to my other Ph.D. committee members and advisors, Prof. Hector Levesque, Prof. Sheila McIlraith and Prof. Fahiem Bacchus, who have set up a supportive research environment for me and given me good suggestions and comments on my work. In particular, I wish to thank Prof. Levesque, my official supervisor at the University of Toronto, for providing me with financial support to academic conferences, listening to my problems and giving me many helpful suggestions during different stages of my doctoral studies. I wish to thank Prof. McIlraith for being very compassionate as an experienced female academic. She offers tremendous support to female students in the department including myself. I wish to thank Prof. Bacchus for his encouragement. I also want to thank Prof. Craig Boutilier for providing guidance and financial support in the early years of my doctoral program.

I am indebted to many former and current student colleagues and friends for providing a stimulating and fun environment in which to learn and grow. I am grateful to Yongmei Liu, Stavros Vassos, Iluju Kiringa and Scott Sanner for useful discussions. I would especially thank Jessica Davies for helping me to improve my English writing and taking time to read and edit a journal version of my work in Chapter 3 of this dissertation.
My sincere thanks are due to many academic colleagues and professors from other universities: Prof. Giuseppe De Giacomo, Prof. Carsten Lutz, Prof. Eyal Amir, Prof. Franz Baader, Prof. Ian Horrocks, Prof. Yan Zhang, Prof. Leonid Libkin, Dr. Antonina Kolokolova, Dr. Birte Glimm, Dr. Hongkai Liu and Dr. Meghyn Bienvenu. They answered my research questions and provided me with useful discussions and comments to my work. I thank Prof. Michael Gruninger from the department of Mechanical and Industrial Engineering for letting me attend his research discussion group, which provided me with an inspiring atmosphere.

I also wish to thank Shahine M. Zoubeiri and Killian Thomson, two students at the University of Toronto, for helping to revise the English grammar in my manuscript.

I owe my loving thanks to my entire extended family. In particular, I am very grateful to my parents-in-law, Zhijian Wang and Yue Cai, who treated me as their own daughter and gave me a hand with my newborn daughter in 2008. I am especially indebted to my parents, Zhenhua Gu and Jinying Shi, for their unconditional love and care throughout my life. They have always been extremely supportive and went far to Canada to help me out when I was overwhelmed with my studies and family responsibilities.

I want to mention my little daughter Sophia S. Wang. Her wonderful smile cheered me up when I was tired. Last but not least, my deepest gratitude goes to my husband Xuhui Wang. His unflagging love, support, encouragement and understanding helped me get through many stressful times and made this work possible.
# Contents

1 Introduction 1

2 Background 6

2.1 The Situation Calculus 6

2.1.1 The Language of Situation Calculus 6

2.1.2 The Basic Action Theory 8

2.1.3 The Regression Operator 12

2.2 Description Logics and Two-Variable First-Order Logics 15

2.2.1 Description Logics 15

2.2.2 $C^2$ and Its Relationship to Description Logics 19

2.3 Order-Sorted Logic 24

2.3.1 The Syntax of Order-Sorted Logic 24

2.3.2 Well-sortness and the Semantics of Order-Sorted Logic 26

3 Decidable Reasoning in a Modified Situation Calculus 30

3.1 Motivations 31

3.2 Modeling Dynamical Systems in a Modified Situation Calculus 34

3.3 Reasoning about Actions using Regression 40

3.3.1 Modified Regression with Lazy Unfolding 40

3.3.2 Some Computational Complexity Results via Regression 49

3.3.3 Examples of BATs and Regression in $L_{sc}^{C^2}$ 54
B Proofs of Lemmas and Theorems in Chapter 3

B.1 The Correctness of the Modified Regression Operator .......................... 199
B.2 $\mathcal{ALCO}(U)$ and $FO_{DL}$ are Equally Expressive .......................... 203
B.3 Restricting Syntax of BATs to Gain Computational Advantages ............. 205
Chapter 1

Introduction

Artificial Intelligence (AI) is a broad and interesting research area with many open questions. One important problem is how to model and reason about dynamical systems efficiently. Such research has broad applications. Reasoning problems in robotics, planning problems and compositions of Web services can all be considered as events or actions involved in a dynamical environment.

Since 1992, the research group led by Dr. R. Reiter and Dr. H. Levesque at the University of Toronto developed a systematic way to model and reason about dynamical systems with a high-level knowledge representation approach [141]. This logical approach is known as the situation calculus and was first proposed by John McCarthy in 1963 [114]. The work at the University of Toronto develops many features of dynamical system modeling, including time, processes, concurrency, exogenous events, reactivity, sensing and knowledge, probabilistic uncertainty, and decision theory [141]. For instance, it provides a foundation for a high-level programming language Golog [101]. Recently, it has also been applied to other research areas. For example, it provides a well-defined semantics for Web services [116, 11] and is used to model cryptographic protocols [40].

The language of the situation calculus is formulated in a general predicate logic. A dynamic world in the situation calculus is modeled as progressing through a series of
situations as a result of various actions being performed within the world. A situation represents a history of action occurrences. Functional terms and logical predicates are used to represent actions and systematic properties that could possibly be affected by actions. In particular, a set of axioms, the basic action theory (BAT), is provided to describe the initial state of a system, the preconditions of executing actions, and how actions affect system properties.

Based on BATs, one can consider different reasoning problems about action and change via logical deduction. For instance, we may check whether a goal specified using first-order sentence can be satisfied after executing a sequence of ground actions starting from the very beginning (this is known as the projection problem). We may also ask questions like whether a sequence of ground actions is executable starting from the very beginning (this is known as the executability problem). We could also deal with planning problems in the situation calculus by solving queries like whether there exists a sequence of actions such that a goal can be achieved after executing the sequence of actions. One of the central reasoning mechanisms used to reason about consequences of actions in the situation calculus is called regression. Using regression, to determine whether a formula is true at the current situation is to determine whether some conditions are true in the previous situation only. By iterating this procedure, one can end up with an equivalent formula depending only on the initial situation. The way how a basic action theory is specified supports regression in a natural way.

The situation calculus provides a systematic and powerful framework for reasoning about actions and their effects. Because it is natural for modeling action and change, we consider to formalize and reason about Web services of Semantic Web [12] in the situation calculus. Semantic Web is based on Description Logics (DLs) [4], which are a family of logics mostly used to describe static knowledge bases. Many expressive DLs offer considerable expressive power going far beyond propositional logic, while ensuring decidable reasoning [18]. However, reasoning about effects of sequences of actions in the
situation calculus is generally undecidable. This motivates us to look for a fragment of the situation calculus so that it has a natural connection with description logics, can describe Web services in a natural way and can make decidable reasoning at least for problems such as the executability problem and the projection problem. This leads to the first part of my research work presented in this thesis.

How to represent and reason about effects of actions grouped in a realistically large taxonomy is another important problem in AI. Although the problem of representing large semantic networks of (static) concepts has been addressed in AI since the 1970s [19] and served as motivation for research on description logics, representing and reasoning about large taxonomies of actions received surprisingly little attention. In the situation calculus, successor state axioms in BATs are used to describe effects of actions for each fluent (property of a dynamical systematic whose truth value may vary in different situations) in a compact way. However, BATs have not been designed to support taxonomic reasoning about objects and actions. In particular, the axioms are “flat” in the sense that the context conditions and effects of actions are specified individually in successor state axioms and BATs in Reiter’s situation calculus do not provide any representation for hierarchies of actions. Intuitively, many events and actions have different degrees of generality: the action of driving a car from home to an office is a specialization of the action of transportation using a vehicle, that is in turn a specialization of the action of moving an object from one location to another. We consider representing specialization relationships between actions using logical axioms to construct hierarchies of actions, and formulate new successor state axioms based on action hierarchies. By doing so, we are able to represent effects of actions in an even more compact way, and additionally such representation sometimes bring us computational benefit in reasoning via regression. That’s the essential motivation of the second part of my research.

The third part of my research considers further improvement of the efficiency of the regression, the central reasoning mechanism in the situation calculus. Note that the cur-
rent situation calculus does not consider the classification of objects. However, studies show that the extension of first-order logic with sorts, such as order-sorted logic, has several advantages over the standard first-order logic. It “allows for more compact and more natural formalizations of AI problems, and can result in drastic search space reduction [171]”. For instance, in deductive databases, Reiter [138] uses boolean combination of monadic predicates to express taxonomies of types (each simple type is represented by a monadic predicate) and develops a typed unification algorithm, which is considered to be more suitable for real world databases than the approach in deductive question-answering research that deals with unrestricted first-order databases. More recently, this line of research results in the DL-Lite framework that is designed for efficient query answering with respect to very large assertion axioms (ABoxes) [23]. As another example, in [130, 166, 170, 171], researchers show that the exploitation of the sort information by sorted unification can result in drastic search space reductions in automated theorem proving. We consider combining order-sorted logic with the current situation calculus. We describe sub-universes of objects (i.e., sorts), action functions and systems properties using sort theories, and formalize BATs using well-sorted formulas based on sorts. Finally, we take advantage of well-sorted unification during regression to terminate regression earlier sometimes.

The rest of the thesis is organized as follows. In Chapter 2, we review the relevant background literature, such as the situation calculus, description logics, the two-variable fragments of the first-order logic with counting quantifiers, and order-sorted logic. In Chapter 3, we consider a fragment of the situation calculus, in which solving the projection problem and the executability problem are decidable. In Chapter 4, we propose a logical representation of large taxonomies of actions, and provide a new way to formalize basic action theories based on the action hierarchies to gain computational advantages. In Chapter 5, we combine the situation calculus with order-sorted logic, and consider the computational advantage of reasoning about action via regression in the modified frame-
work. Finally, in Chapter 6, we conclude with a summary of the thesis and a discussion of some future research directions.

The results of Chapter 3 that considers decidable reasoning in the situation calculus will appear in [70]. Some main results in Section 3.2 and in Section 3.3 appeared in [66], and some preliminary results of this research were published in several other international workshops [64, 65, 67]. The results of Chapter 4 dealing with large taxonomies of actions, have been published in [68]. Finally, a preliminary version of Chapter 5 that considers adopting order-sorted logic into the situation calculus appears in [69].
Chapter 2

Background

In this chapter, we review the background knowledge of the work presented in the following chapters, which include the situation calculus, description logics, $C^2$ (a fragment of first-order logic) and order-sorted logic.

2.1 The Situation Calculus

2.1.1 The Language of Situation Calculus

The basic conceptual and formal ingredients of the situation calculus were first proposed by McCarthy in 1963 [114]. Based on several researchers' study and proposals [133, 33, 72, 153, 45], Reiter [139] provided a solution to the frame problem observed by McCarthy and Hayes [114], and systematically described approaches based on the situation calculus to modeling dynamic worlds [141]. Since the early 1990’s, under the leadership of R. Reiter and H. Levesque, the Cognitive Robotics Group\(^1\) at University of Toronto has been using the situation calculus as a foundation for practical work in planning, control, simulation, etc, which disabuses some limiting views of the situation calculus [141].

The language of the situation calculus $\mathcal{L}_{sc}$ that we adopt here is from [141], specifically

\(^1\)The group website is available at: http://www.cs.utoronto.ca/cogrobo/.
Chapter 2. Background

designed for representing a dynamically changing world, which is mainly a three-sorted first-order language with a few second-order axioms. The three disjoint sorts are:

- **Action**: a first-order term representing actions in dynamic worlds, such as *jump* (the action of jumping), *pickup(x)* (picking up object *x*), and *putOn(r, x, y)* (robot *r* putting object *x* on top of object *y*), etc. The constant and function symbols for actions are completely application-dependent.

- **Situation**: a first-order term which denotes possible world histories. A distinguished constant *S₀* and function symbol *do* are used. *S₀* denotes the initial situation, before any action has been performed; *do(a, s)* denotes the situation that results from performing action *a* in situation *s*.

- **Object**: a catch-all sort representing for everything else depending on the domain of application, such as *Ball₀, Mary*, etc.

In fact, every situation corresponds to a sequence of actions. For example, the initial situation *S₀* corresponds to empty sequence of actions, the situation

\[
\text{do(pickup}(x), \text{do(drop}(y), \text{do(pickup}(y), S₀)))
\]

corresponds to the sequence of actions *pickup(y), drop(y)* and *pickup(x)* from the beginning. Moreover, we will use binary relation *s < s’* to represent that *s* is a proper sub-history of *s’*, and *s \preceq s’* is an abbreviation of *s = s’ ∨ s < s’*.²

Moreover, for any situation term *σ*, a term (a formula, respectively) is called uniform in *σ* iff *σ* is the only situation term in the term (the formula, respectively) if there is any.

Another important term in the situation calculus is fluent. Fluents are predicates (relational fluents) and functions (functional fluents) whose values may vary from situation to situation, used to describe what holds in a situation. By convention, the last

²In [141], the symbol ⊂ was used instead of < and ⊆ was used instead of ≤. We use new symbols here in order to distinguish them from the general inclusion axiom symbol for concepts and roles used in description logics [4].
argument of a fluent is a situation. For example, the fluent $\text{Holding}(r, x, s)$ might stand for the relation of robot $r$ holding object $x$ in situation $s$.

A functional fluent term is prime if it has the form $f(\vec{t}, do([\alpha_1, \ldots, \alpha_n], S_0))$ for $n \geq 1$ and each of the terms $\vec{t}, \alpha_1, \ldots, \alpha_n$ is uniform in $S_0$. Thus, for prime functional fluent terms, $S_0$ is the only term of sort situation (if any) mentioned by $\vec{t}, \alpha_1, \ldots, \alpha_n$.

The logical symbols of the language are $\neg, \land, \exists$. Other connectives and the universal quantifier are the usual abbreviations.

Finally, a distinguished predicate $\text{Poss}(a, s)$ is used to state that action $a$ can be performed in situation $s$. For example, $\text{Poss}(\text{pickup}(R, O), S_0)$ says that $R$ is able to pick up $O$ in the initial situation $S_0$.

This completes the specification of the language $L\text{sc}$. For later convenience, an abbreviation is introduced as follows ([141] Chapter 4.5):

- $do([], s) \stackrel{\text{def}}{=} s$;
- $do([a_1, a_2, \cdots, a_n], s) \stackrel{\text{def}}{=} do(a_n, do(\cdots, do(a_1, s) \cdots))$.

$[a_1, a_2, \cdots, a_n]$ is called a log. Notice that there is a one-to-one correspondence between a log beginning at the initial time and a situation.

### 2.1.2 The Basic Action Theory

A basic action theory (BAT) $\mathcal{D}$ is a set of axioms represented in the situation calculus to model the actions and their effects in a given dynamic system. Here we present a summary. The detailed explanation can be found in [134, 141].

The set $\mathcal{D} = \Sigma \cup \mathcal{D}_{\text{ap}} \cup \mathcal{D}_{\text{ss}} \cup \mathcal{D}_{\text{una}} \cup \mathcal{D}_{S_0}$ consists of following axioms:

- Foundational axioms for situations, denoted as $\Sigma$.

There are four axioms included in $\Sigma$ [134].

1. $\neg s \prec S_0$ says that no situation is a proper history of the initial situation.
2. $s \prec do(a, s') \equiv s \preceq s'$ means that any situation $s$ is a proper history of $do(a, s')$ iff $s$ is a history $s'$. 

3. \( \text{do}(a_1, s_1) = \text{do}(a_2, s_2) \supset a_1 = a_2 \land s_1 = s_2 \) is the unique name axiom for situations.

4. \( (\forall P). P(S_0) \land (\forall a, s)[P(s) \supset P(\text{do}(a, s))] \supset (\forall s)P(s) \) is a second-order induction axiom on situations.

Since the foundational axioms are always the same for all BATs, we will not write them out during description, but assume them to be true for any BATs.

- **Unique name axioms for actions**, denoted as \( \mathcal{D}_{\text{ana}} \).

For any \( n \)-ary action function symbol \( A \),

\[
A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \supset x_1 = y_1 \land \cdots \land x_n = y_n.
\]

For any \( n \)-ary \( (n \in \mathbb{N}) \) function \( A(x_1, \ldots, x_n) \) and any \( m \)-ary \( (m \in \mathbb{N}) \) function \( A'(y_1, \ldots, y_m) \) such that \( A \neq A' \),

\[
A(x_1, \ldots, x_n) \neq A'(y_1, \ldots, y_m).
\]

Since the unique name axioms are always in a standard format, we will not write them out during description, but assume them to be true for any BATs.

- **Initial theory**, denoted as \( \mathcal{D}_{S_0} \).

It is a set of first-order sentences that are uniform in \( S_0 \). Notice that the initial theory may contain sentences mentioning no situation term at all, for example, unique names for individuals, or “timeless” facts like \( \text{dog}(x) \supset \text{mammal}(x) \).

- **Action precondition axioms**, denoted as \( \mathcal{D}_{\text{ap}} \).

For each \( n \)-ary \( (n \in \mathbb{N}) \) action function \( A(\bar{x}) \) \( (\bar{x} = (x_1, x_2, \ldots, x_n)) \), there is one axiom of the form:

\[
\text{Poss}(A(\bar{x}), s) \equiv \Pi_A(\bar{x}, s), \quad (2.1)
\]
where \( \Pi_A(\vec{x}, s) \) is a first-order formula uniform in \( s \) with free variables at most among \( \vec{x}, a \) and \( s \). This axiom characterizes the preconditions of performing action \( A \) in the current situation \( s \).

- **Successor state axioms**, denoted as \( D_{ss} \).

  A successor state axiom (SSA) for an \((n + 1)\)-ary \((n \in \mathbb{N})\) relational fluent \( F(\vec{x}, s) \) \((\vec{x} = \langle x_1, \cdots, x_n \rangle)\) is a first-order sentence of \( L_{sc} \) of the form:

  \[
  F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s),
  \]

  where \( \Phi_F(\vec{x}, a, s) \) is a formula uniform in \( s \) with free variables at most among \( \vec{x}, a \) and \( s \). Similarly, a successor state axiom for an \((n + 1)\)-ary functional fluent \( f(\vec{x}, s) \) \((\vec{x} = \langle x_1, \cdots, x_n \rangle)\) is a first-order sentence of \( L_{sc} \) of the form:

  \[
  f(\vec{x}, do(a, s)) = y \equiv \Phi_f(\vec{x}, y, a, s),
  \]

  where \( \Phi_f(\vec{x}, y, a, s) \) is a formula uniform in \( s \) with free variables at most among \( \vec{x}, y, a \) and \( s \). \(^3\)

The successor state axiom for fluent \( F \) (\( f \), respectively) completely characterizes the value of fluent \( F \) (\( f \), respectively) in the successor resulting from performing a primitive action \( a \) in the situation \( s \). Notice that the models we consider all satisfy the *Markov* property – the truth values of the fluents at the next situation are dependent only on the action and the truth values of the fluents at the current situation.

Consider any BAT \( \mathcal{D} \) which includes finitely many fluents. The *situation-suppressed* terms and formulas are terms and formulas in which the situation arguments of fluents have been removed. We sometimes write a formula in the form of \( \phi[\sigma] \), where \( \phi \) is a

---

\(^3\)Any BAT \( \mathcal{D} \) that has SSAs for functional fluents needs to satisfy the *functional fluent consistency property*, which ensures the conditions defining \( f \)'s value in the next situation, namely \( \Phi_f \), actually define a unique value for \( f \) [141]. The solution to the frame problem for functional fluents of Section 3.2.6 in [141] does satisfy this property.
situation-suppressed formula, $\sigma$ is a situation term, and $\phi[\sigma]$ is the formula with situation term $\sigma$ restored defined recursively as follows.

**Definition 1** For any situation-suppressed formula $\phi$ and situation term $\sigma$ with respect to a background BAT $D$, the formula $\phi[\sigma]$ is obtained by *restoring* situation term $\sigma$ back, which is defined as:

1. If $\phi$ is atomic and is of the form $F(\vec{t})$ where $F$ is a relational fluent symbol in the language of $D$, then $\phi[\sigma] \overset{\text{def}}{=} F(\vec{t}, \sigma)$.
2. If $\phi$ is atomic and includes terms $f_1(\vec{t}_1), \cdots, f_m(\vec{t}_m)$ where $f_i (i = 1..m)$ are function fluent symbols in the language of $D$, then $\phi[\sigma] \overset{\text{def}}{=} \phi', \text{ where } \phi'$ is obtained by replacing $f_i(\vec{t}_i)$ with $f_i(\vec{t}_i, \sigma)$ for each $i (i = 1..m)$.
3. If $\phi$ is not atomic, and is of the form $(\forall x)\phi_1(\vec{x})$ (or $(\exists x)\phi_1(\vec{x})$, respectively) for some variable $x$, then $\phi[\sigma] \overset{\text{def}}{=} (\forall x)(\phi_1[\sigma])$ (or $\phi[\sigma] \overset{\text{def}}{=} (\exists x)(\phi_1[\sigma])$, respectively.)
4. If $\phi$ is not atomic, and is of the form $\neg \phi_1$, then $\phi[\sigma] \overset{\text{def}}{=} \neg(\phi_1[\sigma])$.
5. If $\phi$ is not atomic, and is of the form $\phi_1 \circ \phi_2$, where $\circ$ is either $\lor$, $\land$, $\supset$ or $\equiv$, then $\phi[\sigma] \overset{\text{def}}{=} \phi_1[\sigma] \circ \phi_2[\sigma]$.

Suppose that $D = D_{una} \cup D_{S_0} \cup D_{ap} \cup D_{ss} \cup \Sigma$ is a BAT, $\alpha_1, \cdots, \alpha_n$ is a sequence of ground action terms, and $G(s)$ is a formula uniform in $s$ with one free variable $s$. One of the most important reasoning tasks in the situation calculus is the projection problem, that is, to determine whether

$$D \models G(\text{do}([\alpha_1, \cdots, \alpha_n], S_0)).$$

Another basic reasoning task is the executability problem. Let

$$\text{executable}(\text{do}([\alpha_1, \cdots, \alpha_n], S_0)) \overset{\text{def}}{=} \text{Poss}(\alpha_1, S_0) \land \bigwedge_{i=2}^{n} \text{Poss}(\alpha_i, \text{do}([\alpha_1, \cdots, \alpha_{i-1}], S_0)).$$
Then, the executability problem is to determine whether 

\[ D \models executable(do([\alpha_1, \ldots, \alpha_n], S_0)). \]

Planning and high-level program execution are two important settings where the executability and projection problems arise naturally.

### 2.1.3 The Regression Operator

*Regression* is a central computational mechanism that forms the basis for many planning procedures (Waldinger [164]) and for automated reasoning in the situation calculus (Pednault [132], Pirri and Reiter [134]). Roughly speaking, the regression of a formula \( \phi \) through an action \( a \) is a formula \( \phi' \) that holds prior to \( a \) being performed iff \( \phi \) holds after \( a \). Successor state axioms support regression in a natural way. In [141], Reiter introduces a notation \( \mathcal{R} \) as regression operator, and defines the regression of a *regressable* formula \( W \) of \( \mathcal{L}_{ac} \).

We first review the definition of a regressable formula as follows.

**Definition 2** ([141] Definition 4.5.1) A formula \( W \) of \( \mathcal{L}_{ac} \) is *regressable* iff

1. Every term of sort *Situation* mentioned by \( W \) has the syntactic form 
   \[ do([\alpha_1, \ldots, \alpha_n], S_0) \]  
   for some \( n \geq 0 \), and for terms \( \alpha_1, \ldots, \alpha_n \) of sort *Action*.

2. For every atom of the form \( Poss(\alpha, \sigma) \) mentioned by \( W \), \( \alpha \) has the syntactic form 
   \[ A(t_1, \ldots, t_n) \]  
   for some \( n \)-ary function symbol \( A \) of \( \mathcal{L}_{sc} \).

3. \( W \) does not quantify over situations.

4. \( W \) does not mention the predicate symbol \( \prec \), nor does it mention any equation atom \( \sigma = \sigma' \) for terms \( \sigma, \sigma' \) of sort *Situation*.

\( \square \)
Then, given a BAT $D$, the regression operator $R$ is defined recursively on any regressable formula $W$.

**Definition 3 ([141] Definition 4.7.2)**

1. Suppose $W = \text{Poss}(A(\vec{t}), \sigma)$ where $A(\vec{t})$ and $\sigma$ are terms of sort $Action$ and $Situation$ respectively, and we have action precondition axiom of the form

$$\text{Poss}(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s),$$

without loss of generality, assume that all quantifiers (if any) of $\Pi_A(\vec{x}, s)$ have had their quantified variables renamed to be distinct from the free variables (if any) of $\text{Poss}(A(\vec{t}), \sigma)$, then

$$R[W] = R[\Pi_A(\vec{t}, \sigma)].$$

2. Suppose $W$ is a regressable atom, but not a $\text{Poss}$ atom. There are three possibilities:

(a) $S_0$ is the only term of sort $Situation$ (if any) mentioned by $W$, then

$$R[W] = W.$$

(b) Suppose that $W$ mentions a term of the form $g(\vec{t}, do(\alpha', \sigma'))$ for some functional fluent $g$, and $\alpha'$ and $\sigma'$ are of sort $Action$ and $Situation$ respectively. $g(\vec{t}, do(\alpha', \sigma'))$ mentions a prime functional fluent term $f(\vec{r}, do(\alpha, \sigma))$ where $\alpha$ is a action term and $\sigma$ a situation term uniform in $S_0$. Suppose $f$’s successor state axiom in $D_{ss}$ is

$$f(\vec{x}, do(a, s)) = y \equiv \phi_f(\vec{x}, y, a, s).$$

Without loss of generality, assume that all quantifiers (if any) of $\phi_f(\vec{x}, y, a, s)$ have had their quantified variables renamed to be distinct from the free variables (if any) of $f(\vec{r}, do(\alpha, \sigma))$. Then

$$R[W] = R[(\exists y).\phi_f(\vec{r}, y, \alpha, \sigma) \land W|_{f(\vec{r}, do(\alpha, \sigma))}].$$
Here \( y \) is a variable not occurring free in \( W, \vec{r}, \alpha \) or \( \sigma \).

\textbf{(c)} \( W \) is a relational fluent atom of the form \( F(\vec{t}, \text{do}(\alpha, \sigma)) \) where \( \alpha \) and \( \sigma \) are of sort \textit{Action} and \textit{Situation} respectively. Let \( F \)'s successor state axiom in \( D_{ss} \) be

\[
F(\vec{x}, \text{do}(a, s)) \equiv \Phi_F(\vec{x}, a, s).
\]

Without loss of generality, assume that all quantifiers (if any) of \( \Phi_F(\vec{x}, a, s) \) have had their quantified variables renamed to be distinct from the free variables (if any) of \( F(\vec{t}, \text{do}(\alpha, \sigma)) \). Then

\[
\mathcal{R}[W] = \mathcal{R}[\Phi_F(\vec{t}, \alpha, \sigma)].
\]

3. For non-atomic formulas, regression is defined inductively as follows:

\[
\mathcal{R}[\neg W] = \neg \mathcal{R}[W],
\]
\[
\mathcal{R}[W_1 \land W_2] = \mathcal{R}[W_1] \land \mathcal{R}[W_2],
\]
\[
\mathcal{R}[(\exists x)W] = (\exists x)\mathcal{R}[W].
\]

\( \square \)

The regression theorem proved in [134] shows that one can reduce the evaluation of a regressable sentence \( W \) to a first-order logic theorem proving task in the initial theory together with unique names axioms for actions using the regression operator: for any BAT \( \mathcal{D} \),

\[
\mathcal{D} \models W \iff \mathcal{D}_{s0} \cup \mathcal{D}_{una} \models \mathcal{R}[W].
\]

Hence, we can solve the executability and projection problems using the regression operator in the situation calculus.
2.2 Description Logics and Two-Variable First-Order Logics

Description logics [4] are a family of knowledge representation languages that represent the concept definitions of an application domain in a formally structured way. They are considered the most important knowledge representation formalism unifying and giving a logical basis to the well-known traditions of frame-based systems [123], semantic networks [155] and KL-ONE-like languages [20]. Nowadays, description logics provide a principal logical foundation of sub-languages of the W3C-endorsed Web Ontology Language (OWL) [82] in the area of Semantic Web. In this section, we review a few popular expressive description logics and related fragments of the first-order logic [18].

2.2.1 Description Logics

We start with the language of logic $\mathcal{ALCQIO}$. Let $N_C = \{AC_1, AC_2, \ldots\}$ be a non-empty set of atomic concepts and $N_R = \{R_1, R_2, \ldots\}$ be a non-empty set of atomic roles. In $\mathcal{ALCQIO}$, nominals are allowed. Nominals are singleton concepts obtained by picking one of the object names. An $\mathcal{ALCQIO}$ role is either some $R \in N_R$ or an inverse role $R^-$ for $R \in N_R$. In addition, $(R^-)^- = R$ itself. The set of $\mathcal{ALCQIO}$ concepts is the minimal set built inductively from $N_C$ and $\mathcal{ALCQIO}$ roles using the following rules:

1. all $AC \in N_C$ are concepts,

2. nominals are concepts,

3. if $C$, $C_1$, and $C_2$ are $\mathcal{ALCQIO}$ concepts, $R$ is a role and $n \in \mathbb{N}$, then $\neg C$, $C_1 \sqcap C_2$, $\geq nR.C$, and $\forall R.C.C$ are also $\mathcal{ALCQIO}$ concepts.

Concepts that are not atomic are called complex. A literal concept is a possibly negated concept name. The abbreviations for complex concepts such as $C_1 \sqcup C_2$, $\leq nR.C$, $\exists R.C$, $\forall R.C$ (and other complex concepts) can be easily defined. For example,
\[\exists R.C \overset{\text{def}}{=} \geq 1 R.C\]
\[\top \overset{\text{def}}{=} AC \cup \neg AC \text{ for some } AC \in N_C\]
\[\forall R.C \overset{\text{def}}{=} \neg \exists R.\neg C\]
\[\bot \overset{\text{def}}{=} AC \cap \neg AC \text{ for some } AC \in N_C\]
\[C_1 \sqcup C_2 \overset{\text{def}}{=} \neg (\neg C_1 \cap \neg C_2)\]
\[\leq n R.C \overset{\text{def}}{=} \neg (\geq (n + 1) R.C)\]
\[\exists^n R.C \overset{\text{def}}{=} (\leq n R.C) \cap (\geq n R.C)\]

The semantics of concepts are given denotationally, using the notion of an interpretation \(I = \langle \Delta^I, (\cdot)^I \rangle\), where \(\Delta^I\) is a domain (non-empty universe) of objects, and \((\cdot)^I\) maps from atomic concept names to subsets of the domain (i.e., \(AC^I \subseteq \Delta^I\) for all \(AC \in N_C\)), and atomic role names to sets of pairs over the domain (i.e., \(R^I \subseteq \Delta^I \times \Delta^I\) for all \(R \in N_R\)). Moreover, the interpretation function \((\cdot)^I\) is extended recursively to complex concepts in Table 2.1 and role constructors in Table 2.2. We assume that the interpretation of nominals has to respect the unique name assumption.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td>(\top)</td>
<td>(\Delta^I)</td>
</tr>
<tr>
<td>Bottom</td>
<td>(\bot)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>Nominal</td>
<td>{b}</td>
<td>(b^I \in \Delta^I)</td>
</tr>
<tr>
<td>Negation</td>
<td>(\neg C)</td>
<td>(\Delta^I \setminus C^I)</td>
</tr>
<tr>
<td>Intersection</td>
<td>(C_1 \cap C_2)</td>
<td>(C_1^I \cap C_2^I)</td>
</tr>
<tr>
<td>Union</td>
<td>(C_1 \cup C_2)</td>
<td>(C_1^I \cup C_2^I)</td>
</tr>
<tr>
<td>Qualified</td>
<td>(\geq n R.C)</td>
<td>({\delta \in \Delta^I \mid \exists \delta_1. (\delta, \delta_1) \in R^I \cap \delta_1 \in C^I}\geq n}</td>
</tr>
<tr>
<td>at-least restriction</td>
<td>(\leq n R.C)</td>
<td>({\delta \in \Delta^I \mid \forall \delta_1. (\delta, \delta_1) \in R^I \supset \delta_1 \in C^I}\leq n}</td>
</tr>
<tr>
<td>Existential</td>
<td>(\exists R.C)</td>
<td>({\delta \in \Delta^I \mid \exists \delta_1. (\delta, \delta_1) \in R^I \cap \delta_1 \in C^I}}</td>
</tr>
<tr>
<td>quantification</td>
<td>(\forall R.C)</td>
<td>({\delta \in \Delta^I \mid \forall \delta_1. (\delta, \delta_1) \in R^I \supset \delta_1 \in C^I}}</td>
</tr>
</tbody>
</table>

Table 2.1: The semantics of some common description logic concept constructors.
A TBox $\mathcal{T}$ is a finite set of equality axioms $C_1 \equiv C_2$.\footnote{Sometimes, general inclusion axioms of the form $C_1 \sqsubseteq C_2$ are also allowed, where $C_1, C_2$ are complex concepts.} An equality with an atomic concept on the left-hand side is a concept definition. In the sequel, we always consider TBox $\mathcal{T}$ that is a terminology: a finite set of concept definition formulas with unique left-hand sides, i.e., no atomic concept occurs more than once as a left-hand side. We say that a defined concept name $C_1$ directly uses a concept name $C_2$ with respect to a TBox $\mathcal{T}$ if $C_1$ is defined by a concept definition axiom in $\mathcal{T}$ with $C_2$ occurring on the right-hand side of the axiom. Let uses be the transitive closure of directly uses, and a TBox $\mathcal{T}$ is acyclic if no concept name uses itself with respect to $\mathcal{T}$. An ABox $\mathcal{A}$ is a finite set of assertions of the forms $C(b)$ and $R(b,b_1)$, where $b$ and $b_1$ are some object names, $C$ is a concept, and $R$ is a role. An RBox $\mathcal{H}_R$, called a role hierarchy, is a finite set of role inclusion axioms of the form $R_1 \sqsubseteq R_2$, where $R_1$ ($R_2$, respectively) is either an atomic role or its inverse. A knowledge base in description logics is a tuple $(\mathcal{T}, \mathcal{A})$ (or, a triple $(\mathcal{H}_R, \mathcal{T}, \mathcal{A})$ if role inclusion axioms are allowed). The semantics of terminological and assertional axioms are provided in Table 2.3 below.

The logic $\mathcal{ALCQI}$ ($\mathcal{ALCQO}$, or $\mathcal{ALCIO}$, respectively) can be obtained by disallowing nominals (by disallowing inverse roles, or by disallowing the qualified number restrictions, respectively) in $\mathcal{ALCQIO}$. The logic $\mathcal{ALCO}$ can be obtained by disallowing the qualified number restrictions in $\mathcal{ALCQO}$, that is, comparing to $\mathcal{ALCQO}$, concepts in $\mathcal{ALCO}$ cannot be of the form $\geq nR.C$ or of the form $\leq nR.C$. If we further disallow nominals in $\mathcal{ALCO}$, we obtain a sublanguage $\mathcal{ALC}$. For a DL language $\mathcal{L}$ (e.g., $\mathcal{ALC}$, $\mathcal{ALCO}$ or $\mathcal{ALCQO}$, respectively), we denote the language obtained by adding the universal role $U$ to $\mathcal{L}$ as $\mathcal{L}(U)$ (e.g., $\mathcal{ALC}(U)$, $\mathcal{ALCO}(U)$ or $\mathcal{ALCQO}(U)$, respectively). Note that the semantics of the universal role $U$ is given in Table 2.2. The logic $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$ is obtained from $\mathcal{ALCQIO}$ by introducing concept identity $id(C)$ (relating each individual in $C$ with itself), and allowing complex role expressions: if $R_1, R_2$ are $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$ roles and $C$
is a concept, then \( R_1 \sqcup R_2, R_1 \sqcap R_2, \neg R_1, R_1^\top \) and \( R_1|_C \) are \( \mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id) \) roles too. These complex roles can be used in constructing complex concepts. The semantics of complex roles are given in Table 2.2. Subsequently, we call a role \( R \) *primitive* if it is either \( R \in N_R \) or it is an *inverse role* \( R^- \) for \( R \in N_R \). The logic \( \mathcal{ALCHQIO}(\sqcup, \sqcap, \neg, |, id) \) allows RBox axioms based on the language of \( \mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id) \). Moreover, notice that the universal role can be implicitly constructed in \( \mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id) \): \( U \) can be replaced using \( R \sqcup \neg R \) for any \( R \in N_R \).

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal role</td>
<td>( U )</td>
<td>( \Delta I \times \Delta I )</td>
</tr>
<tr>
<td>Inverse</td>
<td>( R^- )</td>
<td>( { (\delta_1, \delta) \in \Delta I \times \Delta I \mid (\delta, \delta_1) \in R^I } )</td>
</tr>
<tr>
<td>Complement</td>
<td>( \neg R )</td>
<td>( \Delta I \times \Delta I \setminus R^I )</td>
</tr>
<tr>
<td>Intersection</td>
<td>( R_1 \sqcap R_2 )</td>
<td>( R_1^I \cap R_2^I )</td>
</tr>
<tr>
<td>Union</td>
<td>( R_1 \sqcup R_2 )</td>
<td>( R_1^I \cap R_2^I )</td>
</tr>
<tr>
<td>Role restriction</td>
<td>( R</td>
<td>_C )</td>
</tr>
<tr>
<td>Identity</td>
<td>( id(C) )</td>
<td>( { (\delta, \delta) \in \Delta I \times \Delta I \mid \delta \in C^I } )</td>
</tr>
<tr>
<td>Reflexive-transitive closure</td>
<td>( R^* )</td>
<td>( \bigcup_{n \geq 0} (R^I)^n )</td>
</tr>
<tr>
<td>Composition</td>
<td>( R_1 \circ R_2 )</td>
<td>( { (\delta_1, \delta_2) \in \Delta I \times \Delta I \mid \exists \delta \in \Delta I, (\delta_1, \delta) \in R_1^I \land (\delta, \delta_2) \in R_2^I } )</td>
</tr>
</tbody>
</table>

Table 2.2: The semantics of some common description logic role constructors.

There are different reasoning tasks in description logics, such as *concept satisfiability problem* and *ABox consistency problem*, etc. The concept satisfiability problem is to decide given a TBox \( \mathcal{T} \) and a concept \( C \), whether there exists a model \( I \) of \( \mathcal{T} \) such that \( C^I \) is nonempty. Given any description logic knowledge base \( (\mathcal{T}, \mathcal{A}) \), the ABox consistency problem is to check whether there is an interpretation that is a model for both \( \mathcal{T} \) and \( \mathcal{A} \). The complexity of solving the concept satisfiability problem or the ABox consistency problem has been studied for different versions of description logics [174]. For
<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept inclusion</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>$C^I_1 \subseteq C^I_2$</td>
</tr>
<tr>
<td>Role inclusion</td>
<td>$R_1 \sqsubseteq R_2$</td>
<td>$R^I_1 \subseteq R^I_2$</td>
</tr>
<tr>
<td>Concept equality</td>
<td>$C_1 \equiv C_2$</td>
<td>$C^I_1 = C^I_2$</td>
</tr>
<tr>
<td>Role equality</td>
<td>$R_1 \equiv R_2$</td>
<td>$R^I_1 = R^I_2$</td>
</tr>
<tr>
<td>Concept assertion</td>
<td>$C(b)$</td>
<td>$b^I \in C^I$</td>
</tr>
<tr>
<td>Role assertion</td>
<td>$R(b_1, b_2)$</td>
<td>$(b^I_1, b^I_2) \in R^I$</td>
</tr>
</tbody>
</table>

Table 2.3: The semantics of terminological and assertional axioms.

example, for any DL knowledge base with acyclic TBox only, it has been shown that the complexities of solving these two problems are \textsc{PSPACE}-complete in $\mathcal{ALC}$ [150, 111, 6], in $\mathcal{ALCO}$ [150, 159] or in $\mathcal{ALCOO}$ [150, 84, 9]. For any DL knowledge base (with general inclusion TBox axioms), the complexities of solving these two problems are \textsc{ExpTime}-complete in $\mathcal{ALC}$ [145, 34, 6], in $\mathcal{ALCO}$ [145, 83, 174], in $\mathcal{ALCOO}$ [143, 34, 83], in $\mathcal{ALC}(U)$ [156, 77, 112], in $\mathcal{ALCO}(U)$ [143, 2, 80] or in $\mathcal{SHIQ}$ [83]. Solving these two problems for any DL knowledge base in $\mathcal{ALCO}(U)$ or in $\mathcal{ALCOO}(U)$ are \textsc{ExpTime}-complete because of the following reasons. First, $\mathcal{ALC}$ is a subset of $\mathcal{ALCO}(U)$ and $\mathcal{ALCOO}(U)$, which means that solving these two problems in $\mathcal{ALCO}(U)$ or in $\mathcal{ALCOO}(U)$ are \textsc{ExpTime}-hard. Second, $\mathcal{ALCO}(U)$ can be (polynomially) reduced to $\mathcal{ALCO}$ using the (standard) spy-point technique [14, 2], and one can implicitly represent the universal role in $\mathcal{SHIQ}$ [84, 159] so that $\mathcal{ALCOO}(U)$ can be (polynomially) reduced to a subset of $\mathcal{SHIQ}$. This means that the upper bound of solving these two problems in $\mathcal{ALCO}(U)$ or in $\mathcal{ALCOO}(U)$ are still \textsc{ExpTime}.

### 2.2.2 $C^2$ and Its Relationship to Description Logics

The \textit{two-variable first-order logic} $\text{FO}^2$ is a well-known fragment of standard first-order logic, whose formulas can be built with the help of predicate symbols (including equality).
and constant symbols (but without general function symbols) using no more than two variable symbols $x$ and $y$ (free or bound). Note that each variable can be reused arbitrarily often. The *two-variable first-order logic with counting* $C^2$ extends $\text{FO}^2$ by allowing first-order counting quantifiers $\exists \geq m$ and $\exists \leq m$ for all $m \geq 1$ [60, 131, 86, 136]. $\exists^m x. \phi(x)$ ($\exists^m x. \phi(x)$, respectively) means that “there exist at least (at most, respectively) $m$ distinct elements for $x$ that satisfy property $\phi(x)$”. The semantics of $\text{FO}^2$ ($C^2$, respectively) is the same as the semantics of first-order logic (see Def. 4).

**Definition 4** For any formula $\phi$ in $C^2$ (or more generally, in first-order logic with counting quantifiers), a structure $\mathcal{M}$ is a *model* of $\phi$, written as $\mathcal{M} \models \phi$ iff for every interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$ with variable assignment $I$, $\mathbb{I} \models \phi$. In particular, when $\phi$ is a sentence, this does not depend on any variable assignment and $\mathbb{I} = \mathcal{M}$. Moreover, we say that an interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$ satisfies $\phi$, written as $\mathbb{I} \models \phi$, if the following conditions (1)-(9) hold:

1. $\mathbb{I} \models P(t^1_1, \ldots, t^n_1)$ iff $(t^1_1, \ldots, t^n_1) \in P^1$.

2. $\mathbb{I} \models \neg \phi$ iff $\mathbb{I} \not\models \phi$ does not hold.

3. $\mathbb{I} \models \phi_1 \land \phi_2$ iff $\mathbb{I} \models \phi_1$ and $\mathbb{I} \models \phi_2$.

4. $\mathbb{I} \models \phi_1 \lor \phi_2$ iff $\mathbb{I} \models \phi_1$ or $\mathbb{I} \models \phi_2$.

5. $\mathbb{I} \models \phi_1 \supset \phi_2$ iff $\mathbb{I} \models \neg \phi_1 \lor \phi_2$.

6. $\mathbb{I} \models \forall x. \phi$ iff for every $d \in \Delta$, $\mathbb{I} \models \phi[x/d]$, where $\phi[x/d]$ represent the formula obtained by substituting $x$ with $d$.

7. $\mathbb{I} \models \exists x. \phi$ iff there is some $d \in \Delta$ s.t. $\mathbb{I} \models \phi[x/d]$.

8. For any $m \geq 1$, $\mathbb{I} \models \exists^m x. \phi$ iff there are at most $m$ distinct elements $d_i \in \Delta$ ($1 \leq i \leq m$) s.t. $\mathbb{I} \models \phi[x/d_i]$. 
Chapter 2. Background

(9) For any \( m \geq 1 \), \( \mathbb{I} \models \exists^m x.\phi \) iff there are at least \( m \) distinct elements \( d_i \in \Delta \) \((1 \leq i \leq m)\) s.t. \( \mathbb{I} \models \phi[x/d_i] \).

The definition of a structure and a variable assignment in the above definition is the same as in first-order logic, hence details are omitted here.

It is well-known that modal logic is a “big brother” of description logics. In particular, it is proved that the description logic \( ALC \) is a notational variant of a basic multi-modal logic \( K \) \([145]\). For this reason, it is important to note that the \textit{the standard translation} from a basic modal logic to first-order logic is proposed in \([161]\). Later, it was realized that a basic modal logic can be translated to FO\(^2\) \([51, 162]\). The standard translation and other results in modal logic are extensively discussed in \([15]\) and in \([129]\).

Now we consider some relationships between \( C^2 \) and description logics. In \([18]\), an expressive description logic \( B \) is defined,\(^5\) and it is shown that \( C^2 \) and the language \( B \) are \textit{equally expressive}, that is, \( C^2 \) is \textit{as expressive as} \( B \) and vice versa. Generally speaking, a language \( L_2 \) is \textit{as expressive as} language \( L_1 \), if there is a translation function \( transl \) from \( L_1 \) to \( L_2 \) such that for each sentence \( l \) in \( L_1 \), there is a corresponding sentence \( transl(l) \) in \( L_2 \) so that it has the same interpretations as \( l \). Note that there is a 1-1 correspondence between a DL interpretation \( I = \langle \Delta^I, (\cdot)^I \rangle \) and a \( C^2 \) structure \( M \): setting the \( C^2 \) structure domain to be \( \Delta^I \) and setting the \( C^2 \) interpretation of \( AC \) (\( R \), respectively) in \( M \) to be \( AC^I \) for each atomic concept name \( AC \) (for each atomic role name \( R \), respectively) and vice versa. Hence we do not distinguish between both kinds of interpretations (or structures from the first-order logic semantics point of view).

It is shown in \([18]\) that the translation in both directions between \( C^2 \) and \( B \) leads to no more than a linear increase in the size of the translated formula. His translation is

\(^5\)In \([18]\), the language \( B \) is denoted as \( DL – \{\text{trans}, \text{compose}\} \), in which \text{trans} represents the role constructor \textit{reflexive-transitive closure} and \text{compose} represents the role constructor \textit{composition}. We change it to notation \( B \) in order to avoid confusion. Besides, the syntax and semantics of reflexive-transitive closure and composition can be found in Table 2.2.
similar to the standard translation earlier proposed for modal logics. However, note that $B$ includes a role constructor $C_1 \times C_2$ for any two concepts $C_1$ and $C_2$, whose semantics is defined as $C_1^T \times C_2^T$ for any interpretation $I$. This role constructor is not considered as one of the common role constructors for constructing DL languages introduced in [4]. Using the result of the relationship between $B$ and $C^2$ shown in [18] as well as the same approach to proving the result, we prove the following theorem (i.e., Th. 1), which illustrates a direct connection between $C^2$ and the DL language $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$ constructed using only those common concept and role constructors which can be found in [4]. The detailed proof is provided in Appendix A.1.

**Theorem 1** The description logic $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$ and $C^2$ are equally expressive (i.e., each sentence in language $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$ can be translated to a sentence in $C^2$, and vice versa). In addition, the translation in each direction leads to no more than a linear increase in the size of the translated formula.

Notice that $\mathcal{ALCHQIO}(\sqcup, \sqcap, \neg, |, id)$ includes RBox in the knowledge bases, in contrast to $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$ that has no RBox. However, it is obvious that every axiom in RBox still can be translated into a sentence in $C^2$. Hence, $\mathcal{ALCHQIO}(\sqcup, \sqcap, \neg, |, id)$ and $C^2$ are also equally expressive. In [113], another description logic is considered and an alternative translation between that description logic and $\mathcal{FO}^2$ is proposed. It is proved that translation from $\mathcal{FO}^2$-formulae into concepts in the considered DL involves an exponential blow-up in formula size.

This statement has an important consequence. Gradel et al. [61] and Pacholski et al. [131] show that the satisfiability problem for $C^2$ is decidable and recently Pratt-Hartmann [136] proves that this problem is in $\text{NExpTime}$ even when counting quantifiers are coded succinctly. Hence, the satisfiability and/or subsumption problems of concepts with respect to an acyclic or empty TBox in description logic $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)$

---

6 $\mathcal{ALC}$ extended with full Boolean operators on roles, the inverse operator on roles and an identity role.
(\text{ACHQIO}(\sqcup, \sqcap, \neg, |, \text{id}), \text{respectively}) \text{ is also decidable with the same computational complexity. Additional background on description logics and discussion of the connections between description logics with } C^2 \text{ can be found in [4, 18].}

Note that it is not the number of arguments in predicates affects the decidability of the satisfiability problem in } C^2. \text{ What matters is the number of variables that can be used in the formula. In fact, it is known that any first-order formula can be transformed into some equivalent first-order formula containing predicates with no more than two arguments [17]. In particular, Löwenheim [110] showed that any first-order formula } \phi \text{ can be transformed into some equivalent first-order formula } \phi' \text{ containing binary predicates only. Later, in 1931, Herbrand [78] showed that just three binary predicates are enough for such transformation, and Kalmár [88] soon showed that only one binary predicate suffices. The transformation approach proposed by Kalmár can also be found in [89, 90]. The proof of the Church-Herbrand theorem that “the logic of a single dyadic (two-place) predicate is undecidable” can be found in the textbook [16]. Hence, a first-order formula } \phi \text{ is satisfiable iff its equivalent transformation } \phi' \text{ containing only binary predicates is so. Therefore, we can conclude that the satisfiability problem in the class of first-order formulas composed from predicates with no more than two arguments generally is undecidable; otherwise, the decidability problem in the first-order logic would be decidable, which contradicts to the undecidability of the satisfiability problem in first-order logic [27, 160].}

In Chapter 3, when proposing the framework of a modified situation calculus } L_{sc}^{C^2}, \text{ to start with something simple and to make obvious connections to DLs, we restrict the numbers of arguments of action functions and fluents in a dynamical domain. However, we have to keep in mind that the decidability of certain reasoning problems that we will consider is obtained by restricting the number of variables (both free and bound) in the language, not by restricting the number of arguments of predicates.}
2.3 Order-Sorted Logic

Order-sorted logic (OSL) is a logic extending with sorts, which restricts the domain of variables to subsets of the universe (i.e., sorts) [127, 166, 128, 152, 13, 171]. Research shows that the extension of first-order logic with sorts has several advantages over standard first-order logic: It simplifies algebraic specifications [152, 58, 165], reduces search space in automated theorem proving [130, 167, 170], and allows for more compact and more natural formalizations of AI problems [154, 13].

In this section, we review the syntax and semantics of OSL. But first, we introduce the following abbreviations for later convenience.

1. Notation $x : Q$ means that variable $x$ is of sort $Q$ and $V_Q$ is the set of variables of sort $Q$.

2. For any $n$, sort cross-product $Q_1 \times \cdots \times Q_n$ is abbreviated as $\vec{Q}_{1..n}$.

3. For any $n$, term vector $\langle t_1, \ldots, t_n \rangle$ is abbreviated as $\vec{t}_{1..n}$.

4. For any $n$, variable vector $\langle x_1, \ldots, x_n \rangle$ is abbreviated as $\vec{x}_{1..n}$.

5. For any $n$, variable declaration sequence $x_1 : Q_1, \ldots, x_n : Q_n$ is abbreviated as $\vec{x}_{1..n} : \vec{Q}_{1..n}$.

2.3.1 The Syntax of Order-Sorted Logic

A theory in OSL always includes a set of declarations (called sort theory) to describe the hierarchical relationships among sorts and the restrictions on ranges of the arguments of predicates and functions. Formally, a sort theory $\mathbb{T}$ includes a set of term declarations of the form $t : Q$ representing that term $t$ is of sort $Q$, subsort declarations of the form $Q_1 \leq Q_2$ representing that sort $Q_1$ is a (direct) subsort of sort $Q_2$ (i.e., every object of sort $Q_1$ is also of sort $Q_2$), and predicate declarations of the form $P : \vec{Q}_{1..n}$ representing that the $i$-th argument of the $n$-ary predicate $P$ is of sort $Q_i$ for $i = 1..n$. In particular, when
a logic has more than one sort symbol and there are no subsort declarations, it is called
many-sorted logic. A function declaration is a special term declaration where term \( t \) is a
function with distinct variables as arguments: for each \( n \)-ary function \( f \), the abbreviation
of its function declaration is of the form \( f : Q_1 \ldots Q_n \rightarrow Q \), where \( Q_i \) is the sort of the \( i \)-th
argument of \( f \) and \( Q \) is the sort of the value of \( f \). Note that \( c : Q \) is a special function
declaration, representing that constant \( c \) is of sort \( Q \). Arguments of equality “=” can
be of any sort. Below, we consider a finite simple sort theory only, in which there are
finitely many sorts and declarations, the term declarations are all function declarations,
and for each function there is one and only one declaration.

**Example 1** We provide some examples of terminology of OSL. Subsort declaration
\( \text{MovObj} \leq \text{Object} \) represents that any movable object (i.e., of sort \( \text{MovObj} \)) is also an
object (i.e., of sort \( \text{Object} \)). Predicate declaration

\[
\text{InCity} : \text{MovObj} \times \text{City} \times \text{Situation}
\]

restricts the range of each argument of \( \text{InCity} \) to be sort \( \text{MovObj} \), \( \text{City} \) and \( \text{Situation} \),
respectively. Function declaration

\[
\text{load} : \text{Box} \times \text{Truck} \rightarrow \text{Action}
\]

restricts the range of each domain argument of function \( \text{load} \) to be sort \( \text{Box} \) and \( \text{Truck} \),
respectively and the range of its codomain to be sort \( \text{Action} \). Any term \( \text{load}(t_1, t_2, t_3) \)
is said to be of sort \( \text{Action} \). Constant declarations are special function declarations. For
instance, \( \text{Pasadena} : \text{City} \) states that \( \text{Pasadena} \) is a city (i.e., of sort \( \text{City} \)).

For any sort theory \( \mathbb{T} \), subsort relation \( \leq_{\mathbb{T}} \) is a partial ordering defined by the reflexive
and transitive closure of the subsort declarations. Then, following the standard terminol-
ogy of lattice theory, if each pair of sort symbols in \( \mathbb{T} \) has a greatest lower bound (g.l.b.),
we say that the sort hierarchy of \( \mathbb{T} \) is a meet semi-lattice [168].
2.3.2 Well-sortness and the Semantics of Order-Sorted Logic

Given a sort theory $\mathbb{T}$, well-sorted terms and formulas can be defined as follows (see [13]). Moreover, any term or formula that is not well-sorted is called *ill-sorted*.

**Definition 5** A set $T_T$ of well-sorted terms is the least $Q_T$-indexed family of sets $T_{T,Q}$ s.t.:

1. $x \in T_{T,Q}$ for every $x \in V_Q$, i.e., the sort of $x$ is $Q$;
2. $c \in T_{T,Q}$ for every constant declaration $c:Q$ in $T$, i.e., the sort of $c$ is $Q$;
3. $f(t_1..n) \in T_{T,Q}$, if $f:Q_1..n \rightarrow Q$, each $t_i$ is of sort $Q_i'$ and $Q_i' \leq T Q_i$ for $i = 1..n$, i.e., the sort of $f(t_1..n)$ is $Q$.

A well-sorted formula (with respect to $T$) is defined by:

1. $P(t_1..n)$ is a well-sorted formula if $t_i$ is of sort $Q_i'$ for each $i = 1..n$, $P:Q_1..n$ (including equality symbol “=” if any) and $Q_i' \leq T Q_i$ for each $i = 1..n$.
2. If $\phi$ is well-sorted, so is $\neg \phi$.
3. If $\phi_1$ and $\phi_2$ are well-sorted, so are $\phi_1 \land \phi_2$, $\phi_1 \lor \phi_2$, and $\phi_1 \supset \phi_2$.
4. If $x$ is a variable of sort $Q$ and $\phi$ is a well-sorted formula that has free occurrences of $x$, then $\forall x:Q.\phi$ and $\exists x:Q.\phi$.

Note that we follow traditional approaches to sorted reasoning, where sort symbols must not occur as predicates in the formulas [127, 147, 167, 152]. Alternative approaches, called hybrid, allow to mix sort symbols with application specific predicates (see [171, 31, 13]).

The semantics of OSL is defined similar to unsorted logic. Note that the definition of interpretations for well-sorted terms and formulas is the same as in unsorted logic, but the semantics is not defined for ill-sorted terms and formulas.
Definition 6  For any well-sorted formula $\phi$, a $\mathbb{T}$-interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$ of $\phi$ is a tuple for a structure $\mathcal{M}$ and a sort-assignment $I$ from the set of the free variables in $\phi$ to the universe $\Delta$ of $\mathcal{M}$, s.t. it satisfies the following five conditions.

1. For each sort $Q$, $Q^I$ is a subset of the whole universe $\Delta$. In particular, $\bot^I = \emptyset$, and $Q_1^I \subseteq Q_2^I$ for any $Q_1 \leq_T Q_2$.

2. For any predicate declaration $P : Q_{1..n} \rightarrow Q$, $P^I \subseteq Q_1^I \times \cdots \times Q_n^I$ is a relation in $\mathcal{M}$.

3. For any function declaration $f : Q_{1..n} \rightarrow Q$, $f^I : Q_1^I \times \cdots \times Q_n^I \rightarrow Q^I$ is a function in $\mathcal{M}$.

4. $x^I = I(x)$ is in $Q^I$ for any variable $x \in V_Q$, $c^I \in Q^I$ for any constant declaration $c : Q$, and $(f(t_{1..n}))^I \overset{\text{def}}{=} f^I(t_1^I, \ldots, t_n^I)$ for any well-sorted term $f(t_{1..n})$. $I$ is not defined for ill-sorted terms and formulas. Note that since $I$ always maps a variable $x \in V_Q$ to an object in its corresponding set of sort objects $Q^I$. Hence, $I$ is called a sort-assignment below.

5. If $\mathbb{T}$ includes a declaration for equality symbol “$=$”, then $=^I$ must be defined as set $\{(d, d) \mid d \in \Delta\}$, i.e., the equality symbol is interpreted by the identity relation on the whole universe.

For any sort theory $\mathbb{T}$ and a well-sorted formula $\phi$, a structure $\mathcal{M}$ is a $\mathbb{T}$-model of $\phi$, written as $\mathcal{M} \models_{\mathbb{T}} \phi$ iff for every $\mathbb{T}$-interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$, $\mathbb{I} \models_{\mathbb{T}} \phi$. In particular, when $\phi$ is a sentence, this does not depend on any variable assignment and $\mathbb{I} = \mathcal{M}$. Moreover, we say that a $\mathbb{T}$-interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$ satisfies $\phi$, written as $\mathbb{I} \models_{\mathbb{T}} \phi$, if the following conditions (1)-(7) hold:

1. $\mathbb{I} \models_{\mathbb{T}} P(t_{1..n})$ iff $(t_1^I, \ldots, t_n^I) \in P^I$.

2. $\mathbb{I} \models_{\mathbb{T}} \neg \phi$ iff $\mathbb{I} \not\models_{\mathbb{T}} \phi$ does not hold.

3. $\mathbb{I} \models_{\mathbb{T}} \phi_1 \land \phi_2$ iff $\mathbb{I} \models_{\mathbb{T}} \phi_1$ and $\mathbb{I} \models_{\mathbb{T}} \phi_2$. 
Given a sort theory $T$ as the background, a theory $\Phi$ including well-sorted sentences only satisfies a well-sorted sentence $\phi$, written as $\Phi \models_{T}^{os} \phi$, iff every model of $\Phi$ is a model of $\phi$. □

In this document (particularly in Chapter 5), we use $\models_{T}^{os}$ to represent the logical entailment with respect to a sort theory $T$ in order-sorted logic, $\models_{T}^{ms}$ to represent the logical entailment in Reiter’s situation calculus (a many-sorted logic with one standard sort Object), and $\models_{T}^{fo}$ to represent the logical entailment in unsorted predicate logic. Moreover, for any structure $\mathcal{M}$ and any assignment (sort-assignment, respectively) $I$, we use notation $\mathcal{M}, I \models_{T}^{fo} \phi$ ($\mathcal{M}, I \models_{T}^{ms} \phi$ and $\mathcal{M}, I \models_{T}^{os} \phi$, respectively) to denote an interpretation $\langle \mathcal{M}, I \rangle$ satisfies $\phi$ in unsorted logic (Reiter's situation calculus and order-sorted logic, respectively) for the sake of simplicity.

A well-sorted substitution (with respect to $T$) is a substitution $\rho$ s.t. for any variable $x : Q$, $\rho x$ (the result of applying $\rho$ to $x$) is a well-sorted term and its sort is a (non-empty) subsort of $Q$. Given any set $E = \{\langle t_{1,1}, t_{1,2} \rangle, \ldots, \langle t_{n,1}, t_{n,2} \rangle\}$, where each $t_{i,j}$ ($i = 1..n, j = 1..2$) is a well-sorted term, a well-sorted most general unifier (well-sorted mgu) of $E$ is a well-sorted substitution that is an mgu of $E$. It is important that in comparison to mgu in unsorted logic (i.e., predicate logic without sorts), mgu in OSL can include new weakened variables of sorts which are subsorts of the sorts of unified terms. For example, assume that $E = \{\langle x, y \rangle\}$, $x \in V_{Q_{1}}$, $y \in V_{Q_{2}}$ and the g.l.b. of $\{Q_{1}, Q_{2}\}$ is a non-empty sort $Q_{3}$. Then, $\mu = [x/z, y/z]$ ($x$ is substituted by $z$, $y$ is...
substituted by \( z \) for some new variable \( z \in V_{Q_3} \) is a well-sorted mgu of \( E \). Well-sorted mgu neither always exists nor it is unique [168]. However, it is proved that the well-sorted mgu of unifiable sorted terms is unique up to variable renaming when the sort hierarchy of \( T \) is a meet semi-lattice [168].

An order-sorted unification algorithm takes a sort theory \( T \) and a set of tuples of well-sorted terms \( E = \{ \langle t_{1,1}, t_{1,2} \rangle, \ldots, \langle t_{n,1}, t_{n,2} \rangle \} \). It outputs a well-sorted mgu if \( E \) is unifiable with respect to \( T \); otherwise, it outputs a statement that \( E \) is not unifiable. Any order-sorted unification algorithm can be described as the result of a simplification step followed by a solving step, as in the unsorted logic [98]. The simplification step is a succession of decomposition, merging and mutation steps, translating the initial unification problem into an equivalent system composed from fully decomposed equalities of the form \( x = t \), where \( x \) must be a variable. The solving step checks the consistency among fully decomposed equalities in the simplified system. However, different from the unsorted unification problem, solving simple equality \( x = t \), where \( x \) does not appear in \( t \), is no more trivial and requires the knowledge of \( T \) to ensure the well-sortness. Moreover, the simplification process is also more complex than in the unsorted case. For instance, if the function declaration for each function is not unique in \( T \), then the decomposition process of terms becomes non-trivial. In [151, 152], Schmidt-Schaubß shows that syntactic unification in order-sorted logic in general is undecidable for sort theories with (general) term declarations. However, if a sort theory is simple with empty equational theory, whose corresponding sort hierarchy is a meet semi-lattice, finding a unique (well-sorted) mgu between two terms takes the same time as in the unsorted case, i.e., it is linear complexity to the size of terms [152, 87, 171].
Chapter 3

Decidable Reasoning in a Modified Situation Calculus

In this chapter, we consider a modified version of the situation calculus built using a two-variable fragment of the first-order logic extended with counting quantifiers. We mention several additional groups of axioms that can be introduced to capture taxonomic reasoning. We show that the regression operator in this framework can be defined similarly to regression in Reiter’s version of the situation calculus. We also show that the projection and executability problems (the important reasoning tasks in the situation calculus) are decidable in the modified version even if an initial knowledge base is incomplete. We discuss the complexity of solving the projection problem in this modified language. Furthermore, we define description logic based sub-languages of our modified situation calculus. They are based on the description logics $\mathcal{ALCO}(U)$ (or $\mathcal{ALCQO}(U)$, respectively). We show that in these sub-languages solving the projection problem via regression has better computational complexity than in the general modified situation calculus. We mention possible applications to a formalization of Semantic Web services in the modified situation calculus and some connections with reasoning about actions based on description logics. Some results of this work have been published in [64, 65, 66]
Chapter 3. Decidable Reasoning in a Modified Situation Calculus

and most results will appear in [70].

3.1 Motivations

The situation calculus from [141], which can be extended with time, concurrency, stochastic actions, etc, serves as a foundation for the Process Specification Language (PSL) that axiomatizes a set of primitives adequate for describing the fundamental concepts of manufacturing processes (PSL has been accepted as an international standard) [63, 62]. It is also used to provide a well-defined semantics for Web services and a foundation for a high-level programming language Golog [101, 116, 126, 11]. However, because the situation calculus is formulated in a general predicate logic, reasoning about the effects of sequences of actions is undecidable (unless some restrictions are imposed on the theory that axiomatizes the initial state of the world).

The first motivation of our work is to overcome the difficulty mentioned above. We propose to use a two-variable fragment $\text{FO}^2$ of the first-order logic (or $C^2$ that is $\text{FO}^2$ extended with counting quantifiers [131, 136]) as a foundation for an initial theory in a modified situation calculus. Because the satisfiability problem in $\text{FO}^2$ (or in $C^2$) is known to be decidable (it is in $\text{NExpTime}$) [60, 131, 136], we demonstrate that by reducing reasoning about effects of actions to reasoning in this fragment, one can always guarantee decidability no matter what the syntactic form of the theory representing the initial state of the world is. Note an important caveat. We are not going to design a decidable logic for reasoning about actions (see work in this direction reviewed in Section 3.6) by imposing strong restrictions on the language such as allowing only action constants and disallowing more complex action terms. Instead, we consider a fragment of the situation calculus where only particular reasoning problems become decidable, but these problems are exactly those that can be important in applications. Consequently, it should not be a surprise for the readers to see that even if an initial theory is an $\text{FO}^2$
(or $C^2$) theory, that is formulated using object variables $x$ and $y$, we include additional variables ($a$, for actions, and $s$, for situations), action terms and situation terms common in the situation calculus. As we show in this chapter, the reasoning problems that we care about can still be reduced to the theorem proving task in $C^2$ (or in fragments of $C^2$).

The second motivation of our work comes from description logics. Description Logics (DLs) [4] 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 the Web Ontology Language (OWL) [82] in the area of the Semantic Web [5]. Many expressive DLs can be translated to FO$^2$ (or to $C^2$) and offer considerable expressive power going far beyond propositional logic, while ensuring that reasoning is decidable [18]. DLs have been mostly used to describe static knowledge bases. However, several research groups consider formalization of Web service actions using DLs or extensions of DLs. Following the key observation that reasoning about complex actions can be carried out in a fragment of the propositional situation calculus, an epistemic extension of DLs was given to provide a framework for the representation of dynamical systems [35]. However, the representation and reasoning about actions in this framework are strictly propositional, which reduces the representation power of this framework. In [8], another approach was proposed for integrating description logics and action formalisms. They take the well-known description logic $\text{ALCQIO}$ (and its sub-languages) as a foundation and show that the complexity of executability and projection problems (two basic reasoning problems for possibly sequentially composed actions) coincides with the complexity of standard DL reasoning. However, actions (services) are represented in their paper meta-theoretically, not as first-order terms. This may cause certain limitations on the expressiveness of the effects of an action, since it is not possible to quantify over arguments of an action or to describe actions that have global effects. It can also potentially lead to some complications when specifications of other reasoning
tasks are considered because it is not possible to quantify over actions in their framework. Other related work is reviewed in Section 3.6. Contrastingly, we take a different approach and represent actions as first-order terms. We achieve integration of taxonomic reasoning and reasoning about actions by restricting the syntax of the situation calculus and by introducing additional axioms to represent a taxonomy.

Besides Web services, other problems such as database update and ontology update can be considered as dynamical systems with actions, and it is possible to apply our framework to certain reasoning problems in these two areas. There is a wide variety of proposals for specifying database update transactions. In particular, Reiter in [140] formalized the evolution of a database under the effect of an arbitrary sequence of update transactions by using the situation calculus. Based on Reiter’s proposal, we may consider applying our framework to some feasible databases and make the reasoning of the projection problems and the executability problems of sequences of database update transactions more efficient. Recently, ontologies are considered as an ideal tool to describe conceptual domain interests in the areas of Data Integration [100] and the Semantic Web [76]. They can be used to describe the semantics of information and hidden knowledge. There are many different ways to formally represent ontologies, such as Knowledge Interchange Format (KIF) [55], F-Logic [96], and Description Logic based languages (e.g., DL-Lite, DAML+OIL, OWL, ...) [4, 23, 82, 81], etc. The problem of ontology update and erasure considers adding and deleting information in order to reflect a change in the domain of interest the ontology is supposed to represent [38]. In particular, the instance-level change (a.k.a. extensional update) of an ontology is to add or to remove knowledge related to instances. For ontologies represented in DLs, instance-level update and erasure correspond to changes to ABoxes [38]. It is easy to see that instance-level changes to an ontology can be considered as actions in a dynamical system. Consequently, our work can be applied to some types of ontologies to provide a different way of reasoning about instance-level ontology update and erasure.
The structure of the rest of this chapter is as follows. In Section 3.2, we discuss details of our proposal: the language $\mathcal{L}_{sc}^{C^2}$ of our modified situation calculus. In Section 3.3.1, we consider an extension of regression (the main reasoning mechanism in the situation calculus) and investigate the computational complexity via regression in Section 3.3.2. In Section 3.3.4, we consider a fragment of $C^2$ that corresponds to a DL with better complexity properties than $C^2$. Then we define a new situation calculus based on this fragment, which can be considered as a sub-language of $\mathcal{L}_{sc}^{C^2}$. In Section 3.3.3, we consider an example that illustrates potential applications to Semantic Web Services. In Section 3.4, we give exact complexity result of solving the projection problem in $\mathcal{L}_{sc}^{C^2}$ without regression. We give more examples for the practical expressiveness and limitations of $\mathcal{L}_{sc}^{C^2}$, and propose an idea of a practical extension of $\mathcal{L}_{sc}^{C^2}$ for future work in Section 3.5. Finally, in Section 3.6, we discuss other related work and future research directions.

3.2 Modeling Dynamical Systems in a Modified Situation Calculus

In this section, we consider dynamical systems formulated in a modification of the language of the situation calculus so that it can be considered as an extension to $C^2$ (with an additional situation argument).

The key idea is to consider a syntactic modification of the situation calculus such that the executability and projection problems are guaranteed to be decidable as a consequence of the decidability of the satisfiability problem in $C^2$. We will denote this language $\mathcal{L}_{sc}^{C^2}$.

Firstly, the three sorts in $\mathcal{L}_{sc}^{C^2}$ (i.e., Action, Situation and Object) are the same as those in $\mathcal{L}_{sc}$, except that they obey the following restrictions:

\[\text{The reason that we call it a “modified” situation calculus rather than a “restricted” situation calculus is that we extend the situation calculus with other features, such as adding acyclic TBox axioms to BATs.}\]
• (1) all terms of sort \textit{Object} are variables \((x\text{ and } y)\) or constants (i.e., object functional symbols are \textit{not} allowed);

• (2) all action functions include no more than two arguments, and each argument of any term of sort \textit{Action} is either a constant or a variable \((x\text{ or } y)\) of sort \textit{Object};

• (3) variable \(s\) of sort \textit{Situation} and/or variable \(a\) of sort \textit{Action} are the only additional variables allowed in \(D\) in addition to variables \(x, y\).

Secondly, any fluent considered in \(L_{sc}^2\) is a predicate with either two or three arguments (including the one of sort \textit{Situation}). We call fluents with two arguments \textit{dynamic concepts}, and call fluents with three arguments \textit{dynamic roles}. Intuitively, each dynamic concept in \(L_{sc}^2\), say \(F(x, s)\) with variables \(x\) and \(s\) only, can be considered as a changeable concept \(F\) in a dynamical system specified in \(L_{sc}^2\); the truth value of \(F(x, s)\) could vary from one situation to another. Similarly, each dynamic role in \(L_{sc}^2\), say \(F(x, y, s)\) with variables \(x, y\) and \(s\), can be considered as a changeable role \(R\) in a dynamical system specified in \(L_{sc}^2\); the truth value of \(F(x, y, s)\) could vary from one situation to another. In \(L_{sc}^2\), \textit{(static) concepts} (i.e., monadic predicates with no situation argument) and \textit{(static) roles} (i.e., dyadic predicates with no situation argument), if any, are considered as unchangeable taxonomic properties and unchangeable classes of an application domain. Moreover, each concept (static or dynamic) can be either \textit{primitive} or \textit{defined}.

For each primitive dynamic concept, an SSA must be provided in the basic action theory for a given domain. Because defined dynamic concepts are expressed in terms of primitive concepts by axioms in an acyclic TBox, SSAs for them are not provided. In addition, SSAs are provided for dynamic roles.

Thirdly, apart from the standard first-order logical symbols \(\land, \lor, \exists\), with the usual definition of a full set of connectives and quantifiers, \(L_{sc}^2\) also includes counting quantifiers \(\exists \geq m\) and \(\exists \leq m\) for all \(m \geq 1\). Equality \(=\) is allowed in \(L_{sc}^2\) too.

Although we mentioned in Section 2.2.2 that the number of arguments in a predicate
does not affect the decidability of the satisfiability problem in $C^2$, to simplify matters, currently we make the restriction on the number of arguments for action functions and fluents.

The dynamical systems we are dealing with here satisfy the open world assumption (OWA): what is not stated explicitly in an initial theory $D_{S_0}$ is unknown rather than false. In this chapter, the dynamical systems we are interested in can be formalized as a modified BAT $D$ in $L_{sc}^{C^2}$ (also called $L_{sc}^{C^2}$-restricted BAT below) using the following seven groups of axioms in $L_{sc}^{C^2}$: $D = \Sigma \cup D_{ap} \cup D_{ss} \cup D_T \cup D_R \cup D_{una} \cup D_{S_0}$. Five of them ($\Sigma, D_{ap}, D_{ss}, D_{una}, D_{S_0}$) are similar to those groups in a BAT in $L_{sc}$, and the other two ($D_T, D_R$) are introduced to axiomatize description logic related facts and properties (see below). However, because $L_{sc}^{C^2}$ allows at most two object variables, all axioms must conform to the following additional requirements.\(^2\)

- **Action Precondition Axioms $D_{ap}$**: For each action function $A(\vec{x})$ in $L_{sc}^{C^2}$, where $\vec{x}$ can be either empty, $x$, or $\langle x, y \rangle$ if $A$ is a 0-ary, 1-ary, or 2-ary action function symbol respectively, there is an axiom of the form

$$\text{Poss}(A(\vec{x}), s) \equiv \Pi_A(\vec{x})[s],$$

(3.1)

where $\Pi_A(\vec{x})$ is a $C^2$ formula with free variables at most among $\vec{x}$ and $\Pi_A(\vec{x})[s]$ represents the result of situation-suppressed formula $\Pi_A(\vec{x})$ with situation term $s$ restored. This set of axioms characterizes the preconditions of all actions.

- **Successor State Axioms $D_{ss}$**: Let variable vector $\vec{x}$ be either $x$ or $\langle x, y \rangle$. An SSA is specified for each dynamic primitive concept that is not defined in TBox (see below) and for each dynamic role. According to the approach of constructing SSAs provided in [141], without loss of generality, we can assume that the SSA of

\(^2\)Subsequently, we write axioms with action and situation variables, and use action and situation terms. However, we will see that they can be eliminated when we solve the projection problem.
Chapter 3. Decidable Reasoning in a Modified Situation Calculus

$F(\bar{x}, s)$ has the form

$$F(\bar{x}, do(a, s)) \equiv \bigvee_{i=1}^{m_+} [\exists x][\exists y][(\exists x)[\exists y]a = A^+_i(\bar{t}_{(i,+)}^i) \land \psi^+_i(\bar{x}_{(i,+)})(s)] \lor$$

$$F(\bar{x}, s) \land \neg(\bigvee_{j=1}^{m_-} [\exists x][\exists y][(\exists x)[\exists y]a = A^-_j(\bar{t}_{(j,-)}^j) \land \psi^-_j(\bar{x}_{(j,-)})(s))]. (3.2)$$

Here, each vector $\bar{t}_{(i,+)}^i$, $i = 1..m_+$, $(\bar{t}_{(j,-)}^j)$, $j = 1..m_-$, respectively represents a vector of object terms appearing in the corresponding action term, which can be either empty, $O$, $\langle O_1, O_2 \rangle$, $x$, $\langle x, x \rangle$, $\langle O, x \rangle$, $\langle x, O \rangle$, $\langle y, y \rangle$, $\langle O, y \rangle$, $\langle y, O \rangle$, $\langle x, y \rangle$ or $\langle y, x \rangle$ for free variables $x$, $y$ and some object constants $O$, $O_1$, $O_2$. Each variable vector $\bar{x}_{(i,+)}^i$ (or $\bar{x}_{(j,-)}^j$, respectively), $i = 1..m_+$, $j = 1..m_-$, represents a vector of free variables appearing in the corresponding context condition, which can be either empty, $x$, $y$, $\langle x, y \rangle$ or $\langle y, x \rangle$. Moreover, $[\exists x]$ or $[\exists y]$ represents that the quantifier included in the square brackets “[ ]” is optional; and each $\psi^+_i(\bar{x}_{(i,+)}^i)$, $i = 1..m_+$, $(\psi^-_i(\bar{x}_{(j,-)}^j))$, $j = 1..m_-$, respectively, is a $C^2$ formula with variables (both free and bound) among $x$ and $y$ at most. Note that when $m_+$ (or $m_-$, respectively) equals to 0, the corresponding disjunctive sub-formula is equivalent to $false$.

- **Acyclic TBox Axioms $D_T$:** Similar to the TBoxes in DLs, we may define new concepts using TBox axioms. Any group of TBox axioms $D_T$ may include two sub-classes: static TBox $D_{T, st}$ and dynamic TBox $D_{T, dyn}$. Every formula in static TBox is a concept definition formula of the form

$$G(x) \equiv \phi_G(x), (3.3)$$

where $G$ is a monadic predicate symbol. Moreover, $\phi_G(x)$ is a $C^2$ formula with a free variable $x$, and has no fluent in it. Every formula in dynamic TBox is a concept definition formula of the form

$$G(x, s) \equiv \phi_G(x)[s], (3.4)$$

where $\phi_G(x)$ is a $C^2$ formula with free variable $x$, and there is at least one fluent in it. All the concepts appeared on the left-hand side of TBox axioms are called
Chapter 3. Decidable Reasoning in a Modified Situation Calculus

defined concepts. We also require that the set of TBox axioms must be acyclic (acyclicity in \( \mathcal{D}_T \) is defined exactly as it is defined for TBox). Note that the defined dynamic concepts are not provided with SSAs. There is no need to provide an SSA for a defined concept because regression can expand TBox definitions instead of an SSA.

• **RBox Axioms** \( \mathcal{D}_R \): Similar to the idea of RBox axioms in DLs, we may also specify a group of axioms, called RBox axioms below, to support a role taxonomy. Each role inclusion axiom is represented as

\[
R_1(x, y)[s] \supset R_2(x, y)[s], \tag{3.5}
\]

where \( R_1 \) and \( R_2 \) are primitive roles (either static or dynamic). If these axioms are included in the BAT \( \mathcal{D} \), then it is assumed that \( \mathcal{D} \) is specified correctly in the sense that the meaning of any RBox axiom included in the theory is correctly compiled into SSAs. This means that an axiomatizer is responsible for writing \( \mathcal{D}_{ss} \) such that axioms from RBox become logical consequences. That is, it should be provable by induction that

\[
(\mathcal{D} - \mathcal{D}_R) \models \forall s. R_1(x, y)[s] \supset R_2(x, y)[s]
\]

for any axiom \( R_1(x, y)[s] \supset R_2(x, y)[s] \) in \( \mathcal{D}_R \). This is the common approach to state constraints, e.g., it was taken in [141]. In some special (but realistic) cases, RBox axioms can be automatically compiled into \( \mathcal{D}_{ss} \). Let us say \( R_2(x, y, s) \) directly depends on \( R_1(x, y, s) \), if \( R_1(x, y, s) \supset R_2(x, y, s) \) belongs to \( RBox \), and say \( R_3(x, y, s) \) depends on \( R_1(x, y, s) \), if \( R_3(x, y, s) \) directly depends on \( R_2(x, y, s) \), and \( R_2(x, y, s) \) depends on \( R_1(x, y, s) \). Then, we can say that RBox is acyclic, if there is no dynamic role \( R(x, y, s) \) that depends on itself. In [117], it is proved that acyclic state constraints can be automatically compiled into SSAs. Because acyclic RBox is just a special case of state constraints considered in McIlraith’s paper, her approach is applicable to acyclic TBox as well. Additional details related to state constraints
in the situation calculus can be found in [104, 103]. Although RBox axioms are not used by the regression operator, they are used for taxonomic reasoning in the initial theory.

Here, the format of the RBox axioms we introduced are the same as those in DLs provided in [4]. In the future, we may consider dealing with different or more general formats of RBox axioms and/or TBox axioms. For example, it will be natural for us to consider acyclic RBox axioms for new role names defined using role constructors, somewhat similar to how new concepts are defined in acyclic TBox.

- **Initial Theory** $\mathcal{D}_{S_0}$: It is a finite or countably infinite set of $C^2$ sentences that are uniform in $S_0$. It specifies the incomplete information about the initial problem state and also describes all the facts that are not changeable over time in the domain of an application. Since the initial theory should include all axioms that relate to the initial situation $S_0$ or facts that are situation independent, the static TBox axioms $\mathcal{D}_{T,st}$ and all RBox axioms in the initial situation $S_0$ (if any) should be included in $\mathcal{D}_{S_0}$ to support taxonomic reasoning about concepts and roles in the initial situation. In addition, $\mathcal{D}_{S_0}$ also includes all unique name axioms for object constants. Note that this definition of $\mathcal{D}_{S_0}$ includes ABox as a special case. In the sequel, $\mathcal{D}_{S_0}$ is assumed to be finite, unless stated otherwise.

- The remaining two classes (foundational axioms for situations $\Sigma$ and unique name axioms for actions $\mathcal{D}_{una}$) are the same as those in the BATs of the usual situation calculus. Note that these axioms (as well as $\mathcal{D}_{ap}$ and $\mathcal{D}_{ss}$) use more than two variables (e.g., $\mathcal{D}_{ss}$ use action and situation variables in addition to object variables). However, in the next section we will see that when considering the projection problem these axioms will no longer be needed after regression and the resulting formula uses no more than two variables.
3.3 Reasoning about Actions using Regression

After giving the definition of what is an $L_{sc}^{C2}$-restricted BAT in $L_{sc}^{C2}$, we turn our attention to the reasoning tasks. We want to identify reasoning problems that are decidable in $L_{sc}^{C2}$. To achieve this goal, for certain type of formulas in $L_{sc}^{C2}$, we expect the regressed formulas to be $C^2$ formulas.

3.3.1 Modified Regression with Lazy Unfolding

Given a formula $W$ of $L_{sc}^{C2}$ in the domain $D$, the definition of $W$ being regressable (called $L_{sc}^{C2}$ regressable below) is slightly different from the definition of $W$ being regressable in $L_{sc}$ (see [141]).

Definition 7  A formula $W$ of $L_{sc}^{C2}$ is $L_{sc}^{C2}$ regressable iff

1. Each term of sort Situation in $W$ is starting from $S_0$ and has the syntactic form $\text{do}([\alpha_1, \ldots, \alpha_n], S_0)$, where each $\alpha_i$ is ground action term in the language of $L_{sc}^{C2}$.

2. Other than the action terms occurred in atoms of the form $\text{Poss}(\alpha, \sigma)$ and in situ-
ation terms, there are no function terms in $W$. Moreover, variables $x$ and $y$ (free or bound) are the only variables used in $W$, if any.

3. For every atom of the form $\text{Poss}(\alpha, \sigma)$ in $W$, $\alpha$ has the syntactic form $A(t_1, \ldots, t_n)$ for some $n$-ary function symbol $A$ of $L_{sc}^{C2}$, where $n \leq 2$. Moreover, each $t_i$ is either variable $x$, variable $y$ or some constant $O$ if there is any.

4. $W$ does not mention the relation symbols “≺” or “=” between terms of sort Situation.

Def. 7 has several additional restriction conditions compared to Def. 2. The condition that $W$ does not quantify over situations is ensured by condition (1). The requirements of conditions (2) and (3) are obvious, because our language is restricted to $L_{sc}^{C2}$. The
intuition for requiring all situation terms to be ground in condition (1) is as follows. Consider the following counterexample.

**Example 2** Consider an $L_{sc}^{C^2}$-restricted BAT $D$, which includes an SSA

$$F(x, do(a, s)) \equiv a = A(x) \land (\exists y. G(x, y, s)) \lor F(x, s) \quad (3.6)$$

for some fluents $F(x, s), G(x, y, s)$ and action function $A(x)$. Consider a formula

$$\forall x. \exists y. F(x, do(A(O), do(A_1(y), S_0)))$$

(denoted as $W_0$ below)

of $L_{sc}^{C^2}$. To perform a correct regression on $W_0$ in the sense that the formula resulting from regression should be logically equivalent to $W_0$ w.r.t. $D$, we have to rename the quantified variable $y$ in Eq. (3.6) so that it is different from any variables appearing in $W_0$. That is, according to Eq. (3.6), the one step regression on $F$ using the regression operator $\mathcal{R}$ (Def. 3) should be

$$\forall x. \exists y. \mathcal{R}[A(O) = A(x) \land \exists z. G(x, z, do(A_1(y), S_0)) \lor F(x, do(A_1(y), S_0))]$$

Then, the regressed formula is no longer a formula of language $L_{sc}^{C^2}$. Otherwise, if we do not rename the quantified variable $y$ in Eq. (3.6) to ensure the regressed formula is still of language $L_{sc}^{C^2}$, then the one step regression on $F$ using Eq. (3.6) without renaming will result in the following formula:

$$\forall x. \exists y. \mathcal{R}[A(O) = A(x) \land \exists y. G(x, y, do(A_1(y), S_0)) \lor F(x, do(A_1(y), S_0))]$$

It is obvious that the above regressed formula is not logically equivalent to $W_0$, because the variable $y$ that occurs in the situation term $A_1(y)$ should not be quantified by $\exists y$ at the front of $G$.

Hence, to avoid the problem described in Example 2, we require all situation terms to be ground in Def. 7. Below, with a carefully defined regression operator, we are able to show that for every $L_{sc}^{C^2}$ regresssable formula, there is an equivalent regressed $C^2$ formula uniform in $S_0$. 

□
In the language of $\mathcal{L}_{sc}^{C^2}$, we have to be more careful with the definition of the modified regression operator, denoted $\mathcal{R}_{sc}^{C^2}$ below, for two main reasons. First, to deal with TBox we have to extend regression. For an $\mathcal{L}_{sc}^{C^2}$ regresstable formula $W$, we extend the regression operator defined in [141] with the lazy unfolding technique (see [4]) to expand defined dynamic concepts. Second, $\mathcal{L}_{sc}^{C^2}$ uses only two object variables and we have to make sure that after regressing a fluent atom we still get an $\mathcal{L}_{sc}^{C^2}$ formula, i.e., that we never need to introduce new (free or bound) object variables. To deal with the two-variable restriction, we modify our regression operator in comparison to the conventional operator defined in [141] (see Def. 3). The key idea is to reuse variables when doing replacement. For example, when replacing Poss atoms or fluent atoms about $do(\alpha, \sigma)$, the definition of the conventional regression operator in [141] has the assumption that the quantified variables on the right-hand side (RHS) of the corresponding axioms should be renamed to new variables different from the free variables in the atoms to be replaced. This assumption of using new variables for renaming ensures logical equivalence of the original formula and the formula after regression. But in $C^2$ new variables cannot be used. To avoid introducing new variables (as required by Reiter’s regression operator) and to ensure defined dynamic concepts being handled, we modify the regression operator. The possibility of using new variables for renaming is guaranteed by the general format of the axioms in $\mathcal{L}_{sc}^{C^2}$ restricted BATs, given in the previous section and the additional conditions in Def. 7.

The complete formal definition of our modified regression operator $\mathcal{R}_{sc}^{C^2}$ is as follows,\(^3\) where $\sigma$ denotes the term of sort $Situation$, and $\alpha$ denotes the term of sort $Action$. Note that below, if $\Phi(\vec{x})$ represents a formula $\Phi$ whose free variables are at most among a variable vector $\vec{x}$, then for any vector of terms $\vec{t}$ such that $|\vec{t}| = |\vec{x}|$ (i.e., the number of variables in $\vec{t}$ is the same as the number of variables in $\vec{x}$), $\Phi(\vec{t})$ represents the resulting formula obtained by substituting each $x_i$ in vector $\vec{x}$ with $t_i$ in vector $\vec{t}$ if $x_i$ occurs in $\Phi$. For example, in particular, if $\Phi(\vec{x})$ has no free variables, then no substitution happens

\(^3\)It is also called $\mathcal{L}_{sc}^{C^2}$ regression sometimes below to avoid confusion.
and $\Phi(\vec{t})$ is the same as $\Phi(\vec{x})$.

**Definition 8** Consider an $\mathcal{L}_{sc}^{C_2}$ regressable formula $W$ with respect to an $\mathcal{L}_{sc}^{C_2}$-restricted BAT $\mathcal{D}$ in $\mathcal{L}_{sc}^{C_2}$. Then, the modified regression operator $\mathcal{R}^{C_2}$ is recursively defined as follows.

- If $W$ is not atomic, i.e., $W$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W'$, or $Qv.W'$ where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol, then
  \[
  \mathcal{R}^{C_2}[W_1 \lor W_2] = \mathcal{R}^{C_2}[W_1] \lor \mathcal{R}^{C_2}[W_2], \quad \mathcal{R}^{C_2}[\neg W'] = \neg \mathcal{R}^{C_2}[W'],
  \]
  \[
  \mathcal{R}^{C_2}[W_1 \land W_2] = \mathcal{R}^{C_2}[W_1] \land \mathcal{R}^{C_2}[W_2], \quad \mathcal{R}^{C_2}[Qv.W'] = Qv.\mathcal{R}^{C_2}[W'].
  \]

- Otherwise, $W$ is an atom. There are several cases.

  a. If $W$ is a regressable $\text{Poss}$ atom, then it has the form $\text{Poss}(A(\vec{t}), \sigma)$ for terms of sort $\text{Action}$ and $\text{Situation}$ respectively in $\mathcal{L}_{sc}^{C_2}$. Then, there must be an action precondition axiom for $A$ of the form $\text{Poss}(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$, where the argument $\vec{x}$ of sort $\text{Object}$ can either be empty (i.e., $A$ is an action constant), a single variable $x$, or two-variable vector $\langle x, y \rangle$. Because of the syntactic restrictions of $\mathcal{L}_{sc}^{C_2}$ and according to the conditions in Def. 7, each term in $\vec{t}$ can only be a variable $x$, $y$ or some constant if any. Then,

  \[
  \mathcal{R}^{C_2}[W] = \begin{cases} 
  \mathcal{R}^{C_2}[\exists y.x = y \land \Pi_A(x, y, \sigma)] & \text{if } \vec{t} = \langle x, x \rangle, \\
  \mathcal{R}^{C_2}[\exists x.y = x \land \Pi_A(x, y, \sigma)] & \text{if } \vec{t} = \langle y, y \rangle, \\
  \tilde{\Pi}_A(\vec{t}, \sigma) & \text{if } \vec{t} \in \{y, \langle y, O \rangle, \langle O, x \rangle, \langle y, x \rangle\}, \\
  \Pi_A(\vec{t}, \sigma) & \text{otherwise, i.e., if } \vec{t} \text{ is empty or } \\
  \tilde{\vec{t}} \in \{O, x, \langle x, y \rangle, \langle x, O \rangle, \langle y, O \rangle, \langle O, O_1 \rangle\},
  \end{cases}
  \]

where $O$ and $O_1$ are constants and $\tilde{\phi}$ denotes a dual formula for formula $\phi$ obtained by replacing every variable symbol $x$ (free or bound) with variable symbol $y$ and
replacing every variable symbol $y$ (free or bound) with variable symbol $x$ in $\phi$, i.e., $\bar{\phi} = \phi[y/x, y/x]$. In this definition, in order to avoid introducing new variables but still ensure the correctness of regression in the sense that the regressed formula is logically equivalent to $W$ w.r.t. $D$, we consider all the possible syntactic forms of the arguments $\vec{t}$ in action terms and treat them carefully in each of the four cases. Because of the restriction of the language of $L_{sc}^{C2}$ and the additional conditions in Def. 7, we are able to reuse the variables $x$ and $y$ by switching the occurrences of $x$ and $y$ when $\vec{t}$ is either $y$, $\langle y, O \rangle$, $\langle O, x \rangle$ or $\langle y, x \rangle$.

b. If $W$ is a defined dynamic concept, it has the form $G(t, \sigma)$ for some object term $t$ and ground situation term $\sigma$, and there must be a TBox axiom for $G$ of the form $G(x, s) \equiv \phi_G(x, s)$. Because of the restrictions of the language $L_{sc}^{C2}$, term $t$ can only be a variable $x$, $y$ or a constant. Then, we use the lazy unfolding technique as follows:

$$R_{C2}^{C2}[W] = \begin{cases} R_{C2}^{C2}[\phi_G(t, \sigma)] & \text{if } t \in \{O, x\}, \\ R_{C2}^{C2}[\bar{\phi}_G(y, \sigma)] & \text{otherwise, i.e., if } t = y. \end{cases}$$

c. If $W$ is a primitive dynamic concept (a dynamic role, respectively), it is of the form $F(t_1, do(\alpha, \sigma))$ (or $F(t_1, t_2, do(\alpha, \sigma))$, respectively) for some terms $t_1$ (and $t_2$) of sort $Object$, ground term $\alpha$ of sort $Action$ and ground term $\sigma$ of sort $Situation$. Then, there must be an SSA for fluent of the form $F(x, do(a, s)) \equiv \Phi_F(x, a, s)$ ($F(x, y, do(a, s)) \equiv \Phi_F(x, y, a, s)$, respectively), whose detailed syntax is Eq. (3.2). Because of the restriction of the language $L_{sc}^{C2}$, the terms $t_1$ and $t_2$ can only be a variable $x$, $y$ or some constant $O$. Then, when $W$ is a primitive dynamic concept, i.e., $W$ is of the form $F(t_1, do(\alpha, \sigma))$,

$$R_{C2}^{C2}[W] = \begin{cases} R_{C2}^{C2}[\Phi_F(t_1, \alpha, \sigma)] & \text{if } t_1 \in \{O, x\}, \\ R_{C2}^{C2}[\bar{\Phi}_F(y, \alpha, \sigma)] & \text{otherwise, i.e., if } t_1 = y; \end{cases}$$

and, when $W$ is a dynamic role, i.e., $W$ is of the form $F(t_1, t_2, do(\alpha, \sigma))$, 
\[ R^C_2[\exists y. y = x \land \Phi_F(x, y, \alpha, \sigma)] \text{ if } \langle t_1, t_2 \rangle = \langle x, x \rangle; \]
\[ R^C_2[\exists x. x = y \land \Phi_F(x, y, \alpha, \sigma)] \text{ if } \langle t_1, t_2 \rangle = \langle y, y \rangle; \]
\[ R^C_2[\Phi_F(t_1, t_2, \alpha, \sigma)] \text{ if } \langle t_1, t_2 \rangle \in \{\langle y, x \rangle, \langle y, O \rangle, \langle O, x \rangle\}; \]
\[ R^C_2[\Phi_F(t_1, t_2, \alpha, \sigma)] \text{ otherwise, i.e., if } \langle t_1, t_2 \rangle \in \{\langle x, y \rangle, \langle x, O \rangle, \langle O, y \rangle, \langle O, O \rangle\}. \]

According to the restriction of \( L^C_{sc} \) (particularly according to Eq. (3.2), action variable \( a \) and situation variable \( s \) are free and all other variables are either \( x \) or \( y \)) and in \( W \) (i.e., \( F(\vec{t}, do(\alpha, \sigma)) \)) the terms \( \alpha \) and \( \sigma \) are ground. Therefore, the regression of \( W \) can be defined by switching the occurrences of \( x \) and \( y \) in the RHS of the SSA of the fluent \( F(\vec{x}, do(a, s)) \) (\( \vec{x} \) is either \( x \) or \( \langle x, y \rangle \)), i.e., \( \Phi_F(x, y, a, s) \), and by substituting free variables with the corresponding terms \( \vec{t}, \alpha \) and \( \sigma \) when \( \vec{t} \) (\( t_1 \) or \( \langle t_1, t_2 \rangle \)) in \( W \) is either \( y \), \( \langle y, O \rangle \), \( \langle O, x \rangle \) or \( \langle y, x \rangle \). The definition ensures the one-step regression result is still equivalent to \( W \) without using any new variables.

d. If \( W \) is of the form \( A_1(\vec{t}) = A_2(\vec{t'}) \) for some action function symbols \( A_1 \) and \( A_2 \), then by using axioms in \( D_{una,4} \) we define the regression of \( W \) as
\[
R^C_2[W] = \begin{cases} 
false & \text{if } A_1 \neq A_2, \\
true & \text{if } A_1 = A_2 \text{ and } A_1, A_2 \text{ are constant action functions,} \\
\bigwedge_{i=1}^{\vec{t}} t_i = t'_i & \text{otherwise.}
\end{cases}
\]

Otherwise, if \( W \) is any other situation independent atom (including equality between object terms) or \( W \) is a concept or role uniform in \( S_0 \), then
\[
R^C_2[W] = W. \quad (3.7)
\]

\(^4\)Notice that the action functions with different number of arguments always use different function symbols (i.e., different names).
Our intention in case (d.) in Def. 8 is to get a $C^2$ formula (with any situation term suppressed) that has no (in)equality between action terms after regression. We therefore cannot leave (in)equalities between action terms untouched in the regressed formula unlike Reiter’s definition of the regression operator that simply used Eq. (3.7) when dealing with (in)equalities between terms. By using unique name axioms for actions during regression, we can avoid functional terms in the resulting formula. Moreover, in case (a.) and case (c.) of Def. 8, when $\vec{t}$ is $\langle x, x \rangle$ (or $\langle y, y \rangle$, respectively), we define regression by using a quantified variable $y$ (or $x$, respectively); otherwise, we cannot ensure the correctness of regression in the sense that the regressed formula is logically equivalent to $W$ w.r.t. $D$, i.e., $D \models W \equiv R_{C^2}[W]$. In particular, for any $L_{SC}^{C^2}$ regressive formulas $W$ and $W'$ such that $D \models W \equiv W'$, a correct definition of regression should result in $D \models R_{C^2}[W] \equiv R_{C^2}[W']$. For example, in case (c), notice that

$$\models F(x, x, do(\alpha, \sigma)) \equiv (\exists y. x = y \land F(x, y, do(\alpha, \sigma)),$$

and it is easy to see that our definition of $R_{C^2}$ ensures that

$$\models R_{C^2}[F(x, x, do(\alpha, \sigma))] \equiv R_{C^2}[(\exists y. x = y \land F(x, y, do(\alpha, \sigma))].$$

Consider the following counterexample if we perform regression by directly substituting $\langle x, y \rangle$ by $\langle x, x \rangle$ (or $\langle y, y \rangle$) on the RHS of SSAs or precondition axioms.

**Example 3** Consider a $L_{SC}^{C^2}$-restricted BAT $D$, which includes an SSA

$$F(x, y, do(\alpha, s)) \equiv a = A(x) \land (\exists x. G(x, y, s)) \lor F(x, y, s). \quad (3.8)$$

Consider an $L_{SC}^{C^2}$ regressive formula $W = F(x, x, do(A(C), S_0))$. Then if we perform regression by directly substituting $\langle x, y \rangle$ with $\langle x, x \rangle$ on the RHS of Eq. (3.8), we get $A(C) = A(x) \land (\exists x. G(x, x, S_0)) \lor F(x, x, S_0))$, which obviously will not be logically equivalent to $W$ w.r.t. $D$. Indeed, in Eq. (3.8), the variable $y$ in $G$ is free, but once substituted by $x$ directly, it becomes bound by $\exists x$ at the front of $G$, which should not happen.

Based on Def. 8, we are able to prove the following theorems.
Theorem 2 Suppose $W$ is an $\mathcal{L}_{sc}^{C^2}$ regressable formula, then the regression $\mathcal{R}^{C^2}[W]$ defined above terminates in a finite number of steps.

Proof: This immediately follows from Def. 7, acyclicity of the TBox axioms, and from the assumption that $RBox$ axioms are compiled into the SSAs and consequently do not participate in regression. Note also that each time, the application of $\mathcal{R}^{C^2}$ either goes from a formula to a sub-formula, or expands a $Poss$ or a fluent atom using a corresponding precondition axiom or an SSA, but only finitely many expansions are possible because $W$ is a finite formula, mentioning only finitely many ground situation terms. $\square$

Moreover, the following statement can be proved for $\mathcal{R}^{C^2}$ by induction over the structure of $W$. Since the detailed proof of Th. 3 is rather long and mechanical, it is provided in Appendix B.1.

Theorem 3 Suppose $W$ is an $\mathcal{L}_{sc}^{C^2}$ regressable sentence with the background BAT $\mathcal{D}$ in language $\mathcal{L}_{sc}^{C^2}$. Then, $\mathcal{R}^{C^2}[W]$ is an $\mathcal{L}_{sc}^{C^2}$ sentence uniform in $S_0$ and it is a $C^2$ sentence when the situation argument $S_0$ is suppressed. Moreover, $\mathcal{D} \models W \equiv \mathcal{R}^{C^2}[W]$.

Consequently, we have the following theorem.

Theorem 4 Suppose $W$ is an $\mathcal{L}_{sc}^{C^2}$ regressable sentence with the background BAT $\mathcal{D}$ in language $\mathcal{L}_{sc}^{C^2}$. Then, $\mathcal{D} \models W$ iff $\mathcal{D}_{S_0} \models \mathcal{R}^{C^2}[W]$.

Proof: Part of our proof is almost word-by-word repetition of the laborious proof of the regression theorem given in [134]. Therefore, we will only briefly explain the idea of what has been done in [134] and provide details for what is different.

In [134], Pirri and Reiter first proved that “a BAT $\mathcal{D}$ is satisfiable iff $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ is satisfiable”. It is trivial that if a BAT $\mathcal{D}$ is satisfiable then $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ is satisfiable. For the other direction, Pirri and Reiter proved it by constructing a model step by step for $\mathcal{D}$ that interprets axioms in $\mathcal{D} - (\mathcal{D}_{S_0} \cup \mathcal{D}_{una})$ properly, starting from any model of $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$. Similarly, we can also prove that “an $\mathcal{L}_{sc}^{C^2}$-restricted BAT $\mathcal{D}$ is satisfiable iff
\( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \) is satisfiable”. We use the same idea with the following modification. For any model \( M \) of \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \), we also add interpretations for the defined concepts, such that the interpretations for \( G(x)[s] \) are true iff those of \( \phi_G(x)[s] \) are.

Subsequently, in [134], Pirri and Reiter proved the following lemma by using the above result: “Suppose \( W \) is a regressable sentence with the background BAT \( \mathcal{D} \) that is uniform in \( S_0 \), then \( \mathcal{D} \models W \) iff \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models W \)” This lemma is also valid for \( \mathcal{L}_{sc}^{C^2} \). That is, suppose \( W \) is a \( \mathcal{L}_{sc}^{C^2} \) regressable sentence with the background BAT \( \mathcal{D} \) in \( \mathcal{L}_{sc}^{C^2} \), if \( W \) is uniform in \( S_0 \), then \( \mathcal{D} \models W \) iff \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models W \).

Now we prove the following statement: “Suppose \( W \) is an \( \mathcal{L}_{sc}^{C^2} \) regressable sentence with the background BAT \( \mathcal{D} \) in \( \mathcal{L}_{sc}^{C^2} \). If \( W \) is uniform in \( S_0 \) and \( W \) is a \( C^2 \) formula when \( S_0 \) is suppressed, then \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \neg W \) iff \( \mathcal{D}_{S_0} \models \neg W \).” It is trivial to see that if \( \mathcal{D}_{S_0} \models \neg W \), then \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models W \). For the other direction, it is the same as proving if \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \{ \neg W \} \) is inconsistent, so is \( \mathcal{D}_{S_0} \cup \{ \neg W \} \). We can prove it by contradiction. That is, assume that \( \mathcal{D}_{S_0} \cup \{ \neg W \} \) is consistent, then there is a model \( M_0 \), such that \( M_0 \models \mathcal{D}_{S_0} \cup \{ \neg W \} \). We can then construct a model \( M \) such that it has the same domain on \( Object \) sort as \( M_0 \). For any action functions \( A(\bar{x}_1), B(\bar{x}_2) \), we construct \( M \) so that \( (A(\bar{x}_1))^M \neq (B(\bar{x}_2))^M \) for any variable vectors \( \bar{x}_1 \) and \( \bar{x}_2 \) if symbol A is different from B; and for any object terms \( \bar{t}_1 = (t_{1,1} \cdots t_{1,n}) \) and \( \bar{t}_2 = (t_{1,1} \cdots t_{1,n}) \) and any \( n \)-ary action function \( A, (A(\bar{t}_1))^M = (A(\bar{t}_2))^M \) iff \( (t_{1,i})^M = (t_{2,i})^M \) for all \( i = 1 .. n \). Moreover, other interpretations of predicates or terms in \( M \) are the same as that of \( M_0 \). Since \( \mathcal{D}_{S_0} \cup \{ \neg W \} \) has no action terms in it, \( M \) is well defined and is a model of \( \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \{ \neg W \} \), which is a contradiction.

Hence, in summary, we have: suppose \( W \) is an \( \mathcal{L}_{sc}^{C^2} \) regressable sentence with a background BAT \( \mathcal{D} \) in \( \mathcal{L}_{sc}^{C^2} \), if \( W \) is uniform in \( S_0 \) and \( W \) is a \( C^2 \) formula when \( S_0 \) is suppressed, then \( \mathcal{D} \models W \) iff \( \mathcal{D}_{S_0} \models W \).

Then, by Th. 3 and the above proved statement, we have \( \mathcal{D} \models W \) iff \( \mathcal{D} \models \mathcal{R}^{C^2}[W] \) iff \( \mathcal{D}_{S_0} \models \mathcal{R}^{C^2}[W] \). \( \square \)
We can also obtain the following theorem about decidability of the projection problem for $\mathcal{L}^{C_2}_{sc}$ regressable sentence $W$. (In particular, when $W$ is of the form executable($S$) for some ground situation $S$, it becomes the executability problem.)

**Theorem 5** Suppose $W$ is an $\mathcal{L}^{C_2}_{sc}$ regressable sentence with the background BAT $\mathcal{D}$ in language $\mathcal{L}^{C_2}_{sc}$. Then, the entailment problem $\mathcal{D} \models W$ is decidable.

**Proof:** According to Th. 4, $\mathcal{D} \models W$ iff $\mathcal{D}_{S_0} \models \mathcal{R}^{C_2}[W]$, where $\mathcal{R}^{C_2}[W]$ and the axioms in $\mathcal{D}_{S_0}$ are $C_2$ formulas. Therefore, to determine whether or not $\mathcal{D} \models W$ is the same to determine whether or not $\mathcal{D}_{S_0} \land \neg \mathcal{R}^{C_2}[W]$ is unsatisfiable ($\mathcal{D}_{S_0}$ can be considered as a conjunction of all axioms in the initial theory). The later one is a decidable problem according to the fact that the satisfiability problem in $C_2$ is decidable. □

This theorem is important because it guarantees that the projection and executability problems in $\mathcal{L}^{C_2}_{sc}$ are decidable even if the initial knowledge base $\mathcal{D}_{S_0}$ is incomplete. In Section 3.3.3, we give a detailed example that illustrates the basic reasoning tasks described above and reduction techniques for dealing with properties that need more than two variables, and show that using $\mathcal{L}^{C_2}_{sc}$, one can model realistic dynamic domains such as school enrolment services.

### 3.3.2 Some Computational Complexity Results via Regression

Since an executability problem can be reduced to a projection problem in linear time, below we will focus on the study of projection problems only. In this section, we consider the computational time of solving projection problems via regression in $\mathcal{L}^{C_2}_{sc}$. In Section 3.3.4, we propose a sub-class of $\mathcal{L}^{C_2}_{sc}$, which has better computational property when solving projection problems via regression. In Section 3.4, we study the computational complexity of solving projection problems in general (without using regression) and discuss why it is still useful (sometimes) to study solving projection problems via regression even if it has a higher computational upper bound. We first introduce a few
new notations for later convenience.

For any $L_{sc}^{C_2}$ regressable formula $W$, let function $size(W)$ be the size of formula $W$, which is defined recursively:

1. If $W$ is atomic (including equality), then $size(W) = 1$.

2. If $W$ is of the form $\neg W_1$, $\exists x.W_1$, $\forall x.W_1$, $\exists^\geq n x.W_1$, $\exists^\leq n x.W_1$, $\exists y.W_1$, $\forall y.W_1$, $\exists^\geq n y.W_1$, or $\exists^\leq n y.W_1$, then $size(W) = size(W_1) + 1$.

3. If $W$ is of the form $W_1 \land W_2$ or $W_1 \lor W_2$, then $size(W) = size(W_1) + size(W_2) + 1$.

4. If $W$ is of the form $W_1 \supset W_2$, then $size(W) = size(\neg W_1 \lor W_2)$.

5. If $W$ is of the form $W_1 \equiv W_2$, then $size(W) = size(W_1 \supset W_2) + size(W_2 \supset W_1) + 1$.

For any situation term $\sigma = do([\alpha_1, \cdots, \alpha_k], S_0)$, let function $sitLength(\sigma) = k$ represent the number of action terms mentioned in $\sigma$. In particular, $sitLength(S_0) = 0$. For any formula $W$, let function

$$maxSit(W) = \max_\sigma \{sitLength(\sigma) \mid \sigma \text{ appears in } W\}.$$ 

Given any BAT $D$, for any fluent $F$ whose SSA is of the form Eq. (3.2), let function $numFluent(F)$ be the number of fluents (including repeated ones) appearing in $\Phi_F$ (the RHS of the SSA of fluent $F$), and let

$$numFluentSSA(D) = \max_F \{numFluent(F) \mid \text{any } F \text{ that has an SSA in } D\}.$$ 

Besides, let

$$sizeSSA(D) = \max_F \{size(\Phi_F) \mid \text{any SSA } F(\vec{x}, do(a, s)) \equiv \Phi_F \text{ in } D\}.$$ 

Note that $numFluentSSA(D)$ and $sizeSSA(D)$ are different: the former one is the maximal number of fluents SSA appearing in the formulas that are on the RHS of the SSAs of a given BAT $D$, and the latter one is the maximal size of the formulas (including non-fluent atoms and logical connectives) that are on the RHS of the SSAs of a given BAT $D$. 
Moreover, notice that once $D$ is given, $numFluentSSA(D)$ and $sizeSSA(D)$ are fixed. In general, we have the following result.

**Theorem 6** Consider any $\mathcal{L}_{sc}^{C2}$ regressive formula $W$ with a fixed BAT $D$ in $\mathcal{L}_{sc}^{C2}$. Then, answering the query whether $D \models W$ via regression is in the complement of 2-NExpTime with respect to the size of $W$.

**Proof:** According to the discussion in the proof of Th. 5, determining whether $D \models W$ is equivalent to determining whether $D_{S_0} \land \neg R^{C2}[W]$ is unsatisfiable or not, i.e., the complement problem of whether $D_{S_0} \land \neg R^{C2}[W]$ is satisfiable or not.

Note that $D_{S_0} \land \neg R^{C2}[W]$ is a $C^2$ formula (when the situation argument $S_0$ is suppressed). Moreover, according to [136], the satisfiability problem in language $C^2$ is in NExpTime, that is, $\bigcup_{k \in \mathbb{N}} \text{NTIME}(2^{n_k})$ if the input size of the formula is $n$. Note that NTIME($f(n)$) represents the set of decision problems that can be solved by a non-deterministic Turing machine in time $O(f(n))$ (with unlimited space). Since for any given $D$, the size of $D_{S_0}$ is fixed, the size of $\neg(D_{S_0} \land \neg R^{C2}[W])$ is in $\Theta(n_1)$, where $n_1 = \text{size}(R^{C2}[W])$. Hence, answering the query whether $D \models W$ via regression is in $\text{co-NTIME}(2^{n_1})$, i.e., the complement of NExpTime w.r.t. the size of $R^{C2}[W]$.

However, in the worst case, computing $R^{C2}[W]$ takes ExpTime w.r.t. $n$, where $n = \text{max} \text{Sit}(W)$, and causes exponential blow-up in the size of formula $W$ w.r.t. $n$. In detail, without loss of generality, we assume that there is no defined concept in $W$. Otherwise, each defined concept will be replaced by its definitions from the TBox axioms within finite number of $\mathcal{L}_{sc}^{C2}$ regression steps. This can cause no more than a constant increase to the size of the original formula, because TBox is fixed (once $D$ is given), TBox is acyclic, there are only finitely many $TBox$ axioms and the size of the formula on the RHS of each TBox axiom is limited by a constant.

Now, let $h = \max(2, \text{sizeSSA}(D))$, $m = \text{size}(W)$, $n = \text{max} \text{Sit}(W)$, and $k = \text{numFluentSSA}(D)$. Both $k$ and $h$ are constants for the given BAT $D$, and $h > k$. 
Moreover, since all action functions in $L_{sc}^{C^2}$ have no more than two arguments, the regression on equalities between action terms (see case (a.) in Def. 8) results in a formula whose size is no more than three times of the size of the original atomic formulas (including at most one conjunction operator and at most two equality atoms), and such regression applies only once to each equality between action terms. The worst case scenario happens if each SSA mentions all $k$ fluents and the size of the RHS of the SSA is $h$. In this case, each step of regression on a fluent atom creates at most $h$ new branches on the next level of the regression tree (k out of these $h$ branches have atomic fluents as their nodes). The next application of the regression operator replaces fluents in these $k$ nodes by the RHS of the corresponding SSAs, and so on. Since the longest situation term in $W$ before regression is of length $n$, the height of the resulting regression tree is no more than $n + 1$ (including $n$ levels of regressions on fluents and at most 1 level of regression on the equalities between actions), assuming that the regression tree starts at level 0 for root $W$. Finally, we are looking for a total number of leaves in this tree (this number is the size of the regressed formula $RC^2[W]$), which is

$$
\text{size}(RC^2[W]) \leq 3m[1 + (h - k)\sum_{i=0}^{n-1} k^i]
$$

$$
= \begin{cases} 
3m(h - 1)n + 3m & \text{if } k = 1 \\
3m[1 + (h - k)\frac{k^n - 1}{k-1}] & \text{if } k > 1.
\end{cases}
$$

$$
\leq \begin{cases} 
3m(h - 1)n + 3m & \text{if } k = 1 \\
3mhk^n & \text{if } k > 1.
\end{cases}
$$

Clearly, $3mhk^n$ is no more than $3h(m2^{n\log_2(k+1)})$. Formally, it is straightforward to prove by induction according to the recursive definition of the regression operator that $\text{size}(RC^2[W]) \in O(mn)$ when $k = 1$, and $\text{size}(RC^2[W]) \in O(mk^n)$ (which is the same as $\text{size}(RC^2[W]) \in O(m2^{n\log_2(k+1)})$) when $k \geq 2$. Overall, in the worst case scenario, answering the query whether $D \models W$ via regression is in the complement of NTIME$(2^{3hm2^{n\log_2(k+1)}})$ according to the above discussion, where $h$ and $k$ are constants.
That is, answering the query whether $D \models W$ via regression is in the complement of 2-NExpTime (non-deterministic doubly-exponential time). □

The SSA for a fluent $F$ is called *context-free* if the axiom has the syntactic form

$$F(\bar{x}, \text{do}(a, s)) \equiv \gamma^+_F(\bar{x}, a) \lor F(\bar{x}, s) \land \neg \gamma^-_F(\bar{x}, a),$$

that is, both the *positive condition* $\gamma^+_F(\bar{x}, a)$ and the *negative condition* $\gamma^-_F(\bar{x}, a)$ are situation independent (see Chapter 4 in [141]). According to this definition, it is easy to see that all the context conditions $(\psi^+_i(\bar{x}_{(i,+)}))[s]$ for all $1 \leq i \leq m_+$ and $(\psi^-_j(\bar{x}_{(j,-)}))[s]$ for all $1 \leq i \leq m_-$ in Eq. (3.2) are situation independent (i.e., there is no $s$ in any of the context conditions). Note that there is a special case if a positive (or negative, respectively) effect only depends on some action term (i.e., there is no context condition, or, the corresponding context condition is always equivalent to true). Then, we have the following theorem about the computational complexity for reasoning about projection problems.

**Theorem 7** Given an $\mathcal{L}^{C^2}_{sc}$-restricted BAT $D$, suppose that the SSA for a fluent $F$ is *context-free*. Then, the computational complexity of answering the queries of the form $F(\bar{X}, \sigma)$ via regression is in co-NExpTime, where $\bar{X}$ is a vector of object constants and $\sigma$ is a ground situation term.

**Proof:** The result follows from the analysis of the computational complexity of the projection problem in [141] (Chapter 4), which shows that the complexity is at most linear to the complexity of evaluating a sentence in the initial situation under such assumptions. In fact, from the proof of Th. 6, when a fluent $F$ is context-free, $\text{numFluent}(F) = 1$. Let $\text{size}(\Phi_F) = h_0$, then the number of leaves in the regression tree of $F(\bar{X}, \sigma)$, i.e., the size of the regressed formula, equals $n(h_0 - 1) + 1$, which is in $O(n)$. Again, using the same reasoning as in the proof of Th. 6, the problem of answering the queries of the form $F(\bar{X}, \sigma)$ via regression is the complement of the problem whether $D_{S_0} \land \neg F(\bar{X}, \sigma)$
is satisfiable or not. Hence, its computational complexity is in \( co\-NTIME(2^{O(n)}) \), i.e., it is in \( co\-NExpTime \).

Using the same reasoning as in Th. 7, in general, we have the following corollary.

**Corollary 1** Given an \( \mathcal{L}_{sc}^{C^2} \)-restricted BAT \( \mathcal{D} \), suppose that every SSA in \( \mathcal{D} \) is context-free. Then, for any \( \mathcal{L}_{sc}^{C^2} \) regressable formula \( W \), answering the query whether \( \mathcal{D} \models W \) via regression is in \( co\-NExpTime \).

In Section 3.3.4, we further consider some special cases of BATs that have better computational complexity results for solving executability and projection problems (via regression), but less expressive power than \( \mathcal{L}_{sc}^{C^2} \)-restricted BATs.

### 3.3.3 Examples of BATs and Regression in \( \mathcal{L}_{sc}^{C^2} \)

Here, we give some examples of an \( \mathcal{L}_{sc}^{C^2} \)-restricted BAT \( \mathcal{D} \) to illustrate the ideas described above. We first consider a scenario of university enrolment.

**Example 4** Consider some university that provides student administration and management services on the Web: admitting students, paying tuition fees, enrolling or dropping courses and entering grades.

Although the number of object variables in the predicates can be at most two, sometimes, we are still able to handle those features of the systems that require more than two variables. For example, the grade \( z \) of a student \( x \) in a course \( y \) may be represented as a predicate \( grade(x, y, z) \) in the general first-order logic (i.e., with three object variables). Because the number of distinct grades is finite and they can be easily enumerated, e.g., as “A”, “B”, “C”, or “D”, we can handle \( grade(x, y, z) \) by replacing it with a finite number of extra predicates, say \( gradeA(x, y) \), \( gradeB(x, y) \), \( gradeC(x, y) \) and \( gradeD(x, y) \) such that they all involve only two object variables (that can be quantified over).

However, the syntactic restriction of \( \mathcal{L}_{sc}^{C^2} \) limits the expressive power of the language
if an action function or a fluent has more than two arguments vary over infinite domains (such as energy, weight, time, etc), in which case we generally are not able to specify the action precondition or SSAs in $\mathcal{L}_{sc}^{C^2}$ with at most two variables ($x$ and $y$) other than the action variable $a$ and the situation variable $s$. Despite this limitation, we conjecture that many Web services still can be represented with at most two object variables either by introducing extra predicates (just like we did for the predicate $\text{grade}$) or by grounding some of the arguments if their domains are finite and relatively small. In Section 3.5 below, we will discuss a possible future research direction on loosening the restriction of the syntax of action functions and the usage of variables. But for now, we focus our attention on $\mathcal{L}_{sc}^{C^2}$.

Intuitively, it seems that many practical dynamical systems can be specified by using properties and actions with small arities, hence the techniques for arity reductions mentioned above and below require no more than polynomial increase in the number of axioms. The high-level features of our example are specified as the following concepts and roles.

- **Primitive static concepts:** $\text{person}(x)$ ($x$ is a person), $\text{course}(x)$ ($x$ is a course provided by the university).
- **Primitive dynamic concepts:** $\text{incoming}(x, s)$ ($x$ is an incoming student in the situation $s$, it is true when $x$ was admitted).
- **Defined dynamic concepts:** $\text{eligFull}(x, s)$ ($x$ is eligible to be a full-time student by paying more than 5000 dollars tuition fee), $\text{eligPart}(x, s)$ ($x$ is eligible to be a part-time student by paying no more than 5000 dollars tuition), $\text{qualFull}(x, s)$ ($x$ is a qualified full-time student if he or she pays full time tuition fee and takes at least 4 courses), $\text{qualPart}(x, s)$ ($x$ is a part-time student if he or she pays part-time tuition and takes 2 or 3 courses).
- **Static role:** $\text{preReq}(x, y)$ (course $x$ is a prerequisite for course $y$).
- **Dynamic roles:** $\text{hasGrade}(x, y, s)$ ($x$ has a grade for course $y$ in the situation $s$),
tuitPaid(x, y, s) (x paid tuition fee y in the situation s), enrolled(x, y, s) (x is enrolled in course y in the situation s), completed(x, y, s) (x completes course y in the situation s), gradeA(x, y, s), gradeB(x, y, s), gradeC(x, y, s), gradeD(x, y, s).

Web services are specified as actions: reset (at the beginning of each academic year, the system is being reset so that students need to pay tuition fee again to become eligible), admit(x) (the university admits student x), payTuit(x, y) (x pays tuition fee with the amount of y), enroll(x, y) (x enrolls in course y), drop(x, y) (x drops course y), enterA(x, y) (enter grade “A” for student x in course y), enterB(x, y), enterC(x, y), enterD(x, y).

The BAT of the system is as follows (most of the axioms are self-explanatory).

Action Precondition Axioms:

\[
\text{Poss}(\text{reset}, s) \equiv \text{true},
\]

\[
\text{Poss}(\text{admit}(x), s) \equiv \text{person}(x) \land \neg \text{incoming}(x, s),
\]

\[
\text{Poss}(\text{payTuit}(x, y), s) \equiv \text{incoming}(x, s) \land \neg \text{tuitPaid}(x, y, s),
\]

\[
\text{Poss}(\text{drop}(x, y), s) \equiv \text{enrolled}(x, y, s) \land \neg \text{completed}(x, y, s),
\]

\[
\text{Poss}(\text{enterA}(x, y), s) \equiv \text{enrolled}(x, y, s) \land \neg \text{completed}(x, y, s),
\]

and the precondition axiom for enterB(x, y) (enterC(x, y) and enterD(x, y), respectively) is similar to the axiom for enterA(x, y). Moreover, in the traditional situation calculus, the precondition for action enroll(x, y) would be equivalent to

\[
\forall z. \text{preReq}(z, y) \land \text{completed}(x, z, s) \land \neg \text{gradeD}(x, z, s).
\]

However, in the modified situation calculus, we only allow at most two object variables (including free or bound). Fortunately, the number of the courses offered in a university is limited (finite and relatively small) and relatively stable over years (if we manage the students in a college-wise range or department-wise range, the number of courses may be even smaller). Therefore, we can specify the precondition for the action enroll(x, y) for each instance of y. That is, assume that the set of courses is \{CS_1, \cdots, CS_n\}, the precondition axiom for each action CS_i (i = 1..n) is
Poss(enroll(x,CS_i),s) \equiv \\
\forall y.\text{preReq}(y,CS_i) \supset \text{completed}(x,y,s) \land \neg \text{gradeD}(x,y,s).

On the other hand, when we do this transformation, we can omit the statements \text{course}(x)
for each course available at the university in the initial theory.

**Successor State Axioms:**

\begin{align*}
\text{incoming}(x,do(a,s)) & \equiv a=\text{admit}(x) \lor \text{incoming}(x,s), \\
\text{tuitPaid}(x,y,do(a,s)) & \equiv a=\text{payTuit}(x,y) \lor \text{tuitPaid}(x,y,s) \land a \neq \text{reset}, \\
\text{gradeA}(x,y,do(a,s)) & \equiv a=\text{enterA}(x,y) \lor \text{gradeA}(x,y,s) \\
& \quad \land \neg(a=\text{enterB}(x,y) \lor a=\text{enterC}(x,y) \lor a=\text{enterD}(x,y)), \\
\text{enrolled}(x,y,do(a,s)) & \equiv a=\text{enroll}(x,y) \lor \text{enrolled}(x,y,s) \land \neg(a=\text{drop}(x,y) \\
& \quad \lor a=\text{enterA}(x,y) \lor a=\text{enterB}(x,y) \lor a=\text{enterC}(x,y) \lor a=\text{enterD}(x,y)), \\
\text{completed}(x,y,do(a,s)) & \equiv a=\text{enterA}(x,y) \lor a=\text{enterB}(x,y) \lor a=\text{enterC}(x,y) \\
& \quad \lor a=\text{enterD}(x,y) \lor \text{completed}(x,y,s) \land a \neq \text{enroll}(x,y).
\end{align*}

The SSAs for the fluents \text{gradeB}(x,y,s), \text{gradeC}(x,y,s) and \text{gradeD}(x,y,s) are very similar to the one for fluent \text{gradeA}(x,y,s). Therefore they are not repeated here. The SSA for the fluent \text{hasGrade}(x,y,s) is also similar and for this reason it is omitted.

**Acyclic TBox Axioms:** (no static TBox axioms in this example)

\begin{align*}
\text{eligFull}(x,s) & \equiv (\exists y)(\text{tuitPaid}(x,y,s) \land y > 5000), \\
\text{eligPart}(x,s) & \equiv (\exists y)(\text{tuitPaid}(x,y,s) \land y \leq 5000), \\
\text{qualFull}(x,s) & \equiv \text{eligFull}(x,s) \land (\exists \geq 4 y)\text{enrolled}(x,y,s), \\
\text{qualPart}(x,s) & \equiv \text{eligPart}(x,s) \land (\exists \geq 2 y)\text{enrolled}(x,y,s) \land (\exists \leq 3 y)\text{enrolled}(x,y,s).
\end{align*}

An example of the initial theory \(\mathcal{D}_{S_0}\) could be the conjunctions of the following sentences:
∀ \textit{x} . \textit{incoming}(\textit{x}, S_0) \supset \textit{x} = P_2 \lor \textit{x} = P_3,

∀ \textit{x} , \textit{y} . \neg \textit{paidTuit}(\textit{x}, \textit{y}, S_0),

∀ \textit{x} . \textit{preReq}(\textit{x}, CS_4) \equiv \textit{x} = CS_1 \lor \textit{x} = CS_3,

∀ \textit{x} . \textit{x} \neq CS_4 \supset \neg (\exists \textit{y} . \textit{prePeq}(\textit{y}, \textit{x}))

\textit{person}(P_1), \ldots, ; \textit{person}(P_m).

One can also imagine that some RBox axioms, for example,

\textit{gradeA}(\textit{x}, \textit{y}, \textit{s}) \supset \textit{hasGrade}(\textit{x}, \textit{y}, \textit{s}),

may be used for taxonomic reasoning in this domain.

Finally, we give an example of regression of an $\mathcal{L}^{\mathcal{C}^2}_{sc}$ regressable formula:

$\mathcal{R}^{\mathcal{C}^2}[(\exists \textit{x}).\textit{qualFull}(\textit{x}, \textit{do}([\textit{admit}(P_1), \textit{payTuit}(P_1, 6000)], S_0))]$

$= \mathcal{R}^{\mathcal{C}^2}[(\exists \textit{x}).\textit{eligFull}(\textit{x}, \textit{do}([\textit{admit}(P_1), \textit{payTuit}(P_1, 6000)], S_0)) \land$

$(\exists \geq 4 \textit{y}).\textit{enrolled}(\textit{x}, \textit{y}, \textit{do}([\textit{admit}(P_1), \textit{payTuit}(P_1, 6000)], S_0))]

$= \cdots$

$= (\exists \textit{x}).(\exists \geq 4 \textit{y}).\textit{enrolled}(\textit{x}, \textit{y}, S_0) \land ((\exists \textit{y}).\mathcal{R}^{\mathcal{C}^2}[y > 5000 \land$

$tuitPaid(\textit{x}, \textit{y}, \textit{do}([\textit{admit}(P_1), \textit{payTuit}(P_1, 6000)], S_0))])$

$= \cdots$

$= (\exists \textit{x}).(\exists \geq 4 \textit{y}).\textit{enrolled}(\textit{x}, \textit{y}, S_0) \land ((\exists \textit{y}).tuitPaid(\textit{x}, \textit{y}, S_0) \land$

$y > 5000 \lor (\textit{x} = P_1 \land y = 6000 \land y > 5000))$,

which is false given the above initial theory.

Suppose we denote the above BAT as $\mathcal{D}$. Given goal $G$, for example $\exists \textit{x} . \textit{qualFull}(\textit{x})$, and a composite Web service starting from the initial situation, for example,

$\textit{do}([\textit{admit}(P_1), \textit{payTuit}(P_1, 6000)], S_0)$ (we denote the corresponding resulting situation as $S_r$). We can check if the goal is satisfied after the execution of this composite Web service by solving the projection problem whether $\mathcal{D} \models G[S_r]$. In our example, this corresponds to solving whether $\mathcal{D} \models \exists \textit{x} . \textit{qualFull}(\textit{x}, S_r)$. We may also check if a given
(ground) composite Web service, i.e., a sequence of ground services \([A_1, A_2, \cdots, A_n]\), is possible to execute starting from the initial state by solving the executability problem whether \(D \models executable(do([A_1, A_2, \cdots, A_n], S_0))\). For example, we can check if the composite Web service \([\text{admit}(P_1), \text{payTuit}(P_1, 6000)]\) can possibly be executed from the starting state by solving whether \(D \models executable(S_r)\). Note that both entailment problems can be decided (not only for the query that we consider, but also for any query) because they can be reduced to the satisfiability problem in \(C^2\).

Here is another interesting example of the specification of Web services for online shopping.

**Example 5** Consider a Web service dynamical system in which clients are able to buy CDs and books online with credit cards. The system high-level features are specified as concepts and roles:

- Primitive static concepts: \(\text{person}(x)\) (\(x\) is a person), \(\text{cd}(x)\) (\(x\) is a CD), \(\text{book}(x)\) (\(x\) is a book), \(\text{creditCard}(x)\) (\(x\) is a credit card).
- Defined static concept: \(\text{client}(x)\) (\(x\) is a client).
- Primitive dynamic concept: \(\text{instore}(x, s)\) (\(x\) is in store in situation \(s\)).
- Defined dynamic concept: \(\text{valCust}(x, s)\) (\(x\) is a valuable customer in situation \(s\)).
- Static role: \(\text{has}(x, y)\) (\(x\) has \(y\)).
- Dynamic roles: \(\text{boughtCD}(x, y, s)\) (\(x\) bought CD \(y\) in situation \(s\)), \(\text{boughtBook}(x, y, s)\) (\(x\) bought book \(y\) in situation \(s\)), \(\text{bought}(x, y, s)\) (\(x\) bought \(y\) in situation \(s\)).

Web services are specified as actions: \(\text{buyCD}(x, y)\) (\(x\) buys CD \(y\)), \(\text{buyBook}(x, y)\) (\(x\) buys book \(y\)), \(\text{returnCD}(x, y)\) (\(x\) returns CD \(y\)), \(\text{returnBook}(x, y)\) (\(x\) returns book \(y\)), \(\text{order}(x)\) (the Web service agent orders \(x\) from the publisher).

The BAT of the system is as follows (most of the axioms are self-explanatory).

**Action Precondition Axioms:**
Chapter 3. Decidable Reasoning in a Modified Situation Calculus

\[ \text{Poss}(\text{buyCD}(x,y), s) \equiv \text{client}(x) \land \text{cd}(y) \land \text{instore}(y, s), \]
\[ \text{Poss}(\text{buyBook}(x,y), s) \equiv \text{client}(x) \land \text{book}(y) \land \text{instore}(y, s), \]
\[ \text{Poss}(\text{returnCD}(x,y), s) \equiv \text{boughtCD}(x,y,s), \]
\[ \text{Poss}(\text{returnBook}(x,y), s) \equiv \text{boughtBook}(x,y,s), \]
\[ \text{Poss}(\text{order}(x), s) \equiv \text{book}(x) \lor \text{cd}(x). \]

**Successor State Axioms:**

\[ \text{instore}(x, \text{do}(a,s)) \equiv \]
\[ (\exists y)(a = \text{returnCD}(y,x)) \lor (\exists y)(a = \text{returnBook}(y,x)) \lor a = \text{order}(x) \lor \]
\[ \text{instore}(x, s) \land \neg((\exists y)(a = \text{buyCD}(y,x)) \lor (\exists y)(a = \text{buyBook}(y,x))), \]
\[ \text{boughtCD}(x,y,s) \equiv a = \text{buyCD}(x,y) \lor \text{boughtCD}(x,y,s) \land a \neq \text{returnCD}(x,y), \]
\[ \text{boughtBook}(x,y,s) \equiv a = \text{buyBook}(x,y) \lor \text{boughtBook}(x,y,s) \land a \neq \text{returnBook}(x,y), \]
\[ \text{bought}(x,y,s) \equiv a = \text{buyCD}(x,y) \lor a = \text{buyBook}(x,y) \lor \text{bought}(x,y,s) \]
\[ \land \neg(\text{cd}(y) \land a = \text{returnCD}(x,y) \lor \text{book}(y) \land a = \text{returnBook}(x,y)). \]

**Acyclic TBox Axioms:**

Dynamic ones: \[ \text{valCust}(x,s) \equiv \text{person}(x) \land \exists y. (\text{bought}(x,y,s)). \]

Static ones: \[ \text{client}(x) \equiv \text{person}(x) \land (\exists y. (\text{has}(x,y) \land \text{CreditCard}(y))). \]

**RBox Axioms:**

\[ \text{boughtCD} \sqsubseteq \text{bought}, \quad \text{boughtBook} \sqsubseteq \text{bought}. \]

For any \( S = \text{do}([a_1, \cdots, a_n], S_0) \), let \( \text{executable}(S) \) be an abbreviation of the formula

\[ \text{Poss}(a_1, S_0) \land \bigvee_{i=2}^{n} \text{Poss}(a_i, \text{do}([a_1, \cdots, a_{i-1}], S_0)). \]

A sample initial theory \( \mathcal{D}_{S_0} \) can be the conjunction of the following sentences:
creditCard(Visa), creditCard(MasterCard), person(Tom), person(Sam),
cd(Classic), cd(U2), book(Sail), book(Twilight),
has(Tom, Visa) ∨ has(Tom, MasterCard), book(Java),
has(Sam, Visa) ∨ has(Sam, MasterCard),
∀x(instore(x, S₀) ⊕ x = Java).

We also provide some examples of \( L_{sc}^{C2} \) regressable formulas:

\[
\text{executable}(S₁), \ (\exists x)\text{valCust}(x, S₁), \ (\exists y)\text{instore}(y, S₂),
\]
\[
\text{executable}(S₂), \ \text{boughtCD}(Tom, U₂, S₁),
\]
where the ground situation terms \( S₁ \) and \( S₂ \) are given as follows.

\[ S₁ = \text{do}([\text{buyCD}(Tom, \text{Classic}), \text{buyBook}(Tom, \text{Twilight}), \text{buyBook}(Tom, \text{Sail}]), S₀) \]
\[ S₂ = \text{do}([\text{buyCD}(Sam, U₂), \text{returnCD}(Sam, U₂)], S₀). \]

Here is an example of the regression of \( \mathcal{R}^{C₂}[\exists x\text{valCust}(x, S₁)] \).

\[
\mathcal{R}^{C₂}[\exists x\text{valCust}(x, S₁)] = (\exists x)\mathcal{R}^{C₂}[\text{valCust}(x, S₁)]
\]
\[
= \ldots
\]
\[
= (\exists x)(\text{person}(x) \land \exists ≥ 3 y.\mathcal{R}^{C₂}[\text{bought}(x, y, S₁)])
\]
\[
= (\exists x)(\text{person}(x) \land \exists ≥ 3 y.\mathcal{R}^{C₂}[x = Tom \land y = Sail \lor
\text{bought}(x, y, \text{do}([\text{buyCD}(Tom, \text{Classic}), \text{buyBook}(Tom, \text{Twilight}]), S₀))])]
\]
\[
= (\exists x)(\text{person}(x) \land \exists ≥ 3 y. (x = Tom \land y = Sail \lor
\mathcal{R}^{C₂}[\text{bought}(x, y, \text{do}([\text{buyCD}(Tom, \text{Classic}), \text{buyBook}(Tom, \text{Twilight}]), S₀))])]
\]
\[
= \ldots
\]
\[
= (\exists x)(\text{person}(x) \land \exists ≥ 3 y. (x = Tom \land y = Sail \lor x = Tom \land y = Twilight \\
\lor x = Tom \land y = Classic \lor \text{bought}(x, y, S₀))),
\]

which is true with respect to the initial theory \( \mathcal{D}_{S₀} \) given above.
Here is another example of the regression of $(\exists y)\text{instore}(y, S_2)$:

\[
\mathcal{R}^{C^2}[(\exists y)\text{instore}(y, S_2)] \\
= (\exists y)\mathcal{R}^{C^2}[(\exists x)(x = \text{Sam}) \land y = U2 \lor \text{instore}(y, \text{do}(\text{buyCD}(\text{Sam}, U2), S_0))] \\
= (\exists y)(y = U2 \lor \mathcal{R}^{C^2}[\text{instore}(y, \text{do}(\text{buyCD}(\text{Sam}, U2), S_0)])] \\
= (\exists y)(y = U2 \lor \mathcal{R}^{C^2}[\text{instore}(y, S_0) \land \neg((\exists x)(x = \text{Sam}) \land y = U2)]) \\
= (\exists y)(y = U2 \lor \text{instore}(y, S_0) \land y \neq U2),
\]

which is true with respect to the initial theory $\mathcal{D}_{S_0}$ given above. Note that, in the regression above, the dual formula of the RHS of the SSA of $\text{instore}(x, \text{do}(a, s))$ is

\[(\exists x)(a = \text{returnCD}(x, y)) \lor (\exists x)(a = \text{returnBook}(x, y)) \lor a = \text{order}(y) \\
\text{instore}(y, s) \land \neg((\exists x)(a = \text{buyCD}(x, y)) \lor (\exists x)(a = \text{buyBook}(x, y))).\]

### 3.3.4 A Description-Logic Based Situation Calculus

We see from Th. 6 that the computational complexity of solving the projection problems in $\mathcal{L}_{sc}^{C^2}$ (using the regression) is quite high. On the other hand, with context-free SSAs, although we can gain better complexity (see Th. 7), the expressive power of context-free SSAs is quite limited. Motivated by the observation that some DL languages have better computational complexity for concept satisfiability problems and/or ABox consistency problems than $C^2$ (see Section 2.2.2) and the idea of restricting the context conditions in the SSAs similar to context-free SSAs, we now consider another type of restriction on BATs in the language of $\mathcal{L}_{sc}^{C^2}$. We would like to get better complexity results than that of Th. 6 when solving the projection problem in general. At the same time, we consider a fragment that is more expressive than context-free SSAs. Moreover, we will see that this language has natural connections with DLs.

We first consider a sub-language of $C^2$, denoted $FO_{DL}$.

**Definition 9** The language of $FO_{DL}$ includes constants, monadic and dyadic predicates.
Moreover, it is a union of two sub-languages: $\text{FO}_{DL} = \text{FO}^x_{DL} \cup \text{FO}^y_{DL}$, where the detailed definition of $\text{FO}^x_{DL}$ is provided below and $\text{FO}^y_{DL}$ is obtained by renaming every $x$ with $y$ and every $y$ with $x$ for every formula in $\text{FO}^x_{DL}$. The set $\text{FO}^x_{DL}$ is a minimal set of formulas built inductively as follows:

1. true and false are in $\text{FO}^x_{DL}$.
2. If $AC$ is a monadic predicate name, then $AC(x)$ is in $\text{FO}^x_{DL}$.
3. If $b$ is a constant, then $x = b$ is in $\text{FO}^x_{DL}$.
4. If $\phi$ is in $\text{FO}^x_{DL}$, then $\neg \phi$ is in $\text{FO}^x_{DL}$.
5. If $\phi$ and $\psi$ are in $\text{FO}^x_{DL}$, then $\phi \land \psi$ and $\phi \lor \psi$ are in $\text{FO}^x_{DL}$.
6. If $\phi(x)$ is in $\text{FO}^x_{DL}$, and $\phi(x)$ has at most one free variable $x$, and $R$ is a dyadic predicate name, $\sim \phi(y)$ is the dual formula of $\phi(x)$, obtained by renaming every $x$ (both free and bound) with $y$ and every $y$ (both free and bound) with $x$ in $\phi$, then $\exists y. R(x, y) \land \sim \phi(y)$ and $\forall y. R(x, y) \supset \sim \phi(y)$ are in $\text{FO}^x_{DL}$.
7. If $\phi$ is in $\text{FO}^x_{DL}$, $\sim \phi$ is the dual formula of $\phi$, obtained by renaming every $x$ (both free and bound) with $y$ and every $y$ (both free and bound) with $x$ in $\phi$, then $[\exists y. ]\sim \phi(y)$ and $[\forall y. ]\sim \phi(y)$ are in $\text{FO}^x_{DL}$, where $[\exists y. ]$ ([\forall y. ]$, respectively$) means that if $\sim \phi$ has a free variable $y$, then it is quantified by $\exists y$ ($\forall y$, respectively); otherwise, there is no need to add the quantifier.

□

The semantics of $\text{FO}_{DL}$ are the same as the usual semantics of $\text{FO}^2$. Notice that for any $\phi \in \text{FO}^x_{DL}$ ($\phi \in \text{FO}^y_{DL}$, respectively), $\sim \phi$ is in $\text{FO}^y_{DL}$ ($\text{FO}^x_{DL}$, respectively). Moreover, it is easy to see that any sentence (i.e., closed formula) in $\text{FO}_{DL}$ is in both $\text{FO}^x_{DL}$ and $\text{FO}^y_{DL}$. We then are able to prove the following lemma (the detailed proof is provided in Appendix B.2).
Lemma 1  There are syntactic translations between $FO_{DL}$ and the DL language $\mathit{ALCO}(U)$, i.e., they are equally expressive. Moreover, such translations lead to no more than a linear increase in the size of the translated formula.

Recall from the review of DLs in Section 2.2.1 that the satisfiability problem of a concept and/or the consistency problem of an ABox in the DL language $\mathit{ALCO}(U)$ can be solved in $\text{ExpTime}$. This is an improvement over $C^2$ and $FO^2$ (see Section 2.2.2). For this reason we would like to investigate a fragment of $L_{sc}^{C^2}$ based on $FO_{DL}$.

Definition 10  We say that the SSA for a fluent $F$ is $\mathit{ALCO}(U)$-restricted if the SSA of $F$ has the form of Eq. (3.2), where each context condition $\psi^+_i$ (or $\psi^-_j$, respectively) is a formula in $FO_{DL}$ when all situation variables are suppressed. Moreover, we say that the set of SSAs $D_{ss}$ in a BAT $D$ is $\mathit{ALCO}(U)$-restricted if every axiom of a primitive dynamic concept in $D_{ss}$ is $\mathit{ALCO}(U)$-restricted and every axiom of a dynamic role in $D_{ss}$ is both $\mathit{ALCO}(U)$-restricted and context-free.

We say that a concept definition of the form Eq. (3.3) for any defined concept $G$ (including static or dynamic) is $\mathit{ALCO}(U)$-restricted if the formula $\phi_G(x)$ on the RHS of Eq. (3.3) is in $FO_{DL}$. Moreover, we say that the acyclic TBox $D_T$ of a BAT $D$ is $\mathit{ALCO}(U)$-restricted if every axiom in the set is $\mathit{ALCO}(U)$-restricted.

An initial theory $D_{S_0}$ is $\mathit{ALCO}(U)$-restricted if every sentence in it is in $FO_{DL}$ with the initial situation $S_0$ suppressed if there is any.

For any $L_{sc}^{C^2}$ regressable sentence $W$, whose situation terms are all ground, we say that $W$ is $\mathit{ALCO}(U)$-restricted if $W$ is in $FO_{DL}$ with all situation terms suppressed. $\square$

We can then prove the following lemma (its proof is provided in Appendix B.3).

Lemma 2  Consider a BAT $D$ in $L_{sc}^{C^2}$ whose $D_{ss}$ and $D_T$ are $\mathit{ALCO}(U)$-restricted. Let $W$ be an $L_{sc}^{C^2}$ regressable formula that is uniform in a ground situation $S$ and has no appearance of Poss. Let $n = \text{sitLength}(S)$ and $m = \text{size}(W)$. If $W$ is $\mathit{ALCO}(U)$-restricted, then there is a $\Phi_W$ in $FO_{DL}$ such that $\mathcal{R}^{C^2}[W]$ is equivalent to $\Phi_W[S_0]$. It
takes no more than $c \cdot n \cdot \text{size}(\Phi_W)$ steps of logical transformation as introduced in the proof from $R^{C^2}[W]$ (with $S_0$ suppressed) to find such $\Phi_W$, where $c$ is a positive integer. Moreover, $\text{size}(\Phi_W)$ is in $O(2^{hn+3k^2n^2})$ for some positive integer $h$. That is, the size of $\Phi_W$ is no more than exponential in the size of $W$.

Then we have the following complexity result.

**Theorem 8** Consider a BAT $\mathcal{D}$ in $L^{C^2}_{sc}$ whose $D_{ss}$, $D_T$ and $D_{S_0}$ are $\mathcal{ALCO}(U)$-restricted. Let $W$ be any $L^{C^2}_{sc}$ regressable sentence in $\mathcal{D}$ that is uniform in a ground situation $S$ and has no appearance of Poss. If $W$ is $\mathcal{ALCO}(U)$-restricted, then answering the query whether $\mathcal{D} \models W$ via regression can be solved in $2\text{-ExpTime}$ with respect to the size of $W$ when $\mathcal{D}$ is fixed.

**Proof:** First, $\mathcal{D} \models W$ iff $D_{S_0} \models R^{C^2}[W]$ by Th. 4. Also, by Lemma 2, we can find a formula $\Phi$ in $FO_{DL}$ in no more than exponential time with respect to the size of $W$, such that $\models \Phi[S_0] \equiv R^{C^2}[W]$, and the size of $\Phi$ is no more than exponential in the size of $W$. Hence, $D_{S_0} \models R^{C^2}[W]$ iff $D_{S_0} \models \Phi[S_0]$. It is the same as answering whether $D_{S_0} \land \neg \Phi[S_0]$ is unsatisfiable or not, which is a complement problem of whether $D_{S_0} \land \neg \Phi[S_0]$ is satisfiable or not. Let $\Psi$ be the formula $D_{S_0} \land \neg \Phi[S_0]$ with situation term $S_0$ suppressed, and it is easy to see that $\Psi$ is in $FO_{DL}$, because $D_{S_0}$ is in $FO_{DL}$ when the situation term $S_0$ is suppressed and $\Phi$ is in $FO_{DL}$. $\Psi$ has the same size as $D_{S_0} \land \neg \Phi[S_0]$, and it is unsatisfiable iff $D_{S_0} \land \neg \Phi[S_0]$ unsatisfiable. Using the syntactic translation function $\pi$ defined in the proof of Lemma 1 (see Appendix B.2), $\pi(\Psi)$ is a concept in DL language $\mathcal{ALCO}(U)$. To decide whether a concept $\pi(\Psi)$ is satisfiable in $\mathcal{ALCO}(U)$ is in $\text{ExpTime}$ with respect to the size of $\pi(\Phi)$, which is linear in the size of $\Phi$. Hence, deciding whether $D_{S_0} \models R^{C^2}[W]$ is in $\text{co-ExpTime}$ with respect to the size of $D_{S_0} \land \neg \Phi[S_0]$. However, when $\mathcal{D}$ is given the size of $D_{S_0}$ is fixed, hence, deciding whether $D_{S_0} \models R^{C^2}[W]$ is in fact in $\text{co-ExpTime}$ with respect to the size of $\Phi[S_0]$ (which is the same as the size of $\Phi$). Again, because the size of $\Phi$ is exponential in the size of $W$,
deciding whether $D_S \models R^{C^2}[W]$ is in the complement of 2-ExpTime (with respect to the size of $W$), which is the same as 2-ExpTime.

Recall from the review of DLs in Section 2.2.1 that the satisfiability problem of a concept and/or the consistency problem of an ABox in the DL language $\mathcal{ALCQO}(U)$ can also be solved in ExpTime. For this reason, similar to the development above, we can extend $FO_{DL}$ to a sub-language of $C^2$, say $FO_{DL^+}$, by adding counting quantifiers to $FO_{DL}$.

**Definition 11** $FO_{DL^+} = FO_{DL^+}^x \cup FO_{DL^+}^y$, where $FO_{DL^+}^x$ is a minimal set of formulas built inductively below, and $FO_{DL^+}^y = \{ \tilde{\phi} \mid \phi \in FO_{DL^+}^x \}$.

- $true$ and $false$ are in $FO_{DL^+}^x$.
- If $AC$ is a monadic predicate name, then $AC(x)$ is in $FO_{DL^+}^x$.
- If $b$ is a constant, then $x = b$ is in $FO_{DL^+}^x$.
- If $\phi$ is in $FO_{DL^+}^x$, then $\neg \phi$ is in $FO_{DL^+}^x$.
- If $\phi$ and $\psi$ are in $FO_{DL^+}^x$, then $\phi \land \psi$ and $\phi \lor \psi$ are in $FO_{DL^+}^x$.
- If $\phi(x)$ is in $FO_{DL^+}^x$, and $\phi(x)$ has at most one free variable $x$, and $R$ is a dyadic predicate name, $\tilde{\phi}(y)$ is the dual formula of $\phi(x)$, obtained by renaming every $x$ (both free and bound) with $y$ and every $y$ (both free and bound) with $x$ in $\phi$, then $\exists y.R(x, y) \land \tilde{\phi}(y), \forall y.R(x, y) \supset \tilde{\phi}(y), \exists^\geq n.y.R(x, y) \land \tilde{\phi}(y)$ and $\exists^\leq n.y.R(x, y) \land \tilde{\phi}(y)$ for any $n \in \mathbb{N}$ are in $FO_{DL^+}^x$.
- If $\phi$ is in $FO_{DL^+}^x$, $\tilde{\phi}$ is dual formula of $\phi$, obtained by renaming every $x$ (both free and bound) with $y$ and every $y$ (both free and bound) with $x$ in $\phi$, then $[\exists y.]\tilde{\phi}(y), [\forall y.]\tilde{\phi}(y), [\exists^\geq n.y.]\tilde{\phi}(y)$ and $[\exists^\leq n.y.]\tilde{\phi}(y)$ for any $n \in \mathbb{N}$ are in $FO_{DL^+}^x$, where $[\exists y.]$ ([$\forall y.$], $[\exists^\geq n.y.]$, or $[\exists^\leq n.y.]$, respectively) means that if $\tilde{\phi}$ has a free variable $y$, then
it is quantified by $\exists y (\forall y, \exists^\geq y, \text{or} \exists^\leq y$, respectively); otherwise, there is no need to add such quantifier.

\[ \square \]

The semantics of $FO_{DL^+}$ is the same as the usual semantics of $C^2$. Similar to Lemma 1, we are able to prove the following lemma.\(^5\)

**Lemma 3** There are syntactic translations between $FO_{DL^+}$ and the DL language $\mathcal{ALCQO}(U)$, i.e., they are equally expressive. Moreover, such translations lead to no more than a linear increase in the size of the translated formula.

Similarly, we say that the SSA for a fluent $F$ is $\mathcal{ALCQO}(U)$-restricted if the SSA of $F$ has the form of Eq. (3.2), where each context condition $\psi_i^+$ (or $\psi_i^-$, respectively) is a formula in $FO_{DL^+}$ when all situation variables are suppressed. Moreover, we say that the set of SSAs $D_{ss}$ in a BAT is $\mathcal{ALCQO}(U)$-restricted if every axiom of a primitive dynamic concept in $D_{ss}$ is $\mathcal{ALCQO}(U)$-restricted and every axiom of a dynamic role in $D_{ss}$ is both $\mathcal{ALCQO}(U)$-restricted and context-free. We say that a concept definition of the form Eq. (3.3) for any defined concept $G$ (including static or dynamic) is $\mathcal{ALCQO}(U)$-restricted if the formula $\phi_G(x)$ on the RHS of Eq. (3.3) is in $FO_{DL^+}$. Moreover, we say that the acyclic TBox $D_T$ of a BAT $D$ is $\mathcal{ALCQO}(U)$-restricted.

Similar to Lemma 2, we can also prove a lemma for $\mathcal{ALCQO}(U)$-restricted regressable formulas as follows.

**Lemma 4** Consider a BAT $D$ in $\mathcal{L}_{sc}^{C^2}$ whose $D_{ss}$ and $D_T$ are $\mathcal{ALCQO}(U)$-restricted. Let $W$ be an $\mathcal{L}_{sc}^{C^2}$ regressable formula that is uniform in a ground situation $S$ and has no appearance of Poss. Let $n = \text{sitLength}(S)$ and $m = \text{size}(W)$. Then, if $W$ is $\mathcal{ALCQO}(U)$-restricted, i.e., it is in $FO_{DL^+}$ with the situation term $S$ suppressed, there is a $\Phi_W$ in $FO_{DL^+}$ such that $\mathcal{R}^{C^2}[W]$ is equivalent to $\Phi_W[S_0]$. It takes no more than $c \cdot n \cdot \text{size}(\Phi_W)$

\(^5\)The proof of Lemma 3 is exactly the same as the proof of Lemma 1 except that we only need to add translation for counting quantifiers, which is straightforward. Hence, the proof of Lemma 3 is omitted.
steps of logical transformation as introduced in the proof of Lemma 2 from $R^{C_2}[W]$ (with $S_0$ suppressed) to find such $\Phi_W$, where $c$ is a positive integer. Moreover, $\text{size}(\Phi_W)$ is in $O(2^{hmn+3h^2n^2})$ for some positive integer $h$. That is, the size of $\Phi_W$ is no more than exponential in the size of $W$.

**Proof:** The proof is exactly the same as the proof of Lemma 2 (see Appendix B.3), where $\text{FO}_{DL}$ ($\text{FO}^x_{DL}$, or $\text{FO}^y_{DL}$, respectively) is replaced by $\text{FO}_{DL+}$ ($\text{FO}^x_{DL+}$, or $\text{FO}^y_{DL+}$, respectively), and we need to consider sub-cases constructed using the counting quantifiers $\exists \geq n$ and $\exists \leq n$ in the proof. But the proof for cases that are constructed using $\exists \geq n$ and $\exists \leq n$ are the same as the proof for the case using the $\exists$-quantifier (see case (d) of the inductive step of the nested induction on the structure of the regressable formula $W$ in the proof of Lemma 2, Appendix B.3). Hence, the details are omitted here. □

As a consequence, similar to the proof of Th. 8, we also have the following result.

**Theorem 9** Consider a BAT $D$ in $L_{sc}^{C_2}$ whose $D_{ss}$, $D_T$ and $D_{S_0}$ are $\mathcal{ALCO}(U)$-restricted. Let $W$ be any $L_{sc}^{C_2}$ regressable sentence in $D$ that is uniform in a ground situation $S$ and has no appearance of Poss. If $W$ is $\mathcal{ALCO}(U)$-restricted, then answering the query whether $\models D W$ via regression can be solved in 2-ExpTime with respect to the size of $W$ when $D$ is fixed.

According to Th. 8 (or Th. 9, respectively), given a BAT $D$ in $L_{sc}^{C_2}$ whose $D_{ss}$, $D_T$ and $D_{S_0}$ are $\mathcal{ALCO}(U)$-restricted (or $\mathcal{ALCO}(U)$-restricted, respectively), the project problem can be solved in 2-ExpTime via regression if the query $\mathcal{ALCO}(U)$-restricted (or $\mathcal{ALCO}(U)$-restricted, respectively). If, in addition, the RHS of each precondition axiom in $D$ (with the situation term suppressed) is a formula in $\text{FO}_{DL}$ (or $\text{FO}_{DL+}$, respectively) with the situation term $s$ suppressed when the occurrences of all the action arguments are substituted by constants, then the executability problem can also be solved in 2-ExpTime via regression.
3.4 Reasoning about Actions without Regression

In the previous section, we studied the computational upper bound of using regression to solve the projection problem and the executability problem. We are also interested in what the exact computational complexity of projection and executability problems is in general. In [7], an extended version of [8], Baader et al. have studied this problem within a different framework of integrating description logics and actions. In their framework, all actions are ground, and the preconditions and effects of actions are specified for each action individually using DL ABoxes in the sublanguages of $\mathcal{ALCQIO}$ (including itself). Inspired by the approach in [7], we also study this problem and prove a similar result in our framework of $\mathcal{L}_{sc}^{C^2}$ (see Corollary 2 below). As to the difference between our work and [7], we will provide further detailed discussion in Section 3.6.

Now, we first prove the following theorem.

**Theorem 10** Consider any regressable sentence $W$ that is uniform in a ground situation $do([\alpha_1, \cdots, \alpha_n], S_0)$ in $\mathcal{L}_{sc}^{C^2}$ with a $\mathcal{L}_{sc}^{C^2}$-restricted BAT. Then, the projection problem of deciding whether $\mathcal{D} \models W$ can be polynomially reduced to a problem deciding whether $\mathcal{D}' \models W'$ for a theory $\mathcal{D}'$ and a sentence $W'$ in $C^2$.

**Proof:**

Consider an $\mathcal{L}_{sc}^{C^2}$ regressable sentence $W$ that is uniform in a ground $do([\alpha_1, \cdots, \alpha_n], S_0)$ with a background BAT in $\mathcal{L}_{sc}^{C^2}$. Without loss of generality, we can assume that there no predicate $\text{Poss}$ appeared in $W$ (Otherwise, we can use one step $\mathcal{L}_{sc}^{C^2}$ regression operator to replace those atoms of the form $\text{Poss}(\alpha, \sigma)$ with the right-hand side of the precondition axioms, causing no more than linear increase in the size of the formula).

For every fluent $F(\bar{x}, s)$ mentioned in $\mathcal{D}$ and every sub-situation $S_i = do([\alpha_1, \cdots, \alpha_i], S_0)$ of the ground situation $do([\alpha_1, \cdots, \alpha_n], S_0)$ that occurs in the query $W$, we introduce situation-free predicates $F^i(\bar{x})$ into $\mathcal{D}'$ for $i = 0..n$. Intuitively, for any $\bar{x}$ and for every sub-situation $S_i$, $F^i(\bar{x})$ is true iff $F(\bar{x}, S_i)$ is true, i.e., $F^i(\bar{x})$ is an $i$th copy of the fluent
For each fluent we introduce \( n \) copies. For each \( i = 0..n \), we also recursively define a translation function \( g^i \) for any \( \mathcal{L}_{sc}^C \) formula that has no quantifiers over any variables mentioned in situation terms.

1. Let \( a \) be an action variable and \( A \) be a \( k \)-ary action function. For each atom of the form \( a = A(t_1, \ldots, t_k) \), \( g^n(a = A(t_1, \ldots, t_k)) \) is not defined. For values \( i < n \), the result of translation \( g^i(a = A(\vec{t})) \) depends on whether the \((i+1)\)st action mentioned in \( do([\alpha_1, \ldots, \alpha_n], S_0) \) coincides or is distinct from the action \( A \):

\[
g^i(a = A(\vec{t})) \overset{\text{def}}{=} \begin{cases} \text{false} & \text{if } \alpha_{i+1} = B(\vec{t'}) \text{ for some } B \neq A \\ \text{true} & \text{if } \alpha_{i+1} = A \text{ and } k = 0 \\ \bigwedge_{j=1}^k t'_j = t_j & \text{if } \alpha_{i+1} = A(t'_1, \ldots, t'_k) \end{cases}
\]

The intention of this definition is as follows. For each SSA, \( F(\vec{x}, do(a, s)) \equiv \phi_F(\vec{x}, s) \), an axiom \( F^{i+1}(\vec{x}) \equiv g^i(\phi_F(\vec{x})) \) will be added using the translation function \( g^i \) so that \( g^i(\phi_F(\vec{x})) \) represents the equivalent condition of \( F^{i+1}(\vec{x}) \) being true, i.e., \( F(\vec{x}) \) is true after \( \alpha_{i+1} \) is executed in the situation \( S_i \), but \( g^i(\phi_F(\vec{x})) \) should achieve this purpose without mentioning any action or situation terms. Hence, the definition of \( g^i(a = A(t_1, \ldots t_k)) \) is in fact the result of substituting variable \( a \) with action \( \alpha_{i+1} \) and then removing action terms using the unique name axioms for actions.

2. For each situation-independent atom \( P(\vec{t}) \), \( g^i(P(\vec{t})) \overset{\text{def}}{=} P(\vec{t}) \).

3. For each atomic fluent \( F(\vec{t}, \sigma) \), \( g^i(F(\vec{t}, \sigma)) \overset{\text{def}}{=} F^i(\vec{t}) \), where \( \sigma \) is a term of sort Situation.

4. For each non-atomic formula, we have

\[
\begin{align*}
g^i(\neg W) & \overset{\text{def}}{=} \neg g^i(W), \\
g^i(W_1 \circ W_2) & \overset{\text{def}}{=} g^i(W_1) \circ g^i(W_2) \text{ for any operator } \circ \in \{\land, \lor, \exists, \equiv\}, \\
g^i(\forall v.W) & \overset{\text{def}}{=} \forall v.g^i(W) \text{ for any quantifier } \forall \in \{\exists, \forall \exists^m, \exists^m | m \in \mathbb{N}\}.
\end{align*}
\]
Now, let $W' = g^n(W)$ and $\mathcal{D}' = \mathcal{D}'_{S_0} \cup \mathcal{D}'_T \cup \mathcal{D}'_{ss}$, where

$$\mathcal{D}'_{S_0} = \{g^0(\phi) \mid \phi \in \mathcal{D}_{S_0}\},$$

$$\mathcal{D}'_T = \bigcup_{j=0}^{n} \mathcal{D}'_T^j \text{ such that for each } 0 \leq j \leq n,$$

$$\mathcal{D}'_T^j = \{g^j(\phi) \mid \phi \in \mathcal{D}_T\},$$

and $\mathcal{D}'_{ss} = \bigcup_{i=0}^{n-1} \mathcal{D}'_{ss}^i$ such that for each $0 \leq i < n$,

$$\mathcal{D}'_{ss}^i = \{((\forall \vec{x}).F^{i+1}(\vec{x}) \equiv g^i(\Phi_F) \mid \text{Axiom } F(\vec{x}, do(a, s)) \equiv \Phi_F \text{ is in } \mathcal{D}_{ss}\}.$$

It is easy to see that all of the axioms in $\mathcal{D}'$ are $C^2$ formulas and so is $W'$.

Next, we prove that $\mathcal{D} \models W$ iff $\mathcal{D}' \models W'$. The general idea is that as follows (see Fig. 3.1). First, we define a reasoning mechanism, say $\mathcal{K}$, similar to $\mathcal{R}^{C^2}$ using the axioms in $\mathcal{D}'_{ss} \cup \mathcal{D}'_T$, i.e., for any atom $P(\vec{t})$ that $P$ appears on the left-hand side of any axiom in $\mathcal{D}'_{ss} \cup \mathcal{D}'_T$, it will be replaced by the formula on the right-hand side with proper treatment of variables (Step 1 in Fig. 3.1). Second, we prove that $\mathcal{D}' \models W'$ iff $\mathcal{D}'_{S_0} \models \mathcal{K}[W']$. Finally, we prove that $\mathcal{D}_{S_0} \models \mathcal{R}^{C^2}[W]$ iff $\mathcal{D}'_{S_0} \models \mathcal{K}[W']$ (Step 2 in Fig. 3.1).

\[\begin{align*}
\mathcal{D} \models W & \quad \text{(Th. 4)} \quad \mathcal{D}_{S_0} \models \mathcal{R}^{C^2}[W] \\
\quad \uparrow \quad \text{(Step 2)} & \\
\mathcal{D}' \models W' & \quad \text{(Step 1)} \quad \mathcal{D}'_{S_0} \models \mathcal{K}[W']
\end{align*}\]

Figure 3.1: Diagram of the Outline for Proving Th. 10

---

Step 1. We first define the reasoning mechanism $\mathcal{K}$ recursively for a $C^2$ formula $W'$ as follows.

- If $W'$ is not atomic, i.e., $W'$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W_1$, or $Qv.W_1$ where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol, then
\[ \mathcal{K}[W_1 \lor W_2] = \mathcal{K}[W_1] \lor \mathcal{K}[W_2], \quad \mathcal{K}[\neg W_1] = \neg \mathcal{K}[W_1], \]
\[ \mathcal{K}[W_1 \land W_2] = \mathcal{K}[W_1] \land \mathcal{K}[W_2], \quad \mathcal{K}[\forall v.W_1] = \forall v. \mathcal{K}[W_1]. \]

- Otherwise, \( W' \) is an atom. There are several cases.

a. If \( W' \) has the form \( G(t) \) for some predicate \( G \) and some object term \( t \), and there is axiom \( G(x) \equiv \phi_G(x) \) in \( \mathcal{D}' \) for some \( C^2 \) formula \( \phi_G(x) \). Because of the restrictions of the language \( \mathcal{L}^C_{sc} \), term \( t \) can only be a variable \( x, y \) or a constant. Then, \( \mathcal{K} \) is defined as follows:

\[
\mathcal{K}[W] = \begin{cases} 
\mathcal{K}[\phi_G(t)] & \text{if } t \in \{O, x\}, \\
\mathcal{K}[\tilde{\phi}_G(y)] & \text{otherwise, i.e., if } t = y.
\end{cases}
\]

b. If \( W' \) is of the form \( F(t_1) \) (or \( F(t_1, t_2) \), respectively) \( (1 \leq i \leq n) \) for some predicate \( F \) and some terms \( t_1 \) (and \( t_2 \)) of sort \( Object \), and there is an axiom for \( F \) of the form \( F(\bar{x}) \equiv \Phi_F(\bar{x}) \) in \( \mathcal{D}'_{ss} \). Because of the restriction of the language \( \mathcal{L}^C_{sc} \), the terms \( t_1 \) and \( t_2 \) can only be a variable \( x, y \) or some constant \( O \). Hence, when \( W' \) is of the form \( F(t_1) \),

\[
\mathcal{K}[W'] = \begin{cases} 
\mathcal{K}[\Phi_F(t_1)] & \text{if } t_1 \in \{O, x\}, \\
\mathcal{K}[\tilde{\Phi}_F(y)] & \text{otherwise, i.e., if } t_1 = y;
\end{cases}
\]

and, when \( W' \) is of the form \( F(t_1, t_2) \),

\[
\mathcal{K}[W'] = \begin{cases} 
\mathcal{K}[(\exists y. x = y \land \Phi_F(x, y))] & \text{if } \langle t_1, t_2 \rangle = \langle x, x \rangle; \\
\mathcal{K}[(\exists x. y = x \land \Phi_F(x, y))] & \text{if } \langle t_1, t_2 \rangle = \langle y, y \rangle; \\
\mathcal{K}[\tilde{\Phi}_F(y, t_2)] & \text{if } \langle t_1, t_2 \rangle \in \{ \langle y, x \rangle, \langle y, O \rangle, \langle O, x \rangle \}; \\
\mathcal{K}[\Phi_F(t_1, t_2)] & \text{otherwise, i.e, if } \langle t_1, t_2 \rangle \in \\
\{ \langle x, y \rangle, \langle x, O \rangle, \langle O, y \rangle, \langle O, O \rangle \};
\end{cases}
\]

\[
\mathcal{K}[W'] = W'.
\]

c. If \( W' \) is any other atom other than the atom considered in the above cases (including equality between object terms), then

\[
\mathcal{K}[W'] = W'.
\]
Similar to the proof of Th. 4, it is easy to prove that $D' \models W' \iff D_0^0 \models \mathcal{K}[W']$ for $W' = g^n(W)$ and the theory $D'$ defined as above.

Step 2. Let $\mathcal{F} = \{ F(\bar{x}, S_0) \equiv F^0(\bar{x}) \mid \text{for every fluent in } D \}$. Then, it is obvious to see that $\mathcal{F} \models \Phi_0 \equiv \Phi'_0$ (respectively) is the conjunction of all the axioms in $D_{S_0}$ ($D_{S_0}^0$, respectively) according to how $D_{S_0}^0$ is defined. Moreover, it is easy to prove that $\mathcal{F} \models R_{C^2}[W] \equiv \mathcal{K}[W']$ by using structural induction principle. Hence, $D_{S_0} \models R_{C^2}[W] \iff D_{S_0}^0 \models \mathcal{K}[W']$. □

Then, we have the following corollary.

**Corollary 2** Executability and projection are co-NExpTime-complete for $C^2$ regressable sentences with respect to a given $L_{C^2}^{sc}$-restricted BAT.

**Proof:** For any $C^2$ sentence $\phi$, we can construct a special $L_{C^2}^{sc}$-restricted BAT $D$, where $D$ consists of only one axiom $true$ and a $C^2$ regressable sentence $\neg \phi$, then check whether $\phi$ is unsatisfiable iff $D \models \neg \phi$. I.e., the unsatisfiability problem in $C^2$ can be reduced to a projection problem in $L_{C^2}^{sc}$ in linear time. Moreover, based on Th. 10 and the complexity of solving satisfiability problems in $C^2$, it is easy to see that the statement above holds. □

According to Th. 10 and the fact that $\mathcal{ACL}(U)$-restricted ($\mathcal{ACLQ}(U)$-restricted, respectively) BATs and regressable sentences in $FO_{DL}$ ($FO_{DL^+}$, respectively) are special cases of $L_{C^2}^{sc}$-restricted BATs and $C^2$ regressable sentences, we also have the following corollary.

**Corollary 3** Consider a BAT $D$ in $L_{C^2}^{sc}$ whose $D_{ss}$ and $D_T$ are $\mathcal{ACL}(U)$-restricted ($\mathcal{ACLQ}(U)$-restricted, respectively). Let $D_{S_0}$, with the situation term $S_0$ suppressed, be in $FO_{DL}$ ($FO_{DL^+}$, respectively). Let $W$ be any $L_{C^2}^{sc}$ regressable sentence in $D$ that is uniform in a ground situation $S$ and has no appearance of Poss. If $W$, with the situation term $S$ suppressed, is in $FO_{DL}$ ($FO_{DL^+}$, respectively), then answering the query whether
\( \mathcal{D} \models W \) without regression can be solved co-NExpTime with respect to the size of \( W \) when \( \mathcal{D} \) is fixed.

Note that according to the approach of constructing \( \mathcal{D}' \) in the proof of Th. 10, it is easy to see that the corresponding \( \mathcal{D}' \) of an \( \mathbf{ALCQO}(U) \)-restricted (\( \mathbf{ALCQQO}(U) \)-restricted, respectively) BAT can only be considered as a theory in \( C^2 \), but not a theory in \( FO_{DL} \) (\( FO_{DL^+} \), respectively). This is because in case of the SSAs of roles, the constructed \( \mathcal{D}' \) may include axioms of the form \( R^i(x, y) \equiv R^{i-1}(x, y) \land \phi(x, y) \) for some binary predicates \( R^i \) and \( R^{i-1} \) and some \( C^2 \) formula \( \phi(x, y) \),\(^6\) which can not be equivalent to any formula in \( FO_{DL} \) (\( FO_{DL^+} \), respectively).

In Section 3.3.2, we showed that the complexity upper-bound for projection problem in an \( \mathbf{ALCQO}(U) \)-restricted (\( \mathbf{ALCQQO}(U) \)-restricted, respectively) situation calculus is of 2-ExpTime with regression. However, the relationship between 2-ExpTime and co-NExpTime is still an open problem. Moreover, looking for the lower bound for these two classes of problems also remains open for future study.

Hence, according to the above result, for the executability and projection problems in \( \mathcal{L}_{sc}^{C^2} \), we may reduce them to the (un)satisfiability problems in \( C^2 \) directly. Although the complexity upper-bound of solving executability and projection problems for \( C^2 \)-restricted BATs and queries via regression is higher than without regression, it is still necessary to study regression in \( \mathcal{L}_{sc}^{C^2} \). First, this is because regression is a natural reasoning mechanism in the situation calculus, and it would be unnatural not to study whether or not regression works in \( \mathcal{L}_{sc}^{C^2} \), and in the previous sections we show that the regression technique still works for reasoning about projection problems and executability problems in \( \mathcal{L}_{sc}^{C^2} \), where all situation terms are ground in the queries (see Section 3.3.1). Second, reasoning via regression for solving executability and projection problems is not always as bad as its complexity upper-bound and we can have better empirical computational behavior than the worst-case scenario when using regression for some restricted domains.

\(^6\) \( \phi(x, y) \) can be omitted when it is equivalent to true.
(see the end of Section 3.3.2 and Section 3.3.4). Similarly, in the Semantic Web community, the satisfiability problem in the description logic $\mathcal{SROIQ}$, the underlying logic of
the recent Web Ontology Language $\mathsf{OWL~2}$, has very high complexity ($2$-$\mathsf{NExpTime}$-complete) [94]. But nevertheless, existing $\mathsf{OWL~2}$ reasoners (e.g., HermiT [125] from Oxford) handle large practical ontologies without problems [32]. Finally, and most importantly, the approach presented in the proof of Th. 10 only fits for regressable queries with ground situation terms, while the regression technique in Reiter’s situation calculus works for more general queries where situation terms are not necessarily to be ground.
We believe that it is possible to extend our regression operator for more general reasoning problems at least for some restricted BATs in $\mathcal{L}_{sc}^{C^2}$. For example, in the example of school enrolment, it is possible to use our modified regression to reason about some queries with non-ground situation terms, such as $\exists x. \text{incoming}(x, \text{do}(\text{admit}(x), S_0))$ and $\exists x. \forall y. \text{incoming}(x, \text{do}(\text{drop}(x, y), S_0))$, although currently our regression operator in $\mathcal{L}_{sc}^{C^2}$ is only defined for regressable formulas with ground situation terms. Hence, one of our future research topics is to figure out for what kind of BATs in $\mathcal{L}_{sc}^{C^2}$, it is possible to reason about certain queries with non-ground situation terms (generalized queries in $\mathcal{L}_{sc}^{C^2}$) using regression in $\mathcal{L}_{sc}^{C^2}$ and still keep the reasoning problem to be decidable, and whether or not there are restrictions on the format of the generalized queries.

### 3.5 Practical Expressiveness and Limitations

In the previous sections, we have shown that the projection problem and the executability problem in the language of $\mathcal{L}_{sc}^{C^2}$ are decidable. However, the language has certain syntactic restrictions. In this section, we would like to discuss the practical expressiveness of the framework and possible future directions on applications in a general manner.

Besides the Web service examples given in Section 3.3.3, we also considered several well-known International Planning Competition (IPC) domains such as Blocks World,
Logistics, Depots, DriverLog and Satellite, etc. Some domains (e.g., Blocks World) can be represented in \( \mathcal{L}_{sc}^{C2} \) directly (see Example 6).

**Example 6** The following is a simple Blocks World domain represented in \( \mathcal{L}_{sc}^{C2} \), which describes the same Blocks World domain as given in [141].

**Action Precondition Axioms:**

\[
Poss(move(x, y), s) \equiv clear(x, s) \land clear(y, s) \land x \neq y, \\
Poss(moveToTable(x), s) \equiv clear(x, s) \land \neg onTable(x, s).
\]

**Successor State Axioms (SSAs):**

\[
on(x, y, do(a, s)) \equiv a = move(x, y) \lor on(x, y, s) \land \neg [a = moveToTable(x) \lor (\exists y)a = move(x, y)], \\
onTable(x, do(a, s)) \equiv a = moveToTable(x) \lor onTable(x, s) \land \neg (\exists y)a = move(x, y), \\
clear(x, do(a, s)) \equiv (\exists y)[on(y, x, s) \land a = moveToTable(y)] \lor (\exists y)[on(y, x, s) \land (\exists x)a = move(y, x)] \lor clear(x, s) \land \neg (\exists y)a = move(y, x).
\]

In fact, in \( \mathcal{L}_{sc}^{C2} \), there is another way of describing the changes of fluent \( clear(x, s) \). Instead of writing an SSA for \( clear(x, do(a, s)) \), we may give an acyclic TBox axiom as follows:

\[
clear(x, s) \equiv \neg (\exists y)on(y, x, s).
\]

Moreover, note that the SSAs above are also \( \mathcal{ALOO}(U) \)-restricted.

However, it is not so easy to express some of the ICAPS or IPC domains in \( \mathcal{L}_{sc}^{C2} \) directly. For example, in a simple Logistics domain (see Example 7 below), the RHS

\footnote{These competitions are regularly held in conjunction with the International Conference on Automated Planning and Scheduling (ICAPS).}
of the SSA of fluent \(at(x, y, s)\) (object \(x\) is at location \(y\) in the situation \(s\)) needs an additional free variable in the context condition of the action \(drive\): variable \(x\) for the object, variable \(y\) for the destination, and an auxiliary variable, say \(t\), for a truck, that drives from some location to the destination \(y\) with \(x\) in it.

Since we currently focus our attention on the decidability of the projection problem and the executability problem for ground situations and actions, in practice we can loosen the restriction on the number of arguments of actions and the number of variables used in axioms in the framework so that the language can be a little bit more expressive and easier to use when writing basic action theories for applications. The idea of the extension is as follows. We allow action functions to have more than two arguments and loosen the syntactic restrictions of using two object variables only in action precondition axioms and SSAs in a basic action theory in \(\mathcal{L}_{sc}^{C^2}\). So that, a regressable sentence of a projection problem or an executability problem (for ground actions and situations) will still be a \(C^2\) formula after regression and possibly some steps of simplification. Formally, we have:

- For any \(n\)-ary action \(A(\vec{v})\), where \(\vec{v}\) is an \(n\)-ary variable vector which may or may not include variable \(x\) or variable \(y\), its action precondition axiom is of the form

\[
Poss(A(\vec{v}), s) \equiv \Pi_A(\vec{v}, s), \quad (3.9)
\]

where \(\Pi_A(\vec{v}, s)\) is a first-order formula whose free variables are among \(\vec{v}, s\) at most and quantified variables are among variables \(x\) and \(y\) at most. Note that when variable vector \(\vec{v}\) and \(s\) are substituted with ground terms, the RHS of Eq. (3.9) is a \(C^2\) formula.

- For any relational fluent \(F(\vec{x}, s)\), where \(\vec{x}\) is either variable \(x\) or variable vector
Chapter 3. Decidable Reasoning in a Modified Situation Calculus

78

With the above practical extension of $\mathcal{L}_{sc}$, its SSA is of the form

$$F(\vec{x}, do(a, s)) \equiv \bigvee_{i=1}^{m_+} [\exists \vec{v}_{(i,+)}][(\exists \vec{t}_{(i,+)})(a = A^+_i(\vec{v}_{(i,+)}, \vec{t}_{(i,+)}, \vec{x}) \land \psi_i^+(\vec{v}_{(i,+)}, \vec{x})[s])]$$

$$\lor F(\vec{x}, s) \land$$

$$\neg(\bigvee_{j=1}^{m_-} [\exists \vec{v}_{(j,-)}][(\exists \vec{t}_{(j,-)})(a = A^-_j(\vec{v}_{(j,-)}, \vec{t}_{(j,-)}, \vec{x}) \land \psi_j^-(\vec{v}_{(j,-)}, \vec{x})[s])]) \quad (3.10)$$

Here, each vector $\vec{v}_{(i,+)}$, $i = 1..m_+$, ($\vec{v}_{(j,-)}$, $j = 1..m_-$, respectively) represents a vector of variables appeared in both $A^+_i$ and $\psi_i^+$ (both $A^-_j$ and $\psi^-_j$, respectively) other than $a$, $s$ or any free variables in $\vec{x}$. Each vector $\vec{t}_{(i,+)}$, $i = 1..m_+$, ($\vec{t}_{(j,-)}$, $j = 1..m_-$, respectively) only, and which are distinct from $a$, $s$ or any free variables in $\vec{x}$. Each $A^+_i(\vec{v}_{(i,+)}, \vec{t}_{(i,+)}, \vec{x})$ $i = 1..m_+$, ($A^-_j(\vec{v}_{(j,-)}, \vec{t}_{(j,-)}, \vec{x})$, $j = 1..m_-$, respectively) is an action term with variables among $\vec{v}_{(i,+)}$ ($\vec{v}_{(j,-)}$, respectively), $\vec{t}_{(i,+)}$ ($\vec{t}_{(j,-)}$, respectively), and $\vec{x}$ at most. Each $\psi_i^+(\vec{v}_{(i,+)}, \vec{x})$, $i = 1..m_+$, ($\psi_j^-(\vec{v}_{(j,-)}, \vec{x})$, $j = 1..m_-$, respectively), is a first-order formula with free variables among $\vec{v}_{(i,+)}$ ($\vec{v}_{(j,-)}$, respectively) and $\vec{x}$ at most and with bound variables among symbol $x$ and symbol $y$ at most. Note that when variables in $\vec{x}$ are substituted with variable $x$, $y$ or ground terms, and $a$ and $s$ are substituted with ground terms, we may define an extended regression operator of $\mathcal{R}C^2$ to ensure the RHS of Eq. (3.10) to be $C^2$. The reason is that for each clause of the form $a = A^+_i(\vec{v}_{(i,+)}, \vec{t}_{(i,+)}, \vec{x})$ (or $a = A^-_j(\vec{v}_{(j,-)}, \vec{t}_{(j,-)}, \vec{x})$, respectively) in Eq. (3.10), if $a$ is substituted with an action whose action name is not $A^+_i$ (or $A^-_j$, respectively), then it can be simplified as $false$ during regression; otherwise, all variables other than $\vec{x}$, i.e., variables $\vec{v}_{(i,+)}$, $\vec{t}_{(i,+)}$ (or $\vec{v}_{(j,-)}$, $\vec{t}_{(j,-)}$, respectively) are can be replaced with ground terms, so that $\psi_i^+(\vec{v}_{(i,+)}, \vec{x})[s]$ (or $\psi_j^-(\vec{v}_{(j,-)}, \vec{x})[s]$, respectively) will be a $C^2$ formula after substitution and simplification by using the well-known logical validity $\exists x_1, \ldots, x_n.(\wedge_{i=1}^n x_i = t_i) \land \psi(x_1, \ldots, x_n) \equiv \psi(t_1, \ldots, t_n)$, where each $t_i$ ($i = 1..n$) is a ground term.

With the above practical extension of $\mathcal{L}_{sc}^{C^2}$, where the projection problem and the
executability problem are still decidable, we are able to express most of the ICAPS or IPC domains (e.g., Logistics, Depots, DriverLog and Satellite, etc.) straightforwardly. We give an example of a simple Logistics domain to illustrate the proposed extensions of syntax. Note that Logistics cannot be expressed in $C^2$ itself.

**Example 7** The following is a part of a simple Logistics domain represented in the practical extension of $\mathcal{L}_{sc}^{C2}$ defined above.

**Action Precondition Axioms:**

\[
\text{Poss}(\text{drive}(x, y, z), s) \equiv \text{vehicle}(x) \land \text{at}(x, y, s) \land y \neq z,
\]
\[
\text{Poss}(\text{load}(x, y, z), s) \equiv \text{crate}(x) \land \text{vehicle}(y) \land \text{at}(x, z, s) \land \text{at}(y, z, s) \land \neg \text{loaded}(x, s),
\]
\[
\text{Poss}(\text{unload}(x, y, z), s) \equiv \text{crate}(x) \land \text{vehicle}(y) \land \text{at}(y, z, s) \land \text{in}(x, y, s).
\]

**Successor State Axioms:**

\[
\text{loaded}(x, \text{do}(a, s)) \equiv (\exists y, z)a = \text{load}(x, y, z) \lor \text{loaded}(x, s) \land \neg (\exists y, z)a = \text{unload}(x, y, z),
\]
\[
\text{at}(x, y, \text{do}(a, s)) \equiv (\exists z)a = \text{drive}(x, z, y) \lor (\exists t)[(\exists z)a = \text{drive}(t, y, z) \land \text{in}(x, t, s)] \lor \\
\text{at}(x, y, s) \land \neg [(\exists z)a = \text{drive}(x, y, z) \lor (\exists t)[(\exists z)a = \text{drive}(t, y, z) \land \text{in}(x, t, s)]],
\]
\[
\text{in}(x, y, \text{do}(a, s)) \equiv (\exists z)a = \text{load}(x, y, z) \lor \text{in}(x, y, s) \land \neg (\exists z)a = \text{unload}(x, y, z).
\]

The $\mathcal{L}_{sc}^{C2}$ regression operator only needs minor extensions based on above changes to the syntax of action precondition axioms and SSAs, and the decidability will still be ensured if we want to solve the projection problem or the executability problem via regression. Since the changes are trivial, details of the formalization are omitted here. Implementing our work and applying it to interesting domains will be one of the future research directions. Obviously, the practical extension we proposed above for future study is a very minor extension. There are still BATs that can not be represented in this
extension. One example is a BAT with at least one action that can cause global effects on a fluent that has more than two object variables, whose domains are all infinite. Another example is a BAT that needs more than two distinct local variables when specifying context conditions in an SSA.

Given this practical extension of $L_{sc}^{C2}$, if we want to look for a better computational upper bound for the projection problem solved via regression, we may also make restrictions in the practical extension of $L_{sc}^{C2}$ similar to the approach proposed in Section 3.3.4. Then, the syntax of action functions in a domain is not restricted, but answering an $ALCO(U)$-restricted (or $ALCOO(U)$-restricted, respectively) query for ground situations can still be solved in 2-ExpTime.

The general idea is to loosen the syntactic restrictions on the class of formulas $FO_{DL}$ (or $FO_{DL^+}$, respectively). Then, the syntax of the SSAs in a BAT will be restricted based on the updated definition of context formulas.

For example, we may define an extension of $FO_{DL}$, say $E$, as the union of $E^x \cup E^y$, where $E^y$ can be obtained from $E^x$ by replacing every variable symbol $x$ with variable symbol $y$ and every $y$ with $x$, and $E^x$ is a minimal set constructed recursively as follows:

1. $true$ and $false$ are in $E^x$.

2. If $AC$ is a monadic predicate name, then $AC(v)$ ($AC(b)$, respectively) is in $E^x$ for any variable $v \neq y$ (for any constant $b$, respectively).

3. $x = b$ ($x = z$, respectively) is in $E^x$ for any constant $b$ (for any variable symbol $z$ that is distinct from $x$ and $y$, respectively).

4. If $R$ is a binary predicate name, then $R(z_1, z_2)$, $R(b_1, b_2)$, $R(b_1, z_2)$, $R(z_1, b_2)$, $R(x, b_2)$ and $R(x, z_2)$ are in $E^x$ for any constants $b_1$ and $b_2$ and any variable symbols $z_1$ and $z_2$ that are distinct from $x$ and $y$.

5. If $\phi$ is in $E^x$, then $\neg \phi$ is in $E^x$. 
6. If $\phi$ and $\psi$ are in $E^x$, then $\phi \land \psi$ and $\phi \lor \psi$ are in $E^x$.

7. If $\phi(x)$ is in $E^x$, variable $y$ is not free in $\phi(x)$ if there is any, and $R$ is a dyadic predicate name, $\tilde{\phi}(y)$ is the dual formula of $\phi(x)$, obtained by renaming every $x$ (both free and bound) with $y$ and every $y$ (both free and bound) with $x$ in $\phi$, then $\exists y.R(x,y) \land \tilde{\phi}(y), \exists y.R(b,y) \land \tilde{\phi}(y)$ (for any constant $b$), $\exists y.R(z,y) \land \tilde{\phi}(y)$ (for any variable symbol $z$ that is distinct from $x$ and $y$), and $\forall y.R(x,y) \supset \tilde{\phi}(y)$, $\forall y.R(b,y) \land \tilde{\phi}(y)$ (for any constant $b$), $\forall y.R(z,y) \land \tilde{\phi}(y)$ (for any variable symbol $z$ that is distinct from $x$ and $y$) are in $E^x$.

8. If $\phi$ is in $E^x$, $\tilde{\phi}$ is the dual formula of $\phi$, obtained by renaming every $x$ (both free and bound) with $y$ and every $y$ (both free and bound) with $x$ in $\phi$, then $[\exists y.]\tilde{\phi}(y)$ and $[\forall y.]\tilde{\phi}(y)$ are in $E^x$, where $[\exists y.]$ ($[\forall y.]$, respectively) means that if $\tilde{\phi}$ has a free variable $y$, then it is quantified by $\exists y$ ($\forall y$, respectively); otherwise, there is no need to add the quantifier.

The intuition behind the definition of class $E$ is that any variable symbol other than $x$ and $y$ has to be free in a formula in $E$. In fact, it can be considered as a placeholder for some constant that will be substituted for this variable when action variable $a$ is instantiated with a ground action term. Note that $FO_{DL}$ is a sub-class of $E$. Then, similar to Def. 10, we can restrict the syntax of the context conditions in an SSA of a fluent. In particular, we say the SSA for a fluent $F$ is $\mathcal{A\mathcal{C}_O}(U)$-restricted in the practical extension of $\mathcal{L}_{sc}^{C^2}$ if the SSA of $F$ has the form of Eq. (3.10), where each context condition $\psi^+_i$ (or $\psi^-_j$, respectively) is a formula in $E$ when all situation variables are suppressed. Moreover, we say that the set of SSAs $D_{ss}$ in a BAT $D$ in the practical extension of $\mathcal{L}_{sc}^{C^2}$ is $\mathcal{A\mathcal{C}_O}(U)$-restricted if every axiom of a primitive dynamic concept in $D_{ss}$ is $\mathcal{A\mathcal{C}_O}(U)$-restricted and every axiom of a dynamic role in $D_{ss}$ is both $\mathcal{A\mathcal{C}_O}(U)$-restricted and context-free.\footnote{Recall that an SSA of a fluent $F$ is context-free does not mean that all effects of actions on fluent $F$}
We still use the same term “\textit{ALCO}(U)-restricted” because of the following reason. In $\psi_i^+$ ($\psi_j^-$, respectively), all auxiliary variables among $\vec{v}_{(i,+)}$ ($\vec{v}_{(j,-)}$, respectively) are free and appear in $a = A_i^+(\vec{v}_{(i,+)}, \vec{l}_{(i,+)}, \vec{x})$ ($a = A_j^-(\vec{v}_{(j,-)}, \vec{l}_{(j,-)}, \vec{x})$, respectively). When we perform regression for $F(\vec{t}, \text{do}(A, S))$ for some ground action $A$ and ground situation $S$, variables $a$ and $s$ will be replaced by ground terms, and using standard first-order logic simplification techniques, each positive or negative effect clause in Eq. (3.10) either will be equivalent to \textit{false} (because the action function name of $A$ is different from $A_i^+$ or $A_j^-$), or will still be a formula in $\textit{FO}_{DL}$ (because $\vec{v}_{(i,+)}$ for some $i$ ($\vec{v}_{(j,-)}$ for some $j$, respectively) can be replaced by constants obtained from ground action $A$ in the context condition $\psi_i^+$ ($\psi_j^-$, respectively)). Since any formula in $\mathcal{D}_T$ and $\mathcal{D}_{S_0}$ does not involve any action terms when situation terms are suppressed, the definition of \textit{ALCO}(U)-\textit{restricted} for these two classes of axioms will not be changed in Def. 10.

Intuitively, we may have the following conjecture similar to Th. 8, and will leave the proof for future work.

\textbf{Conjecture 1} Consider a BAT $\mathcal{D}$ represented in the practical extension of $\mathcal{L}^{C^2}_{sc}$ whose $\mathcal{D}_{ss}$, $\mathcal{D}_T$ and $\mathcal{D}_{S_0}$ are \textit{ALCO}(U)-\textit{restricted}. Let $W$ be any $\mathcal{L}^{C^2}_{sc}$ regressable sentence in $\mathcal{D}$ that is uniform in a ground situation $S$ and has no appearance of Poss. If $W$ is \textit{ALCO}(U)-\textit{restricted}, i.e., $W$ is in $\textit{FO}_{DL}$ with the situation term suppressed, then answering the query whether $\mathcal{D} \models W$ via regression can be solved in 2-\textit{ExpTime} with respect to the size of $W$ when $\mathcal{D}$ is fixed.

According to Conjecture 1, given a BAT $\mathcal{D}$ in the practical extension of $\mathcal{L}^{C^2}_{sc}$ whose $\mathcal{D}_{ss}$, $\mathcal{D}_T$ and $\mathcal{D}_{S_0}$ are \textit{ALCO}(U)-\textit{restricted}, the project problem can be solved in 2-\textit{ExpTime} via regression if the query (with the situation terms suppressed) is a formula in $\mathcal{E}$. If, in addition, the RHS of each precondition axiom in $\mathcal{D}$ is a formula in $\mathcal{E}$ (with the situation term suppressed) when the occurrences of all the action arguments are substituted by

are unconditional, but means that the context conditions of actions in the SSA are situation independent if there are any.
constants, then the executability problem can also be solved in $2$-$\text{ExpTime}$ via regression.

Note that the latter condition that the SSAs of the dynamic roles in $\mathcal{ALCO}(U)$-restricted $D_{ss}$ need to be context-free may be too restrictive for some practical domains. During the proof of Lemma 2 (Appendix B.3), we noticed that even if the SSAs of the dynamic roles are not context-free, the regression of an $\mathcal{ALCO}(U)$-restricted regresable query with ground situation terms still remains an $\mathcal{ALCO}(U)$-restricted sentence (after applying some logical transformation introduced in the proof). For instance, we observed that although the SSAs (including those for roles) in the Logistics domain of Example 7 are $\mathcal{ALCO}(U)$-restricted and the SSA for the dynamic role $at(x, y, s)$ is not context-free, the regression of an $\mathcal{ALCO}(U)$-restricted regresable query with ground situation terms is still $\mathcal{ALCO}(U)$-restricted. This seems to be a surprise given that Logistics domains have been known for a long time and were not designed to be expressible in our language. However, without the context-free restriction for the SSAs of roles, it is not easy to estimate the size of the resulting $\mathcal{ALCO}(U)$-restricted formula for an $\mathcal{ALCO}(U)$-restricted query. We leave it as a future work.

Although $\mathcal{L}^{C^2}_{sc}$ and its practical expression have obvious expressive limitations, it is possible to apply them to some other interesting dynamical domains besides Semantic Web services and planning problems. Databases and ontologies have very close relationships to DLs and first-order logic. The instances of a type (or a relation, respectively) in a database can be changed via database transactions. The transactions and their effects can be formalized in the situation calculus [140]. Although currently it is not obvious whether or not $\mathcal{L}^{C^2}_{sc}$ (and its practical extension) can be applied to advanced database transaction models [97], it can be used to formalize basic transactions shown in [140] for some databases and their projection and executability problems can be solved efficiently. Similarly, the extensional update of an ontology, i.e., the update and erasure of an ontology at the instance level [38], is close to database update, except that an ontology also includes general knowledge about concepts and their relationships represented as logic
axioms, and the update result has to be consistent with these existing general knowledges. The study of extensional ontology update and erasure is relatively a new area, and only a few works considered this problem. For example, in [73], the authors proposed a syntactic approach to add and remove of ABox assertions for tableau completion graphs created during consistency checking in expressive Description Logics. Other papers, such as [71, 107, 38], proposed semantic approaches of update and erasure. In [71], erasure is studied for ontologies represented in Resource Description Framework (RDF). In [107], Liu et al. considered updating an ABox in a DL with an acyclic TBox following the approach of [172] and also mentioned that update can be applied to a boolean ABox formulated in $C^2$. In [37], De Giacomo et al. used a less expressive DL language $DL$-$Lite$, but defined an update for the case when TBox consists of General Concept Inclusion axioms (GCIs) in comparison to acyclic TBox that is required in [107]. It is shown that the result of an update is always expressible by a $DL$-$Lite$ ABox and a polynomial-time algorithm is provided that computes the update over a $DL$-$Lite$ knowledge base. The more recent paper [21] from the same research group proposed to use Golog-like programs to efficiently reason about actions over ontologies based on a functional view of ontology with cyclic TBox in the case when the ontology is expressed in $DL$-$Lite$. Another recent work of De Giacomo et al. [38] provided polynomial algorithms for approximated instance-level update and erasure for ontologies that can be represented in $DL$-$Lite_\mathcal{F}$. Notice that most of these researches of ontology update and erasure are based on DLs. The application of our $Lsc^2$ framework can possibly provide a very different and new approach for research in this area by using methods developed for reasoning about action and change. The formalization of the application and implementations of using $Lsc^2$ for ontology and database updates need further detailed study and will be one of our future research directions.
3.6 Additional Discussion and Future Work

The major consequence of the results proved above for the problem of service composition is the following. If both atomic services and properties of the world that can be affected by these services have no more than two parameters (other than the situation argument), then we are guaranteed that even in the state of incomplete information about the world, one can always determine whether a sequentially composed service is executable and whether this composite service will achieve a desired effect. The previously proposed approaches made different assumptions: McIlraith and Son [116] assume that the complete information is available about the world when effects of a composite service are computed, and De Giacomo et al. [35] and Finzi et al. [48] consider the propositional fragment of the situation calculus.

In [116, 126], it was proposed to use Golog for composition of Semantic Web services. Because our primitive actions correspond to elementary services, it is desirable to define Golog in our modified situation calculus too. It is surprisingly straightforward to define almost all Golog operators starting from our $C^2$-based situation calculus. The only restriction in comparison with the original Golog [101, 141] is that we cannot define the operator $(\pi x)\delta(x)$, non-deterministic choice of an action argument, because $L_{sc}^{C^2}$ regressive formulas cannot have occurrences of non-ground action terms in situation terms. In the original Golog this is allowed, because the regression operator is defined for a larger class of regressive formulas. However, everything else from the original Golog specifications remain in force, no modifications are required. In addition to providing a well-defined semantics for Web services, our approach also guarantees that the evaluation of tests in Golog programs is decidable (w.r.t. an arbitrary initial theory $D_{S_0}$), which is missing in other approaches (unless one can make the closed world assumption or impose another restriction to regain decidability).
In [7, 8, 9] an integration of the description logic $\mathcal{ALCQIO}$ (and its sub-languages) with an action formalism for reasoning about Web services is proposed. Their paper starts with a description logic $\mathcal{ALCQIO}$ and then defines services (actions) meta-theoretically: an atomic service is defined as the triple of sets of description logic formulas. The actions described in [8] are all ground, the preconditions and effects of the actions are described using ABoxes with certain syntactic restrictions. To solve the executability and projection problems the paper introduces an approach similar to regression, and reduces this problem to description logic reasoning. The main aim is to show how the executability of sequences of actions and a solution to the projection problem can be computed, and how the complexity of solving these problems depends on the chosen description logic.

Despite the fact that our work and [8] have common goals, our developments start differently and proceed in different directions. We start from the syntactically restricted first-order language $C^2$ (that is significantly more expressive than $\mathcal{ALCQIO}$), use it to construct the modified situation calculus (where actions are functional terms rather than ground terms only), and describe the preconditions and effects of action functions using $C^2$-restricted BATs in this language, which solves the frame problem when representing the effects of actions. Moreover, it is possible to represent actions with global effects in our framework. We studied different approaches of solving the executability and projection problems with and without the regression technique. We show that the computational complexity for the executability and projection problems in general is $L_{sc}^{C^2}$ is co-NExpTime-complete, which is the same as that of in the framework of [8] based on $\mathcal{ALCQIO}$, yet $L_{sc}^{C^2}$ is significantly more expressive both for representing actions and their effects. We also considered sub-languages of $L_{sc}^{C^2}$, which are related to $\mathcal{ALCQO}(U)$ and $\mathcal{ALCQO}(U)$ to gain better computational properties for reasoning about actions via regres-

\footnote{[7] is the extended version of [8], and [9] is the technique report which provides additional technical details.}
sion. The restricted sub-languages \((\mathcal{ALCO} U)-\)restricted and \(\mathcal{ALCQO} U)-\)restricted ones\) are incomparable with the framework in [8], since they have different expressive powers. For instance, it is not possible to quantify over arguments of an action or to describe actions that have global effects in [8], while there is restriction on the context-conditions for in the SSAs of roles for \(\mathcal{ALCO} U)-\)restricted and \(\mathcal{ALCQO} U)-\)restricted sub-languages in our work.

Because of using ABoxes to describe atomic services (ground actions), the advantage of [8] is that all reasoning is reduced to reasoning in description logics (and, consequently, can be efficiently implemented especially for less expressive fragments of \(\mathcal{ALCQIO}\)). Our advantages are as follows. The convenience of representing actions as functional terms and the expressive power of \(L_{sc}^{C^2}\), including the compact representation of BATs. Besides, since \(C^2\) and \(\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id)\) are equally expressive, there are some (situation suppressed) formulas in our modified situation calculus that cannot be expressed in \(\mathcal{ALCQIO}\) (in which complex roles are not allowed). In particular, Tobies [158] shows that \(\mathcal{ALCQI}\) has the same complexity as \(C^2\), but \(\mathcal{ALCQI}\) is strictly less expressive than \(C^2\): reflexive binary relations cannot be expressed in \(\mathcal{ALCQI}\). The dynamical systems represented using DLs in [8] can be translated to BATs in our language (note that in [9], the authors provide a translation of their representation of action theory into the situation calculus and the translated result is in fact expressed in \(L_{sc}^{C^2}\)). Because the expressive power of \(L_{sc}^{C^2}\) is generally larger than the language in [8], the other way of the translation is impossible in general. Second, in \(L_{sc}^{C^2}\), one can answer queries with quantifiers over any object variables in fluents, but the approach in [8] does not allow this because its query has to be expressed as an ABox statement in \(\mathcal{ALCQIO}\). Moreover, since our actions are represented as functional terms, for our future research direction, it is possible for us to consider reasoning problems that are more general than the executability and projection problems, where situation terms are no longer ground.

The more recent papers of the same research group continue to explore the research
direction initiated in [8]: in [121, 120], Miličić investigates complexity of planning in a
description logic based action formalism; and in [106], Liu et al. attempt to solve the
ramification problem when a TBox consists of general concept inclusion axioms (GCIs),
and it is no longer an acyclic TBox as in [8]. Considering cyclic TBox in the framework
of $\mathcal{L}_{sc}^{C_2}$ will be another possible research topic in the future.

Propositional dynamic logic (PDL) was derived from dynamic logic and has several
advantageous properties: PDL has the finite model property and is decidable [74]. Its
satisfiability problem is $\text{ExpTime}$-complete [49, 135]. It turned out to be popular not
only for reasoning about regular programs, but also as a logic of action [36, 137]. It is
well known that dynamic logic extends modal logic by associating to every action $a$, basic
or complex, the modal operators $[a]$ and $\langle a \rangle$, thereby making it a multimodal logic. But,
in PDL quantification over actions is not allowed. More recently, some works [24, 42, 41]
adapt PDL to reasoning about actions by quantifying over actions and allowing for
equality between actions. They use regression and formulate the SSAs to solve the frame
problem similar to [141]. However, in their framework, action terms can be constants
or variables only (the domain closure axiom for actions or another similar assumption
is required) and all fluents are propositional only. In our work, actions in BATs can be
first-order terms, and the arity of each action function is no greater than two. Moreover,
in our language, fluents can be dynamic concepts or dynamic roles, not just propositional
statements. Also, as we mentioned above, it is possible to define complex Golog programs
in our language.

In [173], the combined dynamic description language $\mathcal{PDLC}$ has been proposed as an
attempt to reason about dynamics in description logics. From the perspective of modal
logic, Wolter and Zakharyaschev [173] combine polymodal $\mathcal{K}$ with PDL and proves the
decidability of the resulting hybrid logic. $\mathcal{PDLC}$ is somewhat related to the products
of modal logics (see [52, 53, 54] for the definition and survey of results). The issues
related to combining modal logics in a more general context are reviewed in [99]. The
proposed dynamic description logic is intended to define and classify concepts referring to actions and to describe dynamically changing domains by means of varying extensions of concepts. A careful examination of the syntax of \( \text{PDLC} \) shows that actions can only be terms built from atomic actions (i.e., action variables) using standard dynamic logic constructors (composition, alternation, iteration) and from formulas using tests. Another restriction is that only concepts can change after executing an action: \([\alpha]C\) is a concept, where \(\alpha\) is an action term and \(C\) is a concept, but there is no similar constructor for roles. However, for any atomic formula \(\phi\) which is either an ABox statement (\(a : C\), or \(aRb\), where \(a, b\) are object names) or a boolean combination of ABox statements, and for any action term \(\alpha\), \([\alpha]\phi\) is also a formula. The main contribution of [173] is the proof of the theorem that the satisfiability problem for \(\text{PDLC}\)-formulas is decidable, but the complexity of the decision problem and the design of efficient decision algorithms are not explored. Our modified situation calculus can use action functions with arity no greater than two, and our dynamic roles can change after executing a sequence of actions too. However, we do not prove the decidability of the satisfiability problem for arbitrary formulas in our language. Moreover, we conjecture that this problem is undecidable in our language. From the positive side, we demonstrate that the executability and the projection problem are decidable for a wide class of queries and because these problems are the most essential in applications, the ability to solve these problems is sufficient for practical purposes. In [26, 25], the authors propose a logic that is similar to [9, 8, 173]. In the proposed logic, one can not only reason about complex actions similar to [173], but also characterize actions by preconditions and conditional effects as in [8]. Also, a tableau algorithm for deciding satisfiability proposed in [26, 25] is based on an elaborated combination of previously known tableau algorithms.

In our paper [66], we investigated not only regression, but also progression as an alternative approach to solving the projection problem. We considered a modified progression that is weaker than the classical progression [105] for an incomplete knowledge base given
local-effect SSAs defined in [109]. We proved that the modified progression is sound with respect to the classical progression, and we also provided an algorithm to compute our progression for the case when the initial theory is a CNF-based knowledge base (a set of disjunctions of equality-based formulas). Recently, [163] considers a notion of strong progression, a slight variant of the classical progression. In [163], it is shown that the strong progression is first-order definable for a BAT $\mathcal{D}$ with local-effect SSAs and the algorithm for computing progression was proposed for a special case of a BAT $\mathcal{D}$ with the so-called strong local-effect SSAs. Most recently, [108] presents a result stronger than that of [163], which shows that for local-effect actions, progression is always first-order definable and computable. They give a constructive proof for this via the concept of forgetting. They also show first-order definability and computability results for a class of knowledge bases and actions with non-local effects. Moreover, for a certain class of local-effect actions and knowledge bases for representing disjunctive information, they show that progression is not only first-order definable but also efficiently computable. The results in [108] can be adapted to our $\mathcal{L}_{sc}^{C2}$ situation calculus straightforwardly, but nothing more than that. We currently cannot think yet of any result stronger than what they had by using $\mathcal{L}_{sc}^{C2}$. Therefore, it is not very meaningful to have a separate section to repeat what they had done. The further study of progression in $\mathcal{L}_{sc}^{C2}$ can be a future research direction.

There are several other proposals to capture the dynamics of the world in the framework of description logics and/or its slight extensions. Similar to our paper [66], Drescher and Thielscher [44] explored reasoning about actions based on a description logic, but they concentrate on the fluent calculus [144] instead of the situation calculus. Instead of dealing with actions and the changes caused by actions, some of the approaches turned to extensions of description logics with temporal logics to capture the changes of the world over time [3, 4], and some others combined planning techniques with description logics to reason about tasks, plans and goals and exploit descriptions of actions, plans,
and goals during plan generation, plan recognition, or plan evaluation [57]. In [3] and [57], researchers reviewed several other related papers.

Researchers also proposed to describe actions and the changes in terminological knowledge bases, closely related to description logics. For example, C. Kemke [95] describes action concepts by a set of parameters or object variables which refer to concepts in the object taxonomy, and precondition formulas as well as effect formulas describing how the world changes through actions (similar to STRIPS planning systems). In [11], all the actions of e-services are specified as constants, all the fluents of the system have only situation arguments, and BATs are translated under such assumptions into the description logic framework. Also, it has a limited expressive power without using arguments of objects for actions and/or fluents: this may cause a blow-up of the knowledge base.

In the future, we plan to extend this work along several directions. 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 at IBM for an automated system for recommending products and services in the domains of banking, insurance and telephony [124].

The most important direction for future research is an efficient implementation of practical scenarios of reasoning in $\mathcal{L}_{sc}^{C^2}$ and in its fragments: an efficient implementation of a decision procedure for solving the executability and projection problems. This procedure should handle the modified $\mathcal{L}_{sc}^{C^2}$ regression and perform efficient reasoning in $\mathcal{D}_{s_0}$. It should be straightforward to modify existing implementations of the regression operator for our purposes, but it is less obvious which reasoner will work efficiently on practical problems. There are several different directions that can be explored. First, according to [18] and Th. 1, there exists an efficient algorithm for translating $C^2$ formulas to $\mathcal{ALCQIO}(∪, ∩, ¬, |, id)$ formulas. Also, if we consider fragments of $\mathcal{L}_{sc}^{C^2}$ introduced in Section 3.3.4 that guarantee a better complexity of solving the projection problem (see Th. 8), more specifically, a BAT $\mathcal{D}$ whose $\mathcal{D}_{ss}$ and $\mathcal{D}_{T}$ are $\mathcal{ALCQO}(U)$-restricted, then a reason-
ing procedure working with $D_{S_0}$ should be able to handle the description logic $\mathcal{ALCO}(U)$. Consequently, one can try to adapt tableaux-based decision procedures, such as those proposed in [148, 149], for (un)satisfiability checking in $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, \mid, id)$ and in $\mathcal{ALCO}(U)$. Second, one can try to avoid any translation from $C^2$ to $\mathcal{ALCQIO}(\sqcup, \sqcap, \neg, \mid, id)$ and adapt resolution based automated theorem provers for the purposes of reasoning in $D_{S_0}$ [85, 39]. Although in general, the worst-case computational complexity for the reasoning problems in $\mathcal{L}_{sc}^C$ or in its fragments is high, some practical scenarios may facilitate empirically efficient solutions to the projection and executability problems.

Another important research direction is a possible extension of $\mathcal{L}_{sc}^C$ to a language that includes $n$-ary predicates, but still guarantees decidability as we discussed in Section 3.5. Its implementation and application to other practical domains such as Semantic Web services, planning domains, database update and ontology update will be explored as well.

Finally, we would like to explore how our version of the situation calculus can accommodate events considered by John McCarthy in [115].
Chapter 4

Reasoning about Large Taxonomies of Actions

In this chapter, we design a representation based on the situation calculus to facilitate development, maintenance and elaboration of very large taxonomies of actions. This representation leads to more compact and modular basic action theories (BATs) for reasoning about actions than currently possible. We compare our representation with Reiter’s BATs and prove that our representation inherits all useful properties of his BATs. Moreover, we show that our axioms can be more succinct, but extended Reiter’s regression 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 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 in a domain. The major results of this work have been published in [68].
4.1 Motivations

A long-standing and important problem in AI is the problem of how to represent and reason about effects of actions grouped in a realistically large taxonomy, where some actions can be more generic (or more specialized) than others. While the problem of representing large semantic networks of (static) concepts has been addressed in AI research from the 1970s [19] and served as motivation for research on description logics, a related problem of representing and reasoning about large taxonomies of actions received surprisingly little attention. We would like to address this problem using the situation calculus. There are several different formulations of the situation calculus. In this chapter, we would like to concentrate on basic action theories (BATs) introduced in [141], in particular, on successor state axioms (SSAs) proposed by Reiter to solve (sometimes) the frame problem. Recall that BATs are more expressive than STRIPS theories: actions specified using BATs can have context-dependent effects. We propose a representation that allows writing more compact and modular BATs than is currently possible. BATs are logical theories of a certain syntactic form that have several desirable theoretical properties. However, BATs have not been designed to support taxonomic reasoning about objects and actions. Essentially, these theories are “flat” and do not provide representation for hierarchies of actions. This can lead to potential difficulties if one intends to use BATs for the purpose of large scale formalization of reasoning about actions on the commonsense level, when potentially arbitrary actions 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 a specialization of the action of transportation using a vehicle, that is in its turn a specialization of the action of moving an object from one location to another. We represent hierarchies of actions explicitly and use them in our new modular SSAs. However, we show that our new modular SSAs can be translated into “flat” Reiter’s SSAs and, consequently, we inherit all useful properties of his BATs: formulas entailed from Reiter’s BAT remain entailment from a modular BAT; consequently, the projection
Below, in Section 4.2 and Section 4.3, we propose a new representation that helps to design modular BATs and prove that it has the same desirable logical properties as Reiter’s BATs. In Section 4.4, we also discuss the significant computational advantages of using modular BATs in comparison to Reiter’s “flat” BAT. Finally, in Section 4.5, we propose an approach to designing taxonomies of actions and discuss related work.

4.2 Action Hierarchies

In practice, it is not easy to specify and reason with a logical theory $D$ if an application domain includes a very large number of actions. To deal with this problem, we propose to represent events and actions using a hierarchy.

**Definition 12** We use the predicate $sp(a_1, a_2)$ to represent that action $a_1$ is a *direct specialization* of action $a_2$ (action $a_2$ is a *direct generalization* of $a_1$). An *action diagram* is defined by a finite set $H$ of axioms of the syntactic form

$$sp(A_1(\vec{x}), A_2(\vec{y})) \equiv \phi_{A_1,A_2}(\vec{x}, \vec{y})$$

(4.1)

for two action functions $A_1(\vec{x}), A_2(\vec{y})$, where $\phi_{A_1,A_2}(\vec{x}, \vec{y})$ is a satisfiable (i.e., not equivalent to $\bot$) situation-free first-order formula with free variables at most among $\vec{x}, \vec{y}$. Also, $H$ must be such that the following condition hold:

$$H \cup D \models sp(a_1, a_2) \supset (Poss(a_1, s) \supset Poss(a_2, s)).$$

Given any action diagram $H$, we say that a directed graph $G = \langle V, E \rangle$ is a *digraph of $H$* when $V = \{A_1, \cdots, A_n\}$, where all $A_i$’s are distinct action function symbols in $D$ and a directed edge $A_j \rightarrow A_k$ belongs to the edge set $E$ iff there is an axiom of the form $sp(A_j(\vec{x}), A_k(\vec{y})) \equiv \phi_{A_j,A_k}(\vec{x}, \vec{y})$ in $H$. Clearly, for any two nodes in $G$ there is at most
one edge from one node to the other. Otherwise, assume that there are two edges from node $A_1$ to node $A_2$. Then, there are two axioms $sp(A_1(\vec{x}), A_2(\vec{y})) \equiv \phi_{A_1,A_2}(\vec{x},\vec{y})$ and $sp(A_1(\vec{x}), A_2(\vec{y})) \equiv \phi'_{A_1,A_2}(\vec{x},\vec{y})$. $\phi_{A_1,A_2}(\vec{x},\vec{y})$ has to be equivalent to $\phi'_{A_1,A_2}(\vec{x},\vec{y})$. Hence, one of the above axioms can be omitted, and there will be one edge from $A_1$ to $A_2$.

When the digraph $G$ of $\mathcal{H}$ is acyclic, i.e., there is no directed loop in $G$, we call $\mathcal{H}$ an acyclic action diagram. Below, we will only consider acyclic action diagrams. Note that if each action in the digraph of an acyclic action diagram has only one parent (single inheritance case), then the digraph is actually a forest, but generally, there can be actions that have several parents (multiple inheritance case), as shown in Examples 8 and 9.

**Example 8** Consider actions performed in a kitchen, actions such as washing, cooking, frying, etc. Some can be considered as specializations of others. To simplify the example we assume that water and electricity are always available, ignore some other kitchen activities (such as chopping, mixing, etc) and consider the (simplified) action digraph shown in Fig. 4.1. Each edge corresponds to one $sp$ axiom in the set $\mathcal{H}$, for example,

$sp(wash(x), kitchenAct)$, $sp_prepFood(x), kitchenAct)$,

$sp(cook(food, vessel), prepFood(food))$,

$sp(oilyCook(food, vessel), cook(food, vessel))$,

$sp(oilyCook(food, vessel), reheat(food))$,

$sp(microwave(food), reheat(food)), \cdots$, etc.

![Figure 4.1: A (Simplified) Action Digraph for Kitchen Activities](image)
Notice that according to Def. 12, the number of a (direct) specialization can be different from that of its generalization. For instance, in Example 8, $\text{cook}(\text{food}, \text{vessel})$ has one more argument than its generalization $\text{preFood}(\text{food})$. Intuitively, it means that $\text{cook}(\text{food}, \text{vessel})$ can have more specific information reflected through an additional argument $\text{vessel}$ than $\text{preFood}(\text{food})$. We show an additional example where actions in an action diagram $\mathcal{H}$ can have different numbers of arguments.

**Example 9** Let $\text{move}(x, l_1, l_2)$ be an action function, representing an action moving object $x$ from location $l_1$ to location $l_2$. Consider an action $\text{travel}(p, o, d)$: a person $p$ travels from origin $o$ to destination $d$. It can be regarded as a direct specialization of $\text{move}$ – person $p$ moves from location $o$ to location $d$:

$$\text{sp}(\text{travel}(p, o, d), \text{move}(p, o, d)).$$

Consider an action $\text{drive}(p, v, o, d)$, representing that a person $p$ drives a vehicle $v$ from origin $o$ to destination $d$. It can be considered as a direct specialization of action $\text{travel}$ – person $p$ travels from location $o$ to location $d$:

$$\text{sp}(\text{drive}(p, v, o, d), \text{travel}(p, o, d)).$$

It also can be a direct specialization of action $\text{move}$ – vehicle $v$ moves from location $o$ to location $d$:

$$\text{sp}(\text{drive}(p, v, o, d), \text{move}(v, o, d)).$$

It shows that one action can be a direct specialization of different actions, among which some of them might be related to each other with respect to the specialization relationship.

Consider an action $\text{passDr}(p, dr)$: a person $p$ passes through a door $dr$. It is considered as a direct specialization of $\text{move}(p, o, d)$ iff the origin $o$ is the outside of $dr$ and the destination $d$ is the inside of $dr$, or vice versa:

$$\text{sp}(\text{passDr}(p, dr), \text{move}(p, o, d)) \equiv$$

$$\text{outside}(o, dr) \land \text{inside}(d, dr) \lor \text{outside}(d, dr) \land \text{inside}(o, dr),$$
where predicate $\text{outside}(o, dr)$ ($\text{inside}(d, dr)$, respectively) is true iff $o$ ($d$, respectively) is the location that is outside (inside, respectively) of $dr$. □

Here, we will only consider action diagrams with monotonic inheritance of effects. Formally, this restriction for $\mathcal{H}$ can be represented as a second-order formula:

$$\mathcal{D} \cup \mathcal{H} \models (\forall P).sp(a_1, a_2) \land (P(s) \not\equiv P(\text{do}(a_2, s))) \supset (P(\text{do}(a_1, s)) \equiv P(\text{do}(a_2, s))).$$

Note that when $P(s) \equiv P(\text{do}(a_2, s))$ for some formula $P$, it is possible that $P(\text{do}(a_1, s))$ might or might not be equivalent to $P(s)$. That is, if an action causes an effect, its (direct) specialization should cause the same effect. However, a (direct) specialization can cause more changes than its generalizations. Since there are only finitely many (say, $m$) fluents in $\mathcal{D}$, the above second-order formula can be replaced by the finite conjunction (over $j = 1..m$) of first-order formulas (where $F_j(\vec{x}_j, s)$ is $j$-th fluent with object arguments $\vec{x}_j$):

$$sp(a_1, a_2) \land (F_j(\vec{x}_j, s) \not\equiv F_j(\vec{x}_j, \text{do}(a_2, s))) \supset (F_j(\vec{x}_j, \text{do}(a_1, s)) \equiv F_j(\vec{x}_j, \text{do}(a_2, s))).$$

Because in general we need to reason about a direct specialization of another direct specialization of an action, we define (distant) specializations using the predicate $sp^*$.

**Definition 13** The predicate $sp^*(a_1, a_2)$ represents that action $a_1$ is a (distant) specialization of action $a_2$ and is defined as a reflexive-transitive closure of $sp$:

$$sp^*(a_1, a_2) \equiv (\forall P).\{(\forall v)[P(v, v')] \land$$

$$\land (\forall v, v') [sp(v, v') \supset P(v, v')] \land$$

$$\land (\forall v, v', v'')[sp(v, v') \land P(v', v'') \supset P(v, v'')] \supset P(a_1, a_2) \quad (4.2)$$

□

Although Axiom (4.2) requires second-order logic, later we will show in Th. 13 that we can still reduce reasoning about regressable formulas to theorem proving in first-order...
logic only. We denote the set of axioms, including Axiom (4.2) and all axioms in an action diagram $H$, as $H^*$ and call it the action hierarchy (of $H$).

The following properties are trivial consequences of the definition of $sp^*$.

**Proposition 1** $sp^*$ is a partial order relationship on the set of the actions, that is, for any actions $a_1, a_2, a_3$, we have

$$sp^*(a_1, a_1); \ (Reflexivity) \quad (4.3)$$

$$sp^*(a_1, a_2) \land sp^*(a_2, a_3) \supset sp^*(a_1, a_3); \ (Transitivity) \quad (4.4)$$

$$sp^*(a_1, a_2) \land sp^*(a_2, a_1) \supset a_1 = a_2; \ (Antisymmetry) \quad (4.5)$$

$$sp^*(a_1, a_2) \supset a_1 = a_2 \lor \exists a'.(sp(a_1, a') \land sp^*(a', a_2)). \quad (4.6)$$

**Proof:** According to Def. 13, using induction principle on $sp^*$. \hfill $\square$

The following theorem states that the action hierarchies entail the same intuitively clear taxonomic properties as the predicate $sp$.

**Theorem 11** Let $H$ be an acyclic action diagram, whose corresponding action hierarchy is $H^*$. Then, for any action terms $a_1, a_2$, object variable $x$ and situation argument $s$,

$$H^* \cup D \models sp^*(a_1, a_2) \supset (Poss(a_1, s) \supset Poss(a_2, s)).$$

**Proof:** According to Def. 12 and Def. 13, the proof is straightforward by using the induction principle for $sp^*$. \hfill $\square$

Moreover, the following lemma will be convenient later.

**Lemma 5** Consider any acyclic action diagram $H$ together with the unique name axioms for actions $D_{una}$, whose corresponding action hierarchy is $H^*$. For any action functions $A_1(\vec{x})$ and $A_2(\vec{y})$, there is a situation-free first-order formula $\phi_{A_1,A_2}(\vec{x}, \vec{y})$ (including $\top$ and $\bot$) whose free object variables are at most among $\vec{x}$ and $\vec{y}$, such that $A_1(\vec{x})$ is a
(distant) specialization of $A_2(\vec{y})$ iff $\phi_{A_1,A_2}(\vec{x},\vec{y})$. That is,

$$\mathcal{H}^* \cup \mathcal{D}_{una} \models sp^*(A_1(\vec{x}), A_2(\vec{y})) \equiv \phi_{A_1,A_2}(\vec{x},\vec{y}).$$

The formula $\phi_{A_1,A_2}$ can be found from $\mathcal{H}$ in finitely many steps.

**Proof:** In our proof, we show how the formula $\phi_{A_1,A_2}$ can be constructed. Let $G = (V,E)$ be the digraph of the given diagram $\mathcal{H}$, and let $\text{max}(A', A)$ be the maximum of the lengths of all the distinct paths from $A'$ to $A$ in $G$. We prove the following property $P(n)$ for any natural number $n$: “For any action function symbols $A', A$ such that $\text{max}(A', A) = n$, we have $n \leq |V|$, and for any distinct free variables $\vec{x}, \vec{y}$, $sp^*(A'(\vec{x}), A(\vec{y})) \equiv \phi_{A',A}(\vec{x},\vec{y})$ for some first-order formula $\phi_{A',A}$ (including $\top$ and $\perp$) with object variables at most among $\vec{x}$ and $\vec{y}$”.

**Base case:** $P(0)$, $\text{max}(A', A)=0$, two sub-cases.

**Case 1:** $A=A'$, since $sp^*$ is reflexive

$$sp^*(A'(\vec{x}), A(\vec{y})) \equiv A'(\vec{x}) = A(\vec{y})$$

$$\equiv \bigwedge_{i=1}^{\vec{x}} x_i = y_i \text{ (by UNA)}.$$  

**Case 2:** $A\neq A'$, and since $\text{max}(A', A)=0$, which means there is no $sp$ path between $A$ and $A'$, then $sp^*(A'(\vec{x}), A(\vec{y})) \equiv \perp$.

**Inductive step:** Assume that $P(j)$ is true for all $j < n$, we prove $P(n)$, where $n > 0$. Consider any action function symbols $A', A$ such that $\text{max}(A', A) = n$. Because $G$ is acyclic and there is at most one edge from one node to another, the action nodes on each path from $A$ to $A'$ are distinct from each other. So, $n \leq |V|$. Next, we collect all direct generalizations of $A'$ in $G$, say $\{A_1, \ldots, A_t\}$, which are (distant) specializations of $A$. Then,

$$sp^*(A'(\vec{x}), A(\vec{y})) \equiv \bigvee_{i=1}^{t} (\exists \vec{x}_i)(sp(A'(\vec{x}), A_i(\vec{x}_i)) \wedge sp^*(A_i(\vec{x}_i), A(\vec{y}))).$$

For each $i$, $\text{max}(A_i, A) \leq n-1$. By the induction hypothesis, we have $sp^*(A_i(\vec{x}_i), A(\vec{y})) \equiv \phi_{A_i,A}(\vec{x}_i, \vec{y})$ for some situation-free first-order formula $\phi_{A_i,A}$ whose free variables are at most among $\vec{x}_i$ and $\vec{y}$. In $\mathcal{H}$, for each $i$ we have

$$sp(A'(\vec{x}), A_i(\vec{x}_i)) \equiv \phi_{A',A_i}(\vec{x}, \vec{x}_i),$$
where $\phi_{A',A_i}$ is a situation-free first-order formula. Let

$$\phi_{A',A}^{A'}(\bar{x}, \bar{y}) = \bigvee_{i=1}^{t}(\exists \bar{x}_i)[\phi_{A',A_i}(\bar{x}, \bar{x}_i) \land \phi_{A_i,A}(\bar{x}_i, \bar{y})],$$

then $P(n)$ is proved. Notice that $n \leq |V|$; hence such first-order formula can always be obtained in finitely many steps. \hfill \Box

**Example 10** We continue with Example 9. Most of the first-order formulas $\phi_{A_1,A_2}(\bar{x},\bar{y})$ equivalent to $sp^*(A_1(\bar{x}),A_2(\bar{y}))$ are straightforward (either $\top$, $\bot$, or the same as the axioms of $sp$), except for $sp^*(\text{drive}(p,v,o,d),\text{move}(obj,orig,dest))$ for any free variable $p,v,o,d,obj,orig,dest$. By using Def. 13 and the axioms given in Example 9, we have

$$sp(\text{drive}(p,v,o,d),\text{move}(obj,orig,dest)) \supset sp^*(\text{drive}(p,v,o,d),\text{move}(obj,orig,dest))$$

and

$$sp(\text{drive}(p,v,o,d),\text{travel}(p,o,d)) \land sp(\text{travel}(p,o,d),\text{move}(obj,orig,dest)) \supset$$

$$sp^*(\text{drive}(p,v,o,d),\text{move}(obj,orig,dest)),$$

and there are no other non-equivalent axioms for $\text{drive}$ and $\text{move}$. Hence,

$$sp^*(\text{drive}(p,v,o,d),\text{move}(obj,orig,dest))$$

$$\equiv sp(\text{drive}(p,v,o,d),\text{move}(obj,orig,dest)) \lor$$

$$sp(\text{drive}(p,v,o,d),\text{travel}(p,o,d)) \land sp(\text{travel}(p,o,d),\text{move}(obj,orig,dest))$$

$$\equiv v = obj \land o = orig \land d = dest \lor p = obj \land o = orig \land d = dest,$$

which can be simplified as: for any variables $p,v,o,d,obj$,

$$sp^*(\text{drive}(p,v,o,d),\text{move}(obj,o,d)) \equiv p = obj \lor v = obj.$$

\hfill \Box

We also have the following corollary using Lemma 5.

**Corollary 4** Consider any acyclic action diagram $\mathcal{H}$ together with the unique name axioms for actions $\mathcal{D}_{una}$, whose corresponding action hierarchy is $\mathcal{H}^*$. Then, there is a situation-free first-order formula $\Phi$ (including $\top$ and $\bot$), such that

$$\mathcal{H}^* \cup \mathcal{D}_{una} \models sp^*(a,a') \equiv \Phi(a,a').$$

The formula $\Phi$ can be found from $\mathcal{H}$ in finitely many steps.
Chapter 4. Reasoning about Large Taxonomies of Actions

Proof: There are only finitely many distinct action functions in $H^*$, say $A_1(x_1),\ldots,A_k(x_k)$. By using Lemma 5, for each pair of action functions $A_i$ and $A_j$ ($i = 1..k, j = 1..k$), we can find a situation-free first-order formula $\Phi_{i,j}$ such that

$$H^* \cup D_{una} \models sp^*(A_i(x_i), A_j(x_j)) \equiv \Phi_{i,j}(x_i, x_j).$$

Then, let $\Phi(a, a')$ be $\bigwedge_{i=1..k, j=1..k} (\exists x_i, x_j. a = A_i(x_i) \land a' = A_j(x_j) \land \Phi_{i,j}(x_i, x_j))$. □

4.3 Modular BATs

Our goal is to provide a more compact specification of a BAT based on a given hierarchy of actions. We will call such a modified BAT a modular BAT and denote it as $D^H$, where

$$D^H = D_{ap} \cup D_{ss}^H \cup D_{s0}^H \cup \Sigma \cup D_{una}.$$ 

Here, $D_{s0}^H = D_{s0} \cup H^*$, in which $H^*$ is the action hierarchy and $D_{s0}$ describes the usual initial state, the same as Reiter’s initial theory, and $D_{ss}^H$ is the new class of SSAs specified based on $H^*$. In the sequel, let $sp^*_a(a, a')$ be an abbreviation for either $sp^*(a, a')$ or $a = a'$ in Formula (4.9) below.

The new syntactic form of SSAs in $D_{ss}^H$ can be different from Reiter’s format in $D_{ss}$. Intuitively, instead of tediously repeating each individual action on the right-hand side (RHS) of a SSA for a fluent, say $F(x, do(a, s))$, one can take advantage of the action hierarchies and describe the effects of the whole class of action functions at once. One can say that all those actions which are (distant) specializations of some generic action $A(y)$ (actions from the branch going out of $A(y)$), except those (distant) specializations of some other generic actions, say $A(y_l)$ for $1 \leq l \leq h$ (i.e., excluding actions from some branches), can cause the same (positive or negative) effects on $F$ under certain conditions. By doing so, we can represent the effects of actions more compactly. Later we will see that this new form of SSAs also leads to significant computational advantages.
It is convenient to use the following notation related with a fluent $F(\vec{x}, s)$: for any variable vector $\vec{y}_l \ (l \geq 0)$, let $\vec{z}_l = \vec{y}_l - \vec{x}$ (i.e., $\vec{z}_l$ are the new variables mentioned in $\vec{y}_l$ but not in $\vec{x}$). Note that, on the RHS of the SSA of $F(\vec{x}, s)$, those new variables $\vec{z}_l$ need to be existentially quantified. Formally speaking, the modified SSA of a relational fluent $F(\vec{x}, s)$ has the form

$$F(\vec{x}, do(a, s)) \equiv \bigvee_i \psi_i^+(\vec{x}, a, s) \lor F(\vec{x}, s) \land \neg \bigvee_j \psi_j^-(\vec{x}, a, s),$$

(4.7)

where each $\psi_i^+(\vec{x}, a, s)$ or $\psi_i^-(\vec{x}, a, s)$ has either the syntactic form

$$\exists \vec{z}. a = A(\vec{y}) \land \gamma(\vec{x}, \vec{z}, s),$$

(4.8)

or the following syntactic form

$$(\exists \vec{z}_0)[sp^*(a, A(\vec{y}_0)) \land \gamma(\vec{x}, \vec{z}_0, s) \land \bigwedge_{l=1}^h \neg (\exists \vec{z}_l) sp^*_l(a, A_l(\vec{y}_l))].$$

(4.9)

In (4.9), $\gamma$ is a formula uniform in $s$ that has $\vec{x}, \vec{z}_0, s$ at most as its free variables. Notice that whenever $\vec{z}_l \ (l \geq 0)$ is empty, then there is no existential quantifier over $\vec{z}_l$. In addition, in (4.9), when index $h = 0$, the conjunction over $l$ does not exist. One can prove that axiomatizers can always write modified SSAs in $D^H$ with $h = 0$ in (4.9), i.e., without using the negative statements. This due to an intuitively obvious observation that when one talks about the effects of an action, it does no matter what this action “is not”, but it matters what this action specializes. From the DAG interpretation of $sp^*$, this observation is obvious too. However, with negation (when $h > 0$), axiomatizers gain flexibility of writing SSAs that can deliver more computational advantages. Details can be found in the next section (see Example 13).

Other classes of axioms such as the initial theory $D_{S0}$, the precondition axioms $D_{ap}$, the foundational axioms $\Sigma$ and unique name axioms for actions $D_{una}$ have the same formats as in [141]. It is easy to see that a modular BAT $D^H$ differs from Reiter’s BAT $D$ in the following aspects: $D^H_{S0}$ includes the action hierarchy $H^*$ and can use $sp^*$ to specify SSAs for classes of actions, while $D_{ss}$ enumerates each action individually.
However, according to Lemma 6 (with a constructive proof), the modular BATs and Reiter's BATs are related.

**Lemma 6** For any $D^H_{ss}$ in a modular BAT $D^H$, there exists a class $D_{ss}$, a collection of SSAs of the syntactic form given in Reiter's BAT [141] for all fluents appeared in $D^H$. In particular, for each $F(\vec{x}, do(a, s)) \equiv \phi_F(\vec{x}, a, s)$ in $D^H_{ss}$ where $\phi_F$ may have occurrences of the predicate $sp^*$, there exists a formula $F(\vec{x}, do(a, s)) \equiv \phi'_F(\vec{x}, a, s)$ in $D_{ss}$, which does not mention $sp^*$ and satisfies $D^H \models F(\vec{x}, do(a, s)) \equiv \phi'_F(\vec{x}, a, s)$.

**Proof:** Assume that $D^H$ includes $k$ action functions, say $A_1(\vec{v}_1), \ldots, A_k(\vec{v}_k)$, in total. For each relational fluent $F(\vec{x}, s)$, assume that its SSA in $D^H$ is of the form (4.7) whose positive and negative effect conditions have the syntactic form (4.9), then we substitute $a$ with each action function, say $A_i(\vec{v}_i)$ (without loss of generality, we assume that variables in $\vec{v}_i$ are all new variables never used in the SSA of $F$), and on the RHS obtained by this substitution from the SSA of $F(\vec{x}, do(A_i(\vec{v}_i), s))$, replace every occurrence of $sp^*$ (that has two action functions as arguments) with its equivalent first-order formula (that exists according to Lemma 5). This replacement results in an axiom of the following form

$$F(\vec{x}, do(A_i(\vec{v}_i), s)) \equiv \psi^+_i(\vec{x}, \vec{v}_i, s) \lor F(\vec{x}, s) \land \neg \psi^-_i(\vec{x}, \vec{v}_i, s).$$

Whenever $\psi^+_i(\vec{x}, \vec{v}_i, s)$ ($\psi^-_i(\vec{x}, \vec{v}_i, s)$, respectively) are consistent conditions (situation calculus formulas uniform in $s$), $A_i(\vec{v}_i)$ has a positive effect (a negative effect) on $F$ under such condition. Hence, the following yields the logically equivalent SSA of $F$ in the usual BAT of [141]:

$$F(\vec{x}, do(a, s)) \equiv [\bigvee_{i=1}^{k}(\exists \vec{v}_i)(a = A_i(\vec{v}_i) \land \psi^+_i(\vec{x}, \vec{v}_i, s))] \lor F(\vec{x}, s) \land \neg[\bigvee_{j=1}^{k}(\exists \vec{v}_j)(a = A_j(\vec{v}_j) \land \psi^-_j(\vec{x}, \vec{v}_j, s))].$$

Notice that the above axiom can be simplified by removing inconsistent clauses. Hence the lemma is proved. □

We then have the following important property:

**Theorem 12** For each $D^H$, there exists an equivalent $D$ of the format given in [141],
where equivalence means that for any first-order regressable sentence $W$ that has no occurrences of the predicate $sp$, $D^H \models W$ iff $D \models W$.

Proof: Use Lemma 6 and Lemma 5 to get rid of $sp^*$ in $D^H$. \qed

Here we provide some examples for the new way of representing SSAs, and compare them with Reiter’s format.

Example 11 We continue with Example 8 (recall Figure 4.1). Now, consider a fluent $fCooked(x, s)$ (food $x$ is cooked in the situation $s$), the modular BAT version of its SSA could be:

$$fCooked(x, do(a, s)) \equiv (\exists y) sp^*(a, cook(x, y)) \lor fCooked(x, s).$$

Another example is a SSA for the fluent $dirtyVes(x, s)$ (it will be false after washing a vessel $x$ in some manner, or it will be true when $x$ is used to prepare food or drink):

$$dirtyVes(y, do(a, s)) \equiv dirtyVes(y, s) \land \neg sp^*(a, wash(y)) \lor (\exists x) sp^*(a, cook(x, y)) \lor (\exists x) a = makeSalad(x, y) \lor (\exists x) sp^*(a, prepDrink(x, y)).$$

The Reiter’s SSA for fluent $fCooked(x, s)$ is the following (with a bigger taxonomy of actions, it will be much longer):

$$fCooked(x, do(a, s)) \equiv$$

$$(\exists y)[ a = cook(x, y) \lor a = lowOilCook(x, y) \lor a = steam(x, y) \lor a = boil(x, y) \lor a = stew(x, y) \lor a = broil(x, y) \lor a = bake(x, y) \lor a = roast(x, y) \lor a = ovenCook(x, y) \lor a = pressureCook(x, y) \lor a = oilyCook(x, y) \lor a = fry(x, y) \lor a = deepFry(x, y) \lor a = stir(x, y) \lor a = parboil(x, y) \lor a = grill(x, y)] \lor fCooked(x, s).$$
Here is Reiter’s representation of the SSA for fluent $\text{dirtyVes}(y,s)$:

\[
\text{dirtyVes}(y,\text{do}(a,s)) \equiv \exists x [a = \text{cook}(x,y) \lor a = \text{lowOilCook}(x,y) \lor a = \text{oilyCook}(x,y) \lor a = \text{steam}(x,y) \\
\lor a = \text{boil}(x,y) \lor a = \text{stew}(x,y) \lor a = \text{broil}(x,y) \lor a = \text{bake}(x,y) \\
\lor a = \text{ovenCook}(x,y) \lor a = \text{roast}(x,y) \lor a = \text{pressureCook}(x,y) \lor a = \text{fry}(x,y) \\
\lor a = \text{deepFry}(x,y) \lor a = \text{stir}(x,y) \lor a = \text{reheat}(x,y) \lor a = \text{microwave}(x,y) \\
\lor a = \text{makeHotDr}(x,y) \lor a = \text{makeColdDr}(x,y) \lor \text{dirtyVes}(y,s) \land a \neq \text{wash}(y,s)].
\]

The definitions of the regression operator and the regresasurable sentences in $\mathcal{D}^H$ are all the same as demonstrated in [141]. In particular, for any formula of the form $\text{sp}^*(\alpha_1,\alpha)$, since it is a situation-independent atom, $\mathcal{R}[\text{sp}^*(\alpha_1,\alpha)] = \text{sp}^*(\alpha_1,\alpha)$, and later we will define the extended regression $\mathcal{E}$ to replace it with an equivalent first-order situation independent formula. Similar to the regression theorem [141], we have

\[
\mathcal{D}^H \models W \iff \mathcal{D}^H_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W]
\]

for any regresasurable sentence $W$.

Now, let $\mathcal{E}[\mathcal{R}[W]]$ (called the extended regression of $W$) be the operator that eliminates all occurrences (if any) of the form $\text{sp}^*(\alpha_1,\alpha_2)$ for some action terms $\alpha_1$ and $\alpha_2$ in $\mathcal{R}[W]$ in favor of some first-order formulas $\phi$ that exists according to Lemma Corollary 4. Then, we have:

**Theorem 13** For each $\mathcal{D}^H$ and for any first-order regresasurable sentence $W$,

\[
\mathcal{D}^H \models W \iff \mathcal{D}^H_{S_0} \cup \mathcal{H} \cup \mathcal{D}_{una} \models \mathcal{E}[\mathcal{R}[W]].
\]

This theorem is important because of the following reason. Although $\mathcal{D}^H_{S_0} \cup \mathcal{D}_{una}$ (and hence $\mathcal{D}^H$) includes the second-order definition of the predicate $\text{sp}^*$, all occurrences
of \( sp^* \) in sentence \( R[W] \) can be replaced by first-order sentences in finitely many steps according to the Lemma 5. Consequently, one can use regression in our modular BATs to reduce projection and executability problems to theorem proving in first-order logic only.

4.4 Advantages of Modular BATs

Using action hierarchies and specifying BATs modularly not only provides a compact way of representing effects of actions, but sometimes leads to a more computationally efficient (than Reiter’s) solution of the projection problem.

Example 12 Continuing with Example 11, we now consider a ground action \( \alpha = deepFry(Egg_1, FryingPan_1) \), and the regression of \( fCooked(Egg_1, do(\alpha, S_0)) \). Using Reiter’s SSA for this fluent, regression involves 16 recursive calls to go from disjunctions on the RHS to atoms, and then regression involves checking 16 equality clauses between actions when regressing on the positive conditions in the SSA of \( fCooked \) (see the axiom above). Using the modular BAT, extended regression of the positive conditions involves only one step of regression for predicate \( sp^* \), and finally the replacement of \( sp^*(\alpha, cook(Egg_1, y)) \) with the corresponding first-order formula, i.e., the operator \( E \), takes at most four steps of recursive computation (see Figure 4.1).

Apart from specific examples, let us discuss in general the following problems: when we can actually gain computational advantages using action hierarchies and how much we can gain, whether there is any possible computational disadvantage in using action hierarchies alone, and if so, whether it can be avoided.

According to the definition in the previous section, the digraph of an acyclic action diagram is in fact a directed acyclic graph (DAG). Computing the first-order formula equivalent to \( sp^*(A_1(\vec{x}), A_2(\vec{y})) \) for any pair of action functions \( A_1(\vec{x}) \) and \( A_2(\vec{y}) \) is similar to finding all paths from \( A_1 \) to \( A_2 \) in the corresponding digraph. The latter problem has
the computational complexity of $\Theta(p)$ where $p$ is the number of all the distinct edges in the digraph on any path from $A_1$ to $A_2$, and therefore has a computational complexity of $O(e)$ where $e$ is the number of all edges in the digraph (i.e., $e$ is the number of $sp$ axioms in $\mathcal{H}$). As a consequence, this yields the following encouraging and important result.

We start with the case $h = 0$ in (4.9). Let $\phi(a, \bar{x}, s)$ denote $\big(\exists \bar{z}_0\big)[sp^*(a, A(\bar{y}_0)) \land \gamma(\bar{x}, \bar{z}_0, s)]$. In general, to provide an equivalent SSA of $F(\bar{x}, s)$ in Reiter’s representation, $\phi(a, \bar{x}, s)$ has to be replaced by an uniform formula $\psi(a, \bar{x}, s)$ of the form $\big(\exists \bar{z}_0\big)[\psi(a, \bar{y}_0) \land \gamma(\bar{x}, \bar{z}_0, s)]$. Here, $\psi(a, \bar{y}_0)$ might have the form $\big(a = A(\bar{y}_0) \lor \bigvee_{i=1}^{n_A-1} \big(\exists \bar{z}_i\big)(a = A_i(\bar{y}_i) \land \psi_i(\bar{y}_0, \bar{y}_i))\big)$, where each $A_i(\bar{y}_i)$ ($1 \leq i \leq n_A - 1$) is a specialization of $A(\bar{y}_0)$ under the condition $\psi_i(\bar{y}_0, \bar{y}_i)$, $n_A$ is the total number of specializations of $A$. The formula $\psi(a, \bar{y}_0)$ is a logically equivalent replacement of $sp^*(a, A(\bar{y}_0))$ in $\phi$ (see Lemma 6). Let the action diagram $\mathcal{H}$ in $\mathcal{D}^H$ be acyclic and the corresponding action digraph rooted at $A$ have a tree structure (the most general actions are considered as roots and the most specialized actions are considered as leaves). Then, the computational time of extended regression $E[R[sp^*(\alpha, A(\bar{t}', \bar{z}_0))]]$ in the clause $\phi(\alpha, \bar{t}, S)$, for any object terms $\bar{t}$ and any situation term $do(\alpha, S)$, is no worse than and (sometimes) can be exponentially faster than computational time of Reiter’s regression on $\psi(a, \bar{t}, S)$.

**Theorem 14** If the sub-tree rooted at $A$ in the digraph of $\mathcal{H}$ is a complete $k$-ary tree ($k \geq 2$) with $n_A$ action functions as its nodes, the computational complexity of extended regression $E[R[sp^*(\alpha, A(\bar{t}', \bar{z}_0))]]$ is $\Theta(\log_k n_A)$, while the computational complexity of Reiter’s regression $R[\psi(a, \bar{t}, S)]$ on an equivalent replacement $\psi(a, \bar{t}, S)$ is $\Theta(n_A)$.

**Proof:** When a DAG of an action hierarchy has a tree structure or a forest structure, there is at most one path between any two action functions. In particular, assume that the digraph of the action hierarchy is a complete $k$-ary ($k \geq 2$) tree structure and consider the regression of $F(\bar{t}, do(\alpha, S))$ for any action term $\alpha$ and situation term $S$. To specify the equivalent SSA in Reiter’s format, $\phi(\alpha, \bar{t}, S)$ needs to be replaced by $\psi(\alpha, \bar{t}, S)$. It is
easy to see that one-step regression of the above clause in Reiter’s format takes $\Theta(n_A)$ steps (subsequently, additional time is required to regress recursively $R[\gamma(\vec{t}, \vec{z}_0, S)]$). To perform one-step regression using the modular BAT format, it is sufficient to regress $sp^*(\alpha, A(\vec{t}', \vec{z}_0)) \land \gamma(\vec{t}, \vec{z}_0, S)$ (which takes $\Theta(1)$ time, excluding again the time that subsequently required to compute recursively $R[\gamma(\vec{t}, \vec{z}_0, S)]$) and then replace $sp^*(\alpha, A(\vec{t}', \vec{z}_0))$ with the equivalent first-order formula. This last replacement step can be considered as finding the path from $\alpha$ to $A_0$ with the computational complexity of $\Theta(\log_k n_A)$. Finally, regression makes the same number of recursive calls in both cases: the formula $\gamma$ is the same.

However, if we do not allow the usage of (in)equality between action terms (e.g., $a = A_i$) in $D_{ss}^H$, we may (sometimes) lose computational advantages when the effects of actions for some fluents involve very few actions in a large taxonomy or when the structure of a DAG is not a forest, especially if it is close to a complete DAG. Since a dense DAG of $n$ nodes has at most the order of $n^2$ edges (a complete DAG of $n$ nodes has $n(n - 1)/2$ edges in total), computing the first-order formula equivalent to the predicate $sp^*$ has complexity $O(n^2)$. To avoid such computational disadvantages, we can easily use both (in)equality between action terms and predicate $sp^*$ in modular BATs. Whenever the (sub)digraph rooted at some action function symbol $A$ has a tree structure (even mostly a tree structure with a few extra edges) and most of its specializations have the same effects under certain common conditions on some fluent $F$, one can use the $sp^*$ predicate for $A$ to gain both the computational advantage and the advantage of compact representation when writing the SSA for $F$. Otherwise, one can use the (in)equality format to avoid computational disadvantages.

From the knowledge engineering point of view, using action hierarchies and modular BAT representation may also allow us to manage the update of dynamic systems with large taxonomies of actions more easily. First, with the clearly specified relationships between actions, axiomatizers can avoid the possibility of missing effects of some actions
when the number of actions becomes very large. Secondly, when a new action is introduced (or a part of action hierarchy $H^*$ has to be elaborated) and a BAT needs to be updated, it is possible to update the action diagram axioms by adding (or deleting respectively) a few axioms only and leave most of the axioms in the BAT untouched. For example, if a new specialization of cooking approach is invented, we only need to add a specialization axiom for the new action function and action function $cook()$ and leave the modular BAT axioms (especially for the SSAs of the existing fluents) unmodified.

Now, we illustrate the advantage of using the negated component in clause (4.9) (i.e., allowing $h>0$).

**Example 13** We continue with Example 12. Consider a fluent $nonBBQ(x,s)$: $x$ is cooked without grilling. Its SSA in $D^H$ can be written without negation in (4.9), i.e. when $h=0$:

\[
nonBBQ(x, do(a,s)) \equiv (\exists y)[sp^*(a, oilyCook(x,y)) \lor sp^*(a, ovenCook(x,y)) \lor a=steam(x,y) \lor a=stew(x,y) \lor
sp^*(a, boil(x,y))] \lor nonBBQ(x,s) \land \neg(\exists y)a=grill(x,y)
\] (4.10)

Alternatively, it can be written with negation (i.e., $h>0$) as:

\[
nonBBQ(x, do(a,s)) \equiv (\exists y)[sp^*(a, oilyCook(x,y)) \lor
sp^*(a, lowOilCook(x,y)) \land \neg(\exists z)(a=lowOilCook(x,z) \lor a=grill(x,z)) \lor nonBBQ(x,s) \land \neg(\exists y) a=grill(x,y)
\] (4.11)

Consider $\mathcal{E}[R[nonBBQ(Egg_1, do(\alpha, S_0))]]$, extended regression where $\alpha$ is the same as in Example 12. It takes one step less using Formula (4.11) than using Formula (4.10) during regression (regardless the quantifiers). Clearly, the more branches $lowOilCook(x,y)$ has that have positive effects on $nonBBQ(x,s)$ without extra context conditions, the more computational advantage we can obtain by allowing $h \geq 0$ and using Formula (4.11) during regression. □
4.5 How to Construct a Taxonomy of Actions

As we see, hierarchies of actions can lead to important computational advantages. An important practical question remains how an axiomatizer should approach the problem of constructing a hierarchy of actions given only a set of effect axioms which specify for each fluent what actions have a (positive or negative) effect on the fluent. In a somewhat similar vein, the construction of BATs proposed in [141] starts from effect axioms and demonstrates that under the causal completeness assumption, SSAs can be constructed from effect axioms. We continue to consider only action diagrams $H$ with monotonic inheritance of effects. In this subsection, without loss of generality, we may assume that all the variables used below in the action functions and the fluents are distinct from each other. Consider a BAT $D$ which includes a set of $n$ action functions, say $\{A_i(\vec{x}_i) \mid i = 1..n\}$, and $m$ fluents, say $\{F_j(\vec{y}_j) \mid j = 1..m\}$, that might be affected by any of the above actions. For any action function $A(\vec{x})$, without loss of generality, we assume that its positive effect axiom for any relational fluent $F(\vec{y},s)$ has the syntactic form
\[
\psi_{A,F}^+(\vec{x},\vec{y},s) \supset F(\vec{y},do(A(\vec{x})),s), \tag{4.12}
\]
and its negative effect axiom for $F(\vec{y},s)$ is of the form
\[
\psi_{A,F}^-(\vec{x},\vec{y},s) \supset \neg F(\vec{y},do(A(\vec{x})),s). \tag{4.13}
\]

**Definition 14** For an action function $A(\vec{x})$ and a fluent $F(\vec{y},s)$, which has effect axioms of the form (4.12,4.13), we say that an action $A$ has effect on a relational fluent $F$ (or $F$ could be affected by $A$) iff either $\not\models \psi_{A,F}^+(\vec{x},\vec{y},s) \equiv F(\vec{y},s)$ or $\models \psi_{A,F}^-(\vec{x},\vec{y},s) \equiv \neg F(\vec{y},s)$. For any action function $A(\vec{x})$, a special meta-function $N_e(A)$ is used to represent the number of fluents that can be affected by $A$.

For any two action functions $A_1(\vec{x}_1)$ and $A_2(\vec{x}_2)$, we say that $A_1$ causes no less effects than $A_2$ iff there exists no fluent such that $A_1$ has no effect on it but $A_2$ has. We say that $A_1$ causes more effects than $A_2$ iff $A_1$ has no less effects than $A_2$ and there exists at least one fluent such that $A_1$ has an effect on it but $A_2$ does not. \[\square\]
Note that if $A_1$ causes more effects than $A_2$, then $N_e(A_1) > N_e(A_2)$; however, it is not necessarily true the other way around: actions might affect different sets of fluents. Given effect axioms, for any pair of actions $A_1, A_2$, a straightforward linear time $O(m)$ procedure can check whether $A_1$ causes more effects than $A_2$.

We would like to provide general guidelines on how an axiomatizer can construct an action diagram $H$ for $D$. Under the assumption of monotonic inheritance, if $A_1$ is a specialization of $A_2$, then it causes no less effects than $A_2$ and $N_e(A_1) \geq N_e(A_2)$. Thus, to return a set $H$ that represents an action diagram $H$, it is enough to start with generic actions $A$ that have the smallest value of $N_e(A)$ and proceed towards more specialized actions checking on each iteration if the next action we consider is a specialization of one of the previously considered actions.

1. Sort the action functions and get the sequence $A_1(\vec{x}_1), \ldots, A_n(\vec{x}_n)$ such that $N_e(A_{i_1}) \leq N_e(A_{i_2})$ for $i_1 < i_2$.

2. Initially, let $i = 2$ (index $1 \leq i \leq n$) and $H = \emptyset$.

3. If $i > n$, then return $H$ and terminate; else assign $j = i$ and continue: look for $A_j$’s that are generalizations of $A_i$.

4. Decrement $j = j - 1$. If $j = 0$ (i.e., all candidates $A_j$ have been already considered), then increment $i = i + 1$ and go to step 3 (i.e., take a next action $A_{i+1}$ from the sorted sequence we obtained at step 1); else if $N_e(A_i) = N_e(A_j)$, go to step 4; else continue.

5. For any pair of indices $i, j$ such that $1 \leq j \leq i - 1$, if there is a path in $H$ from $i$ to $j$, then we already know that $A_i$ is a specialization of $A_j$ and because specialization is a transitive relation there is no need to add a new directed edge from $A_i$ to $A_j$ and we can go to step 4; else continue.
6. If $A_i(\vec{x}_i)$ is a specialization of $A_j(\vec{x}_j)$ under first-order condition $\phi_{i,j}$, then update
   
   \[ H = H \cup \{(A_i(\vec{x}_i), A_j(\vec{x}_j), \phi_{i,j})\} \]
   and go to step 4; else, go to step 4 immediately.

To implement the last step for any two action functions $A_i(\vec{x}_i)$ and $A_j(\vec{x}_j)$, provided
that axiomatizers are able to write effect axioms of the form (4.12,4.13), we formulate
the following principles to determine whether or not $A_i(\vec{x}_i)$ is a specialization of $A_j(\vec{x}_j)$
under some condition $\phi$.

a. If $A_i$ causes more effects than $A_j$, “guess” a first-order formula $\phi$, whose free variables
   include at most $\vec{x}_j$ and $\vec{x}_i$, such that for any relational fluent $F(\vec{y}, s)$ that could be
   affected by both $A_i$ and $A_j$,
   
   \[
   D \models \phi \supset \text{Poss}(A_i(\vec{x}_i), s) \supset \text{Poss}(A_j(\vec{x}_j), s),
   \]
   
   \[
   D \models \phi \supset (\psi_{A_i,F}(\vec{x}_i, \vec{y}, s) \equiv \psi_{A_j,F}(\vec{x}_j, \vec{y}, s)),
   \]
   
   \[
   D \models \phi \supset (\psi_{A_i,F}(\vec{x}_i, \vec{y}, s) \equiv \psi_{A_j,F}(\vec{x}_j, \vec{y}, s)).
   \]

   If one can find such $\phi$, then $A_i$ is a specialization of $A_j$ under the condition $\phi$.

b. Otherwise, $A_i$ is not a specialization of $A_j$.

Note that for any action functions $A_1(\vec{x})$, $A_2(\vec{y})$ and any first-order formula $\phi$, each tuple
$(A_1(\vec{x}), A_2(\vec{y}), \phi)$ in the returned set $H$ corresponds to an axiom $sp(A_1(\vec{x}), A_2(\vec{y})) \equiv \phi$,
and the collection of all these axioms results in an action diagram $\mathcal{H}$.

In general, to determine whether one action is a specialization of another under certain
condition is undecidable. Hence, the axiomatizers have to observe the preconditions and
effects of actions, guess formula $\phi$ (using their intuition), and construct action diagrams
manually. In the future, we would like to consider whether it is possible to generate
action diagrams automatically in some special cases.
4.6 Discussion and Future Work

There are a few papers related to our work that we would like to mention. Lifschitz and Ren in [102] consider modular theories in the propositional action representation language $C+$, and address the problem of the development of libraries of reusable, general-purpose knowledge components. In comparison to them, we explore how to manage a large number of actions in the predicate logic using a hierarchical representation for actions in the situation calculus. We propose a representation, which not only facilitates writing axioms succinctly, but for realistic taxonomies can also gain computational advantages in solving the projection problem. We will further explore connections of our modular representation with the knowledge management system [10].

Kautz and Allen in [93] and Kautz in the subsequent paper [92] develop frameworks for plan recognition using hierarchies of plans, in which primitive actions and plan instances belong to certain event types represented as unary predicates, and a hierarchy of plans is a collection of restricted-form axioms specifying relationships between various event types. They concentrate on formalizing and reasoning about taxonomic relationships among actions and plans. However, their axiomatizations of actions (preconditions and effects of the actions) are limited as they do not address the projection problem.

Keneiwa and Tojo in [91] give an ontological framework to represent actions/events and their hierarchical relationships in information systems using an order-sorted second-order logic. In this framework, events (or actions) are represented as predicates rather than terms, and the authors consider taxonomical reasoning about relationships between events rather than reasoning about effects of actions. The authors do not provide axiomatizations of the dynamic aspects of actions and do not explore computational properties of their framework.

Devanbu and Litman in [43] propose a plan-based knowledge representation and reasoning system, called CLASP (CLAssification of Scenarios and Plans). CLASP extends the notions of subsumption from terminological languages to plans by allowing the con-
struction of plans from concepts corresponding to actions and using plan description forming operators for choice, sequencing and looping (similar to propositional dynamic logic). All actions in CLASP are represented in the style of STRIPS [47], which is less expressive than general Reiter’s BATs and our modular BATs.

Our formalism is very different from all the papers mentioned above [93, 92, 43]. We use a specialization relation between primitive action functions, and provide a formal axiomatization of the dynamic aspects of actions using full predicate logic (hence, our theory is quite expressive). Also, we gain both representational and computational advantages by using the action hierarchies.

The extensive research on Hierarchical Task Networks (HTN) can be traced to the pioneering work of Sacerdoti on ABSTRIPS (see [142]). It considers a completely different recursive decomposition of complex actions (i.e., plans, or non-primitive tasks that can be represented as Golog programs [50]) into constituents. However, it does not explore large taxonomies of primitive actions or whether these taxonomies can provide any computational advantages when solving the projection problem.

Our work is motivated in part by the well-known hierarchies of verbs (full troponym) in WordNet [46]. Exploring connections with other frameworks (e.g., FrameNet, Levin’s taxonomy, VerbNet, etc) in computational linguistics and natural language processing is a possible direction for our future research.

Moreover, currently we consider only hierarchies of primitive actions. In the future, we may also consider hierarchies of complex actions (plans), and explore what criteria should be followed for constructing such hierarchies, and whether we can construct them automatically from the existing hierarchies of primitive actions and our BATs.
Chapter 5

An Order-Sorted Situation Calculus

In this chapter, we propose a theory for reasoning about action and change based on order-sorted predicate logic where one can consider an elaborate taxonomy of objects. We will first discuss our motivation and then provide the description of the language. Again, we are interested in the projection problem: whether a statement is true after executing a sequence of actions. We design a regression operator that takes advantage of well-sorted unification between terms based on a given sort theory so that we may terminate regression earlier whenever it is possible. We show that answering projection queries in our logical theories is sound and complete with respect to that of in Reiter’s basic action theories. Moreover, we demonstrate that our regression operator based on order-sorted logic can provide significant computational advantages in comparison to Reiter’s regression operator. Some results of this work have been published in [69].

5.1 Motivation

Starting from 1970s, many-sorted reasoning and taxonomies gained in popularity in deductive databases and automated reasoning. In particular, McSkimin [119] subdivides a domain into semantic categories, uses them to build a semantic category graph and argues that this semantic graph would provide computational advantages in a query
answering system. These ideas have been implemented in MRPPS (the Maryland Refutation Proof Procedure System) described in [122] and in [118]. In deductive databases, Reiter [138] also uses boolean combination of monadic predicates to express taxonomies of types (each simple type is represented by a monadic predicate). In [138], Reiter also develops a typed unification algorithm and argues that his approach is more suitable for real world databases than the approach in deductive question-answering research that deals with unrestricted first-order databases. More recently, this line of research results in the DL-Lite framework that is designed for efficient query answering with respect to very large ABox [23].

In his influential paper [75] titled “A Logic of Actions”, P. Hayes proposed an outline of a logical theory for reasoning about action based on many-sorted logic with equality. His paper inspired subsequent work on many-sorted logics in AI. In particular, A. Cohn [30, 31] developed expressive many-sorted logic and reviewed all previous work in this area. Reasoning about action and change based on the situation calculus has been extensively developed in [141]. However, it considers a logical language with sorts for actions, situations and just one catch-all sort Object for the rest that remains unelaborated. Surprisingly, even if the idea proposed by Hayes seems straightforward, there is still no formal study of logical and computational properties of a version of the situation calculus with many related sorts for objects in the domain. Perhaps, this is because mathematical proofs of these properties are not straightforward although the intuition is. We undertake this study and demonstrate that reasoning about action with elaborated sorts has significant computational advantages in comparison to reasoning without them. In contrast to an approach to many-sorted reasoning [146, 169, 79] where variables of different sorts range over unrelated universes, we consider a case when sorts are related to each other, so that one can construct an elaborated taxonomy. This is often convenient for representation of common-sense knowledge about a domain.

As an example, we consider a kitchen domain which involves activities such as cutting,
washing, frying, baking, boiling and preparing a drink, etc. It also involves many objects of different sorts (in another word, types). For example, there are cups, cooking vessels, and food, etc. For each sort, it might have subsorts of objects. For example, a cup can be a paper cup, a glass cup or a mug, etc; a plate can be a paper plate, a plastic plate or a pottery plate, etc; a cooking vessel can be a frying pan, a pot or an oven-cooking vessel, etc; and, food can be meat, vegetables or dairy, etc. Assume that an autonomous agent is able to perform cooking activities in this kitchen domain. Whenever it performs an action, proper objects should be involved. For example, it is impossible to fry an egg in a paper cup. It is possible that some object of sort food can be cooked after executing a sequence of actions, while it is obvious that a cup will never be cooked no matter what actions are executed.

Generally speaking, we are usually interested in a comprehensive taxonomic structure for sorts, where sorts may inherit from each other and may have non-empty intersections. Note that even if both many-sorted logic and order-sorted logic can be translated to unsorted logic, using sorted ones can bring about significant computational advantages, for example in deduction, compared to unsorted logic. This was a primary driving force for [167] and [30]. Hence, we consider formulating the situation calculus in an order-sorted (predicate) logic to describe taxonomic information about objects. Based on the newly formulated language of the order-sorted situation calculus, we consider solving the projection problem via regression. We show that regression in the order-sorted situation calculus can benefit from well-sorted unification. One can gain computational efficiency by terminating regression steps earlier when objects of incommensurable sorts are involved.

Note that in this chapter, to distinguish different entailments in different logics, we use symbol $\models^\text{os}$ to represent the logical entailment with respect to a sort theory $T$ in order-sorted logic, symbol $\models^\text{ms}$ to represent the logical entailment in Reiter’s situation calculus (a many-sorted logic with one standard sort $\text{Object}$) as in previous chapters, and
symbol $\models_{\mathbf{L}}$ to represent the logical entailment in unsorted predicate logic.

### 5.2 Order-Sorted Basic Action Theories

We now describe the language of order-sorted situation calculus $\mathbf{L}^{\mathbf{OS}}$. $\mathbf{L}^{\mathbf{OS}}$ includes a set of sorts $\mathbf{Sort} = \mathbf{Sort}_{\text{obj}} \cup \{\top, \bot, \text{Action}, \text{Situation}\}$, where $\top$ represents the whole universe, $\bot$ is the empty sort, $\text{Action}$ is the sort for all actions, $\text{Situation}$ is the sort for all situations, and $\mathbf{Sort}_{\text{obj}}$ is a set of subsorts of $\text{Object}$ including sort $\text{Object}$ itself. Similar to Reiter’s situation calculus, we assume that everything in the whole universe is either an action, or a situation, or an object (that belongs to some subsort of $\text{Object}$), and each sort $\text{Action}$, $\text{Situation}$ or $\text{Object}$ includes infinitely many elements. We also assume that for every sort (except $\bot$) there is at least one ground term (constant) of this sort to avoid the problem with “empty sorts” [59, 171]. Moreover, the number of individual variable symbols of each sort in $\mathbf{Sort}$ is countably infinite. For the sake of simplicity, we do not consider functional fluents here. But, it is straightforward to extend our work to languages that have functional fluents.

In the following, we will define order-sorted basic action theories (order-sorted BATs) and consider dynamical systems that can be described using such order-sorted BATs. An order-sorted BAT $\mathcal{D} = (\mathcal{T}, \mathcal{D})$ includes the following two parts of theories.\(^1\)

- $\mathcal{T}$ is a sort theory based on a finite set of sorts $\mathcal{Q}$ such that $\mathcal{Q} \subseteq \mathbf{Sort}$ and $\{\bot, \top, \text{Object}, \text{Action}, \text{Situation}\} \subseteq \mathcal{Q}$. Moreover, the sort theory includes the following declarations for finitely many predicates and functions:

  1. Subsort declarations of the form $Q_1 \leq Q_2$ for $Q_1, Q_2 \in \mathcal{Q} - \{\top, \text{Action}, \text{Situation}\}$, and subsort declarations: $\text{Object} \leq \top$, $\text{Action} \leq \top$, $\text{Situation} \leq \top$, $\bot \leq \text{Action}$, $\bot \leq \text{Situation}$. Here, we only consider those sort theories whose sort hierar-

\(^1\)Note that the bold $\mathbf{D}$ denotes our modification of Reiter’s BAT, and the calligraphic $\mathcal{D}$ denotes the union of a sort theory $\mathcal{T}$ and a modified BAT $\mathcal{D}$. 
2. One and only one predicate declaration of the form $F : \bar{Q}_{1..n}$ for each $n$-ary relational fluent $F$ in the system, where $Q_i \leq_T Object$ and $Q_i \neq \bot$ for $i = 1..(n-1)$, and $Q_n$ is $Situation$.

3. One and only one predicate declaration for the special predicate $Poss$, that is, $Poss : Action \times Situation$.

4. One and only one predicate declaration of the form $P : \bar{Q}_{1..n}$ for each $n$-ary situation independent predicate $P$ in the system, where $Q_i \leq_T Object$ and $Q_i \neq \bot$ for $i = 1..n$.

5. A special declaration for the equality symbol $= : \top \times \top$. This means that equality is interpreted by the identity relation on the whole universe (domain).

6. One and only one function declaration of the form $A : \bar{Q}_{1..n} \rightarrow Action$ for each $n$-ary action function $A$ in the system, where $Q_i \leq_T Object$ and $Q_i \neq \bot$ for $i = 1..n$. Note that, when $n = 0$, the declaration is of form $A : Action$ for constant action function $A$.

7. One and only one function declaration of the form $f : \bar{Q}_{1..n} \rightarrow Q_{n+1}$ for each $n$-ary ($n \geq 0$) situation independent function $f$ (other than action functions), where each $Q_i \leq_T Object$ and $Q_i \neq \bot$ for each $i = 1..(n+1)$. When $n = 0$, it is a function declaration for a constant, denoted as $c : Q$ for constant $c$ of sort $Q$.

8. One and only one function declaration $do : Action \times Situation \rightarrow Situation$, and $S_0 : Situation$ for the initial situation $S_0$.

The formal definition of the semantics of a sort theory can be found in the background section (see 2.3.2).

- $\mathbf{D}$ is a set of axioms represented using well-sorted sentences with respect to $\mathbb{T}$, which includes the following subsets of axioms.
1. Foundational axioms Σ for situations, which are the same as those in [141].

2. A set $\mathcal{D}_{una}$ of unique name axioms for actions: for any two distinct action function symbols $A$ and $B$ with declarations $A : \vec{Q}_{1..n} \to Action$ and $B : \vec{Q}'_{1..m} \to Action$, we have
   \[(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, \vec{y}_{1..m} : \vec{Q}'_{1..m}). A(\vec{x}_{1..n}) \neq B(\vec{y}_{1..m}).\]
   Moreover, for each action function symbol $A$, we have
   \[(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, \vec{y}_{1..n} : \vec{Q}_{1..n}). A(\vec{x}_{1..n}) = A(\vec{y}_{1..n}) \supset \bigwedge_{i=1}^{n} x_i = y_i.\]

3. The initial theory $\mathcal{D}_{S_0}$, which includes well-sorted (first-order) sentences that are uniform in $S_0$.

4. A set $\mathcal{D}_{ap}$ of precondition axioms for actions represented using well-sorted formulas. For each action symbol $A$, whose sort declaration is $A : \vec{Q}_{1..n} \to Action$, its precondition axiom is of the form
   \[(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, s : Situation). Poss(A(\vec{x}_{1..n}), s) \equiv \Pi_A(\vec{x}_{1..n}, s),\]  
   (5.1)
   where $\Pi_A(\vec{x}_{1..n}, s)$ is a well-sorted formula uniform in $s$, whose free variables are at most among $\vec{x}_{1..n}$ and $s$.

5. A set $\mathcal{D}_{ss}$ of successor state axioms (SSAs) for fluents represented using well-sorted formulas: for each fluent $F$ with declaration $F : \vec{Q}_{1..n} \times Situation$, its SSA is of the form
   \[(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, a : Action, s : Situation). F(\vec{x}_{1..n}, do(a, s)) \equiv \phi_F(\vec{x}_{1..n}, a, s),\]  
   (5.2)
   where $\phi_F(\vec{x}_{1..n}, a, s)$ is a well-sorted formula uniform in $s$, whose free variables are at most among $\vec{x}_{1..n}$ and $a, s$.

Here is a simple example of an order-sorted BAT.

**Example 14** (Transport Logistics) We present an order-sorted BAT $\mathcal{D}$ of a simplified example of logistics. $\mathbb{T}$ includes the following subsort declarations for objects:
\[ \text{MovObj} \leq \text{Object}, \quad \bot \leq \text{City}, \quad \bot \leq \text{Box}, \quad \bot \leq \text{Truck}, \]

\[ \text{Truck} \leq \text{MovObj}, \quad \text{City} \leq \text{Object}, \quad \text{Box} \leq \text{MovObj}, \]

where \( \text{MovObj} \) is the sort of movable objects, and other sorts are self-explanatory. The predicate declarations are

\[ \text{InCity}: \text{MovObj} \times \text{City} \times \text{Situation}, \quad \text{On}: \text{Box} \times \text{Truck} \times \text{Situation} \]

for the fluents \( \text{InCity}(o, l, s) \) and \( \text{On}(o, t, s) \). The function declarations for actions \( \text{load}(b, t) \), \( \text{unload}(b, t) \) and \( \text{drive}(t, c_1, c_2) \) are obvious. That is,

\[ \text{load}: \text{Box} \times \text{Truck} \rightarrow \text{Action} \]
\[ \text{unload}: \text{Box} \times \text{Truck} \rightarrow \text{Action} \]
\[ \text{drive}: \text{Truck} \times \text{City} \times \text{City} \rightarrow \text{Action} \]

To illustrate object functions, we introduce a situation-independent function \( \text{twinCity} \), whose function declaration is

\[ \text{twinCity}: \text{City} \rightarrow \text{City}. \]

Besides \( S_0: \text{Situation} \), the constant declarations may include:

\[ B_1: \text{Box}, \quad B_2: \text{Box}, \quad T_1: \text{Truck}, \]
\[ T_2: \text{Truck}, \quad \text{Toronto}: \text{City}, \quad \text{Boston}: \text{City}. \]

Axioms in \( D_{S_0} \) can be:

\[ \forall x: \text{City}. \exists y: \text{City}. y \neq x \land y = \text{twinCity}(x), \]
\[ \exists x: \text{Box}. \text{InCity}(x, \text{Boston}, S_0), \]
\[ (\forall x: \text{Box}, t: \text{Truck}). \neg \text{On}(x, t, S_0), \]
\[ \text{InCity}(T_1, \text{Boston}, S_0) \lor \text{InCity}(T_2, \text{Boston}, S_0). \]

As an example, the precondition axiom for \( \text{load} \) is:

\[ (\forall x: \text{Box}, t: \text{Truck}, s: \text{Situation}). \text{Poss}(\text{load}(x, t), s) \equiv \]
\[ \neg \text{On}(x, t, s) \land \exists y: \text{City}. \text{InCity}(x, y, s) \land \text{InCity}(t, y, s), \]

and the preconditions for \( \text{unload} \) and \( \text{drive} \) are obvious. As an example, the SSA of fluent \( \text{InCity} \) is:
\[(\forall d: \text{MovObj}, c: \text{City}, a: \text{Action}, s: \text{Situation}). \text{InCity}(d, c, do(a, s)) \equiv \]
\[(\exists t: \text{Truck}, c_1: \text{City}). a = \text{drive}(t, c_1, c) \land (d = t \lor \exists b: \text{Box}. b = d \land \text{On}(b, t, s)) \lor \]
\[\text{InCity}(d, c, s) \land \neg (\exists t: \text{Truck}, c_1: \text{City}). a = \text{drive}(t, c, c_1) \land c \neq c_1 \land (d = t \lor \exists b: \text{Box}. b = d \land \text{On}(b, t, s)),\]
\[(\forall b: \text{Box}, t: \text{Truck}, a: \text{Action}, s: \text{Situation}). \]
\[\text{On}(b, t, do(a, s)) \equiv a = \text{load}(b, t) \]
\[\lor \text{On}(b, t, s) \land \neg a = \text{unload}(b, t).\]

and the SSA of fluent \text{On} is obvious. \qed

5.3 Order-Sorted Regression and Reasoning

Based on the order-sorted BATs described in the previous section, we now consider the central reasoning mechanism in the order-sorted situation calculus.

5.3.1 Order-Sorted Regression

The definition of a regressable formula of \(L^{OS}\) is almost the same as the definition of a regressable formula of \(L_{sc}\) except that instead of being stated for a formula in \(L_{sc}\), it is formulated for a well-sorted formula in \(L^{OS}\). That is,

\textbf{Definition 15} A formula \(W\) of \(L^{OS}\) is \textit{regressable} (with respect to an order-sorted BAT \(D\)) iff

1. \(W\) is a well-sorted first-order formula with respect to \(T\).

2. Each term of sort \text{Situation} mentioned by \(W\) has syntactic form \(do([\alpha_1, \cdots, \alpha_n], S_0)\) for some \(n \geq 0\), where \(\alpha_1, \cdots, \alpha_n\) are terms of sort \text{action}.

3. For each atom of the form \text{Poss}(\alpha, \sigma) mentioned by \(W\), \(\alpha\) has the form \(A(\tilde{t}_{1..n})\) for some \(n\)-ary action function symbol \(A\) of \(L^{OS}\).
4. $W$ does not quantify over situations.

5. $W$ does not mention the predicate symbol $\prec$, nor does it mention any equality atom $\sigma = \sigma'$ for terms $\sigma, \sigma'$ of sort $\text{Situation}$. 

\[ \square \]

Note that in Def. 15, all items, except for the first one, are the same as in [141]. The first item in this definition is important because we would like to avoid meaningless (ill-sorted) queries being regressed without paying attention to sorts of arguments first.

Here is a simple example for regressable formulas in the order-sorted situation calculus.

**Example 15** Consider the BAT $\mathcal{D}$ from Example 14. Let $W$ be

$$\exists d: \text{Box} . d = \text{Boston} \land \text{On}(d, T_1, \text{do}(\text{load}(B_1, T_1), S_0)),$$

which is a (well-sorted) regressable sentence (with respect to $\mathcal{D}$); while

$$\text{On}(\text{Boston}, T_1, \text{do}(\text{load}(B_1, T_1), S_0))$$

is ill-sorted and therefore is not regressable. 

\[ \square \]

The regression operator $\mathcal{R}^{os}$ in $\mathcal{L}^{OS}$ is defined recursively similar to the regression operator in [141]. We take advantages of the sort theory during regression: when there is no well-sorted mgu for equalities between terms that occur in a conjunctive sub-formula of a query, this sub-formula is logically equivalent to false and it should not be regressed any further. It will be shown that this key idea helps eliminate useless sub-trees of a regression tree.

**Definition 16** Consider a regressable formula $W$ in $\mathcal{L}^{OS}$ with respect to a background order-sorted BAT $\mathcal{D} = (\mathbb{T}, \mathcal{D})$. The regression of $W$, $\mathcal{R}^{os}[W]$, is recursively defined as follows. In what follows, $\vec{t}$ and $\vec{\tau}$ are tuples of terms, $\alpha$ and $\alpha'$ are terms of sort $\text{Action}$, $\sigma$ and $\sigma'$ are terms of sort $\text{Situation}$, and $W$ is a regressable formula of $\mathcal{L}^{OS}$. 

1. If $W$ is a non-atomic formula and is of the form $\neg W_1, W_1 \lor W_2, (\exists v : Q).W_1$ or $(\forall v : Q).W_1$, for some regressable formulas $W_1, W_2$ in $L^{OS}$, then

\[
R^{os}[\circ W_1] = \circ R^{os}[W_1] \text{ for constructor } \circ \in \{\neg, (\exists x : Q), (\forall x : Q)\}
\]

\[
R^{os}[W_1 \lor W_2] = R^{os}[W_1] \lor R^{os}[W_2].
\]

2. Else, if $W$ is a non-atomic formula, $W$ is not of the form $\neg W_1, W_1 \lor W_2, (\exists v : Q).W_1$ or $(\forall v : Q).W_1$, but of the form $W_1 \land W_2 \land \cdots \land W_n$ ($n \geq 2$), where each $W_i$ ($i=1..n$) is not of the form $W_i,1 \land W_i,2$ for some sub-formulas $W_i,1, W_i,2$ in $W_i$. After using commutative law for $\land$, without loss of generality, there are two sub-cases:

2(a) Suppose that for some $j$, $j=1..n$, each $W_i$ ($i=1..j$) is of the form $t_{i,1} = t_{i,2}$ for some (well-sorted) terms $t_{i,1}, t_{i,2}$, and none of $W_k$, $k=(j+1)..n$ is an equality between terms (it is possible that $W_k$ is a negation of an equality between terms). In particular, when $j=n$, $\bigwedge_{k=j+1}^n W_k \equiv true$. Then,

\[
R^{os}[W] = \begin{cases} 
W_1 \land W_2 \land \cdots \land W_j \land R^{os}[W'_0] & \text{if there is a well-sorted mgu } \mu \\
 & \text{for } \{\langle t_{i,1}, t_{i,2} \rangle | i = 1..j\}; (5.3) \\
false & \text{otherwise.}
\end{cases}
\]

Here, $W'_0$ is a new formula obtained by applying mgu $\mu$ to $\bigwedge_{k=j+1}^n W_k$ and it is existentially-quantified at front for every newly introduced sort weakened variable in $\mu$. Moreover, note that based on the assumption that we consider meet semi-lattice sort hierarchies only, such mgu is unique if it exists. Notice that $W_1 \land \cdots \land W_j$ needs to be kept, because it carries unification information between terms $\{\langle t_{i,1}, t_{i,2} \rangle | i = 1..j\}$ that cannot be omitted, and the unifiability of these terms does not mean $W_1 \land \cdots \land W_j \equiv true$. We provide an example in Example 16 after this definition.

2(b) Otherwise, $R^{os}[W] = R^{os}[W_1] \land \cdots \land R^{os}[W_n]$.

3. Otherwise, $W$ is atomic. There are four sub-cases.
3(a) Suppose that $W$ is of the form $\text{Poss}(A(\vec{t}), \sigma)$ for an action term $A(\vec{t})$ and a situation term $\sigma$, and the action precondition axiom for $A$ is of the form (5.1). Without loss of generality, assume that all variables in Axiom (5.1) have been renamed (with variables of the same sorts) to be distinct from the free variables (if any) of $W$. Then,

$$\mathcal{R}^{\text{os}}[W] = \mathcal{R}^{\text{os}}[\Pi_A(\vec{t}, \sigma)].$$

3(b) Suppose that $W$ is of the form $\text{F}(\vec{t}, \text{do}(\alpha, \sigma))$ for some relational fluent $F$. Let $F$’s SSA be of the form (5.2). Without loss of generality, assume that all variables in Axiom (5.2) have had been renamed (with variables of the same sorts) to be distinct from the free variables (if any) of $W$. Then,

$$\mathcal{R}^{\text{os}}[W] = \mathcal{R}^{\text{os}}[\phi_F(\vec{t}, \alpha, \sigma)].$$

3(c) Suppose that atom $W$ is of the form $t_1 = t_2$, for some well-sorted terms $t_1, t_2$ (including action functions). Then,

$$\mathcal{R}^{\text{os}}[W] = \begin{cases} W & \text{if there is a well-sorted mgu } \mu \text{ for } \langle t_1, t_2 \rangle; \\ \text{false} & \text{otherwise.} \end{cases}$$

3(d) Otherwise, if atom $W$ has $S_0$ as its only situation term, then

$$\mathcal{R}^{\text{os}}[W] = W.$$
Example 16  Consider the order-sorted BAT $\mathcal{D}$ from Example 14 and the query $W$ from Example 15. Then, it is easy to see that $\mathcal{R}^{os}[W] = \text{false}$, since there is no well-sorted mgu for $(d, Boston)$, where $d : \text{Box}$. Now, let $W_1$ be

$$\neg \forall d : \text{Box}. \, d \neq Boston \lor \neg \text{On}(d, T_1, \text{do}(\text{load}(B_1, T_1), S_0)).$$

$W_1$ is a sentence that is equivalent to $W$. It is easy to check that $\mathcal{R}^{os}[W_1]$ is a formula equivalent to false (with respect to $\mathcal{D}$).

Here is another example to illustrate the necessity of keeping $W_1 \land \cdots \land W_j$ in 2(a) of Def. 16. We consider the regression of a well-sorted formula

$$\text{InCity}(B_1, city, \text{do}(\text{drive}(T_1, Boston, Toronto), S_0)),$$

where $city$ is a free variable of sort $\text{City}$.

$$\mathcal{R}^{os}[\text{InCity}(B_1, city, \text{do}(\text{drive}(T_1, Boston, Toronto), S_0))]$$

$$= \mathcal{R}^{os}[(\exists t : \text{Truck}, \, c_1 : \text{City}). \, \text{drive}(T_1, Boston, Toronto) = \text{drive}(t, c_1, city)$$

$$\land (B_1 = t \lor \exists b : Box. \, b = B_1 \land \text{On}(b, t, S_0)) \lor \text{InCity}(B_1, city, S_0)$$

$$\land \neg (\exists t : \text{Truck}, \, c_1 : \text{City}. \, \text{drive}(T_1, Boston, Toronto) = \text{drive}(t, city, c_1) \land city \neq c_1 \land$$

$$\land (B_1 = t \lor \exists b : Box. \, b = B_1 \land \text{On}(b, t, S_0)))]$$

$$= \cdots \cdots$$

$$= (\exists t : \text{Truck}, \, c_1 : \text{City}). \, \text{drive}(T_1, Boston, Toronto) = \text{drive}(t, c_1, city)$$

$$\land \exists b : Box. \, b = B_1 \land \text{On}(B_1, T_1, S_0) \lor \text{InCity}(B_1, city, S_0)$$

$$\land \neg (\exists t : \text{Truck}, \, c_1 : \text{City}. \, \text{drive}(T_1, Boston, Toronto) = \text{drive}(t, city, c_1) \land$$

$$\land \exists b : Box. \, b = B_1 \land \text{On}(b, T_1, S_0))$$

In the above formula, for instance, if we omit $\text{drive}(T_1, Boston, Toronto) = \text{drive}(t, c_1, city)$ (an example of the component $W_1 \land \cdots \land W_j$ in 2(a) of Def. 16), we will lose the unification information between the variable $city$ and the constant $\text{Toronto}$, and won't be able to maintain logical equivalence between the original formula and the regressed result. □
5.3.2 Reasoning in $L^{OS}$

Given an order-sorted BAT $D = (T, D)$ and the order-sorted regression operator defined above, to show the correctness of the newly defined regression operator, we prove the following theorems similar to that of in [141].

**Theorem 15** If $W$ is a (well-sorted) regressable formula with respect to an order-sorted BAT $D = (T, D)$ in the language of $L^{OS}$, then $R^{os}[W]$ is a well-sorted $L^{OS}$ formula (including $false$) that is uniform in $S_0$. Moreover, $D \models_{os} W \equiv R^{os}[W]$.

**Proof:** The proof of this theorem is very similar to the proof of Theorem 2 in [134]. The only difference is when $W$ is of the form of $W_1 \wedge W_2 \wedge \cdots \wedge W_n$, we have to consider different cases discussed in Def. 16(2) and use the new definition of the modified regression operator. We prove the theorem by induction. Similar to the proof in [134], we define a well-founded ordering relation.

Consider the set $\Lambda$ of all countably infinite sequences of natural numbers with a finite number of non-zero elements, and the following binary relation $\prec_r$ (the reverse lexicographic order) on this set:

$$(\lambda_1, \lambda_2, \ldots) \prec_r (\lambda'_1, \lambda'_2, \ldots) \text{ iff for some } m, \lambda_m \prec_r \lambda'_m, \text{ and for all } n > m, \lambda_n = \lambda'_n.$$  

$(\Lambda, \prec_r)$ is well founded with minimal element $(0, 0, \ldots)$. Again overloading $\prec_r$, we define an ordering on $\Lambda \times \mathbb{N}$ by:

$$(\lambda, n) \prec_r (\lambda', n') \text{ iff } n < n', \text{ or } n = n' \text{ and } \lambda \prec_r \lambda'.$$

The relation $\prec_r$ on $\Lambda \times \mathbb{N}$ is well founded with minimal element $((0, 0, \ldots), 0)$, and therefore can serve as a basis for an inductive proof.

For $n \geq 0$, define the length of the situation term $do([\alpha_1, \ldots, \alpha_n], S_0)$ to be $n$. Whenever $g(t_1, \ldots, t_m)$ is a term, $t_i \ (i = 1..m)$ are its proper subterms. An occurrence of a situation term in a formula $W$ is maximal if its occurrence is not as a proper subterm of some situation term. Given a regresstable formula $W$, define the index of $W$ to be

$ind(W) = ((C, \lambda_1, \lambda_2, \ldots), P), \ldots$
where $C (P, \text{respectively})$ is the total number of logical connectives and quantifiers (atoms of the form $\text{Poss}(\alpha, \sigma)$, respectively) mentioned by $W$; and, for each $m \geq 1$, $\lambda_m$ is the number of occurrences in $W$ of maximal situation term of length $m$.

Now we start to prove the theorem by induction on the index of a regressable sentence. **Base Case:** When the index is $((0, 0, \ldots), 0)$, there are two sub-cases:

(a) $W$ is atomic, has no occurrences of $\text{Poss}(\alpha, \sigma)$, is not of the form $t_1 = t_2$, and $S_0$ is its only term of sort $\text{Situation}$ (if any) mentioned by $W$.

(b) $W$ is atomic and of the form $t_1 = t_2$ for some (well-sorted) terms $t_1$ and $t_2$.

For sub-case (a), according to the definition of the regression operator $R^{os}$, it is obvious that $R^{os}[W]$ is a well-sorted formula uniform in $S_0$ and $D \models^{os} W \equiv R^{os}[W]$. For sub-case (b), when there is a well-sorted mgu $\mu$ for $t_1$ and $t_2$, $R^{os}[W]$ is well-sorted and uniform in $S_0$, and $D \models^{os} W \equiv R^{os}[W]$, since $R^{os}[W] = W$; otherwise, $W$ is unsatisfiable based on the assumption that every sort is not empty, hence $D \models^{os} W \equiv false$, which is also well-sorted and uniform in $S_0$ and $D \models^{os} W \equiv R^{os}[W]$ according to the definition of the regression operator $R^{os}$.

**Inductive Step:** Suppose $((0, 0, \ldots), 0) \prec_r \text{ind}(W)$ and assume that the theorem holds for any regressable formula of index $\prec_r \text{ind}(W)$.

1. If $W$ is atomic, there are several sub-cases:

   (a) $W$ is a regressable $\text{Poss}$ atom of the form $\text{Poss}(A(\vec{t}, \sigma))$. $D \models^{os} W \equiv \Pi(\vec{t}, \sigma)$ by variable renaming assumption. According to the definition of the regression operator $R^{os}$ for $\text{Poss}$ (see Def. 16 case 3(a)), $\Pi(\vec{t}, \sigma)$ is well-sorted and regressable, whose index is smaller than than $\text{ind}(W)$. Then, according to the induction hypothesis, $D \models^{os} \Pi(\vec{t}, \sigma) \equiv R^{os}[\Pi(\vec{t}, \sigma)]$, where the RHS equals to $R^{os}[W]$, is well-sorted and uniform in $S_0$. Overall, we also have $D \models^{os} W \equiv R^{os}[W]$.

   (b) Similarly, it is easy to prove that $R^{os}[W]$ is well-sorted and uniform in $S_0$, such that $D \models^{os} W \equiv R^{os}[W]$ according to Def. 16 case 3(b), when $W$ is a relational fluent of the form $F(\vec{t}, do(\alpha, \sigma))$. 

2. If $W$ is a non-atomic formula, there are two different cases.

(a) When $W$ is of the form $\neg W_1$, $[W_1 \lor W_2]$, $(\exists v : Q)W_1$ or $(\forall v : Q)W_1$ for some regressable formulas $W_1$ and $W_2$ in $L^{OS}$, according to the definition of the regression operator $R^{os}$ and the induction hypothesis, it is obvious that $R^{os}[W]$ is well-sorted and uniform in $S_0$, and $D \models_{T} W \equiv R^{os}[W]$.

(b) Otherwise, $W$ is of the form $W_1 \land W_2 \land \cdots \land W_n$ for some regressable formulas $W_1, \cdots, W_n$ in $L^{OS}$, where $n \geq 2$. There are three sub-cases.

(b.1) Suppose that each $W_i$ ($i = 1..n$) is of the form $t_i,1 = t_i,2$ for some (well-sorted) terms $t_i,1, t_i,2$. If there is a well-sorted mgu $\mu$ for $\{\langle t_i,1, t_i,2 \rangle \mid i = 1..n\}$, according to the definition of $R^{os}$, i.e., $R^{os}[W] = W$, it is obvious to see that $R^{os}[W]$ is well-sorted and uniform in $S_0$, and $D \models_{T} W \equiv R^{os}[W]$. Otherwise, $W$ is unsatisfiable based on the assumption that every sort is not empty, hence $D \models_{T} W \equiv false$, which is well-sorted and uniform in $S_0$, and $D \models_{T} W \equiv R^{os}[W]$ according to the definition of the regression operator $R^{os}$.

(b.2) Suppose that for some $j (j = 1..n-1)$ such that each $W_i$ ($i = 1..j$) is of the form $t_i,1 = t_i,2$ for some (well-sorted) terms $t_i,1, t_i,2$, and none of $W_k$ ($k = j + 1..n$) is of the form $t_k,1 = t_k,2$ for any terms $t_k,1, t_k,2$. If there is a well-sorted mgu $\mu$ for $\{\langle t_i,1, t_i,2 \rangle \mid i = 1..j\}$, which is unique given the sort hierarchy is a meet semi-lattice, according to the definition of $R^{os}$ and the induction hypothesis, we have

$$D \models_{T} W \equiv W_1 \land \cdots \land W_j \land W'_j + 1$$

$$\equiv W_1 \land \cdots \land W_j \land R^{os}[W'_j + 1]$$

$$\equiv R^{os}[W],$$

where $W'_j + 1$ is a new formula obtained by substituting any free variables in $W_j + 1 \land \cdots \land W_n$ using the well-sorted mgu $\mu$, and it is existentially-quantified at front for every newly introduced sorted variable in $\mu$ when sorts of unified terms have to be weakened, and $R^{os}[W]$ is well-sorted and uniform in $S_0$. Otherwise, $W$ is unsatisfiable based on the assumption that every sort (other than the empty sort)
is not empty, hence $D \models_\mathbb{T} W \equiv false$, which is well-sorted and uniform in $S_0$, and $D \models_\mathbb{T} W \equiv \mathcal{R}^{os}[W]$ according to the definition of the regression operator $\mathcal{R}^{os}$.

(b.3) Otherwise, since $\mathcal{R}^{os}[W] = \mathcal{R}^{os}[W_1] \land \cdots \land \mathcal{R}^{os}[W_n]$, it is easy to see that $\mathcal{R}^{os}[W]$ is well-sorted and uniform in $S_0$, and $D \models_\mathbb{T} W \equiv \mathcal{R}^{os}[W]$ according to the induction hypothesis.

□

To prove a similar regression theorem in the order-sorted situation calculus like the regression theorem in Reiter’s situation calculus, we first prove the following relative satisfiability theorem like the relative satisfiability theorem of Reiter’s situation calculus in [134].

**Theorem 16** Given an order-sorted BAT $D = (\mathbb{T}, D)$, $D$ is satisfiable with respect to $\mathbb{T}$ iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is satisfiable with respect to $\mathbb{T}$.

**Proof:** The approach of proving this theorem is exactly the same as the one in the proof of the relative satisfiability theorem of Reiter’s situation calculus in [134], except that in the direction of “$\Leftarrow$”, we will construct a model for $D$ in order-sorted logic, given that there is a model of $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ that is consistent with the sort theory $\mathbb{T}$ in order-sorted logic.

The “$\Rightarrow$” direction is obvious. We only need to show the “$\Leftarrow$” direction. Suppose $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ has a model $\mathcal{M}_0$ consistent with the given sort theory $\mathbb{T}$. Define a structure $\mathcal{M}$ as follows: $\mathcal{M}$’s domain for any sort in $\mathbb{T}$ that is not sort $Situation$ is interpreted the same as the domain of the sort interpreted by $\mathcal{M}_0$. $\mathcal{M}$’s domain for the sort $Situation$, denoted as $Situation^\mathcal{M}$, is the set of all finite sequences of elements of $Action^\mathcal{M}$, where $Action^\mathcal{M} = Action^{M_0}$, i.e., the domain of sort $Action$ in $\mathcal{M}_0$. Next, we define how $S_0$, $do$, predicates and functions other than $do$ are interpreted by $\mathcal{M}$. For any symbol $P$ in $\mathcal{D}$, its interpretation in $\mathcal{M}$ will be denoted as $P^\mathcal{M}$.

1. $S_0^\mathcal{M} = [\emptyset]$, the empty sequence.
2. Whenever $\alpha^M \in \text{Action}^M$ and $\sigma^M = [\alpha^M_1, \alpha^M_2, \ldots, \alpha^M_n] \in \text{Situation}^M$, then
   \[ \text{do}^M(\alpha, \sigma) = [\alpha^M_1, \alpha^M_2, \ldots, \alpha^M_n, \alpha^M]. \]

3. $\sigma \prec^M \sigma'$ iff $\sigma^M$ is a proper subsequence of $(\sigma')^M$.

4. Let $\mathcal{M}$ interpret all situation independent predicates and functions (other than $\text{do}$) exactly the same as $\mathcal{M}_0$. In particular, the interpretation of $\equiv$ are the same in $\mathcal{M}$ and $\mathcal{M}_0$ satisfying unique name axioms for objects in $\mathcal{D}_{S_0}$. Also note that action functions are also situation independent functions, and therefore $\mathcal{M}$ satisfies $\mathcal{D}_{\text{una}}$.

5. Next, we specify how $\mathcal{M}$ interprets fluents and predicate $\text{Poss}$ at $S_0$.

\[ \text{(a)} \] For any (well-sorted) relational fluent $F(\bar{x}_{1..n}, s)$ and any sort-assignment $I$ that assigns $\bar{x}_{1..n}$ and $s$ to elements in their corresponding sort domains\(^2\), $\mathcal{M}, I \models^T_{\mathcal{S}} F(\bar{x}_{1..n}, S_0)$ (i.e., $\mathcal{M}, I$ is a $T$-interpretation that satisfies $F(\bar{x}_{1..n}, S_0)$) iff $\mathcal{M}_0, I \models^S_{\mathcal{S}} F(\bar{x}_{1..n}, S_0)$. It follows that $\mathcal{M}$ is a model of $\mathcal{D}_{S_0}$ iff $\mathcal{M}_0$ is a model of $\mathcal{D}_{S_0}$.

\[ \text{(b)} \] We specify how $\mathcal{M}$ interprets $\text{Poss}$ on $S_0$. Let $\alpha$ be a (ground) element in the domain of sort $\text{Action}$. There are two possibilities:

\[ \text{(b.1.)} \] There is a sort-assignment $I$ assigning $\alpha$ to variable $a$, and an action function $A(\bar{x}_{1..n})$ of language $\mathcal{L}^{\text{OS}}$ such that $\mathcal{M}, I \models^T_{\mathcal{S}} a = A(\bar{x}_{1..n})$. Action function $A$ must have an action precondition axiom of the form like Axiom 5.1. Because $\Pi_A(\bar{x}_{1..n}, s)$ is uniform in $s$, $\Pi_A(\bar{x}_{1..n}, S_0)$ has been assigned a truth value under $\mathcal{M}$ and any sort-assignment. Then, let

\[ \mathcal{M}, I \models^S_{\mathcal{T}} \text{Poss}(a, S_0) \text{ iff } \mathcal{M}, I \models^T_{\mathcal{S}} \Pi_A(\bar{x}_{1..n}, S_0). \]

Under unique name axioms for actions, it specifies a unique truth value for $\text{Poss}(a, S_0)$:

\(^2\)The formal definition of a sort-assignment can be found in Section 2.3.2, Def. 6(4).
Suppose there is a sort-assignment $I'$ that is just like $I$ except that it assigns a different tuple of domain elements to $\vec{x}_{1..n}$ and $M, I' \models_T^{os} a = A(\vec{x}_{1..n})$. But this cannot happen because $M$ satisfies the unique name axioms for actions.

Suppose there is a sort-assignment $I'$ that is just like $I$ except that it assigns $\alpha$ to $a$ and an action function $B$ of $L^{OS}$ different from $A$ such that $M, I' \models_T^{os} a = B(\vec{x}_{1..n})$. But again, this cannot happen because $M$ satisfies the unique name axioms for actions.

(b.2.) For every sort-assignment $I$ assigning $\alpha$ to $a$ and every action function $A$ in $L^{OS}$, $M, I \models_T^{os} a \neq A(\vec{x}_{1..n})$. We are free to specify whether or not $M, I \models_T^{os} Poss(a, S_0)$. So, we just say it does.

6. Now we inductively specify the interpretation of $M$ for fluents and $Poss$ other than for $S_0$. Assume that $M$ interprets $Poss$ and all fluents at a situation $s$, we specify how $M$ interprets these at $do(a, s)$. Moreover, both the precondition axioms and successor state axioms will be satisfied under such construction.

Formally, suppose $\sigma$ is an element in the domain of sort $Situation$, and for every sort-assignment $I$ that assigns $\sigma$ to variable $s$, $M, I$ has interpreted $Poss$ and all fluents at the situation $s$.

(a) Suppose $F$ is a relational fluent with an SSA of the form Axiom 5.2. For every sort-assignment $I$ assigning $\sigma$ to variable $s$, we define

$$M, I \models_T^{os} F(\vec{x}_{1..n}, do(a, s)) \iff M, I \models_T^{os} \phi_F(\vec{x}_{1..n}, a, s),$$

where $\phi_F(\vec{x}_{1..n}, a, s)$ is an abbreviation of the RHS of the SSA of $F$.

(b) The construction of how $M, I$ interprets $Poss(a, do(a, s))$ for every sort-assignment $I$ that assigns $\sigma$ to variable $s$ is exactly parallel to that of case 5 above for $S_0$, using the fact that $M, I$ interprets all fluents at $do(a, s)$. 
This completes the construction of $M$. By the nature of this construction, it satisfies $\mathcal{D}$ with respect to sort theory $\mathcal{T}$. 

**Theorem 17** If $W$ is a (well-sorted) regresable formula with respect to an order-sorted BAT $\mathcal{D}$ in $\mathcal{L}^{OS}$, then

$$
\mathcal{D} \models \mathcal{T}^{os} \iff \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{T}^{os} \mathcal{R}^{os}[W].
$$

**Proof:** According to Th. 15, it is the same as to prove $\mathcal{D} \models \mathcal{T}^{os} \phi \iff \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{T}^{os} \phi$ for any (well-sorted) regresable sentence $\phi$ that is uniform in $S_0$. The "$\Rightarrow$" direction is obvious. We now show the "$\Leftarrow$" direction. Suppose that $\mathcal{D} \models \mathcal{T}^{os} \phi$. Then, $\mathcal{D} \cup \{\neg \phi\}$ is unsatisfiable with respect to $\mathcal{T}$ in order-sorted logic. Let $\mathcal{D}_{S_0} \cup \{\neg \phi\}$ be a new initial theory $\mathcal{D}'_{S_0}$. Then, $\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \{\neg \phi\}$ is unsatisfiable with respect to $\mathcal{T}$ in order-sorted logic according to Th. 16. That is, $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{T}^{os} \phi$. 

Hence, to reason whether $\mathcal{D} \models \mathcal{T}^{os} W$ is the same as to compute $\mathcal{R}^{os}[W]$ first and then to reason whether $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{T}^{os} \mathcal{R}^{os}[W]$. Besides, according to Th. 15, it is easy to see that the following consequence holds.

**Corollary 5** If $W_1$ and $W_2$ are regresable formulas in $\mathcal{L}^{OS}$ such that $\models \mathcal{T}^{os} W_1 \equiv W_2$, then $\mathcal{D} \models \mathcal{T}^{os} \mathcal{R}^{os}[W_1] \equiv \mathcal{R}^{os}[W_2]$.

Intuitively, Corollary 5 states that the regressed results of two logically equivalent regresable formulas (possibly having different syntactic forms only) are still equivalent.

### 5.4 Order-Sorted Situation Calculus v.s. Reiter’s Situation Calculus

Although BATs and regresable formulas in $\mathcal{L}^{OS}$ are based on order-sorted logic, they can be related to BATs and regresable formulas in Reiter’s situation calculus.
First, given a well-sorted formula \( W \) with respect to the sort theory of some order-sorted BAT \( D \) (or simply say, with respect to \( D \)) in \( L^{OS} \), we define what is a *translation* of \( W \) in Reiter’s situation calculus \( L_{sc} \). Some concepts are introduced here for later convenience. For any sort \( Q \) in the language of \( L^{OS} \), we introduce a unary predicate \( Q(x) \), which will be true iff \( x \) is of sort \( Q \) in \( L^{OS} \). Note that we can use same symbols for both sorts and their corresponding unary predicates based on the assumption that all sort symbols are distinct from usual predicate symbols in \( L^{OS} \).

**Definition 17** Consider any well-sorted formula \( \phi \) with respect to a background order-sorted BAT \( D \) in \( L^{OS} \). A *translation* of \( \phi \) to a (well-sorted) sentence in Reiter’s situation calculus, denoted as \( tr(\phi) \), is defined as:

\[
tr(P(\vec{t})) \overset{def}{=} P(\vec{t}) \quad \text{for every atom } P(\vec{t}) \text{ (including } s \prec s' \text{ and } s = s' \text{ for any situations } s \text{ and } s');
\]

\[
tr(\neg \phi) \overset{def}{=} \neg tr(\phi);
\]

\[
tr((\exists x: \bot)\phi) \overset{def}{=} false;
\]

\[
tr((\forall x: Q)\phi) \overset{def}{=} \neg tr((\exists x: Q. \neg \phi));
\]

\[
tr((\exists x: Q)\phi) \overset{def}{=} (\exists x: Q)tr(\phi), \text{ if } Q \in \{\text{Object, Action, Situation}\};
\]

\[
tr((\exists x: T)\phi) \overset{def}{=} (\exists x: Object)tr(\phi) \lor (\exists x: Action)tr(\phi) \lor (\exists x: Situation)tr(\phi);
\]

\[
tr((\exists x: Q)\phi) \overset{def}{=} (\exists y: Object)[Q(y) \land tr(\phi(x/y))], \text{ if } Q \notin \{\top, \bot, \text{Object, Action, Situation}\};
\]

\[
tr(\phi \circ \psi) \overset{def}{=} tr(\phi) \circ tr(\psi) \text{ for } \circ \in \{\lor, \land, \land, \lor, \land, \land\}.
\]

\[\square\]

The intuition behind the definition above is obvious, for any well-sorted formula \( W \) in \( L^{OS} \), we can always find an "equivalent" formula in Reiter’s format. The meaning of equivalence between \( W \) and \( tr(W) \) is formally given in Lemma 7 below.
We would like to show that the order-sorted situation calculus $L^{OS}$ is correct, or sound, in the sense that for any BAT $D$ in $L^{OS}$ we can always find a way to represent the BAT in Reiter’s situation calculus $L_{sc}$ (known as the corresponding BAT $D'$ of $D$ in $L_{sc}$) such that for any regressable formula $W$, it can be entailed by $D$ iff the translation of $W$ in Reiter’s situation calculus $L_{sc}$ can be entailed by the corresponding BAT $D'$ in $L_{sc}$. Later, the corresponding BAT $D'$ of $D$ is denoted as $TR(D)$ to remind that it is constructed out of $D$. That is,

**Theorem 18 (Soundness)** For any order-sorted BAT $D = (T, D)$ in $L^{OS}$, there exists a corresponding BAT $D'$ (denoted as $TR(D)$ below), such that, for any regressable sentence (i.e., a query) $W$, we have

$$D \models_{T}^{os} W \iff TR(D) \models_{ms} tr(W).$$

It is hard to prove Th. 18 directly. Inspired by the standard relativization of order-sorted logic to unsorted predicate logic [127, 147, 167, 152], our general idea of proving Th. 18 is as follows (see the diagram in Figure 5.1). In Step 1, we construct a BAT $TR(D)$ (called the corresponding Reiter’s BAT of $D$ above) in Reiter’s situation calculus. In Step 2, we prove that there is an unsorted theory $D''$ (strong relativization of $D$) and an unsorted first-order sentence $W''$ (relativization of $W$) such that $D \models_{T}^{os} W$ iff $D'' \models_{fo} W''$. In Step 3, we show that $TR(D) \models_{ms} tr(W)$ iff $D'' \models_{fo} W''$, for some unsorted theory $D''$ (standard relativization of $TR(D)$) and first-order sentence $W''$ (relativization of $tr(W)$). Finally, in Step 4, we show that $D'' \models_{fo} W''$ iff $D' \models_{fo} W''$.

\[\begin{align*}
D \models_{T}^{os} W &\quad \iff \quad D'' \models_{fo} W'' \\
TR(D) \models_{ms} tr(W) &\quad \iff \quad D'' \models_{fo} W''
\end{align*}\]

\[\begin{align*}
&\quad \Downarrow \quad \text{(Step 1)} \\
\quad \text{(Step 2)} \\
\quad \text{(Step 3)} \quad &\quad \Downarrow \quad \text{(Step 4)}
\end{align*}\]

Figure 5.1: Diagram of the Outline for Proving Th. 18

The following definition of relativization of order-sorted logic to unsorted predicate
logic (Def. 18) and the bridge axioms (Def. 19) were given in [127, 147, 167, 152].

**Definition 18** For any well-sorted formula $\phi$ (with respect to a sort theory $T$) in $\mathcal{L}^{OS}$, $rel(\phi)$, a relativization of $\phi$, is an unsorted formula defined recursively as follows.

\[
rel(P(\vec{t})) \overset{\text{def}}{=} P(\vec{t});
\]
\[
rel(\neg \phi) \overset{\text{def}}{=} \neg rel(\phi);
\]
\[
rel(\phi \circ \psi) \overset{\text{def}}{=} rel(\phi) \circ rel(\psi) \quad \text{for} \ \circ \in \{\land, \lor, \supset\};
\]
\[
rel((\forall x:Q)\phi) \overset{\text{def}}{=} (\forall y)(Q(y) \supset rel(\phi[x/y]));
\]
\[
rel((\exists x:Q)\phi) \overset{\text{def}}{=} (\exists y)(Q(y) \land rel(\phi[x/y])).
\]
Moreover, for any set Set of well-sorted formulas,
\[
rel(Set) \overset{\text{def}}{=} \{rel(\phi) \mid \phi \in \text{Set}\}.
\]

\[\]

Note that Reiter’s situation calculus $\mathcal{L}_{sc}$ is in fact based on many-sorted logic, which is a special case of order-sorted logic, with three disjoint sorts ($Action$, $Situation$ and $Object$). All formulas in $\mathcal{L}_{sc}$ are well-sorted with respect to the sort theory of $\mathcal{L}_{sc}$ with all quantified variables restricted to suitable sorts by default. Hence, the definition of $rel$ can also be applied to any formula or a set of formulas in Reiter’s situation calculus.

Now, it is straightforward to prove the following lemma (Lemma 7) for $rel$ and $tr$ by structural induction, which shows the equivalence relationship between a well-sorted formula $W$ in $\mathcal{L}^{OS}$ and its translation $tr(W)$ in $\mathcal{L}_{sc}$.

**Lemma 7** Consider any well-sorted formula $\phi$ (with respect to a background BAT $\mathcal{D}$) in $\mathcal{L}^{OS}$. Then, given the default assumption that everything in the universe is either an action, or a situation, or an object, we have $\models^{so} rel(tr(\phi)) \equiv rel(\phi)$.

**Definition 19** For any sort theory $T$, which includes predicate declarations, function declarations and/or subsort declarations, the set of bridge axioms of $T$, $BA(T)$, is a set of unsorted formulas as follows:
(a) \((\forall x). Q_2(x) \supset Q_1(x)\) for each \(Q_2 \leq Q_1 \in \mathbb{T}\);

(b) \(Q(c)\) for each \(c: Q \in \mathbb{T}\);

(c) \((\forall \vec{x}_{1..n}). \bigwedge_{i=1}^{n} Q_i(x_i) \supset Q(f(\vec{x}_{1..n}))\) for each function \(f: \vec{Q}_{1..n} \rightarrow Q \in \mathbb{T}\).

Note that in particular, when we compute the bridge axioms for a sort theory \(\mathbb{T}\) in a given order-sorted BAT \(\mathcal{D}\), \(\text{Situation}(S_0)\) is always included in \(BA(\mathbb{T})\) for \(S_0: \text{Situation}\) in \(\mathbb{T}\) and the axioms of the form (c) are introduced for all functions, including action functions and the special situation function \(do(a, s)\).

Based on the definition of relativization and the bridge axioms, the following lemma has been proved in [152, 167, 13].

**Lemma 8** For any well-sorted sentence \(\phi\) with respect to a sort theory \(\mathbb{T}\), we have that 
\[\models_{\mathbb{T}}^\os \phi \iff BA(\mathbb{T}) \models^o \text{rel}(\phi).\]

We then define the standard relativization of a BAT as follows.

**Definition 20** Consider a sort theory \(\mathbb{T}\) in an order-sorted (or many-sorted) logic and a set of well-sorted axioms \(\mathcal{D}\) with respect to the given sort theory. Then, the standard relativization of \(\mathcal{D}\), an unsorted theory, is defined as
\[REL(\mathcal{D}) \overset{\text{def}}{=} \text{rel}(\mathcal{D}) \cup BA(\mathbb{T}).\]

In particular, for any BAT \(\mathcal{D}_1\) in Reiter’s situation calculus \(\mathcal{L}_{sc}\) that has a finite set \(\mathbb{T}_{\mathcal{D}_1}\) of function declarations and predicate declarations for all predicates and functions appeared in \(\mathcal{D}_1\), the standard relativization of \(\mathcal{D}_1\) is
\[REL(\mathcal{D}_1) \overset{\text{def}}{=} \text{rel}(\mathcal{D}_1 - \phi_{\Sigma}) \cup BA(\mathbb{T}_{\mathcal{D}_1}) \cup \{\text{rel}(\phi_{\Sigma})\}.\]
where $\phi_\Sigma$ is one of the foundational axiom that represents the second-order induction axiom

$$\forall P. P(S_0) \land (\forall a, s)[P(s) \supset P(do(a, s))] \supset (\forall s)P(s), \quad (5.4)$$

and the relativization of $\phi_\Sigma$, $rel(\phi_\Sigma)$, is defined as

$$\forall P. P(S_0) \land (\forall a, s)[Action(a) \land Situation(s) \supset (P(s) \supset P(do(a, s)))]$$

$$\supset (\forall s)Situation(s) \supset P(s). \quad (5.5)$$

It is easy to see that the standard relativization of a BAT of Reiter’s situation calculus is a very slight extension of the standard relativization of a set of well-sorted (first-order) formulas by applying the (standard) relativization function to a (second-order) well-sorted formula. Therefore, similar to the Relativization Theorem proved in [152], we have the following lemma.

**Lemma 9** Consider any regressable formula $W$ with a background BAT $\mathcal{D}$ in Reiter’s situation calculus $\mathcal{L}_{sc}$. Then,

$$\mathcal{D} \models_{\text{ns}} W \text{ iff } REL(\mathcal{D}) \models_{\text{fo}} rel(W).$$

Now we proceed to Step 1 mentioned in Figure 5.1 of the outline. Consider any order-sorted BAT $\mathcal{D}$. We construct the corresponding Reiter’s BAT of $\mathcal{D}$, denoted as $TR(\mathcal{D})$, that will be the Reiter’s BAT we are looking for in Th. 18. In $TR(\mathcal{D})$, we introduce three new special predicates $SortedObj(x)$, $SortedAct(a)$ and $SortedSit(s)$. Intuitively, for any term $t$ (a, or $s$, respectively) of sort object (action, or situation, respectively), $SortedObj(t)$ ($SortedAct(a)$, or $SortedSit(s)$, respectively) means that $t$ (a, or $s$, respectively) needs to be well-sorted with respect to the given sort theory $T$ in the order-sorted BAT $\mathcal{D}$. Note that the reason why we introduce three different special predicates for well-sorted terms ($SortedObj(x)$, $SortedAct(a)$ and $SortedSit(s)$) is because Reiter’s
Situation Calculus is a many-sort logic with three sorts only and his BATs have a particular syntactic format. For instance, every formula in an initial theory needs to be uniform in the initial situation $S_0$, and every SSA has to be of the form $F(\vec{x}, do(a,s)) \equiv \phi_F(\vec{x}, a, s)$.

In order to construct a BAT in Reiter’s Situation Calculus that satisfies Th. 18, we need to “encode” information about the well-sortedness of terms into the constructed BAT. However, if we introduce only one predicate to describe well-sortedness, say $sorted(x)$ for any $x$ (including objects, actions and situations) representing $x$ is well-sorted, then it would be problematic when we want to axiomatize the property of $sorted$ – it neither can be considered as an axiom in the initial theory (since it is not uniform in $S_0$), nor can be considered as an SSA (since its last argument is not of sort situation).

Notice that in [141], sorted quantifiers are omitted as a convention, because their sorts are always obvious from context. Hence, when we construct the BAT $TR(\mathcal{D})$ in Reiter’s situation calculus below, all free variables are implicitly universally sorted-quantified according to their obvious sorts. The declarations for functions and predicates (including for predicates $SortedObj$, $SortedAct$ and $SortedSit$) are always standard, hence are not mentioned here.

- $TR(\mathcal{D})$ includes the standard foundational axioms and the set of unique name axioms for action functions in Reiter’s situation calculus.

- The initial theory of $TR(\mathcal{D})$, say $\mathcal{D}'_{S_0}$, includes the following axioms. Below, in $TR(\mathcal{D})$, each free variable is universally quantified with a default sort $Object$ (or $Q_i$ itself, respectively) if $Q_i \leq T\ Object$ (or $Q_i \not\leq T\ Object$, respectively).

1. For any well-sorted sentence $\phi \in \mathcal{D}_{S_0}$, $tr(\phi)$ is in $\mathcal{D}'_{S_0}$.

2. For each declaration $Q_2 \leq Q_1$ in $T$, add an axiom

   $$tr(\forall x: T)(\exists y_2: Q_2.x = y_2) \supset (\exists y_1: Q_1.x = y_1)).$$

3. For any constant declaration $c: Q$ where $Q \leq T\ Object$ and $Q \neq Object$, add an axiom $Q(c)$. Note that other constant declarations will still be kept in the
sort theory of $TR(D)$ in language $L_{sc}$ by default. For example, $S_0: Situation$, $C: Object$ for any constant object $C$ appeared in $TR(D)$ and $A: Action$ for any constant action function $A$.

4. For each (situation-independent) function $f$ (including action function) whose declaration is $f: \tilde{Q}_{1..n} \rightarrow Q$ in $T$ ($n \geq 1$), add an axiom

$$tr((\forall \bar{x}_{1..n} : \tilde{Q}_{1..n}).(\exists y : Q).y = f(\bar{x}_{1..n})).$$

5. We also include the following axioms in the initial theory of $TR(D)$:

(a) $SortedObj(y) \equiv tr(\bigwedge_{i=1}^{k}(\exists x_{i,1} : Q_{i,1}, \ldots, x_{i,n_{i}} : Q_{i,n_{i}}).y = f_{i}(x_{i,1}, \ldots, x_{i,n_{i}}) \land \bigwedge_{j=1}^{n_{i}} SortedObj(x_{i,j}))$, where $f_{1}, \ldots, f_{k}$ (including constant Objects) are all functions other than action functions and do function included in $D$, and the function declaration for each $f_{i}$ in $T$ is $f_{i} : Q_{i,1} \times \cdots \times Q_{i,n_{i}} \rightarrow Q_{i,1+n_{i}}$ (each $Q_{i,j} \leq T_{Object}$);

(b) $SortedAct(a) \equiv tr(\bigwedge_{i=1}^{m}(\exists x_{i,1} : Q_{i,1}, \ldots, x_{i,n_{i}} : Q_{i,n_{i}}).a = A_{i}(x_{i,1}, \ldots, x_{i,n_{i}}) \land \bigwedge_{j=1}^{n_{i}} SortedObj(x_{i,j}))$, where $A_{1}, \ldots, A_{m}$ (including constant action functions) are all action functions included in $D$, and the function declaration for each $A_{i}$ in $T$ is $A_{i} : Q_{i,1} \times \cdots \times Q_{i,n_{i}} \rightarrow Action$ (each $Q_{i,j} \leq T_{Object}$);

(c) $tr(P(\bar{x}_{1..n}) \supset \bigwedge_{i=1}^{n}(\exists y_{i} : Q_{i,y_{i}} = x_{i} \land SortedObj(x_{i})))$ for each situation-independent predicate $P : \tilde{Q}_{1..n} \in T$;

(d) $SortedSit(S_{0})$;

(e) $tr(F(\bar{x}_{1..n}, S_{0}) \supset \bigwedge_{i=1}^{n}(\exists y_{i} : Q_{i,y_{i}} = x_{i} \land SortedObj(x_{i})))$ for each fluent $F : \tilde{Q}_{1..n} \times Situation \in T$. Here, all $y_{i}$'s are distinct auxiliary variables never appearing in $\bar{x}_{1..n}$.

- For action $A(\bar{x}_{1..n})$ whose precondition axiom in $D_{ap}$ has the form Eq. (5.1), we replace it with a precondition axiom in the format of Reiter's situation calculus:

$$Poss(A(\bar{x}_{1..n}), s) \equiv \Pi^{'}_{A}(\bar{x}_{1..n}, s)$$  (5.6)
where $\Pi'_A(\vec{x}_{1..n}, s)$ is uniform in $s$, resulting from
\[ tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}).(\bigwedge_{i=1}^n x_i = y_i \land SortedObj(x_i)) \land \Pi_A(\vec{y}_{1..n}, s)). \]
Here, all $y_i$’s are distinct auxiliary variables never appearing in $\Pi_A(\vec{x}_{1..n}, s)$.

- The set of successor state axioms of $TR(D)$ now includes the following axioms:

  1. For each relational fluent $F(\vec{x}_{1..n}, s)$, whose SSA in $D_{ss}$ is of the form Eq. (5.2), we replace it with SSA in the format of Reiter’s situation calculus Eq. (3.2) as follows:

\[
F(\vec{x}_{1..n}, do(a, s)) \equiv \phi'_F(\vec{x}_{1..n}, a, s)
\]

where $\phi'_F(\vec{x}_{1..n}, a, s)$ is a $L_{sc}$ formula uniform in $s$, resulting from
\[
tr(SortAct(a) \land SortedSit(s) \land (\exists \vec{y}_{1..n} : \vec{Q}_{1..n}).(\bigwedge_{i=1}^n x_i = y_i \land SortedObj(x_i)) \land \phi_F(\vec{y}_{1..n}, a, s)).
\]
Here, all $y_i$’s are distinct auxiliary variables never appearing in $\phi_F(\vec{x}_{1..n}, s)$;

  2. $SortedSit(do(a, s)) \equiv SortedAct(a) \land SortedSit(s)$.

Now, we define a different relativization, the strong relativization, for BATs in order-sorted situation calculus $L^{OS}$ (Def. 21). The reasons to define a strong relativization other than using the standard relativization are as follows.

(1) We include the sort theory in each order-sorted BAT in the language of $L^{OS}$, while Reiter’s situation calculus mentions sort declarations generally in the signature of $L_{sc}$.

(2) We are not able to use standard relativization to relate order-sorted BATs to BATs in Reiter’s situation calculus directly because of the particular syntactic formats of BATs in Reiter’s situation calculus.

(3) We expect that any predicates (including fluents) will be true only if they are for “reasonable” types of objects in unsorted logic, i.e., for well-sorted terms with
respect to a given sort theory in order-sorted logic.

For these reasons, we developed the strong relativization for order-sorted BATs to prove Th. 18.

**Definition 21** For any order-sorted BAT $\mathcal{D} = (\mathbb{T}, D)$ in $\mathcal{L}^{OS}$, besides introducing unary predicates that correspond to sorts in $\mathbb{T}$, same as the special new predicates introduced in the corresponding Reiter’s BAT of $\mathcal{D}$, $TR(\mathcal{D})$, we also use $SortedObj(x)$, $(SortedAct(a)$ and $SortedSit(s)$, respectively) to represent that $t$ (a, or $s$, respectively) is well-sorted with respect to the given sort theory $\mathbb{T}$ in the order-sorted BAT $\mathcal{D}$.

The **strong relativization of** $\mathcal{D}$ **is an unsorted theory defined as**

$$REL_{S}(\mathcal{D}) \overset{\text{def}}{=} rel_{S}(\mathcal{D}) \cup BA(\mathbb{T}),$$

where $rel_{S}(\mathcal{D})$ is a set of axioms including the following axioms:

(a) $(\forall \bar{x}_{1..n}). \bigwedge_{i=1}^{n} Q'_{i}(x_{i}) \supset Q'_{n+1}(f(\bar{x}_{1..n}))$, where each $Q'_j$ ($j = 1..n+1$) is a predicate in $\{Action, Situation, Object\}$ and its corresponding sort $Q'_j$ satisfies $Q_j \leq T Q'_j$, for any function $f : Q_{1..n} \rightarrow Q_{n+1} \in \mathbb{T}$ (including constant functions, action functions and do($a, s$));

(b) all axioms in $rel(\mathcal{D}_{S_0} \cup \Sigma - \{\phi_{\Sigma}\})$, where $\phi_{\Sigma}$ is Axiom (5.4);

(c) the relativization of Axiom (5.4), i.e., Axiom (5.5);

(d) $(\forall \bar{x}_{1..n}, \bar{y}_{1..m}). \bigwedge_{i=1}^{n} (Object(x_{i}) \land Object(y_{i})) \supset (A(\bar{x}_{1..n}) = A(\bar{y}_{1..n}) \supset \bigwedge_{i=1}^{n} x_{i} = y_{i})$ for each action function symbol $A$;

(e) $(\forall \bar{x}_{1..n}, \bar{y}_{1..m}). \bigwedge_{i=1}^{n} Object(x_{i}) \land \bigwedge_{j=1}^{m} Object(y_{j}) \supset A(\bar{x}_{1..n}) \neq B(\bar{y}_{1..m})$ for any two distinct action function symbols $A$ and $B$.

---

3In particular, when $n = 0$, $f(\bar{x}_{1..n})$ is a constant function $c$, and we have $Q'(f)$, where $Q'$ is a predicate in $\{Action, Situation, Object\}$ and its corresponding sort $Q'$ satisfies $Q \leq T Q'$, for $c : Q$ in $\mathbb{T}$ (including constant action functions and the initial situation $S_0$).
Chapter 5. An Order-Sorted Situation Calculus

144

(f) \( \forall y.\text{Object}(y) \supset [\text{SortedObj}(y) \equiv \bigvee_{i=1}^{k}(\exists x_{i,1}, \ldots, x_{i,n_{i}}). y = f_{i}(x_{i,1}, \ldots, x_{i,n_{i}}) \wedge (\bigwedge_{j=1}^{n_{i}} Q_{i,j}(x_{i,j}) \wedge \text{SortedObj}(x_{i,j}))] \), where \( f_{1}, \ldots, f_{k} \) (including constant Objects) are all functions other than action functions and \( \text{do} \) function included in \( \mathcal{D} \), and the function declaration for each \( f_{i} \) in \( \mathbb{T} \) is \( f_{i} : Q_{i,1} \times \cdots \times Q_{i,n_{i}} \rightarrow Q_{i,n_{i}+1} \) (each \( Q_{i,j} \leq_{\mathbb{T}} \text{Object} \)). Note that for any \( i \), if \( n_{i} = 0 \) (i.e., \( f_{i} \) is a constant object), there are no quantifiers for variables \( x_{i,1}, \ldots, x_{i,n_{i}} \) at the front and \( \bigwedge_{j=1}^{n_{i}} Q_{i,j}(x_{i,j}) \wedge \text{SortedObj}(x_{i,j}) \equiv \text{true} \).

(g) \( \forall a.\text{Action}(a) \supset [\text{SortedAct}(a) \equiv \bigvee_{i=1}^{m}(\exists x_{i,1}, \ldots, x_{i,n_{i}}). a = A_{i}(x_{i,1}, \ldots, x_{i,n_{i}}) \wedge (\bigwedge_{j=1}^{n_{i}} Q_{i,j}(x_{i,j}) \wedge \text{SortedObj}(x_{i,j}))] \), where \( A_{1}, \ldots, A_{m} \) (including constant action functions) are all action functions included in \( \mathcal{D} \), and the function declaration for each \( A_{i} \) in \( \mathbb{T} \) is \( A_{i} : Q_{i,1} \times \cdots \times Q_{i,n_{i}} \rightarrow \text{Action} \) (each \( Q_{i,j} \leq_{\mathbb{T}} \text{Object} \)). Note that for any \( i \), if \( n_{i} = 0 \) (i.e., \( A_{i} \) is a constant action function), there are no quantifiers for variables \( x_{i,1}, \ldots, x_{i,n_{i}} \) at the front and \( \bigwedge_{j=1}^{n_{i}} Q_{i,j}(x_{i,j}) \wedge \text{SortedObj}(x_{i,j}) \equiv \text{true} \).

(h) \( \forall \vec{x}_{1..n}. \bigwedge_{i=1}^{n} \text{Object}(x_{i}) \supset [P(\vec{x}_{1..n}) \supset \bigwedge_{i=1}^{n} Q_{i}(x_{i}) \wedge \text{SortedObj}(x_{i})] \) for each situation-independent predicate \( P : \vec{Q}_{1..n} \in \mathbb{T} \).

(i) \text{SortedSit}(S_{0});

(j) \( \forall a, s.\text{Action}(a) \wedge \text{Situation}(s) \supset [\text{SortedSit}(\text{do}(a, s)) \equiv \text{SortedAct}(a) \wedge \text{SortedSit}(s)] \);

(k) \( \forall \vec{x}_{1..n}, a, s. \bigwedge_{i=1}^{n} \text{Object}(x_{i}) \wedge \text{Action}(a) \wedge \text{Situation}(s) \supset [F(\vec{x}_{1..n}, \text{do}(a, s)) \equiv \bigwedge_{i=1}^{n} Q_{i}(x_{i}) \wedge \text{SortedObj}(x_{i}) \wedge \text{SortedAct}(a) \wedge \text{SortedSit}(s) \wedge \text{rel}(\phi_{F}(\vec{x}_{1..n}, a, s))] \) for each fluent \( F \), whose SSA in \( \mathcal{D} \) is of the form Axiom (5.2).

(l) \( \forall \vec{x}_{1..n}. \bigwedge_{i=1}^{n} \text{Object}(x_{i}) \supset [F(\vec{x}_{1..n}, S_{0}) \supset \bigwedge_{i=1}^{n} Q_{i}(x_{i}) \wedge \text{SortedObj}(x_{i})] \) for each fluent \( F : \vec{Q}_{1..n} \times \text{Situation} \in \mathbb{T} \).

(m) \( \forall \vec{x}_{1..n}, s. \bigwedge_{i=1}^{n} \text{Object}(x_{i}) \wedge \text{Situation}(s) \supset [\text{Poss}(A(\vec{x}_{1..n}), s) \equiv \bigwedge_{i=1}^{n} (Q_{i}(x_{i}) \wedge \text{SortedObj}(x_{i})) \wedge \text{SortedSit}(s) \wedge \text{rel}(\Pi_{A}(\vec{x}_{1..n}, s))] \)
for each \( n \)-ary action function \( A \), whose precondition axiom in \( \mathcal{D} \) is of the form Axiom (5.1).

\[ \square \]

According to order-sorted logic, any (functional) term \( t \) that is of some sort \( Q \) (such as \( \text{Object} \), \( \text{Action} \) and \( \text{Situation} \), etc.) is not necessarily always to be well-sorted. Hence, the strong relativization of an order-sorted BAT \( \mathcal{D} \), \( \text{REL}_S(\mathcal{D}) \), does not include \( \text{Object}(x) \sqsupset \text{SortedObj}(x) \), \( \text{Action}(a) \sqsupset \text{SortedAct}(a) \) or \( \text{Situation}(s) \sqsupset \text{SortedSit}(s) \). On the other hand, the unsorted predicates \( \text{SortedObj}(x) \) (\( \text{SortedAct}(a) \), and \( \text{SortedSit}(s) \), respectively) in the strong relativization, will correspond to sorted predicates \( \text{SortedObj} : \text{Object} \) (\( \text{SortedAct} : \text{Action} \) and \( \text{SortedSit} : \text{Situation} \), respectively) in the Reiter’s BAT \( \mathcal{D}' \) that we will construct for Th. 18. According to the (standard) relativization of \( \mathcal{D}' \), which will include special predicates \( \text{SortedObj} : \text{Object} \), \( \text{SortedAct} : \text{Action} \) and \( \text{SortedSit} : \text{Situation} \), it is not necessary to add \( \text{SortedObj}(x) \sqsupset \text{Object}(x) \), \( \text{SortedAct}(a) \sqsupset \text{Action}(a) \) or \( \text{SortedSit}(s) \sqsupset \text{Situation}(s) \) to the strong relativization of the order-sorted BAT \( \mathcal{D} \), because such properties have nothing to do with relativization, and will be satisfied as a “side-effect” when the declarations for predicates are unique.

Given the definition of strong relativization (Def. 21), we can also prove a relativization theorem as follows for the strong relativization similar to the Sort Theorem proved in [167] and/or the Relativization Theorem proved in [152]. The following Lemma 10 constitutes Step 2 in Figure 5.1 with the outline of proving the soundness property (Th. 18).

**Lemma 10** Consider any regressable formula \( W \) with a background BAT \( \mathcal{D} = (\mathcal{T}, \mathcal{D}) \) in order-sorted situation calculus \( \mathcal{L}^{OS} \). Then,

\[
\mathcal{D} \models^W \mathcal{T} \iff \text{REL}_S(\mathcal{D}) \models^\text{fo} \text{rel}(W).
\]

**Proof:** Note that to prove this lemma is the same as to prove \( \mathcal{D} \cup \{ \neg W \} \) has no \( \mathcal{T} \)-model in order-sorted logic iff \( \text{REL}_S(\mathcal{D}) \cup \{ \neg \text{rel}(W) \} \) has no model in unsorted logic, which is the same as to prove \( \mathcal{D} \cup \{ \neg W \} \) has a \( \mathcal{T} \)-model in order-sorted logic iff \( \text{REL}_S(\mathcal{D}) \cup \{ \neg \text{rel}(W) \} \)
has a model in unsorted logic.

“⇐”: Assume that $REL_S(D) \cup \{ \neg \text{rel}(W) \}$ has a model $M_r$. We now construct a $T$-model $M$ of $D \cup \{ \neg W \}$.

For each sort symbol $Q$ (including $T$) in $T$, which has a corresponding unary predicate $Q$ in $REL_S(D)$, we define set $Q^M$ to be same as set $Q^{M_r}$ in unsorted logic.

For any constant $c$ symbol, if it has a declaration $c : Q$ in $T$, let $c^M = c^{M_r}$; otherwise, it is undefined. For any function symbol $f$ (including action functions and $\text{do}(a,s)$), whose declaration is $f : \vec{Q}_1 .. n \rightarrow Q_{n+1}$, $f^M$ is a mapping from domains $Q_1^M \times \cdots \times Q_n^M$ to domain $Q_{n+1}^M$, and for any ground term vector $\vec{t}_{1..n}$ such that $t_i^M$ is defined, $t_i^M = t_i^{M_r}$ and is in $Q_i^M$, let $(f(\vec{t}_{1..n}))^M = f^{M_r}(t_1^{M_r}, \ldots, t_n^{M_r})$. Such definition is feasible because all functions satisfy axioms (c) in $BA(T)$. Moreover, $(f(\vec{t}_{1..n}))^M \in Q_{n+1}^M$. For any predicate $P$ in $D$ (including situation-independent predicates, equality and fluents), whose predicate declaration is $P : \vec{Q}_1 .. n$, let $P^M = P^{M_r} \cap Q_1^{M_r} \times \cdots \times Q_n^{M_r}$, which fits the declaration for $P$.

Now using structural induction we can prove a statement ($P1$): “$M \models_T^{os} \phi$ iff $M_r \models^{fo} \text{rel}(\phi)$ for any well-sorted sentence $\phi$”.

Base case: If $\phi = P(t_1, \ldots, t_n)$ is a well-sorted ground atom for some predicate $P$ (including fluents and the special predicate $Poss$), then it is easy to see that $\text{rel}(\phi) = \phi$ and $M \models_T^{os} \phi$ iff $M_r \models^{fo} \text{rel}(\phi)$ according to the definition of $M$ on sort symbols, ground terms and predicates.

Inductive step: There are several different cases depending on the structure of $\phi$ (e.g., $\neg \phi_1, \phi_1 \lor \phi_2, \forall x.Q.\phi_1(x)$, etc). The proof for each case is very similar, we only show one case, say $\phi = \forall x.Q.\phi_1(x)$, as an example.

First, assume that $M_r \models^{fo} \text{rel}(\phi)$, i.e., $M_r \models^{fo} \forall x.Q(x) \supset \text{rel}(\phi_1(x))$. For any $d^M \in Q^M$, let $I(x \rightarrow d)$ be a sort-assignment mapping variable $x$ to $d$, which is still an ordinary variable assignment. Then, $M_r, I \models^{fo} Q(x) \supset \text{rel}(\phi_1(x))$. Moreover, since $d^M \in Q^M$, i.e., $d^{M_r} \in Q^{M_r}$, then $M_r, I \models^{fo} \text{rel}(\phi_1(x))$, and therefore $M, I \models_T^{os} \phi_1(x)$ by induction hypothesis. Overall, we have $M \models_T^{os} \forall x.Q.\phi_1(x)$.
Second, assume that \( \mathcal{M} \models_{T}^{os} \forall x : Q.\phi_1(x) \). For any (usual) assignment \( I(x \rightarrow d) \) that maps variable \( x \) to an element \( d \) in the whole domain, there are two cases. (1) If \( d_{Mr} \in Q_{Mr}, \) i.e., \( d_{M} \in Q_{M} \), then \( I \) is also a sort-assignment such that \( \mathcal{M}, I \models_{T}^{os} \phi_1(x), \) and therefore \( \mathcal{M}, I \models_{o} rel(\phi_1(x)) \) by induction hypothesis. (2) Otherwise, if \( d_{Mr} \notin Q_{Mr}, \) it is obvious that \( \mathcal{M}, I \models_{o} Q(x) \supset rel(\phi_1(x)) \). Overall, \( \mathcal{M}, I \models_{o} rel(\phi(x)) \) for any \( I \).

Based on (P1), for well-sorted sentence \( \neg W \), \( \mathcal{M} \models_{T}^{os} \neg W \) because \( rel(\neg W) = \neg rel(W) \) and \( \mathcal{M}_{r} \models_{o} \neg rel(W) \).

Now we check that \( \mathcal{M} \) is a model of \( D \).

First, according to the definition of \( \mathcal{M} \), it fits the predicate declarations for all predicates in \( D \). Since \( \mathcal{M}_{r} \) is a model of \( BA(T) \), which therefore enforces \( Q_{1}^{M} \subseteq Q_{2}^{M} \) for every \( Q_{1} \leq Q_{2} \). Similarly, since \( \mathcal{M}_{r} \) satisfies axioms of the form \( (b) \) ((c), respectively) in Def. 19, \( \mathcal{M} \) fits the sort declarations for constants (functions, respectively). It is easy to see that \( \mathcal{M} \) is a model of axioms in \( D_{S_{0}} \cup \Sigma \) based on the definition of \( \mathcal{M} \) and Statement (P1). Since \( \mathcal{M}_{r} \) satisfies Axioms (d-e) in Def. 21, it is obvious that \( \mathcal{M} \) satisfies unique name axioms in \( D_{una} \) represented using well-sorted formulas.

To show that \( \mathcal{M} \) is also a model of all axioms in \( D_{ss} \cup D_{ap} \), we first prove a statement (P2) “\( \mathcal{M}_{r} \models_{o} SortedSit(t_{S}) \) for every well-sorted ground situation term \( t_{S}, \mathcal{M}_{r} \models_{o} SortedAct(t_{A}) \) for every well-sorted ground action term \( t_{A} \) and \( \mathcal{M}_{r} \models_{o} SortedObj(t) \) for every well-sorted ground term \( t \), where \( t \) is either a constant of sort \( Q \leq T \) Object or is a ground functional term (object term) other than actions and situations” by complete induction on the number of nested layers of terms.

Since the proofs for object terms, action terms and situation terms are very similar, we just show the proof for object terms as an example.

Base case: For any well-sorted constant \( c \) that is not of sort \( Action \) or \( Situation \), according to the bridge axioms for constant functions in Def. 19(b) and the axioms

---

\(^4\)The only free variable in \( \phi_1 \) is \( x \) of sort \( Q \) and \( I \) maps it to an element in the domain of \( Q \), which makes \( I \) a sort-assignment by Def. 6.
in Def. 21(a,f), we have that \( c^{M_r} \in \text{SortedObj}^{M_r} \).

**Inductive step:** Assume that \( t \) is a ground term \( f(t_1, \ldots, t_n) \) where \( f \) has a functional declaration \( f : \tilde{Q}_{1..n} \rightarrow Q_{n+1} \) and each \( Q_i \) \( (i = 1..n + 1) \) is a subsort of \( \text{Object} \). We also assume that \( M_r \models^{fo} \text{SortedObj}(t_j) \) \( (j = 1..n) \) by induction hypothesis. Since each \( t_j \) is a well-sorted term, according to the definition of well-sorted terms and how \( M \) is defined based on \( M_r \), it is easy to see that \( M_r \models^{fo} Q_j(t_j) \) \( (j = 1..n) \). Hence, \( M_r \models^{fo} \text{SortedObj}(t) \) according to the bridge axioms of function declarations in Def. 19(c) and the axioms in Def. 21(a,f). Overall, \((P2)\) is proved.

Based on \((P1)\) and \((P2)\), we show that \( M \) is a model of the SSA (Axiom 5.2) of any \((n + 1)\)-ary fluent \( F(\tilde{x}_{1..n}, s) \) \( (n \geq 0) \). For any sort-assignment \( I \) that maps \( x_i \in Q_i^{M_r} \) \( (i = 1..n) \) \( (a \text{ and } s, \text{ respectively}) \) to an element \( d_i \in \text{Object}^{M_r} \) \( (\alpha \in \text{Action}^{M_r} \text{ and } \sigma \in \text{Situation}^{M_r} \text{ respectively}) \), we prove for both implications in a well-sorted SSA.

1. Assume that \( M, I \models^{as} F(\tilde{x}_{1..n}, da(a, s)) \). Since \( I \) is still a (usual) variable assignment, \( M_r, I \models^{fo} \bigwedge_{i=1}^{n} \text{Object}(x_i) \land \text{Action}(a) \land \text{Situation}(s) \) (by using the bridge axioms for subsort declarations), \( M_r, I \models^{fo} F(\tilde{x}_{1..n}, do(a, s)) \) (by the definition of \( M \)) and \( M_r, I \models^{fo} \bigwedge_{i=1}^{n} (Q_i(x_i) \land \text{SortedObj}(x_i)) \land \text{SortedAct}(a) \land \text{SortedSit}(s) \) (by \((P2)\) and \( I \) is a sort-assignment). Therefore, \( M_r, I \models^{fo} \text{rel}(\phi_F(\tilde{x}_{1..n}, a, s)) \) (since \( M_r \) is a model of axioms in Def. 21(k-l)). Then, \( M, I \models^{fo} \phi_F(\tilde{x}_{1..n}, a, s) \) (by \((P1)\)).

2. Assume that \( M, I \models^{as} \phi_F(\tilde{x}_{1..n}, a, s) \). Then \( M_r, I \models^{fo} \text{rel}(\phi_F(\tilde{x}_{1..n}, a, s)) \) (by \((P1)\)). Moreover, we have that \( M_r, I \models^{fo} \bigwedge_{i=1}^{n} (Q_i(x_i) \land \text{SortedObj}(x_i)) \land \text{SortedAct}(a) \land \text{SortedSit}(s) \) (by \((P2)\) and the definition of \( M \)) and \( M_r, I \models^{fo} \bigwedge_{i=1}^{n} \text{Object}(x_i) \land \text{Action}(a) \land \text{Situation}(s) \) (by using the bridge axioms). Then, we have \( M_r, I \models^{fo} F(\tilde{x}_{1..n}, do(a, s)) \) (since \( M_r \) is a model of axioms in Def. 21(k-l)). So, \( M, I \models^{fo} F(\tilde{x}_{1..n}, do(a, s)) \) (by the definition of \( M \)).

Overall, we have proved that \( M \) is a model of the (well-sorted) SSA of \( F(\tilde{x}_{1..n}, s) \).

Similarly, we can show that \( M \) is a model of the precondition axiom (Axiom 5.1) of any \( n \)-ary action function \( A(\tilde{x}_{1..n}) \) \( (n \geq 0) \). Consider any sort-assignment \( I \) for
that maps variable \( x_i \) (\( i = 1..n \)) \( (s, \) respectively) to an element \( d_i \in Object^M \) \((\sigma \in Situation^M \) respectively), we prove both implications in a well-sorted precondition axiom.

(1) If \( \mathcal{M}, I \models^o \text{Poss}(A(\vec{x}_{1..n}), s) \), then \( \mathcal{M}_r, I \models^o \text{Poss}(A(\vec{x}_{1..n}), s) \) (by the definition of \( \mathcal{M} \)). Moreover, \( \mathcal{M}_r, I \models^o \bigwedge_{i=1}^n (Q_i(x_i) \land SortedObj(x_i)) \land SortedSit(s) \) (by (P2) and \( I \) is a sort-assignment). Also, we have that \((d_i)^M \in Object^M \) and \( \sigma^M \in Situation^M \) (by using the bridge axioms for subsort declarations). So, \( \mathcal{M}_r, I \models^o \bigwedge_{i=1}^n Object(x_i) \land Situation(s) \). Therefore, \( \mathcal{M}_r, I \models^o \text{rel}(\Pi_A(\vec{x}_{1..n}, s)) \) (since \( \mathcal{M}_r \) is a model of axioms in Def. 21(m)). Then, \( \mathcal{M}, I \models^o \Pi_A(\vec{x}_{1..n}, s) \) (by (P1)).

(2) If \( \mathcal{M}, I \models^o \Pi_A(\vec{x}_{1..n}, s) \), then \( \mathcal{M}_r, I \models^o \text{rel}(\Pi_A(\vec{x}_{1..n}, s)) \) (since \( I \) is a sort-assignment). Moreover, we have that \( \mathcal{M}_r, I \models^o \bigwedge_{i=1}^n (Q_i(x_i) \land SortedObj(x_i)) \land SortedSit(s) \) (by (P2) and the definition of \( \mathcal{M} \)) and \( \mathcal{M}_r, I \models^o \bigwedge_{i=1}^n Object(x_i) \land Situation(s) \) (by using the bridge axioms of subsort declarations). Then, we have \( \mathcal{M}_r, I \models^o \text{Poss}(A(\vec{x}_{1..n}), s) \) (since \( \mathcal{M}_r \) is a model of axioms in Def. 21(m)). So, \( \mathcal{M}, I \models^o \text{Poss}(A(\vec{x}_{1..n}), s) \) (by the definition of \( \mathcal{M} \)).

Overall, we proved that \( \mathcal{M} \) is a model of the (well-sorted) precondition axiom of action \( A(\vec{x}_{1..n}) \).

Based on the above proof, we have that if \( REL_S(D) \cup \{\neg \text{rel}(W)\} \) has a model \( \mathcal{M}_r \) then \( D \cup \{\neg W\} \) has a model \( \mathcal{M} \).

“\( \Rightarrow \)”: Assume that \( D \cup \{\neg W\} \) has a model \( \mathcal{M} \). We construct a model, say \( \mathcal{M}_r \), of \( REL_S(D) \cup \{\neg \text{rel}(W)\} \) in unsorted logic. Let the domain of \( \mathcal{M}_r \), denoted as \( \Delta \), be the same as \( \top^M \). We then define the interpretation of \( \mathcal{M}_r \) for constants, predicates and functions. For each unary predicate symbol \( Q \), which corresponds to a sort \( Q \) in \( D \), we define set \( Q^{\mathcal{M}_r} \) to be the same as the set \( Q^M \). For each predicate \( P \) other than unary predicates that correspond to sorts, we define set \( P^{\mathcal{M}_r} \) to be the same as the set
Note that for any ground term vector \( \vec{t} \), if one of the terms, say \( t_j \), is ill-sorted with respect to the given sort theory, \( P(\vec{t}) \) is not defined in \( \mathcal{M} \), while \( f^{\mathcal{M}_r} \not\in P^{\mathcal{M}_r} \).

For each constant \( c \) (including constant action functions and initial situation \( S_0 \)), we define \( c^{\mathcal{M}_r} \) to be the same as \( c^{\mathcal{M}} \). For each \( n \)-ary function \( f \) (including \( do \), action functions and situation-independent functions), whose unique term declaration in \( \mathcal{D} \) is \( f : Q_{1..n} \rightarrow Q_{n+1} \), and any term vector \( \vec{t}_{1..n} \) in \( \Delta^n \), let \( f^{\mathcal{M}} \) be a 1-1 mapping from \( \Delta^n \) to \( \Delta \), and we define \( (f(\vec{t}_{1..n}))^{\mathcal{M}_r} = f^{\mathcal{M}_r}(t_1^{\mathcal{M}_r}, \ldots, t_n^{\mathcal{M}_r}) \) recursively based on the assumption that the interpretation of each term \( t_i^{\mathcal{M}_r} \) \( (i = 1..n) \) has been defined.

- If each \( t_i^{\mathcal{M}_r} = t_i^{\mathcal{M}} \) and \( (f(\vec{t}))^{\mathcal{M}} \) is defined in \( \mathcal{M} \), then let \( (f(\vec{t}))^{\mathcal{M}_r} = (f(\vec{t}))^{\mathcal{M}} \).

- Else, if \( (f(\vec{t}))^{\mathcal{M}} \) is not defined in \( \mathcal{M} \), but for each \( i = 1..n \), \( t_i^{\mathcal{M}_r} \) is in \( (Q_i')^{\mathcal{M}_r} \) for some unary predicate \( Q_i' \), whose corresponding sort \( Q_i \) in \( \mathcal{D} \) satisfies that \( Q_i' \in \{Action, Situation, Object\} \) and \( Q_i \leq_T Q_i' \), \( (f(\vec{t}))^{\mathcal{M}_r} \) can be defined arbitrarily satisfying that: \( (f(\vec{t}))^{\mathcal{M}_r} \) is assigned to an element in \( (Q_{n+1}')^{\mathcal{M}_r} \) that had never been used to interpret any terms yet, where \( Q_{n+1}' \) is a unary predicate whose corresponding sort \( Q_{n+1} \) in \( \mathcal{D} \) satisfies that \( Q_{n+1}' \in \{Action, Situation, Object\} \) and \( Q_i \leq_T Q_{n+1}' \).

- Otherwise, let \( (f(\vec{t}))^{\mathcal{M}_r} \) be an arbitrary element in the whole domain \( \Delta \) that has not been assigned to any term yet. This is doable since \( \Delta \) is infinite.

Note that the interpretations to all functions are doable because we have finitely many functions and each sort \( Action, Situation \) or \( Object \) in \( L^{OS} \) is assumed to have infinitely many elements. For predicate \( SortedObj \), let \( SortedObj^{\mathcal{M}_r} = \{ t^{\mathcal{M}_r} \mid \text{there is a well-sorted ground term } t \text{ that is defined in } \mathcal{M} \text{ and } t^{\mathcal{M}} \in Object^{\mathcal{M}} \} \). Similarly, let \( SortedAct^{\mathcal{M}_r} = \{ t^{\mathcal{M}_r} \mid \text{there is a well-sorted ground term } t \text{ that is defined in } \mathcal{M} \text{ and } t^{\mathcal{M}} \in Action^{\mathcal{M}} \} \) and \( SortedSit^{\mathcal{M}_r} = \{ t^{\mathcal{M}_r} \mid \text{there is a well-sorted ground term } t \text{ that is defined in } \mathcal{M} \text{ and } t^{\mathcal{M}} \in Sit^{\mathcal{M}} \} \).
defined in $\mathcal{M}$ and $t^\mathcal{M} \in \text{Situation}^\mathcal{M}$}. In fact, according to our definition, it is easy to see that for any well-sorted ground term $t$, $t^\mathcal{M}$ is defined and $t^{\mathcal{M}_r} = t^\mathcal{M}$.

Now, using structural induction we can prove a statement (P3): “$\mathcal{M}_r \models^{o} \text{rel}(\phi)$ iff $\mathcal{M} \models^{o} \phi$ for any well-sorted formula $\phi$”.

Base case: If $\phi = P(t_1, \ldots, t_n)$ is a well-sorted ground atom, then $\text{rel}(\phi) = \phi$. According to the definition of $P^{\mathcal{M}_r}$, then $\mathcal{M}_r, I \models^{o} \text{rel}(\phi_1(x))$ (by the assumption), and $\mathcal{M}_r, I \models^{o} Q(x) \supset \text{rel}(\phi_1(x))$. Otherwise, $d^{\mathcal{M}_r} \notin Q^{\mathcal{M}_r}$, then $\mathcal{M}_r, I \models^{o} Q(x) \supset \text{rel}(\phi_1(x))$. Overall, $\mathcal{M}_r \models^{o} \forall x. Q(x) \supset \text{rel}(\phi_1(x))$, i.e., $\mathcal{M}_r \models^{o} \text{rel}(\phi)$.

Inductive step: There are several different cases depending on the structure of $\phi$ (e.g., $\neg \phi_1$, $\phi_1 \lor \phi_2$, $\forall x : Q. \phi_1(x)$, etc). The proofs for each case are very similar, we only show one case, say $\phi = \forall x : Q. \phi_1(x)$, as an example.

First, assume that $\mathcal{M} \models^{o} \phi$, i.e., for any sort-assignment $I(x \rightarrow d)$ such that $d^{\mathcal{M}} \in Q^{\mathcal{M}}$ we have $\mathcal{M}, I \models^{o} \phi_1(x)$. Then, for any (usual) assignment $I$ that maps variable $x$ to an element $d$ in the whole domain, if $d^{\mathcal{M}_r} \in Q^{\mathcal{M}_r}$, then $d^{\mathcal{M}_r} \in Q^{\mathcal{M}_r}$, $\mathcal{M}, I \models^{o} \phi_1(x)$ (by the assumption), and $\mathcal{M}_r, I \models^{o} Q(x) \supset \text{rel}(\phi_1(x))$. Otherwise, $d^{\mathcal{M}_r} \notin Q^{\mathcal{M}_r}$, then $\mathcal{M}_r, I \models^{o} Q(x) \supset \text{rel}(\phi_1(x))$. Overall, $\mathcal{M}_r \models^{o} \forall x. Q(x) \supset \text{rel}(\phi_1(x))$, i.e., $\mathcal{M}_r \models^{o} \text{rel}(\phi)$.

Second, assume that $\mathcal{M}_r \models^{o} \text{rel}(\phi)$. For any sort-assignment $I$ that maps $x$ to an element $d$, where $d^{\mathcal{M}} \in Q^{\mathcal{M}}$, then $I$ can be considered as a usual assignment with $d^{\mathcal{M}_r} \in Q^{\mathcal{M}_r}$, and $\mathcal{M}_r, I \models^{o} \text{rel}(\phi_1(x))$ (by assumption). Hence, $\mathcal{M}, I \models^{o} \phi_1(x)$ (by induction hypothesis). Overall, $\mathcal{M} \models^{o} \phi$.

Notice that $\neg \text{rel}(W)$ equals to $\text{rel}(\neg W)$. Moreover all axioms in the BAT of $\mathcal{D}$ are well-sorted. Based on the above proof, we have that if $\mathcal{M}$ is a model of $\mathcal{D} \cup \{\neg W\}$ then $\mathcal{M}_r$ is a model of $\text{REL}_S(\mathcal{D}) \cup \{\neg \text{rel}(W)\}$ by (P3).

Now, we show that $\mathcal{M}_r \models^{o} \text{REL}_S(\mathcal{D})$ by checking for each axiom in Def. 21.

(a) Based on the interpretation of $\mathcal{M}_r$ for functions, it is easy to see that $\mathcal{M}_r$ is a model of any axiom in Def. 21(a).

(b) By (P3), $\mathcal{M}_r$ is a model of all axioms in $\text{rel}(\mathcal{D}_{S_0} \cup \Sigma \setminus \{\phi_\Sigma\}$, where $\phi_\Sigma$ is Axiom (5.4)
(Def. 21(b)).

(c) Based on how \( \mathcal{M}_r \) is defined and (P3), it is a model of Axiom (5.5).

(d) Since we always choose distinct assignments for action terms when we define \( \mathcal{M}_r \), it is easy to see that \( \mathcal{M}_r \) is a model of all axioms in Def. 21(d,e).

(e) Because we have one and only one term declaration for each function (constant, in particular), it is easy to see that any ground term in \( \mathcal{L}^{OS} \) is well-sorted, if and only if it is a constant that has a term declaration, or it is a functional term \( f(t_1, \ldots, t_n) \), where each subterm \( t_i \) is well-sorted and \( t^M_i \) is in \( Q^M_i \) for \( f : \tilde{Q}_{1..n} \to Q_{n+1} \) such that \( Q_i \leq_T Object \). Hence, according to the definitions of \( SortedObj^{M_r} \), \( SortedAct^{M_r} \) and \( SortedSit^{M_r} \), it is easy to see that \( \mathcal{M}_r \) is a model of all axioms in Def. 21(f,g,i,j). For example, since \( S_0 \) is a well-sorted constant and \( S^M_0 \in \text{Situation}^{M_r} \), hence \( \mathcal{M}_r \) is a model of \( SortedSit(S_0) \) (Def. 21(i)).

(f) According to the definition of the interpretation of \( \mathcal{M}_r \) for each \( n \)-ary predicate \( P \) (including fluents), for any ground term vector \( \vec{t} = (t_1, \ldots, t_n) \), \( (\vec{t})^M_r \in P^M_r \) iff \( \vec{t}^M_r \) is defined and \( (\vec{t})^M_r \in P^M_r \). Hence, each \( t_i \) has to be a well-sorted term and \( t^M_i \in t^M_i \) according to the predicate declaration of \( P \) in the sort theory. According to the definition of \( SortedObj \), \( SortedAct \) and \( SortedSit \), it is easy to see that \( \mathcal{M}_r \) is a model of all axioms in Def. 21(h,l).

(g) For any (usual) assignment \( I \) that maps \( \vec{x}_{1..n}, a, s \) to element \( d_1, \ldots, d_n, \alpha \) and \( \sigma \), such that each \( d_i \in Object^{M_r}, \alpha \in Action^{M_r} \) and \( \sigma \in Situation^{M_r} \), we prove the equivalence in Def. 21(k) as follows.

First, assume that \( \mathcal{M}_r, I \models^{fo} F(\vec{x}_{1..n},a,s) \), then similar to the reasoning of case (f) above, by the definition of \( \mathcal{M}_r \) and \( I \), we have that \( \mathcal{M}_r, I \models^{fo} F(\vec{x}_{1..n},do(a,s)) \supset \bigwedge_{i=1}^{n}(Q_i(x_i) \land SortedObj(x_i)) \land SortedAct(a) \land SortedSit(s) \), and then each \( d_i \in Q^M_i, \alpha \in Action^{M} \) and \( \sigma \in Situation^{M} \). Moreover, \( \mathcal{M}, I \models^{es} F(\vec{x}_{1..n},do(a,s)) \) (by
Therefore, we can prove Step 2 in Figure 5.1 using Lemma 10.

We now provide the details of proving Th. 18 by following the ideas presented in Figure 5.1.
Proof of Theorem 18.

Step 1: Let $\mathcal{D}' = TR(\mathcal{D})$. It is easy to see that $tr(W)$ is a regressable formula and $\mathcal{D}' = TR(\mathcal{D})$ is a BAT in the language of Reiter’s situation calculus, which is based on many-sorted logic with three sorts – Object, Action and Situation.

Step 2: Let $\mathcal{D}'' = REL_s(\mathcal{D})$ and $W'' = rel(W)$. By Lemma 10, we already have

$$\mathcal{D} \models_{\mathcal{T}} W \text{ iff } \mathcal{D}'' \models_{\mathcal{F}} W''$$

Step 3: Let $W''' = rel(tr(W))$ and $\mathcal{D}''' = REL(\mathcal{D}')$ (see Def. 20). Using Lemma 9, we have

$$\mathcal{D}' \models_{\mathcal{M}} tr(W) \text{ iff } \mathcal{D}''' \models_{\mathcal{F}} W'''$$

Step 4: We now show that $\mathcal{D}'' \models_{\mathcal{F}} W'' \text{ iff } \mathcal{D}''' \models_{\mathcal{F}} W'''$.

According to Lemma 7 and how $\mathcal{D}''$, $W''$, $\mathcal{D}'''$ and $W'''$ are defined, it is straightforward to see that for any sentence $\phi$ in $\mathcal{D}'' \cup \{\neg W''\}$, one can always find a sentence $\phi'$ in $\mathcal{D}''' \cup \{\neg W'''\}$ and vise versa. Hence, $\mathcal{D}'' \cup \{\neg W''\}$ is unsatisfiable iff $\mathcal{D}''' \cup \{\neg W'''\}$ is unsatisfiable.

We provide an example to illustrate some axioms in the corresponding Reiter’s BAT of an order-sorted BAT.

**Example 17** Consider the BAT $\mathcal{D}$ from Example 14. Most of the axioms in $TR(\mathcal{D})$ are obvious and we just provide examples of the axiom of SortedObj, a precondition axiom and an SSA:

$\text{SortedObj}(x) \equiv x = B_1 \lor x = B_2 \lor x = Boston \lor x = Toronto$

$$\lor x = T_1 \lor x = T_2 \lor \exists y.\text{City}(y) \land x = \text{twinCity}(y);$$

$\text{Poss}(\text{load}(x,t),s) \equiv \text{Box}(x) \land \text{Truck}(t) \land \text{SortedObj}(x) \land \text{SortedObj}(t)$

$$\land \text{SortedSit}(s) \land \neg \text{On}(x,t,s) \land (\exists y.\text{City}(y) \land \text{InCity}(x,y,s) \land \text{InCity}(t,y,s));$$

$\text{InCity}(d,c,do(a,s)) \equiv \text{MovObj}(d) \land \text{City}(c) \land \text{SortedObj}(d) \land \text{SortedObj}(c)$

$$\land \text{SortedAct}(a) \land \text{SortedSit}(s)[(\exists t,c_1.\text{Truck}(t) \land \text{City}(c_1) \land a = \text{drive}(t,c_1,c)$$

$$\land (d = t \lor \exists b.\text{Box}(b) \land b = d \land \text{On}(b,t,s))$$
It is important to notice that any query in $\mathcal{L}^{OS}$ has to be well-sorted with respect to a given background order-sorted BAT $\mathcal{D}$; while, in general, a query that can be answered in the corresponding Reiter’s BAT of $\mathcal{D}$ are not necessarily well-sorted with respect to $\mathcal{D}$. Below, Th. 19 shows that for any query that can be answered in $TR(\mathcal{D})$, it can be answered in $\mathcal{D}$ in a “well-sorted” way too.

**Theorem 19 (Completeness)** Let $\mathcal{D}$ be an order-sorted BAT in $\mathcal{L}^{OS}$, and $TR(\mathcal{D})$ be its corresponding Reiter’s BAT. Consider any regressable formula $W$ in Reiter’s situation calculus, in which there is no appearance of special predicates $SortedObj$, $SortedAct$ or $SortedSit$, $W$ can be translated to a (well-sorted) formula with respect to $\mathcal{D}$, denoted as $os(W)$ below, such that $TR(\mathcal{D}) \models^{ms} tr(os(W)) \equiv W$. Furthermore, we have $TR(\mathcal{D}) \models^{ms} W$ iff $\mathcal{D} \models^{os} os(W)$ when $W$ is a regressable sentence with respect to $TR(\mathcal{D})$.

To prove Th. 19, we first define some new concepts and prove a lemma.

**Definition 22** Let $\mathcal{D}$ be a BAT in the order-sorted situation calculus $\mathcal{L}^{OS}$, and $TR(\mathcal{D})$ be its corresponding Reiter’s BAT. Any term $t$ in Reiter’s situation calculus is a possibly sortable term with respect to $\mathcal{D}$, if one of the following conditions holds:

1. $t$ is a variable of sort $Action$, $Object$ or $Situation$ in $\mathcal{L}_{sc}$;
2. $t$ is a constant $c$, and $c:Q$ in $\mathcal{T}$ (we say that the sort of $c$ is $Q$ with respect to $\mathcal{D}$); or,
3. $t$ is of form $f(\vec{t}_{1..n})$, function declaration $f:Q_{1..n} \rightarrow Q$ in $\mathcal{T}$, for every $i$ ($i = 1..n$), $t_i$ either is a variable or is a non-variable possibly sortable term of sort $Q'_i$ with respect to $\mathcal{D}$ and $Q'_i \preceq \mathcal{T} Q_i$ in $\mathcal{T}$ (we say that the sort of $f(\vec{t}_{1..n})$ is $Q$ with respect to $\mathcal{D}$).

Similarly, any atom $P(\vec{t}_{1..n})$ in Reiter’s situation calculus (well-sorted with respect to $TR(\mathcal{D})$) and $P$ is not $SortedObj$, $SortedAct$ or $SortedSit$, is a possibly sortable atom with respect to $\mathcal{D}$, if for every $i$, $t_i$ either is a variable or is a non-variable term of sort $Q'_i$ with respect to $\mathcal{D}$ satisfying that:

\[
\lor InCity(d, c, s) \land \neg (\exists t, c_1.Truck(t) \land City(c_1) \land a = drive(t, c, c_1) \\
\land (d = t \lor \exists b.Box(b) \land b = d \land On(b, t, s))].
\]
(a) it is a possibly sortable term with respect to $\mathcal{D}$; and
(b) $P: \vec{Q}_{1..n}$ is in $\mathbb{T}$ and $Q'_i \leq_T Q_i$ with respect to $\mathcal{D}$.

□ Note that the predicate of a possibly sortable atom can be equality, $\text{Poss}$ or any predicate appeared in $\mathcal{D}$.

Given any $\mathcal{D}$ in order-sorted situation calculus, it is easy to see that every atom (term, respectively) in $TR(\mathcal{D})$ that can be considered as well-sorted with respect to $\mathcal{D}$ is always a possibly sortable atom (term, respectively); while a possibly sortable atom (term, respectively) is not necessarily well-sorted with respect to $\mathcal{D}$. We provide some simple examples of the terms and atoms defined in Def. 22.

Example 18 We continue with Example 17. The query

$$\exists x.\exists y.\text{InCity}(x, \text{twinCity}(y), \text{do}(\text{load}(B_1, T_1), S_0))$$

in $TR(\mathcal{D})$ is not well-sorted with respect to $\mathcal{D}$, but is possibly sortable with respect to $\mathcal{D}$. The query

$$\exists x.\text{InCity}(x, \text{twinCity}(B_1), \text{do}(\text{load}(B_1, T_1), S_0))$$

in $TR(\mathcal{D})$ is not possibly sortable with respect to $\mathcal{D}$, because $\text{twinCity}(B_1)$ is not a possibly sortable term with respect to $TR(\mathcal{D})$. □

Moreover, for later convenience, we proved the following lemma.

Lemma 11 Let $\mathcal{D}$ be a BAT in the order-sorted situation calculus $\mathcal{L}^{OS}$, and $TR(\mathcal{D})$ be its corresponding Reiter’s BAT. Then, for any atom $P(\vec{t}_{1..n})$ that is well-sorted in $\mathcal{L}_{sc}$ but is not possibly sortable with respect to $\mathcal{D}$, we have that $TR(\mathcal{D}) \models_{\text{ms}} P(\vec{t}_{1..n}) \equiv \text{false}$.

Proof: Assume that the declaration of an $n$-ary $P$ predicate (could be symbol $=$) in $\mathcal{D}$ is $P: Q_1 \times \cdots \times Q_n$. According to how $TR(\mathcal{D})$ is constructed, it is easy to prove that $TR(\mathcal{D}) \models_{\text{ms}} P(\vec{x}_{1..n}) \supset y_n = x_n \wedge P_S(x_n) \wedge$, where $P_S(x_n)$ represents $\text{SortedObj}(x_n)$ if $Q_n \leq_T \text{Object}$, or presents $\text{SortedSit}(x_n)$ if
\( Q_n = \text{Situation} \). Since the proofs for situation-independent predicates and for fluents are very similar, we will just provide the detail for situation-independent predicates only.

For any object term \( t \), we can prove by induction that \( TR(\mathcal{D}) \models^{ms} \text{SortedObj}(t) \supset tr(\bigwedge_{i=1}^{k}(\exists y_i : Q'_i \cdot y_i = t_i \land \text{SortedObj}(t_i))) \), where \( t_i (i = 1..n) \) are all the proper subterms of term \( t \) (i.e., not including \( t \) itself), and \( Q'_i \) is the sort of the \( j \)-th argument in the function declaration of function \( f \) assuming that \( t_i \) is the \( j \)-th argument of a functional term \( f(t') \) in term \( t \). For later convenience, we call \( f(t') \) the \textit{parental term} of \( t_i \) and call \( Q'_i \) as the \textit{argument sort} of \( t_i \) in its parental term.

We prove the lemma for a situation-independent predicate \( P(t_{1..n}) \) by contradiction. Assume that it is not true that \( TR(\mathcal{D}) \models^{ms} P(t_{1..n}) \equiv false \). I.e., there is a model \( \mathcal{M} \) of \( TR(\mathcal{D}) \) and an variable assignment of all free variables appeared in \( P(t_{1..n}) \) such that \( \mathcal{M}, I \models^{ms} P(t_{1..n}) \). As shown above we proved \( \mathcal{M}, I \models^{ms} tr((\exists y_{1..n} : \tilde{Q}_{1..n}) \cdot \bigwedge_{i=1}^{n}(y_i = t_i \land \text{SortedObj}(t_i))) \). However, since \( P(t_{1..n}) \) is not possibly sortable, according to Def. 22, there is a term \( t_j \) in \( \tilde{t}_{1..n} \) which is either not possibly sortable, or \( t_j \) is a functional term \( f \) whose sort is not a subsort of \( Q_j \) in \( \mathcal{D} \). For the previous case that \( t_j \) is a term that is not possibly sortable, then there is a proper subterm \( t'_j \) of \( t_j \) such that there is no model and assignment of \( TR(\mathcal{D}) \) satisfying \( \text{SortedObj}(t'_j) \land Q'_j(t'_j) \) where \( Q'_j \) is the argument sort of \( t'_j \) in its parental term, because it is easy to prove that \( TR(\mathcal{D}) \models^{ms} \text{SortedObj}(t'_j) \supset Q''_j(t'_j) \), where \( t'_j \) is a term of the form \( f_0(t'') \) (including constant) and \( f_0 : \tilde{Q}_0 \rightarrow Q''_j \) (or \( f_0 : Q''_j \) for constant) in the sort theory of \( \mathcal{D} \), and \( Q''_j \) is never a subsort of \( Q'_j \). This conflicts with \( \mathcal{M}, I \models^{ms} P(t_{1..n}) \) for some \( \mathcal{M} \) and assignment \( I \). For the later case that \( t_j \) is a functional term \( f \) whose sort is not a subsort of \( Q_j \) in \( \mathcal{D} \), then there is no model and assignment of \( TR(\mathcal{D}) \) satisfying \( \text{SortedObj}(t_j) \land Q_j(t_j) \), because it is easy to prove that \( TR(\mathcal{D}) \models^{ms} \text{SortedObj}(t_j) \supset Q''_j(t_j) \), where \( t_j \) is a term of the form \( f_0(t'') \) (including constant) and \( f_0 : \tilde{Q}_0 \rightarrow Q''_j \) (or \( f_0 : Q''_j \) for constant) in the sort theory of \( \mathcal{D} \), and \( Q''_j \) is never a subsort of \( Q_j \). This is conflict with \( \mathcal{M}, I \models^{ms} P(t_{1..n}) \) for some \( \mathcal{M} \) and assignment \( I \).
Hence, overall, we have that $\text{TR} (\mathcal{D}) \models_{ns} P (\vec{t}_{1..n}) \equiv false$. 

Now we define a function which transforms a formula in $\mathcal{L}_{sc}$ with respect to $\text{TR} (\mathcal{D})$ to a well-sorted formula in $\mathcal{L}_{OS}$ with respect to $\mathcal{D}$.

**Definition 23** Let $\mathcal{D}$ be a BAT in the order-sorted situation calculus $\mathcal{L}_{OS}$, $\text{TR} (\mathcal{D})$ be its corresponding Reiter’s BAT and $W$ be a regressable sentence in $\mathcal{L}_{sc}$ with respect to the background BAT $\text{TR} (\mathcal{D})$. Then, function $os(W)$ is defined as:

1. If $W$ is either of the form $(\forall x)W_1$, $(\exists x)W_1$, where the default sort of $x$ in $\text{TR} (\mathcal{D})$ is $Q$ (either Object, Action or Situation), then $os(\forall x)W_1 \overset{def}{=} (\forall x:Q)os(W_1)$, and $os(\exists x.W_1) \overset{def}{=} (\exists x:Q)os(W_1)$.

2. If $W$ is one of the form $\neg W_1$, $W_1 \land W_2$, $W_1 \lor W_2$, then
   
   $os(\neg W_1) \overset{def}{=} \neg os(W_1)$,
   $os(W_1 \land W_2) \overset{def}{=} os(W_1) \land os(W_2)$,
   $os(W_1 \lor W_2) \overset{def}{=} os(W_1) \lor os(W_2)$.

3. If $W$ is atomic and not possibly sortable, then $W \overset{def}{=} false$.

4. If $W$ is atomic and possibly sortable, assume that $\text{var}(W) = \langle x_1, \cdots, x_n \rangle$ is the vector of free variables that appear from left to right in $W$ (including repeated ones). For each $i = 1..n$, suppose that $x_i$ appears as an argument of a function $f_i$ in some term or as an argument of a predicate $P_i$ in $W$. Suppose $x_i$ appears in the $k_i$-th position of $f_i$ ($P_i$, respectively) in $W$, let $Q_i$ be the sort appeared in the $k_i$-th position of the declaration of $f_i$ ($P_i$, respectively) in the sort theory of $\mathcal{D}$. Then, let $I_W = \{ i \mid x_i \in \text{var}(W), Q_i \leq Object, Q_i \neq Object \}$, and $\vec{y} : \vec{Q} = \{ y_i : Q_i \mid i \in I_W \}$, where $y_i$’s are auxiliary variables never appeared in $W$ and each $y_i$ is distinct from others. And,
   
   $os(W) \overset{def}{=} (\exists \vec{y} : \vec{Q})(W_0 \land \bigwedge_{i \in I_W} x_i = y_i)$, where $W_0$ is obtained from substituting each $x_i$ with $y_i$ for $i \in I_W$. 

\qed
Now, we give the proof of Th. 19 as follows.

**Proof of Theorem 19.** For any query $W$ in Reiter’s situation calculus, let $W’ = os(W)$.

By using structural induction, it is easy to prove that $W’$ is a well-sorted query with respect to $D$ in order-sorted logic and $TR(D) \models^{ms} W \equiv tr(W’)$.

Since the inductive steps are obvious, we only mention base case of the proof here.

**Base case:**

1. $W$ is atomic but is not possibly sortable, $W’ = false$ which is well-sorted regressable sentence with respect to $D$, and $tr(W’) = false$. By Lemma 11, we have that $TR(D) \models^{ms} W \equiv tr(W’)$. 

2. $W = P(\vec{t}_1..n)$ is atomic and possibly sortable, where $P$ can be either situation-independent or a fluent. Assume that $P : Q_1' \times \cdots \times Q_n'$ is the predicate declaration in $D$ for $P$. Note that $Q_n’$ is *Situation* when $P$ is a fluent. It is easy to check that $os(W)$ is well-sorted with respect to $D$ according to the definition of sortable terms and the $os$-function. Moreover, according to the way of constructing $TR(D)$, it is easy to prove that we have

   $TR(D) \models^{ms} tr(os(W))$

   $\equiv (\exists \vec{y})(W_0 \land \bigwedge_{i \in I_W} (x_i = y_i \land Q_i(y_i)))$

   $\equiv W \land \bigwedge_{i \in I_W} Q_i(x_i)$

   $\equiv W \land tr(\psi(t_n)) \land \bigwedge_{i=1}^{n-1}(\exists y_i : Q_i' \cdot t_i = y_i \land SortedObj(t_i)))$

   $\equiv W,$

   where $\psi(t_n)$ is $SortedSit(t_n)$ if $P$ is a fluent, otherwise it is $(\exists y_n : Q_n' \cdot t_n = y_n \land SortedObj(t_n))$. The definition of notation $I_W$ and $W_0$ can be found in Def. 23, and $\vec{y}$ are the auxiliary variables $y_i$’s not appeared in $W$.

   Finally, by Th. 18 and $TR(D) \models^{ms} W \equiv tr(os(W))$, it is easy to see that $D \models^{os} os(W)$ iff $TR(D) \models^{ms} tr(os(W))$ iff $TR(D) \models^{ms} W$ when $W$ is a regressable sentence. □

The following example illustrates the definition of $os$ function and the idea of Th. 19.

**Example 19** Consider the $TR(D)$ in Example 17. Let $On(twinCity(Boston), T_1, s)$
Chapter 5. An Order-Sorted Situation Calculus

(\text{denoted as } W_3) \text{ be a regressable formula in } \mathcal{L}_{sc}, \text{ where } s \text{ is a variable of sort situation. It is easy to see that } W_3 \text{ is not possibly sortable. According to the way } TR(\mathcal{D}) \text{ is constructed, we have } TR(\mathcal{D}) \models^{ms} On(o,t,s) \supset Box(o). \text{ Then, for any situation } s, \text{ if } TR(\mathcal{D}) \models^{ms} On(twinCity(Boston),T_1,s), \text{ we need to have } TR(\mathcal{D}) \models^{ms} Box(twinCity(Boston)), \text{ which in fact does not hold according to the axioms in } TR(\mathcal{D}). \text{ Hence, } TR(\mathcal{D}) \models^{ms} W_3 \equiv os(W_3), \text{ where } os(W_3) = false.

Let \( W_4 \) be \( \forall s. \exists c. \neg InCity(B_1,twinCity(c),s) \), which is a regressable sentence in \( \mathcal{L}_{sc} \), where \( c: Object \text{ and } s: Situation \) hold by default. Then, \( os(W_4) \) is \( \forall s: Situation. \exists c: Object. \neg (\exists c_1: City.c_1 = c \land InCity(B_1,twinCity(c_1),s)) \). Since \( TR(\mathcal{D}) \models^{ms} InCity(B_1,twinCity(c),s) \supset City(c) \), it is easy to see that \( TR(\mathcal{D}) \models^{ms} W_4 \equiv tr(os(W_4)) \).

5.5 Computational Advantages of \( \mathcal{L}^{OS} \)

In this section, we discuss the advantages of using order-sorted logic and the order-sorted regression operator based on it.

Given any BAT \( \mathcal{D} \) in \( \mathcal{L}^{OS} \), it is easy to see that Reiter’s regression operator \( R \) [141] still can be applied to (well-sorted) regressable formulas (with respect to \( \mathcal{D} \)) in the language of \( \mathcal{L}^{OS} \). For example, \( R[\exists x: Q.W_1] = \exists x: Q.R[W_1] \) for any regressable formula \( \exists x: Q.W_1 \) in \( \mathcal{L}^{OS} \), and \( R[W_1 \land W_2] = R[W_1] \land R[W_2] \) for any regressable formula \( W_1 \land W_2 \) in \( \mathcal{L}^{OS} \), etc. Moreover, one can prove that \( R[W] \) is a formula in \( \mathcal{L}^{OS} \) uniform in \( S_0 \) and \( D \models^{os} W \equiv R[W] \). However, using the order-sorted regression operator \( R^{os} \) sometimes can give us computational advantages in comparison to using Reiter’s regression operator \( R \). But firstly, we show that the computational complexity of using \( R^{os} \) is no worse than that of \( R \).

For the regression operator \( R \) that can be used either in \( \mathcal{L}^{OS} \) or in \( \mathcal{L}_{sc} \) (\( R^{os} \) used in \( \mathcal{L}^{OS} \), respectively), we can construct a regression tree rooted at \( W \) for any regressable
query $W$ in either language. Each node in a regression tree of $R[W]$ ($R^{os}[W]$, respectively) corresponds to a sub-formula computed by regression, and each edge corresponds to one step of regression according to the definition of the regression operator. In the worst-case scenario, for any query $W$ in $L^{OS}$, the regression tree of $R^{os}[W]$ will have the same number of nodes as the regression tree of $R[W]$ (and linear with respect to the number of nodes in the regression tree of $R[tr(W)]$ with respect to $TR(D)$). Moreover, based on the assumption that our sort theory of $D$ is simple with empty equational theory, whose corresponding sort hierarchy is a meet semi-lattice, finding a unique (well-sorted) mgu takes the same time as in the unsorted case [152, 87, 171]. Hence, the overall computational complexity of building the regression tree of $R^{os}[W]$ is at most linear with respect to the size of Reiter’s regression tree.

**Theorem 20** Consider any regressable sentence $W$ with a background BAT $D$ in order-sorted situation calculus $L^{OS}$. Then, in the worst-case scenario, the complexity of computing $R^{os}[W]$ is the same as that of computing $R[W]$, which is also the same as the complexity of computing $R[tr(W)]$ in the corresponding Reiter’s BAT $TR(D)$.

**Proof:** In the worst-case scenario, for any query $W$ in $L^{OS}$, the regression tree of $R^{os}[W]$ constructed according to the definition of $R^{os}$ will have the same number of nodes as the regression tree of $R[W]$ constructed according to the definition of $R$ (and linear to the number of nodes in the regression tree of $R[tr(W)]$ constructed using $TR(D)$). Moreover, based on the assumption that our sort hierarchy of $D$ is a meet semi-lattice, finding (well-sorted) mgu takes linear time with respect to the size of the equalities, hence it will be at most linear to the size of $R^{os}[W]$ (i.e., the size of the leaves of the regression tree of $R^{os}[W]$).

On the other hand, under some circumstances, the regression of a query in $L^{OS}$ using $R^{os}$ instead of $R$ will give us computational advantages. Consider any query (i.e., a regressable sentence) $W$ with a background BAT $D$ in $L^{OS}$. Then, the computation
of \( \mathcal{R}^{os}[W] \) with respect to \( \mathcal{D} \) can sometimes terminate earlier than that of \( \mathcal{R}[W] \) with respect to \( \mathcal{D} \), and also earlier than the computation of \( \mathcal{R}[tr(W)] \) with respect to \( TR(\mathcal{D}) \). In particular, we have the following property.

**Property 1** Consider an order-sorted BAT \( \mathcal{D} \) in \( \mathcal{L}^{os} \), at least one of the SSAs in \( \mathcal{D} \) is not context-free, and any regressable formula \( W \) of the syntactic form \( t_{1,1} = t_{1,2} \land \ldots \land t_{m,1} = t_{m,2} \land W_1 \). Let the size of \( W \) (including the length of the terms in \( W \)) be \( n \). If there is no well-sorted mgu for equalities between terms \( \{ (t_{i,1}, t_{i,2}) \mid i = 1, \ldots, m \} \), then in the worst-case scenario, computing \( \mathcal{R}^{os}[W] \) runs in \( O(n) \), while computing \( \mathcal{R}[W] \) with respect to \( \mathcal{D} \) (\( \mathcal{R}[tr(W)] \) with respect to \( TR(\mathcal{D}) \)) runs in time \( O(2^n) \). Moreover, the size of the resulting formula of \( \mathcal{R}^{os}[W] \), which is false, is always constant, while the size of the resulting formula using \( \mathcal{R} \) is \( O(2^n) \).

**Proof:** The proof of the theorem is very obvious. According to the definition of Reiter’s regression operator, the equalities will be kept and regression will be further performed on \( W_1 \) (or on \( tr(W_1) \) in \( TR(\mathcal{D}) \), respectively), which in general takes exponential time with respect to the length of \( W_1 \) and causes exponential blow-up in the size of the formula. Once Reiter’s regression has terminated, a theorem prover will find that the resulting formula is false either because there is no mgu for terms when reasoning is performed in \( \mathcal{L}^{os} \) (or, due to the clash between sort related predicates when reasoning in \( \mathcal{L}_{sc} \), respectively). Hence, using the order-sorted regression operator can sometimes prune the regression tree built by \( \mathcal{R} \) exponentially (with respect to the size of the regressed formula), and therefore make regression terminated exponentially faster. \( \square \)

We provide an example below to show the computational advantage of using \( \mathcal{R}^{os} \). This example also illustrates the the class of conjunctive queries in Property 1 is common in regression and leads to significant savings if regression trees are pruned earlier on.

**Example 20** Consider the BAT \( \mathcal{D} \) from Example 14. Let \( W_5 \) be a \( \mathcal{L}^{os} \) query (i.e., a (well-sorted) regressable sentence)
\( \text{InCity}(T_1, \text{Toronto}, \text{do}(\text{drive}(T_1, \text{Boston}, \text{Toronto}), S_1)), \)

where \( S_1 \) is a well-sorted ground situation term that involves a long sequence of actions. According to the SSA of \( \text{InCity} \), at the branch of computing \( \mathcal{R}^\text{os}[\exists b : \text{Box}.b = T_1 \land \text{On}(b, t, S_1)] \) in the regression tree, since there is no well-sorted mgu for \((b, T_1)\), the application of order-sorted regression equals to \text{false} immediately. However, using Reiter’s regression operator (no matter in \( \mathcal{D} \) or in \( \text{TR}(\mathcal{D}) \)), his operator will keep doing regression on \( \text{On}(b, t, S_1) \) until getting (a potentially huge) sub-formula uniform in \( S_0 \). Once his regression has terminated, such sub-formula will also be proved equivalent to \text{false} with respect to the initial theory (\( \mathcal{D}_{S_0} \) or \( \text{TR}(\mathcal{D})_{S_0} \), respectively) using a theorem prover, for the same reason as above.

\[ \square \]

In addition, since our sort theory of a BAT \( \mathcal{D} \) in \( \mathcal{L}^{OS} \) is finite and it has one and only one declaration for each function and predicate symbol, for any query \( W \) (with respect to \( \text{TR}(\mathcal{D}) \)) in \( \mathcal{L}_{sc} \), it takes linear time (with respect to the length of the query) to check whether or not a term (an atom, respectively) in Reiter’s situation calculus is possibly sortable with respect to the sort theory in \( \mathcal{D} \) and to find a well-sorted formula \( \text{os}(W) \) in \( \mathcal{L}^{OS} \) that satisfies Th. 19. But, reasoning whether \( \mathcal{D} \models^\text{os} \text{os}(W) \) (starting from finding \( \text{os}(W) \)) sometimes can terminate exponentially earlier than finding whether \( \text{TR}(\mathcal{D}) \models^\text{ms} W \). Observe that reasoning about \( \text{TR}(\mathcal{D}) \models^\text{ms} W \) directly, for the formula \( W \) mentioned in Property 2, using regression \( \mathcal{R} \) could result in a exponentially large regression tree when computing \( \mathcal{R}[W] \). Also, the size of the resulting formula can be exponentially larger than that of \( W \). Moreover, it still needs further computational steps to find whether \( \text{TR}(\mathcal{D})_{S_0} \cup \text{TR}(\mathcal{D})_{\text{una}} \models^\text{ms} \mathcal{R}[W] \). In particular, we have

**Property 2** Assume that \( W = F(\vec{t}, \text{do}(\{\alpha_1, \cdots, \alpha_n\}, S_0)) \) is an atomic fluent in \( \mathcal{L}_{sc} \) that is not possibly sortable with respect to \( \mathcal{D} \). Then, it takes at most linear time (with respect to the length of the whole formula) to terminate reasoning \( \text{TR}(\mathcal{D}) \models^\text{ms} W \) by checking whether \( W \) is possibly sortable and computing the corresponding \( \text{os}(W) \) (which is \text{false}).
However, in the worst-case scenario, it takes exponential time (with respect to the length of the whole formula) to determine $\text{TR}(D)|_m^W$ by using the usual regression.

Note that the worst-case scenario mentioned in Property 2 often happens when a BAT is not context-free. That is, it is common that the usual regression operator leads to a regressed query whose length is exponential in the length of the original formula. Furthermore, even the corresponding $\text{os}(W)$ of any query $W$ is not false, according to the previous discussion, we sometimes still can gain further computational advantages during computing $R^{\text{os}}[\text{os}(W)]$ when reasoning by order-sorted regression in $L^{\text{OS}}$ instead of reasoning by regression in $L_{sc}$.

5.6 Related Work and Possible Future Directions

It is well-known that PDDL supports typed (sorted) variables and many implemented planners can take advantage of types [56]. A formal semantics for the typed ADL subset of PDDL was proposed in [28] and in [29] using ES, a dialect of SC, where types are represented using unary predicates. We also would like to contribute towards a formal logical foundation of PDDL, but in a different way using order-sorted logic. Moreover, our work focuses on the relations between Reiter’s BATs and our new order-sorted BATs and the computational advantages which regression in order-sorted BATs can provide (sometimes).

In the previous sections, we propose a logical theory for reasoning about action with respect to a taxonomy of objects based on order-sorted logic, define a regression-based reasoning mechanism taking advantage of sort theories, and discuss the computational advantages of our theory. One possible future work can be extending our logic to hybrid order-sorted logic [31, 13, 171]. Another possibility is to consider efficient reasoning in our framework by identifying specialized classes of queries or decidable fragments [1]. We also plan to work on an efficient implementation of our theory. Finally, in the
previous chapter, we proposed a framework of representing action hierarchies without classifications of objects, while in this chapter we considered sort hierarchies over objects only. It would be interesting to combine these two parts and consider reasoning about action taxonomies in the order-sorted situation calculus.
Chapter 6

Conclusions

Summary

In this thesis, we consider the problem of promoting the efficiency of reasoning about action in the situation calculus [141] from three different aspects.

In Chapter 3, we proposed a modified situation calculus based on the two-variable predicate logic with counting quantifiers. Within such framework, we studied two important problems of reasoning about actions – the projection problem and the executability problem. We showed that it is still possible to solve these two problems via regression by defining a new regression operator for regressable formulas in the modified situation calculus. We also proved that solving the projection and executability problems via regression is decidable and the upper bound of the computational complexity of the projection and the executability problems is in $2\text{-NExpTime}$. On the other hand, we proved that generally these two problems are co-$\text{NExpTime}$-complete in the modified situation calculus. Then we considered restricting further the regressable formulas and BATs in the modified situation calculus further based on the description logics $\mathcal{ALCO}(U)$ and $\mathcal{ALCOQO}(U)$, so that solving the projection and execution problems via regression may have better computational complexity than in the $C^2$-based situation calculus. In
addition, we showed that our modified situation calculus has a natural connection with DLs. Because some DLs, such as $ALCQIO(\sqcup, \sqcap, \neg, |, id)$, and $ALCQO(U)$, can be easily translated into $C^2$, Semantic Web knowledge bases described using these DLs can be transformed into theories represented in $L^C_{sc}$ or its fragments. Moreover, since $L^C_{sc}$ is also a fragment of the situation calculus, it can be used to describe and reason about dynamic aspects of the Semantic Web and ontologies (e.g., Semantic Web services and extensional ontology update) in a natural way.

In Chapter 4, we considered a problem of how to represent and reason about the effects of actions grouped in a realistically large taxonomy, where some actions can be more generic (or more specialized) than others. We proposed a hierarchical representation of actions based on the situation calculus to facilitate development, maintenance and elaboration of very large taxonomies of actions. In this representation, a finite set of axioms is used to describe specialization relationships between action functions, representing large taxonomies of actions hierarchically. The effects of actions in SSAs are represented for groups of actions rather than for individual actions. Such representation leads to more compact and modular basic action theories for reasoning about action than currently possible. We compared our new formalism of BATs with Reiter’s BATs, and proved that our representation inherits all the useful properties of his BATs. Moreover, we showed that our axioms can be more succinct, while still using an extended Reiter’s regression operator to solve the projection problem. Furthermore, such 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 a general guideline on how a taxonomy of actions can be constructed from the given set of effect axioms, if these axioms characterize completely the properties of actions in a domain.

In Chapter 5, we considered further improvement of the efficiency of the regression, the central reasoning mechanism in the situation calculus. We extended Reiter’s situation
calculus with the order-sorted logic, and named it the order-sorted situation calculus. In the new formalism, objects are classified into different types, called sorts, and sort theories that include subsort declarations and term declarations (particularly for action functions and fluents) are given in addition to the usual initial theories. For example, the arguments of an action function should be sub-classes of objects, and subsort declarations are used to describe hierarchies of object sorts, allowing taxonomic reasoning about objects. Then, we investigated what a well-sorted BAT is and what a well-sorted regression is under such framework. We considered extending the current regression operator with well-sorted unification techniques. With the modified regression, we gained computational efficiency by terminating the regression earlier when reasoning tasks are ill-sorted and by reducing the size of regression trees for well-sorted objects. We also studied the connection between the order-sorted situation calculus and BATs in the new language and Reiter’s situation calculus and BATs in it. We showed that for each query $W$ and a BAT $\mathcal{D}$ in the order-sorted situation calculus, we can always find an equivalent query $W'$ and a corresponding BAT $\mathcal{D}'$ in Reiter’s situation calculus, such that $\mathcal{D}$ entails $W$ iff $\mathcal{D}'$ entails $W'$, and for any query that can be entailed by $\mathcal{D}'$, there is an equivalent query in the order-sorted situation calculus that can be entailed by $\mathcal{D}$.

**Some Future Research Directions**

Finally, I conclude my work by briefly discussing some possible future research directions. In the previous chapters, we considered improving reasoning about action in the situation calculus from three different perspectives. In the future, we would like to consider the possibility of combining some of them together. For instance, combining the idea of order-sorted situation calculus and action hierarchies could allow us to take advantage of taxonomic knowledge both of objects and of actions. However, such combinations may need further careful study and a new formalization of the language.
The problem of automatic composition of Web services \cite{157} is to assemble atomic Web services, which are platform-independent software components. This problem has received interest from both the business world and the Semantic Web research community. It can be stated as an AI planning problem. Although we have shown that it is possible to describe Semantic Web services in our modified situation calculus, there is still much work to do to address Web service composition problems, such as automatic Web travel planning problems. We may also consider combining the research of the decidable fragment of the situation calculus with the idea of action hierarchies in dealing with the problem of Web service composition, where it could involve a large number of actions.

Plan recognition \cite{93} refers to the task of inferring the plan or plans of an intelligent agent from observations of the agent’s actions or the effects of those actions. Usually, plan recognition involves the problem of modeling a map of classes of actions or sequences of actions (i.e., plans). By using the map and the observations, the recognizer fills out details of the plan or the intention of the agent. Currently, the hierarchies of actions we considered in Chapter 4 are only for primitive actions. In the future, we would like to consider extending the idea to formalizing the hierarchies of sequences of actions and the possibility of applying the extended framework to the problem of plan recognition.

In this thesis, we mainly focused on the projection problem and executability problem via regression in reasoning about action, especially as demonstrated in Chapter 3. In the future, we also would like to consider whether or not our work can be extended to deal with more general reasoning problems such as answering queries with quantifications over actions and/or situations. We also would like to look into the possibility of combining the most recent research work in reasoning about action via progression \cite{163,108} into our approaches.
Bibliography


Bibliography


[70] Yilan Gu and Mikhail Soutchanski. A description logic based situation calculus. Annals of Mathematics and Artificial Intelligence (AMAI) Special Issue on Commonsense Reasoning for the Semantic Web, March 2010. Available at HTTP: http://www.springerlink.com/content/hr0m7q4h16210365/.


[97] Iluju Kiringa. Towards a general theory of advanced transaction models in the situation calculus. In Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Dan


Appendix A

Proofs of Theorems in Chapter 2

A.1 \( \mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id) \) and \( C^2 \) are Equally Expressive

In this section, we will provide detailed proof for Th. 1. First, we prove the following two lemmas.

**Lemma 12** \( C^2 \) is as expressive as the language of \( \mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id) \). In addition, the translation leads to no more than a linear increase in the size of the translated formula.

**Proof:** Similar to the proof in [18], we present the translation function from \( \mathcal{ALCQIO}(\sqcup, \sqcap, \neg, |, id) \) to \( C^2 \) in several variants that behave as follows: \( \tau^x() \) makes \( x \) be the free variable of the monadic predicate, which is produced for its argument concept, while \( \tau^y() \) makes the free variable be \( y \). So, for an atomic concept \( AC \in C_N \), \( \tau^x(C) = C(x) \), while \( \tau^y(C) = C(y) \). For an atomic role \( R \in C_N \), \( \tau^{x,y}(R) \) produces a dyadic predicate \( R(x, y) \), while \( \tau^{y,x}(R) \) produces a dyadic predicate \( R(y, x) \). The translation functions \( \tau^x() \), \( \tau^y() \), and \( \tau^{x,y}() \) are presented in the following two tables (Table A.1 and Table A.2). \( \tau^{y,x}() \) is obtained from \( \tau^{x,y}() \) by simultaneously exchanging all occurrences of \( x \) and \( y \) (whether free or bound).
The translation function \( \tau() \) can now be defined simply as
\[
\tau(C) \overset{\text{def}}{=} \tau^x(C)
\]
for any concept \( C \),
Appendix A. Proofs of Theorems in Chapter 2

\[ \tau(R) \overset{\text{def}}{=} \tau^{x,y}(R) \] for any role \( R \).

Then, the translation of terminological and assertional axioms can be defined as:

- \( \tau(C(b)) \overset{\text{def}}{=} \exists x. \tau^x(C) \land x = b \) for any concept assertion \( C(b) \);
- \( \tau(R(b, b')) \overset{\text{def}}{=} \exists x. \exists y. \tau^{x,y}(R) \land x = b \land y = b' \) for any role assertion \( R(b, b') \);
- \( \tau(C_1 \sqsubseteq C_2) \overset{\text{def}}{=} \forall x. \tau^x(C_1) \supset \tau^x(C_2) \) for any concept inclusion \( C_1 \sqsubseteq C_2 \) if any;
- \( \tau(C_1 \equiv C_2) \overset{\text{def}}{=} \forall x. \tau^x(C_1) \equiv \tau^x(C_2) \) for any concept equality \( C_1 \equiv C_2 \) if any;
- \( \tau(R_1 \sqsubseteq R_2) \overset{\text{def}}{=} \forall x. \forall y. \tau^{x,y}(R_1) \supset \tau^{x,y}(R_2) \) for any role inclusion \( R_1 \sqsubseteq R_2 \) if any.

For any DL interpretation \( I \), and the conventional FO interpretation \( I_1 \) such that \( \Delta^{I_1} = \Delta^I \) and \( AC^{I_1} = AC^I \) (\( R^{I_1} = R^I \), respectively ) for each atomic concept \( AC \) (atomic role \( R \), respectively), it is straightforward to prove by induction that \( (\phi)^I = (\tau(\phi))^{I_1} \) for any formula \( \phi \) in \( \text{ALCIQO}(\sqcup, \sqcap, \neg, |, \text{id}) \).

In addition, it is obvious that the translation from \( \text{AICQIO}(\sqcup, \sqcap, \neg, |, \text{id}) \) to \( C^2 \) can be done in linear time and causes no more than a linear increase in the size of the translated formula according to the translation function \( \tau \) defined above. \( \square \)

Lemma 13 The language of \( \text{AICQIO}(\sqcup, \sqcap, \neg, |, \text{id}) \) is as expressive as \( C^2 \). In addition, the translation leads to no more than a linear increase in the size of the translated formula.

Proof: We proceed by structural induction on the syntax of formulas in \( C^2 \) with up to two free variables \( x \) and \( y \). Table A.3 lists all possible kinds of formulas \( \Gamma(x) \) that have a single free variable \( x \), and shows how each kind is translated into a concept \( C_T \). Let \( N_C = \{ AC \mid AC(x) \text{ or } AC(y) \text{ is a monadic predicate in language } C^2 \} \), and \( N_R = \{ R \mid R(x, y) \text{ or } R(y, x) \text{ is a dyadic predicate in language } C^2 \} \). The translation of formulas with a single free variable \( y \) is identical, except for the case when \( \Gamma(y) \) is of the form \( \exists x. \Psi(x, y) \), \( (\exists \geq n x. \Psi(x, y) \), and \( \exists \leq n x. \Psi(x, y) \), respectively), when we need to invert the relationship represented by \( \Psi \). So, it is translated as \( (\exists (R_{\Psi})^- \cdot T \) (\( \geq n (R_{\Psi})^- \cdot T \), and \( \leq n (R_{\Psi})^- \cdot T \), respectively).
Appendix A. Proofs of Theorems in Chapter 2

<table>
<thead>
<tr>
<th>$\Gamma(x)$</th>
<th>$C_\Gamma$</th>
<th>$\Gamma(x)$</th>
<th>$C_\Gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC(x), AC \in NC$</td>
<td>$AC$</td>
<td>$\Psi() \land \Phi(x)$</td>
<td>$C_{\Psi()} \cap C_{\Phi}$</td>
</tr>
<tr>
<td>$R(x,b), R \in NR$</td>
<td>$\exists R, {b}$</td>
<td>$\Psi(x) \land \Phi(x)$</td>
<td>$C_{\Psi} \cap C_{\Phi}$</td>
</tr>
<tr>
<td>$R(b,x), R \in NR$</td>
<td>$\exists R^-, {b}$</td>
<td>$\exists y. \Psi(x,y)$</td>
<td>$\exists R_{\Psi}. \top$</td>
</tr>
<tr>
<td>$R(x,x), R \in NR$</td>
<td>$\exists (R \cap id(\top)). \top$</td>
<td>$\exists^\geq_n y. \Psi(x,y)$</td>
<td>$\geq n R_{\Psi}. \top$</td>
</tr>
<tr>
<td>$x = b$</td>
<td>${b}$</td>
<td>$\exists^\leq_n y. \Psi(x,y)$</td>
<td>$\leq n R_{\Psi}. \top$</td>
</tr>
<tr>
<td>$x = x$</td>
<td>$\top$</td>
<td>$\exists y. \Psi(x)$</td>
<td>$C_{\Psi}$</td>
</tr>
<tr>
<td>$\neg \Psi(x)$</td>
<td>$\neg C_{\Psi}$</td>
<td>$\exists^\geq_n y. \Psi(x)$</td>
<td>$C_{\Psi}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\exists^\leq_n y. \Psi(x)$</td>
<td>$C_{\Psi}$</td>
</tr>
</tbody>
</table>

Table A.3: A translation from $C^2$ to $\mathcal{ALCQIO}$($\sqcup, \sqcap, \neg, \mid, id$) for formulas with a single free variable $x$.

Formulae of the form $\Gamma(x, y)$ with two free variables are translated to roles $R_\Gamma$ relating $x$ to $y$ according to Table A.4. In Table A.4, notice that the role constructor $C_1 \times C_2$ for

<table>
<thead>
<tr>
<th>$\Gamma(x, y)$</th>
<th>$R_\Gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(x,y), R \in NR$</td>
<td>$R$</td>
</tr>
<tr>
<td>$R(y,x), R \in NR$</td>
<td>$R^-$</td>
</tr>
<tr>
<td>$x = y$</td>
<td>$id(\top)$</td>
</tr>
<tr>
<td>$\neg \Psi(x,y)$</td>
<td>$\neg R_{\Psi}$</td>
</tr>
<tr>
<td>$\Psi(x) \land \Phi(y)$</td>
<td>$C_{\Psi} \times C_{\Phi}$</td>
</tr>
<tr>
<td>$\Psi(x,y) \land \Phi()$</td>
<td>$R_{\Psi} \sqcap R_{\Phi()}$</td>
</tr>
<tr>
<td>$\Psi(x,y) \land \Phi(x)$</td>
<td>$R_{\Psi} \sqcap (C_{\Phi} \times \top)$</td>
</tr>
<tr>
<td>$\Psi(x,y) \land \Phi(y)$</td>
<td>$R_{\Psi} \sqcap (\top \times C_{\Phi})$</td>
</tr>
<tr>
<td>$\Psi(x,y) \land \Phi(x,y)$</td>
<td>$R_{\Psi} \sqcap R_{\Phi}$</td>
</tr>
</tbody>
</table>

Table A.4: A translation from $C^2$ to $\mathcal{ALCQIO}$($\sqcup, \sqcap, \neg, \mid, id$) for formulas with two free variables.
any two concepts $C_1$ and $C_2$ is introduced in [18], whose semantics is defined as $C_1^\mathcal{I} \times C_2^\mathcal{I}$ given any interpretation $\mathcal{I}$. It is easy to see that $\times$ can be replaced using the standard role constructors in $\mathsf{ALCQIO}(\sqcup, \sqcap, \neg, |, \text{id})$, that is,

$$C_1 \times C_2 \overset{\text{def}}{=} ((R \sqcup \neg R)|_{C_1})^\mathcal{I} \sqcap (R \sqcup \neg R)|_{C_2}$$

for any atomic role $R \in N_R$.

When a formula $\Gamma()$ without free variables occurs as a conjunct, then the number of free variables (1 or 2) in its context determines its translation: a concept or a role. In the case that a concept is desired, we need a translated concept $C_{\Gamma()}$ with the property that for any conventional $C^2$ interpretation $\mathcal{I}_1$ and a DL interpretation $\mathcal{I}$ such that $\Delta_{\mathcal{I}_1} = \Delta^\mathcal{I}$ and $AC_{\mathcal{I}_1} = AC^\mathcal{I}$ ($R_{\mathcal{I}_1} = R^\mathcal{I}$, respectively) for each atomic concept $AC$ (atomic role $R$, respectively), $\mathcal{I}_1 \models \Gamma() \equiv \text{true}$ iff $(C_{\Gamma})^\mathcal{I} = \Delta^\mathcal{I}$, and $\mathcal{I}_1 \models \Gamma() \equiv \text{false}$ iff $(C_{\Gamma})^\mathcal{I} = \emptyset$. Table A.5 provides such translations. In contexts where we require roles, the translation is just $R_{\Gamma()} = C_{\Gamma()} \times C_{\Gamma()}$.

For any formula $\phi$ in $C^2$, the translation function $\text{transl}$ can now be defined as:

$\text{transl}(\phi) = C_{\phi}$ if $\phi$ has no free variables, or has only one free variable $x$ or $y$; and $\text{transl}(\phi) = R_{\phi}$ if $\phi$ has exactly two free variables.

We can prove by induction that for any conventional $C^2$ interpretation $\mathcal{I}_1$ and a DL interpretation $\mathcal{I}$ such that $\Delta_{\mathcal{I}_1} = \Delta^\mathcal{I}$ and $AC_{\mathcal{I}_1} = AC^\mathcal{I}$ ($R_{\mathcal{I}_1} = R^\mathcal{I}$, respectively) for each atomic concept $AC$ (atomic role $R$, respectively), we have $\mathcal{I}_1 \models \phi \equiv \text{true}$ iff $(\text{transl}(\phi))^\mathcal{I} = \Delta^\mathcal{I}$ and $\mathcal{I}_1 \models \phi \equiv \text{false}$ iff $(\text{transl}(\phi))^\mathcal{I} = \emptyset$ for any (closed) sentence $\phi$.

It is obvious that the translation from $C^2$ to $\mathsf{ALCQIO}(\sqcup, \sqcap, \neg, |, \text{id})$ can be done in linear time and causes no more than a linear increase in the size of the translated formula according to the translation function $\tau$ defined above. \[\square\]

**Theorem 1** (Section 2.2.2) The description logic $\mathsf{ALCQIO}(\sqcup, \sqcap, \neg, |, \text{id})$ and $C^2$ are equally expressive. In addition, translation in both directions leads to no more than a
## Appendix A. Proofs of Theorems in Chapter 2

<table>
<thead>
<tr>
<th>( \Gamma() )</th>
<th>( C_{\Gamma()} )</th>
<th>( \Gamma() )</th>
<th>( C_{\Gamma()} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{true} )</td>
<td>( \top )</td>
<td>( \exists x. \Psi() )</td>
<td>( C_{\Psi()} )</td>
</tr>
<tr>
<td>( \text{false} )</td>
<td>( \bot )</td>
<td>( \exists y. \Psi() )</td>
<td>( C_{\Psi()} )</td>
</tr>
<tr>
<td>( C(b) )</td>
<td>( \forall (\top \times {b}). C )</td>
<td>( \exists^2 x. \Psi(x) \geq nU.C_{\Psi} )</td>
<td></td>
</tr>
<tr>
<td>( R(b, b) )</td>
<td>( \forall (\top \times {b}). (\exists R. {b}) )</td>
<td>( \exists^2 y. \Psi(y) \geq nU.C_{\Psi} )</td>
<td></td>
</tr>
<tr>
<td>( R(b', b) )</td>
<td>( \forall (\top \times {b'}). (\exists R. {b}) )</td>
<td>( \exists^2 x. \Psi() )</td>
<td>( C_{\Psi()} )</td>
</tr>
<tr>
<td>( b = b )</td>
<td>( \top )</td>
<td>( \exists^2 y. \Psi() )</td>
<td>( C_{\Psi()} )</td>
</tr>
<tr>
<td>( b' = b )</td>
<td>( \bot )</td>
<td>( \exists^2 x. \Psi(x) \leq nU.C_{\Psi} )</td>
<td></td>
</tr>
<tr>
<td>( \neg \Psi() )</td>
<td>( \neg C_{\Psi()} )</td>
<td>( \exists y. \Psi(y) \leq nU.C_{\Psi} )</td>
<td></td>
</tr>
<tr>
<td>( \Psi() \land \Phi() )</td>
<td>( R_{\Psi()} \sqcap R_{\Phi()} )</td>
<td>( \exists^2 x. \Psi() )</td>
<td>( C_{\Psi()} )</td>
</tr>
<tr>
<td>( \exists x. \Psi(x) )</td>
<td>( \forall U.C_{\Psi} )</td>
<td>( \exists y. \Psi(y) )</td>
<td>( C_{\Psi()} )</td>
</tr>
<tr>
<td>( \exists y. \Psi(y) )</td>
<td>( \forall U.C_{\Psi} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| \( \neg \Psi() \) | \( \neg C_{\Psi()} \) | \( \exists^2 x. \Psi(x) \leq nU.C_{\Psi} \) |
| \( \Psi() \land \Phi() \) | \( R_{\Psi()} \sqcap R_{\Phi()} \) | \( \exists^2 x. \Psi() \) | \( C_{\Psi()} \) |
| \( \exists x. \Psi(x) \) | \( \forall U.C_{\Psi} \) | \( \exists y. \Psi(y) \) | \( C_{\Psi()} \) |
| \( \exists y. \Psi(y) \) | \( \forall U.C_{\Psi} \) | |

Table A.5: A translation from \( C^2 \) to \( \mathbf{ALCQIO}(\sqcup, \sqcap, \neg, \mid, \text{id}) \) for formulas without free variables.

**linear increase in the size of the translated formula.**

**Proof:** It is a direct consequence of combining Lemma 12 and Lemma 13. \( \square \)
Appendix B

Proofs of Lemmas and Theorems in Chapter 3

B.1 The Correctness of the Modified Regression Operator

In this section we provide a detailed proof for Th. 3 in Section 3.3.1.

**Theorem 3** (Section 3.3.1) Suppose $W$ is an $\mathcal{L}_{sc}^{C^2}$ regressable sentence with the background $BAT$ $\mathcal{D}$ in language $\mathcal{L}_{sc}^{C^2}$. Then, $\mathcal{R}^{C^2}[W]$ is an $\mathcal{L}_{sc}^{C^2}$ sentence uniform in $S_0$ and it is a $C^2$ sentence when the situation argument $S_0$ is suppressed. Moreover, $\mathcal{D} \models W \equiv \mathcal{R}^{C^2}[W]$.

**Proof:** We prove it by induction on the number of regression steps.

Base case: It takes one step to terminate the regression.

If $W$ is of the form $A_1(\vec{t}) = A_2(\vec{t}^\prime)$ for some action function symbols $A_1$ and $A_2$, then there are three sub-cases:

1. If $A_1 \neq A_2$, $\mathcal{R}^{C^2}[W] = false$ (by definition), which is uniform in $S_0$ and is a $C^2$ sentence. Note that $\mathcal{D} \models W \equiv false$ by the unique name axioms for actions in $\mathcal{D}$. Hence, $\mathcal{D} \models \mathcal{R}^{C^2}[W] \equiv W$. 
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

(2) If $A_1 = A_2$ and $A_1, A_2$ are constant action functions, $\mathcal{R}^{C^2}[W] = true$ (by definition), which is uniform in $S_0$ and is a $C^2$ sentence. Note that $\mathcal{D} \models W \equiv true$ by the unique name axioms for actions in $\mathcal{D}$. Hence, $\mathcal{D} \models \mathcal{R}^{C^2}[W] \equiv W$.

(3) Otherwise, i.e., $A_1 = A_2$ and $A_1, A_2$ are not constant action functions, then $\mathcal{R}^{C^2}[W] = \bigwedge_{i=1}^{[\vec{t}]} t_i = t'_i$ (by definition), which is uniform in $S_0$ and is a $C^2$ sentence. Note that $\mathcal{D} \models W \equiv \bigwedge_{i=1}^{[\vec{t}]} t_i = t'_i$ by the unique name axioms for actions in $\mathcal{D}$. Hence, $\mathcal{D} \models \mathcal{R}^{C^2}[W] \equiv W$.

Otherwise, $W$ is any other situation independent atom (including equality between object terms) or $W$ is a concept or role uniform in $S_0$, so $\mathcal{R}^{C^2}[W] = W$ (by definition), and it is obvious that $\mathcal{R}^{C^2}[W]$ is uniform in $S_0$ and is a $C^2$ formula when $S_0$ is suppressed. Moreover, $\mathcal{D} \models \mathcal{R}^{C^2}[W] \equiv W$.

Inductive step: Assume that our theorem is true for any regression that takes no more than $n$ steps ($n \geq 1$), now we prove it is true for any regression that takes $n + 1$ steps.

There are several cases as follows.

a. $W$ is of the form $Poss(A(\vec{t}), \sigma)$, for terms of sort action and situation, respectively, in $\mathcal{L}^{C_2}_{sc}$. Assume that the precondition axiom for action function $A(\vec{x})$ is of the form $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$, where $\vec{x}$ is either empty, $x$, or $\langle x, y \rangle$. There are four sub-cases:

(a.1) If $\vec{t} = \langle x, x \rangle$, then

$$\mathcal{D} \models \mathcal{R}^{C^2}[W] \equiv \mathcal{R}^{C^2}[\exists y.x=y \land Poss(A(x, y), \sigma)]$$

$$= \exists y.x=y \land \mathcal{R}^{C^2}[\Pi_A(x, y, \sigma)] \quad \text{(by the definition of $\mathcal{R}^{C^2}$)}$$

$$\equiv \exists y.x=y \land \Pi_A(x, y, \sigma) \quad \text{(by the induction hypothesis)}$$

$$\equiv \exists y.x=y \land Poss(A(x, y), \sigma) \quad \text{(by $D_{ap}$)}$$

$$\equiv Poss(A(x, x), \sigma)$$

$$= W$$

Moreover, by the induction hypothesis that $\mathcal{R}^{C^2}[\exists y.x=y \land \Pi_A(x, y, \sigma)]$ is uniform in $S_0$ and is a $C^2$ formula (when $S_0$ is suppressed), and so is $\mathcal{R}^{C^2}[W]$.

(a.2) Similarly to case (a.1) above, we can prove that the theorem is true if $\vec{t} = \langle y, y \rangle$. 


Appendix B. Proofs of Lemmas and Theorems in Chapter 3

(a.3) If \( \vec{t} \in \{y, \langle y, O \rangle, \langle O, x \rangle, \langle y, x \rangle \} \), we need to ensure the result of substituting \( \vec{t} \) into the precondition axiom is still logically equivalent to the original one. It can be proved case by case. We will just show one case as an example, and the rest of the cases can be proved similarly. For example, when \( \vec{t} \) is \( \langle y, O \rangle \), \( \vec{x} \) can only be \( \langle x, y \rangle \) in the precondition axiom. It is obvious that \( Poss(A(y, x), s) \equiv \widetilde{\Pi}_A(y, x, s) \) is logically equivalent to \( Poss(A(x, y), s) \equiv \Pi_A(x, y, s) \) by renaming all \( x \) with \( y \) and all \( y \) with \( x \) (free or bound). Hence, we are able to substitute \( \vec{t} \) into the precondition of \( Poss(A(y, x), s) \) without introducing new variables. Then,

\[
D \models \mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\widetilde{\Pi}_A(y, O, \sigma)] \quad (\text{by the definition of } \mathcal{R}^{C^2})
\]

\[
\equiv \widetilde{\Pi}_A(y, O, \sigma) \quad (\text{by the induction hypothesis})
\]

\[
\equiv Poss(A(y, O), \sigma) \quad (\text{by the renamed precondition axiom})
\]

\[
= W
\]

Moreover, by the induction hypothesis that \( \mathcal{R}^{C^2}[\widetilde{\Pi}_A(y, O, \sigma)] \) is uniform in \( S_0 \) and is a \( C^2 \) formula (when \( S_0 \) is suppressed), and so is \( \mathcal{R}^{C^2}[W] \).

(a.4) Otherwise, i.e., if \( \vec{t} \) either is empty or \( \vec{t} \in \{O, x, \langle x, y \rangle, \langle x, O \rangle, \langle O, y \rangle, \langle O, O_i \rangle \} \), it is obvious that we can substitute \( \vec{t} \) directly into the precondition axiom without causing any problem. That is,

\[
D \models \mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\Pi_A(\vec{t}, \sigma)] \quad (\text{by the definition of } \mathcal{R}^{C^2})
\]

\[
\equiv \Pi_A(\vec{t}, \sigma) \quad (\text{by the induction hypothesis})
\]

\[
\equiv Poss(A(\vec{t}), \sigma) \quad (\text{by } D_{ap})
\]

\[
= W
\]

Again, by using the induction hypothesis, \( \mathcal{R}^{C^2}[\Pi_A(\vec{t}, \sigma)] \) is uniform in \( S_0 \) and is a \( C^2 \) formula (when \( S_0 \) is suppressed), and so is \( \mathcal{R}^{C^2}[W] \).

b. \( W \) is a defined dynamic concept of the form \( G(t, \sigma) \) for some object term \( t \) and ground situation term \( \sigma \), and there must be a TBox axiom for \( G \) of the form \( G(x, s) \equiv \phi_G(x, s) \). Because of the restrictions of the language \( \mathcal{L}^{C^2}_{sc} \), term \( t \) can only be a variable \( x, y \) or a constant. There are two sub-cases.
(b.1) When \( t \in \{O, x\} \), it is obvious see that

\[
\mathcal{D} \models \mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\phi_G(t, \sigma)] \quad \text{(by the definition of } \mathcal{R}^{C^2})
\]

\[
\equiv \phi_G(t, \sigma) \quad \text{(by the induction hypothesis)}
\]

\[
\equiv G(t, \sigma) \quad \text{(by the TBox axiom)}
\]

\[
= W
\]

Again, by using the induction hypothesis, \( \mathcal{R}^{C^2}[\phi_G(t, \sigma)] \) is uniform in \( S_0 \) and is a \( C^2 \) formula (when \( S_0 \) is suppressed), and so is \( \mathcal{R}^{C^2}[W] \).

(b.2) When \( t \) is variable \( y \), then we can rename all \( x \) (\( y \), respectively) in the TBox axiom with \( y \) (\( x \), respectively), and still get an equivalent TBox axiom: \( G(y, s) \equiv \tilde{\phi}_G(y, s) \). Then,

\[
\mathcal{D} \models \mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\tilde{\phi}_G(y, \sigma)] \quad \text{(by the definition of } \mathcal{R}^{C^2})
\]

\[
\equiv \tilde{\phi}_G(y, \sigma) \quad \text{(by the induction hypothesis)}
\]

\[
\equiv G(y, \sigma) \quad \text{(by the renamed TBox axiom)}
\]

\[
= W
\]

Again, by using the induction hypothesis, \( \mathcal{R}^{C^2}[\tilde{\phi}_G(y, \sigma)] \) is uniform in \( S_0 \) and is a \( C^2 \) formula (when \( S_0 \) is suppressed), and so is \( \mathcal{R}^{C^2}[W] \).

c. \( W \) is a primitive dynamic concept (a dynamic role, respectively) of the form \( F(t_1, do(\alpha, \sigma)) \) (or \( F(t_1, t_2, do(\alpha, \sigma)) \), respectively) for some terms \( t_1 \) (and \( t_2 \)) of sort \( \text{object} \), ground term \( \alpha \) of sort \( \text{action} \) and ground term \( \sigma \) of sort \( \text{situation} \). There must be an SSA for fluent \( F \) of the form \( F(\tilde{x}, do(\alpha, s)) \equiv \Phi_F(\tilde{x}, \alpha, s) \), whose detailed syntax is Eq. (3.2). Because of the restriction of the language \( \mathcal{L}^{C^2}_sc \), the terms \( t_1 \) and \( t_2 \) can only be a variable \( x \), \( y \) or some constant \( O \). In fact, the discussion of sub-cases for a primitive dynamic concept \( F(t_1, do(\alpha, \sigma)) \) is very similar to the proof for defined concepts except that instead of using a TBox axiom, we will use the SSA of \( F \). The discussion of sub-cases for a dynamic role \( F(t_1, t_2, do(\alpha, \sigma)) \) is very similar to the proof for an atom of the form \( Poss(A(t_1, t_2), \sigma) \) except that instead of using precondition axioms, we use the SSA of \( F \). Since it is straightforward, details are
omitted here.

d. $W$ is not atomic, i.e., $W$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W'$, or $Q_v.W'$ where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol. This is the last case we need to consider for the inductive step. Therefore, it is obvious that there are four sub-cases depending on the different forms of $W$. Because the discussions for all sub-cases are very similar except that they use different logical constructors, we will provide details for one of the sub-cases, and omit the rest. As an example, we consider the sub-case that $W$ is of the form $W_1 \lor W_2$. Then,

$$
\mathcal{D} \models R^{C^2}[W] = R^{C^2}[W_1] \lor R^{C^2}[W_2] \quad \text{(by the definition of } R^{C^2})
$$

$$
\equiv W_1 \lor W_2 \quad \text{(by the induction hypothesis)}
$$

$$
= W
$$

Again, by using the induction hypothesis, $R^{C^2}[W_1]$ and $R^{C^2}[W_2]$ are uniform in $S_0$ and are both $C^2$ formulas (when $S_0$ is suppressed), hence $R^{C^2}[W]$ is still uniform in $S_0$ and is a $C^2$ formula (when $S_0$ is suppressed).

Overall, we proved that for any $\mathcal{L}^{C^2}_{sc}$ regressable sentence $W$ with the background BAT $\mathcal{D}$ in language $\mathcal{L}^{C^2}_{sc}$, $R^{C^2}[W]$ is an $\mathcal{L}^{C^2}_{sc}$ sentence uniform in $S_0$ and it is a $C^2$ sentence when the situation argument $S_0$ is suppressed. Moreover, $\mathcal{D} \models W \equiv R^{C^2}[W]. \quad \Box$

### B.2 $\mathcal{ALCO}(U)$ and $FO_{DL}$ are Equally Expressive

In this section, we prove Lemma 1 presented in Section 3.3.4. Notice that in the proof of this Lemma, we provide purely syntactic translation functions between $\mathcal{ALCO}(U)$ and $FO_{DL}$.

**Lemma 1** (Section 3.3.4) There are syntactic translations between $FO_{DL}$ and the DL language $\mathcal{ALCO}(U)$, i.e., they are equally expressive. Moreover, such translation leads to
no more than a linear increase in the size of the translated formula.

**Proof:** We first prove that there is a syntactic translation function from $\mathcal{ALCO}(U)$ to $FO_{DL}$.

A syntactic translation $\tau$ from $\mathcal{ALCO}(U)$ to $FO_{DL}$ for any concept $C$ is defined as follows: $\tau(C) \overset{\text{def}}{=} \tau^x(C)$ for any concept $C$. $\tau^x()$ makes $x$ be the free variable of the monadic predicate, which is produced for its argument concept (see Table B.1). During translation we also need a variant of $\tau$ – $\tau()$ makes $y$ be the free variable of the monadic predicate (see Table B.1).

<table>
<thead>
<tr>
<th>Term $C$</th>
<th>$\tau^x(C)$</th>
<th>$\tau^y(C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC, AC \in NC$</td>
<td>$AC(x)$</td>
<td>$AC(y)$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$true$</td>
<td>$true$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$false$</td>
<td>$false$</td>
</tr>
<tr>
<td>${b}$</td>
<td>$x=b$</td>
<td>$y=b$</td>
</tr>
<tr>
<td>$\neg C_1$</td>
<td>$\neg \tau^x(C_1)$</td>
<td>$\neg \tau^y(C_1)$</td>
</tr>
<tr>
<td>$C_1 \sqcap C_2$</td>
<td>$\tau^x(C_1) \land \tau^x(C_2)$</td>
<td>$\tau^y(C_1) \land \tau^y(C_2)$</td>
</tr>
<tr>
<td>$\exists R.C_1, R \in NR$</td>
<td>$\exists y. R(x,y) \land \tau^y(C_1)$</td>
<td>$\exists x. R(x,y) \land \tau^x(C_1)$</td>
</tr>
<tr>
<td>$\forall R.C_1, R \in NR$</td>
<td>$\forall y. R(x,y) \lor \tau^y(C_1)$</td>
<td>$\forall x. R(x,y) \lor \tau^x(C_1)$</td>
</tr>
<tr>
<td>$\exists U.C_1$</td>
<td>$\exists y. \tau^y(C_1)$</td>
<td>$\exists x. \tau^x(C_1)$</td>
</tr>
<tr>
<td>$\forall U.C_1$</td>
<td>$\forall y. \tau^y(C_1)$</td>
<td>$\forall x. \tau^x(C_1)$</td>
</tr>
</tbody>
</table>

Table B.1: A syntactic translation from $\mathcal{ALCO}(U)$ to $FO_{DL}$.

Then, the translation of terminological and assertional axioms can be defined as:

$\tau(C(b)) \overset{\text{def}}{=} \exists x. \tau^x(C) \land x = b$ for any concept assertion $C(b)$;

$\tau(R(b,b')) \overset{\text{def}}{=} \exists x. x = b \land \exists y. \tau^x(y)(R) \land y = b'$ for any role assertion $R(b,b')$;

$\tau(C_1 \sqsubseteq C_2) \overset{\text{def}}{=} \forall x. \neg \tau^x(C_1) \lor \tau^x(C_2)$ for any GCI axiom $C_1 \sqsubseteq C_2$ if there is any;
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

\( \tau(C_1 \equiv C_2) \stackrel{\text{def}}{=} (\forall x. \neg \tau^x(C_1) \lor \tau^x(C_2)) \land (\forall x. \neg \tau^x(C_2) \lor \tau^x(C_1)) \) for any concept equality axiom \( C_1 \equiv C_2 \) if there is any.

In addition, according to the definition of \( \tau \) in Table B.1 and the fact that there are no nested appearances of \( \sqsubseteq \) and \( \equiv \) in DL KBs, it is obvious that the translation from \( \mathcal{ALCO}(U) \) to \( FO_{DL} \) can be done in linear time and causes no more than a linear increase in the size of the translated formula.

Now, we prove that there is a syntactic translation function from \( FO_{DL} \) to \( \mathcal{ALCO}(U) \).

A syntactic translation \( \pi \) from \( FO_{DL} \) to \( \mathcal{ALCO}(U) \) for any formula \( \Phi \in FO_{DL} \) is defined in Table B.2.

Moreover, it is obvious that the translation from \( FO_{DL} \) to \( \mathcal{ALCO}(U) \) can be done in linear time and causes no more than a linear increase in the size of the translated formula according to the translation function \( \pi \) defined above.

\[ \Box \]

B.3 Restricting Syntax of BATs to Gain Computational Advantages

In this section, we will prove Lemma 2 in Section 3.3.4. But first, we define an operator \( \epsilon \) on any \( \mathcal{L}_{sc}^{C^2} \) regressive formula \( W \), such that it will replace all atomic formula of the form \( A_1(\vec{t}) = A_2(\vec{t}') \) for some action terms \( A_1(\vec{t}) \) and \( A_2(\vec{t}') \) using the unique name axioms for actions in \( D_{una} \) for any given BAT \( D \).

**Definition 24** For any given BAT \( D \) and an \( \mathcal{L}_{sc}^{C^2} \) regressive formula \( W \) in it, we define \( \epsilon \) recursively as follows:

- If \( W \) is of the form \( A_1(\vec{t}) = A_2(\vec{t}') \) for some action terms \( A_1(\vec{t}) \) and \( A_2(\vec{t}') \) (i.e., equality between action terms), then
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

<table>
<thead>
<tr>
<th>$\Phi$</th>
<th>$\pi(\Phi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC(x)$, $AC(x)$ is atomic</td>
<td>$AC$</td>
</tr>
<tr>
<td>$AC(y)$, $AC(y)$ is atomic</td>
<td>$AC$</td>
</tr>
<tr>
<td>$true$</td>
<td>$\top$</td>
</tr>
<tr>
<td>$false$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$x=b$, $b$ is a constant</td>
<td>${b}$</td>
</tr>
<tr>
<td>$y=b$, $b$ is a constant</td>
<td>${b}$</td>
</tr>
<tr>
<td>$\neg \Psi$</td>
<td>$\neg \pi(\Psi)$</td>
</tr>
<tr>
<td>$\Psi_1 \lor \Psi_2$</td>
<td>$\pi(\Psi_1) \cup \pi(\Psi_2)$</td>
</tr>
<tr>
<td>$\Psi_1 \land \Psi_2$</td>
<td>$\pi(\Psi_1) \cap \pi(\Psi_2)$</td>
</tr>
<tr>
<td>$\exists y. R(x, y) \land \Psi(y), R \in N_R$</td>
<td>$\exists R. \pi(\Psi(y))$</td>
</tr>
<tr>
<td>$\exists x. R(y, x) \land \Psi(x), R \in N_R$</td>
<td>$\exists R. \pi(\Psi(x))$</td>
</tr>
<tr>
<td>$\forall y. R(x, y) \lor \Psi(y), R \in N_R$</td>
<td>$\forall R. \pi(\Psi(y))$</td>
</tr>
<tr>
<td>$\forall x. R(y, x) \lor \Psi(x), R \in N_R$</td>
<td>$\forall R. \pi(\Psi(x))$</td>
</tr>
<tr>
<td>$\exists y. \Psi(y), \Psi(y)$ has only one free variable $y$</td>
<td>$\exists U. \pi(\Psi(y))$</td>
</tr>
<tr>
<td>$\exists x. \Psi(x), \Psi(x)$ has only one free variable $x$</td>
<td>$\exists U. \pi(\Psi(x))$</td>
</tr>
<tr>
<td>$\forall y. \Psi(y), \Psi(y)$ has only one free variable $y$</td>
<td>$\forall U. \pi(\Psi(y))$</td>
</tr>
<tr>
<td>$\forall x. \Psi(x), \Psi(x)$ has only one free variable $x$</td>
<td>$\forall U. \pi(\Psi(x))$</td>
</tr>
</tbody>
</table>

Table B.2: A syntactic translation from $FO_{DL}$ to $\mathcal{ALCO}(U)$.

$\epsilon[W] = \begin{cases} 
false & \text{if } A_1 \neq A_2, \\
true & \text{if } A_1 = A_2 \text{ and } A_1, A_2 \text{ are constant action functions,} \\
\left| \prod_{i=1}^{\mid W \mid} t_i = t_i' \right| & \text{otherwise.}
\end{cases}$

Otherwise, if $W$ is any other situation independent atom, then

$\epsilon[W] = W$.

- Otherwise, if $W$ is not atomic, i.e., $W$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W'$,
or $Qv.W'$ where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol, then

$$
\epsilon[W_1 \lor W_2] = \epsilon[W_1] \lor \epsilon[W_2], \quad \epsilon[-W'] = -\epsilon[W'],
$$

$$
\epsilon[W_1 \land W_2] = \epsilon[W_1] \land \epsilon[W_2], \quad \epsilon[Qv.W'] = Qv.\epsilon[W'].
$$

Note that $\epsilon$ can be considered as performing one step of $L_{sc}^{C^2}$ regression on equalities between action terms in the given $L_{sc}^{C^2}$ regressable formula $W$, and it is easy to see that if $W$ is uniform in situation $S$, then $\epsilon[W]$ is still uniform in situation $S$. Moreover, we can prove the following property for $\epsilon$.

**Property 3** For any given BAT $D$ and an $L_{sc}^{C^2}$ regressable formula $W$ in $D$, we have that $\epsilon[W]$ is still $L_{sc}^{C^2}$ regressable and $D \models W \equiv \epsilon[W]$.

**Proof:** It is easy to prove by induction on the structure of $W$.

**Base case:** If $W$ is atomic, there are two sub-cases.

(1) $W$ is of the form $A_1(t) = A_2(t)$ for some action terms $A_1(t)$ and $A_2(t)$. If $A_1 \neq A_2$, we have $D \models W \equiv false$ by axioms in $D_{una}$, which is $D \models W \equiv \epsilon[W]$, since $\epsilon[W] = false$ by the definition of $\epsilon$; else, if $A_1 = A_2$ and $A_1, A_2$ are constant action functions, then by axioms in $D_{una}$, $D \models W \equiv true$, therefore $D \models W \equiv \epsilon[W]$ according to the definition of $\epsilon$; otherwise, $D \models W \equiv \bigwedge_{i=1}^{[\ell]} t_i = t'_i$ by axioms in $D_{una}$, which is $D \models W \equiv \epsilon[W]$, since $\epsilon[W] = \bigwedge_{i=1}^{[\ell]} t_i = t'_i$ by the definition of $\epsilon$. Moreover, it is obvious that $\epsilon[W]$ is still $L_{sc}^{C^2}$ regressable.

(2) Otherwise, $W$ is atomic and not of the above form. By the definition of $\epsilon$, we have $\epsilon[W] = W$, hence $D \models W \equiv \epsilon[W]$. Moreover, it is obvious that $\epsilon[W]$ is still $L_{sc}^{C^2}$ regressable.

**Inductive step:** $W$ is not atomic and $W$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W'$, or $Qv.W'$ where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol. Then for each sub-case, it is easy to prove that $D \models W \equiv \epsilon[W]$ by the induction hypothesis. For instance, if $W$ is of the form $W_1 \lor W_2$, then
\[ \mathcal{D} \models W = W_1 \lor W_2 \]
\[ \equiv \epsilon[W_1] \lor \epsilon[W_2] \quad \text{(by the induction hypothesis)} \]
\[ = \epsilon[W_1 \lor W_2] \quad \text{(by the definition of } \epsilon) \]
\[ = \epsilon[W]. \]

Moreover, it is obvious that \( \epsilon[W] \) is still \( \mathcal{L}_{sc}^{C^2} \) regressable by the induction hypothesis that \( \epsilon[W_1] \) and \( \epsilon[W_2] \) are both \( \mathcal{L}_{sc}^{C^2} \) regressable.

It is easy to see that for other sub-cases, such as \( W_1 \land W_2, \neg W', \) and \( Qv.W' \) where \( Q \) represents a quantifier (including counting quantifiers), the proof is very similar to the sub-case of \( W_1 \lor W_2 \) and therefore details are omitted here.

Overall, \( \mathcal{D} \models W \equiv \epsilon[W] \) for any \( \mathcal{L}_{sc}^{C^2} \) regressable formula \( W \) in \( \mathcal{D} \) and \( \epsilon[W] \) is still \( \mathcal{L}_{sc}^{C^2} \) regressable. \( \Box \)

We prove the following lemma that will be useful when proving Lemma 2 in Section 3.3.4. Notice that the lemma says that the regression of \( W \) is the same as (not just equivalent to) the regression of \( \epsilon[W] \).

**Lemma 14** Consider any given BAT \( \mathcal{D} \), the \( \mathcal{L}_{sc}^{C^2} \) regression operator \( \mathcal{R}_{C^2} \) defined in Section 3.3.1, and any \( \mathcal{L}_{sc}^{C^2} \) regressable formula \( W \) in \( \mathcal{D} \). Then, \( \mathcal{R}_{C^2}[W] = \mathcal{R}_{C^2}[\epsilon[W]] \).

**Proof:** It is easy to prove by induction on the structure of \( W \).

Base case: If \( W \) is atomic, there are two sub-cases.

1. \( W \) is of the form \( A_1(\vec{t}) = A_2(\vec{t'}) \) for some action terms \( A_1(\vec{t}) \) and \( A_2(\vec{t}) \). If \( A_1 \neq A_2 \), we have \( \mathcal{R}_{C^2}[W] = \text{false} \) by the definition of \( \mathcal{R}_{C^2} \) in Section 3.3.1, and \( \mathcal{R}_{C^2}[\epsilon[W]] = \mathcal{R}_{C^2}[\text{false}] = \text{false} \) by the definitions of \( \epsilon \) and \( \mathcal{R}_{C^2} \), therefore, \( \mathcal{R}_{C^2}[W] = \mathcal{R}_{C^2}[\epsilon[W]] \);

else, if \( A_1 = A_2 \) and \( A_1, A_2 \) are constant action functions, by the definition of \( \epsilon \) and \( \mathcal{R}_{C^2} \), it is easy to see that \( \mathcal{R}_{C^2}[W] = \mathcal{R}_{C^2}[\epsilon[W]] = \text{true} \);

otherwise, we have \( \mathcal{R}_{C^2}[W] = \mathcal{R}_{C^2}[\bigwedge_{i=1}^{[\parallel]} t_i = t'_i] \) by the definition of \( \mathcal{R}_{C^2} \),

and since \( \mathcal{R}_{C^2}[\epsilon[W]] = \mathcal{R}_{C^2}[\bigwedge_{i=1}^{[\parallel]} t_i = t'_i] \) by the definition of \( \epsilon \) and \( \mathcal{R}_{C^2} \), it is easy to see that \( \mathcal{R}_{C^2}[W] = \mathcal{R}_{C^2}[\epsilon[W]] \).
(2) Otherwise, $W$ is atomic and not of the above form, by the definition of $\epsilon$, we have $\epsilon[W] = W$, hence $R^{C_2}[W] = R^{C_2}[\epsilon[W]]$.

**Inductive step:** $W$ is not atomic and $W$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W'$, or $Qv.W'$ where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol. Then for each sub-case, it is easy to prove that $R^{C_2}[W] = R^{C_2}[\epsilon[W]]$ by the induction hypothesis. For instance, if $W$ is of the form $W_1 \lor W_2$, then

\[
R^{C_2}[W] = R^{C_2}[W_1] \lor R^{C_2}[W_2] \quad \text{(by the definition of } R^{C_2})
\]

\[
= R^{C_2}[\epsilon[W_1]] \lor R^{C_2}[\epsilon[W_2]] \quad \text{(by the induction hypothesis)}
\]

\[
= R^{C_2}[\epsilon[W_1] \lor \epsilon[W_2]] \quad \text{(by the definition of } R^{C_2})
\]

\[
= R^{C_2}[\epsilon[W_1 \lor W_2]] \quad \text{(by the definition of } \epsilon)
\]

\[
= R^{C_2}[\epsilon[W]].
\]

It is easy to see that for other sub-cases, such as $W_1 \land W_2$, $\neg W'$, and $Qv.W'$ where $Q$ represents a quantifier (including counting quantifiers), the proof is very similar to the sub-case of $W_1 \lor W_2$ and therefore details are omitted here.

Overall, $R^{C_2}[W] = R^{C_2}[\epsilon[W]]$ for any $L^{C_2}_{sc}$ regressable formula $W$ in $D$. \hfill \Box

Moreover, according to the definition of $\epsilon$, it is straightforward to prove the following property of $\epsilon$. Because the proof is rather obvious, it is omitted here.

**Property 4** Given any $L^{C_2}_{sc}$ regressable formula $W$ whose size is $m$, i.e., $m = \text{size}(W)$, it takes no more than $m$ steps to obtain $\epsilon[W]$, and the size of $\epsilon[W]$ is no more than $3m$.

We also recursively define a one-step regression operator $\rho$ for any $L^{C_2}_{sc}$ regressable formula $W$, which has no appearances of Poss, such that it performs one step of $L^{C_2}_{sc}$ regression on each fluent in $W$. This operator $\rho$ will also be useful in the proof of Lemma 2. The formal definition of $\rho$ is as follows, where $\sigma$ denotes the term of sort situation, and $\alpha$ denotes the term of sort action.

**Definition 25** Given a BAT $D$ in

- If $W$ is not atomic, i.e., $W$ is of the form $W_1 \lor W_2$, $W_1 \land W_2$, $\neg W'$, or $Qv.W'$.
where $Q$ represents a quantifier (including counting quantifiers) and $v$ represents a variable symbol, then

$$
\rho[W_1 \lor W_2] = \rho[W_1] \lor \rho[W_2], \quad \rho[\neg W'] = \neg \rho[W'],
$$

$$
\rho[W_1 \land W_2] = \rho[W_1] \land \rho[W_2], \quad \rho[Qv.W] = Qv.\rho[W'].
$$

- Otherwise, $W$ is an atom. There are several cases.

  a. If $W$ is a situation independent atom, or $W$ is a concept or role uniform in $S_0$, then $\rho[W] = W$.

  b. If $W$ is a defined dynamic concept, so it has the form $G(t, \sigma)$ for some object term $t$ and situation term $\sigma$, then there must be a TBox axiom for $G$ of the form $G(x, s) \equiv \phi_G(x, s)$. Because of the restrictions of the language $\mathcal{L}_{sc}^{C^2}$, term $t$ can only be a variable $x$, $y$ or a constant. Then, we use the lazy unfolding technique:

$$
\rho[W] = \begin{cases} 
\rho[\phi_G(t, \sigma)] & \text{if } t \text{ is not variable } y, \\
\tilde{\rho}[\phi_G(y, \sigma)] & \text{otherwise.}
\end{cases}
$$

  c. If $W$ is a primitive dynamic concept (a dynamic role, respectively), it has the form $F(t_1, do(\alpha, \sigma))$ or $F(t_1, t_2, do(\alpha, \sigma))$ for some terms $t_1$ (and $t_2$) of sort object, term $\alpha$ of sort action and term $\sigma$ of sort situation. Then there must be an SSA for fluent $F$, whose detailed syntax is Eq. (3.2). Because of the restriction of the language $\mathcal{L}_{sc}^{C^2}$, the terms $t_1$ and $t_2$ can only be a variable $x$, $y$ or a constant $O$ and $\alpha$ can only be an action function with no more than two arguments of sort object.

Then, when $W$ is a concept,

$$
\rho[W] = \begin{cases} 
\Phi_F(t_1, \alpha, \sigma) & \text{if } t_1 \text{ is not variable } y, \\
\tilde{\Phi}_F(y, \alpha, \sigma) & \text{otherwise, i.e., if } t_1 = y;
\end{cases}
$$

and, when $W$ is a role,

$$
\rho[W] = \begin{cases} 
(\exists y)(x = y \land \Phi_F(x, y, \alpha, \sigma)) & \text{if } t_1 = x, t_2 = x, \\
(\exists x)(y = x \land \Phi_F(x, y, \alpha, \sigma)) & \text{if } t_1 = y, t_2 = y, \\
\tilde{\Phi}_F(y, t_2, \alpha, \sigma) & \text{if } t_1 = y, t_2 \in \{x, O\} \text{ or } t_1 = O, t_2 = x, \\
\Phi_F(t_1, t_2, \alpha, \sigma) & \text{otherwise.}
\end{cases}
$$
Note that the operator $\rho$ ($\mathcal{R}^{C^2}$, respectively) is generally defined for any $C^2$ regressable formula. For some restricted BATs, when applying $\mathcal{R}^{C^2}$ to any regressable formulas that is in $FO_{DL}$ or $FO_{DL+}$ respectively (with any situation term suppressed), below we will show that the resulting formula is still in $FO_{DL}$ or $FO_{DL+}$ respectively (with any situation term suppressed) via $\epsilon$ and $\rho$. In particular, notice that any formula in $FO_{DL}$ or $FO_{DL+}$ respectively does not have any predicate of the form $x = x$, $y = y$, $R(x, x)$ or $R(y, y)$. Hence, we do not need to worry about the first two sub-cases of the one-step regression $\rho$ for roles, where object terms can be $(x, x)$ or $(y, y)$.

Similar to the proof of Property 3, we can prove the following property for $\rho$ by using induction on the structure of formulas.

**Property 5** For any given BAT $\mathcal{D}$ and an $\mathcal{L}^{C^2}_{sc}$ regressable formula $W$ in $\mathcal{D}$, we have that $\rho[W]$ is still $\mathcal{L}^{C^2}_{sc}$ regressable and $\mathcal{D} \models W \equiv \rho[W]$.

In addition, also using induction on the structure of the formulas, it is straightforward to prove the following property, which is useful in the proof of Lemma 2. Because the proof is rather obvious, it is omitted here.

**Property 6** Consider a BAT $\mathcal{D}$ in the language of $\mathcal{L}^{C^2}_{sc}$, if a given $\mathcal{L}^{C^2}_{sc}$ regressable formula $W$ is uniform in $\text{do}(\alpha, S)$ for some ground action $\alpha$ and ground situation $S$, and predicate Poss does not appear in $W$, then $\rho[W]$ is uniform in $S$ and there is still no appearance of Poss.

Again, similar to the proof of Lemma 14, we can prove the following lemma.

**Lemma 15** Consider any given BAT $\mathcal{D}$, the $\mathcal{L}^{C^2}_{sc}$ regression operator $\mathcal{R}^{C^2}$ defined in Section 3.3.1, and any $\mathcal{L}^{C^2}_{sc}$ regressable formula $W$ in $\mathcal{D}$. Then, $\mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\rho[W]]$.

Moreover, according to the definition of $\rho$, it is straightforward to prove the following property of $\rho$. 
**Property 7** Consider any $\mathcal{L}_{sc}^{C^2}$ regressable formula $W$ with a background BAT $\mathcal{D}$, including a finite set $\mathcal{D}_{TBox}$ of acyclic TBox axioms. Assume that there is no appearance of Poss in $W$. Let $m = \text{size}(W)$, $h = \max(2, \text{sizeSSA}(\mathcal{D}))$, $h_0 = |\mathcal{D}_{TBox}|$ (i.e., the size of $\mathcal{D}_{TBox}$) and $h_1 = \max \{ \text{size}(\Phi_G) \mid G(x) \equiv \Phi_G(x) \text{ is a TBox axiom in } \mathcal{D}_{TBox} \}$ if $h_0 \neq 0$, or $h_1 = 0$ otherwise. Notice that $h$, $h_0$ and $h_1$ are fixed when $\mathcal{D}$ is given. Then, it takes no more than $m(h_0 + 1)$ steps to obtain $\rho[W]$, whose size is no more than $c \cdot m$, where $c$ is a constant that equals to $h_1 h_0$.

We also have the following corollary of Lemma 14 and Lemma 15.

**Corollary 6** Consider any given BAT $\mathcal{D}$, the $\mathcal{L}_{sc}^{C^2}$ regression operator $\mathcal{R}^{C^2}$ defined in Section 3.3.1, and any $\mathcal{L}_{sc}^{C^2}$ regressable formula $W$ in $\mathcal{D}$. Then, $\mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\epsilon[\rho[W]]]$.

**Proof:** By Lemma 14, $\mathcal{R}^{C^2}[\epsilon[\rho[W]]] = \mathcal{R}^{C^2}[\rho[W]]$, and by Lemma 15, $\mathcal{R}^{C^2}[\rho[W]] = \mathcal{R}^{C^2}[W]$. Therefore, $\mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\epsilon[\rho[W]]]$. \(\square\)

Now, we provide a detailed proof of Lemma 2 in Section 3.3.4.

**Lemma 2** (Section 3.3.4) Consider a BAT $\mathcal{D}$ in $\mathcal{L}_{sc}^{C^2}$ whose $\mathcal{D}_{ss}$ and $\mathcal{D}_T$ are $\text{ALCO}(U)$-restricted. Let $W$ be any $\mathcal{L}_{sc}^{C^2}$ regressable formula in $\mathcal{D}$ that is uniform in a ground situation $S$ and has no appearance of Poss. Let $n = \text{sitLength}(S)$ and $m = \text{size}(W)$. Then if $W$ with the situation term $S$ suppressed is in $FO_{DL}$, there is a formula $\Phi_W$ in $FO_{DL}$ such that $\mathcal{R}^{C^2}[W]$ is equivalent to $\Phi_W[S_0]$. It takes no more than $c \cdot n \cdot \text{size}(\Phi_W)$ steps of logical transformations as introduced in the proof from $\mathcal{R}^{C^2}[W]$ (with $S_0$ suppressed) to find such $\Phi_W$ for some constant number $c$. Moreover, $\text{size}(\Phi_W)$ is in $O(2^{h_0 m + 3h^2 n^2})$ for some positive integer $h$. That is, the size of $\Phi_W$ is no more than exponential in the size of $W$.

**Proof:** Without loss of generality, we assume that there is no defined concept in $W \in FO_{DL}$. Otherwise, each defined concept will be replaced by its definitions from the TBox axioms with finite steps of $\mathcal{L}_{sc}^{C^2}$ regression, which causes no more than a constant
increase in the size of the original formula, because TBox is fixed (once $\mathcal{D}$ is given), TBox is acyclic, there are only finitely many TBox axioms. The size of the formula on the RHS of each TBox axiom is limited from above by a constant.

We will first prove such formula $\phi_W \in FO_{DL}$ always exists, and then estimate the size of the formula. Note that the proof of Lemma 2 is not obvious, given that the regression operator itself does not generally preserve the syntactic form of a formula on its input.

Now, we define a notation for later convenience. If $W$ is a formula uniform in any situation $s$, we denote the formula with all situation terms suppressed (if any) in $W$ simply as $W[-s]$. Moreover, to simplify the presentation of the proof, below we write $W_1 \equiv W_2$ whenever $|= W_1 \equiv W_2$ for any formulas $W_1$ and $W_2$. We will first prove the following more specific statement (Statement (1)) below with respect to the given BAT $\mathcal{D}$: Consider any ground situation $S$ and a $\mathcal{L}_{sc}^{C^2}$ regressable formula $W$ with the background BAT $\mathcal{D}$, where $W$ is uniform in $S$ and has no occurrences of Poss. If $W[-S]$ is in $FO_{x}^{DL}$ ($FO_{y}^{DL}$, respectively), there is a formula $\varphi$ in $FO_{x}^{DL}$ ($FO_{y}^{DL}$, respectively) such that $R^{C^2}[W]$ is equivalent to $\varphi[S_0]$.

The structure of our proof will consist of two nested proofs by induction, where the internal proof by induction will include an analysis of many sub-cases. The main proof will proceed by induction on the length of $S$, i.e., the number of actions involved in $S$. Inside the inductive step of this proof, we will prove the statement by induction on the structure of a $\mathcal{L}_{sc}^{C^2}$ regressable formula $W$. In the latter, the most time consuming parts will be two cases: when $W$ is a primitive dynamic concept (a fluent with one object argument and one situation argument); or, when $W$ is of the form $\exists y.R(x, y, S) \land W_1(y)[S]$ for some dynamic role $R$ (a fluent with two object arguments and one situation argument) and formula $W_1(y) \in FO_{DL}^{y}$. These two cases are laborious and require an analysis of numerous sub-cases depending on the structure of logical formulas in SSAs.

**Base case of the induction on the length of $S$:**

If $S = S_0$, then let $\varphi = W[-S_0]$, and it is trivial to see that Statement (1) is true.
Inductive step of the induction on the length of $S$:

Now, without loss of generality, we assume that $S = do(\alpha, S_1)$ and Statement (1) is true for any $\mathcal{L}_{sc}^{C^2}$ regressable formula $W'$ that is uniform in $S_1$ and has no appearance of $\text{Poss}$.

We prove Statement (1) for any $\mathcal{L}_{sc}^{C^2}$ regressable formula $W$ that is uniform in $S$ and has no appearance of $\text{Poss}$ by induction on the structure of $W[-S]$.

Since every formula in $FO_{DL}^y$ is a dual formula to a formula in $FO_{DL}^x$, the proof for Statement (1) where $W[-S]$ is in $FO_{DL}^y$ is “dual” to the proof for Statement (1) where $W[-S]$ is in $FO_{DL}^x$, in the sense that we only need to replace every appearance of $x$ with $y$ and $y$ with $x$. Hence, below we will only provide a detailed proof for Statement (1) where $W[-S] \in FO_{DL}^x$, and omit details of the proof of Statement (1) where $W[-S] \in FO_{DL}^y$.

In order to prove Statement (1) for ground situation $S$, we will prove Statement (1) and the following statement (Statement (2)) for $S$ together using the induction proof on the structure of $W$:

For any $\mathcal{L}_{sc}^{C^2}$ regressable formula $W$ that is uniform in $S$ (where $S = do(\alpha, S_1)$) and has no appearance of $\text{Poss}$, if $W[-S]$ is in $FO_{DL}^y$ ($FO_{DL}^x$, respectively), then there is a formula $\varphi$ in $FO_{DL}^y$ ($FO_{DL}^x$, respectively), which can be found in no more than $c \cdot \text{size}(\varphi)$ steps for some constant positive integer $c$. Moreover, $\varphi[S_1]$ is equivalent to $\epsilon[\rho[W]]$, and $\varphi[S_1]$ is $\mathcal{L}_{sc}^{C^2}$ regressable with no appearance of $\text{Poss}$.

Base case of the induction on the structure of $W[-S]$:

First, we consider when $W[-S]$ is in $FO_{DL}^x$ and is atomic. There are in total three cases (a-c) below.

a. $W[-S]$ is either true or false. Then, $\epsilon[\rho[W]][-S_1]$ is still true or false, which is in $FO_{DL}^x$; and, $(\mathcal{R}^{C^2}[W])[-S_0]$ is still true or false, which is in $FO_{DL}^x$. Hence, it is trivial to see that Statement (1) and Statement (2) hold.

b. $W[-S]$ is of the form $x = b$ for some constant $b$. Then, $\epsilon[\rho[W]][-S_1]$ is still $x = b$, which is in $FO_{DL}^x$; and, $(\mathcal{R}^{C^2}[W])[-S_0]$ is still $x = b$, which is in $FO_{DL}^x$. Again, it is trivial to see that Statement (1) and Statement (2) hold.
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

215

c. $W[-S]$ is a monadic predicate. Then, there are two sub-cases:

If $W$ is situation-independent, then $\epsilon[\rho[W]][-S_1] = W = W[-S]$, which is in $FO_{DL}^x$; and,

$(\mathcal{R}^{c_2}[W])[-S_0] = W = W[-S]$, which is in $FO_{DL}^x$. Again, it is trivial to see that Statement (1) and Statement (2) hold.

Otherwise, $W = F(x, S)$ for some fluent $F$. Assume that fluent $F(x, s)$ has an SSA of the form Eq. (3.2), whose context conditions (with situation terms suppressed) are all in $FO_{DL}^x$. Depending on whether the context conditions are in $FO_{DL}^x$ (e.g., cases (1)-(12) in Table B.3) or in $FO_{DL}^y$ (e.g., cases (1’)-(12’) in Table B.3), what variables appear in action functions and/or in the conditions (none, $x$ only, $y$ only, $x$ and $y$), and whether or not the variables are quantified, the SSA of $F$ is

$$F(x, do(a, s)) \equiv \bigvee_{i=1}^{m_+} \phi_i^+(x, a, s) \lor F(x, s) \land \neg(\bigvee_{j=1}^{m_-} \phi_j^-(x, a, s)), \quad (B.1)$$

where each $\phi_i^+(x, a, s)$ ($\phi_j^-(x, a, s)$, respectively) is a formula that has the syntactic form of one of the following cases listed in Table B.3 and all the cases we described in Note 1 below. Recall that we prove this lemma for those SSAs which have $\mathcal{ALCO}(U)$-restricted context formulas only. Notice that in Table B.3, $\psi(x)$ ($\psi(y)$, respectively) is a formula in $FO_{DL}^x$ ($FO_{DL}^y$, respectively) with at most one free variable $x$ ($y$, respectively). In cases (1) and (1’), $A$ represents some constant action function. In cases (2)-(6) and (2’)-(6’), $A$ represents some unary action function name. And, in cases (7)-(12) and (7’)-(12’), $A$ represents some binary action function name. Moreover, in Table B.3, $[\exists y.]$ represents that $\exists y.$ only appears when $\psi(y)$ has a free variable $y$. Note that we need $\exists y.$ to bound the variable $y$ in $\psi(y)$, because otherwise $W$ would not be in $FO_{DL}^x$, but would have both $x$ and $y$ as free variables. To make sure that we have exhausted all the possibilities, the cases we listed could include some duplications. For example, case (6) in fact is the same as case (4) by renaming; case (1’) is in fact the same as case (1), because $[\exists y.]\psi(y)$
is a formula in $FO^r_{DL}$ (according to the definition of $FO^r_{DL}$). We require these restrictions to the format of the SSAs of dynamic roles to make sure that we are able to get a formula that can be still expressed (as an equivalent formula) in $FO_{DL}$ after regression.

<table>
<thead>
<tr>
<th></th>
<th>Formula 1</th>
<th>Formula 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a = A \land \psi(x)[s]$</td>
<td>$a = A \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>2</td>
<td>$a = A(x) \land \psi(x)[s]$</td>
<td>$a = A(x) \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>3</td>
<td>$\exists x.a = A(x) \land \psi(x)[s]$</td>
<td>$\exists x.a = A(x) \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>4</td>
<td>$\exists x(a = A(x)) \land \psi(x)[s]$</td>
<td>$\exists x(a = A(x)) \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>5</td>
<td>$\exists y.a = A(y) \land \psi(x)[s]$</td>
<td>$\exists y.a = A(y) \land \psi(y)[s]$</td>
</tr>
<tr>
<td>6</td>
<td>$\exists y(a = A(y)) \land \psi(x)[s]$</td>
<td>$\exists y(a = A(y)) \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>7</td>
<td>$\exists y.a = A(x, y) \land \psi(x)[s]$</td>
<td>$\exists y.a = A(x, y) \land \psi(y)[s]$</td>
</tr>
<tr>
<td>8</td>
<td>$\exists x, \exists y.a = A(x, y) \land \psi(x)[s]$</td>
<td>$\exists x, \exists y.a = A(x, y) \land \psi(y)[s]$</td>
</tr>
<tr>
<td>9</td>
<td>$\exists x.\exists y(a = A(x, y)) \land \psi(x)[s]$</td>
<td>$\exists x.\exists y(a = A(x, y)) \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>10</td>
<td>$\exists y.\exists x(a = A(x, y)) \land \psi(x)[s]$</td>
<td>$\exists y.\exists x(a = A(x, y)) \land \psi(y)[s]$</td>
</tr>
<tr>
<td>11</td>
<td>$\exists y.\exists x(a = A(x, y)) \land \psi(x)[s]$</td>
<td>$\exists y.\exists x(a = A(x, y)) \land [\exists y.]\psi(y)[s]$</td>
</tr>
<tr>
<td>12</td>
<td>$\exists y(\exists x(a = A(x, y))) \land \psi(x)[s]$</td>
<td>$\exists y(\exists x(a = A(x, y))) \land [\exists y.]\psi(y)[s]$</td>
</tr>
</tbody>
</table>

Table B.3: All possible syntactic forms for $\phi_i^+(x, a, s)$ or $\phi_j^-(x, a, s)$ in Eq. (B.1)

**Note 1** Let $O$, $O_1$ and $O_2$ be some constant objects. There are also cases for $a = A(O)$ (or $a = A(O_1, O_2)$, respectively) in $\phi_i^+$ and/or $\phi_j^-$ in the SSA of $F$, which can be proved similarly to the cases in Table B.3 where $a = A$. There are also cases for $a = A(O_1, x)$ (or $a = (x, O_1)$, respectively) in $\phi_i^+$ and/or $\phi_j^-$ in the SSA of $F$, which can be proved similarly to the cases in Table B.3 where $a = A(x)$. There are also cases for $a = A(O_1, y)$ (or $a = (y, O_1)$, respectively) in $\phi_i^+$ and/or $\phi_j^-$ in the SSA of $F$, which can be proved similarly to the cases in Table B.3 where $a = A(y)$. There are also cases for $a = A(y, x)$ in $\phi_i^+$ and/or $\phi_j^-$ in the SSA of $F$, which can be proved
similarly to the cases in Table B.3 where \( a = A(x, y) \). Moreover, note that using the unique name axioms for object constants, we can simplify the formulas by replacing all (in)equalities between object constants with either \( \text{true} \) or \( \text{false} \) in the resulting formula that is equivalent to \( \epsilon[\rho[W]] \) for any \( L_{sc}^{C2} \) regressable formula \( W \). Moreover, such simplification takes at most a constant number of logical transformation steps with respect to the size of the resulting formula.

We first prove case by case for all possible syntactic forms of \( \phi_i^+(x, \alpha, S_1) \) (\( \phi_j^-(x, \alpha, S_1) \), respectively), that there exist some formulas in \( FO_{DL}^c \) for any \( i \) (\( j \), respectively) such that they are logically equivalent to \( \epsilon[\phi_i^+(x, \alpha, S_1)] \) (\( \epsilon[\phi_j^-(x, \alpha, S_1)] \), respectively) when the situation term \( S_1 \) is restored. Later, it will be convenient to talk these formulas as \( \nu_i^+ \) (or \( \nu_j^- \), respectively). Moreover, finding the equivalent formulas takes a constant number of steps of logical transformations with respect to the size of \( \epsilon[\phi_i^+(x, \alpha, S_1)] \) (\( \epsilon[\phi_j^-(x, \alpha, S_1)] \), respectively) by using unique name axioms for object constants or by using well-known first-order logical equivalent tautologies.

Here is one trivial sub-case: if the function name of \( \alpha \) is not \( A \), then in each of the aforementioned cases (1)-(12), (1’)-(12’) and Note 1, \( \epsilon \) of each formula (\( \phi_i^+ \) or \( \phi_j^- \)) equals \( \text{false} \), which is still in \( FO_{DL}^c \). Hence, below we only discuss the case when \( \alpha \) has the same function name (with the same number of arguments) as the given action function name in each case, and we let \( O, O_1 \) and \( O_2 \) be some constants. Notice that in case (1), since the context condition \( \psi(x) \) is in \( FO_{DL}^c \), \( \psi(x)[S_1] \) does not contain any equality between action terms, hence \( \epsilon[\psi(x)[S_1]] = \psi(x)[S_1] \). The same reasoning will be used in other cases, and detailed explanations are omitted to avoid repetition.

\[(1) \ a = A \land \psi(x)[s] \]

Assume that \( \alpha = A \), then
\[
\epsilon[\alpha = A \land \psi(x)[S_1]] = \epsilon[\alpha = A] \land \epsilon[\psi(x)[S_1]]
\]
\[
= true \land \psi(x)[S_1] \equiv \psi(x)[S_1].
\]

Clearly, \(\psi(x)\) is already in \(FO^x_{DL}\), and it takes no steps of logical transformations to find the equivalent formula.

\(1') \ a = A \land [\exists y.] \psi(y)[s]\)

Because \([\exists y.] \psi(y)\) is in fact in \(FO^x_{DL}\), the proof for \(1'\) is the same as for \(1\).

\(2) \ a = A(x) \land \psi(x)[s]\)

Assume that \(\alpha = A(O)\), then
\[
\epsilon[\alpha = A(x) \land \psi(x)[S_1]] = \epsilon[A(O) = A(x)] \land \psi(x)[S_1]
\]
\[
= (x = O) \land \psi(x)[S_1]
\]
\[
= (x = O \land \psi(x))[S_1].
\]

Clearly, given that \(\psi(x)\) is in \(FO^x_{DL}\), we have that \(x = O \land \psi(x)\) is in \(FO^x_{DL}\), and it takes no steps of logical transformations.

\(2') \ a = A(x) \land [\exists y.] \psi(y)[s]\)

Because \([\exists y.] \psi(y)\) is in fact in \(FO^y_{DL}\), the proof for \(2'\) is the same as for \(2\).

\(3) \ \exists x. a = A(x) \land \psi(x)[s]\)

Assume that \(\alpha = A(O)\), then
\[
\epsilon[\exists x. a = A(x) \land \psi(x)[S_1]] = \exists x. \epsilon[A(O) = A(x) \land \psi(x)[S_1]]
\]
\[
= (\exists x. x = O \land \psi(x))[S_1].
\]

Clearly, the resulting formula \((\exists x. x = O \land \psi(x))\) is in \(FO^y_{DL}\) since any closed sentence is in both \(FO^x_{DL}\) and \(FO^y_{DL}\). It takes no steps of logical transformations.

\(3') \ \exists x. a = A(x) \land [\exists y.] \psi(y)[s]\)

Note that \([\exists y.] \psi(y)\) has no free variable \(x\), hence \(x\) is in fact only quantified over \(a = A(x)\), hence case \(3'\) is equivalent to case \(4'\) below.

\(4) \ \exists x (a = A(x)) \land \psi(x)[s]\)

Assume that \(\alpha = A(O)\), then
\[ \\
e[\exists x(\alpha = A(x)) \land \psi(x)[S_1]] = e[\exists x(A(O) = A(x)) \land \psi(x)[S_1]] \\
= (\exists x(x = O) \land \psi(x))[S_1].  \\
\]
Because \( \exists x(x = O) \) and \( \psi(x) \) are in \( FO^r_{DL} \), \( \exists x(x = O) \land \psi(x) \) is in \( FO^r_{DL} \). It takes no steps of logical transformations.

\((4')\) \( \exists x(a = A(x)) \land [\exists y.]\psi(y)[s] \)
Because \([\exists y.]\psi(y)\) is in fact in \( FO^r_{DL} \), case \((4')\) is a special case of case \((2)\).

\((5)\) \( \exists y.a = A(y) \land \psi(x)[s] \)
Assume that \( \alpha = A(O) \), then
\[
\epsilon[\exists y.\alpha = A(y) \land \psi(x)[S_1]] = \epsilon[\exists y.\alpha = A(y) \land \psi(x)[S_1]] \\
= \exists y(y = O) \land \psi(x)[S_1] \\
= (\exists y(y = O) \land \psi(x))[S_1].  \\
\]
Clearly, the closed formula \( \exists y(y = O) \land \psi(x) \) is in \( FO^r_{DL} \). It takes one step of logical transformations to minimize the quantification scope of \( \exists y \). Note that although the resulting formula can be simplified to \( \psi(O)[S_1] \), but our regression operator and the operator \( \epsilon \) does not do any simplification.

\((5')\) \( \exists y.a = A(y) \land \psi(y)[s] \)
Assume that \( \alpha = A(O) \), then
\[
\epsilon[\exists y.\alpha = A(y) \land \psi(y)[S_1]] = \epsilon[\exists y.\alpha = A(y) \land \psi(y)[S_1]] \\
= \exists y.a = A(y) \land \psi(y)[S_1] \\
= \exists y.y = O \land \psi(y)[S_1] \\
= (\exists y.y = O \land \psi(y))[S_1].  \\
\psi(y) \) is in \( FO^y_{DL} \), hence \( y = O \land \psi(y) \) is in \( FO^y_{DL} \) and \( \exists y.y = O \land \psi(y) \) is in \( FO^r_{DL} \). It takes no steps of logical transformations.

\((6)\) \( \exists y(a = A(y)) \land \psi(x)[s] \)
Case \((6)\) is equivalent to case \((5)\), because in case \((5)\) the quantification range of \( y \) is in fact only over \( a = A(y) \). Hence, the statement is true for case \((6)\) by the definition of \( FO^r_{DL} \).
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

(6') \( \exists y (a = A(y)) \land [\exists y.] \psi(y)[s] \) Because [\exists y.] \psi(y) is in \( FO^r_{DL} \), case (6') is a special case of (6).

(7) \( \exists y. a = A(x, y) \land \psi(x)[s] \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y. \alpha = A(x, y) \land \psi(y)[S_1]] \\
= \epsilon[\exists y. A(O_1, O_2) = A(x, y) \land \psi(x)[S_1]] \\
= (\exists y. x = O_1 \land y = O_2 \land \psi(x))[S_1] \\
\equiv (x = O_1 \land \exists y(y = O_2 \land \psi(x)))[S_1].
\]

Clearly, given that \( \psi(x) \) is in \( FO^r_{DL} \), we have that \( x = O_1 \land \exists y(y = O_2 \land \psi(x)) \) is in \( FO^r_{DL} \) by the definition of \( FO^r_{DL} \). It takes one step of logical transformations to minimize the quantification scope of \( \exists y \).

(7') \( \exists y. a = A(x, y) \land \psi(y)[s] \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y. \alpha = A(x, y) \land \psi(y)[S_1]] \\
= (\exists y. x = O_1 \land y = O_2 \land \psi(y))[S_1] \\
\equiv (x = O_1 \land (\exists y.y = O_2 \land \psi(y)))[S_1].
\]

Clearly, given that \( \psi(y) \) is in \( FO^r_{DL} \), we have that \( x = O_1 \land (\exists y.y = O_2 \land \psi(y)) \) is in \( FO^r_{DL} \) by the definition of \( FO^r_{DL} \). It takes one step of logical transformations to minimize the quantification scope of \( \exists y \).

(8) \( \exists y(a = A(x, y)) \land \psi(x)[s] \)

Case (8) is equivalent to case (7), because in case (7) the quantification range of \( y \) is in fact only over \( a = A(x, y) \).

(8') \( \exists y(a = A(x, y)) \land [\exists y.] \psi(y)[s] \)

Because [\exists y.] \psi(y) is in \( FO^r_{DL} \), case (8') is a special case of case (8).

(9) \( \exists x. \exists y. a = A(x, y) \land \psi(x)[s] \)

Case (9) is equivalent to case (10), because in case (9) the quantification range of \( y \) is in fact only over \( a = A(x, y) \).

(9') \( \exists x. \exists y. a = A(x, y) \land \psi(y)[s] \)
Case (9') is equivalent to case (11'), because in case (9') the quantification range of \( x \) is in fact only over \( a = A(x, y) \).

(10) \( \exists x. \exists y(a = A(x, y)) \land \psi(x)[s] \)

Assume that \( \alpha = A(O_1, O_2) \), then

\[
\epsilon[\exists x. \exists y(a = A(x, y)) \land \psi(x)[S_1]] = \epsilon[\exists x. \exists y(A(O_1, O_2) = A(x, y)) \land \psi(x)[S_1]] \\
= (\exists x. \exists y(x = O_1 \land y = O_2) \land \psi(x))[S_1] \\
\equiv (\exists x. x = O_1 \land \exists y(y = O_2) \land \psi(x))[S_1].
\]

It is easy to see that the resulting formula is in \( FO_{DL}^x \), and it takes no more than two steps of logical transformations to do the transformation.

(10') \( \exists x. \exists y(a = A(x, y)) \land [\exists y.]\psi(y)[s] \)

Because \( [\exists y.]\psi(y) \) is in \( FO_{DL}^x \), case (10') is a special case of case (10).

(11) \( \exists y. \exists x(a = A(x, y)) \land \psi(x)[s] \)

Case (11) is equivalent to case (12), because in case (11) the quantification range of \( y \) is in fact only over \( a = A(x, y) \).

(11') \( \exists y. \exists x(a = A(x, y)) \land \psi(y)[s] \)

Assume that \( \alpha = A(O_1, O_2) \), then

\[
\epsilon[\exists y. \exists x(a = A(x, y)) \land \psi(y)[S_1]] = \epsilon[\exists y. \exists x(A(O_1, O_2) = A(x, y)) \land \psi(y)[S_1]] \\
= \exists y. \exists x(x = O_1 \land y = O_2) \land \psi(y)[S_1] \\
\equiv (\exists y. \exists x(x = O_1) \land y = O_2 \land \psi(y))[S_1] \\
\equiv \exists x(x = O_1) \land \exists y(y = O_2) \land \psi(y))[S_1].
\]

It is easy to see that \( \exists y. \exists x(x = O_1) \land y = O_2 \land \psi(y) \) is in \( FO_{DL}^x \), and it takes one step to minimize the scope of \( \exists x \) and another one step to minimize the scope of \( \exists y \).

(12) \( \exists y(\exists x(a = A(x, y))) \land \psi(x)[s] \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y(\exists x(\alpha = A(x, y))) \land \psi(x)[S_1]] = \epsilon[\exists y(\exists x(A(O_1, O_2) = A(x, y)) \land \psi(x)[S_1]] \\
= \exists y(\exists x(x = O_1 \land y = O_2)) \land \psi(x)[S_1] \\
\equiv (\exists y(\exists x(x = O_1) \land \psi(x)))[S_1] \\
\equiv (\exists x(x = O_1) \land \exists y(y = O_2) \land \psi(x))[S_1].
\]

It is easy to see that \(\exists y(\exists x(x = O_1) \land y = O_2) \land \psi(x)\) is in \(FO_{DL}^x\), and it takes one step of logical transformations to minimize the scope of \(\exists x\) and another one step to minimize the scope of \(\exists y\).

(12') \(\exists y(\exists x(a = A(x, y))) \land [\exists y.]\psi(y)[s]\)

Because \([\exists y.]\psi(y)\) is in \(FO_{DL}^x\), case (12') is a special case of case (12).

When \(a\) is substituted by a ground action \(\alpha\) and \(s\) is substituted by a ground situation \(S_1\), by the definition of regression using the SSA of the form (B.1), \(\epsilon\) and \(\rho\),

\[
\epsilon[\rho[F(x, S)]] = \epsilon[\rho[F(x, do(\alpha, S_1))]] \\
= \epsilon \left[ \bigvee_{i=1}^{m_+} \phi_i^+(x, \alpha, S_1) \lor F(x, S_1) \land \neg \left( \bigvee_{j=1}^{m_-} \phi_j^-(x, \alpha, S_1) \right) \right] \\
\quad \text{(by the definition of } \rho \text{ and (B.1)),} \\
= \bigvee_{i=1}^{m_+} \epsilon[\phi_i^+(x, \alpha, S_1)] \lor F(x, S_1) \land \neg \left( \bigvee_{j=1}^{m_-} \epsilon[\phi_j^-(x, \alpha, S_1)] \right) \\
\quad \text{(by the definition of } \epsilon\), \\
= \bigvee_{i=1}^{m_+} \nu_i^+(x)[S_1] \lor F(x, S_1) \land \neg \left( \bigvee_{j=1}^{m_-} \nu_j^-(x)[S_1] \right) \\
\quad \text{(according to the proof of cases (1)-(12),(1')-(12')),} \\
= \left( \bigvee_{i=1}^{m_+} \nu_i^+(x) \lor F(x) \land \neg \left( \bigvee_{j=1}^{m_-} \nu_j^-(x) \right) \right)[S_1], \tag{B.2}
\]

such that each \(\nu_i^+(x)\) (\(\nu_j^-(x)\), respectively) is a formula in \(FO_{DL}^x\), which has at most one free variable \(x\). Each \(\nu_i^+(x)\) (\(\nu_j^-(x)\), respectively) is logically equivalent to \(\epsilon[\phi_i^+(x, \alpha, S_1)][-S_1]\) (\(\epsilon[\phi_j^-(x, \alpha, S_1)][-S_1]\), respectively). Clearly, the formula on the RHS of Eq. (B.2) is regressable, uniform in \(S_1\), and is in \(FO_{DL}^x\) (with \(S_1\) suppressed) according to the definition of the set \(FO_{DL}^x\). Moreover, it takes only a constant number of steps with respect to the size of the resulting formula to find the equivalent formula. Then, using the induction hypothesis for situation \(S_1\) and Corollary 6 (i.e., \(\mathcal{R}C^2[F(x, S)] = \mathcal{R}C^2[\epsilon[\rho[F(x, S)]]]\)), we
have that \( \left( R^{C^2}[F(x, do(\alpha, S_1))] \right) [-S_0] \) is still be equivalent to some formula in \( FO^y_{DL} \).

Similarly, we can prove Statements (1) and (2) for the case when \( W[-S] \) is in \( FO^y_{DL} \) and is atomic.

**Inductive step of the induction on the structure of \( W[-S] \):**

Now, we complete our remaining cases when \( W \) (hence, \( W[-S] \)) is not atomic. Recall that \( S = do(\alpha, S_1) \). There are totally four cases (a-d).

a. \( W[-S] \) is of the form \( \neg W_1 \), where \( W_1 \in FO^y_{DL} \).

It is obvious that \( W = \neg W_1[S] \), and \( \epsilon[\rho[W]] = \neg \epsilon[\rho[W_1[S]]] \). Moreover, by the induction hypothesis on the structure of \( W \), there is a formula \( \phi_1 \in FO^y_{DL} \) such that \( \epsilon[\rho[W_1[S]]] \equiv \phi_1[S_1] \), which is regresable, uniform in \( S_1 \), has no appearance of \( Poss \) and can be found \( c \cdot size(\phi_1) \) for some integer \( c \). Hence, \( \epsilon[\rho[W]] \equiv \neg \phi_1[S_1] \), and Statement (2) is true for \( W \). Then, according to Corollary 6, \( R^{C^2}[W] = R^{C^2}[\epsilon[\rho[W]]] \equiv R^{C^2}[\neg \phi_1[S_1]] \). Next, by the induction hypothesis on \( S_1 \), \( R^{C^2}[W][-S_0] \) is equivalent to some formula in \( FO^y_{DL} \), and it is easy to see that Statement (1) is true for \( W \) that is uniform in situation \( S \).

b. \( W[-S] \) is of the form \( W_1 \wedge W_2 \) or of the form \( W_1 \vee W_2 \), where \( W_1, W_2 \in FO^y_{DL} \).

Then, if \( W[-S] \) is of the form \( W_1 \wedge W_2 \), it is obvious that \( W = (W_1 \wedge W_2)[S] \), so \( \epsilon[\rho[W]] = \epsilon[\rho[W_1[S]]] \wedge \epsilon[\rho[W_2[S]]] \). By the induction hypothesis on the structure of \( W \), there are formulas \( \phi_1, \phi_2 \in FO^y_{DL} \) such that \( \epsilon[\rho[W_1[S]]] \equiv \phi_1[S_1] \) and \( \epsilon[\rho[W_2[S]]] \equiv \phi_2[S_1] \), which are regresable, uniform in \( S_1 \), has no appearance of \( Poss \) and can be found in \( c_1 \cdot size(\phi_1) \) and \( c_2 \cdot size(\phi_2) \) steps for some positive constants \( c_1 \) and \( c_2 \) respectively. Hence, \( \epsilon[\rho[W]] \equiv (\phi_1 \wedge \phi_2)[S_1] \), which can be found in \( c \cdot (size(\phi_1) + size(\phi_2) + 1) \) steps for some integer \( c = max(c_1, c_2) \), and Statement (2) is true for \( W \). Then, according to Corollary 6, \( R^{C^2}[W] = R^{C^2}[\epsilon[\rho[W]]] \equiv R^{C^2}[(\phi_1 \wedge \phi_2)[S_1]] \). Next, by the induction hypothesis on \( S_1 \), \( R^{C^2}[W][-S_0] \) is equivalent to some formula in \( FO^y_{DL} \), and it is easy to see that Statement (1) is true for
W that is uniform in S.

It is very similar to prove that Statements (1) and (2) are true when W[−S] is of the form W₁ ∨ W₂, and details are omitted here.

c. W[−S] is of the form [∃y.]W₁(y) or [∀y.]W₁(y), where W₁(y) is in FO₆DL.

Then, if W[−S] is of the form [∃y.]W₁(y), ε[ρ[W]] = [∃y.].ε[ρ[W₁(y)[S]]]. By the induction hypothesis on the structure of W, there is a formula φ₁(y) ∈ FO₆DL such that ε[ρ[W₁(y)[S]]] ≡ φ₁(y)[S₁], which is regresssable, uniform in S₁, has no appearance of Poss and can be found in c · size(φ₁(y)) steps for some integer c. Hence, ε[ρ[W]] ≡ ([∃y.]φ₁(y))[S₁], and Statement (2) is true for W. Then, according to Corollary 6, Rₑᶜ[W] = Rₑᶜ[ε[ρ[W]]] ≡ Rₑᶜ[([∃y.]φ₁(y))[S₁]]. Next, by the induction hypothesis on S₁, Rₑᶜ[W][−S₀] is equivalent to some formula in FOᵥDL, and it is easy to see that Statement (1) is true for W that is uniform in situation S.

It is very similar to prove that Statements (1) and (2) are true when W[−S] is of the form [∀y.]W₁(y), and details are omitted here.

d. W[−S] is of the form ∃y.R(x, y) ∧ W₁(y) or ∀y.R(x, y) ⊃ W₁(y), where R(x, y) is a dynamic predicate and W₁(y) is in FO₆DL.

We first consider the case when W[−S] is of the form ∃y.R(x, y) ∧ W₁(y). Then, W = ∃y.R(x, y)[S] ∧ W₁(y)[S], and there are two sub-cases:

(d.1) If R is situation-independent, then ε[ρ[W]] = [∃y.]R(x, y) ∧ ε[ρ[W₁(y)[S]]]. By the induction hypothesis on the structure of W₁, there is a formula φ₁(y) ∈ FO₆DL such that ε[ρ[W₁(y)[S]]] ≡ φ₁(y)[S₁], which is regresssable, uniform in S₁, has no appearance of Poss and can be found in c · size(φ₁(y)) steps for some integer c. Hence, ε[ρ[W]] ≡ ([∃y.]R(x, y) ∧ φ₁(y))[S₁], and Statement (2) is true for W. Then, according to Corollary 6, Rₑᶜ[W] = Rₑᶜ[ε[ρ[W]]] ≡ Rₑᶜ[([∃y.]R(x, y) ∧ φ₁(y))[S₁]]. Next, by the induction hypothesis on S₁, Rₑᶜ[W][−S₀] is equivalent to some for-
mula in $FO^{x}_{DL}$, and it is easy to see that Statement (1) is true for $W$ that is uniform in situation $S$.

(d.2) Otherwise, if $R$ is a fluent, then $\epsilon[\rho[W]] = [\exists y.]\epsilon[\rho[R(x, y, S)]] \land \epsilon[\rho[W_1(y)[S]]]$.

Moreover, we need to consider different sub-cases for the SSA of $R$. Notice that according to the definition of a $D_{ss}$ that is $\mathcal{ALCO}(U)$-restricted, all dynamic roles are both $\mathcal{ALCO}(U)$-restricted and context-free. So, depending on whether the context conditions are in $FO^{x}_{DL}$, the cases (1)-(16) in Table B.4, or $FO^{y}_{DL}$, the cases (1')-(16') in Table B.4, what variables appear in action functions and/or in the conditions (none, $x$ only, $y$ only, $x$ and $y$), and whether or not the variables are quantified, the SSA of $R$ is

$$R(x, y, do(a, s)) \equiv \bigvee_{i=1}^{m_{+}} \phi_{i}^{+}(x, y, a) \lor R(x, y, s) \land \neg \bigvee_{j=1}^{m_{-}} \phi_{j}^{-}(x, y, a),$$  \hspace{1cm} (B.3)

where each $\phi_{i}^{+}(x, y, a)$ ($\phi_{j}^{-}(x, y, a)$, respectively) is a situation-independent formula whose syntactic form is one of the following cases listed in Table B.4 and the cases we described in Note 2. Recall that we prove the lemma for those SSAs which have $\mathcal{ALCO}(U)$-restricted context formulas only. Notice that in Table B.4, $\psi(x)$ ($\psi(y)$, respectively) is a formula in $FO^{x}_{DL}$ ($FO^{y}_{DL}$, respectively) with at most one free variable $x$ ($y$, respectively). In the cases (1) and (1'), $A$ represents some constant action function. In the cases (2)-(7) and (2')-(7') , $A$ represents some unary action function name. And, in the cases (8)-(16) and (8')-(16'), $A$ represents some binary action function name. Again, to assure that we have exhausted all the possibilities, the cases we listed in Table B.4 exhaust all possible combinations of actions, quantifiers and the format of the additional condition $\psi$, and may include some other duplications. For example, in fact, the case (6) is the same as the case (3) by renaming.

**Note 2** For any formula $\psi_{1}(x)$ ($\psi_{1}(y)$, respectively) in $FO^{x}_{DL}$ ($FO^{y}_{DL}$, respectively), there are also cases where the context conditions are either of the form
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

Table B.4: All possible syntactic forms for $\phi_i^\uparrow(x, y, a)$ or $\phi_j^\uparrow(x, y, a)$ in Eq. (B.3)

<table>
<thead>
<tr>
<th></th>
<th>$a = A \land \psi(x)$</th>
<th>1'</th>
<th>$a = A \land \psi(y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$a = A(x) \land \psi(x)$</td>
<td>2'</td>
<td>$a = A(x) \land \psi(y)$</td>
</tr>
<tr>
<td>3</td>
<td>$\exists x (a = A(x)) \land \psi(x)$</td>
<td>3'</td>
<td>$\exists x (a = A(x)) \land \psi(y)$</td>
</tr>
<tr>
<td>4</td>
<td>$\exists x. a = A(x) \land \psi(x)$</td>
<td>4'</td>
<td>$\exists x. a = A(x) \land \psi(y)$</td>
</tr>
<tr>
<td>5</td>
<td>$a = A(y) \land \psi(x)$</td>
<td>5'</td>
<td>$a = A(y) \land \psi(y)$</td>
</tr>
<tr>
<td>6</td>
<td>$\exists y (a = A(y)) \land \psi(x)$</td>
<td>6'</td>
<td>$\exists y (a = A(y)) \land \psi(y)$</td>
</tr>
<tr>
<td>7</td>
<td>$\exists y. a = A(y) \land \psi(x)$</td>
<td>7'</td>
<td>$\exists y. a = A(y) \land \psi(y)$</td>
</tr>
<tr>
<td>8</td>
<td>$a = A(x, y) \land \psi(x)$</td>
<td>8'</td>
<td>$a = A(x, y) \land \psi(y)$</td>
</tr>
<tr>
<td>9</td>
<td>$\exists x (a = A(x, y)) \land \psi(x)$</td>
<td>9'</td>
<td>$\exists x (a = A(x, y)) \land \psi(y)$</td>
</tr>
<tr>
<td>10</td>
<td>$\exists x. a = A(x, y) \land \psi(x)$</td>
<td>10'</td>
<td>$\exists x. a = A(x, y) \land \psi(y)$</td>
</tr>
<tr>
<td>11</td>
<td>$\exists y (a = A(x, y)) \land \psi(x)$</td>
<td>11'</td>
<td>$\exists y (a = A(x, y)) \land \psi(y)$</td>
</tr>
<tr>
<td>12</td>
<td>$\exists y. a = A(x, y) \land \psi(x)$</td>
<td>12'</td>
<td>$\exists y. a = A(x, y) \land \psi(y)$</td>
</tr>
<tr>
<td>13</td>
<td>$\exists y (\exists x (a = A(x, y))) \land \psi(x)$</td>
<td>13'</td>
<td>$\exists y (\exists x (a = A(x, y))) \land \psi(y)$</td>
</tr>
<tr>
<td>14</td>
<td>$\exists y. \exists x (a = A(x, y)) \land \psi(x)$</td>
<td>14'</td>
<td>$\exists y. \exists x (a = A(x, y)) \land \psi(y)$</td>
</tr>
<tr>
<td>15</td>
<td>$\exists x. \exists y (a = A(x, y)) \land \psi(x)$</td>
<td>15'</td>
<td>$\exists x. \exists y (a = A(x, y)) \land \psi(y)$</td>
</tr>
<tr>
<td>16</td>
<td>$\exists x. \exists y. a = A(x, y) \land \psi(x)$</td>
<td>16'</td>
<td>$\exists x. \exists y. a = A(x, y) \land \psi(y)$</td>
</tr>
</tbody>
</table>

There are also cases for $a = A(O)$ (or $a = A(O_1, O_2)$, respectively) in $\phi_i^\uparrow$ and/or $\phi_j^\uparrow$ in the SSA of $R$, which can be proved similarly to the cases in Table B.4 where $a = A$. There are also cases for $a = A(O_1, x)$ (or $a = (x, O_1)$, respectively) in $\phi_i^\uparrow$.
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

and/or $\phi^-_j$ in the SSA of $R$, which can be proved similarly to the cases in Table B.4 where $a = A(x)$. There are also cases for $a = A(O_1, y)$ (or $a = (y, O_1)$, respectively) in $\phi^+_i$ and/or $\phi^-_j$ in the SSA of $R$, which can be proved similarly to the cases in Table B.4 where $a = A(y)$. There are also cases for $a = A(y, x)$ in $\phi^+_i$ and/or $\phi^-_j$ in the SSA of $R$, which can be proved similarly to the cases in Table B.4 where $a = A(x, y)$. Notice that using unique name axioms for object constants, we can replace all (in)equalities between object constants with either true or false in the resulting formula that is equivalent to $\epsilon[\rho[W]]$ for any $L_{\text{sc}}^2$ regressable formula $W$. Moreover, such logical transformations take at most linear number of steps with respect to the size of the resulting formula.

We first show that for any case in Table B.4, $\epsilon[\phi^+_i(x, y, \alpha)] \ (\epsilon[\phi^-_j(x, y, \alpha)])$, respectively) results in a formula that is equivalent to a conjunction of one formula in $FO^\alpha_{DL}$ and another formula in $FO^\gamma_{DL}$, i.e., $(\nu(x) \land \eta(y))$ for some $\nu(x) \in FO^x_{DL}$ and $\eta(y) \in FO^y_{DL}$. In particular, we will see in the proof below for all cases in Table B.4, any resulting formula is in one of the four specific forms of $\nu(x) \land \eta(y)$: $\nu(x)$ (when $\eta(y)$ is true), $\eta(y)$ (when $\nu(x)$ is true), $\nu(x) \land y = O$ for some constant $O$, or $x = O \land \eta(y)$ for some constant $O$. We introduce these new notations $\nu(x)$ and $\eta(y)$ to clearly show that for each context condition, its regression is expressible as a conjunction of formulas with free variables $x$ and $y$ separated if there are any of them: $\nu(x)$ has no free occurrences of $y$ and $\eta(y)$ has no free occurrences of $x$.

From now on, without particular emphasis, all the cases we discuss below are the cases in Table B.4. Moreover, for similar proofs, some details are skipped. Again, note that the operators $\rho$ and $\epsilon$ do not perform logical simplifications.

Here is one trivial sub-case: if the function name of $\alpha$ is not $A$, then in each of the aforementioned cases (1)-(16), (1')-(16') and in Note 2, $\epsilon$ of each formula ($\phi^+_i$ or $\phi^-_j$) equals false, which is still in $FO^\alpha_{DL}$. Hence, below we only discuss the condition
that $\alpha$ has the same function name (with the same number of arguments) as the
given action function name in each case, and we let $O$, $O_1$ and $O_2$ be some constants.

(1) $a = A \land \psi(x)$

Assume that $\alpha = A$, then
\[
\epsilon[\alpha = A \land \psi(x)] = true \land \psi(x).
\]
Clearly, $true \land \psi(x)$ is of the form $\nu(x) \land \eta(y)$: let $\nu(x)$ be $\psi(y)$ and let $\eta(y)$ be $true$.

(1') $a = A \land \psi(y)$

Assume that $\alpha = A$, then
\[
\epsilon[\alpha = A \land \psi(y)] = true \land \psi(y).
\]
Clearly, $true \land \psi(y)$ is of the form $\nu(x) \land \eta(y)$: let $\nu(x)$ be $true$ and let $\eta(y)$ be $\psi(y)$.

(2) $a = A(x) \land \psi(x)$

Assume that $\alpha = A(O)$, then
\[
\epsilon[\alpha = A(x) \land \psi(x)] = \epsilon[A(O) = A(x)] \land \psi(x)
= (x=O) \land \psi(x).
\]
Clearly, $x = O \land \psi(x)$ is in $FO_{DL}$ and is of the form $\nu(x) \land \eta(y)$: let $\nu(x)$ be $x=O \land \psi(x)$ and let $\eta(y)$ be $true$.

(2') $a = A(x) \land \psi(y)$

Assume that $\alpha = A(O)$, then
\[
\epsilon[\alpha = A(x) \land \psi(y)] = \epsilon[A(O) = A(x)] \land \psi(y)
= (x=O) \land \psi(y).
\]
Clearly, $x=O \land \psi(y)$ is of the form $\nu(x) \land \eta(y)$: let $\nu(x)$ be $x=O$ and let $\eta(y)$ be $\psi(y)$.

(3) $\exists x (a = A(x)) \land \psi(x)$

Assume that $\alpha = A(O)$, then
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

229

\[ \epsilon[\exists x (\alpha = A(x)) \land \psi(x)] = \epsilon[\exists x (A(O) = A(x))] \land \epsilon[\psi(x)] \]

\[ = \exists x (x = O) \land \psi(x). \]

Clearly, \( \exists x (x = O) \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \nu(x) \) be \( \psi(x) \) and let \( \eta(y) \) be \( \exists x (x = O) \).

(3') \( \exists x (a = A(x)) \land \psi(y) \)

Assume that \( \alpha = A(O) \), then

\[ \epsilon[\exists x (a = A(x)) \land \psi(y)] = \epsilon[\exists x (A(O) = A(x))] \land \epsilon[\psi(y)] \]

\[ = \exists x (x = O) \land \psi(y). \]

Clearly, \( \exists x (x = O) \land \psi(y) \) is of the form \( \nu(x) \land \eta(y) \): let \( \nu(x) \) be \( \psi(x) \) and let \( \eta(y) \) be \( \exists x (x = O) \land \psi(y) \).

(4) \( \exists x.a = A(x) \land \psi(x) \)

Assume that \( \alpha = A(O) \), then

\[ \epsilon[\exists x.a = A(x) \land \psi(x)] = \exists x.\epsilon[A(O) = A(x)] \land \epsilon[\psi(x)] \]

\[ = \exists x.x = O \land \psi(x). \]

Clearly, \( \exists x.x = O \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \nu(x) \) be \( \psi(x) \) and let \( \eta(y) \) be \( \exists x.x = O \land \psi(x) \).

(4') \( \exists x.a = A(x) \land \psi(y) \)

Case (4') is equivalent to case (3'), because in case (4') the quantification range of \( x \) is in fact only over \( a = A(x) \).

(5) \( a = A(y) \land \psi(x) \) (The proof is similar to that of case (2') above.)

Assume that \( \alpha = A(O) \), then

\[ \epsilon[\alpha = A(y) \land \psi(x)] = y = O \land \psi(x). \]

Clearly, \( y = O \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \nu(x) \) be \( \psi(x) \) and let \( \eta(y) \) be \( y = O \).

(5') \( a = A(y) \land \psi(y) \) (The proof is similar to that of case (2) above.)

Assume that \( \alpha = A(O) \), then

\[ \epsilon[\alpha = A(y) \land \psi(y)] = y = O \land \psi(y). \]
Clearly, \( y = O \land \psi(y) \) is in \( FO_{DL}^y \) and is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be \( y = O \land \psi(y) \) and let \( \nu(x) \) be \( true \).

(6) \( \exists y(a = A(y)) \land \psi(x) \) (The proof is similar to that of case (3') above.)

Assume that \( \alpha = A(O) \), then
\[
\epsilon[\exists y(a = A(y)) \land \psi(x)] = \exists y(y = O) \land \psi(x).
\]
Clearly, \( \exists y(y = O) \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be \( y = O \land \psi(y) \) and let \( \nu(x) \) be \( \exists y(y = O) \land \psi(x) \).

(6') \( \exists y(a = A(y)) \land \psi(y) \) (The proof is similar to that of case (3) above.)

Assume that \( \alpha = A(O) \), then
\[
\epsilon[\exists y(a = A(y)) \land \psi(y)] = \exists y(y = O) \land \psi(y).
\]
Clearly, \( \exists y(y = O) \land \psi(y) \) is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be \( y = O \land \psi(y) \) and let \( \nu(x) \) be \( \exists y(y = O) \land \psi(y) \).

(7) \( \exists y. a = A(y) \land \psi(x) \) (The proof is similar to that of case (3) above.)

Case (7) is equivalent to case (6), because in case (7) the quantification range of \( y \) is in fact only over \( y = A(y) \).

(7') \( \exists y. a = A(y) \land \psi(y) \) (The proof is similar to that of case (4) above.)

Assume that \( \alpha = A(O) \), then
\[
\epsilon[\exists y. \alpha = A(y) \land \psi(y)] = \exists y. y = O \land \psi(y).
\]
Clearly, \( \exists y. y = O \land \psi(y) \) is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be \( y = O \land \psi(y) \) and let \( \nu(x) \) be \( \exists y. y = O \land \psi(y) \).

(8) \( a = A(x, y) \land \psi(x) \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\alpha = A(x, y) \land \psi(x)] = \epsilon[A(O_1, O_2) = A(x, y) \land \psi(x)]
\]
\[
= y = O_2 \land x = O_1 \land \psi(x).
\]
Clearly, \( y = O_2 \land x = O_1 \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be \( y = O_2 \) and let \( \nu(x) \) be \( x = O_1 \land \psi(x) \).

(8') \( a = A(x, y) \land \psi(y) \)
Assume that $\alpha = A(O_1, O_2)$, then
\[\epsilon[\alpha = A(x, y) \land \psi(y)] = \epsilon[A(O_1, O_2) = A(x, y) \land \psi(y)] = x = O_1 \land y = O_2 \land \psi(y).\]
Clearly, $x = O_1 \land y = O_2 \land \psi(y)$ is of the form $\nu(x) \land \eta(y)$: let $\eta(y)$ be $y = O_2 \land \psi(y)$ and let $\nu(x)$ be $x = O_1$.

(9) $\exists x(a = A(x, y)) \land \psi(x)$

Assume that $\alpha = A(O_1, O_2)$, then
\[\epsilon[\exists x(\alpha = A(x, y)) \land \psi(x)] = \epsilon[\exists x(A(O_1, O_2) = A(x, y)) \land \psi(x)] = \exists x(x = O_1 \land y = O_2) \land \psi(x) = y = O_2 \land \exists x(x = O_1) \land \psi(x).\]
Clearly, $y = O_2 \land \exists x(x = O_1) \land \psi(x)$ is of the form $\nu(x) \land \eta(y)$: let $\eta(y)$ be $y = O_2$ and let $\nu(x)$ be $\exists x(x = O_1) \land \psi(x)$.

(9') $\exists x(a = A(x, y)) \land \psi(y)$

Assume that $\alpha = A(O_1, O_2)$, then
\[\epsilon[\exists x(\alpha = A(x, y)) \land \psi(y)] = (\exists x(x = O_1 \land y = O_2) \land \psi(y)) = \exists x(x = O_1) \land y = O_2 \land \psi(y).\]
Clearly, $\exists x(x = O_1) \land y = O_2 \land \psi(y)$ is of the form $\nu(x) \land \eta(y)$: let $\eta(y)$ be $y = O_2 \land \psi(y)$ and let $\nu(x)$ be $\exists x(x = O_1)$.

(10) $\exists x.a = A(x, y) \land \psi(x)$

Assume that $\alpha = A(O_1, O_2)$, then
\[\epsilon[\exists x(\alpha = A(x, y)) \land \psi(x)] = (\exists x.x = O_1 \land y = O_2 \land \psi(x)) = y = O_2 \land \exists x.x = O_1 \land \psi(x).\]
Clearly, $y = O_2 \land \exists x.x = O_1 \land \psi(x)$ is of the form $\nu(x) \land \eta(y)$: let $\nu(x)$ be true and let $\eta(y)$ be $y = O_2 \land \exists x.x = O_1 \land \psi(x)$.

(10') $\exists x.a = A(x, y) \land \psi(y)$

Case (10') is equivalent to case (9'), because in case (10') the quantification range of $x$ is in fact only over $a = A(x, y)$.

(11) $\exists y(a = A(x, y)) \land \psi(x)$
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y(\alpha = A(x, y)) \land \psi(x)] = (\exists y(x = O_1 \land y = O_2) \land \psi(x))
\]
\[
\equiv \exists y(y = O_2) \land x = O_1 \land \psi(x).
\]

Clearly, \( \exists y(y = O_2) \land x = O_1 \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be true
and let \( \nu(x) \) be \( \exists y(y = O_2) \land x = O_1 \land \psi(x) \).

(11') \( \exists y(a = A(x, y)) \land \psi(y) \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y(\alpha = A(x, y)) \land \psi(y)] = (\exists y(x = O_1 \land y = O_2) \land \psi(y))
\]
\[
\equiv \exists y(y = O_2) \land x = O_1 \land \psi(y).
\]

Clearly, \( \exists y(y = O_2) \land x = O_1 \land \psi(y) \) is of the form \( \nu(x) \land \eta(y) \): let \( \eta(y) \) be \( \psi(y) \)
and let \( \nu(x) \) be \( \exists y(y = O_2) \land x = O_1 \).

(12) \( \exists y.a = A(x, y) \land \psi(x) \)

Case (12) is equivalent to case (11), because in case (12) the quantification range
of \( y \) is in fact only over \( a = A(x, y) \).

(12') \( \exists y.a = A(x, y) \land \psi(y) \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y.a = A(x, y) \land \psi(y)] = (\exists y.x = O_1 \land y = O_2 \land \psi(y))
\]
\[
\equiv x = O_1 \land \exists y.y = O_2 \land \psi(y).
\]

Clearly, \( x = O_1 \land \exists y.y = O_2 \land \psi(y) \) is of the form \( \nu(x) \land \eta(y) \): let \( \nu(x) \) be \( x = O_1 \)
and let \( \eta(y) \) be \( \exists y.y = O_2 \land \psi(y) \).

(13) \( \exists y(\exists x(a = A(x, y))) \land \psi(x) \)

Assume that \( \alpha = A(O_1, O_2) \), then
\[
\epsilon[\exists y(\exists x(a = A(x, y))) \land \psi(x)] = \exists y(\exists x(x = O_1 \land y = O_2)) \land \psi(x)
\]
\[
= \exists y(\exists x(x = O_1 \land y = O_2)) \land \psi(x)
\]
\[
\equiv \exists x(x = O_1) \land \exists y(y = O_2) \land \psi(x).
\]

Clearly, \( \exists x(x = O_1) \land \exists y(y = O_2) \land \psi(x) \) is of the form \( \nu(x) \land \eta(y) \): let \( \nu(x) \) be \( \exists y(y = O_2) \land \psi(x) \) and let \( \eta(y) \) be \( \exists x(x = O_1) \).

(13') \( \exists y(\exists x(a = A(x, y))) \land \psi(y) \)
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

233

Assume that \(\alpha = A(O_1, O_2)\), then

\[
\epsilon[\exists y(\exists x(\alpha = A(x, y))) \land \psi(y)] = \exists y(\exists x = O_1 \land y = O_2) \land \psi(y)
\]

\[
= \exists y(\exists x = O_1 \land y = O_2) \land \psi(y).
\]

Clearly, \(\exists y(\exists x = O_1 \land y = O_2) \land \psi(y)\) is of the form \(\nu(x) \land \eta(y)\): let \(\nu(x)\) be true and let \(\eta(y)\) be \(\exists y(\exists x = O_1 \land y = O_2) \land \psi(y)\).

(14) \(\exists y.\exists x(a = A(x, y)) \land \psi(x)\)

Case (14) is equivalent to case (13), because in case (14) the quantification range of \(y\) is in fact only over \(a = A(x, y)\).

(14') \(\exists y.\exists x(a = A(x, y)) \land \psi(y)\)

Assume that \(\alpha = A(O_1, O_2)\), then

\[
\epsilon[\exists y.\exists x(\alpha = A(x, y)) \land \psi(y)] = \exists y.\exists x = O_1 \land y = O_2 \land \psi(y)
\]

\[
\equiv \exists x = O_1 \land \exists y. y = O_2 \land \psi(y).
\]

Clearly, \(\exists x = O_1 \land \exists y. y = O_2 \land \psi(y)\) is of the form \(\nu(x) \land \eta(y)\): let \(\nu(x)\) be \(\exists y. y = O_2 \land \psi(y)\) and let \(\eta(y)\) be \(\exists x = O_1\).

(15) \(\exists x.\exists y(a = A(x, y)) \land \psi(x)\)

Assume that \(\alpha = A(O_1, O_2)\), then

\[
\epsilon[\exists x.\exists y(\alpha = A(x, y)) \land \psi(x)] = \exists x.\exists y = O_1 \land y = O_2 \land \psi(x)
\]

\[
\equiv \exists y = O_2 \land \exists x. x = O_1 \land \psi(x).
\]

Clearly, \(\exists y = O_2 \land \exists x. x = O_1 \land \psi(x)\) is of the form \(\nu(x) \land \eta(y)\): let \(\nu(x)\) be \(\exists y = O_2\) and let \(\eta(y)\) be \(\exists x. x = O_1 \land \psi(x)\).

(15') \(\exists x.\exists y(a = A(x, y)) \land \psi(y)\)

Case (15') is equivalent to case (13), because in case (15') the quantification range of \(x\) is in fact only over \(a = A(x, y)\).

(16) \(\exists x.\exists y.a = A(x, y) \land \psi(x)\)

Case (16) is equivalent to case (14'), because in case (16) the quantification range of \(y\) is in fact only over \(a = A(x, y)\).

(16') \(\exists x.\exists y.a = A(x, y) \land \psi(y)\)

Case (16') is equivalent to case (13), because in case (16') the quantification range
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

of $x$ is in fact only over $a = A(x, y)$.

Notice that for each of the cases above, it takes no more than one step of logical transformations to find the equivalent formula of the form $\nu(x) \land \eta(y)$ such that $\nu(x) \in FO_{DL}^c$ and $\eta(y) \in FO_{DL}^b$, which takes a constant number of steps with respect to the size of the resulting formula. Now, we complete the proof of Statement (2) for case (d.2). Recall that we consider the last remaining case when $W = \exists y. R(x, y, S) \land W_1(y)[S]$.

$$
\epsilon[\rho[W]] = \epsilon[\rho[\exists y. R(x, y, S) \land W_1(y)[S]]]
\begin{align*}
&= \exists y. \epsilon[\rho[R(x, y, S)]] \land \epsilon[\rho[W_1(y)[S]]] \\
&= \exists y. \epsilon[\rho[R(x, y, S)]] \land W_1'(y)[S_1]
\end{align*}
$$

(by the induction hypothesis on $W_1(y)[S_1]$, let $\epsilon[\rho[W_1(y)[S]]] = W_1'(y)[S_1]$ be a formula uniform in $S_1$ and $W_1'(y)$ is in $FO^b_{DL}$; moreover, $W_1'(y)$ can be found in no more than $c \cdot size(W_1'(y))$ steps for some constant $c$)

$$
= \exists y. (\bigvee_{i=1}^{m+} \phi^+_i(x, y, \alpha) \lor R(x, y, S_1) \land \neg (\bigvee_{j=1}^{m-} \phi^-_j(x, y, \alpha)) \land W_1'(y)[S_1])
\begin{align*}
&= \exists y. (\bigvee_{i=1}^{m+} \nu^+_i(x) \land \eta^+_i(y) \lor R(x, y, S_1) \land \neg (\bigvee_{j=1}^{m-} \nu^-_j(x) \land \eta^-_j(y)) \land W_1'(y)[S_1])
\end{align*}
$$

(according to the proof above, for each $i$, let $\epsilon[\phi^+_i(x, y, \alpha)] = (\nu^+_i(x) \land \eta^+_i(y))$ be a formula uniform in $S_1$ for some $\nu^+_i(x) \in FO^b_{DL}$ and $\eta^+_i(y) \in FO^b_{DL}$; for each $j$, let $\epsilon[\phi^-_j(x, y, \alpha)] = (\nu^-_j(x) \land \eta^-_j(y))$ be a formula uniform in $S_1$ for some $\nu^-_j(x) \in FO^b_{DL}$ and $\eta^-_j(y) \in FO^b_{DL}$)

$$
= \exists y. (\bigvee_{i=1}^{m+} (\nu^+_i(x) \land \eta^+_i(y) \land W_1'(y)) \lor R(x, y) \land W_1'(y) \land \bigwedge_{j=1}^{m-} (\neg \nu^-_j(x) \land \eta^-_j(y))[S_1])
\begin{align*}
&= \{ \bigvee_{i=1}^{m+} (\nu^+_i(x) \land \exists y(\eta^+_i(y) \land W_1'(y))) \lor \exists y. \bigwedge_{j=1}^{m-} (\neg \nu^-_j(x) \lor \neg \eta^-_j(y)) \land R(x, y) \land W_1'(y))[S_1] \\
&= \{ \bigvee_{i=1}^{m+} (\nu^+_i(x) \land \exists y(\eta^+_i(y) \land W_1'(y)) \lor \bigwedge_{k=1}^{2^{m-}} (\neg \nu^-_j(x) \land \exists y(R(x, y) \land W_1'(y)) \land \neg \eta^-_j(y))[S_1] \}
\end{align*}
$$
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

(by applying distributive law over conjunction of \(j\) and then minimizing the scope of \(\exists y\). for the negative conditions using the laws:

\[
\exists y. P_1(y) \lor P_2(y) \equiv \exists y(P_1(y)) \lor \exists y(P_2(y)), \text{ and}
\exists y. P_1(y) \land P_2(y) \equiv P_1 \land \exists y(P_2(y)) \text{ when } y \text{ is not free in } P_1
\]

\[
= W'[S_1],
\]

where each \(N_k\) \((k = 1, 2, \cdots, m_-)\) enumerates a subset of indices \(\{1, 2, \cdots, m_-\}\), and \(\overline{N_k} = \{1, 2, \cdots, m_-\} - N_k\), i.e., \(\overline{N_k}\) is the complement set of \(N_k\). It is clear the formula on the RHS of Eq. (B.4) (denoted as \(W'[S_1]\) above) is equivalent to \(\epsilon[\rho[W]]\), is \(L^{C^2}_{sc}\) regressable, and is in \(FO^x_{DL}\) when \(S_1\) is suppressed, and has no appearance of \(Poss\). It is easy to see that to find it, it takes no more than \(c \cdot \text{size}(W')\) for some integer \(c\). Hence, Statement (2) is true for \(W\). Moreover, according to Corollary 6, we have that \(R^{C^2}[W] = R^{C^2}[\epsilon[\rho[W]]] \equiv R^{C^2}[W'[S_1]]\). Then, by the induction hypothesis on formulas uniform in \(S_1\), we have \(R^{C^2}[W][-S_0]\) will be equivalent to some formula in \(FO^x_{DL}\).

It is very similar to prove that Statements (1) and (2) are true when \(W[-S]\) is of the form \(\forall y. R(x, y) \supset W_1(y)\), and details are omitted here.

Similarly, we can show that Statement (1) and Statement (2) are true when \(W[-S]\) is in \(FO^y_{DL}\) and is not atomic. Overall, we proved for Statement (1).

Now, consider any \(L^{C^2}_{sc}\) regressable \(W\) that is uniform in a ground situation \(S\). When \(W[-S]\) is in \(FO^x_{DL}\), assume that we have found some \(\Phi_W\) that is in \(FO^x_{DL}\), such that \(R^{C^2}[W] \equiv \Phi_W[S_0]\). Below we will estimate an upper bound on the size of \(\Phi_W\). Let \(n = \text{sitLength}(S)\) \((n \in \mathbb{N} \text{ and } n \geq 0)\), i.e., the number of action terms involved in \(S\). Let \(m = \text{size}(W)\) \((m \in \mathbb{N} \text{ and } m \geq 1)\). Let function \(f(m, n)\) be the size of \(\Phi_W\), which is a non-decreasing function.
Firstly, it is straightforward that $f(1, 0) = 1$.

Secondly, when $m = 1$, $W$ is atomic, which is either true, or false, or $x = b$ for some constant $b$, or a situation-independent predicate, or a primitive dynamic concept. We now consider $n \geq 1$. Assume that $S = \text{do}(\alpha_n, S_1)$, and $\text{sitLength}(S_1) = n - 1$. According to the discussion above of “the base case of the induction on the structure of $W[-S]$” (i.e., cases (a-c)), for any $n \in \mathbb{N}$ and $n \geq 1$, $f(1, n) \leq f(3h, n-1)$, where $h = \max(2, \text{sizeSSA}(D))$ ($h$ is a constant number for the given $D$). By Corollary 6, we have $\mathcal{R}^{C2}[W] = \mathcal{R}^{C2}[^\epsilon[\rho[W]]]$, where $\epsilon[\rho[W]]$ is uniform in $S_1$ (no matter whether it is situation-independent or not), and is equivalent to some $\Phi_1 \in FO^x_{DL}$, whose size is no more than $3h$ (including all cases when $W$ is atomic). Moreover, the equivalent formula $\Phi_W[S_0]$ that we are looking for can be obtained by looking for the equivalent formula of $\mathcal{R}^{C2}[\Phi_1[S_1]]$, whose size is no more than $f(3h, n-1)$.

Thirdly, we consider any $m \geq 2$ and $n \in \mathbb{N}$. In fact, when $m \geq 2$, $W$ is not atomic. According to the definition of $FO^x_{DL}$, there are three sub-cases.

1. $W[-S]$ is of the form $\neg W_1$, or $\exists y.W_1(y)$, or $\forall y.W_1(y)$ where $W_1$ is in $FO^x_{DL}$, or $W_1(y)$ is in $FO^y_{DL}$. It is easy to see that $f(m, n) = f(m-1, n) + 1$ according to the definition of the regression operator $\mathcal{R}^{C2}$ and the way $FO^x_{DL}$ constructed. For example, $\mathcal{R}^{C2}[\neg W_1[S]] = \neg \mathcal{R}^{C2}[W_1[S]]$, if we find a formula $\Phi_1$ in $FO^x_{DL}$ such that $\Phi_1[S_0]$ is equivalent to $\mathcal{R}^{C2}[W_1[S]]$, then $\Phi_W = \neg \Phi_1$ is the formula that we are looking for.

2. $W[-S]$ is of the form $W_1 \lor W_2$, or $W_1 \land W_2$ where $W_1$ and $W_2$ are in $FO^x_{DL}$. It is easy to see that $f(m, n) = f(\text{size}(W_1), n) + f(\text{size}(W_2), n) + 1$, which is no more than $2f(m-1, n) + 1$, according to the definition of the regression operator $\mathcal{R}^{C2}$ and the way $FO^x_{DL}$ constructed.

3. $W[-S]$ is of the form $\exists y.R(x, y) \land W_1(y)$, or $\forall y.R(x, y) \supset W_1(y)$, where $W_1(y)$ is in $FO^x_{DL}$. According to the definition of $\text{size}$ in Section 3.3.2, we have $\text{size}(W_1(y)) = \ldots$
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

$m - 3$. For instance, we consider the case when $W[-S]$ is of the form $\exists y. R(x, y) \land W_1(y)$, and it is similar for the case when $W[-S]$ is of the form $\forall y. R(x, y) \supset W_1(y)$.

According to the definition of $\mathcal{R}^{C^2}$, we have

$$
\mathcal{R}^{C^2}[W] = \mathcal{R}^{C^2}[\exists y. R(x, y)[S] \land W_1(y)[S]]
= \exists y. \mathcal{R}^{C^2}[R(x, y)[S]] \land \mathcal{R}^{C^2}[W_1(y)[S]]
= \begin{cases} 
\exists y. \mathcal{R}^{C^2}[R(x, y, S)] \land \mathcal{R}^{C^2}[W_1(y)[S]] & \text{if } R \text{ is a fluent,} \\
\exists y. R(x, y) \land \mathcal{R}^{C^2}[W_1(y)[S]] & \text{otherwise.}
\end{cases}
$$

Assume that $\mathcal{R}^{C^2}[W_1(y)[S]]$ with the initial situation $S_0$ suppressed is equivalent to some $\Phi_1(y) \in FO_{DL}^y$. When $R$ is situation-independent, $\exists y. R(x, y) \land \Phi_1(y)$ is the formula that we are looking for, whose size is $f(m - 3, n) + 3$. When $R$ is a dynamic role, we assume that its SSA is of the form Eq. (B.3), $S_i = do([\alpha_1, \cdots, \alpha_{n-i}], S_0)$ for any $i=1..n-1$, and $S = do(\alpha_n, S_1)$. Since it is difficult to estimate the upper of the size of $R[\exists y. R(x, y, S) \land W_1(y)[S]]$ in this case recursively using Eq. (B.4), we will compute its regression result and then estimate its size. Note that we give the computation of the regression first and then provide the explanations for some major steps in the computation right after the equations.

$$
\mathcal{R}^{C^2}[W] = \exists y. \mathcal{R}^{C^2}[R(x, y, S)] \land \mathcal{R}^{C^2}[W_1(y)[S]]
= \exists y. \mathcal{R}^{C^2}[R(x, y, S)] \land \Phi_1(y)[S_0]
= \exists y. \bigvee_{i=1}^{m_+} \mathcal{R}^{C^2}[\phi_i^+(x, y, \alpha_n)] \lor \mathcal{R}^{C^2}[R(x, y, S_1)] \land \neg (\bigvee_{j=1}^{m_-} \mathcal{R}^{C^2}[\phi_j^-(x, y, \alpha_n)]) \land \Phi_1(y)[S_0]
= \exists y. \bigvee_{i=1}^{m_+} \mathcal{R}^{C^2}[\epsilon[\rho[\phi_i^+(x, y, \alpha_n)]]] \lor \mathcal{R}^{C^2}[R(x, y, S_1)] \land \neg (\bigvee_{j=1}^{m_-} \mathcal{R}^{C^2}[\epsilon[\rho[\phi_j^-(x, y, \alpha_n)]]]) \land \Phi_1(y)[S_0]
$$

(use Corollary 6)

$$
= \exists y. \bigvee_{i=1}^{m_+} (\mu_{i,n}^+(x) \land \eta_{i,n}^+(y)) \lor \mathcal{R}^{C^2}[R(x, y, S_1)] \land (\bigwedge_{j=1}^{m_-} (\nu_{j,n}^-(x) \lor \neg \eta_{j,n}^-(y))) \land \Phi_1(y)[S_0]
$$

(use the result in Statement (2) for each $\phi_i^+(x, y, \alpha_n)$ and $\phi_j^-(x, y, \alpha_n)$, i.e., for each $i$,

$\mathcal{R}^{C^2}[\epsilon[\rho[\phi_i^+(x, y, \alpha_n)]]] \equiv \nu_{i,n}^+(x) \land \eta_{i,n}^+(y)$ for some $\nu_{i,n}^+(x) \in FO_{DL}^x$ and some $\eta_{i,n}^+(y) \in FO_{DL}^y$;

also, for each $j$, $\mathcal{R}^{C^2}[\epsilon[\rho[\phi_j^-(x, y, \alpha_n)]]] \equiv \nu_{j,n}^-(x) \land \eta_{j,n}^-(y)$

for some $\nu_{j,n}^-(x) \in FO_{DL}^x$ and some $\eta_{j,n}^-(y) \in FO_{DL}^y$, respectively)
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

238

of \(R\) into a sort of disjunctive normal form (DNF) (from Step (B.6) to Step (B.9)) based

steps of logical transformations. First, we transform 

\[(\exists x, y, R(x, y, \alpha_{n-1})) \equiv \exists x, y, R(x, y, \alpha_{n-1}) \land \Phi_1(y)[S_0] \land \neg(\exists x, y, R(x, y, \alpha_{n-1})) \land (\bigwedge_{j=1}^{m} (\neg \nu_{j,n}^+(x) \lor \neg \eta_{j,n}^-(y))) \land \Phi_1(y)[S_0] \]

(each \(\nu_{i,n}^+(x), \eta_{i,n}^-(y), \nu_{j,n}^-(x), \eta_{j,n}^-(y)\) are situation-independent,
and apply one-step regression on \(R(x, y, S_1)\), where \(S_1 = do(\alpha_{n-1}, S_2)\))

\[\equiv \exists y, \left(\bigvee_{i=1}^{m_+} (\nu_{i,n}^-(x) \land \eta_{i,n}^+(y)) \lor \bigvee_{i=1}^{m_-} (\nu_{i,n}^-(x) \land \eta_{i,n}^+(y)) \lor \bigwedge_{i=1}^{m_-} (\neg \nu_{i,n}^+(x) \lor \neg \eta_{i,n}^-(y)) \land \Phi_1(y)[S_0] \right) \land \bigwedge_{i=1}^{m_-} (\neg \nu_{i,n}^+(x) \lor \neg \eta_{i,n}^-(y)) \land \Phi_1(y)[S_0] \]

(use the result in Statement (2) for each \(\phi_i^+(x, \alpha_{n-1})\) and \(\phi_j^-(x, \alpha_{n-1})\))

\[= \cdots \left\{ \begin{array}{l} \gamma^+ = \bigvee_{i=1}^{m_+} (\nu_{i,n}^+(x) \land \eta_{i,n}^+(y)), \gamma^- = \bigwedge_{i=1}^{m_-} (\neg \nu_{i,n}^+(x) \lor \neg \eta_{i,n}^-(y)), l = 1..n \right\} \]

\[= \exists y, \left(\gamma^+_n \lor (\gamma^-_{n-1} \lor (\cdots \lor (\gamma^+_1 \lor R(x, y, S_0) \land \gamma^-_1) \lor \cdots) \lor \gamma^-_{n-1}) \land \gamma^-_n \right) \land \Phi_1(y)[S_0] \]

\[= \left\{ \exists y, (\gamma^+_n \lor \gamma^-_{n-1} \lor \gamma^-_j \lor \cdots \lor \gamma^+_j \lor R(x, y) \land \bigwedge_{i=1}^{n} \gamma^-_i \lor \Phi_1(y)[S_0] \right\} \]

\[= \cdots \left( \text{use distributive law to obtain a sort of DNF format} \right) \]

\[= \left\{ \exists y, \left(\bigvee_{i=1}^{u} \Phi_{S,i}(x, y) \land \Phi_1(y)[S_0] \right) \text{ for some index } u \right\} \text{ (B.9)} \]

(each \(\Phi_{S,i}(x, y)\) is a conjunction of some of the sub-formulas in the set \(\{R(x, y)\} \cup \{\nu_{i,j}^+(x), \eta_{i,j}^+(y), \nu_{i,j}^-(y), \eta_{i,j}^-(y) | i = 1..m_+, j = 1..m_-, l = 1..n\}\))

\[= \left\{ \bigvee_{i=1}^{u} \Phi_{S,i}(x, y) \land \Phi_1(y)[S_0] \right\} \text{ (B.10)} \]

\[= \left\{ \bigvee_{i=1}^{u} \Psi_{S,i}(x) \right\}[S_0] \text{ (B.11)} \]

where each \(\nu_{i,j}^+(x) \land \eta_{i,j}^+(y)\) (\(\nu_{j,i}^-(x) \land \eta_{j,i}^-(y)\), respectively) is equivalent to the result of \(\mathcal{R}^{C2}[\epsilon[\rho[\phi_i^+(x, y, \alpha_k)]]] (\mathcal{R}^{C2}[\epsilon[\rho[\phi_j^-(x, y, \alpha_k)]]], \) respectively). Here, each \(\nu_{i,j}^+(x)\) (\(\nu_{j,i}^-(x)\), respectively) is in \(FO^x_{DL}\) with at most one free variable \(x\), and each \(\eta_{i,j}^+(y)\) (\(\eta_{j,i}^-(y)\), respectively) is in \(FO^y_{DL}\) with at most one free variable \(y\), according to the proof for the cases (1)-(16) and (1')-(16') in Table B.4. Notice that in order to obtain an equivalent formula of \(\mathcal{R}^{C2}[W] [-S_0] \) in \(FO^x_{DL}\), we need to perform the following steps of logical transformations. First, we transform \(\mathcal{R}^{C2}[R(x, y, S)] \) in Step (B.6) into a sort of disjunctive normal form (DNF) (from Step (B.6) to Step (B.9)) based
on the assumption that each $\nu_{i,j}^+, (\eta_{i,j}^+(y), \nu_{i,j}^-(x), \eta_{i,j}^-(y)$, respectively) is "atomic", i.e., when each of these sub-formulas is considered as an atom, after using the distributive law, the resulting sub-formula $\bigvee_{i=1}^n \Phi_{S,i}(x, y)$ in Step B.9 is a DNF formula. Since the resulting formula is too long, we omit the details and only provide one example of a sub-formula in Step (B.8). For instance, we perform the distributive law over subformula $\gamma_{n-1}^+ \land \gamma_n^-:
\gamma_{n-1}^+ \land \gamma_n^- \equiv \bigvee_{i=1}^{m_+} (\nu_{i,n-1}^+(x) \land \eta_{i,n-1}^+(y) )) \land (\bigwedge_{j=1}^{m_-} (\neg \nu_{j,n}^-(x) \lor \neg \eta_{j,n}^-(y)))$
where $N_k \subseteq \{1, 2, \cdots, m_-(k=1, 2, \cdots, m_-)\}$ enumerates all sub-sets of $\{1, 2, \cdots, m_-\}$, and $\overline{N}_k = \{1, 2, \cdots, m_-\} - N_k$, i.e., is the complement set of $N_k$. Next, we distribute $\Phi_1(y)[S_0]$ into the resulting DNF formula (from Step (B.9) to Step (B.10)). Finally, we push $\exists y$ inside into each conjunctive clause and minimize the scope of each quantifier $\exists y$ (from Step (B.9) to Step (B.11)). In Step (B.9), after using the commutative law of conjunctions, each $\Phi_{S,i}(x, y)$ is either of the form $\nu_{S,i}(x) \land \eta_{S,i}(y)$ or of the form $\nu_{S,i}(x) \land R(x, y) \land \eta_{S,i}(y)$ for some $\nu_{S,i}(x) \in FO^r_{DL}$ and some $\eta_{S,i}(y) \in FO^r_{DL}$. From Step (B.10) to Step (B.11), there are two cases for each index $i$: if $\Phi_{S,i}(x, y)$ is of the form $\nu_{S,i}(x) \land \eta_{S,i}(y)$, then $\Psi_{S,i}(x)$ is $\nu_{S,i}(x) \land (\exists y.\eta_{S,i}(y) \land \Phi_1(y))$; else if $\Phi_{S,i}(x, y)$ is of the form $\nu_{S,i}(x) \land R(x, y) \land \eta_{S,i}(y)$, then $\Psi_{S,i}(x)$ is $\nu_{S,i}(x) \land (\exists y.R(x, y) \land \eta_{S,i}(y) \land \Phi_1(y))$. Hence, the resulting formula in Step (B.11), with $S_0$ suppressed, is in $FO^r_{DL}$, and we denote the formula (with $S_0$ suppressed) as $\Phi_W$.

Note that we did not use the result of Statement (2) to estimate the upper bound, because for each one-step regression, in order to get an equivalent formula in $FO_{DL}$ that is uniform in $S_1$, we had to distribute $W_1(y)$ (see Eq. (B.4)) at every inductive step, which will finally result in a formula uniform in $S_0$ with larger size.

Now we estimate the size of $\Phi_W$ when $R$ is a fluent according to the way it is con-
Moreover, we can perform a similar estimation for the case where $g(y, R) = m$ where $f(n)$ is constructed above. First, for any $n \geq 0$ and any situation $S$ where $sitLength(S) = n$, we estimate an upper bound on the size of the DNF formula (denoted as $g(n)$ below), which is equivalent to $R^C^2[R(x, y, S)]$ and constructed specifically according to the above steps (B.6-B.9). Note that for each $i = 1..m_+$, $j = 1..m_-$ and $l = 1..n$, $size(\nu^+_i(x))$, $size(\eta^+_j(y))$, $size(\nu^-_j(x))$, $size(\eta^-_j(y))$, respectively, is no more than $h + 2$ according to the above for the cases (1)-(16) and (1')-(16') in Table B.4. Moreover, according to the definition of function $size()$ in Section 3.3.2, the logical constructors should also be counted. Also, for any $m_-$ and $m_+$ for any role $R$, we always have $m_- < h$ and $m_+ < h$ (recall that constant number $h = \max(2, sizeSSA(D))$). According to Step B.11, $f(m, n) \leq (f(m-3, n)+3)g(n)$, where $f(m-3, n) = size(\Phi_1(y))$.

Overall, when $m \geq 2$, $f(m, n) \leq \max(f(m - 1, n) + 1, 2f(m - 1, n) + 1, f(m - 3, n) + 3, g(n)(f(m - 3, n) + 3)A)$.

Below, we show that $g(n) \leq c_1 2^{nh}$ for some constant $c_1 = (2(h + 3) + (h + 4)h^2 + 2)2^{2h}$. Moreover, we can perform a similar estimation for the case when $W[-S]$ is of the form $\forall y. R(x, y) \supseteq W(y)$.

$$g(n) \leq 2(h + 3)m_+ + (2^{m_-})m_+(2(h + 3) + m_-(h + 4)) + (2^{m_-})^2m_+(2(h + 3) + 2m_-(h + 4)) + \ldots + (2^{m_-})^nm_+(2(h + 3) + (n - 1)m_-(h + 4)) + (2^{m_-})^n(2 + nm_-(h + 4))$$

$$= 2(h + 3)m_+ \sum_{i=0}^{n-1} (2^{m_-})^i + (h + 4)m_- \sum_{i=1}^{n} i(2^{m_-})^i + 2(2^{m_-})^n$$

$$= 2(h + 3)m_+ \sum_{i=0}^{n-1} (2^{m_-})^i + (h + 4)m_+m_- \sum_{i=1}^{n} i(2^{m_-})^i + 2(2^{m_-})^n$$

$$< 2(h + 3)h \sum_{i=0}^{n-1} (2^h)^i + (h + 4)h^2 \sum_{i=1}^{n} i(2^h)^i + 2(2^h)^n$$

$$\leq 2(h + 3)h(2^h)^n + (h + 4)h^2 n(2^h)^{n+1} + 2(2^h)^n$$

$$\leq 2(h + 3)h(2^h)^n + (h + 4)h^2 (2^h)^{n+2} + 2(2^h)^n$$

$$\leq (2(h + 3) + (h + 4)h^2 + 2)(2^h)^{n+2}$$

$$\leq c_1 2^{nh} \quad \text{(let constant number } c_1 = (2(h + 3) + (h + 4)h^2 + 2)2^{2h}).$$
Appendix B. Proofs of Lemmas and Theorems in Chapter 3

We can perform a similar estimation for the case when $W[-S]$ is in $FO^u_{DL}$. Overall, for $m \geq 1$, $n \geq 1$, the upper bound of $f(m, n)$ is

$$f(m, n) \leq \max(f(m-1, n) + 1, 2f(m-1, n) + 1, f(m-3, n) + 3, g(n)(f(m-3, n) + 3))$$

$$\leq c_12^{hn}(f(m-1, n) + 3)$$

$$\leq c_12^{hn}(c_12^{hn}(f(m-2, n) + 3) + 3)$$

$$\leq \cdots$$

$$\leq c_12^{hn}(c_12^{hn}(\cdots(2^{hn}(f(1, n) + 3)) + \cdots + 3) + 3)$$

$$= (c_12^{hn})^{m-1}f(1, n) + 3 \sum_{i=0}^{m-2}(c_12^{hn})^i$$

$$\leq (c_12^{hn})^{m-1}(f(1, n) + 3)$$

$$\leq c_1^{m-1}2^{hn(m-1)}(f(3h, n - 1) + 3)$$

$$\leq c_1^{m-1}2^{hn(m-1)}(c_1^{3h-1}2^{hn(3h-1)})(f(1, n - 1) + 3) + 3)$$

$$\leq c_1^{m-1}2^{hn(m-1)}(c_1^{3h-1}2^{hn(3h-1)})(f(3h, n - 2) + 3) + 3)$$

$$\leq \cdots$$

$$\leq c_1^{m-1}2^{hn(m-1)}((c_1^{3h-1}2^{hn(3h-1)})^nf(1, 0) + 3 \sum_{i=0}^{n}(c_1^{3h-1}2^{hn(3h-1)})^i)$$

$$\leq c_1^{m-1}2^{hn(m-1)}((c_1^{3h-1}2^{hn(3h-1)})^n + 3(c_1^{3h-1}2^{hn(3h-1)})^n+1)$$

$$\leq 4c_1^{m-1}2^{hn(m-1)}(c_1^{3h-1}2^{hn(3h-1)})^{n+1}$$

$$= 4c_1^{(n+1)(3h-1)+(m-1)+2}hn(m-1)+h(3h-1)n(n+1)$$

$$= 2^{2c_2((n+1)(3h-1)+(m-1)+h(3h-1)n(n+1)+2}$$

where constant numbers $c_2 = \log_2 c_1$ and $c_1 = (2(h + 3) + (h + 4)h^2 + 2)2^{2h}$. When $n = 0$, $f(m, 0) = 1$, which is clearly no more than $2^{2c_2((n+1)(3h-1)+(m-1)+h(3h-1)n(n+1)+2}$ as well. Hence, we finally have

$$f(m, n) \in O(2^{hmn+3h^2n^2}), \text{ where constant } h = \max(2, \text{sizeSSA}(D)).$$

That is, the size of the equivalent formula of $R^{C^2}[W]$ that we are looking for is no more than exponential in the size of the given formula $W$. \[\square\]