Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

Incomplete Causal Laws in the Situation Calculus Using Free Fluents

Marcelo Arenas, Jorge A. Baier, Juan S. Navarro
Universidad Católica de Chile
{marenas, jabaier, juansnn}@ing.puc.cl

Sebastian Sardina
RMIT University, Australia
sebastian.sardina@rmit.edu.au

Abstract

We propose a simple relaxation of Reiter's basic action theories, based on fluents without successor state axioms, that accommodates incompleteness beyond the initial database. We prove that fundamental results about basic action theories can be fully recovered and that the generalized framework allows for natural specifications of various forms of incomplete causal laws. We illustrate this by showing how the evolution of incomplete databases, guarded action theories, and non-deterministic actions can be conveniently specified.

1 Introduction

The situation calculus [McCarthy and Hayes, 1969; Reiter, 2001] is a popular and well-established second-order logical formalism for representing and reasoning about dynamic systems. Basic action theories (BATs) [Reiter, 1991] are axiomatizations in the situation calculus of a certain shape describing how the world evolves under the effects of actions. Importantly, BATs provide a parsimonious and effective way to solve the frame problem within classical logic, a problem that resisted solution for decades [McCarthy and Hayes, 1969; Shanahan, 1997]. This type of theories have been studied at depth, and have been shown elaboration tolerant via multiple extensions, such as time [Pinto, 1994], high-level programs [Levesque *et al.*, 1997], ramifications [Pinto, 1999; McIlraith, 2000], and concurrency and natural actions [Reiter, 1996].

BATs enjoy a number of good properties. Among them are the relative satisfiability and regression theorems. Both reduce important tasks—including the projection task [Pirri and Reiter, 1999; Reiter, 2001]—to first-order theorem proving on the axiomatization for the initial situation. This is important because the situation calculus is a second-order logic.

In BATs there ought to be *exactly one* successor state axiom per fluent. Each of these axioms characterizes precisely how each fluent evolves with actions. Possibly because of such a requirement, there is an informal understanding that BATs are not able to accommodate incomplete knowledge on the causal laws of the domain. In the literature, this reflected by at least three extensions to Reiter's formalism for handling incomplete knowledge at the causal level. *Guarded action theories* (GATs) [De Giacomo and Levesque, 1999], allow both the representation of the occlusion principle [Sandewall, 1994] and non-deterministic actions. An important limitation of GATs is that they do not come with a complete regression procedure. The Probabilistic Situation Calculus [Pinto *et al.*, 2000] is major variant of Reiter's BATs which can represent non-deterministic and probabilistic actions, requiring different foundational axioms, a different meta-theory, and a re-work of important theorems. Finally, Delgrande and Levesque [2013] propose treatment a for non-deterministic actions that builds on an epistemic view of the Situation Calculus, which also requires different foundational axioms.

In this work, we show how non-determinism in causal laws can be accommodated in the Situation Calculus by simply allowing some fluents to have no successor state axiom. We call such fluents *free fluents*, which are analogous to "determining fluents" in the Event Calculus [Shanahan, 1999]. Our approach, which is a small modification to the simplest form of the Situation Calculus, does not need any changes to foundational axioms. Moreover, it allows the modeling of a range of applications, such as incomplete databases, guarded action theories, and non-deterministic actions.

Below we prove that fundamental theorems on BATs can be fully recovered in our relaxation. Specifically, relative satisfiability and the regression theorem still apply in our generalized theories. Then we show how free fluents can be applied to model non-deterministic actions and databases with incomplete knowledge. We continue by showing that GATs can be mapped to our BATs. This has significance since known forms of regression for GATs are limited to certain types of regressable formulas, while our regression is not. We finalize with a discussion of related work in which we analyze a few advantages, besides simplicity, that our approach has over Pinto *et al.*'s and Delgrande and Levesque's approaches. Moreover, we explain that, surprisingly, free fluents can be "compiled away" provided we allow infinite distinguished elements in the domain, albeit yielding substantially more convoluted theories.

2 Action Theories in the Situation Calculus

The Situation Calculus [Reiter, 2001] $\mathcal{L}_{sitcalc}$, is a many-sorted second-order language with equality, designed for representing dynamically changing worlds whose changes are the result of ***actions***. Terms of sort ***situation*** are finite se-

quences of actions: the empty sequence is denoted by the distinguished constant S_0, and $do(a, s)$ denotes the situation that results from performing action a in situation s. For legibility, we write $do([A_1, \ldots, A_k], s)$ to denote the situation resulting from executing actions $A_1, \ldots, A_k$ in situation s. Sort *object* is a catch-all sort for terms that denote objects of the world.

In $\mathcal{L}_{sitcalc}$, ***fluents*** describe the dynamic aspects of the world. A relational (functional) fluent is a predicate (function) whose last argument is a situation, and thus whose truth value (value) can change from situation to situation. For example, relational fluent $Broken(x, s)$ can be used to denote that object x is broken in situation s, and functional fluent $salary(x, s)$ to denote the salary of employee x in situation s.

Actions need not be executable in all situations. Predicate $Poss(a, s)$ is used to state that action a is executable in situation s. Finally, the distinguished binary predicate $s \sqsubset s'$ states that situation s represents a sub-history of situation s, that is, $s' = do([A_1, \ldots, A_k], s)$, for some non-empty sequence of actions $A_1, \ldots, A_k$.

A formula φ of $\mathcal{L}_{sitcalc}$ is called ***uniform*** in situation term σ [Reiter, 2001] if the only situation term mentioned in φ is σ, neither *Poss*, $\sqsubset$, nor equalities between situations are mentioned, and there is no quantification over situations.

In this paper we embrace Reiter's ***basic action theories*** (BATs), which is a well-established form to build $\mathcal{L}_{sitcalc}$ theories. Let $\mathcal{D} = \mathcal{D}_{poss} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \Sigma$ such that: (1) $\mathcal{D}_{poss}$ is the set of *precondition axiom* of the form $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$, where $\Pi_A(\vec{x}, s)$ is a formula uniform in s; (2) $\mathcal{D}_{ssa}$ is the set of *successor state axiom* characterizing how fluents evolve w.r.t. actions (see below); (3) $\mathcal{D}_{una}$ is the set of *unique names axioms* for actions; (4) $\mathcal{D}_{S_0}$ is a set of formulae uniform in S_0, defining the initial situation; and (5) Σ is a set of domain-independent foundational axioms.

In particular, the set $\mathcal{D}_{ssa}$ contains one ***successor state axiom*** (SSA) per relational fluent F and functional fluent f of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$ and $f(\vec{x}, do(a, s)) = y \equiv \Phi_f(\vec{x}, y, a, s)$, resp., where Φ_F and Φ_f are formulas uniform in s characterizing how F and f evolve, resp., in situation s when action a is performed. Reiter [1991] defined a way to generate successor SSAs from a set of so-called ***effect axioms***, stating how an action changes the value of a fluent.

A BAT is a set of axioms $\mathcal{D}$ of the form above and satisfying the *consistency property* such that for each functional SSA $f(\vec{x}, do(a, s)) = y \equiv \Phi_f(\vec{x}, y, a, s)$ in $\mathcal{D}_{ssa}$ we have:[1]

$$\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \exists y.\Phi_f(\vec{x}, y, a, s),$$
$$\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \Phi_f(\vec{x}, y_1, a, s) \wedge \Phi_f(\vec{x}, y_2, a, s) \supset y_1 = y_2.$$

In the rest of the paper we also utilize Reiter's notion of ***legal (or executable) situation*** [2001], that is, one in which all actions are possible. For ground situation terms:

$$Legal(do([A_1, \ldots, A_i], s)) \doteq \bigwedge_{i=1}^{n} Poss(A_i, do([A_1, \ldots, A_{i-1}], S_0)).$$

[1] In this paper, non-quantified variables are assumed universally quantified at the outermost level.

2.1 Two Key Results: Regression & Satisfiability

A fundamental task in the situation calculus is the ***projection task***: given a BAT $\mathcal{D}$ and a query formula φ uniform in $do([A_1, \ldots, A_n], s)$, check whether $\mathcal{D} \models \varphi$ holds.

Reiter [1991] proves that the projection task, for a general class of sentences φ—the ***regressable sentences***—can be carried out only using *first-order* theorem proving on the initial situation. Roughly speaking, a sentence W in $\mathcal{L}_{sitcalc}$ is regressable iff each term of sort situation has the form $do([A_1, \ldots, A_n], S_0)$ and each atom of the form $Poss(\alpha, s)$ is such that α has the form $A(t_1, \ldots, t_n)$. For such sentences, a regression operator $\mathcal{R}[\varphi]$ can be defined to transform φ into a formula—the *regression of* φ—that is uniform in S_0 (i.e., it only talks about what is true in the initial situation).

Theorem 1 (Regression Theorem; Reiter, 1991). *If φ is an $\mathcal{L}_{sitcalc}$ regressable query sentence and $\mathcal{D}$ is a BAT, then:*

$$\mathcal{D} \models \varphi \quad \textit{iff} \quad \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[\varphi].$$

This is an important result from a practical standpoint in that second-order axioms in Σ are *not* needed: a query against the initial situation and unique name axioms is sufficient.

The other fundamental result for BATs states that the consistency of a BAT depends, basically, on building a consistent initial situation:

Theorem 2 (Relative Satisfiability; Reiter, 1991, 2001). *A BAT $\mathcal{D}$ is satisfiable iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is satisfiable.*

3 Basic Actions Theories with Free Fluents

In this section, we lift the requirement of having one successor state axiom per fluent in a basic action theory. It turns out that the fundamental results for BATs can be be recovered.

A ***basic action theory with free fluents*** (BAT with free fluents) $\mathcal{D}$ is defined exactly as a basic action theory but where the set $\mathcal{D}_{ssa}$ contains *at most one* successor state axiom per fluent. A ***free fluent*** is one that has no successor state axiom, and hence, as "determining fluents" in the Event Calculus [Shanahan, 1999], they are not subject to the common law of inertia. Intuitively, a free fluent can be used to denote a non-determined truth value in a situation (relational free fluent) or to denote an un-determined object or action value in a situation (functional free fluent). For the sake of readability, we shall annotate fluent symbols with a hat symbol, such as $\hat{F}(\vec{x}, s)$ or $\hat{f}(\vec{x}, s)$, to denote that these are free fluents.

Our first fundamental result states that BATs with free fluents satisfy the Relative Satisfiability condition.

Theorem 3 (Relative Sat.). *A BAT with free fluents $\mathcal{D}_{poss} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \Sigma$ is satisfiable iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is satisfiable.*

The second fundamental result relates to regression. We prove that BATs with free fluents do admit a sound and complete regression operator, which as with standard basic action theories, can be used to solve the projection task by querying the initial situation and unique name axioms only.

Without loss of generality, and for the sake of legibility, we define our regression operator for formulae of a certain shape. A formula φ of $\mathcal{L}_{sitcalc}$ is in ***normal form*** (for regression) iff it satisfies the following syntactic conditions: *(i)* every term

of sort action in φ is of the form either $A(\vec{u})$ or $a(\vec{u})$, where A is a constant of sort action, a is a variable of sort action and every component of tuple $\vec{u}$ is either a constant or a variable of sort object; and *(ii)* every atomic formula in φ mentioning a fluent is of the form either $F(\vec{u}, \sigma)$ or $f(\vec{u}, \sigma) = v$, where F is a relational fluent, f is a functional fluent, σ is a term of sort situation (where every term of sort action satisfies condition (i)), every component of $\vec{u}$ is a variable or a constant of sort object, and v is a variable or a constant of sort object.

It is not difficult to see that, by using some extra variables, every formula φ of $\mathcal{L}_{sitcalc}$ can be transformed into a logically equivalent formula $\mathcal{T}(\varphi)$ that is in normal form. In addition, if the original formula is regressable (see Section above), so will be its normal form equivalent. For instance, the formula

$$\varphi \doteq F(c_1, f(s_1), do(A(c_1, g(c_3, do(B(c_4, h(x, s_2)), s_3))), S_0))$$

is logically equivalent to the normal form formula:

$$\begin{aligned}\mathcal{T}(\varphi) \doteq \exists y_1 \exists y_2 \exists y_3 . [\, & y_1 = f(s_1) \wedge y_2 = h(x, s_2) \wedge \\ & y_3 = g(c_3, do(B(c_4, y_2), s_3)) \wedge \\ & F(c_1, y_1, do(A(c_1, y_3), S_0))].\end{aligned}$$

By assuming formulae are in normal form, a simpler regression operator can be defined, as it is not necessary to roll back fluents in a special order in terms of prime functional fluents [Reiter, 2001, Definition 4.7.1].

Next, given a *regressable formula φ in normal form*, the ***regression operator $\mathcal{R}$ over*** φ, denoted by $\mathcal{R}[\varphi]$, is inductively defined following [Reiter, 2001, Definition 4.7.4]. Concretely, assume first that φ is an atomic formula:

- If φ does not mention any fluent (i.e., it is a rigid atom), *only mentions free fluents*, or is of the form $F(\vec{u}, S_0)$ or $f(\vec{u}, S_0) = v$, then $\mathcal{R}[\varphi] = \varphi$.
- If φ is of the form $F(\vec{u}, do(\alpha, \sigma))$, where F is a non-free relational fluent with successor state axiom $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, then assuming that all quantified variables in $\Phi_F(\vec{x}, a, s)$ are renamed to be distinct from the variables in $F(\vec{u}, do(\alpha, \sigma))$, we have that $\mathcal{R}[\varphi] = \mathcal{R}[\mathcal{T}(\Phi_F(\vec{u}, \alpha, \sigma))]$.
- If φ is of the form $f(\vec{u}, do(\alpha, \sigma)) = v$, where f is a non-free functional fluent with successor state axiom $f(\vec{x}, do(a, s)) = y \equiv \Phi_f(\vec{x}, y, a, s)$, then assuming that all quantified variables in $\Phi_f(\vec{x}, y, a, s)$ are renamed to be fresh, we have that $\mathcal{R}[\varphi] = \mathcal{R}[\mathcal{T}(\Phi_f(\vec{u}, v, \alpha, \sigma))]$.
- If φ is of the form $Poss(A(\vec{u}), \sigma)$, where the precondition axiom for A is $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$, then assuming that all quantified variables of $\Pi_A(\vec{x}, s)$ are renamed to be fresh, then $\mathcal{R}[\varphi] = \mathcal{R}[\mathcal{T}(\Pi_A(\vec{u}, \sigma))]$.

For non-atomic formulas, $\mathcal{R}[\varphi]$ is defined as usual: (1) if $\varphi = \neg\psi$, then $\mathcal{R}[\varphi] = \neg\mathcal{R}[\psi]$; (2) if $\varphi = \psi_1 \wedge \psi_2$, then $\mathcal{R}[\varphi] = \mathcal{R}[\psi_1] \wedge \mathcal{R}[\psi_2]$; and (3) if $\varphi = \exists x\, \psi$, then $\mathcal{R}[\varphi] = \exists x. \mathcal{R}[\psi]$.

As with standard BATs, the regression operator yields a formula whose validity can be checked by considering only the initial state and the unique name axioms for actions.

Theorem 4 (Regression). *Suppose φ is a regressable sentence of $\mathcal{L}_{sitcalc}$ and $\mathcal{D}$ is a BAT with free fluents. Then:*

$$\mathcal{D} \models \varphi \quad \textit{iff} \quad \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[\varphi].$$

It is important to note that the formula $\mathcal{R}[\varphi]$ may *not* be uniform in S_0, as free fluents are not "rolled back" by $\mathcal{R}$ and hence will be left with a situation term in their last argument different from S_0. Nonetheless, because such fluents have no constraint whatsoever, the special interpretation of situations given by the foundational axioms in Σ is irrelevant. Thus, the non-uniform formula that is left can still be checked against $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ (with no special interpretation of situations).

4 Non-Deterministic Actions

An important feature of dynamic systems that our BATs can represent is actions with non-deterministic effects. A standard approach to modeling such actions is to associate an action with a set of effects. For example, the action of rolling a dice can be captured with a combination of a successor state axiom and a precondition axiom as follows:

$$\begin{aligned}& \textit{diceNo}(d, do(a, s)) = n \equiv \\ & \quad (a = \text{ROLL}(d) \wedge \hat{v}(d, s) = n) \vee \\ & \quad\quad (\textit{diceNo}(d, s) = n \wedge a \neq \text{ROLL}(d)); \\ & \textit{Poss}(\text{ROLL}(d), s) \equiv [\hat{v}(d, s) \geq 1 \wedge \hat{v}(d, s) \leq 6)].\end{aligned}$$

The SSA states that the dice's top number is equal to the value of $\hat{v}(d, s)$, a free fluent, while the precondition axiom, following Bacchus *et al.* [1995]'s idea for modeling noisy sensors, constrains the value of $\hat{v}(d, s)$ to be between 1 and 6 (otherwise, the action of rolling the action is not legally executable). Note that the free fluent $\hat{v}(d, s)$ has d as an argument, allowing a possibly different assignment to each dice.

Then, to verify whether a certain property Q holds in a ground situation S, we check whether $\mathcal{D} \models \textit{Legal}(S) \supset Q(S)$. Observe that this disregards models where S is not legal. Thus, by exploiting our regression result (Theorem 4) it is possible to prove that $\mathcal{D} \models \textit{Legal}(do(\text{ROLL}(d_1), S)) \supset \bigvee_{i=1}^{6} \textit{diceNo}(d_1, do(\text{ROLL}(d_1), S)) = i$ whereas, for any $k \notin \{1, \ldots, 6\}$, $\mathcal{D} \models \textit{Legal}(do(\text{ROLL}(d_1), S)) \supset \neg(\textit{diceNo}(d_1, do(\text{ROLL}(d_1), S)) = k)$.

More complex situations that involve arbitrary non-deterministic effects can also be accommodated in our BATs. Consider for example the *Liar's dice* game, popular in South America, in which players place a number of dices (typically 5) in a opaque cup. In one shot, all players roll the dice by turning their cups upside down onto the table. The game goes from here, but we just focus on how we can model the turning of all cups with a *single* non-deterministic action, TURNALL.

We use fluent $\textit{in}(d, c, s)$ to state that the dice d is in cup c in situation s. Now the SSA for fluent *diceNo* is as follows:

$$\begin{aligned}& \textit{diceNo}(d, do(a, s)) = n \equiv \\ & \quad (a = \text{TURNALL} \wedge \exists c\, \textit{in}(d, c, s) \wedge \hat{v}(d, s) = n) \vee \\ & \quad (\textit{diceNo}(d, s) = n \wedge (a \neq \text{TURNALL} \vee \neg\exists c\, \textit{in}(d, c, s))).\end{aligned}$$

To restrict the range of $\hat{v}$ to legal dice values, we use the precondition axiom:

$$\begin{aligned}& \textit{Poss}(\text{TURNALL}, s) \equiv \\ & \quad \forall d\, [(\exists c\, \textit{in}(d, c, s)) \rightarrow (\hat{v}(d, s) \geq 1 \wedge \hat{v}(d, s) \leq 6)].\end{aligned}$$

Note how using a free fluent, together with a combination of successor state and precondition axioms as above, allows

us to model a rolling dice action that affects an arbitrary number of dices at the same time. It is not difficult to see how those axioms would look like for an action $\text{TURNCUP}(c)$ to roll only those dices in cup c.

5 Databases with Incomplete Information

A fundamental result about BATs shows they can be used to formalize the evolution of a database under a sequence of updates [Reiter, 1995]. For example, consider a relation *Teaches*(*did*, *pid*, *cid*) in a university database that stores, for every department with identifier *did*, the identifier *pid* of a professor teaching the course with identifier *cid*. A standard way to update this relation is by executing actions $\text{ADD}(did, pid, cid)$ and $\text{DEL}(did, pid, cid)$, which add and delete a tuple from the relation *Teaches*, respectively. This naturally gives rise to the following SSA for *Teaches*:

$$\begin{aligned} &\mathit{Teaches}(did, pid, cid, do(a, s)) \equiv \\ &\quad a = \text{ADD}(did, pid, cid) \vee \\ &\quad (\mathit{Teaches}(did, pid, cid, s) \wedge a \neq \text{DEL}(did, pid, cid)). \end{aligned}$$

Nonetheless, the formalization proposed by Reiter [1995] is not amenable to handling databases with *null values*. For example, we may need to record that a professor teaches a course for a specific department without knowing the identifier of the course. To do so, we would like an action $\text{ADD\_PD}(did, pid)$ to add the fact that a professor with identifier *pid* teaches in a department with identifier *did*. Note that such an update *is* possible in a relational database, and will generate a tuple with a *null* value in the cell for the course identifier [Ramakrishnan and Gehrke, 2002]. Unfortunately within Reiter's BATs, any SSA for *Teaches* would specify every single value of a new tuple added by any action. Indeed, while the effects of actions $\text{ADD}(did, pid, cid)$ and $\text{DEL}(did, pid, cid)$ can be stated by the *effect axioms*:

$$\begin{aligned} a = \text{ADD}(did, pid, cid) &\supset \mathit{Teaches}(did, pid, cid, do(a, s)), \\ a = \text{DEL}(did, pid, cid) &\supset \neg\mathit{Teaches}(did, pid, cid, do(a, s)), \end{aligned}$$

one we be tempted to write an effect axiom such as:

$$a = \text{ADD\_PD}(did, pid) \supset \exists x\, \mathit{Teaches}(did, pid, x, do(a, s)), \quad (1)$$

which expresses that $\text{ADD\_PD}(did, pid)$ adds a tuple with some (unknown) value. Unfortunately, this axiom is not in the form allowed by Reiter [1995], which does not admit existential quantification in the consequent of the implication.

At this point, the notion of free fluent comes to the rescue. Indeed, the generation of an "unknown" value in axiom (1) can be naturally modeled by using a functional free fluent $\hat{f}$:

$$a = \text{ADD\_PD}(did, pid) \wedge cid = \hat{f}(s) \supset \mathit{Teaches}(did, pid, cid, do(a, s)).$$

Note that $\hat{f}$ should not have a successor state axiom since the value of *cid*, which is equal to $\hat{f}(s)$, should not be determined when the action $\text{ADD\_PD}(did, pid)$ is executed. In addition, observe that $\hat{f}$ has to be a fluent (i.e., with its last argument of sort situation), otherwise after multiple executions of action $\text{ADD\_PD}(did, pid)$ professor *pid* would wind up teaching the same course in department *did*. Taking into account the new effect axiom, we obtain the following SSA for *Teaches*:

$$\begin{aligned} &\mathit{Teaches}(did, pid, cid, do(a, s)) \equiv \\ &\quad a = \text{ADD}(did, pid, cid) \vee \\ &\quad (a = \text{ADD\_PD}(did, pid) \wedge cid = \hat{f}(s)) \vee \\ &\quad (\mathit{Teaches}(did, pid, cid, s) \wedge a \neq \text{DEL}(did, pid, cid)). \end{aligned}$$

Fluent $\hat{f}$ above only has s as an argument in this example, since the action $\text{ADD\_PD}(did, pid)$ is meant to add just one tuple in a specific situation. However, an action can add several tuples in the same situation, in which case we will need to consider free functional fluents with extra arguments. For example, assume our database contains a relation *Aff*(*pid*, *did*), which indicates that the professor with identifier *pid* is affiliated to the department with identifier *did*. Moreover, assume that the execution of an action $\text{ADD\_AFF}(did)$ has as effect that every professor affiliated with department *did* teaches a course in such a department. Since the specific course is not specified when executing $\text{ADD\_AFF}(did)$, the effect of this action is modeled by using a functional free fluent $\hat{g}$:

$$\begin{aligned} a = \text{ADD\_AFF}(did) \wedge \mathit{Aff}(pid, did) \wedge \\ cid = \hat{g}(pid, s) \supset \mathit{Teaches}(did, pid, cid, do(a, s)). \end{aligned}$$

Notice that in this case *pid* cannot be removed as an argument of $\hat{g}$, as otherwise all the professors affiliated with the department *did* will end up teaching the same course as the result of executing $\text{ADD\_AFF}(did)$.

Null values are widely used in theory and practice to represent missing information in databases. In relational database systems, only one symbol `NULL` is used to represent the presence of a null value, without assuming that nulls in different positions are equal. A way to formalize this semantics is by using different symbols for null values in different positions, which is what we are doing with a successor state axiom like the one before for *Teaches*. But free fluents allow us to go further, towards modeling more expressive database formalisms with incomplete information. For example, if we assume that relation *Teaches*(*did*, *pid*, *pname*, *cid*) also stores the name *pname* of professor with identifier *pid*, then when modeling the effect of action $\text{ADD\_PD}(did, pid)$, we do not expect the name of the professor *pid* to change every time this action is executed. In fact, we expect this name to be the same over time even if we do not have it in the database. Thus, we would like to associate only one null value for professor *pid* that corresponds to his/her name, and which can be used in several different positions in the database. Such a database is called a Naïve table [Imielinski and Jr., 1984; Abiteboul *et al.*, 1995], and its evolution can be formalized by using the following effect axioms:

$$\begin{aligned} a = \text{ADD\_PD}(did, pid) \wedge cid = \hat{f}(s) \wedge \\ \exists x \exists y\, \mathit{Teaches}(x, pid, pname, y, s) \supset \\ \mathit{Teaches}(did, pid, pname, cid, do(a, s)), \end{aligned}$$

$$a = \text{ADD\_PD}(did, pid) \wedge \neg\exists x \exists y\, Teaches(x, pid, pname, y, s) \wedge cid = \hat{f}(s) \wedge pname = \hat{h}(s) \supset Teaches(did, pid, pname, cid, do(a, s)).$$

Note that when $\text{ADD\_PD}(did, pid)$ is executed, if we find a name for professor *pid* in table *Teaches*, then this name is reused, even if it is a null value. Otherwise, the (unknown) name of the professor is created by using free fluent $\hat{h}$.

We can prove that our framework can specify update policies proposed for databases with incomplete information [Abiteboul and Grahne, 1985]. However, we focus on this section on how free fluents allow the specification of update policies that go beyond. Below, we give two such examples: data anonymization and the specification of general conditions on null values.

Anonymization. Consider the well-known area of *data anonymization* [Lakshmanan *et al.*, 2005; Tao, 2008; Zhou *et al.*, 2008]. In our university database, we would like to have an action to anonymize the information about professor identifiers, professor names and course identifiers in the table *Teaches*. We model this with action ANONYMIZE, that replaces every value by a null, and can be described by:

$$a = \text{ANONYMIZE} \wedge Teaches(did, pid, pname, cid, s) \wedge x = \hat{f}_1(pid, s) \wedge y = \hat{f}_2(pname, s) \wedge z = \hat{f}_3(cid, s) \supset Teaches(did, x, y, z, do(a, s))$$

$$a = \text{ANONYMIZE} \wedge Teaches(did, pid, pname, cid, s) \supset \neg Teaches(did, pid, pname, cid, do(a, s)).$$

The first axiom replaces every occurrence of a professor identifier *pid* by the same unknown value $\hat{f}_1(pid, s)$, and likewise for *pname* and *cid*. The second axiom removes existing tuples from the table. Notice that in this case *pid* cannot be removed as an argument of $\hat{f}_1$, as otherwise all professors will end up having the same identifier as the result of executing ANONYMIZE, and likewise for $\hat{f}_2$ and $\hat{f}_3$.

Even though data has been anonymized, its structure has not been lost. Indeed, immediately after action ANONYMIZE is executed, we can analyze sensitive information without revealing the identities of the entities involved. For example, consider the following query $Q_1(s)$:

$$\exists did_1 \exists did_2 \exists pid \exists pname \exists cid\, (Teaches(did_1, pid, pname, cid, s) \wedge Teaches(did_2, pid, pname, cid, s) \wedge did_1 \neq did_2),$$

which asks whether there exist two different departments where the same professor is teaching the same course. Then we have that the answer to Q_1 in S_0 is true if and only if it is true in $do(\text{ANONYMIZE}, S_0)$; in other words, if $\mathcal{D}$ is the BAT with free fluents of our running example, then have that $\mathcal{D} \models Q_1(S_0)$ if and only if $\mathcal{D} \models Q_1(do(\text{ANONYMIZE}, S_0))$. However, if we consider a more specific query $Q_2(x, y, s)$ that asks for the identifiers and names of the professors that are teaching the same course in two different departments

$$\exists did_1 \exists did_2 \exists cid\, (Teaches(did_1, x, y, cid, s) \wedge Teaches(did_2, x, y, cid, s) \wedge did_1 \neq did_2),$$

and we have that $\mathcal{D} \models Q_2(c, d, S_0)$ for a professor with identifier c and name d, then $\mathcal{D} \not\models Q_2(c, d, do(\text{ANONYMIZE}, S_0))$ so the identity of this professor is not revealed.

Conditions on null values. Our formalism can also accommodate updates on Conditional tables [Imielinski and Jr., 1984; Abiteboul *et al.*, 1995], which allow to have global *conditions* on null values and to add or remove tuples according to some conditions. In what follows, we show an example of such updates but considering a more general condition on null values than those in [Imielinski and Jr., 1984; Abiteboul *et al.*, 1995]. Assume that in our running example the salary of professors is stored in a functional fluent $salary(did, pid, s)$. Setting a specific salary value can be done via an action $\text{SET\_SAL}(did, pid, sal)$. Here we go further, showing how to model an action $\text{SET\_SALR}(did, pid, salL, salH)$, that sets the salary of a professor to a value within the range $[salL, salH]$? To represent the "unknown" value of the salary we use the effect axiom:

$$a = \text{SET\_SALR}(did, pid, salL, salH) \wedge sal = \hat{f}(salL, salH, s) \supset salary(did, pid, do(a, s)) = sal,$$

where $\hat{f}(salL, salH, s)$ is a free functional fluent, whose range is restricted by including the following axiom in $\mathcal{D}_{poss}$:

$$Poss(\text{SET\_SALR}(did, pid, salL, salH), s) \equiv [\hat{f}(salL, salH, s) \geq salL \wedge \hat{f}(salL, salH, s) \leq salH)].$$

Now to verify whether a query Q holds after the updates determined by situation S, we check whether $\mathcal{D} \models Legal(S) \supset Q(S)$. Recall that $Legal(s)$ denotes whether situation s is executable, that is, whether the precondition of each action in s holds. For example, consider the query $Q(x, s) = salary(math, p1, s) \geq x$, which checks whether a Math professor with identifier $p1$ has a salary above the threshold x. In addition, consider action $\alpha \doteq \text{SET\_SALR}(cs, p1, 50, 90)$ that updates the professor's salary to some value between \$50K and \$90K. Then, as expected, we obtain that:

$$\mathcal{D} \models Legal(do(\alpha, S_0)) \supset \neg Q(100, do(\alpha, S_0)).$$

This can be verified via regression:

$$\mathcal{R}[Legal(do(\alpha, S_0)) \supset \neg Q(100, do(\alpha, S_0))] = (\hat{f}(50, 90, S_0) \geq 50 \wedge \hat{f}(50, 90, S_0) \leq 90)) \supset \neg(\hat{f}(50, 90, S_0) \geq 100).$$

In fact, we have that (assuming arithmetic is incorporated in classical entailment $\models$):

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models (\hat{f}(50, 90, S_0) \geq 50 \wedge \hat{f}(50, 90, S_0) \leq 90)) \supset \neg(\hat{f}(50, 90, S_0) \geq 100),$$

so $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[Legal(do(\alpha, S_0)) \supset \neg Q(100, do(\alpha, S_0))]$. As mentioned before, the result of the regression only needs to be checked against the initial situation and the set of unique name axioms, even if some free fluents are present. An analogous regression-based query can be performed to conclude that $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[Legal(do(\alpha, S_0)) \supset Q(40, do(\alpha, S_0))]$.

6 Guarded Action Theories via Free Fluents

Guarded action theories [De Giacomo and Levesque, 1999; De Giacomo *et al.*, 2001] (GATs) are an extension of BATs, where successor state axioms are generalized with guard conditions of the form:

$$\alpha(\vec{x}, a, s) \supset [F(\vec{x}, do(a, s)) \equiv \psi(\vec{x}, a, s)], \qquad (2)$$

where $\alpha(\vec{x}, a, s)$ and $\psi(\vec{x}, a, s)$ are uniform in s. Formula $\alpha(\vec{x}, a, s)$ is the *guard* of the axiom denoting the conditions under which the causal law encoded in $\psi(\vec{x}, a, s)$ applies.[2] In a GAT, each fluent can have zero, one, or more ***guarded successor state axioms*** (GSSAs) of the above form.

Guarded theories allow for expressing incomplete knowledge in the causal laws of the domain and accommodate the principle of occlusion [Sandewall, 1994]: in some circumstances, the effect of actions on a fluent are unknown and the common sense *law of inertia* does not apply. For example, if the robot is alone in the building, the state of doors are completely determined by the robot's open and close actions:

$$\begin{aligned} &\mathit{Alone}(s) \supset \\ &\quad \mathit{DoorOpen}(x, do(a, s)) \equiv \\ &\qquad a = \text{OPEN}(x) \vee (\mathit{DoorOpen}(x, s) \wedge a \neq \text{CLOSE}(x)). \end{aligned}$$

Generalized Regression. De Giacomo and Levesque [1999] provided a *generalized* form of regression for GBATs, but one that is "tricky, because of the interaction between the various GSFAs and GSSAs, requiring us to solve (auxiliary) projection tasks at each step." Concretely, the generalized regression requires that the theory *entails the guards* of all used GSSAs: a formula φ can be regressed to a formula ψ in a theory Σ, iff for each fluent $F(\vec{t}, do(\alpha, \sigma))$ mentioned in φ there exists a GSSA as in (2) such that $\Sigma \models \alpha(\vec{t}, \alpha, \sigma)$.

While De Giacomo and Levesque claimed that this type of regression achieves a "sensible compromise between syntactic transformations and logical reasoning," the fact is that, unlike regression for BATs, it relies on logical reasoning to decide which GSSAs to apply.

6.1 Representing GSSAs Using Free Fluents

We show here that, under plausible assumptions on GSSAs, allowing free fluents in BATs is enough to capture GATs. This is significant in that regression with no logical reasoning can also be applied to guarded theories.

So, given a set of $n \geq 0$ GSSAs for $F(\vec{x}, s)$ of the form:

$$\alpha_i(\vec{x}, a, s) \supset [F(\vec{x}, do(a, s)) \equiv \phi_i(\vec{x}, a, s)],$$

with $i \in \{1, \ldots, n\}$, we define its corresponding SSA as:

$$\begin{aligned} &F(\vec{x}, do(a, s)) \equiv \\ &\quad [\bigvee_i \alpha_i(\vec{x}, a, s) \wedge \phi_i(\vec{x}, a, s)] \vee [\bigwedge_i \neg\alpha_i(\vec{x}, a, s) \wedge \hat{H}(\vec{x}, a, s)], \end{aligned}$$

where $\hat{H}(\vec{x}, a, s)$ is a free relational fluent. If a fluent F has no GSSA, then its SSA is just $F(\vec{x}, do(a, s)) \equiv \hat{H}(\vec{x}, a, s)$.

When the GSSAs for a fluent respect a certain coherency among each other, the transformed successor state axiom captures all corresponding GSSAs. Concretely, a set of GSSAs for fluent $F(\vec{x}, s)$ as above is ***coherent*** if for every pair of GSSA $i, j \in \{1, \ldots, n\}$, $\Sigma \cup \mathcal{D}_{una}$ entails

$$\alpha_i(\vec{x}, a, s) \wedge \alpha_j(\vec{x}, a, s) \supset [\phi_i(\vec{x}, a, s) \equiv \phi_j(\vec{x}, a, s)].$$

The following result states that coherent GSSAs can be combined together into a single successor state axiom, such that the original GAT theory and the resulting BAT with free fluents agree on every formula in the original vocabulary.

Theorem 5. *Let $\mathcal{D}$ be a GAT with coherent fluents, and $\mathcal{D}'$ be the BAT with free fluents obtained by replacing all GSSAs of each fluent with the combined SSA as above. Then, for every formula φ in the vocabulary of $\mathcal{D}$: $\mathcal{D} \models \varphi$ iff $\mathcal{D}' \models \varphi$.*

It follows that, for coherent theories, we obtain a *complete* form of regression for guarded theories, that is, we can *always* regress formulas regardless of the truth of the guards. (Note that non-coherent GATs may translate into consistent BATs.)

Let us close the section with an example. Consider the guarded theory with GSSAs:

- $\texttt{True} \supset [G(do(a, s)) \equiv G(s)]$,
- $\texttt{True} \supset [F(do(a, s)) \equiv G(s)]$,
- $G(s) \supset [W(do(a, s)) \equiv \mathit{True}]$,

and an empty $\mathcal{D}_{S_0}$. Then, the formula:

$$\Phi \doteq F(do(\alpha, S_0)) \supset W(do(\alpha, S_0)) \qquad (3)$$

cannot be regressed using generalized regression [De Giacomo and Levesque, 1999], as the theory does not entail $G(S_0)$, which is necessary to regress fluent $W(s)$.

Now, the corresponding theory with free fluents includes SSAs $G(do(a, s)) \equiv G(s)$ and $F(do(a, s)) \equiv G(s)$ for fluents $G(s)$ and $F(s)$, and the following SSA for $W(s)$:

$$W(do(a, s)) \equiv G(s) \vee \neg(G(s) \wedge \hat{H}(a, s)).$$

By using regression for theories with free fluents, as described before, we obtain that formula Φ regresses to

$$G(S_0) \supset [G(S_0) \vee \neg(G(S_0) \wedge \hat{H}(\alpha, S_0))],$$

which turns out to be a tautology. By Theorem 5, this indicates that Φ holds in the original guarded theory.

7 Related Work

Other languages for reasoning about action have also been extended to provide support for non-determinism. The Event Calculus logical language supports various kinds of non-deterministic causal laws [Miller and Shanahan, 2002]. In one of these forms, the so-called *determining fluents* [Shanahan, 1999] are used. Such fluents are analogous to our free fluents: their truth value is not determined initially or by any other causal law, and they are used to condition the effect that becomes active depending on how they are interpreted. In the Event Calculus, undetermined fluents are natural because there are no requirements for successor state axioms for fluents, like in Reiter's BATs.

The $\mathcal{A}$ language [Gelfond and Lifschitz, 1993] has also been extended with nondeterminism by Kartha and Lifschitz [1994]. Rather than using free fluents, Kartha and Lifschitz define a specific operator for specifying incomplete

[2] Standard successor state axioms can be represented using the trivial *True* guard. For simplicity, we do not consider here sensors.

causal laws, and provide a logical account of this operator via logical circumscription.

Dynamic Linear Time Temporal Logic (DLTL) [Giordano *et al.*, 2001] is a language for reasoning about action in which it is required to mention whether or not a fluent f will respect the *law of intertia* after performing an action a. Our coin toss example could be modeled in this framework by explicitly specifying that fluent $Heads$ does not satisfy the law of inertia after performing action $Toss$. While formulae analogous to successor state axioms can be directly specified in DLTL for fluents that are not affected by a non-deterministic action, a potentially large number of frame axioms seem to be needed to specify fluents that are.

In the Situation Calculus, the approaches most related to ours are those by Delgrande and Levesque [2013] (with regression treated in [Belle and Levesque, 2014]) and Pinto *et al.* [2000]. Both provide mechanisms to associate a non-deterministic action with a number of *deterministic* actions. In our dice example, Delgrande and Levesque's axiom:

$$Alt(\text{ROLL}(d), \text{DETROLL}(d, n), 0, s) \equiv 1 \leq n \wedge n \leq 6 \quad (4)$$

would establish that the deterministic action $\text{DETROLL}(d, n)$, which sets the top of dice d to n, is a possible choice of "nature" when an agent executes action $\text{ROLL}(d)$. Similarly, Pinto *et al.* replace regular actions with pairs of the form $\langle \text{ROLL}(d), \text{DETROLL}(d, n) \rangle$ for every possible value of n.

In these two approaches the parameters of deterministic actions arguably *simulate* non-deterministic values (in our example, the dice's top after performing $\text{DETROLL}(d, n)$ becomes equal to parameter n). As such, it is not immediately obvious how to use them to model non-deterministic actions like those in our Liar's Dice and anonymization examples (see Sections 4 and 5, respectively). Intrinsic to both examples is the setting of multiple, unbounded non-deterministic values by performing a single non-deterministic action. For instance, notice that replacing the right-hand side of (4) by $\exists c\, \mathit{in}(d, c, s) \wedge 1 \leq n \wedge n \leq 6$, would not allow to model correctly the action TURNALL in our Liar's Dice example, as we would be indicating that both d and n are possible choices of "nature." A possible, but non-trivial way out for these approaches would be to provide support to associate a non-deterministic action with a sequence of deterministic actions of unbounded length.

Another epistemic related approach in the Situation Calculus is that of Demolombe and Pozos Parra [2000] (studied further by Petrick and Levesque [2002]). Here fluents of the form KF and $K\neg F$ are used to express that "fluent F is known to be true (respectively, false)". This framework allows expressing some non-deterministic actions by modeling the fluents affected by them at the knowledge level. For example in our coin toss example $K\mathit{Heads}$ and $K\neg\mathit{Heads}$ could be used to express that we know the coin is heads-up and tails-up respectively. Queries then need to refer to this particular fluent representation. It is not immediately obvious how null values can be modeled using this approach because, intuitively, knowledge (or lack of knowledge) can only refer to complete tuples.

8 Discussion and Future Work

In this work, we have shown that simply relaxing the "one SSA per fluent" requirement on Situation Calculus BATs [Pirri and Reiter, 1999; Reiter, 2001] allows for convenient specification of various types of incompleteness in causal laws without giving up any of the original properties. Concretely, we studied the "free fluent technique" in BATs, a technique which has arguably been already introduced and used in the Event Calculus [Shanahan, 1999; Miller and Shanahan, 2002] and Sandewall [1994]'s Feature and Fluents reasoning about action formalisms in the '90s. It is then remarkable that such a technique had not been studied yet within the Situation Calculus. We showed this technique has broad applications and that it can simulate some existing extensions of the Situation Calculus and, hence, that it is more general than these. Furthermore, we extended key theorems of the Situation Calculus to free fluents.

It turns out, maybe surprisingly, that free fluents can be compiled away using *static* relations, yielding then standard BATs. Informally, this is because, due to Relative Satisfiability (Theorem 3), the situation argument of free fluents is independent from the foundational axioms:

Theorem 6. *Given a BAT with free fluents $\mathcal{D}$, which allows models with infinite objects, there exists a standard BAT $\mathcal{D}'$ and a rewrite φ' of any formula φ s.t. $\mathcal{D} \models \varphi$ iff $\mathcal{D}' \models \varphi'$.*

The theory $\mathcal{D}'$ above is arguably very convoluted. Among other things, it involves the use of static functions for non-deterministic values, each of which requires an *index* parameter which must be updated (incremented) by each action.

We close by pointing out two options for future work that we are interested on. First, we would like to investigate the use of GOLOG-like high-level programs [Levesque *et al.*, 1997] as a complex mechanism for database update. For example, the non-deterministic program:

$$\delta = \pi \mathit{sal}.(\mathit{sal} \geq \mathit{salL} \wedge \mathit{sal} \leq \mathit{salH}); \text{SET\_SAL}(\mathit{did}, \mathit{pid}, \mathit{sal}),$$

can achieve the update discussed before to set the salary of professor *pid* on department *did* to an unspecified value that falls within the range $[\mathit{salL}, \mathit{salH}]$.

Secondly, a side issue which came up while encoding guarded theories into theories with free fluents is the lack of a principled methodology on how to build guarded successor state axioms from effect axioms, as available for Reiter's standard basic action theories. We believe it is possible to obtain guarded successor state axioms from basic effect axioms, but under a more sophisticated explanation closure assumption than the ones used for basic action theories.

Acknowledgments

We acknowledge the support of the Australian Research Council (DP120100332) and the Millennium Nucleus Center for Semantic Web Research under Grant NC120004. This work originated as part of a Visiting Professorship of the last author from the School of Engineering at Universidad Católica de Chile. We thank the anonymous reviewers for their feedback which was helpful to improve the article final version.

References

[Abiteboul and Grahne, 1985] Serge Abiteboul and Gösta Grahne. Update semantics for incomplete databases. In *VLDB*, pages 1–12, 1985.

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Bacchus *et al.*, 1995] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about noisy sensors in the situation calculus. In *IJCAI*, pages 1933–1940, 1995.

[Belle and Levesque, 2014] Vaishak Belle and Hector J. Levesque. PREGO: an action language for belief-based cognitive robotics in continuous domains. In *AAAI*, pages 989–995, 2014.

[De Giacomo and Levesque, 1999] Giuseppe De Giacomo and Hector J. Levesque. Projection using regression and sensors. In *IJCAI*, pages 160–165, 1999.

[De Giacomo *et al.*, 2001] Giuseppe De Giacomo, Hector J. Levesque, and Sebastian Sardina. Incremental execution of guarded theories. *ACM Transactions on Computational Logic*, 2(4):495–525, October 2001.

[Delgrande and Levesque, 2013] James P. Delgrande and Hector J. Levesque. A formal account of nondeterministic and failed actions. In *IJCAI*, 2013.

[Demolombe and Pozos Parra, 2000] Robert Demolombe and Maria del Pilar Pozos Parra. A simple and tractable extension of situation calculus to epistemic logic. In *ISMIS*, pages 515–524, 2000.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 2-4:301–323, 1993.

[Giordano *et al.*, 2001] Laura Giordano, Alberto Martelli, and Camilla Schwind. Reasoning about actions in dynamic linear time temporal logic. *Logic Journal of the IGPL*, 9(2):273–288, 2001.

[Imielinski and Jr., 1984] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

[Kartha and Lifschitz, 1994] Neelakantan G. Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *KR*, pages 341–350, 1994.

[Lakshmanan *et al.*, 2005] Laks V. S. Lakshmanan, Raymond T. Ng, and Ganesh Ramesh. To do or not to do: The dilemma of disclosing anonymized data. In *SIGMOD*, pages 61–72, 2005.

[Levesque *et al.*, 1997] Hector J. Levesque, Ray Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

[McIlraith, 2000] Sheila A. McIlraith. Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence*, 116(1-2):87–121, 2000.

[Miller and Shanahan, 2002] Rob Miller and Murray Shanahan. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *LNCS*, pages 452–490. Springer, 2002.

[Petrick and Levesque, 2002] Ron Petrick and Hector J. Levesque. Knowledge equivalence in combined action theories. In *KR*, Toulouse, France, April 2002.

[Pinto *et al.*, 2000] Javier Pinto, Amílcar Sernadas, Cristina Sernadas, and Paulo Mateus. Non-determinism and uncertainty in the situation calculus. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 8(2):127–150, 2000.

[Pinto, 1994] Javier A. Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1994.

[Pinto, 1999] Javier Pinto. Compiling ramification constraints into effect axioms. *Computational Intelligence*, 15:280–307, 1999.

[Pirri and Reiter, 1999] Fiora Pirri and Ray Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):261–325, 1999.

[Ramakrishnan and Gehrke, 2002] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, third edition, 2002.

[Reiter, 1991] Ray Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *AI and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Reiter, 1995] Ray Reiter. On specifying database updates. *Journal of Logic Programming*, 25(1):53–91, 1995.

[Reiter, 1996] Ray Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *KR*, pages 2–13, 1996.

[Reiter, 2001] Ray Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems*. Oxford U. Press, 1994.

[Shanahan, 1997] Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. The MIT Press, 1997.

[Shanahan, 1999] Murray Shanahan. The event calculus explained. In *Artificial Intelligence Today*, volume 1600 of *LNCS*, pages 409–430. Springer, 1999.

[Tao, 2008] Yufei Tao. Privacy preserving publication: Anonymization frameworks and principles. In *Handbook of Database Security - Applications and Trends*, pages 489–508. 2008.

[Zhou *et al.*, 2008] Bin Zhou, Jian Pei, and Wo-Shun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations*, 10(2):12–22, 2008.