Transforming Situation Calculus Action Theories for Optimised Reasoning

Christopher Ewin
National ICT Australia and
Computing & Information Systems
The University of Melbourne
chris.ewin@nicta.com.au

Adrian R. Pearce
National ICT Australia and
Computing & Information Systems
The University of Melbourne
adrianrp@unimelb.edu.au

Stavros Vassos
DIAG
Sapienza University of Rome
Rome, Italy
vassos@dis.uniroma1.it

Abstract

Among the most frequent reasoning tasks in the situation calculus are projection queries that query the truth of conditions in a future state of affairs. However, in long running action sequences solving the projection problem is complex. The main contribution of this work is a new technique which allows the length of the action sequences to be reduced by reordering independent actions and removing dominated actions; maintaining semantic equivalence with respect to the original action theory. This transformation allows for the removal of actions that are problematic with respect to progression, allowing for periodical update of the action theory to reflect the current state of affairs. We provide the logical framework for the general case and give specific methods for two important classes of action theories. The work provides the basis for handling more expressive cases, such as the reordering of sensing actions in order to delay progression, and forms an important step towards facilitating ongoing planning and reasoning by long-running agents. It provides a mechanism for minimising the need for keeping the action history while appealing to both regression and progression.

Introduction

In this paper we are concerned with reasoning about action and change in the general case where information about the environment is represented as a first-order logical theory. In practical settings two of the most typical reasoning tasks required are: performing *projection* queries that query the truth of conditions in a future state of affairs; and *updating* or *progressing* the world representation to reflect the current state of affairs after a sequence of actions has been performed.

To illustrate optimised reasoning for these tasks we focus on the logical language of the situation calculus (McCarthy and Hayes 1969; Reiter 2001), where the environment is formalized as an action theory including a set of first-order sentences for specifying the initial state of the environment and a set of first-order axioms for specifying the effects of named actions with respect to the changing properties of the environment. The first task amounts to the question of whether a first-order sentence corresponding to the projection query is entailed by the action theory, while the second involves finding a new updated first-order representation of the world which is typically referred to as progression.

These two tasks are very much connected as a proper solution for progression ensures that the truth value of all possible projection queries is preserved in the updated representation of the environment. However, solving the progression problem in the general case is quite tricky, in fact requiring the use of second-order axioms (Lin and Reiter 1997; Vassos and Levesque 2008; 2013) for the updated representation. The identified problematic cases that require second-order logic are mathematically involved and are rarely encountered, occurring infrequently in practice.

Consequently, syntactically restricted forms of action theories have been investigated in order to ensure that a first-order progression exists and can be effectively computed. Examples for first-order progressable action theories include those with *local-effect actions* (Liu and Levesque 2005; Vassos, Lakemeyer, and Levesque 2008) which restrict actions to affect only ground atoms that are composed exclusively by using arguments of the action. A type of first-order 'surgery' can be performed to the initial theory, removing the truth values of (finitely many) ground atoms—i.e., by *forgetting* in the sense of (Lin and Reiter 1994)—and setting their new values accordingly. A map of progressable action theory classes can be found in (Vassos and Patrizi 2013), however, they are quite limited and a wide range of cases do not fit in the identified classes—for which we have no means of performing a (logically correct) first-order update.

In practice this means that if the desired action theory includes even a single action that does not fit into the identified progressable cases, then updating of the action theory cannot be guaranteed: from the time that a problematic action α_1 is executed no more updating can be performed. Of course, projection queries can still be decided by the original action theory by forming appropriate entailment queries of the form: 'Will ϕ be true after actions $\langle \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m \rangle$ are performed in the initial state of the environment?', where $\alpha_1, \ldots, \alpha_n$ are the actions that have been already executed (for which the action theory unfortunately cannot be updated due to α_1) and $\beta_1, \ldots, \beta_m$ are the ones with respect to which we want to project in the future.

Although *regression* can be used to answer history-based queries, for many practical cases where the action theory is used by long-living agents in order to reason about their actions, such approaches become infeasible as the history of actions constantly increases.

An important observation is that similar to action α_1, that makes further updates impossible, there can be another action, α_k, that brings the environment to a state of affairs that is easier to represent and reason about. For example, α_k could be a *reseting* action that brings the environment to the initial state of affairs, or one that essentially *subsumes* the effects of α_1 in a way that the new representation is first-order progressable—via setting all problematic atoms to a fixed or sensed truth value. Current approaches do not consider this type of reasoning with respect to the action history, leaving out a wide range of practical cases that can be handled by such a hybrid approach to progression.

We therefore investigate sufficient and necessary conditions for transforming and simplifying the history of actions in a logically correct way by combining two basic types of operations: *swapping independent actions* in the action history; and *eliminating dominated actions* by removing them entirely from the action history. This transformation will often allow to remove actions that are problematic with respect to progression, thus allowing to periodically update the action theory to reflect the current state of affairs.

We provide a mechanism for minimising the need for keeping the action history $\langle \alpha_1, \ldots, \alpha_n \rangle$ of the *actions that have already been executed*, and having a representation of the environment that is *as much up-to-date as possible*. In case reasoning over the past history of events is also required then a separate mechanism that keeps track of the true action history and the original action theory can be employed. Importantly, our proposed mechanism can also be used to reason about and plan for the actions needed in order for a progression to become feasible when the length of the history of actions reaches a certain limit.

Finally, our approach can be also useful in cases where all actions fall in the first-order progressable classes—for the purpose of obtaining computational benefits. Since the updating of the action theory is typically a much more computationally demanding procedure than that of projection, it can be beneficial to keep a history of executed actions up to some fixed (past horizon) length anticipating that a future action may greatly simplify the history. Depending on the characteristics of the domain and the projection queries that are needed, such an approach has the potential to save a lot of computational resources. For instance, in cases where there is a large degree of uncertainty in the form of incomplete or disjunctive information which would, under normal circumstances, lead to an exponential growth of the progressed theory with respect to the number of actions executed, a later sensing action that settles uncertainty would lead to larger computational savings, as compared to those consumed in order to perform projection queries using the longer history of executed actions.

Our contribution is a framework for specifying transformations in a logically correct way in the general case. A promising new class of global-effect actions is introduced, termed *resetting* actions, which facilitate transformation based on dominance principles. We specify practical procedures for realizing the proposed transformations, allowing for swapping and elimination of actions for handling non-progressable instances; or the interleaving of regression and progression as required. We illustrate how such theories correspond to common agent planning tasks, using a slightly modified `Sokoban` domain.

Formal preliminaries

The logical language of situation calculus

The language $\mathcal{L}$ of the situation calculus as presented in (Reiter 2001) is a three-sorted first-order logic language with equality and some limited second-order features. The sorts are: *action*, *situation*, and a catch-all sort *object* for everything else depending on the domain of application.

Similar to a normal one-sorted first-order language, $\mathcal{L}$ includes function and predicate symbols. In this case since there are three sorts, each of the symbols has a type that specifies the sorts for the arguments it takes. The situation calculus includes symbols only of certain types each of which has a special role in the representation of the world and its dynamics.

An action term or simply an *action* represents an atomic action that may be performed in the world. For example consider the action $move(l_1, l_2)$ that may be used to represent that a robot moves from location l_1 to location l_2. A situation term or simply a *situation* represents a world history as a sequence of actions. The constant S_0 is used to denote the *initial situation* where no actions have occurred. Sequences of actions are built using the function symbol *do*, such that $do(\alpha, \sigma)$ represents the successor situation resulting from performing action α in situation σ.

A *relational fluent* is a predicate whose last argument is a situation, and thus whose truth value can change from situation to situation. For example, $RobotAt(l, \sigma)$ may be used to represent that the robot lies at location l in situation σ. In order to simplify the analysis we have restricted the language $\mathcal{L}$ so that there are no functional fluent symbols in $\mathcal{L}$, that is, functions whose last argument is a situation. This is not a restriction on the expressiveness of $\mathcal{L}$ as functional fluents can be represented by relational fluents with a few extra axioms.

Actions need not be executable in all situations, and the predicate atom $Poss(\alpha, \sigma)$ states that action α is executable in situation σ. For example, $Poss(move(l_1, l_2), \sigma)$ is intended to represent that the action $move(l_1, l_2)$ can be executed in situation σ. The language $\mathcal{L}$ also includes the binary predicate symbol $\sqsubset$ which provides an ordering on situations. The atom $s \sqsubset s'$ means that the action sequence s' can be obtained from the sequence s by performing one or more actions in s. We will typically use the notation $\sigma \sqsubseteq \sigma'$ as a macro for $\sigma \sqsubset \sigma' \vee \sigma = \sigma'$.

In this paper, we shall restrict our attention to a language $\mathcal{L}$ with a finite number of relational fluent symbols that only take arguments of sort object (apart their last situation argument), an infinite number of constant symbols of sort object $\mathcal{C} = \{c_1, c_2, \ldots\}$, and a finite number of function symbols of sort action that take arguments of sort object. We adopt the following notation with subscripts and superscripts: α and a for terms and variables of sort action; σ and s for terms and variables of sort situation; t and x, y, z, w for terms and variables of sort object. Also, we use A for action function symbols, F, G for fluent symbols, and b, c, d, e for constants

of sort object. Finally, we will typically write $\phi(\vec{x})$ to state that the free variables of the formula are among $\vec{x}$.

The well-formed first-order formulas of $\mathcal{L}$ are defined inductively similarly to a normal one-sorted language but also respecting that each parameter has a unique sort. As far as the second-order formulas of $\mathcal{L}$ are concerned, only quantification over relations is allowed and the well-formed formulas are defined inductively similarly to a normal second-order language.

Often we will focus on sentences that refer to a particular situation. For this purpose, for any situation term σ, we define the set of *uniform formulas in* σ to be all those (first-order or second-order) formulas in $\mathcal{L}$ that do not mention any other situation terms except for σ, do not mention *Poss*, and where σ is not used by any quantifier (Lin and Reiter 1997). Finally, we use δ to denote a sequence of actions of the form $\langle \alpha_1, \ldots, \alpha_n \rangle$, and $do(\delta, S_0)$ as a shorthand for the situation term $do(\alpha_n, \cdots do(\alpha_1, S_0))$.

Basic action theories

Within the language $\mathcal{L}$, one can formulate action theories that describe how the world changes as the result of the available actions. We focus on a variant of the *basic action theories (BATs)* (Reiter 2001) of the following form:[1]

$$\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_0 \cup \mathcal{D}_{fnd},$$

where:

1. $\mathcal{D}_{ap}$ is the set of action precondition axioms (PAs), one per action symbol A, of the form $Poss(A(\vec{y}), s) \equiv \Pi_A(\vec{y}, s)$, where $\Pi_A(\vec{y}, s)$ is first-order and uniform in s. PAs characterize the conditions under which actions are physically possible.
2. $\mathcal{D}_{ss}$ is the set of successor state axioms (SSAs), one per fluent symbol F, of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, where $\Phi_F(\vec{x}, a, s)$ is first-order and uniform in s. SSAs describe how fluents change between situations as the result of actions.
3. $\mathcal{D}_{una}$ is the set of unique-names axioms for actions.
4. $\mathcal{D}_0$, the *initial knowledge base (KB)*, is a set of first-order sentences uniform in S_0 describing the initial situation S_0.
5. $\mathcal{D}_{fnd}$ is the set of domain independent axioms of the situation calculus, formally defining the legal situations. A second-order induction axiom is included in $\mathcal{D}_{fnd}$.

Probably the most interesting component of a basic action theory is the set successor state axioms, which together encode the dynamics of the domain being represented. Technically, SSAs are meant to capture the effects and non-effects of actions. To achieve that in a parsimonious way, one typically follows the well-known solution to the frame problem by means of successor state axioms of the following form (Reiter 2001):

$$F(\vec{x}, do(a, s)) \equiv \gamma^+_F(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma^-_F(\vec{x}, a, s),$$

where both $\gamma^+_F(\vec{x}, a, s)$ and $\gamma^-_F(\vec{x}, a, s)$ are first-order formulas uniform in s encoding the positive and negative effects, respectively, of action a on fluent F at situation s.

[1] For legibility, we typically omit leading universal quantifiers.

Transforming situation calculus histories

In this section we provide the logical foundations for two basic operations that will allow to transform an action history in a way that is logically correct with respect to a basic action theory $\mathcal{D}$.

Swapping consecutive actions

We start with the logical specification of the notion of consecutive actions being *swappable* in an action history of the form $do(\delta, S_0)$, where δ is a sequence of ground actions. We will introduce two versions of the notion, *just-in-time swappable* and *always swappable*. For the following definitions let $\mathcal{D}$ be a basic action theory of the type we introduced in the previous section, and let δ be $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$, where k is in $\{1, \ldots, n-1\}$, and each of the elements of δ is a ground action term.

Definition 1. We say that actions α_k and α_{k+1} are *just-in-time swappable* in δ wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\mathcal{D} \models \forall \vec{x}.\ F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1} \rangle, S_0)) \equiv F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k \rangle, S_0)).$$

So, we can swap actions α_k and α_{k+1} in δ as long as in all models of $\mathcal{D}$ it happens that the interpretation of all fluent atoms is preserved in the two versions of the action history. The *just-in-time* terminology is used to stress that it may be due to the axioms in this particular instance of $\mathcal{D}_0$ and the action sequence δ that ensure that this condition holds, while in fact actions α_k and α_{k+1} would not necessarily be swappable if another $\mathcal{D}_0$ and δ were assumed.

Example 1. Consider the case of fluent $F(s)$ that is only affected by actions set_F and $unset_F$ such that set_F has a conditional effect that makes $F(s)$ true when $G(s)$ holds, and $unset_F$ has the conditional effect that makes $F(s)$ false when $G(s)$ holds. The SSA for $F(s)$ is then as follows:

$$F(do(a, s)) \equiv (a = set_F \wedge G(s)) \vee F(s) \wedge \neg(a = unset_F \wedge G(s)).$$

Consider now an action sequence δ as above in which action α_k is set_F and α_{k+1} is $unset_F$, assuming there are also other fluents and actions in the language. By looking at the SSA for $F(s)$ we can conclude that these actions may have "conflicting" effects and therefore they should not be considered swappable in general, as the order of applying these two actions may result to a different truth value for $F(s)$.

Nonetheless, if we also take into account $\mathcal{D}_0$ and the particular course of action that takes us to the situation where α_k and α_{k+1} are performed, we may conclude that we can actually swap the actions. For instance, consider a $\mathcal{D}_0$ that includes the fact $\neg G(S_0)$, and the following SSA for $G(s)$:

$$G(do(a, s)) \equiv a = set_G \vee G(s) \wedge \neg(a = unset_G).$$

In the case that none of the actions $\alpha_1, \ldots, \alpha_{k-1}$ is set_G then condition $G(s)$ is in fact not satisfied in the situation when α_k is performed, and also in the resulting situation in which α_{k+1} is performed. As a result, actions α_k and α_{k+1}

have no effect in the truth value of $F(s)$, and actually no effects whatsoever in δ in every model of $\mathcal{D}$. For the same reason, reversing the order of these two actions also results in no effects in the truth value of any fluent atom, and we can conclude that they can be safely swapped, or more precisely that α_k and α_{k+1} are just-in-time swappable in δ wrt $\mathcal{D}$.[2]

Note then that if none of the actions $\alpha_1, \ldots, \alpha_{k-1}$ is $unset_G$ and exactly one is set_G then condition $G(s)$ is satisfied in the situation when α_k is performed, and also in the resulting situation in which α_{k+1} is performed. As a result, actions α_k and α_{k+1} have conflicting effects and are not just-in-time swappable in δ wrt $\mathcal{D}$.

Definition 1 is a precise characterization of when two consecutive actions may be swapped in a particular action sequence without affecting the interpretations of fluents. In order to check this in general one needs to look into the properties of the situation where the actions are performed and verify that no conflict arises. As we saw in Example 1 the same actions for the same $\mathcal{D}$ may be just-in-time swappable in one sequence of actions and not in another.

There can be also cases of actions that the actual situation in which they are performed does not play any role as for example actions that may only affect different fluents. We can capture this notion which we call *always swappable* with a similar definition that removes the dependency on the particular $\mathcal{D}_0$ the action sequence δ as follows.

Definition 2. We say that actions α and α' are *always swappable* wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}.\, F(\vec{x}, do(\langle \alpha, \alpha' \rangle, S_0)) \equiv F(\vec{x}, do(\langle \alpha', \alpha \rangle, S_0)).$$

Essentially, Definition 2 requires that starting from any possible extension for all fluents, applying the two actions in question will have the same effect regardless of the order the actions are applied.

Example 2. Actions set_F and $unset_F$ are not always swappable wrt $\mathcal{D}$, and also the same holds for all α, α' such that $\alpha \in \{set_F, unset_F\}$ and $\alpha' \in \{set_G, unset_G\}$. On the other hand, if we assume that $\mathcal{D}$ also includes the following successor state axiom for H, $H(do(a, s)) \equiv a = set_H \vee H(s) \wedge \neg(a = unset_H)$, then for all α, α' such that $\alpha \in \{set_F, unset_F, set_G, unset_G\}$ and $\alpha' \in \{set_H, unset_H\}$, α, α' are always swappable wrt $\mathcal{D}$.

As we will see later, this distinction will become important when we specify algorithmic ways to decide whether consecutive actions are swappable. Next, we show some straightforward results about projection that follow from these definitions. The *simple projection* problem is deciding whether a condition about a particular situation in the future holds (Reiter 2001), and the next theorem shows that swapping actions according to our definitions preserves the result for any extension of the action sequence in the question.

Theorem 1. *Let δ be $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$, where k is in $\{1, \ldots, n-1\}$, and each of the elements of δ is a ground action term. Let $\phi(s)$ be a (first-order or second-order) formula in $\mathcal{L}$ that is uniform in s. If actions α_k and α_{k+1} are always swappable wrt $\mathcal{D}$ or just-in-time swappable in δ wrt $\mathcal{D}$ then the following holds:*

$$\mathcal{D} \models \phi(do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)) \equiv \phi(do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k, \ldots, \alpha_n \rangle \cdot \zeta, S_0)),$$

where ζ is any sequence of ground action terms.

Proof. For the case of just-in-time swappable we work as follows. Let M be an arbitrary model of $\mathcal{D}$. We show by induction on the situation terms σ such that $do(\langle \alpha_1, \ldots, \alpha_{k-1} \rangle, S_0) \sqsubseteq \sigma$, that for all fluents F and all sequences of action terms ζ, $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1} \rangle \cdot \zeta, S_0)$ iff $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k \rangle \cdot \zeta, S_0)$. For the base case we use Definition 1 and the induction step is straightforward. The theorem then follows by induction on the construction of the formulas ϕ that are uniform in s.

For the case of always swappable let M be an arbitrary model of $\mathcal{D}$. Since $\mathcal{D} - \mathcal{D}_0$ admits all possible extensions for the fluents in S_0 it follows that there is a model M' of $\mathcal{D} - \mathcal{D}_0$ with the same domain for all sorts as M such that for all μ and for all fluents F, $M, \mu \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k-1} \rangle, S_0))$ iff $M', \mu \models F(\vec{x}, S_0)$. By Definition 2 then it follows that $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1} \rangle, S_0)$ iff $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k \rangle, S_0)$, from which point we proceed as in the case of just-in-time swappable actions. □

Theorem 1 shows that swapping actions that are just-in-time or always swappable in δ preserves the entailment of any projection query that refers to some particular situation that comes after $do(\delta, S_0)$. This generalizes to more expressive sentences that refer to situations that come after $do(\delta, S_0)$, not necessary being uniform in one situation.

For example, we can show that swapping actions preserves sentences that may also quantify over future situations such as in the set $\mathcal{L}^F_\sigma$ defined in (Vassos and Levesque 2013), which is a form of the so-called *generalized projection task*. For σ being $do(\alpha, S_0)$, an example of a sentence in $\mathcal{L}^F_\sigma$ is the following: $\forall s(do(\alpha, S_0) \sqsubseteq s \supset \psi(s))$, which states that after executing action α in S_0 then $\psi(s)$ remains true always for all the future situations s from that point on.

Theorem 2. *Let $\mathcal{L}^F_\sigma$ be the set of first-order formulas that refer to the future of σ as defined in (Vassos and Levesque 2013) that allows limited quantification over situations, and $\phi(\sigma)$ be a sentence in $\mathcal{L}^F_\sigma$ for σ being $do(\delta, S_0)$. If actions α_k and α_{k+1} are always or just-in-time swappable in δ wrt $\mathcal{D}$ then the following holds:*

$$\mathcal{D} \models \phi(do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)) \equiv \phi(do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k, \ldots, \alpha_n \rangle \cdot \zeta, S_0)),$$

where ζ is any sequence of ground action terms.

Proof. Similar to the proof of Theorem 1 except that we do induction on the construction of the formulas ϕ in $\mathcal{L}^F_\sigma$. □

Also it is easy to show that always swappable implies just-in-time swappable but not the opposite.

[2] In fact they could also be eliminated as we will see later.

Corollary 1. *If actions α_k and α_{k+1} are always swappable wrt $\mathcal{D}$ then for any sequence of actions δ in which they appear consecutively they are just-in-time swappable in δ wrt $\mathcal{D}$, but the opposite is not true.*

We now turn our attention to how swappable actions may be detected in practice. Definitions 1 and 2 provide a semantical account of the requirement that characterizes when two actions can be safely swapped in an action sequence. In particular, as we noted earlier, for just-in-time swappable actions we need to be able to project on the situation where the actions are applied and check the effects of actions conditioned on the action history. On the other hand, the notion of actions being always swappable is decoupled from the action history, allowing us to check for a simpler condition. The following theorem shows an equivalent characterization of always swappable actions in terms of the effects of the actions and their successor state axioms.

Theorem 3. *Actions α and α' are always swappable wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}\{&(\gamma_F^+(\vec{x}, \alpha, S_0) \vee \gamma_F^+(\vec{x}, \alpha', do(\alpha, S_0)) \\ &\equiv \gamma_F^+(\vec{x}, \alpha', S_0) \vee \gamma_F^+(\vec{x}, \alpha, do(\alpha', S_0))) \\ &\wedge (\gamma_F^-(\vec{x}, \alpha, S_0) \vee \gamma_F^-(\vec{x}, \alpha', do(\alpha, S_0)) \\ &\equiv \gamma_F^-(\vec{x}, \alpha', S_0) \vee \gamma_F^-(\vec{x}, \alpha, do(\alpha', S_0)))\} \\ \wedge \neg\exists \vec{x}\{&\gamma_F^+(\vec{x}, \alpha, S_0) \wedge \gamma_F^-(\vec{x}, \alpha', S_0) \\ &\vee \gamma_F^+(\vec{x}, \alpha', S_0) \wedge \gamma_F^-(\vec{x}, \alpha, S_0)\}.\end{aligned}$$

Proof (sketch). Observe that two actions are not always swappable only if in some model the application of one action in S_0 changes the truth value of the effect formulas of the other action or if they have conflicting effects. The first condition is covered by the universally quantified formula that covers all the cases, and the second condition is covered by the negated existentially quantified formula.

In the case of *context-free* actions such that γ_F^+ and γ_F^- do not depend on s, this is simplified as follows.

Corollary 2. *Let $\mathcal{D}$ be a basic action theory such that all successor state axioms are context-free in the sense that they have the following form: $F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a)$. Actions α and α' are always swappable wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \neg\exists \vec{x}\{&\gamma_F^+(\vec{x}, \alpha, S_0) \wedge \gamma_F^-(\vec{x}, \alpha', S_0) \\ &\vee \gamma_F^+(\vec{x}, \alpha', S_0) \wedge \gamma_F^-(\vec{x}, \alpha, S_0)\}.\end{aligned}$$

These results allow us to reduce the semantic notion expressed in Definition 2 into a more concrete one about the effects of actions as expressed in the positive and negative effect formulas γ^+ and γ^- in the successor state axioms of the fluents in $\mathcal{L}$. Even though this condition is still an entailment question about the effects of actions, as we will see later for special cases of successor state axioms we will be able to further reduce it to simpler tests that we can easily evaluate.

Next we proceed to a similar analysis for actions that can be considered redundant and can be safely omitted.

Eliminating actions

Now we turn our attention to simplifying the action history by *eliminating* an action α_k that is *dominated* by actions $\alpha_{k+1}, \ldots, \alpha_n$ in a sequence of ground action terms δ of the form $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$. As in the case of swappable actions, we identify two versions for this notion.

Definition 3. We say that action α_k is *just-in-time dominated* by α_{k+1} in δ wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\begin{aligned}\mathcal{D} \models \forall \vec{x}.\, F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1} \rangle, S_0)) \equiv \\ F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_{k+1} \rangle, S_0)).\end{aligned}$$

Example 3. Consider the fluent $F(x, s)$ with the following successor state axiom:

$$\begin{aligned}F(x, do(a, s)) \equiv\ & a = set_F(x) \vee a = resetTrue_F \wedge G(s) \vee \\ & F(s) \wedge \neg(a = unset_F(x) \vee a = resetFalse_F).\end{aligned}$$

Action $set_F(c)$ sets the truth value of atom $F(c, s)$ to true, $unset_F(x)$ sets it to false, and there are two resetting actions that set all atoms to true and false. Action $resetTrue_F$ is conditional and only has this global effect when $G(s)$ holds, while $resetFalse_F$ is unconditional. Assume that $G(s)$ has the same successor state axiom as in Example 1. Consider the action δ: $\langle unset_G, unset_F(c), set_F(c), resetTrue_F \rangle$. Then, $unset_F(c)$ is just-in-time dominated by $set_F(c)$ in δ wrt to $\mathcal{D}$, but $set_F(c)$ is not just-in-time dominated by $resetTrue_F$ in δ wrt to $\mathcal{D}$.

Similarly to always swappable actions we introduce the notion of *always dominated* actions that removes the dependency on the particular $\mathcal{D}_0$ and the action sequence δ as follows.

Definition 4. We say that action α is *always dominated* by α' wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}.\, F(\vec{x}, do(\langle \alpha, \alpha' \rangle, S_0) \equiv F(\vec{x}, do(\alpha', S_0)).$$

Example 4. Consider the action sequence δ': $\langle set_G, unset_F(c), set_F(c), resetFalse_F \rangle$. Then $set_F(c)$ is always dominated by $resetFalse_F$ in δ wrt to $\mathcal{D}$.

Similar to the analysis of swappable actions, the notions of just-in-time and always dominated actions preserve the entailment of first-order (and second-order) sentences that refer to situations after δ. The next theorem shows this for sentences of the simple projection problem like Theorem 1, and a similar theorem can be obtained for more expressive sentences for the generalized progression problem such as the one addressed in Theorem 2.

Theorem 4. *Let δ be $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$, where k is in $\{1, \ldots, n-1\}$, and each of the elements of δ is a ground action term. Let $\phi(s)$ be a (first-order or second-order) formula in $\mathcal{L}$ that is uniform in s. If action α_k is always dominated by α_{k+1} wrt $\mathcal{D}$ or just-in-time dominated by α_{k+1} in δ wrt $\mathcal{D}$ then the following holds:*

$$\begin{aligned}\mathcal{D} \models \phi(do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)) \equiv \\ \phi(do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)),\end{aligned}$$

where ζ is any vector of ground action terms.

Also it is easy to show that always domination implies just-in-time domination but not the opposite.

Corollary 3. *If action α_k is always dominated by α_{k+1} wrt $\mathcal{D}$ then for any vector of actions δ in which they appear consecutively and α_{k+1} is after α_k, α_k is just-in-time dominated by α_{k+1} in δ wrt $\mathcal{D}$, but the opposite is not true.*

The following theorem shows an equivalent characterization of always domination of actions in terms of the effects of the actions and their successor state axioms.

Theorem 5. *Action α is always dominated by α' wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}\{&\gamma_F^+(\vec{x}, \alpha', S_0) \equiv \gamma_F^+(\vec{x}, \alpha', do(\alpha, S_0)) \\ &\wedge \gamma_F^-(\vec{x}, \alpha', S_0) \equiv \gamma_F^-(\vec{x}, \alpha', do(\alpha, S_0))\} \\ \wedge \forall \vec{x}\{&(\gamma_F^+(\vec{x}, \alpha, S_0) \vee \ \gamma_F^-(\vec{x}, \alpha, S_0)) \\ &\supset (\gamma_F^+(\vec{x}, \alpha', S_0) \vee \gamma_F^-(\vec{x}, \alpha', S_0))\}.\end{aligned}$$

Proof (sketch). Observe that α is not always dominated by α' only if in some model the application of α in S_0 changes the truth value of the effect formulas of α' or if some of the effects of α is not reset by α'.

In the case of *context-free* actions such that γ_F^+ and γ_F^- do not depend on s, this is simplified as follows.

Corollary 4. *Let $\mathcal{D}$ be a basic action theory such that all successor state axioms are context-free in the sense that they have the following form: $F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a)$. Action α is always dominated by α' wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}\{&(\gamma_F^+(\vec{x}, \alpha, S_0) \vee \ \gamma_F^-(\vec{x}, \alpha, S_0)) \\ &\supset (\gamma_F^+(\vec{x}, \alpha', S_0) \vee \gamma_F^-(\vec{x}, \alpha', S_0))\}.\end{aligned}$$

These notions provide the logical specification for transforming and simplifying action histories in a way that preserves the entailment of first-order projection queries. What becomes interesting then is identifying efficient procedures that can identify when sets of actions can be swapped and when an action can be eliminated for certain classes of action theories $\mathcal{D}$.

Action theories with single-value fluents and resetting actions

We now turn our attention to basic action theories with relational fluents that have a function-like behavior in the following sense. For each fluent $F(\vec{y}, s)$, we distinguish the last argument of sort object as the *output* of the fluent and the rest of the arguments of sort object as the *input*. In order to make this explicit we will write $F(\vec{x}, v, s)$ where v is the the output and $\vec{x}$ is input. We will also require that in every model of the action theory for every input $\vec{x}$ there is exactly one object for v (up to equality) such that $F(\vec{x}, v, s)$ holds. We call fluents of this type *single-value fluents*.

Definition 5. *Single-value* fluents are fluents for which the following holds:

$$\mathcal{D} \models \forall \vec{x}, v\{F(\vec{x}, v, s) \supset \neg\exists v'(F(\vec{x}, v', s) \wedge v' \neq v)\}.$$

This is a very common case in many practical domains, such as those involving mobile agents (i.e. a mobile agent can be in only one position in any given situation).[3] The next definition introduces the single-value successor state axioms which preserve this property.

Definition 6. The successor state axiom for $F(\vec{x}, v, s)$ has *resetting-actions* iff $\gamma_F^+(\vec{x}, v, a, s)$ is a disjunction of formulas of the form:

$$\exists \vec{z}(a = A(\vec{y}) \wedge v = y_i \wedge \phi(\vec{y}, s)),$$

and $\gamma_F^-(\vec{x}, v, a, s)$ is a disjunction of formulas of the form:

$$\exists \vec{z}(a = A(\vec{y}) \wedge \neg v = y_i \wedge \phi(\vec{y}, s)),$$

where A is an action symbol, $\vec{y}$ contains $\vec{x}$, $\vec{z}$ corresponds to the remaining variables of $\vec{y}$, y_i is a variable in $\vec{y}$, and $\phi(\vec{y}, s)$ (called a *context formula*) is a first-order formula uniform in s, and γ_F^+,γ_F^- come in pairs in the sense that for each disjunct in γ_F^+ there is one in γ_F^- that is identical except for the atom $v = y_i$ and vice versa. As in the case of functional fluents in (Reiter 2001), we also require the following sentence be entailed by $\mathcal{D}$:

$$\mathcal{D} \models \gamma_F^+(\vec{x}, v, \alpha, s) \supset \neg\exists v'(v' \neq v \wedge \gamma_F^-(\vec{x}, v', \alpha, s)).$$

A basic action theory $\mathcal{D}$ is one of resetting-actions if every SSAs in $\mathcal{D}_{ss}$ is a resetting SSA.

Note that the positive effects of a resetting SSA are in fact local-effect as they have exactly the same form, requiring that any affected atom is built using arguments of action $A(\vec{y})$, while the negative effects are a special form of global effects. Similarly to (Liu and Levesque 2005), the instantiation of a local-effect SSA on a ground action term can be simplified using the unique names axioms for actions as follows.

Lemma 1. *Let α be the ground action term $A(\vec{e})$, where $\vec{e}$ is a vector of constants and suppose that the SSA for F is resetting. Then $\gamma_F^+(\vec{x}, v, \alpha, s)$ is logically equivalent to a formula of the following form:*

$$(\vec{x} = \vec{c}_1 \wedge v = d_1 \wedge \phi_1(s)) \vee \cdots \vee (\vec{x} = \vec{c}_m \wedge v = d_m \wedge \phi_m(s)),$$

where each of the $\vec{c}_1, \ldots, \vec{c}_m$ is a vector of constants contained in $\vec{e}$, and $\phi_1(s), \ldots, \phi_m(s)$ are first-order and uniform in s. Each combination of $\vec{c}_i$, $\vec{d}_i$ must be distinct, and for any $\vec{x} = \vec{c}_i$, at most one $\phi_i(s)$ can hold for any model of s. Similarly, $\gamma_F^-(\vec{x}, \alpha, s)$ is logically equivalent to a formula of the following form:

$$\begin{aligned}&(\vec{x} = \vec{c}_1 \wedge \neg v = d_1 \wedge \phi_1(s)) \vee \cdots \\ &\vee (\vec{x} = \vec{c}_m \wedge \neg v = d_m \wedge \phi_m(s)).\end{aligned}$$

An example follows.

Example 5. Consider a modified instance of the Sokoban problem such that the surface of any cell can be either clean or oily. Consider three actions: $move(x)$, which moves the

[3] We note that a "regular" relational fluent could still be expressed by means of special constants *true* and *false* and with some extra syntactical machinery.

player to position x, $push(b, x, y)$, which pushes block b from position x to position y and $clean(x, t)$, which sets the truth value of x being oily to t. We use three fluents: *PlAt*(x, s), which says the player is in position x in situation s, *BlAt*(b, x, s), which says that block b is in position x in situation s and *Oily*(x, t, s), which says the surface of x contains oil where t is a variable that will take an object representing True or False. There are also appropriate $Poss$ axioms for walls, connectivity, and other necessary restrictions. The successor state axioms for this problem are as follows:

$$\begin{aligned}
&\textit{PlAt}(v, do(a, s)) \equiv \exists x(a = move(x) \wedge v = x)\\
&\quad \vee \exists b, x, y(a = push(b, x, y) \wedge v = x)\\
&\quad \vee \textit{PlAt}(v, s) \wedge \neg(\exists x(a = move(x) \wedge \neg v = x)\\
&\quad \vee \exists b, x, y(a = push(b, x, y) \wedge \neg v = x))\\
&\textit{BlAt}(b, v, do(a, s)) \equiv \exists x, y(a = push(b, x, y) \wedge v = y)\\
&\quad \vee \textit{BlAt}(b, v, s) \wedge \neg\exists x, y(a = push(b, x, y) \wedge \neg v = y)\\
&\textit{Oily}(x, v, do(a, s)) \equiv \exists t(a = clean(x, t) \wedge v = t))\\
&\quad \vee \textit{Oily}(x, v, s) \wedge \neg\exists t(a = clean(x, t) \wedge \neg v = t))
\end{aligned}$$

We begin by introducing the notion of the *effect set* of an action. The idea is to represent a set of situation-suppressed ground fluent atoms that will be affected by an action, as well as the conditions under which the fluent atom will be affected. This is formalised as follows:

Definition 7. Let α_k be a ground resetting action. The *effect set* of α_k is the following set:

$$\begin{aligned}
\Phi_k = \{\phi \supset F(\vec{c}, d) \mid\\
\vec{x} = \vec{c} \wedge \phi \wedge v = d \text{ appears in } \gamma_F^+(\vec{x}, v, \alpha, s)\}.
\end{aligned}$$

Without loss of generality, we assume that the ground versions of the γ^+ and γ^- formulas are simplified according to Lemma 1. The effect sets can then be obtained directly. We observe the following fundamental property of the effect set:

Lemma 2. *Let $F(\vec{x}, v, s)$ be a single-value fluent with positive and negative successor state axioms $\gamma_F^+(\vec{x}, v, \alpha, s)$ and $\gamma_F^-(\vec{x}, v, \alpha, s)$ respectively. Let M be any model of $\mathcal{D}$. Let α_k be a ground resetting action with effect set Φ_k. Then for all $\vec{x} = \vec{c}, v = d$: $M \models \gamma_F^+(\vec{x}, v, \alpha_k, s)$ iff there exists a $\phi \supset F(\vec{c}, d)$ in Φ_k such that $M \models \phi$.*
$M \models \gamma_F^-(\vec{x}, v, \alpha_k, s)$ iff there exists a $\phi \supset F(\vec{c}, e)$ in Φ_k such that $d \neq e$ and $M \models \phi$.

Proof. From the construction of the effect set and the form of the successor state axioms. □

Resetting actions without contexts

We begin by considering a subset of resetting actions where each ϕ_i is $true$. This allows us to reorder or eliminate actions without needing to consider the effects that one action can have on the contexts of the other. We introduce a procedure for determining whether two context-free actions are swappable by directly inspecting their effect sets:

Theorem 6. *Let actions α_1 and α_2 be consecutive resetting actions without contexts with effect sets Φ_1 and Φ_2 respectively. Then α_1 and α_2 are always swappable iff for all elements of Φ_1 of the form $true \supset F(\vec{c}, d_1)$ there is no element of Φ_2 of the form $true \supset F(\vec{c}, d_2)$ such that $d_1 \neq d_2$.*

Proof. From Corollary 2 and Lemma 2. □

While this is a restricted class of actions, it is nonetheless sufficient to cover a number of interesting cases, such as the Sokoban example.

Example 6. Consider the SSAs used in Example 5 and a sample action sequence $\langle move(pos_1), clean(pos_1, F)\rangle$. We first calculate the effect sets for each action: $\Phi_1 = \{true \supset \textit{PlAt}(pos_1)\}$, $\Phi_2 = \{true \supset \textit{Oily}(pos_1, F)\}$. From Theorem 6 we can observe that $move(pos_1)$ and $clean(pos_1, F)$ are always swappable.

We now introduce a procedure for determining whether one resetting action without a context always dominates another by directly inspecting their effect sets.

Theorem 7. *Let actions α_1 and α_2 be consecutive resetting actions without contexts with effect sets Φ_1 and Φ_2 respectively. Then α_2 always dominates α_1 iff for all elements of Φ_1 of the form $true \supset F(\vec{c}, d_1)$ there exists an element of Φ_2 of the form $true \supset F(\vec{c}, d_2)$.*

Proof. From Corollary 4 and Lemma 2 □

Note that determining whether actions can be just-in-time eliminated purely by inspection of the successor state axioms is not possible. This is because of the possibility that some or all of those fluent atoms already held the same value before the action was taken. In the extreme case, for example, that an action occurred twice and identically set all fluent atoms, the second occurrence of that action could be said to be dominated by *any* subsequent action.

Below is an example demonstrating the idea of always-dominating for resetting actions without contexts.

Example 7. Consider the SSAs used in Example 5 and a sample action sequence $\langle move(pos_1), push(b_1, pos_2, pos_3)\rangle$. We first calculate the effect sets for each action: $\Phi_1 = \{true \supset \textit{PlAt}(pos_1)\}$, $\Phi_2 = \{true \supset \textit{PlAt}(pos_2), true \supset \textit{BlAt}(b_1, pos_2)\}$. From Theorem 7 we can observe that $push(b_1, pos_2, pos_3)$ dominates $move(pos_1)$ and thus the action sequence can be simplified to $\langle push(b_1, pos_2, pos_3)\rangle$.

Resetting actions with conjunctive queries as contexts

We now consider a larger class of resetting actions where the context formulas ϕ_i consist of a quantifier-free conjunction of ground fluent atoms. We require that each context formula ϕ_i is satisfiable in the sense that if an element $F(\vec{c}, d)$ is mentioned then neither $\neg F(\vec{c}, d)$ nor $F(\vec{c}, e)$ is mentioned for some $e \neq d$. Reasoning about the properties of these actions is not so straightforward, as determining the value of a context formula for an action requires detailed inspection of the knowledge base. Moreover, the application of one action

can affect the contexts of another. We introduce the following sound but incomplete method for determining whether a pair of conjunctive resetting actions are always swappable:

Theorem 8. *Let actions α_1 and α_2 be consecutive conjunctive resetting actions with effect sets Φ_1 and Φ_2 respectively. Then α_1 and α_2 are always swappable if:*

- *For all elements of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$, there does not exist an element of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$ such that $F_1 = F_2, \vec{c_1} = \vec{c_2}$, $d_1 \neq d_2$, $\phi_1 \wedge \phi_2$ is satisfiable.*
- *For all elements of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$, there does not exist an element of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$ such that ϕ_2 mentions $F_1(\vec{c_1}, e)$ for some e.*
- *For all elements of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$, there does not exist an element of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$ such that ϕ_1 mentions $F_2(\vec{c_2}, e)$ for some e.*

Proof (sketch). The first point above corresponds with the second condition of Theorem 3 and follows from Lemma 2. For the second condition of Theorem 3, we consider a slightly stronger requirement such that $\gamma^+_F(\vec{x}, v, \alpha_1, S_0) \equiv \gamma^+_F(\vec{x}, \alpha_1, do(\alpha_2, S_0))$ and $\gamma^+_F(\vec{x}, v, \alpha_2, S_0) \equiv \gamma^+_F(\vec{x}, \alpha_2, do(\alpha_1, S_0))$ for all $\vec{c}, v$ and similarly for negative SSAs. From Lemma 2, any $F(\vec{c}, d)$ for which there is no $\phi \supset F(\vec{c}, e)$ in Φ_1 for some e cannot affect $\gamma^+_F(\vec{x}, v, \alpha_2, S_0) \equiv \gamma^+_F(\vec{x}, \alpha_2, do(\alpha_1, S_0))$, and similarly for the other conditions.

This result is incomplete due to the possibility that for all models, a pair of context formulas in Φ_1 and Φ_2 referring to the same fluent may be unsatisfiable, or that one may imply the other. We expect such cases to be uncommon in practice, however, as in the majority of cases we expect to reorder actions that do not affect the same fluents. We now consider the complexity of always-swapping for conjunctive queries:

Theorem 9. *The complexity of determining whether two conjunctive resetting actions are always swappable is $O(nm)$ where n is the number of elements in the effect set and m is the number of elements in the context formula ϕ. For resetting actions without contexts, this reduces to $O(n)$.*

Proof. The first item in Theorem 8 requires that for each of the n elements of the effect set, the satisfiability of $\phi_1 \wedge \phi_2$ for any corresponding element in the other set is determined. Since $\phi_1 \wedge \phi_2$ is a conjunctive query, this can be done in $O(nm)$ time. The second and third points require each of the $n \times m$ members of an effect set to be checked for membership in the other effect set, with a complexity of $O(nm)$. For resetting actions without contexts, the second and third points are not required, and the first part does not require a satisfiability check. This reduces to $O(n)$, as the problem is essential checking set membership. □

We also note that the satisfiability check can be omitted while still retaining a sound result. To demonstrate, we consider a version of the Sokoban example used previously, modified to include conjunctive contexts. Imagine now that pushing a box over an oily surface will cause it to move one additional unit further than expected. We modify the formalism as follows:

Example 8. Let the push action now take five arguments, $push(b, x, y, z, t)$ where t is a variable that will take an object representing a truth value. This action will push block b from position x to position y. If the surface of y is oily, the block will be moved to z instead. The successor state axiom for *BlAt* is now as follows:

$$\begin{aligned}
&\textit{BlAt}(b, v, do(a, s)) \equiv \\
&\exists x, y, z(a = push(b, x, y, z, t) \wedge v = y \wedge \neg \textit{Oily}(y, t, s)) \\
\vee &\exists x, y, z(a = push(b, x, y, z, t) \wedge v = z \wedge \textit{Oily}(y, t, s)) \\
\vee &\textit{BlAt}(b, v, s) \wedge \neg\{ \\
&\exists x, y, z(a = push(b, x, y, z, t) \wedge \neg v = y) \wedge \neg \textit{Oily}(y, t, s) \\
\vee &\exists x, y, z(a = push(b, x, y, z, t) \wedge \neg v = z \wedge \textit{Oily}(y, t, s))\}
\end{aligned}$$

Consider now an action sequence $\langle clean(pos_1, F), push(b_1, pos_2, pos_3, pos_4, T)\rangle$. We firstly calculate the effect sets of the two actions: $\Phi_1 = \{true \supset \textit{Oily}(pos_1, F)\}$ $\Phi_2 = \{\neg \textit{Oily}(pos_3, T) \supset \textit{BlAt}(b_1, pos_3), \textit{Oily}(pos_3, T) \supset \textit{BlAt}(b_1, pos_4), true \supset \textit{PlAt}(pos_2)\}$. Now using Theorem 8, we can observe that $clean(pos_1, F)$ and $push(b_1, pos_2, pos_3, pos_4, T)$ are always swappable.

We now propose a method for identifying when one resetting action is always dominated by a subsequent action, and can thus be removed.

Theorem 10. *Let α_1 and α_2 be consecutive conjunctive resetting actions with effect sets Φ_1 and Φ_2 respectively. Let $\phi_1[F(\vec{c})]$ refer to the disjunction of all ϕ_1 for which $\phi_1 \supset F(\vec{c}, d)$ is in Φ_1 for any d and $\phi_2[F(\vec{c})]$ refer to the disjunction of all ϕ_2 for which $\phi_2 \supset F(\vec{c}, d)$ is in Φ_2 for any d. Then α_2 always dominates α_1 iff for all $\vec{c}$, $\phi_1[F(\vec{c})] \supset \phi_2[F(\vec{c})]$ is valid and for all elements of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$, there is no element of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$ such that ϕ_2 mentions $F_1(\vec{c_1}, e)$ for some e and $\phi_1 \wedge \phi_2$ is satisfiable.*

Proof (sketch). Focus firstly on the second part of Theorem 5. From Lemma 2, $\gamma^-_F(\vec{x}, v, \alpha_1, S_0) \vee \gamma^+_F(\vec{x}, v, \alpha_1, S_0)$ iff there exists a $\phi_1 \supset F(\vec{x}, v')$ in Φ_1 such that ϕ_1 is $true$ (similarly for α_2). Now $(\gamma^-_F(\vec{x}, v, \alpha_1, S_0) \vee \gamma^+_F(\vec{x}, v, \alpha_1, S_0)) \supset (\gamma^-_F(\vec{x}, v, \alpha_2, S_0) \vee \gamma^+_F(\vec{x}, v, \alpha_2, S_0))$ is valid iff $\phi_1[F(\vec{c})] \supset \phi_2[F(\vec{c})]$ is valid for all models. For the first part of Theorem 5, for each $\phi_2 \supset F_2(\vec{c_2}, d_2)$ in Φ_2, consider a model, M such that every $F(\vec{x}, y)$ mentioned in ϕ_2 holds with the exception of some $F(\vec{c}, d)$ for which $\phi_1 \supset F(\vec{c}, d)$ appears in Φ_1. Then $\gamma^+_F(\vec{x}, v, \alpha_2, S_0) \not\equiv \gamma^+_F(\vec{x}, v, \alpha_2, do(\alpha_1, S_0))$ for $x = c_2, v = d_2$ iff ϕ_1 is also satisfiable M. The approach is similar for the case of $F(\vec{c}, e)$ in ϕ_1 as well as for the negative effects.

We now consider the complexity of always-dominating for conjunctive queries:

Theorem 11. *The complexity of determining whether one conjunctive-resetting action dominates another is $O(n.2^{mk})$ where n is the number of elements in the effect set, k is the number of times a $\phi \supset F(\vec{c}, d)$ appears in the effect set for fixed $\vec{c}$ but varying d and m is the number of elements in the context formula ϕ. For resetting actions without contexts, this reduces to $O(n)$.*

Proof. The first condition in Theorem 10 requires that for each of the n elements of the effect set, the validity of $\phi_1[F(\vec{c})] \supset \phi_2[F(\vec{c})]$ is determined. Each $\phi[F(\vec{c})]$ contains $m \times k$ terms, the complexity of determining the validity for each formula of this type is thus $O(2^{mk})$. For the second part, each of the $n \times m$ elements of the contexts in Φ_2 must be checked for membership in Φ_1. If found, the satisfiability of $\phi_1 \wedge \phi_2$ must be determined. Since $\phi_1 \wedge \phi_2$ is a conjunctive query, the satisfiability check can be done in $O(m)$ time. The total complexity is thus $O(n.2^{mk}) + O(nm^2)$. For resetting actions without contexts, the second part is not required, and the first part does not require a validity check. This reduces to $O(n)$, as the problem is essential checking set membership. □

We expect such a method to be feasible in practice, as it is exponential only in the size of the context formulas and the number of times a particular $F(\vec{c})$ appears in the effect set. We now show an example of dominating conjunctive resetting actions.

Example 9. Using the successor state axioms developed in Example 8, consider now an action sequence $\langle push(b_1, pos_1, pos_2, pos_3, T),$
$push(b_1, pos_2, pos_3, pos_4, T)\rangle$. We firstly calculate the effect sets of the two actions: $\Phi_1 = \{\neg \mathit{Oily}(pos_2, T) \supset \mathit{BlAt}(b_1, pos_2), \mathit{Oily}(pos_2, T) \supset \mathit{BlAt}(b_1, pos_3), true \supset \mathit{PlAt}(pos_1)\}$ $\Phi_2 = \{\neg \mathit{Oily}(pos_3, T) \supset \mathit{BlAt}(b_1, pos_3), \mathit{Oily}(pos_3, T) \supset \mathit{BlAt}(b_1, pos_4), true \supset \mathit{PlAt}(pos_2)\}$. Now using Theorem 10, $push(b_1, pos_2, pos_3, pos_4, T)$ always dominates $push(b_1, pos_1, pos_2, pos_3, T)$.

Discussion and Related Work

To the best of our knowledge this is the first approach that looks into formalizing operations over an action history that preserves truth of (simple and generalized) projection queries in the final situation. It helps us handle cases which cannot be easily progressed, or actions which can result in a non-progressible theory, by swapping and eliminating actions—achieving transformed theories which can be later progressed. Note that the work does not extend the known classes of progressable theories (Vassos and Patrizi 2013), in contrast the work contributes to a setting where regression and progression can be invoked and interleaved in sensible ways. It is potentially applicable to a wide range of queries, for different action languages.

A similar approach in spirit is nonetheless employed in the *implementations* of reasoning systems that rely on constraints solvers. In particular, for reasoning about action the `Flux` system for implementing fluent calculus theories (Thielscher 2004) uses (among other things) constraints to represent the effects implied to fluents with arithmetic arguments. These constraints specify how the old value of fluents relates to the new one, and are appended to the constraint store as the action history grows. Periodically, `Flux`'s constraint store invokes heuristic techniques for simplifying the history by eliminating redundant constraints.

There are, nonetheless, parallels to work being done in classical planning. In particular (Chrpa, McCluskey, and Osborne 2012) identifies pairs of actions as being either *independent*, *shared*, *nested* or *interleaved*, and goes on to define conditions under which these different types of actions can be removed from a sequence. Working in the classical planning domain, however, allows Chrpa to consider a clear and finite set of effects for each action, which is not the case in the Situation Calculus. Also, in order to realise the modified plan, Chrpa is required to consider the preconditions of each action, whereas by focusing on the projection problem one is able to bypass these, allowing more actions to be dominated. Finally, Chrpa does not consider the possibility of reordering actions.

Symmetry breaking, and the related issue of planning landmarks, have also been shown to be promising for achieving efficiency in planning domains (Domshlak, Katz, and Shleyfman 2013). Interestingly, dominance relations are known to entail the special case of symmetry in constraint programming (Chu, Garcia del la Banda, and Stuckey 2012). Dominance relations might therefore form a basis of a range of constraint optimisation techniques for action languages. This paper follows these insights by realising re-ordering for action histories for a wide range of rich theories.

There are a number of settings where transformation could be effectively used, including *sensing* actions and reasoning about the domination of *sets of actions*. Sensing actions provide new information about the value of fluents and could dominate earlier actions. It may therefore be possible to delay progression, in the event that progression is not possible in one situation due to missing information about certain fluents, until later when sensing actions can learn fluent values which dominate the effects of earlier actions. If we can delay progression until *after* sensing actions are performed, this could facilitate the progression of the action theory at a later time. The dominance of sets of actions over one another could significantly extend the number of cases under which dominance was possible and facilitate new ways of reordering and elimination on set theoretic grounds.

The generality of our approach has the potential to be utilised for wider classes of action theory. For example, the approach could be extended beyond conjunctive queries towards arbitrary context formulas. The extension of the approach to an even wider class of global effect actions, would also be very useful, as global effect theories are known to be particularly problematic with respect to progression.

Conclusions

This work takes the important step of formalizing the transformation of theories of action in order to optimise frequently executed reasoning tasks. The transformation allows action sequences to be shortened, via the swapping and reordering of independent actions and elimination of dominating actions. This can facilitate easier, more optimised, forms of reasoning. Our technique is potentially applicable to a wide range of rich first order theories.

We have identified a new class of action theory which allows for a restricted form of global effects–resetting actions–which falls outside of any known progressable theory of action. It overcomes problems with progression commonly encountered in the agent reasoning tasks illustrated. Our

technique facilitates a new approach for tackling situations falling outside the known classes of progressable theories, and can also be used for optimising the progression of theories known to be progressable. It also enables more efficient entailment queries to be answered by progressing where possible, only performing regressing where necessary.

Acknowledgements

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. The authors acknowledge support of EU Project FP7-ICT 318338 (OPTIQUE) and Sapienza Award 2013 "Spiritlets" project.

References

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012. Determining redundant actions in sequential plans. In *ICTAI*, 484–491.

Chu, G.; Garcia del la Banda, M.; and Stuckey, P. J. 2012. Exploiting subprogem dominance in constraint programming. *Constraints* 1:1–38.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Lin, F., and Reiter, R. 1994. Forget it! In Greiner, R., and Subramanian, D., eds., *Working Notes, AAAI Fall Symposium on Relevance*, 154–159. American Association for Artificial Intelligence.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, 522–527. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence* 4:463–502.

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*.

Thielscher, M. 2004. FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programing* 5(4-5):533–565.

Vassos, S., and Levesque, H. 2008. On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, AAAI'08, 1004–1009. Chicago, Illinois, USA: AAAI Press.

Vassos, S., and Levesque, H. J. 2013. How to progress a database III. *Artificial Intelligence* 195:203–221.

Vassos, S., and Patrizi, F. 2013. A classification of first-order progressable action theories in situation calculus. In *Proceedings of the 23rd international joint conference on Artificial intelligence*, IJCAI'13, 1132–1138.

Vassos, S.; Lakemeyer, G.; and Levesque, H. 2008. First-order strong progression for local-effect basic action theories. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, KR'08, 662–272. AAAI Press.