

Action Theories over Generalized Databases with Equality Constraints (Extended Abstract)

Fabio Patrizi and Stavros Vassos

Department of Computer, Control, and Management Engineering (DIAG)
Sapienza University of Rome
Rome, Italy

Abstract

In this work we study action theories of the situation calculus such that the initial KB is a generalized database with equality constraints (GFDBs). We show that GFDBs characterize the class of definitional KBs and that they are closed under progression. We also show that, under conditions, generalized projection queries can be decided based on an induced transition system and evaluation of local conditions over states.

Introduction

Situation calculus basic action theories (BATs) (Reiter 2001) consist of a first-order knowledge base (KB) which describes the initial state of a given domain, and a set of first-order axioms that specify how the properties of the domain change under the effects of actions. Two important reasoning problems are *projection*, i.e., *predicting* whether a condition would hold if a series of actions were to be performed, and *progression*, i.e., *updating* the KB by a new description that reflects the current state after actions have been performed.

For the general case it has been shown that second-order logic may be required to capture the updated KB (Lin and Reiter 1997; Vassos and Levesque 2013) and an overview of some special cases where it becomes first-order is studied in (Vassos and Patrizi 2013). Similarly, answering projection queries is in the general case undecidable and a few decidable cases have been studied, e.g., when the KB is a regular database (Reiter 2001) or the case of a modified version of the situation calculus language built using a two-variable fragment of first-order logic (Gu and Soutchanski 2007).

Notably, the case when the KB has the form of a *generalized database with constraints* (Kanellakis, Kuper, and Revesz 1995), which allows to specify relations with possibly *infinitely many tuples*, has not been investigated. In this work we show that for a special type of BATs whose KB is a generalized database with equality constraints, projection is decidable and a first-order progression can always be computed. We then look into richer forms of projection that may refer to more than one possible evolution of the initial KB, e.g., capturing properties of the form “after execution of $\alpha_1, \dots, \alpha_n$ condition ϕ always holds in the future” and specify a condition that ensures decidability also for them.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

BATs with generalized fluent databases

The situation calculus as presented by Reiter (2001) is a three-sorted first-order language with equality (and some limited second-order features). The sorts are used to distinguish between actions, situations, and objects. A *situation* represents a world history as a sequence of actions. 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 $do(a, s)$ denotes the successor situation resulting from performing action *a* in situation *s*. A (relational) *fluent* is a predicate whose last argument is a situation, whose value can change from situation to situation. We also assume a finite number of fluent and action symbols, \mathcal{F} and \mathcal{A} , and an infinite number of constants \mathcal{C} .

Reiter (2001) investigates the case where the initial knowledge base (KB) is a *definitional theory* characterized as follows: $\bigwedge_{F_i \in \mathcal{F}} \forall \vec{x}_i. F(\vec{x}_i, S_0) \equiv \phi_i(\vec{x}_i)$, where $\phi_i(\vec{x}_i)$, called the *definition* for F_i , is an unrestricted first-order formula mentioning no situations. Definitional KBs express *complete* information for fluents under the assumption of the unique-name axioms for constants captured in a set \mathcal{E} . For example the following axiom states that there are exactly two atoms true for $In(x_1, x_2, S_0)$, namely $In(box, it_1, S_0)$ and $In(box, it_2, S_0)$:

$$\forall x \forall y (In(x, y, S_0) \equiv (x = box \wedge (y = it_1 \vee y = it_2))).$$

Nonetheless, the definition for a fluent can be any unrestricted first-order formula built over the constants in \mathcal{C} and equality. For instance, it could have the following form implying an infinity of ground atoms that are true in S_0 :

$$\forall x \forall y (In(x, y, S_0) \equiv (x \neq box \wedge (y = it_1 \vee y = it_2))).$$

More complicated definitions can be formed also using quantification which is actually not adding to the expressiveness as first-order theories of equality admit quantifier elimination (Enderton and Enderton 2001).

We now present definitions from (Kanellakis, Kuper, and Revesz 1995) in the context of the situation calculus.

Definition 1. An *equality constraint* is a literal formula $x\theta y$ or $x\theta c$, where $c \in \mathcal{C}$ and θ is = or \neq . A *generalized k-tuple* over variables x_1, \dots, x_k is a finite conjunction ψ of equality constraints whose all variables are free and among x_1, \dots, x_k . A *generalized relation of arity k* is a finite set

$R = \{\psi_1, \dots, \psi_q\}$, of generalized k -tuples over x_1, \dots, x_k . The formula corresponding to a generalized relation R is the disjunction $\psi_1 \vee \dots \vee \psi_q$. We will use ϕ_R to denote the quantifier-free formula corresponding to relation R .

We will be dealing with the so-called *basic action theories* of the form $\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_0 \cup \Sigma$, where: \mathcal{D}_{ap} is a set of action precondition axioms (APs) and \mathcal{D}_{ss} is a set of successor state axioms (SSAs), one per fluent symbol $F_i \in \mathcal{F}$, \mathcal{D}_{una} is the set of unique-names axioms for actions, \mathcal{D}_0 describes the initial situation S_0 , and Σ is a set of foundational axioms which formally define legal situations. We identify BATs over generalized databases as follows.

Definition 2. A set \mathcal{D}_0 of first-order sentences uniform in S_0 is a *generalized fluent database (GFDB)* iff it has the form $\bigwedge_{F_i \in \mathcal{F}} \forall \vec{x}_i. F(\vec{x}_i, S_0) \equiv \phi_i(\vec{x}_i)$, where $\phi_i(\vec{x}_i)$ corresponds to a generalized relation over \vec{x} , i.e., is a disjunction of conjunctions of equality constraints. A basic action theory \mathcal{D} is a *basic action theory over a generalized fluent database (BAT-GFDB)* iff it also includes the set of unique-names axioms for constants \mathcal{E} , and \mathcal{D}_0 is a GFDB.

Theorem 1. Let ϕ be a definitional KB. There exists a generalized fluent database ϕ' such that $\mathcal{E} \models \phi \equiv \phi'$.

Kanellakis *et. al* (1995) show that for such KBs there is also a decidable procedure for evaluating queries. Note also that equivalence of generalized fluent databases can also be decided, formed as an appropriate query. With these tools available we now proceed to show how solutions for progression and projection can be obtained for BAT-GFDBs.

Progression of BAT-GFDBs

In order to do a one-step progression of a BAT \mathcal{D} with respect to a ground action α we need to replace \mathcal{D}_0 in \mathcal{D} by a suitable set \mathcal{D}_α of sentences uniform in $do(\alpha, S_0)$ so that the original theory \mathcal{D} and the theory $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_\alpha$ are equivalent with respect to how they describe the situation $do(\alpha, S_0)$ and the situations in the future of $do(\alpha, S_0)$.

In a seminal paper Lin and Reiter (1997) gave a model-theoretic definition for the progression \mathcal{D}_α of \mathcal{D}_0 wrt α and \mathcal{D} that achieves this goal. Finding such a \mathcal{D}_α is a difficult task and it has been shown that second-order logic may be required in the general case (Lin and Reiter 1997; Vassos and Levesque 2013). Nonetheless, for the definitional KBs, and as a consequence also for the special case of generalized fluent databases, there is a simple way to progress. Note that an SSA, that has the form $F(\vec{x}, do(a, s)) \equiv \Phi(\vec{x}, a, s)$, characterizes the conditions under which F has a specific value at situation $do(a, s)$ as a function of situation s and action a . A progression can then be obtained by replacing fluent atoms $F(\vec{o}, S_0)$ in the right-hand side of the SSA by their definition in the KB (Reiter 2001). It is interesting to look into how this method works when \mathcal{D}_0 is a GFDB.

Note that since each $\Phi(\vec{x}_i, \alpha, S_0)$ in the right-hand side of SSAs is in general unrestricted, e.g., may include quantifiers, \mathcal{D}_α is not guaranteed to be in the form of a generalized fluent database even though \mathcal{D}_0 is. The point in using a form like the generalized fluent database is that it allows us to perform query evaluation using the methods and existing

technologies in constraint databases instead of performing more general theorem proving. Therefore, we want progression to preserve the form of \mathcal{D}_0 . The previous substitution method does well in preserving the form of a *definitional KB* but does not preserve the form in the case of a GFDB.

The idea then is to consider generalized tuples as the “base” formulas that we use to express any generalized fluent relation. This is similar to a regular database where we would update \mathcal{D}_0 into a \mathcal{D}_α such that for every fluent a finite list of tuples is specified. Theorem 1 then provides a way to transform, by means of quantifier elimination, the resulting \mathcal{D}_α of the substitution method into the form of a GFDB.

Theorem 2. Let \mathcal{D} be a BAT over a generalized fluent database and α a ground action. Then there exists a first-order progression \mathcal{D}_α of \mathcal{D}_0 wrt α and \mathcal{D} that is in the form of a generalized fluent database.

As a result, we can *iteratively* progress a BAT-GFDB and characterize the state of any ground situation as a GFDB.

Finally, since every definitional KB can be expressed as a generalized fluent database, this also illustrates a subtle detail about progression. Both a progression \mathcal{D}_α according to the substitution method and a progression \mathcal{D}'_α according to Theorem 2 qualify as logically correct progressions of \mathcal{D}_0 and are logically equivalent (under the assumption of \mathcal{E}). Nonetheless, \mathcal{D}_0 is more of a *logical specification of the changes* that need to be made due to action α and \mathcal{D}'_α more of a *materialized update* of these changes.

Another way to look at it is that the progression based on the substitution method is purely syntactic (linear to the size of \mathcal{D}_0) and does not involve any form of evaluation; in a sense, the fluents are not updated to a new truth value but, rather, the new truth values are still specified with respect to the initial situation. As there are to date no implemented systems for performing progression, this difference is less easy to identify because, as far as the logical specification is concerned, \mathcal{D}_0 is a well-behaved progression. In practice, though, we would expect that unless a method that goes along the lines of Theorem 2 is followed, the progressed theory would in fact look much similar to the original theory \mathcal{D} , and queries over the current state would perform similar to using a *regression* technique over the original theory \mathcal{D} .

Projection over BAT-GFDBs

The (*simple*) *projection problem* is the task of *predicting* whether a condition holds at a particular time in the future after a series of ground actions have been executed (Reiter 2001). The following is a straightforward result.

Theorem 3. Let \mathcal{D} be a BAT-GFDB, $\alpha_1, \dots, \alpha_n$ a sequence of ground actions, and $\phi(s)$ a first-order formula uniform in s . Then determining whether or not the following holds is decidable: $\mathcal{D} \models \phi(do(\alpha_n, \dots do(\alpha_1, S_0)))$.

This is not a new result and can be proven by means of regression and the fact that \mathcal{E} is decidable. Our previous analysis also shows that simple projection queries over a BAT-GFDB can be decided by iteratively progressing \mathcal{D}_0 wrt $\alpha_1, \dots, \alpha_n$ according to Th. 2 and then evaluating the query over the resulting generalized fluent database following the

method of (Kanellakis, Kuper, and Revesz 1995). Depending on the type of queries, and the frequency that actions occur, either approach may be preferred under conditions.

We now proceed to show a major result about the decidability of richer projection queries over BAT-GFDBs that may also quantify over future situations. A *generalized* version of the projection problem is when ϕ may refer to any number or combination of future situations. For example, $\forall s(do(\alpha, S_0) \sqsubseteq s \supset \psi(s))$ states that after executing action α condition $\psi(s)$ will hold in all situations s in the future, i.e., that it will always remain true.

We consider the language \mathcal{L}_p of *generalized projection queries* φ . \mathcal{L}_p is defined on top of the language \mathcal{L}_n , whose formulas ϕ are as follows: $\phi := x = c \mid x = y \mid F(\vec{x}, s) \mid F(\vec{x}, \sigma) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x.\phi$, for F a fluent symbol, c a constant, and σ a ground situation term. \mathcal{L}_p formulas are defined as: $\varphi := \phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists s.\sigma \sqsubseteq s \wedge \varphi$, where $\phi \in \mathcal{L}_n$ is any formula uniform in s or in a ground situation term σ , whose free variables (if any) are only of sort situation.

We also consider a class of BAT-GFDBs which we call *C*-bounded. To define it, let T_V be the (finite) set of all generalized tuples ψ that use equality constraints with (only variable) symbols from the finite set of variables V , and s.t. ψ does not contain multiple occurrences of some equality constraint. Notice that since generalized tuples are conjunctions (thus multiple occurrences of a conjunct do not change their semantics), T_V essentially contains all the possible generalized tuples one can build using symbols from V .

Definition 3. Let \mathcal{D} be a BAT-GFDB and B a natural number. A ground situation term σ is said to be *constant-bounded by B in \mathcal{D}* , *C*-bounded by B for short, iff for every fluent $F_i(\vec{x}, s) \in \mathcal{F}$, it is the case that $\mathcal{D} \models \bigvee_{\Psi \in 2^{T_V}} \exists \vec{y}. F(\vec{x}, \sigma) \equiv \bigvee_{\psi \in \Psi} \psi$, where: V is partitioned into X and Y , with X the set of variables occurring in \vec{x} , Y any set of variable symbols such that $|Y| = B$, and \vec{y} are the free variables of ψ coming from Y . \mathcal{D} is said to be *constant-bounded by a finite bound B* , *C*-bounded by B for short, iff every ground situation term of \mathcal{D} that is executable is also *C*-bounded by B .

Notice that Ψ above is a set of generalized tuples, thus the formula $\bigvee_{\psi \in \Psi} \psi$ is a generalized relation. Intuitively, Def. 3 requires that the definition of each fluent in σ is a generalized relation mentioning at most B distinct constants.

For this class of theories, we have the following result.

Theorem 4. *Given a BAT-GFDB \mathcal{D} that is *C*-bounded by some B , and a generalized projection query φ that is a sentence in \mathcal{L}_p , it is decidable to check whether $\mathcal{D} \models \varphi$.*

The proof starts by introducing the notion of *transition system over generalized fluent databases (GFDB-TS)*. These are transition systems with states labelled by generalized fluent databases and transitions labelled by ground actions. The aim of GFDB-TSs is to capture the information, represented as GFDBs, about both the states associated with the executable situations of an action theory, and their transitions, together with the respective actions. In particular the GFDB-TS $T_{\mathcal{D}}$ induced by an action theory \mathcal{D} can be defined, which accounts for the evolution of the states of the theory starting from S_0 .

Then, a definition of satisfaction of \mathcal{L}_p sentences wrt GFDB-TSs is provided, and it is proven that, for any generalized projection query φ , $\mathcal{D} \models \varphi$ iff $T_{\mathcal{D}} \models \varphi$. Thus, one can check whether $\mathcal{D} \models \varphi$ by using the induced GFDB-TS $T_{\mathcal{D}}$ instead of the theory \mathcal{D} . However, this does not guarantee decidability, as $T_{\mathcal{D}}$ is infinite-state, in general.

The final step of the proof consists in defining a procedure to build a finite-state GFDB-TS $\hat{T}_{\mathcal{D}}$ such that, for any sentence $\varphi \in \mathcal{L}_p$, $T_{\mathcal{D}} \models \varphi$ iff $\hat{T}_{\mathcal{D}} \models \varphi$. The construction exploits the key observations that if \mathcal{D} is *C*-bounded then: (i) it is first-order progressable; and (ii) its states can be partitioned into finitely many equivalence classes of logically equivalent GFDBs. Essentially, the procedure amounts to iteratively progressing \mathcal{D} from S_0 , by executing all possible ground actions obtained using a finite subset $H \subseteq \mathcal{C}$ of constants, until a state whose label is logically equivalent to that of some state visited in the past is generated. Once $\hat{T}_{\mathcal{D}}$ is available, checking whether $\hat{T}_{\mathcal{D}} \models \varphi$ can be done by applying a straightforward recursive procedure.

Conclusions

In this paper we looked into situation calculus action theories over generalized fluent databases with equality constraints (GFDB), connecting the situation calculus with constraint query languages. We showed that GFDBs characterize the class of definitional KBs with fluent predicates only, and that for action theories over such KBs (BAT-GFDBs), the KBs are *closed under progression*. Our results show that simple projection queries over BAT-GFDBs are decidable in general. Also, extending the notion of *boundedness* proposed in (De Giacomo, Lespérance, and Patrizi 2012), we introduced the notion of *C-boundedness* and showed that, under this, a wide class of generalized projection queries that include quantification over situations is decidable.

Acknowledgements The authors acknowledge support of the EU Project FP7-ICT 318338 (OPTIQUE) and the Sapienza Award 2013 “Spiritlets” project.

References

- De Giacomo, G.; Lespérance, Y.; and Patrizi, F. 2012. Bounded situation calculus action theories and decidable verification. In *Proc. of KR’12*.
- Enderton, H., and Enderton, H. B. 2001. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition.
- Gu, Y., and Soutchanski, M. 2007. Decidable reasoning in a modified situation calculus. In *Proc. of IJCAI’07*.
- Kanellakis, P. C.; Kuper, G. M.; and Revesz, P. Z. 1995. Constraint query languages. *Journal of Computer and System Sciences* 51(1):26–52.
- Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.
- Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT 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 Rossi, F., ed., *Proc. of IJCAI’13*.