Reasoning in the Situation Calculus with Limited Belief

Christoph Schwering
School of Computer Science and Engineering
The University of New South Wales
Sydney NSW 2052, Australia
c.schwering@unsw.edu.au

Abstract

First-order action formalisms like the situation calculus are powerful tools for modelling dynamic environments, but they are also notorious for having little practical relevance due to their computational complexity. This paper aims to develop an expressive yet practically useful situation calculus variant. To this end, we amalgamate the situation calculus with a first-order logic of limited belief which controls the complexity of reasoning by stratifying beliefs into levels. We show that reasoning in this formalism is decidable, prove soundness with respect to the original epistemic situation calculus, and introduce an implementation of the reasoning system.

1 Introduction

Since the early days of the field of Artificial Intelligence, reasoning about knowledge in the presence of actions and change has been regarded as a crucial component of autonomous machines [McCarthy, 1959]. Knowledge Representation research has produced a number of logical formalisms to address this need, like the situation calculus [McCarthy, 1963; Reiter, 2001], the event calculus [Kowalski and Sergot, 1989], the fluent calculus [Thielscher, 1998], or the propositional $\mathcal{A}$ language [Gelfond and Lifschitz, 1993]. Numerous additional concepts have been integrated into these frameworks, such as knowledge and belief and introspection, sensing, belief revision, multiple agents, uncertainty, and time and concurrency, to name but a few.

However, the practical impact of these logical action formalisms has been very limited; in fact, they are notorious for being computationally way too expensive for practical applications, which is why the Planning community resorts to different languages [McDermott *et al.*, 1998]. Nevertheless the need for reasoning in highly expressive action languages is real and still unmet in domains such as cognitive robotics [Levesque and Reiter, 1998; Levesque and Lakemeyer, 2008].

This paper aims to unify the expressivity of an action formalism with better computational properties. We devise a variant of the *epistemic situation calculus* [Lakemeyer and Levesque, 2011] and give it a semantics based on *limited belief* [Lakemeyer and Levesque, 2014; Schwering, 2017].

The situation calculus is a language for reasoning about actions and change. The typical reasoning task we face in the situation calculus is the *projection problem*: given a description of the initial state, a goal formula, and a sequence of actions, we want to know whether the goal formula is true after the sequence of actions. The projection problem is cast as a question of logical entailment, and the situation calculus being a first-order language, this is undecidable in general.

One way to avoid undecidability in first-order logic is through limited belief. Unlike prefix-vocabulary classes [Börger *et al.*, 1997] and description logics [Baader, 2003], logics of limited belief restrict the semantics instead of the syntax of the language. More precisely, belief is stratified into levels which become more and more complete but also more difficult to compute.

This paper is not the first attempt to marry the situation calculus to limited belief. An earlier approach by Claßen and Lakemeyer [2009] builds on a variant of limited belief [Liu *et al.*, 2004] that has been surpassed both in expressivity and theoretical properties by more recent developments. Also, this approach does not give semantics to actions in the logic, and the concept of sensing is not present at all. A second approach is due to Lakemeyer and Levesque [2014]. Here actions are built into the logic and sensing is accounted for, but lacking function symbols, the language only supports unparameterised actions. By contrast, our approach in the present paper goes beyond that using a definition of action functions which is expressive enough to allow for parameters, but restrictive enough to maintain the decidability of limited belief. Finally, an issue that concerns both of the earlier approaches is that their contributions remain at the theoretical level, whereas this paper aims to go further by introducing an implementation of the limited epistemic situation calculus.

The rest of the paper is structured as follows. The next section introduces a variant of the epistemic situation calculus. This fully fledged first-order epistemic logic will serve as a role model for the later development of a limited counterpart. Section 2 also introduces basic action theories and regression, the standard approach to projection in the situation calculus. Then Section 3 proceeds to develop an alternative, limited semantics of the language from Section 2, gives soundness and decidability results, and introduces an implementation of a reasoning system for the logic. Then we conclude and discuss future work.

2 The Epistemic Situation Calculus

The situation calculus is a framework for reasoning about knowledge in the presence of actions. The key idea is that the state of the world is determined by the sequence of actions that have been executed since the initial state. Such a sequence of actions is called *situation*, and the value of functions and predicates may vary from situation to situation.

In this paper we build on the modal variant of the situation calculus due to Lakemeyer and Levesque [2011]. Compared to the axiomatic approach [Reiter, 2001; Scherl and Levesque, 2003] this logic exhibits often much simpler proofs, especially when it comes to epistemic reasoning.

This section introduces the syntax and semantics of Lakemeyer and Levesque's epistemic situation calculus, the concept of basic action theories, and regression as a means to solve the projection problem. The presentation largely follows [Lakemeyer and Levesque, 2011]. A notable difference however lies in the definition of terms in our language, which shall become essential when we develop an alternative, limited semantics of this language in Section 3.

2.1 The Language

The language is a first-order logic with functions, equality, and modal operators for knowledge and actions.

The terms of the language come in two disjoint sorts, *action* and *object*. Both sorts have an infinite supply of variables and function symbols of every arity; the object sort additionally comes with infinitely many standard names.

- An *action term* is a variable or an expression $A(t_1, \ldots, t_j)$, where A is a j-ary function symbol and every t_i is an object variable or standard name.
- An *object term* is a variable, a standard name, or an expression $f(t_1, \ldots, t_j)$, where f is a j-ary function and every t_i is an object variable or standard name or an action term.

As usual, a term is *ground* when it contains no variable.

The purpose of standard names is to uniquely designate different individuals. In other words, standard names satisfy the unique-names assumption and an infinitary domain closure. The *object standard names* come as special symbols that are distinct from the normal function symbols. The action sort does feature such special symbols. Instead, we define every ground action term to be an *action standard name*.

Notice that the above definition of action and object terms imposes restrictions on the possible nesting of terms. In particular, an object function cannot occur nested in another action or object function, and an action function cannot occur nested in another action function.

For example, we use an object standard name Sally to represent one specific person. When it is unclear who Sally's father is, this individual can be referred by an object term fatherOf(Sally), which is then subject to interpretation; for instance, fatherOf(Sally) might be Frank or Fred. When Mia gives birth to her daughter Sally, this event can be modelled with an action term birth(Mia, Sally). Another action could test the paternity for Fred of Sally, represented by test(Fred, Sally). Mia, Sally, and Fred being standard names, they refer to distinct individuals, and so do the action standard names birth(Mia, Sally) and test(Fred, Sally).

To model that an action like the paternity test may produce new knowledge, the epistemic situation calculus employs a distinguished function $\mathrm{sf}(n)$, whose value indicates the sensing result of action n.

The set of *formulas* is the least set that contains all expressions $t_1 = t_2$, $\neg\alpha$, $(\alpha \vee \beta)$, $\exists x \alpha$, $[t]\alpha$, $\Box\alpha$, $\mathbf{K}\phi$, $\mathbf{O}\phi$, where t_1 is an arbitrary term, t_2 is a term that mentions no object function, α and β are arbitrary formulas, x is a variable, t is an action term, and ϕ is a formula that mentions neither $\mathbf{K}$ nor $\mathbf{O}$. A *sentence* is a formula without free variables, and α^x_t denotes the result of substituting t for all free occurrences of the variable x in α.

$[t]\alpha$ and $\Box\alpha$ are the action operators and mean that "α is true after executing action t" and "α is true now and in every future situation," respectively. $\mathbf{K}\phi$ and $\mathbf{O}\phi$ are the knowledge operators; they are read as "ϕ is known" and "ϕ is *all* that is known," respectively. The latter operator is also known as only-knowing operator [Levesque, 1984]. It is particularly useful to capture the idea of a knowledge base, which typically is assumed to exhaustively describe what the agent knows.

We use the usual abbreviations $\neq$, $\wedge$, $\forall$, $\supset$, $\equiv$. Sometimes we allow ourselves to omit brackets when the operator precedence is clear from context.

As an example of the language's expressivity consider the formula $[\mathrm{birth}(\mathrm{Mia}, \mathrm{Sally})]\mathbf{K}\forall x(\mathrm{spouseOf}(\mathrm{Mia}) = x \supset \mathrm{fatherOf}(\mathrm{Sally}) = x)$, which is to say that after Mia gives birth to Sally, it is known that Mia's spouse is the father of Sally.

2.2 The Semantics

Models in this logic have three components: a set of possible worlds e, an actual world w, and a situation z. A situation z is a sequence of action standard names, or action sequence for short, that represent the history of executed actions; we write $\langle\rangle$ for the empty sequence and use $\cdot$ as concatenation operator. A world w is a function that maps every ground object term $f(n_1, \ldots, n_j)$ and situation z to an object standard name $w(f(n_1, \ldots, n_j), z)$.

Truth of a sentence α in a model e, w, z is written $e, w, z \models \alpha$. We begin with the semantics of the non-modal fragment:

1. $e, w, z \models t = n$ iff
   t and n are identical standard names, or
   $w(t, z)$ and n are identical standard names;
2. $e, w, z \models (\alpha \vee \beta)$ iff $e, w, z \models \alpha$ or $e, w, z \models \beta$;
3. $e, w, z \models \neg\alpha$ iff $e, w, z \not\models \alpha$;
4. $e, w, z \models \exists x \alpha$ iff
   $e, w, z \models \alpha^x_n$ for some standard name n of the sort of x.

Note how Rule 4 handles quantification by substituting standard names. Standard names thus effectively serve as a fixed, countably infinite universe of discourse. We refer to [Levesque, 1984] for a discussion why this is no effective limitation for our purposes.

We proceed with the semantics of the action operators, which simply append new actions to the current situation z:

5. $e, w, z \models [n]\alpha$ iff $e, w, z \cdot n \models \alpha$;
6. $e, w, z \models \Box\alpha$ iff $e, w, z \cdot \hat{z} \models \alpha$ for all action sequences $\hat{z}$.

We now give the rules for the knowledge operators, which complete the semantics. Recall that $\mathrm{sf}(n)$ is a distinguished function symbol that models the sensing result of action n. The idea is that upon performing action n, the agent learns the real-world value of $\mathrm{sf}(n)$. For an action sequence z, we write $w \simeq_z \hat{w}$ to say that $w(\mathrm{sf}(n), \hat{z}) = \hat{w}(\mathrm{sf}(n), \hat{z})$ for all prefixes $\hat{z} \cdot n$ of z; that is, w and $\hat{w}$ agree on the sensing outcomes throughout z. Then knowledge is defined as follows:

7. $e, w, z \models \mathbf{K}\phi$ iff $e, \hat{w}, z \models \phi$ for all $\hat{w} \in e$ with $w \simeq_z \hat{w}$;
8. $e, w, z \models \mathbf{O}\phi$ iff
   $e, w, z \models \mathbf{K}\phi$ and $\hat{e}, w, z \not\models \mathbf{K}\phi$ for all $\hat{e} \supsetneq e$.

Observe that $\mathbf{O}\phi$ implies $\mathbf{K}\phi$, but additionally maximises the set of possible worlds e. The intuitive effect is that the agent considers possible every scenario that is compatible with ϕ.

We say a sentence α is *valid*, written $\models \alpha$, iff for all e, w, $e, w, \langle\rangle \models \alpha$.

2.3 Basic Action Theories

One particular way of modelling a dynamic domain in this language is called *basic action theory* [Reiter, 2001]. A basic action theory over a finite set of function symbols $\mathcal{F}$ comes with two types of axioms: the dynamic axioms govern how the functions in $\mathcal{F}$ evolve over the course of action, and the initial axioms describe the initial situation.

For every $f \in \mathcal{F}$ there shall be one *dynamic axiom*, which is of either of the following forms:

1. $\Box\forall x_1 \ldots \forall x_j \forall y \forall a\big([a]f(x_1, \ldots, x_j) = y \equiv \gamma\big)$;
2. $\Box\forall x_1 \ldots \forall x_j \forall y\big(f(x_1, \ldots, x_j) = y \equiv \pi\big)$;

where γ and π shall not mention any action or knowledge operators and satisfy the functional consistency property that y is uniquely determined by the other free variables. The first axiom is called a *successor-state axiom*, as it relates the value of $f(\vec{x}) = y$ after action a to the value of γ before a. The second axiom is a *definitional axiom*, since it relates the value of $f(\vec{x}) = y$ to the value of π in the same situation. A basic action theory shall contain a definitional axiom for the $\mathrm{sf}(a)$ function. To ensure there are no cyclic dependencies, we require that π mentions no defined functions.

The other component of a basic action theory describes what is true in the *initial situation*. Typically there is one sentence δ_0 that refers to what is true initially in the *real world*, and another sentence δ_1 that describes what the agent *knows* initially. Both δ_0 and δ_1 shall mention no action or knowledge operator and only functions from $\mathcal{F}$ with a successor-state axioms.

Given a basic action theory with dynamic axioms Δ and descriptions of the initial situation δ_0, δ_1, the *projection problem* boils down to deciding logical entailments of

$$\Delta \wedge \delta_0 \wedge \mathbf{O}(\Delta \wedge \delta_1).$$

For instance, to show that ϕ is known after actions $t_1, \ldots, t_j$, we need to prove $\models \Delta \wedge \delta_0 \wedge \mathbf{O}(\Delta \wedge \delta_1) \supset [t_1] \ldots [t_j]\mathbf{K}\phi$. In this paper, we are only concerned with queries that do not involve the $\Box$ operator.

Example

As an example, let us model a family scenario where Sally reasons about her genealogy with a basic action theory. First, Mia gives birth to Sally, so according to the Latin law *mater semper certa est*, Mia must be Sally's mother. The father, however, remains uncertain to Sally; all she knows about the matter is that Mia is in a relationship with either Frank or Fred, so one of them must be the father (we implicitly assume Mia is faithful to her spouse). To reach clarification, Sally requests a paternity test for Fred, and since the test result is negative, Sally rightly concludes that Frank is her father.

We model the scenario using functions motherOf, fatherOf, and spouseOf. We have two actions: $\mathrm{birth}(x, y)$ and $\mathrm{test}(x, y)$ to say that x gives birth to y and to test whether x is the father of y. The paternity test is modelled with the sf function: $\mathrm{sf}(\mathrm{test}(x, y))$ shall return Yes if x is the father of y, and No otherwise, where Yes and No are two distinguished standard names. The scenario can now be modelled as follows:

- $\Box\forall x\forall y\forall a\big([a]\mathrm{motherOf}(x) = y \equiv a = \mathrm{birth}(y, x) \vee a \neq \mathrm{birth}(y, x) \wedge \mathrm{motherOf}(x) = y\big)$;
- $\Box\forall x\forall y\forall a\big([a]\mathrm{fatherOf}(x) = y \equiv \exists\hat{y}(a = \mathrm{birth}(\hat{y}, x) \wedge \mathrm{spouseOf}(\hat{y}) = y) \vee \forall\hat{y}(a \neq \mathrm{birth}(\hat{y}, x) \wedge \mathrm{fatherOf}(x) = y)\big)$;
- $\Box\forall x\forall y\forall a\big([a]\mathrm{spouseOf}(x) = y \equiv \mathrm{spouseOf}(x) = y\big)$;
- $\Box\forall y\forall a\big(\mathrm{sf}(a) = y \equiv \exists x\exists\hat{y}(a = \mathrm{test}(\hat{y}, x) \wedge \mathrm{fatherOf}(x) = \hat{y}) \wedge y = \mathrm{Yes} \vee \forall x\forall\hat{y}(a \neq \mathrm{test}(\hat{y}, x) \vee \mathrm{fatherOf}(x) \neq \hat{y}) \wedge y = \mathrm{No}\big)$.

As for the initial situation, we have that Frank is Mia's spouse, but Sally only knows it is Frank or Fred:

- $\delta_0 \overset{\text{def}}{=} \mathrm{spouseOf}(\mathrm{Mia}) = \mathrm{Frank}$;
- $\delta_1 \overset{\text{def}}{=} (\mathrm{spouseOf}(\mathrm{Mia}) = \mathrm{Frank} \vee \mathrm{spouseOf}(\mathrm{Mia}) = \mathrm{Fred})$.

To conclude this example, let us consider a few projection tasks: after Sally is born, Mia is known to be the mother, but it is not known whether Frank or Fred is the father, and only after the paternity test for Fred, Frank's fatherhood becomes certain. These statements can be translated to the following projection tasks, where Λ denotes the basic action theory $\Delta \wedge \delta_0 \wedge \mathbf{O}(\Delta \wedge \delta_1)$:

1. $\models \Lambda \supset [\mathrm{birth}(\mathrm{Mia}, \mathrm{Sally})]\mathbf{K}\,\mathrm{motherOf}(\mathrm{Sally}) = \mathrm{Mia}$;
2. $\models \Lambda \supset [\mathrm{birth}(\mathrm{Mia}, \mathrm{Sally})]\neg\mathbf{K}\,\mathrm{fatherOf}(\mathrm{Sally}) = \mathrm{Frank}$;
3. $\models \Lambda \supset [\mathrm{birth}(\mathrm{Mia}, \mathrm{Sally})]\mathbf{K}\,(\mathrm{fatherOf}(\mathrm{Sally}) = \mathrm{Frank} \vee \mathrm{fatherOf}(\mathrm{Sally}) = \mathrm{Fred})$;
4. $\models \Lambda \supset [\mathrm{birth}(\mathrm{Mia}, \mathrm{Sally})][\mathrm{test}(\mathrm{Fred}, \mathrm{Sally})]\mathbf{K}\,(\mathrm{fatherOf}(\mathrm{Sally}) = \mathrm{Frank})$.

We prove the second statement. Suppose $e, w, \langle\rangle \models \Lambda$. Let $n = \mathrm{birth}(\mathrm{Mia}, \mathrm{Sally})$. We show $e, \hat{w}, n \models \mathrm{fatherOf}(\mathrm{Sally}) \neq \mathrm{Frank}$ for some $\hat{w} \in e$ with $w \simeq_n \hat{w}$. Consider $\hat{w} \in e$ with $\hat{w}(\mathrm{spouseOf}(\mathrm{Sally}), \langle\rangle) = \mathrm{Fred}$. Then $w \simeq_n \hat{w}$, since $w(\mathrm{sf}(n), \langle\rangle) = \hat{w}(\mathrm{sf}(n), \langle\rangle)$, and $\hat{w}(\mathrm{fatherOf}(\mathrm{Sally}), n) = \hat{w}(\mathrm{spouseOf}(\mathrm{Sally}), \langle\rangle) \neq \mathrm{Frank}$, which completes the proof. As for the fourth entailment, note that the outcome of the paternity test, $\mathrm{sf}(\mathrm{test}(\mathrm{Fred}, \mathrm{Sally}))$, will be No, which rules out the worlds $\hat{w}$ with $\hat{w}(\text{fatherOf}(\text{Sally}), n) = \text{Fred}$, so Frank remains the only possible father.

2.4 Regression

The structure of successor-state axioms gives rise to a simple procedure to reduce statements about future situations to statements about the initial situation: we can replace the functions in the query formula with the right-hand side of the corresponding dynamic axiom, and repeat this until no action operators are left.

For instance, $[\text{birth}(\text{Mia}, \text{Sally})]\text{motherOf}(\text{Sally}) = \text{Mia}$ reduces to the formula $\text{birth}(\text{Mia}, \text{Sally}) = \text{birth}(\text{Mia}, \text{Sally}) \lor (\text{birth}(\text{Mia}, \text{Sally}) \neq \text{birth}(\text{Mia}, \text{Sally}) \land \text{motherOf}(\text{Sally}) = y)$.

This procedure is called *regression* [Reiter, 2001]. We say a formula is *regressable* with respect to a basic action theory over $\mathcal{F}$ when it mentions no $\Box$ or $\mathbf{O}$ operators and all functions are from $\mathcal{F}$. We now define the regression operator $\mathcal{R}[\alpha]$ with respect a basic action theory for regressable formulas α. To ease the technical treatment we assume that no variable in α is bound twice. Then $\mathcal{R}[\alpha]$ shall stand for $\mathcal{R}[\langle\rangle, \alpha]$, which is defined as follows:

1. if f has a successor-state axiom with right-hand side γ:
   $\mathcal{R}[z \cdot t, f(t_1, \ldots, t_j) = t_{j+1}] \overset{\text{def}}{=} \mathcal{R}[z, \gamma^{x_1 \ldots x_j\, y \quad a}_{t_1 \ldots t_j\, t_{j+1}\, t}]$;
   if f has a definitional axiom with right-hand side π:
   $\mathcal{R}[z, f(t_1, \ldots, t_j) = t_{j+1}] \overset{\text{def}}{=} \mathcal{R}[z, \pi^{x_1 \ldots x_j\, y}_{t_1 \ldots t_j\, t_{j+1}}]$;
   otherwise:
   $\mathcal{R}[z, t_1 = t_2] \overset{\text{def}}{=} t_1 = t_2$;
2. $\mathcal{R}[z, (\alpha \lor \beta)] \overset{\text{def}}{=} (\mathcal{R}[z, \alpha] \lor \mathcal{R}[z, \beta])$;
3. $\mathcal{R}[z, \neg\alpha] \overset{\text{def}}{=} \neg\mathcal{R}[z, \alpha]$;
4. $\mathcal{R}[z, \exists x\, \alpha] \overset{\text{def}}{=} \exists x\, \mathcal{R}[z, \alpha]$;
5. $\mathcal{R}[z, [t]\alpha] \overset{\text{def}}{=} \mathcal{R}[z \cdot t, \alpha]$;
6. $\mathcal{R}[z \cdot t, \mathbf{K}\,\phi] \overset{\text{def}}{=} \mathcal{R}[z, \big(\text{sf}(t) = x \supset \mathbf{K}\,(\text{sf}(t) = x \supset [t]\phi)\big)]$;
   $\mathcal{R}[\langle\rangle, \mathbf{K}\,\phi] \overset{\text{def}}{=} \mathbf{K}\,\mathcal{R}[\langle\rangle, \phi]$.

Rule 1 implements the idea of replacing functions with the right-hand side of its corresponding axiom. Rule 6 uses a similar idea to regress knowledge, except that in place of an axiom from the basic action theory we use the following theorem that relates what is known after an action to what is known before that action, conditioned on the sensing result:

Theorem 1
$\models \Box\forall a\big([a]\mathbf{K}\,\phi \equiv \forall x\big(\text{sf}(a) = x \supset \mathbf{K}\,(\text{sf}(a) = x \supset [a]\phi)\big)\big)$.

Proof. Let $\hat{n} = w(\text{sf}(n), z)$. Then $e, w, z \models [n]\mathbf{K}\,\phi$ iff $e, \hat{w}, z \cdot n \models \phi$ for all $\hat{w} \in e$ with $w \simeq_z \hat{w}$ and $\hat{n} = \hat{w}(\text{sf}(n), z)$ iff $e, \hat{w}, z \models (\text{sf}(n) = \hat{n} \supset [n]\phi)$ for all $\hat{w} \in e$ with $w \simeq_z \hat{w}$ iff $e, w, z \models \mathbf{K}\,(\text{sf}(n) = \hat{n} \supset [n]\phi)$ iff $e, w, z \models \forall x(\text{sf}(n) = x \supset \mathbf{K}\,(\text{sf}(a) = x \supset [n]\phi))$. $\Box$

For example, $[\text{test}(\text{Fred}, \text{Sally})]\mathbf{K}\,(\text{fatherOf}(\text{Sally}) = \text{Frank})$ is regressed to $\forall x(\phi \supset (\mathbf{K}\phi \supset \text{fatherOf}(\text{Sally}) = \text{Frank})$, where ϕ is the regression of $\text{sf}(\text{test}(\text{Fred}, \text{Sally})) = x$, which after some simplifications reduces to $(\text{fatherOf}(\text{Sally}) = \text{Fred} \land x = \text{Yes} \lor \text{fatherOf}(\text{Sally}) \neq \text{Fred} \land x = \text{No})$

Regression is a very elegant way to handle projection, and we shall use it in this paper to reduce reasoning about actions to a static reasoning case. The next theorem says that this is correct [Reiter, 2001; Schwering *et al.*, 2017]:

Theorem 2 *Let α be regressable with respect to a basic action theory with dynamic axioms Δ and descriptions of the initial situation δ_0, δ_1.*
Then $\models \Delta \land \delta_0 \land \mathbf{O}(\Delta \land \delta_1) \supset \alpha$ iff $\models \delta_0 \land \mathbf{O}\delta_1 \supset \mathcal{R}[\alpha]$.

3 The Limited Epistemic Situation Calculus

In this section we devise a variant of the epistemic situation calculus from Section 2 based on *limited belief*. The idea behind limited belief is to stratify belief into levels: level 0 comprises only the explicit beliefs; every following level draws additional inferences by doing another case split. The rationale behind this technique of limiting belief by case splits is the hypothesis that in many practical situations few case splits – perhaps one or two – suffice to obtain meaningful results [Schwering, 2017].

3.1 The Language

The syntax follows the same rules as in Section 2.1 with the addition of formulas $\mathbf{K}_k\,\phi$ for natural numbers $k \geq 0$. We read $\mathbf{K}_k\,\phi$ as "ϕ is known at level k."

3.2 The Semantics

We extend the semantics from Section 2.2 to incorporate expressions of the form $\mathbf{K}_k\,\phi$. This semantics fundamentally builds on three concepts: clause subsumption, unit propagation, and case splits. First we need some additional terminology.

A *literal* is an expression of the form $[z]t = n$ or $[z]t \neq n$ where z is a sequence of action standard names, t is a ground term or a standard name, and n is a standard name. We abuse notation and identify a literal $[\langle\rangle]\ell$ with ℓ, and $[z \cdot n]\ell$ with $[z][n]\ell$. A literal is *valid* when it is of the form $[z]n = n$, or $[z]n \neq \hat{n}$ for distinct standard names $n, \hat{n}$, or $[z]t \neq n$ for t, n of distinct sorts. A literal ℓ_1 *subsumes* a literal ℓ_2 when ℓ_1, ℓ_2 are identical or ℓ_1, ℓ_2 are of the form $[z]t = n_1$ and $[z]t \neq n_2$ for distinct n_1, n_2. Two literals ℓ_1, ℓ_2 are *complementary* when ℓ_1, ℓ_2 are of the form $[z]t = n$ and $[z]t \neq n$ (or vice versa), or ℓ_1, ℓ_2 are of the form $[z]t = n_1$ and $[z]t = n_2$ for distinct n_1, n_2.

A *clause* is a finite set of literals. A clause with a single literal is a *unit clause*. Again we abuse notation and identify non-empty clauses $\{\ell_1, \ldots, \ell_j\}$ with formulas $(\ell_1 \lor \ldots \lor \ell_j)$, and we write $[z]\{\ell_1, \ldots, \ell_j\}$ to denote $\{[z]\ell_1, \ldots, [z]\ell_j\}$. The above terminology for literals carries over to clauses as follows. A clause is *valid* when it contains a valid literal, or a literal $t = n$ and its negation $t \neq n$, or two literals $t \neq n_1$ and $t \neq n_2$ for distinct names n_1, n_2. A clause c_1 *subsumes* a clause c_2 when every literal $\ell_1 \in c_1$ subsumes a literal $\ell_2 \in c_2$. The *unit propagation* $c \setminus \{\hat{\ell} \mid \ell, \hat{\ell} \text{ are complementary}\}$ of a clause c with a literal ℓ is the clause obtained by removing from c all literals that are complementary to ℓ.

A *setup* s is a set of clauses that includes all valid clauses and is closed under unit propagation and subsumption: if c is valid or subsumed by some $\hat{c} \in s$, then $c \in s$, and if $c, \ell \in s$, then $c \setminus \{\hat{\ell} \mid \ell, \hat{\ell} \text{ are complementary}\} \in s$. We write $s \cup \hat{s}$ for the setup obtained by combining the setups s and $\hat{s}$.

We first define a semantics for formulas that mention no knowledge operator. We write $s, z, k \mathrel{|\!\approx} \phi$ to say that the setup s satisfies ϕ in situation z at belief level k, defined as follows:

1. if c is a clause:
   $s, z, 0 \approx c$ iff $[z]c \in s$;
2. if $(\phi \vee \psi)$ is not a clause:
   $s, z, 0 \approx (\phi \vee \psi)$ iff $s, z, 0 \approx \phi$ or $s, z, 0 \approx \psi$;
3. $s, z, 0 \approx \neg(\phi \vee \psi)$ iff $s, z, 0 \approx \neg\phi$ and $s, z, 0 \approx \neg\psi$;
4. $s, z, 0 \approx \neg\neg\phi$ iff $s, z, 0 \approx \phi$;
5. $s, z, 0 \approx \exists x \phi$ iff
   $s, z, 0 \approx \phi^x_n$ for some standard name n of the sort of x;
6. $s, z, 0 \approx \neg\exists x \phi$ iff
   $s, z, 0 \approx \neg\phi^x_n$ for all standard names n of the sort of x;
7. $s, z, 0 \approx [n]\phi$ iff $s, z \cdot n, 0 \approx \phi$;
8. $s, z, 0 \approx \Box\phi$ iff $s, z \cdot \hat{z}, 0 \approx \phi$ for all action sequences $\hat{z}$;
9. $s, z, k+1 \approx \phi$ iff
   for some ground term t,
   for all standard names n of the sort of t,
   $s \cup \{t = n\}, z, k \approx \phi$.

Intuitively, $s, z, k \approx \phi$ starts off by doing k case splits in Rule 9, and then proceeds to decompose the formula up to clause level and tests membership in s.

Let us illustrate this semantics with an example. Let s contain spouseOf(Mia) = Frank ∨ spouseOf(Mia) = Fred as well as for every n the clause spouseOf(Mia) $\neq n$ ∨ [birth(Mia, Sally)]fatherOf(Sally) $= n$. This setup requires belief level 1 to obtain that Frank or Fred is Sally's father after Mia gave birth to Sally: At belief level 0, we simply check for subsumption of [birth(Mia, Sally)]fatherOf(Sally)= Frank∨[birth(Mia, Sally)]fatherOf(Sally)=Fred, which fails. At belief level 1, however, we can split spouseOf(Mia): when we add spouseOf(Mia) = Frank to s, then we obtain [birth(Mia, Sally)]fatherOf(Sally) = Frank by unit propagation; analogously for Fred; and every other value of spouseOf(Mia) produces the empty clause by unit propagation; so in every case [birth(Mia, Sally)]fatherOf(Sally) = Frank∨[birth(Mia, Sally)]fatherOf(Sally)=Fred is subsumed and the query hence satisfied at level 1.

We can now use the $\approx$ truth relation to incorporate limited belief into the epistemic situation calculus. To account for the sensing history in the setup, we define the sensing outcome literals $\mathsf{SF}(w, z)$ inductively as $\mathsf{SF}(w, \langle\rangle) = \{\}$ and $\mathsf{SF}(w, z \cdot n) = \mathsf{SF}(w, z) \cup \{[z]\mathrm{sf}(n) = \hat{n} \mid \hat{n} = w(\mathrm{sf}(n), z)\}$. With the understanding that Rules 1–8 from Section 2.2 are retrofitted with an additional parameter s for the setup, we add the following rule for limited belief:

9. $s, e, w, z \models \mathbf{K}_k\phi$ iff $s \cup \mathsf{SF}(w, z), z, k \approx \phi$.

Adding the sensing outcome literals to s corresponds to the effect of filtering all worlds $\hat{w} \in e$ that violate $w \simeq_z \hat{w}$ in Rule 7.

Furthermore we replace Rule 8 with a new rule that takes the setup parameter into account:

8'. $s, e, w, z \models \mathbf{O}\phi$ iff
   $s, e, w, z \models \mathbf{K}\phi$ and $s, \hat{e}, w, z \not\models \mathbf{K}\phi$ for all $\hat{e} \supsetneq e$ and
   $s, e, w, z \models \mathbf{K}_0\phi$ and $\hat{s}, e, w, z \not\models \mathbf{K}_0\phi$ for all $\hat{s} \subsetneq s$.

This rule minimises the setup s to obtain the analogous effect as maximising e does.

Continuing our above example setup, we easily obtain $s, e, w, \langle\rangle \models \mathbf{K}_1$[birth(Mia, Sally)](fatherOf(Sally)=Frank∨ fatherOf(Sally) = Fred) for every e and w.

Reasoning is sound in the limited epistemic situation calculus with respect to its unlimited counterpart:

Theorem 3
If $\models \mathbf{O}\phi \supset [t_1] \ldots [t_j]\mathbf{K}_k\psi$, *then* $\models \mathbf{O}\phi \supset [t_1] \ldots [t_j]\mathbf{K}\psi$.

Proof. Suppose the antecedent holds and let $s, e, w, \langle\rangle \models \mathbf{O}\phi$. We need to show $s, e, w, z \models \mathbf{K}\psi$, where $z = t_1 \cdot \ldots \cdot t_j$. By assumption, $\hat{w} \in e$ iff $s, e, \hat{w}, \langle\rangle \models \phi$. Let $\hat{w} \in e$ with $w \simeq_z \hat{w}$ be arbitrary. It suffices to show that $s, e, \hat{w}, z \models \psi$. Let $\hat{s} = \{[\hat{z}]f(\vec{n}) = \hat{n} \mid \hat{n} = \hat{w}(f(\vec{n}), \hat{z})\}$. By induction on $|\phi|$ we can show $\hat{s}, \langle\rangle, 0 \approx \phi$. Thus there is some minimal $\tilde{s} \subseteq \hat{s}$ such that $\tilde{s}, z, 0 \approx \phi$. Then by assumption, $\tilde{s} \cup \mathsf{SF}(w, z), z, k \approx \psi$. Since by construction $\tilde{s} \cup \mathsf{SF}(w, z) \subseteq \hat{s}$, also $\hat{s}, z, k \approx \psi$. By another induction on $|\psi|$, $s, e, \hat{w}, z \models \psi$. □

In the propositional case, limited belief becomes complete at high enough belief levels:

Theorem 4 *Let ϕ, ψ be quantifier- and $\Box$-free.*
If $\models \mathbf{O}\phi \supset [t_1] \ldots [t_j]\mathbf{K}\psi$, *then* $\models \mathbf{O}\phi \supset [t_1] \ldots [t_j]\mathbf{K}_k\psi$ *for large enough k.*

Proof sketch. Let $\|\phi\|$ and $\|\psi\|$ be the number of object functions that occur in ϕ and ψ. Choosing $k = \|\phi\| + \|\psi\| + j$ then allows to split on all relevant functions in ϕ and ψ. □

3.3 Proper$^+$ Basic Action Theories

An important class of knowledge bases in the context of limited belief are proper$^+$ knowledge bases introduced by Lakemeyer and Levesque [2002]. A knowledge base ϕ is *proper*$^+$ when it is a conjunction where each conjunct is of the form $\forall x_1 \ldots \forall x_j c$ or $\Box \forall x_1 \ldots \forall x_j c$ where c is a disjunction of expressions $[t_1] \ldots [t_j]t_{j+1} = t_{j+2}$ or $[t_1] \ldots [t_j]t_{j+1} \neq t_{j+2}$. A basic action theory can be brought into proper$^+$ form by bringing it into prenex-CNF and Skolemization; we call such a transformed theory a *proper*$^+$ *basic action theory*.

Moreover, reasoning in such proper$^+$ knowledge bases is decidable:

Theorem 5 *Let Δ, δ_1 be a proper$^+$ basic action theory and ϕ be $\Box$-free. Then* $\models \mathbf{O}(\Delta \wedge \delta_1) \supset [t_1] \ldots [t_j]\mathbf{K}_k\phi$ *is decidable.*

Proof sketch. Since ϕ does not mention $\Box$, the length of the relevant situations is bounded, and we can instantiate the $\Box$ operators in Δ for these lengths. It moreover suffices to instantiate every variable with all standard names occurring in the reasoning task, plus those standard names that can be formed from action function symbols, and a finite amount of additional standard names that represent the infinitely many ones that are not mentioned in the formulas. Finally, it suffices to consider a finite set of terms for splitting. □

Decidability requires our strict definition of action standard names that precludes nested functions such as birth(birth(Mia, Sally), Sally). If such nested standard names were allowed, it would be possible to represent Robinson arithmetic [1950] with unit propagation obtaining the theory of the natural numbers, which is undecidable. This justifies our somewhat intricate definition of the action sort in Section 2.1.

In the propositional case and for fixed belief level and a fixed number of actions outside of $\mathbf{K}$, deciding such implications becomes even tractable:

Theorem 6 *Let Δ, δ_1 be a proper$^+$ basic action theory and ϕ be quantifier- and $\Box$-free. Let $l = |\Delta| + |\delta_1| + |\phi| + j$. Then $\models \mathbf{O}(\Delta \land \delta_1) \supset [t_1] \ldots [t_j]\mathbf{K}_k\phi$ is decidable in time $\mathcal{O}(2^{k+j} l^{k+3})$.*

Bringing the dynamic axioms into proper$^+$ form and then reason about them in the logic of limited is one way of decidable reasoning in the situation calculus. Alternatively, we can regress the query first with respect to dynamic axioms and reason about the regressed query with limited belief. Below, we denote by $(\alpha)_k$ the formula obtained by replacing in α every $\mathbf{K}$ with $\mathbf{K}_k$. Directly from Theorems 2 and 3 we obtain the following soundness result:

Corollary 7 *Let $\Delta, \delta_0, \delta_1$ be a basic action theory, and let α be regressable with respect to Δ. If $\models \delta_0 \land \mathbf{O}\delta_1 \supset (\mathcal{R}[\alpha])_k$, then $\models \Delta \land \delta_0 \land \mathbf{O}(\Delta \land \delta_1) \supset \alpha$.*

Such regressed queries are decidable as well (provided they mention no function symbols outside of $\mathbf{K}$) by Theorem 5:

Corollary 8 *Let Δ, δ_1 be a basic action theory, $\delta 1$ be proper$^+$, and α be regressable with respect to Δ and without functions outside of $\mathbf{K}$. Then $\models \mathbf{O}\delta_1 \supset (\mathcal{R}[\alpha])_k$ is decidable.*

3.4 A Reasoning System

As part of the limited-belief reasoning system LIMBO [Schwering, 2017], we have implemented procedures for deciding projection problems using the limited epistemic calculus.

The previous subsection has outlined two basic approaches to projection:

1. Translate the basic action theory into proper$^+$ form and let the reasoner do its work on the resulting knowledge base, which includes knowledge about actions and their effects.
2. Regress the query to eliminate the actions and then test for entailment by the description of the initial situation.

At the current stage, LIMBO only provides the second method; an implementation of the first one is underway.

An earlier prototype of LIMBO implemented both methods. Somewhat surprisingly, the second method turned out to be significantly faster than the first one in (albeit small-scale) experiments. Most likely, the cause for this was the large number of terms that need to be split during reasoning, thus leading to a high branching factor during search. Even when a preprocessing step was added to filter irrelevant splitting terms, regression remained faster, because said filtering needed to trace back fluents in a way very similar to regression.

LIMBO uses a very light-weight representation of literals $(\neg)t_1 = t_2$, where a single 64-bit integer uniquely identifies the literal and also includes all relevant information for testing subsumption, complementarity, and validity. While such compact representation is important for runtime performance reasons on the one hand, it is incompatible with an explicit representation of the situation z in a literal $[z](\neg)t_1 = t_2$ on the other hand. The syntactic restrictions of literals in LIMBO however offer us a simple way out: in a literal $[t_1] \ldots [t_i](\neg)f(t_{i+1}, \ldots, t_j) = t_{j+1}$ for an object function f, the terms t_i do not involve any object functions and hence are situation-independent. Therefore we can represent the literal by $(\neg)f_i(t_1, \ldots, t_i, t_{i+1}, \ldots, t_j) = t_{j+1}$, where f_i is a new function symbol.

```
Rigid Sort Action                                         1
Sort Human, Result                                        2
Fun birth/2, test/2 -> Action                             3
Fun spouseOf/1, motherOf/1, fatherOf/1 -> Human           4
Sensor Fun sf/Action -> Result                            5
Name Sally, Mia, Frank, Fred -> Human                     6
Name Yes, No -> Result                                    7
Var a -> Action                                           8
Var x, y, z -> Human                                      9
Var r -> Result                                           10

Real: spouseOf(Mia) = Frank                               11

KB: spouseOf(Mia) = Frank v                               12
    spouseOf(Mia) = Fred
KB: [] [a] motherOf(x) = y <->                            13
            a = birth(y,x) v
            a /= birth(y,x) ^ motherOf(x) = y
KB: [] [a] fatherOf(x) = y <->                            14
   ex z (a = birth(z,x) ^ spouseOf(z) = y) v
   fa z (a /= birth(z,x) ^ fatherOf(x) = y)
KB: [] sf(a) = r <->                                      15
      ex x ex y (a = test(x,y) ^
                 fatherOf(x) = y) ^ r = Yes v
      fa x fa y (a /= test(x,y) v
                 fatherOf(x) /= y) ^ r = No

REG [birth(Mia,Sally)]                                    16
                     K<0> (motherOf(Sally) = Mia)
REG [birth(Mia,Sally)]                                    17
                  M<1> (fatherOf(Sally) /= Frank)
REG [birth(Mia,Sally)]                                    18
                 K<1> (fatherOf(Sally) = Frank v
                       fatherOf(Sally) = Fred)
REG [birth(Mia,Sally)] [test(Sally,Fred)]                 19
                     K<1> fatherOf(Sally) = Frank
```

Listing 1: An encoding of the example from Section 2.3 in LIMBO's TUI. **REG** indicates regression; **K<k>** and **M<k>** are $\mathbf{K}_k$ and $\mathbf{M}_k$.

The LIMBO reasoning system provides some additional features and expressivity which we skipped in this paper to ease the presentation: formulas $\mathbf{M}_k\phi$ for sound but incomplete reasoning about what is considered possible (note that $\neg\mathbf{K}_k\neg\phi$ is unsound, as $\mathbf{K}_k\neg\phi$ is sound but incomplete knowledge), a conditional belief operator $\mathbf{B}_{k,l}\phi \Rightarrow \psi$ to say that if ϕ then *usually* also ψ [Schwering and Lakemeyer, 2016], introspection for all belief operators, and multiple sorts.

LIMBO is implemented as a C++ library and is freely available.[1] As an illustration, an encoding of the projection tasks from Section 2.3 in LIMBO's textual problem description language is given in Listing 1 (where line 18 uses the sound $\mathbf{M}_k\neg\phi$ instead of the potentially unsound $\neg\mathbf{K}_k\phi$).

[1]Code: www.github.com/schwering/limbo
Running example: www.cse.unsw.edu.au/~cschwering/limbo/tui#birth

4 Conclusion

We have introduced a logic for decidable reasoning about incomplete knowledge in the context of actions and sensing. The approach is based on the situation calculus and a theory of limited belief which controls the cost of reasoning by stratifying beliefs into levels. Compared to earlier approaches, our logic features additional expressivity due to functions and, in particular, parameterised actions. We moreover introduced a software system for reasoning about actions and belief, which implements this logic. Unlike other implementations of the situation calculus, our system avoids a closed-world assumption and includes expressivity of first-order logic.

This work is part of a bigger project that aims at bringing to life KR concepts. Next steps are to add knowledge base progression as an alternative approach to projection, and to evaluate the system in dynamic domains. Other interesting additions to the reasoning system are multi-agent belief as well as belief revision. We also plan to combine the system with the programming language GOLOG [Levesque *et al.*, 1997] and hope to employ the system in cognitive robotics. Practical applications will however require better runtime performance; we hope to transfer SAT technology such as clause learning to our theory and implementation of limited belief.

References

[Baader, 2003] Franz Baader. *The Description Logic Handbook: Theory, Implementation, and Applications.* 2003.

[Börger *et al.*, 1997] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem.* Springer, 1997.

[Claßen and Lakemeyer, 2009] Jens Claßen and Gerhard Lakemeyer. Tractable first-order Golog with disjunctive knowledge bases. In *Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning*, 2009.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2), 1993.

[Kowalski and Sergot, 1989] Robert Kowalski and Marek Sergot. A logic-based calculus of events. In *Foundations of knowledge base management.* 1989.

[Lakemeyer and Levesque, 2002] Gerhard Lakemeyer and Hector J. Levesque. Evaluation-based reasoning with disjunctive information in first-order knowledge bases. In *Proceedings of the Eighth Conference on Principles of Knowledge Representation and Reasoning*, 2002.

[Lakemeyer and Levesque, 2011] Gerhard Lakemeyer and Hector J. Levesque. A semantic characterization of a useful fragment of the situation calculus with knowledge. *Artificial Intelligence*, 175(1), 2011.

[Lakemeyer and Levesque, 2014] Gerhard Lakemeyer and Hector J. Levesque. Decidable reasoning in a fragment of the epistemic situation calculus. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2014.

[Levesque and Lakemeyer, 2008] Hector J. Levesque and Gerhard Lakemeyer. Cognitive robotics. In *Handbook of Knowledge Representation*. Elsevier, 2008.

[Levesque and Reiter, 1998] Hector J. Levesque and Ray Reiter. High-level robotic control: Beyond planning. In *AAAI Spring Symposium on Integrating Robotics Research*, volume 37, 1998.

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

[Levesque, 1984] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2), 1984.

[Liu *et al.*, 2004] Yongmei Liu, Gerhard Lakemeyer, and Hector J. Levesque. A logic of limited belief for reasoning with disjunctive information. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning*, 2004.

[McCarthy, 1959] John McCarthy. Programs with common sense. In *Proceedings of the Symposium on Mechanization of Thought Processes*. Her Majesty's Stationary Office, 1959.

[McCarthy, 1963] John McCarthy. Situations, actions, and causal laws. Technical report, Stanford University, 1963.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL—the planning domain definition language. Technical report, 1998.

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

[Robinson, 1950] Raphael M. Robinson. An essentially undecidable axiom system. In *Proceedings of the international Congress of Mathematics*, volume 1, 1950.

[Scherl and Levesque, 2003] Richard Scherl and Hector J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2), 2003.

[Schwering and Lakemeyer, 2016] Christoph Schwering and Gerhard Lakemeyer. Decidable reasoning in a first-order logic of limited conditional belief. In *Proceedings of the Twenty-Second European Conference on Artificial Intelligence*, 2016.

[Schwering *et al.*, 2017] Christoph Schwering, Gerhard Lakemeyer, and Maurice Pagnucco. Belief revision and projection in the epistemic situation calculus. *Artificial Intelligence*, 251, 2017.

[Schwering, 2017] Christoph Schwering. A reasoning system for a first-order logic of limited belief. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[Thielscher, 1998] Michael Thielscher. Introduction to the fluent calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(14), 1998.