Integrating Action Calculi and Description Logics

Conrad Drescher and Michael Thielscher

Department of Computer Science,
Dresden University of Technology
Nöthnitzer Str. 46, 01187 Dresden, Germany

Abstract. General action languages, like e.g. the Situation Calculus, use full classical logic to represent knowledge of actions and their effects in dynamic domains. Description Logics, on the other hand, have been developed to represent static knowledge with the help of decidable subsets of first order logic. In this paper, we show how to use Description Logic as the basis for a decidable yet still expressive action formalism. To this end, we use ABoxes as decidable state descriptions in the basic Fluent Calculus. As a second contribution, we thus obtain an independent semantics – based on a general action formalism – for a recent method for ABox-Update.

1 Introduction

General action languages like the Situation Calculus [1] or the Fluent Calculus [2] are highly expressive formalisms for representing knowledge of actions and effects in dynamic domains. In this way, they provide the formal foundations for programming languages and systems for the design of logically reasoning agents who can execute high-level strategies and solve planning problems [3]. However, the use of full classical logic as the basis for these calculi implies, in general, undecidability even of static questions such as whether the current state knowledge entails that a specific action is executable. The existing solutions to this problem often restrict the action calculi to being essentially propositional and/or employing the closed-world assumption. Description Logics, on the other hand, provide expressive but decidable languages for the representation of static knowledge. In particular, they are of far greater expressivity than propositional logic. Efficient decision procedures have been developed and implemented for a variety of such logics [4].

In this paper, we show how to integrate Description Logics into a general action formalism. Our motivation is two-fold: On the one hand, the integration allows to restrict the expressiveness of general reasoning about actions to expressive yet decidable fragments of first order logic. This also provides the formal foundations for integrating decision procedures for Description Logics into action programming languages and systems, which will allow agents to resort to these algorithms whenever they have to verify conditions against their state

J. Hertzberg, M. Beetz, and R. Englert (Eds.): KI 2007, LNAI 4667, pp. 68–83, 2007.

knowledge. On the other hand, the integration of Description Logics into an action language provides a semantics for a recent definition of ABox-Update [5], which is thus embedded into a general formalism for reasoning about actions and change.

The specific contributions of this paper are the following:

1. We show how ABoxes can be used as expressive, decidable state descriptions in the basic Fluent Calculus.
2. We provide semantics for ABox-Update by capturing them with Fluent Calculus state update axioms.
3. We lay the theoretical foundations for a practical action programming language built on top of Description Logic reasoners.

The rest of the paper is organized as follows: In Section 2, we recall the basics of the Fluent Calculus and give a brief introduction to Description Logics. In Section 3, we show how ABoxes can be used as state descriptions in the Fluent Calculus, and we prove that state update axioms provide a correct characterization of ABox-Update. Furthermore, we show how to integrate simple TBox reasoning and discuss some of the problems that arise in the general case. After a discussion of related work, we conclude with a summary and outlook.

2 Preliminaries

In this section, we introduce the general action formalism Fluent Calculus; we assume familiarity with the classical Situation Calculus. We then recall the very essentials of Description Logics.

2.1 Fluent Calculus

The Fluent Calculus is a general action formalism: it enables the axiomatization of dynamic domains, i.e. of initial knowledge about the world, action preconditions and action effects. As running example of a dynamic domain we will use the following simplistic online-store scenario; we will give a Fluent Calculus axiomatization of this scenario at the end of this section.

Example 1. Initially, all that is known is that customer John has ordered the item *NiceBook*. An order cancellation can be processed only if the order is known. If the order already has been paid for, the customer is entitled to a refund.

We refer to the mutable properties of a dynamic domain as the *fluents*. In the Situation Calculus fluents are modelled as first order atoms, extended by an additional argument for a point in time, e.g. $\mathrm{Ordered}(\mathrm{John}, \mathrm{NiceBook}, S_0)$. The Fluent Calculus extends the Situation Calculus with an explicit notion of a *state* associated with a situation, denoted $\mathrm{State}(S_0)$. Intuitively, a state may be identified with the set of all the fluents that hold at any one time. To this end *reification* is employed: both fluents and states are modelled as terms; cf. $\mathrm{Holds}(\mathrm{Ordered}(\mathrm{John}, \mathrm{NiceBook}), \mathrm{State}(S_0))$. This allows to apply first-order

quantification to fluents and states, which in turn is helpful for devising a solution to the famous Frame Problem. In the following we give a compact, formal introduction to the technical basics of Fluent Calculus and the axiom schemes employed to encode dynamic domains.

Basics of Fluent Calculus. Fluent Calculus is based on many-sorted classical logic with equality. The standard sorts are OBJECT, ACTION, SITUATION, FLUENT and STATE, with FLUENT a sub-sort of STATE.[1] A term of sort FLUENT is a *fluent* – analogously we speak of *states*, *situations*, *actions* and *objects*. Situations are sequences of actions rooted in an initial situation S_0 – e.g. Do(Order(NiceBook), S_0). Just as in the classical Situation Calculus, they provide a branching time structure for Fluent Calculus. At the heart of Fluent Calculus is an axiomatization of states representing combinations of fluents.

Definition 1 (basic signature). *The signature of Fluent Calculus contains:*

- *A countable infinity of function symbols into sort* OBJECT *and* FLUENT *– but only a finite number thereof into sort* ACTION.[2]
- *Two symbols for functions into sort situation:*
  - S_0 : SITUATION *— the initial situation.*
  - *Do* : ACTION × SITUATION → SITUATION *— mapping a situation to its successor, as the result of executing an action.*
- *Three symbols for functions into states:*
  - $\emptyset$: STATE *— the empty state.*
  - $\circ$: STATE × STATE → STATE *— for conjoining fluents into states and states into bigger states.*
  - *State* : SITUATION → STATE *— denoting the state of a situation.*
- *A binary predicate symbol Poss* : ACTION × SITUATION *— relating action preconditions to situations.*

To gain an intuition for the role played by $\circ$, compare Situation Calculus'

$$\text{Ordered}(\text{John}, \text{NiceBook}, S_0) \wedge \text{Ordered}(\text{Mary}, \text{EvenNicerBook}, S_0)$$

with Fluent Calculus'

$$(\exists z)\text{State}(S_0) = \text{Ordered}(\text{John}, \text{NiceBook}) \circ \text{Ordered}(\text{Mary}, \text{EvenNicerBook}) \circ z.$$

Definition 2 (holds macro). *A fluent f is said to hold in a state z if the latter is composed of f and some other state z' via $\circ$; a fluent holds in a situation if it holds in the state of the situation:*

$$\mathit{Holds}(f, z) \stackrel{def}{=} (\exists z')z = f \circ z' \mathit{and}$$
$$\mathit{Holds}(f, s) \stackrel{def}{=} \mathit{Holds}(f, \mathit{State}(s)).$$

[1] By convention, variables x,a,s,f and z are used for objects, actions, situations, fluents and states, respectively.

[2] Each with arguments of sort OBJECT only.

The foundational axioms Σ_{state} of the Fluent Calculus govern the behavior of states.

Definition 3 (foundational axioms). [3]

$$(z_1 \circ z_2) \circ z_3 = z_1 \circ (z_2 \circ z_3) \qquad (\textit{ Associativity })$$
$$z_1 \circ z_2 = z_2 \circ z_1 \qquad (\textit{ Commutativity })$$
$$\neg Holds(f, \emptyset) \qquad (\textit{ Empty state })$$
$$Holds(f_1, f_2) \supset f_1 = f_2 \qquad (\textit{ Irreducibility })$$
$$Holds(f, z_1 \circ z_2) \supset (Holds(f, z_1) \vee Holds(f, z_2)) \qquad (\textit{ Decomposition })$$
$$(\forall f)(Holds(f, z_1) \equiv Holds(f, z_2)) \supset z_1 = z_2 \qquad (\textit{ State equality })$$
$$(\forall P)(\exists z)(\forall f)(Holds(f, z) \equiv P(f)) \qquad (\textit{ State existence })$$

where P is a unary predicate variable of sort FLUENT.

The last axiom ensures the existence of a state for every combination of fluents. For a detailed introduction to this and the other axioms the interested reader is referred to [6].

Definition 4 (finite state). *A finite state ϑ is a term $f_1 \circ \ldots \circ f_n$ such that each f_i $(1 \leq i \leq n)$ is a fluent. If $n = 0$, then $\vartheta = \emptyset$.*

Definition 5 (fluent addition/subtraction). *The following macros provide an intuitive notation for describing relations between different states:*

- $z_1 + f \stackrel{def}{=} z_1 \circ f$
- $z_1 - f = z_2 \stackrel{def}{=} (z_2 = z_1 \vee z_2 \circ f = z_1) \wedge \neg Holds(f, z_2)$

These definitions are recursively extended to addition and subtraction of finite states ϑ^+ and ϑ^-: these will consist of the positive and negative effects of actions.

We next introduce formulas capable of expressing which (fluent or non-fluent) properties hold in a state and in a situation, respectively.

Definition 6 (state/situation formula). *A state formula $\Delta(z)$ is a first order formula with free state variable z and without any occurrences of states other than in expressions of the form $Holds(f, z)$, and without actions or situations. Replacing every occurrence of z by $State(s)$ in a state formula $\Delta(z)$, we obtain a situation formula $\Delta(s)$.*

Definition 7 (unique name axioms). *Every Fluent Calculus instance includes a set Σ_{una} of unique-name axioms that contains a formula of the form*

$$f_i(\boldsymbol{x}) \neq f_j(\boldsymbol{y})$$

[3] Variables not within the scope of any quantifier are to be read as universally quantified throughout this paper unless otherwise stated.

for each pair of distinct function symbols of sort FLUENT *as well as for each pair of distinct function symbols of sort* ACTION *and a formula of the form*

$$(f_i(\boldsymbol{x}) = f_i(\boldsymbol{y}) \supset \boldsymbol{x} = \boldsymbol{y}),$$

for each function symbol of sort FLUENT *or* ACTION.

Domain Specifications. In order to specify a dynamic domain we need to axiomatize knowledge about the initial state, action preconditions and the effects resulting from action execution. Formally, a domain is specified as a set of axioms $\Sigma = \Sigma_{\text{state}} \cup \Sigma_{\text{una}} \cup \Sigma_{\text{init}} \cup \Sigma_{\text{poss}} \cup \Sigma_{\text{sua}}$, where :

- $\Sigma_{\text{init}} = \{(\exists z)\text{State}(S_0) = z \wedge \Delta(z)\}$, with $\Delta(z)$ a state formula,
- Σ_{poss} is a set of precondition axioms, one for each action, and
- Σ_{sua} is a set of state update axioms, one for each action.

Definition 8 (precondition axiom). *A precondition axiom for action $A(\boldsymbol{x})$ is a formula $Poss(A(\boldsymbol{x}), s) \equiv \Delta(s)$, where $\Delta(s)$ is a situation formula with free variables among $\boldsymbol{x}$ and s.*

Definition 9 (state update axiom). *A state update axiom is a formula of the form*

$$\begin{aligned} &Poss(A(\boldsymbol{x}), s) \supset \\ &\qquad (\exists \boldsymbol{y}_1)(\Delta_1(s) \wedge State(Do(A(\boldsymbol{x}), s)' = State(s) - \vartheta_1^- + \vartheta_1^+) \\ &\qquad \vee ... \vee \\ &\qquad (\exists \boldsymbol{y}_n)(\Delta_n(s) \wedge State(Do(A(\boldsymbol{x}), s)' = State(s) - \vartheta_n^- + \vartheta_n^+). \end{aligned}$$

The finite states ϑ_i^- and ϑ_i^+ with free variables among $\boldsymbol{x}, \boldsymbol{y}_i$ are the negative and positive effects of $A(\boldsymbol{x})$ under condition $\Delta(s)$. $\Delta(s)$ itself is a situation formula with free variables among $\boldsymbol{x}, \boldsymbol{y}_i$ and s.

Example 1. *(continued) The online-store scenario from example 1 is axiomatized in Fluent Calculus as follows, illustrating each type of axiom:*

$$(\exists z)State(S_0) = z \wedge Holds(Ordered(John, NiceBook), z),$$

$$Poss(CancelOrder(customer, item), s) \equiv Holds(Ordered(customer, item), s),$$

$$\begin{aligned} &Poss(CancelOrder(customer, item), s) \supset \\ &\quad (Holds(Paid(item), s) \wedge State(Do(CancelOrder(customer, item), s)) = \\ &\qquad State(s) - Ordered(customer, item) + Refund(customer, item)) \\ &\quad \vee \\ &\quad (\neg Holds(Paid(item), s) \wedge State(Do(CancelOrder(customer, item), s)) = \\ &\qquad State(s) - Ordered(customer, item)). \end{aligned}$$

Name	Syntax	Semantics
negation	$\neg C$	$\mathfrak{D}^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{\ x \mid \exists y(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{\ x \mid \forall y(x,y) \in R^{\mathcal{I}} \supset y \in C^{\mathcal{I}}\}$

Fig. 1. Syntax and semantics of $\mathcal{ALC}$

This axiomatization entails

$$\neg Holds(Ordered(John, NiceBook), Do(CancelOrder(John, NiceBook), S_0))$$

and

$$Holds(Paid(NiceBook), Do(CancelOrder(John, NiceBook), S_0) \supset Holds(Refund(John, NiceBook), Do(CancelOrder(John, NiceBook), S_0)).$$

2.2 Description Logics

In this section, we recall those facts about Description Logics (DLs) that are essential to the ensuing discussion. A gentle introduction can be found in [4]. Description Logics are a family of Knowledge Representation formalisms; typically, they are decidable fragments of classical first order logic. In the following we employ the term Description Logic solely for such fragments.

A particular DL is based on a set of concept names N_C (unary predicates), a set of role names N_R (binary predicates), a set of individual names N_I (constants), and a number of constructors for inductively defining complex concepts and roles.

The semantics of Description Logics is defined via interpretations $\mathcal{I} = (\mathfrak{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\mathfrak{D}^{\mathcal{I}}$ is a non-empty set of individuals. The interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $C \in N_C$ to a subset $C^{\mathcal{I}}$ of $\mathfrak{D}^{\mathcal{I}}$, each role name $R \in N_R$ to a binary relation $R^{\mathcal{I}}$ on $\mathfrak{D}^{\mathcal{I}}$, and each individual name $I \in N_I$ to an individual $I^{\mathcal{I}} \in \mathfrak{D}^{\mathcal{I}}$. The semantics is extended inductively to complex concepts and roles. Figure 1 introduces the syntax and semantics of the core DL $\mathcal{ALC}$.

Definition 10 (ABox). *An assertional box (ABox) is a finite, non-empty set of concept assertions C(I) and role assertions $R(I_1, I_2)$ and $\neg R(I_1, I_2)$, where C and R may be complex concepts and roles, respectively.*

For example, {Outbound $\sqcup$ Delivered(Package)} is an ABox expressing uncertainty over the whereabouts of a particular package.

A number of highly-optimized tableau-based reasoners for effectively deciding even very expressive DLs are available [7,8,9].

3 Integration

We will now lay the theoretical foundations for an integration of Description Logics into Fluent Calculus. We will first show how the latter can use DL ABoxes as structured and decidable world descriptions. We then turn our attention to a recently proposed method for ABox-Update: After recalling the essential definitions we establish a Fluent Calculus semantics for these updates, thus relating them to standard AI action calculi. Furthermore, these findings will enable us to identify fragments of Fluent Calculus where questions of action applicability and effects resulting from action execution can effectively be computed.

3.1 ABoxes and State Formulas

We now establish a connection between Description Logic ABoxes and Fluent Calculus state formulas. This connection will be a consequence of a more general result on the relation between first order sentences and state formulas.

Consider an arbitrary countable first order language $\mathcal{L}$.[4] We can then define a Fluent Calculus instance that contains exactly one function symbol F_i of sort FLUENT for every predicate symbol $P_i \in \mathcal{L}$ (except equality). Moreover, its terms $\boldsymbol{t}$ of sort OBJECT are precisely the terms of $\mathcal{L}$.

Definition 11. *The mapping τ_z takes first order sentences in $\mathcal{L}$ to state formulas $\Delta(z)$:*

$$\begin{aligned}
\tau_z(P_i(\boldsymbol{t})) &= \mathit{Holds}(F_i(\boldsymbol{t}), z) \\
\tau_z(t_1 = t_2) &= (t_1 = t_2) \\
\tau_z(\varphi \wedge \psi) &= \tau_z(\varphi) \wedge \tau_z(\psi) \\
\tau_z(\neg\varphi) &= \neg\tau_z(\varphi) \\
\tau_z(\exists x\varphi) &= \exists x\tau_z(\varphi).
\end{aligned}$$

Theorem 1 (first order sentences and state formulas). *A first order sentence φ in a countable language $\mathcal{L}$ has a model iff $\{\tau_z(\varphi)\} \cup \Sigma_{state}$ has a model.*

Proof. ($\Rightarrow$)
First, observe that we can restrict our attention to certain models of φ, namely the term models obtained via the standard Henkin construction [10]. The domain $\mathfrak{D}$ of these models consists of equivalence classes on all the terms of $\mathcal{L}$. Let $\mathcal{M}_1 = (\mathfrak{D}_{\mathcal{M}_1}, \cdot^{\mathcal{M}_1}) \models \varphi$ be such a model. Let $\mathfrak{F}$ be the set of all fluents built from terms occurring in an equivalence class in $\mathfrak{D}_{\mathcal{M}_1}$.
Then $\mathcal{M}_2 = (\mathfrak{D}_{\text{object}}, \mathfrak{D}_{\text{fluent}}, \mathfrak{D}_{\text{state}}, \cdot^{\mathcal{M}_2}) \models \{\tau_z(\varphi)\} \cup \Sigma_{\text{state}}$ where

- $\mathfrak{D}_{\text{object}} = \mathfrak{D}_{\mathcal{M}_1}$,
- $\mathfrak{D}_{\text{fluent}} = \{\{f\} \mid f \in \mathfrak{F}\}$,
- $\mathfrak{D}_{\text{state}} = \mathcal{P}(\mathfrak{F})$, the power set of $\mathfrak{F}$,

[4] In the following we assume without loss of generality that $\mathcal{L}$ contains equality.

– $t^{\mathcal{M}_2} = t^{\mathcal{M}_1}$ for objects t,
– $F(\boldsymbol{t})^{\mathcal{M}_2} = \{F(\boldsymbol{t}^{\mathcal{M}_2}\}$ for each fluent $F(\boldsymbol{t})$,
– $\emptyset^{\mathcal{M}_2} = \{\}$,
– $(z_1 \circ z_2)^{\mathcal{M}_2} = (z_1)^{\mathcal{M}_2} \cup (z_2)^{\mathcal{M}_2}$, and
– $z^{\mathcal{M}_2} = \{\ F_i(\boldsymbol{t}^{\mathcal{M}_2}) \mid \mathcal{M}_1 \vDash P_i(\boldsymbol{t})\}$.

Interpreting $\emptyset$ as empty-set, $\circ$ as set-union and states as sets of fluents is a model of the foundational axioms Σ_{state} of Fluent Calculus [6]. The proof is completed by structural induction on φ.

($\Leftarrow$) In this case simply let $\mathcal{M}_3 = (\mathfrak{D}_{\text{object}}, \mathfrak{D}_{\text{fluent}}, \mathfrak{D}_{\text{state}}, \cdot^{M_3}) \vDash \{\tau_z(\varphi)\} \cup \Sigma_{\text{state}}$. Then $\mathcal{M}_4 = (\mathfrak{D}_{\text{object}}, \cdot^{\mathcal{M}_4}) \vDash \phi$ where

– $t^{\mathcal{M}_4} = t^{\mathcal{M}_3}$ for terms t of sort *object* and
– $P_i^{\mathcal{M}_4} = \{(\boldsymbol{t}^{\mathcal{M}_4}) \mid \mathcal{M}_3 \vDash \text{Holds}(F_i(\boldsymbol{t}), z)\}$.

This proof, too, is completed by structural induction on φ. □

This result justifies an intuitive identification of state formulas with the more familiar first order sentences. Moreover, it enables us to transfer known decidability or complexity results for fragments of first order logic to instances of the Fluent Calculus, where state formulas are restricted accordingly. In particular this applies to Description Logic ABoxes. Using ABoxes as state formulas, in an actual implementation we can resort to DL reasoners in order to decide static state knowledge, e.g. action preconditions. Researchers in DL have investigated a great number of DLs of varying strength; from these we can choose a logic that we deem appropriate for the task under consideration.

3.2 Updated ABoxes and State Update Axioms

In a recent paper, a method for updating Description Logic ABoxes has been proposed. Next we will briefly recall essential definitions and results; for in-depth coverage, the interested reader is referred to [5]. Subsequently, we will provide a Fluent Calculus semantics for ABox-Update, and thus relate the latter to a standard AI formalism.

ABox-Update. After introducing the syntactic objects describing an ABox-Update, we restate the semantic considerations underlying the whole approach.

Definition 12 (conditional ABox update). *A conditional update $\mathcal{U}$ is a finite, non-empty set of expressions φ/ψ, where the condition φ is an ABox assertion and the postcondition ψ is a concept/role literal. Consistency of the condition part φ_i for a number of expressions φ_i/ψ_i implies the consistency of their postconditions ψ_i. The condition part may be omitted by writing $\top/\psi$, where $\top$ abbreviates a tautology.*

The semantics of ABox-Update is defined using the possible models approach of Winslett [11]; that is, for every interpretation $\mathcal{I}$ we define an updated interpretation $\mathcal{I}'$. E.g., if $\mathcal{U} = \{\varphi_1/C(I_1), \varphi_2/\neg C(I_2)\}$ and $\mathcal{I}$ entails both φ_1 and φ_2, then

$\mathcal{I}'$ should interpret C as $\mathcal{I}$, but include the individual I_1 into the interpretation of C and exclude the individual I_2 from it. These should be the only changes to occur. The following definition captures this minimal change policy.

Definition 13 (conditional interpretation update). *Let $\mathcal{U}$ be a conditional update and $\mathcal{I}$, $\mathcal{I}'$ interpretations such that $\mathfrak{D}^{\mathcal{I}} = \mathfrak{D}^{\mathcal{I}'}$ and $\mathcal{I}$ and $\mathcal{I}'$ agree on the interpretation of individual names. Then $\mathcal{I}'$ is the result of updating $\mathcal{I}$ with $\mathcal{U}$, written $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$, if the following holds for all concept names $C \in N_C$ and role names $R \in N_R$:*

$$\begin{aligned} C^{\mathcal{I}'} = (C^{\mathcal{I}} \cup \{\ I^{\mathcal{I}} \mid \varphi/C(I) \in \mathcal{U} \wedge \mathcal{I} \vDash \varphi\}) \\ \setminus \{\ I^{\mathcal{I}} \mid \varphi/\neg C(I) \in \mathcal{U} \wedge \mathcal{I} \vDash \varphi\} \text{and} \\ R^{\mathcal{I}'} = (R^{\mathcal{I}} \cup \{\ (I_1^{\mathcal{I}}, I_2^{\mathcal{I}}) \mid \varphi/R(I_1, I_2) \in \mathcal{U} \wedge \mathcal{I} \vDash \varphi\}) \\ \setminus \{\ (I_1^{\mathcal{I}}, I_2^{\mathcal{I}}) \mid \varphi/\neg R(I_1, I_2) \in \mathcal{U} \wedge \mathcal{I} \vDash \varphi\}. \end{aligned}$$

Let $\mathcal{M}(\mathcal{A})$ denote the set of all models of an ABox $\mathcal{A}$.

Definition 14 (updated ABox). *For an ABox $\mathcal{A}$ and a conditional update $\mathcal{U}$ the updated ABox $\mathcal{A}'$ is defined model-theoretically such that:*

$$\mathcal{M}(\mathcal{A}') = \{\ \mathcal{I}' \mid \mathcal{I} \in \mathcal{M}(\mathcal{A})\ \wedge \mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'\}.$$

For applying a conditional update $\mathcal{U}$ to an ABox $\mathcal{A}$ resulting in ABox $\mathcal{A}'$ we also write $\mathcal{A}' = \mathcal{A} * \mathcal{U}$. In spite of some negative results in [5] it has been established for a whole range of DLs that these admit ABox-Update; i.e. for arbitrary $\mathcal{A}$ and $\mathcal{U}$ the updated ABox $\mathcal{A}' = \mathcal{A} * \mathcal{U}$ always exists. For the DLs ranging from $\mathcal{ALCO}^{@}$ to $\mathcal{ALCQIO}^{@}$– which are closely related to the familiar $\mathcal{SHOIN}(D)$underlying the Ontology Web Language (OWL) – algorithms for computing updated ABoxes have been presented. For an updated ABox $\mathcal{A}' = \mathcal{A} * \mathcal{U}$ there are polynomials p_1, p_2 and q such that

- $|\mathcal{A}'| \leq 2^{p_1(|\mathcal{A}|)} \cdot 2^{p_2(|\mathcal{U}|)}$ and
- $\mathcal{A}'$ is computed in time $q(|\mathcal{A}'|)$.

For repeated updates the final ABox can be exponential only in the size of the original ABox and the total size of all updates.

The authors of [5] also propose two mechanisms for obtaining smaller updated ABoxes; in both cases the result of updating is exponential only in the size of the update. One is based on introducing abbreviations for some complex concepts. The other eliminates the asymmetry between concepts and roles typically found in DLs: it introduces powerful operators on roles. Update algorithms for such DLs are also given; the strongest DL under consideration is as expressive as the two variable fragment of first order logic with counting quantifiers [12].

Fluent Calculus Semantics for ABox-Update. We will now establish a Fluent Calculus semantics for any DL that is both embeddable into first order logic and closed under the above definition of update. To do so, for a given DL, ABoxes $\mathcal{A}$, $\mathcal{A}'$ and update $\mathcal{U}$ with $\mathcal{A}' = \mathcal{A} * \mathcal{U}$, we will define a corresponding

domain axiomatization Σ in a suitable Fluent Calculus instance. We will then prove that for every model of Σ there are models $\mathcal{I}$ and $\mathcal{I}'$ of $\mathcal{A}$ and $\mathcal{A}'$ satisfying $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$ and vice versa.

First, we associate with $\mathcal{U}$ the name Update. The Fluent Calculus instance is defined such that

- it contains exactly one action, namely Update, and
- there is a bijection between
  - the objects and the individual names N_I, and
  - the function symbols of sort FLUENT and the union of the concept and role names, $N_C \cup N_R$.

Next, since we consider only first order embeddable DLs, we can clearly define a mapping τ_z from ABoxes to state formulas $\Delta(z)$, analogously to the mapping from Definition 11; similarly, τ_s maps ABoxes to situation formulas. In the domain axiomatization Σ to be constructed, let

$$\Sigma_{\text{init}} = \{(\exists z)\text{State}(S_0) = z \wedge \tau_z(\mathcal{A})\}.$$

We now turn to the construction of a state update axiom corresponding to the update $\mathcal{U} = \{\varphi_1/\psi_1, \ldots, \varphi_n/\psi_n\}$. Define the set $\mathcal{E}_1 = \{\varphi_i/\psi_i \mid \varphi_i/\psi_i \in \mathcal{U}\} \cup \{\neg\varphi_i/\text{nil} \mid \varphi_i/\psi_i \in \mathcal{U}\}$ and let $\mathcal{E}_2$ be the set of all subsets of $\mathcal{E}_1$ that are maximally consistent with regard to the condition part φ_i. Note that $\mathcal{E}_2$ will be exponential in the size of $\mathcal{U}$. For every member $\mathcal{E}_3$ of $\mathcal{E}_2$ we form an update formula

$$\gamma(s) \stackrel{def}{=} \Delta(s) \wedge (\exists z)\text{State}(\text{Do}(\text{Update}), s) = \text{State}(s) - \vartheta^- + \vartheta^+$$

where

- $\Delta(s)$ denotes the conjunction of all the situation formulas in the set $\{\tau_s(\varphi) \mid \varphi/\psi \in \mathcal{E}_3 \vee \varphi/\text{nil} \in \mathcal{E}_3\}$, and
- ϑ^+ (respectively, ϑ^-) denotes the finite state consisting of the ground fluents corresponding to the assertions ψ such that $\varphi/\psi \in \mathcal{E}_3$ (respectively, $\varphi/\neg\psi \in \mathcal{E}_3$). [5]

Then Σ contains the single state update axiom

$$\Sigma_{\text{sua}} = \{\text{Poss}(\text{Update}, s) \supset \Gamma(s)\},$$

where $\Gamma(s)$ denotes the disjunction of all the $\gamma(s)$ resulting from the above construction. Observe that all the $\gamma(s)$ are mutually exclusive.

Example 2. Consider the update
$\mathcal{U} = \{\top/\neg\text{Ordered}(\text{John}, \text{NiceBook}), \text{Paid}(\text{NiceBook})/\text{Refund}(\text{John}, \text{NiceBook})\}$.

[5] If there is no such assertion we obtain the empty state $\emptyset$.

The above construction yields – after a little simplification –

$$\begin{aligned}&\mathrm{Poss}(\mathrm{Update}, s) \supset \\ &\quad (\mathrm{Holds}(\mathrm{Paid}(\mathrm{NiceBook}), s) \wedge \mathrm{State}(\mathrm{Do}(\mathrm{Update}, s)) = \\ &\qquad \mathrm{State}(s) - \mathrm{Ordered}(\mathrm{John}, \mathrm{NiceBook}) + \mathrm{Refund}(\mathrm{John}, \mathrm{NiceBook})) \\ &\quad \vee \\ &\quad (\neg\mathrm{Holds}(\mathrm{Paid}(\mathrm{NiceBook}), s) \wedge \mathrm{State}(\mathrm{Do}(\mathrm{Update}, s)) = \\ &\qquad \mathrm{State}(s) - \mathrm{Ordered}(\mathrm{John}, \mathrm{NiceBook})).\end{aligned}$$

Finally, we define the set of precondition axioms to be

$$\Sigma_{\mathrm{poss}} = \{\mathrm{Poss}(\mathrm{Update}, s) \equiv \top\},$$

completing the definition of Σ.

Before stating the main theorem, we recall a fundamental result about Fluent Calculus [6] that will be essential to our discussion.

Theorem 2 (fluent calculus foundational theorem). *Let ϑ^+ and ϑ^- be two finite states. Then foundational axioms Σ_{state} together with $z' = z - \vartheta^- + \vartheta^+$ entail*

$$\begin{aligned}Holds(f, z') \equiv\ & Holds(f, \vartheta^+) \\ & \vee \\ & Holds(f, z) \wedge \neg Holds(f, \vartheta^-).\end{aligned}$$

Theorem 3 (fluent calculus semantics for ABox-Update). *For an ABox $\mathcal{A}$, an update $\mathcal{U}$ and the corresponding domain axiomatization Σ it holds that $\mathcal{A}$ has a model $\mathcal{I}$ with $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$ if and only if Σ has a model. Moreover, in a model of Σ, $State(S_0)$ and $State(Do(Update, S_0))$ relate in the same way as $\mathcal{I}$ and $\mathcal{I}'$.*

Proof. ($\Rightarrow$)
We will only give a sketch of the proof. As in the proof of Theorem 1 we can restrict our attention to Henkin-style term interpretations: When constructing $\tau_z(\mathcal{A})$ we simultaneously construct the first order representation of $\mathcal{A}$, using the same variable names. A term model of this is readily turned into a model of $\mathcal{A}$. We then interpret the objects by their equivalence classes, and fluents by fluent terms built from these equivalence classes as in the proof of Theorem 1. We extend this treatment to situations and actions: here we restrict the respective universes to the set of ground situations and actions built using only terms of sort OBJECT occurring in an equivalence class. Interpreting $\circ$ and $\emptyset$ as set union and empty set as before, we fix the interpretation of $\mathrm{State}(S_0)$ as the set of fluents corresponding to atoms that are true in $\mathcal{I}$. We observe that, once we have fixed the interpretation of $\mathrm{State}(S_0)$, the model of Σ is uniquely determined, due to Theorem 2 and the fact that the conditions $\Delta(s)$ in the state update axiom are mutually exclusive. Theorem 2 is also the key to proving that $\mathrm{State}(S_0)$ and $\mathrm{State}(\mathrm{Do}(\mathrm{Update}, S_0))$ are related in the same way as $\mathcal{I}$ and $\mathcal{I}'$.

($\Leftarrow$)
Let $\mathcal{M}_2 = (\mathfrak{D}_{\text{object}}, \mathfrak{D}_{\text{fluent}}, \mathfrak{D}_{\text{state}}, \mathfrak{D}_{\text{situation}}, \mathfrak{D}_{\text{action}}, \cdot^{\mathcal{M}_2}) \vDash \Sigma$.
Set $\mathcal{I}_3 = (\mathfrak{D}_{\text{object}}, \cdot^{\mathcal{I}_3})$, $\mathcal{I}_4 = (\mathfrak{D}_{\text{object}}, \cdot^{\mathcal{I}_4})$ where

- $P_i^{\mathcal{I}_3} = \{(\boldsymbol{t}^{\mathcal{M}_2}) \mid \mathcal{M}_2 \vDash \text{Holds}(F_i(\boldsymbol{t}), \text{State}(S_0))\}$,
- $P_i^{\mathcal{I}_4} = \{(\boldsymbol{t}^{\mathcal{M}_2}) \mid \mathcal{M}_2 \vDash \text{Holds}(F_i(\boldsymbol{t}), \text{State}(\text{Do}(\text{Update}, S_0)))\}$ and
- $\cdot^{\mathcal{I}_3}$, $\cdot^{\mathcal{I}_4}$ and $\cdot^{\mathcal{M}_2}$ agree on sort OBJECT.

Then $\mathcal{I}_3 \vDash \mathcal{A}$ and $\mathcal{I}_3 \Longrightarrow_{\mathcal{U}} \mathcal{I}_4$. □

Figure 2 depicts the relationship just established.

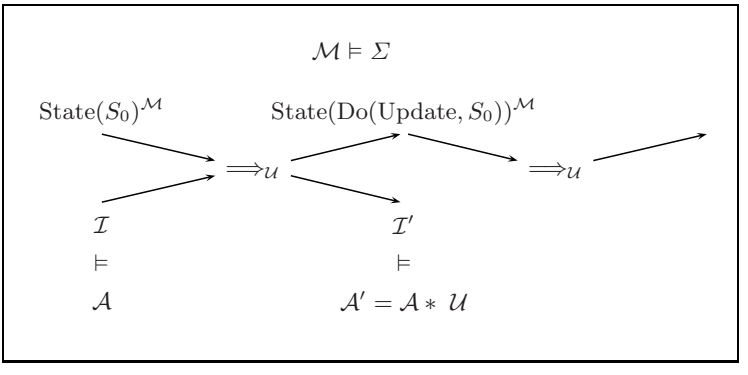

Fig. 2. Fluent Calculus semantics for ABox-Update

This result has two important consequences: On the one hand, by establishing a Fluent Calculus semantics for ABox-Update, it relates the latter to an established, general action formalism. On the other hand, it provides the formal underpinnings of using the update algorithms of [5] for computing updated states in a Fluent Calculus that uses ABoxes as state descriptions. Using an accordingly restricted Fluent Calculus instead of plain ABox-Update the notion of update resides within the language instead of being meta-logical.

3.3 TBoxes and Domain Constraints

The reader already familiar with Description Logics may wonder why we have not yet mentioned TBoxes. By allowing the definition of concepts in terms of other concepts, these contribute considerably to the expressive power of DLs.

Definition 15 (TBox/knowledge base). *$C \equiv D$ is a concept definition, where C is a defined concept name and D is a complex concept. A TBox $\mathcal{T}$ is a finite set of concept definitions. An interpretation $\mathcal{I}$ satisfies a concept definition $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ satisfies a TBox $\mathcal{T}$, if it satisfies all concept definitions in $\mathcal{T}$. A Knowledge Base is a pair $KB = (\mathcal{T}, \mathcal{A})$, with TBox $\mathcal{T}$ and ABox $\mathcal{A}$.*

A TBox $\mathcal{T}$ is a *terminology* if every defined concept is defined only once. A defined concept name C *directly uses* a concept name D if D occurs on the right hand side of the concept definition. A terminology is acyclic if no concept name is connected with itself via the transitive closure of *directly uses*. Reasoning in a knowledge base $KB = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is an acyclic terminology, can always be reduced to reasoning wrt. the empty TBox by unfolding the definitions [4].

For example, in our online-store scenario we can introduce the concept of a good customer with the help of the TBox

$$\{\text{GoodCustomer} \equiv \text{PurchasedManyItems} \sqcap \text{PaidOnTime}\}.$$

In action formalisms the concept of a domain constraint allows to state general knowledge and laws that have to be satisfied by every world state.

Definition 16 (domain constraint). *In Fluent Calculus a domain constraint is a formula of the form* $(\forall s)\Delta(s)$*, where* $\Delta(s)$ *is a situation formula.*

TBoxes are captured neatly by appropriate domain constraints. E.g. we map the above TBox to

$$(\forall s.\forall x)\text{Holds}(\text{GoodCustomer}(x), s) \equiv \\ \text{Holds}(\text{PurchasedManyItems}(x), s) \wedge \text{Holds}(\text{PaidOnTime}(x), s).$$

It is trivial, but potentially useful, to admit acyclic TBoxes. We can faithfully apply the update algorithms from [5] to an ABox serving as world state description after unfolding the TBox, resulting in a potentially exponential blowup. However, in the ABoxes that serve as action preconditions in a domain axiomatization, we can admit defined concepts without unfolding them into the ABox. This is possible since the semantics of the undefined concepts uniquely determines the semantics of the defined ones. The above result on the semantic correspondence between ABox-Update and Fluent Calculus state update axioms can be extended to take acyclic TBoxes into account.

If we admit general TBoxes, semantic problems arise. The semantics of the undefined concepts no longer uniquely determines the semantics of the TBox. As a consequence the one-to-one relation between original and updated interpretation – that is at the heart of ABox-Update – can not be maintained. This issue is well known to researchers in action formalisms as the Ramification Problem [13]. Considerable effort went into singling out intended interpretations, usually by appealing to some notion of causality [14,15,16]. This work should prove helpful when extending the definition of interpretation update.

4 Summary

4.1 Related Work

Recently, a number of works have addressed the issue of finding a decidable yet expressive logical framework for reasoning about actions and change. In the

following we will relate our work to other DL-based approaches. Such approaches have continued to attract considerable interest, not least since Description Logics form the foundation of the Semantic Web, and a dynamic view of the web is intuitively very appealing.

In [17] de Giacomo et al. show that DL-Lite is closed under update in the above sense; they also present a polynomial algorithm for computing updated ABoxes. The Description Logic DL-Lite is of reduced expressivity, but admits tractable reasoning and updated ABoxes of polynomial size. They also address updates in the presence of general TBoxes. If the models of the update and the general TBox have an empty intersection their algorithm guarantees correctness; otherwise it returns with an error. Our framework can also be instantiated with DL-Lite ABoxes; returning an error is not an option for an autonomous agent.

Liu et al. [18] provide an in-depth discussion of the semantic problems that arise when updating ABoxes in the presence of general TBoxes. They also observe that these problems are closely related to the ramification problem. As a solution they propose to provide the domain axiomatizer with a syntactic means to indicate which assertions may fluctuate freely during the update.

Baader et al. [19] is another work on DL-based reasoning about action and change. They employ reasoning similar to regression and among many other results, they outline how their work can be regarded as an instance of the Situation Calculus. Gu and Soutchanski [20] directly define a modified Situation Calculus, based on a DL with role operators that is equally expressive as $\mathcal{C}^2$. They adapt regression from the general Situation Calculus to their setting extended with acyclic TBoxes. They address the problem of using progression, i.e. update, instead of regression in [21]. To this end, since fluents are not reified in the Situation Calculus, they have to appeal to second order logic. An in-depth comparison of their work will be subject of future work. The fact that Situation Calculus and Fluent Calculus semantically agree has been shown in [22].

Employing existing DL reasoners we have to start reasoning from scratch after each update. In [23] the problem of incremental maintenance of a solver state is addressed under a very simple semantics for ABox-Update. It would be nice to extend these ideas to updates under the possible models approach.

4.2 Conclusion

We have shown how to integrate Description Logics into a general action formalism. We have thus restricted the latter to a decidable, yet expressive fragment of classical first order logic. To do so, we have proved that ABoxes can serve as a faithful substitute for state formulas in Fluent Calculus. Moreover, by proving that Fluent Calculus state update axioms correctly capture ABox-Update, we have related the latter to established research in reasoning about action and change. Our work lays the theoretical foundations for an integration of DL reasoning and update algorithms into a practical agent programming language. There are a number of interesting open issues for future work:

– Applying existing solutions to the ramification problem to handle ABox-Update in the presence of general TBoxes.

– Integrating inference algorithms for Description Logic problems into a general action programming language, like e.g. FLUX [2].

References

1. McCarthy, J.: Situations, actions, and causal laws. Technical Report AIM-2, AI Project, Stanford University (1963)
2. Thielscher, M.: FLUX: A logic programming method for reasoning agents. Theory and Practice of Logic Programming 5, 533–565 (2005)
3. Lespérance, Y., Levesque, H.J., Lin, F.D., Marcu, R.R., Scherl, R.B.: A logical approach to high-level robot programming—A progress report. In: Papers from the 1994 AAAI Fall Symposium, AAAI (1994)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
5. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) KR, pp. 46–56. AAAI Press (2006)
6. Thielscher, M.: Reasoning Robots: The Art and Science of Programming Robotic Agents. Applied Logic Series, vol. 33. Kluwer Academic Publishers, Dordrecht (2005)
7. Sirin, E., Parsia, B.: Pellet: An OWL DL reasoner. In: Proceedings of the 2004 International Workshop on Description Logics (DL2004) (2004)
8. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
9. Haarslev, V., Möller, R.: Racer system description. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 701–705. Springer, Heidelberg (2001)
10. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press, London (1972)
11. Winslett, M.: Reasoning about action using a possible models approach. In: aaai88, pp. 89–93 (1988)
12. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. Journal of Logic, Language, and Information 14, 369–395 (2005)
13. Ginsberg, M.L., Smith, D.E.: Reasoning about action II: the qualification problem. Artificial Intelligence 35, 311 (1988)
14. Lin, F.: Embracing causality in specifying the indirect effects of actions. In: Mellish, C.S. (ed.) Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, pp. 1985–1991. Morgan Kaufmann, San Francisco (1995)
15. Thielscher, M.: Ramification and causality. Artificial Intelligence Journal 89, 317–364 (1997)
16. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence 153, 49–104 (2004)
17. Giacomo, G.D., Lenzerini, M., Poggi, A., Rosati, R.: On the update of description logic ontologies at the instance level. In: Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI 2006) (2006)
18. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Description logic actions with general TBoxes: a pragmatic approach. In: Proceedings of the 2006 International Workshop on Description Logics (DL2006) (2006)

19. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proceedings of AAAI-05 (2005)
20. Gu, Y., Soutchanski, M.: A logic for decidable reasoning about services. In: Proceedings of the 4th International Workshop on AI for Service Composition (ECAI 2006) (2006)
21. Gu, Y., Soutchanski, M.: Decidable reasoning in a modified situation calculus. In: Proceedings of International Joint Conference on AI (IJCAI 2007) (2007)
22. Schiffel, S., Thielscher, M.: Reconciling situation calculus and fluent calculus. In: Proceedings of AAAI-06, Boston, MA, pp. 287–292. AAAI Press (2006)
23. Halaschek-Wiener, C., Parsia, B., Sirin, E., Kalyanpur, A.: Description logic reasoning for dynamic aboxes. In: Proceedings of the 2006 International Workshop on Description Logics (DL2006) (2006)