J Autom Reasoning (2012) 49:1–52
DOI 10.1007/s10817-010-9210-1

A Family of Dynamic Description Logics for Representing and Reasoning About Actions

Liang Chang · Zhongzhi Shi · Tianlong Gu · Lingzhong Zhao

Received: 6 January 2009 / Accepted: 18 November 2010 / Published online: 15 December 2010

Abstract Description logics provide powerful languages for representing and reasoning about knowledge of static application domains. The main strength of description logics is that they offer considerable expressive power going far beyond propositional logic, while reasoning is still decidable. There is a demand to bring the power and character of description logics into the description and reasoning of dynamic application domains which are characterized by actions. In this paper, based on a combination of the propositional dynamic logic PDL, a family of description logics and an action formalism constructed over description logics, we propose a family of dynamic description logics $DDL(X^{@})$ for representing and reasoning about actions, where X represents well-studied description logics ranging from the

This work was partially supported by the National Natural Science Foundation of China (Nos. 60903079, 60775035, 60963010, 60803033, 61035003), the National Basic Research Program of China (No. 2007CB311004), the State Key Laboratory of Software Engineering (SKLSE) and the 111 Project (No. B07037).

L. Chang (✉) · T. Gu · L. Zhao
School of Computer and Control, Guilin University of Electronic Technology, Guilin, 541004, China
e-mail: changl@guet.edu.cn

T. Gu
e-mail: cctlgu@guet.edu.cn

L. Zhao
e-mail: zhaolingzhong@guet.edu.cn

L. Chang
State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, 430072, China

Z. Shi
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704-28, Beijing, 100080, China
e-mail: shizz@ics.ict.ac.cn

$\mathcal{ALCO}$ to the $\mathcal{ALCHOIQ}$, and $X^@$ denotes the extension of X with the @ constructor. The representation power of $DDL(X^@)$ is reflected in four aspects. Firstly, the static knowledge of application domains is represented as RBoxes and acyclic TBoxes of the description logic X. Secondly, the states of the world and the pre-conditions of atomic actions are described by ABox assertions of the description logic $X^@$, and the post-conditions of atomic actions are described by primitive literals of $X^@$. Thirdly, starting with atomic actions and ABox assertions of $X^@$, complex actions are constructed with regular program constructors of PDL, so that various control structures on actions such as the "Sequence", "Choice", "Any-Order", "Iterate", "If-Then-Else", "Repeat-While" and "Repeat-Until" can be represented. Finally, both atomic actions and complex actions are used as modal operators for the construction of formulas, so that many properties on actions can be explicitly stated by formulas. A tableau-algorithm is provided for deciding the satisfiability of $DDL(X^@)$-formulas; based on this algorithm, reasoning tasks such as the realizability, executability and projection of actions can be effectively carried out. As a result, $DDL(X^@)$ not only offers considerable expressive power going beyond many action formalisms which are propositional, but also provides decidable reasoning services for actions described by it.

Keywords Description logic · Dynamic description logic · Action theory · Satisfiability-checking algorithm · Reasoning tasks

1 Introduction

Description logics (DLs) are well-known for representing and reasoning about knowledge of static application domains. They are playing a central role in the Semantic Web [5], serving as the basis of the W3C-recommended Web ontology language OWL [21, 39]. The main strength of description logics is that they offer considerable expressive power going far beyond propositional logic, while reasoning is still decidable [3].

There is a natural trend to bring the power and character of DLs into the description and reasoning of dynamic application domains which are characterized by actions. The study of integrating DLs with action formalisms is driven by two factors. One is the expressive gap between well-studied action formalisms: they are either based on first- or higher-order logics and do not admit decidable reasoning, like the Situation Calculus [28, 36] and the Fluent Calculus [40], or are decidable but only propositional, like those based on propositional dynamic logics or propositional temporal logics [6, 9, 16]. The other factor is the target of the semantic Web services [29]: to enable automatic discovery, composition, invocation and interoperation of Web services, by describing Web services' capabilities and contents in an unambiguous and computer-interpretable language; towards this target, an obvious concern is to combine in some way the static knowledge provided by ontologies on the Semantic Web with the dynamic descriptions of the computations provided by Web services [7, 26].

An action formalism constructed over DLs of the $\mathcal{ALCOIQ}$ family was proposed by Baader et al. [4]. In that formalism, acyclic TBoxes and ABoxes of DLs are used to specify the domain constraints and the states of the world respectively. Each

atomic action is described by a triple $(pre, occ, post)$, where pre is a finite set of ABox assertions for specifying the pre-conditions, occ is a finite set of occlusions for indicating some primitive literals which might change arbitrarily as while as the action is executed, and $post$ is a finite set of conditional post-conditions of the form φ/ψ, where φ is an ABox assertion and ψ is a primitive literal. The semantics of atomic actions is defined by means of transition relations on DL-interpretations, where each transition relation is restricted by the minimal-change semantics. Taking each finite sequence of atomic actions as a composite action, Baader et al. investigated the executability problem and the projection problem of actions. It was shown that both of these problems could be reduced to standard inference problems of description logics and therefore were decidable still. Following Baader et al's work, Liu et al. [24] proposed an approach to incorporate general TBoxes into the action formalism; Miličić [30, 31] investigated the planning problem and demonstrated that the plan existent problem was still decidable.

With Baader et al.'s formalism, an atomic action (or a simple Web service) named $buyBook_{a,b}$ might be described as a triple $buyBook_{a,b} = (pre, occ, post)$, where

- $pre = \{customer(a), book(b)\}$,
- $occ = \{\}$, and
- $post = \{instore(b)/bought(a, b), instore(b)/\neg instore(b)\}$.

This description states that the action $buyBook_{a,b}$ is applicable if a is a customer and b is a book; moreover, if b is in store before executing the action, then the result of the execution is that a has bought b and b is not in store any more. Concepts occurring in this description could be further specified by concept definitions; for example, the concept *customer* might be specified as follows:

$$customer \equiv person \sqcap \exists holds.creditCard$$

which states that each customer is a person holding a credit card.

Baader et al.'s formalisms [4, 24, 30] provide considerable expressive power for describing actions and Web services; they also provide desirable computational properties such as decidability, soundness and completeness of deduction procedures. However, a common limitation of them is that atomic actions can only be organized as finite sequences; many complex control structures on actions, such as the "Choice", "Any-Order", "Iterate", "If-Then-Else", "Repeat-While" and "Repeat-Until" structures specified in the OWL-based Web service ontology OWL-S [27], are not supported. Therefore, in order to describe and reason about complex compositions of Web services [33], there is a demand to enhance Baader et al.'s formalisms with more control structures.

In this paper, by embracing Baader et al.'s action formalisms into a dynamic logic, we propose a family of dynamic description logics $DDL(X^@)$ for representing and reasoning about actions, where X denotes the description logics ranging from the $\mathcal{ALCO}$ to the $\mathcal{ALCHOIQ}$, and $X^@$ is an extension of X with the @ constructor [23].

The logic $DDL(X^@)$ can be treated as a combination of the propositional dynamic logic PDL [14, 15, 35], the description logic $X^@$, and the action formalism proposed by Baader et al. [4]. Firstly, the syntax of roles, concepts, RBoxes, TBoxes and ABoxes of $DDL(X^@)$ are the same with those of the description logic $X^@$, with the exception that the @ constructor is not allowed in TBoxes. Secondly, the syntax of $DDL(X^@)$-formulas is similar to the syntax of PDL-formulas, except that

the propositions in PDL-formulas are replaced here by ABox assertions. Finally, the syntax of actions is the same in both PDL and $DDL(X^@)$; however, each atomic action in $DDL(X^@)$ will be further specified by a triple $(pre, occ, post)$, thus preserving Baader et al.'s action formalism. From the point of view of knowledge reasoning, a feature of $DDL(X^@)$ is that many inference problems on actions such as the realizability problem, the executability problem, and the projection problem, can be reduced to the satisfiability problem which is equipped with tableau decision algorithms. Therefore, $DDL(X^@)$ provides a family of powerful languages for representing and reasoning about dynamic application domains.

It should be noted that the minimal description logic considered in $DDL(X^@)$ is the logic $\mathcal{ALCO}^@$. The reason is that both the "nominals" and the "@" constructor are needed in our algorithms for deciding the satisfiability of formulas. However, from the point of view of knowledge representation, any sublanguage of the description logic $\mathcal{ALCHOIQ}^@$, such as the $\mathcal{ALC}$ and the $\mathcal{ALCO}$, can be used in $DDL(X^@)$ for the description of static domain knowledge.

For the simplicity of presentation, when the logic $DDL(X^@)$ is presented for the first time, each atomic action in it is just specified as a tuple (P, E), where P is a finite set of ABox assertions for describing pre-conditions, and E is a finite set of primitive literals for describing unconditional post-conditions. We will develop a tableau algorithm for the logic in such a case, and then extend both the logic and the reasoning mechanisms to support occlusions and conditional post-conditions in the description of atomic actions.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to description logics used in $DDL(X^@)$. Section 3 presents the syntax and semantics of $DDL(X^@)$, and gives an example to illustrate its expressive power. In Section 4, reasoning tasks on the knowledge described by $DDL(X^@)$ are formally defined; it is demonstrated that three primary reasoning tasks on actions can be reduced to the satisfiability problem on formulas. Section 5 provides a tableau algorithm for deciding the satisfiability of $DDL(X^@)$-formulas; the termination, soundness and completeness of this algorithm are proved, and the complexity of it is investigated. In Section 6, both the logic $DDL(X^@)$ and the reasoning mechanisms are extended to support occlusions and conditional post-conditions in the description of atomic actions, so that the logic is compatible with Baader et al.'s action formalism. Section 7 investigates the relationship between $DDL(X^@)$ and some known formalisms, and Section 8 concludes the paper.

2 Description Logics

As a kind of languages for knowledge representation, the intuition of description logics is to define concepts of a domain and then use these concepts to specify properties of objects and individuals occurring in the domain. Primitive symbols of description logics are a set N_R of *role names*, a set N_C of *concept names*, and a set N_I of *individual names*. Starting from these symbols, each description logic provides a set of *constructors* to form roles and complex concepts.

$\mathcal{ALC}$ (Attributive Language with Complements) is one of the most influential description logics [37]. It provides negation, conjunction, disjunction, existential restriction and value restriction constructors for the construction of concepts. To

meet the needs of applications that require more expressivity, various extensions of $\mathcal{ALC}$ were proposed by adding constructors into it. The availability of additional constructors is usually indicated by concatenating the corresponding letters [4]; for example, $\mathcal{H}$ stands for role hierarchies, $\mathcal{Q}$ stands for number restrictions, $\mathcal{I}$ stands for inverse roles, and $\mathcal{O}$ stands for nominals. In $DDL(X^{@})$, a special constructor denoted by "@" will be used; this constructor is known from hybrid logic [1, 23] and is slightly non-standard in description logics.

Constructors used in the description logics ranging from $\mathcal{ALC}$ to $\mathcal{ALCHOIQ}^{@}$ are listed in Table 1, where $A_i \in N_C$, $R_i \in N_R$, $p \in N_I$, and n is a non-negative integer.

The semantics of concepts and roles constructed in description logics are defined in terms of an *interpretation* $I = (\Delta^I, \cdot^I)$, where Δ^I is a non-empty set composed of individuals, and $\cdot^I$ is an interpretation function which maps each concept name $A_i \in N_C$ to a set $A_i^I \subseteq \Delta^I$, maps each role name $R_i \in N_R$ to a binary relation $R_i^I \subseteq \Delta^I \times \Delta^I$, and maps each individual name $p_i \in N_I$ to an individual $p_i^I \in \Delta^I$. The extension of $\cdot^I$ to arbitrary concepts and roles is inductively defined, as shown in the forth column of Table 1.

If role hierarchies are supported by a description logic, then each role hierarchy is also called a *role inclusion axiom*, and each finite set of role inclusion axioms is called an *RBox*.

A *concept definition* is an identity of the form $A \equiv C$, where A is a concept name and C a concept. A *TBox* is a finite set of concept definitions with unique left-hand sides [4]. A TBox is said to be *acyclic* if there are no cyclic dependencies between the definitions. Every TBox mentioned in this paper is assumed to be acyclic.

An *ABox assertion* is of the form $C(p)$, $\neg C(p)$, $R(p, q)$ or $\neg R(p, q)$, where $p, q \in N_I$, C is a concept, and R is a role. A finite set of ABox assertions is called an *ABox*.

Given an interpretation $I = (\Delta^I, \cdot^I)$, it is a *model* of an RBox $\mathcal{R}$, denoted by $I \models \mathcal{R}$, iff $R^I \subseteq R'^I$ for every role inclusion axiom $R \sqsubseteq R' \in \mathcal{R}$; it is a *model* of a TBox $\mathcal{T}$, denoted by $I \models \mathcal{T}$, iff $A^I = C^I$ for every concept definition $A \equiv C \in \mathcal{T}$; it is a *model* of an ABox $\mathcal{A}$, denoted by $I \models \mathcal{A}$, iff $p^I \in C^I$ (resp. $p^I \notin C^I$, $(p^I, q^I) \in R^I$, and $(p^I, q^I) \notin R^I$) for every ABox assertion $C(p)$ (resp. $\neg C(p)$, $R(p, q)$ and $\neg R(p, q)$) contained in $\mathcal{A}$.

Various reasoning problems are considered for description logics. For the purpose of this paper, we investigate the ABox consistency problem.

Table 1 Syntax and semantics of the DLs ranging from $\mathcal{ALC}$ to $\mathcal{ALCHOIQ}^{@}$

	Constructor	Syntax	Semantics
$\mathcal{ALC}$	Concept name	A_i	$A_i^I \subseteq \Delta^I$
	Role name	R_i	$R_i^I \subseteq \Delta^I \times \Delta^I$
	Negation	$\neg C$	$\Delta^I \backslash C^I$
	Conjunction	$C \sqcap D$	$C^I \cap D^I$
	Disjunction	$C \sqcup D$	$C^I \cup D^I$
	Existential restriction	$\exists R.C$	$\{x \in \Delta^I \mid$ there is a $y \in \Delta^I$ with $(x, y) \in R^I$ and $y \in C^I\}$
	Value restriction	$\forall R.C$	$\{x \in \Delta^I \mid$ for all $y \in \Delta^I$: if $(x, y) \in R^I$, then $y \in C^I\}$
$\mathcal{H}$	Role hierarchy	$R \sqsubseteq R'$	if $(x, y) \in R^I$, then $(x, y) \in R'^I$
$\mathcal{O}$	Nominal	$\{ \mathrm{p} \}$	$\{ p^I \}$
$\mathcal{I}$	Inverse role	R^-	$\{ (y, x) \mid (x, y) \in R^I \}$
$\mathcal{Q}$	Qualified number restriction	$\geq n R.C$	$\{ x \in \Delta^I \mid \sharp\{ y \in \Delta^I \mid (x, y) \in R^I$ and $y \in C^I\} \geq n \}$
		$\leq n R.C$	$\{ x \in \Delta^I \mid \sharp\{ y \in \Delta^I \mid (x, y) \in R^I$ and $y \in C^I\} \leq n \}$
@	@ constructor	$@_p C$	Δ^I if $p^I \in C^I$, and $\emptyset$ otherwise.

An ABox $\mathcal{A}$ is said to be *consistent* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ iff there is an interpretation $I = (\Delta^I, \cdot^I)$ such that $I \models \mathcal{R}$, $I \models \mathcal{T}$ and $I \models \mathcal{A}$.

It is known that the ABox consistency problem is PSPACE-complete for the DLs $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$ and $\mathcal{ALCOQ}$ [4], is PSPACE-complete for both $\mathcal{ALCHO}$ and $\mathcal{ALCHOQ}$ [32], is EXPTIME-complete for $\mathcal{ALCOI}$ [1], and is NEXPTIME-complete for $\mathcal{ALCOIQ}$ [41]. Moreover, since this reasoning problem is EXPTIME-complete for the DL $\mathcal{SHOI}$ and NEXPTIME-complete for $\mathcal{SHOIQ}$ [44], we have the result that it is EXPTIME-complete for $\mathcal{ALCHOI}$ and NEXPTIME-complete for $\mathcal{ALCHOIQ}$.

As mentioned in the introduction, the @ constructor will be used in our deciding algorithms. Therefore, it is necessary to investigate the ABox consistency problem in the case that the @ constructor is occurring in ABoxes but absent from TBoxes.

For any DL $X \in \{\mathcal{ALCO}, \mathcal{ALCHO}, \mathcal{ALCOI}, \mathcal{ALCOQ}, \mathcal{ALCHOI}, \mathcal{ALCHOQ}, \mathcal{ALCOIQ}, \mathcal{ALCHOIQ}\}$, if the @ constructor is absent from TBoxes, then the ABox consistency problem of the DL $X^@$ can be reduced to the ABox consistency problem of the logic X. More precisely, for any ABox $\mathcal{A}$ of $X^@$ and any RBox $\mathcal{R}$ and TBox $\mathcal{T}$ of X, we can operate according to the following steps [25]:

1. Convert every concept occurring in $\mathcal{A}$ into its negation normal form (i.e., negation signs occur only in front of concept names or nominals); this conversion can be done by pushing negations inwards according to the following equivalences:

$$\begin{aligned}
\neg\neg C &= C & \neg(@_p C) &= @_p \neg C \\
\neg(C \sqcap D) &= \neg C \sqcup \neg D & \neg(C \sqcup D) &= \neg C \sqcap \neg D \\
\neg(\exists R.C) &= \forall R.\neg C & \neg(\forall R.C) &= \exists R.\neg C \\
\neg(\leq n R.C) &= \geq (n+1) R.C & \neg(\geq (n+1) R.C) &= \leq n R.C \\
\neg(\geq 0 R.C) &= A_i \sqcap \neg A_i \ \textit{for any } A_i \in N_C.
\end{aligned}$$

2. For every concept of the form $@_p C$ occurring in $\mathcal{A}$, replace it by a concept $\exists u.(\{p\} \sqcap C)$, where u is any role name not occurring in $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}$. Repeat this process, until the @ constructor does not occur in $\mathcal{A}$ any more.

It is obvious that the above two steps can be done in polynomial time in the size of $\mathcal{A}$. Let $\mathcal{A}'$ be the resulting ABox. Then it is easy to see that $\mathcal{A}'$ is an ABox of the DL X, and the size of $\mathcal{A}'$ is polynomial in the size of $\mathcal{A}$. Furthermore, it can be proved that $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$ iff $\mathcal{A}'$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$ [25]. Therefore, we get the following result:

Theorem 1 *Let $X \in \{\mathcal{ALCO}, \mathcal{ALCHO}, \mathcal{ALCOI}, \mathcal{ALCOQ}, \mathcal{ALCHOI}, \mathcal{ALCHOQ}, \mathcal{ALCOIQ}, \mathcal{ALCHOIQ}\}$. For any ABox $\mathcal{A}$ of the logic $X^@$, and any RBox $\mathcal{R}$ and TBox $\mathcal{T}$ of the logic X, the complexity upper-bound for deciding the consistency of $\mathcal{A}$ w.r.t. $\mathcal{R}$ and $\mathcal{T}$ is PSPACE if $X \in \{\mathcal{ALCO}, \mathcal{ALCHO}, \mathcal{ALCOQ}, \mathcal{ALCHOQ}\}$, is EXPTIME if $X \in \{\mathcal{ALCOI}, \mathcal{ALCHOI}\}$, and is NEXPTIME if $X \in \{\mathcal{ALCOIQ}, \mathcal{ALCHOIQ}\}$.*

3 The Dynamic Description Logic $DDL(X^@)$

In this section, we firstly present the syntax and semantics of $DDL(X^@)$, and then introduce an example to illustrate its expressive power.

3.1 Syntax of $DDL(X^{@})$

Primitive symbols of $DDL(X^{@})$ are a set N_I of individual names, a set N_R of role names, a set N_C of concept names, and a set N_A of action names. Starting from these symbols, basic citizens of $DDL(X^{@})$ such as roles, concepts, actions and formulas, are inductively defined with the help of constructors coming from both the description logic $X^{@}$ and the propositional dynamic logic PDL.

Roles of $DDL(X^{@})$ are defined by the same syntax rule of the roles of the DL $X^{@}$. For example, R is a role of the dynamic description logic $DDL(\mathcal{ALCO}^{@})$ if and only if $R \in N_R$. As another example, roles of $DDL(\mathcal{ALCHOIQ}^{@})$ are formed according to the following syntax rule:

$$R \ ::= \ R_i \ \mid \ R_i^{-}$$

where $R_i \in N_R$. For the simplicity of presentation, if inverse roles are supported by $DDL(X^{@})$, then we introduce the function Inv presented in [22] to return the inverse of a role, i.e., $Inv(R_i) := R_i^{-}$ and $Inv(R_i^{-}) := R_i$ for any $R_i \in N_R$.

In the case that role hierarchies are supported by $X^{@}$, we call each role hierarchy of the form $R \sqsubseteq R'$ as a *role inclusion axiom*, and call each finite set of role inclusion axioms as an *RBox* of $DDL(X^{@})$.

Concepts of $DDL(X^{@})$ are defined by the same syntax rule of the concepts of the DL $X^{@}$. For example, concepts of $DDL(\mathcal{ALCO}^{@})$ are formed according to the following syntax rule:

$$C, C' \ ::= \ A_i \mid \neg C \mid C \sqcup C' \mid C \sqcap C' \mid \forall R.C \mid \exists R.C \mid \{p\} \mid @_p C$$

where $A_i \in N_C$, $p \in N_I$, and R is a role. As another example, concepts of $DDL(\mathcal{ALCHOIQ}^{@})$ are formed according to the following syntax rule:

$$\begin{aligned} C, C' \ ::= \ & A_i \mid \neg C \mid C \sqcup C' \mid C \sqcap C' \mid \forall R.C \\ & \mid \exists R.C \mid \leq nR.C \mid \geq nR.C \mid \{p\} \mid @_p C \end{aligned}$$

where $A_i \in N_C$, $p \in N_I$, R is a role, and n is a non-negative integer.

Concepts of the form $\top$ and $\bot$ are introduced as abbreviations of $C \sqcup \neg C$ and $C \sqcap \neg C$ respectively, where C is any concept.

A *concept definition* is an identity of the form $A \equiv C$, where A is a concept name and C is a concept of $DDL(X^{@})$.

For each finite set $\mathcal{T}$ of concept definitions, if no concept name occurs on the left-hand sides for more than once, and no @ constructor occurs on the right-hand sides, then we call $\mathcal{T}$ a *TBox* of $DDL(X^{@})$.

A TBox is said to be *acyclic* if there are no cyclic dependencies between the definitions contained in it. In this paper, we assume that every TBox of $DDL(X^{@})$ is acyclic.

With respect to a TBox $\mathcal{T}$, a concept name $A_i \in N_C$ is called *defined* if and only if it occurs on the left-hand side of some concept definition contained in $\mathcal{T}$, and is called *primitive* otherwise.

For any concept C and any TBox $\mathcal{T}$, we use $C_{\mathcal{T}}$ to denote the *expansion of C w.r.t.* $\mathcal{T}$, and define it as the concept constructed as follows: for any concept name A occurring in C, if it is defined by some concept definition $A \equiv D \in \mathcal{T}$, then replace each occurrence of A in C with D; repeat this process, until no defined concept names occur in C.

Formulas of $DDL(X^{@})$ are formed according to the following syntax rule:

$$\varphi, \varphi' \ ::= \ C(p) \ | \ R(p,q) \ | \ <\pi>\varphi \ | \ [\pi]\varphi \ | \ \neg\varphi \ | \ \varphi \vee \varphi' \ | \ \varphi \wedge \varphi'$$

where $p, q \in N_I$, C is a concept, R is a role, and π is an action. Formulas of the form $C(p)$, $R(p,q)$, $<\pi>\varphi$, $[\pi]\varphi$, $\neg\varphi$, $\varphi \vee \varphi'$ and $\varphi \wedge \varphi'$ are respectively called *concept assertion*, *role assertion*, *diamond assertion*, *box assertion*, *negation formula*, *disjunction formula*, and *conjunction formula*.

Formulas of the form $\varphi \rightarrow \varphi'$, $\varphi \leftrightarrow \varphi'$, *true* and *false* are introduced as abbreviations of $\neg\varphi \vee \varphi'$, $(\neg\varphi \vee \varphi') \wedge (\neg\varphi' \vee \varphi)$, $\top(p)$ and $\bot(p)$ respectively, where p is any individual name.

Concept assertions, role assertions, negations of concept assertions, and negations of role assertions are all called *ABox assertions*. A finite set of ABox assertions is called an *ABox* of $DDL(X^{@})$.

With respect to a TBox $\mathcal{T}$, an ABox assertion is called a *primitive literal* if it is of the form $A(p)$, $\neg A(p)$, $R(p,q)$ or $\neg R(p,q)$ with A a primitive concept name, R a role and $p, q \in N_I$.

For any ABox assertion ψ, we use $\psi^{\neg}$ to denote an ABox assertion which is logically equivalent with $\neg\psi$, and define it as follows: if ψ is of the form $C(p)$, $\neg C(p)$, $R(p,q)$ or $\neg R(p,q)$, then $\psi^{\neg}$ is the ABox assertion $\neg C(p)$, $C(p)$, $\neg R(p,q)$ and $R(p,q)$ respectively.

For any ABox $\mathcal{A}$, we use $\mathcal{A}^{\neg}$ to denote the set $\{ \psi^{\neg} \mid \psi \in \mathcal{A} \}$, and use $Conj(\mathcal{A})$ to denote the conjunction of all the ABox assertions contained in $\mathcal{A}$.

For any ABox $\mathcal{A}$ and any RBox $\mathcal{R}$, we use $\mathcal{A}^{*}_{\mathcal{R}}$ to denote the *closure* of $\mathcal{A}$ w.r.t. $\mathcal{R}$, and define it as the smallest set satisfying the following conditions:

- $\mathcal{A} \subseteq \mathcal{A}^{*}_{\mathcal{R}}$;
- if $R(p,q) \in \mathcal{A}^{*}_{\mathcal{R}}$, then $Inv(R)(q,p) \in \mathcal{A}^{*}_{\mathcal{R}}$;
- if $\neg R(p,q) \in \mathcal{A}^{*}_{\mathcal{R}}$, then $\neg Inv(R)(q,p) \in \mathcal{A}^{*}_{\mathcal{R}}$;
- if $R(p,q) \in \mathcal{A}^{*}_{\mathcal{R}}$ and $R \sqsubseteq R' \in \mathcal{R}$, then $R'(p,q) \in \mathcal{A}^{*}_{\mathcal{R}}$; and
- if $\neg R'(p,q) \in \mathcal{A}^{*}_{\mathcal{R}}$ and $R \sqsubseteq R' \in \mathcal{R}$, then $\neg R(p,q) \in \mathcal{A}^{*}_{\mathcal{R}}$.

Actions of $DDL(X^{@})$ are formed according to the following syntax rule:

$$\pi, \pi' \ ::= \ \alpha \ | \ \varphi? \ | \ \pi \cup \pi' \ | \ \pi;\pi' \ | \ \pi^{*}$$

where $\alpha \in N_A$, and φ is a formula. Actions of the form α, $\varphi?$, $\pi \cup \pi'$, $\pi;\pi'$ and π^{*} are respectively called *atomic action*, *test action*, *choice action*, *sequential action* and *iterated action*.

With respect to a TBox $\mathcal{T}$, an *atomic action definition* of $DDL(X^{@})$ is of the form $\alpha \equiv (P, E)$, where

- $\alpha \in N_A$,
- P is a finite set of ABox assertions for describing the pre-conditions, and
- E is a finite set of primitive literals for describing the post-conditions.

For each finite set $\mathcal{A}_C$ of atomic action definitions, if no action name occurs on the left-hand sides for more than once, then we call $\mathcal{A}_C$ an *ActBox* of $DDL(X^{@})$.

With respect to an ActBox $\mathcal{A}_C$, an atomic action α is called *defined* if and only if α occurs on the left-hand side of some atomic action definition contained in $\mathcal{A}_C$;

an action π (or a formula φ) is called *defined* if and only if all the atomic actions occurring in π (resp. occurring in φ) are defined w.r.t. $\mathcal{A}_C$. In $DDL(X^@)$, we assume that all the actions and formulas are defined w.r.t. some ActBox.

For any atomic action α, if it is defined w.r.t. some ActBox, then we use $\mathbf{Pre}_\alpha$ to denote the set of its pre-conditions, and use $\mathbf{Eff}_\alpha$ to denote the set of its post-conditions.

A *knowledge base* of $DDL(X^@)$ is of the form $K = (\mathcal{R}, \mathcal{T}, \mathcal{A}_C, \mathcal{A})$, where $\mathcal{R}$, $\mathcal{T}$, $\mathcal{A}_C$ and $\mathcal{A}$ are respectively an RBox, a TBox, an ActBox and an ABox.

3.2 Semantics of $DDL(X^@)$

The semantic model of $DDL(X^@)$ is a combination of the interpretation of description logics and the model of propositional dynamic logic.

A $DDL(X^@)$-*model* is of the form $M = (W, T, \Delta, I)$, where,

- W is a non-empty finite set of states;
- T is a function which maps each action name $\alpha \in N_A$ to a binary relation $T(\alpha) \subseteq W \times W$;
- Δ is a non-empty set of individuals; and
- I is a function which associates with each state $w \in W$ a DL-interpretation $I(w) = < \Delta, \cdot^{I(w)} >$, where the function $\cdot^{I(w)}$
  - maps each concept name $A_i \in N_C$ to a set $A_i^{I(w)} \subseteq \Delta$,
  - maps each role name $R_i \in N_R$ to a binary relation $R_i^{I(w)} \subseteq \Delta \times \Delta$, and
  - maps each individual name $p \in N_I$ to an individual $p^{I(w)} \in \Delta$, with the constraints that $p^{I(w)} = p^{I(w')}$ for any state $w' \in W$, and $p^{I(w)} \neq q^{I(w)}$ for any individual name q which is different from p. Since interpretations of p are the same in every state, the interpretation $p^{I(w)}$ is also represented as p^I.

It should be noted that here we take the constant domain assumption [42] and the unique name assumption [3]. Moreover, individual names contained in N_I are treated as rigid designators [42].

Given a model $M = (W, T, \Delta, I)$, the semantics of roles, concepts, formulas and actions of $DDL(X^@)$ are defined inductively as follows.

Firstly, for any state $w \in W$, each role R is interpreted as a binary relation $R^{I(w)} \subseteq \Delta \times \Delta$, and each concept C is interpreted as a set $C^{I(w)} \subseteq \Delta$. The concrete semantic definitions are similar to those of the description logic $X^@$, except that here each interpretation is associated with a state. For example, in the case that $X^@$ is the description logic $\mathcal{ALCHOIQ}^@$, the semantics of roles and concepts of $DDL(\mathcal{ALCHOIQ}^@)$ are defined inductively as follows:

1. $(R^-)^{I(w)} = \{(y, x) \mid (x, y) \in R^{I(w)}\}$;
2. $(\neg C)^{I(w)} = \Delta \backslash C^{I(w)}$;
3. $(C \sqcup D)^{I(w)} = C^{I(w)} \cup D^{I(w)}$;
4. $(C \sqcap D)^{I(w)} = C^{I(w)} \cap D^{I(w)}$;
5. $(\forall R.C)^{I(w)} = \{ x \in \Delta \mid \text{for all } y \in \Delta\text{: if } (x, y) \in R^{I(w)}, \text{ then } y \in C^{I(w)}\}$;
6. $(\exists R.C)^{I(w)} = \{ x \in \Delta \mid \text{there is some } y \in \Delta \text{ such that } (x, y) \in R^{I(w)} \text{ and } y \in C^{I(w)}\}$;
7. $\{p\}^{I(w)} = \{p^I\}$;

8. $(\leq nS.C)^{I(w)} = \{x \in \Delta \mid \sharp\{y \in \Delta \mid (x, y) \in S^{I(w)} \text{ and } y \in C^{I(w)}\} \leq n\}$;
9. $(\geq nS.C)^{I(w)} = \{x \in \Delta \mid \sharp\{y \in \Delta \mid (x, y) \in S^{I(w)} \text{ and } y \in C^{I(w)}\} \geq n\}$;
10. $(@_p C)^{I(w)} = \Delta$ if $p^I \in C^{I(w)}$, and $\emptyset$ otherwise.

Secondly, for any state $w \in W$, the satisfaction-relation $(M, w) \models \varphi$ (or simply $w \models \varphi$ if M is understood) for any formula φ is defined inductively as follows:

11. $(M, w) \models C(p)$ iff $p^I \in C^{I(w)}$;
12. $(M, w) \models R(p, q)$ iff $(p^I, q^I) \in R^{I(w)}$;
13. $(M, w) \models < \pi > \varphi$ iff some state $w' \in W$ exists with $(w, w') \in T(\pi)$ and $(M, w') \models \varphi$;
14. $(M, w) \models [\pi]\varphi$ iff for every state $w' \in W$: if $(w, w') \in T(\pi)$ then $(M, w') \models \varphi$;
15. $(M, w) \models \neg\varphi$ iff it is not the case that $(M, w) \models \varphi$;
16. $(M, w) \models \varphi \vee \psi$ iff $(M, w) \models \varphi$ or $(M, w) \models \psi$;
17. $(M, w) \models \varphi \wedge \psi$ iff $(M, w) \models \varphi$ and $(M, w) \models \psi$.

Finally, each action π is interpreted as a binary relation $T(\pi) \subseteq W \times W$ according to the following definitions:

18. $T(\varphi?) = \{ (w, w) \mid w \in W \text{ and } (M, w) \models \varphi \}$;
19. $T(\pi \cup \pi') = T(\pi) \cup T(\pi')$;
20. $T(\pi; \pi') = \{(w, w') \mid \text{there is some state } u \in W \text{ with } (w, u) \in T(\pi) \text{ and } (u, w') \in T(\pi')\}$;
21. $T(\pi^*)$ = reflexive and transitive closure of $T(\pi)$.

A model M *satisfies* an RBox $\mathcal{R}$, denoted by $M \models \mathcal{R}$, if and only if $R^{I(w)} \sqsubseteq R'^{I(w)}$ for every role inclusion axiom $R \sqsubseteq R' \in \mathcal{R}$ and every state $w \in W$.

A model M *satisfies* a TBox $\mathcal{T}$, denoted by $M \models \mathcal{T}$, if and only if $A^{I(w)} = C^{I(w)}$ for every concept definition $A \equiv C \in \mathcal{T}$ and every state $w \in W$.

A state w of a model M *satisfies* an ABox $\mathcal{A}$, denoted by $(M, w) \models \mathcal{A}$, if and only if $(M, w) \models \varphi_i$ for every ABox assertion $\varphi_i \in \mathcal{A}$.

A feature of $DDL(X^@)$ is that each atomic action α is further specified by some atomic action definition $\alpha \equiv (P, E)$, where P and E respectively describe the pre-conditions and the post-conditions for the execution of α. We adopt the minimal-change semantics used in [4] and define the semantics of atomic action definitions as follows.

With respect to an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$, a model $M = (W, T, \Delta, I)$ *satisfies* an atomic action definition $\alpha \equiv (P, E)$, in symbols $M \models_{\mathcal{R},\mathcal{T}} \alpha \equiv (P, E)$, if and only if $M \models \mathcal{R}$, $M \models \mathcal{T}$, and

$$T(\alpha) = \{ (w, w') \in W \times W \mid (M, w) \models P, \; both\ A^+ \cap A^- = \emptyset\ and\ A^{I(w')} = (A^{I(w)} \cup A^+) \setminus A^-\ for\ each\ concept\ name\ A\ which\ is\ primitive\ w.r.t.\ \mathcal{T},\ and\ both\ R^+ \cap R^- = \emptyset\ and\ R^{I(w')} = (R^{I(w)} \cup R^+) \setminus R^-\ for\ each\ role\ name\ R. \},$$

where A^+, A^-, R^+ and R^- are some sets constructed as follows:

- $A^+ := \{ p^I \mid A(p) \in E \}$,
- $A^- := \{ p^I \mid \neg A(p) \in E \}$,

– $R^+ := \{ (p^I, q^I) \mid R(p, q) \in E^*_{\mathcal{R}} \}$,
– $R^- := \{ (p^I, q^I) \mid \neg R(p, q) \in E^*_{\mathcal{R}} \}$.

According to this definition, for any pair $(w, w') \in T(\alpha)$, any primitive concept name A, and any role name R, the interpretations $A^{I(w)}$ and $A^{I(w')}$ should satisfy that $A^+ \subseteq A^{I(w')}$, $A^- \cap A^{I(w')} = \emptyset$, and nothing else changes from $A^{I(w)}$ to $A^{I(w')}$; the interpretations $R^{I(w)}$ and $R^{I(w')}$ should satisfy that $R^+ \subseteq R^{I(w')}$, $R^- \cap R^{I(w')} = \emptyset$, and nothing else changes from $R^{I(w)}$ to $R^{I(w')}$. Thus, $T(\alpha)$ enforces the minimal-change semantics.

A model M *satisfies* an ActBox $\mathcal{A}_\mathcal{C}$ w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$, in symbols $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_\mathcal{C}$, if and only if $M \models_{\mathcal{R},\mathcal{T}} \alpha \equiv (P, E)$ for every atomic action definition $\alpha \equiv (P, E) \in \mathcal{A}_\mathcal{C}$.

3.3 Example Description by $DDL(X^@)$

As an example, consider a Web service system in which customers are able to buy/return CDs and books online with credit cards [18]. In this section, we model some high-level features of this system by $DDL(\mathcal{ALCHOIQ}^@)$.

First of all, primitive symbols which will be used are listed as follows:

$$N_C := \{ \mathit{person}, \mathit{creditcard}, \mathit{cd}, \mathit{book}, \mathit{customer}, \mathit{VIPCust}, \mathit{captiousCust}, \mathit{instore} \};$$

$$N_R := \{ \mathit{boughtCD}, \mathit{boughtBook}, \mathit{bought}, \mathit{has}, \mathit{holds}, \mathit{returned} \};$$

$$N_I := \{ \mathit{Tom}, \mathit{Jack}, \mathit{Mastercard}, \mathit{Visa}, \mathit{KingLear}, \mathit{HarryPotter}, \mathit{GrimmsFairyTales}, \mathit{BackStreetBoys}, \mathit{SchubertSymphonien} \};$$

$$\begin{aligned} N_A := \{ & \mathit{buyCD}_{\mathit{Tom,Bac}}, \mathit{buyCD}_{\mathit{Tom,Sch}}, \mathit{buyCD}_{\mathit{Jack,Bac}}, \mathit{buyCD}_{\mathit{Jack,Sch}}, \\ & \mathit{buyBook}_{\mathit{Tom,Kin}}, \mathit{buyBook}_{\mathit{Tom,Har}}, \mathit{buyBook}_{\mathit{Tom,Gri}}, \\ & \mathit{buyBook}_{\mathit{Jack,Kin}}, \mathit{buyBook}_{\mathit{Jack,Har}}, \mathit{buyBook}_{\mathit{Jack,Gri}}, \\ & \mathit{returnCD}_{\mathit{Tom,Bac}}, \mathit{returnCD}_{\mathit{Tom,Sch}}, \mathit{returnCD}_{\mathit{Jack,Bac}}, \mathit{returnCD}_{\mathit{Jack,Sch}}, \\ & \mathit{returnBook}_{\mathit{Tom,Kin}}, \mathit{returnBook}_{\mathit{Tom,Har}}, \mathit{returnBook}_{\mathit{Tom,Gri}}, \\ & \mathit{returnBook}_{\mathit{Jack,Kin}}, \mathit{returnBook}_{\mathit{Jack,Har}}, \mathit{returnBook}_{\mathit{Jack,Gri}}, \\ & \mathit{order}_{\mathit{Bac}}, \mathit{order}_{\mathit{Sch}}, \mathit{order}_{\mathit{Kin}}, \mathit{order}_{\mathit{Har}}, \mathit{order}_{\mathit{Gri}} \}. \end{aligned}$$

Starting from these primitive symbols, an RBox $\mathcal{R}_{shop}$ and a TBox $\mathcal{T}_{shop}$ are constructed for describing the static domain knowledge. The RBox $\mathcal{R}_{shop}$ is composed of the following role inclusion axiom:

$$\mathit{bought} \sqsubseteq \mathit{has}$$

and the TBox $\mathcal{T}_{shop}$ is composed of the following concept definitions:

$$\begin{aligned} \mathit{customer} &\equiv \mathit{person} \sqcap \exists \mathit{holds}.\mathit{creditCard} \\ \mathit{VIPCust} &\equiv \mathit{customer} \sqcap \geq 3\, \mathit{bought}.(\mathit{cd} \sqcup \mathit{book}) \\ \mathit{captiousCust} &\equiv \mathit{customer} \sqcap \geq 2\, \mathit{returned}.\mathit{cd} \end{aligned}$$

These definitions state that a customer is a person holding some credit card; a VIP customer is a customer who has bought at least 3 CDs or books; and a captious customer is a customer who has returned at least 2 CDs.

The knowledge on atomic Web services is described by an ActBox $\mathcal{A}_{Cshop}$ with five groups of atomic action definitions.

Firstly, for describing the service that *Tom* buy the CD *BackStreetBoys*, an atomic action named $buyCD_{Tom,Bac}$ is specified as follows:

$$buyCD_{Tom,Bac} \equiv (\{customer(Tom), cd(BackStreetBoys), instore(BackStreetBoys)\}, \{\neg instore(BackStreetBoys), bought(Tom, BackStreetBoys)\}).$$

According to this definition, the action $buyCD_{Tom,Bac}$ is applicable if *Tom* is a customer and *BackStreetBoys* is a CD in store; furthermore, if it is executed, then the result is that *Tom* has bought *BackStreetBoys* and *BackStreetBoys* is not in store any more. The atomic actions $buyCD_{Tom,Sch}$, $buyCD_{Jack,Bac}$ and $buyCD_{Jack,Sch}$ are similarly defined in $\mathcal{A}_{Cshop}$, respectively for describing the services that *Tom* buy the CD *SchubertSymphonien*, *Jack* buy the CD *BackStreetBoys*, and *Jack* buy the CD *SchubertSymphonien*.

Secondly, for describing the service that *Tom* buy the book *KingLear*, an atomic action named $buyBook_{Tom,Kin}$ is defined as follows:

$$buyBook_{Tom,Kin} \equiv (\{customer(Tom), book(KingLear), instore(KingLear)\}, \{\neg instore(KingLear), bought(Tom, KingLear)\}).$$

The atomic actions $buyBook_{Tom,Har}$, $buyBook_{Tom,Gri}$, $buyBook_{Jack,Kin}$, $buyBook_{Jack,Har}$ and $buyBook_{Jack,Gri}$ are similarly defined.

Thirdly, for describing the service that *Tom* return the CD *BackStreetBoys*, an atomic action named $returnCD_{Tom,Bac}$ is defined as follows:

$$returnCD_{Tom,Bac} \equiv (\{customer(Tom), cd(BackStreetBoys), bought(Tom, BackStreetBoys), \neg instore(BackStreetBoys)\}, \{instore(BackStreetBoys), returned(Tom, BackStreetBoys), \neg has(Tom, BackStreetBoys)\}).$$

The atomic actions $returnCD_{Tom,Sch}$, $returnCD_{Jack,Bac}$ and $returnCD_{Jack,Sch}$ are similarly defined.

Fourthly, for describing the service that *Tom* return the book *KingLear*, an atomic action named $returnBook_{Tom,Kin}$ is defined as follows:

$$returnBook_{Tom,Kin} \equiv (\{customer(Tom), book(KingLear), bought(Tom, KingLear), \neg instore(KingLear)\}, \{instore(KingLear), returned(Tom, KingLear), \neg has(Tom, KingLear)\}).$$

The atomic actions $returnBook_{Tom,Har}$, $returnBook_{Tom,Gri}$, $returnBook_{Jack,Kin}$, $returnBook_{Jack,Har}$ and $returnBook_{Jack,Gri}$ are similarly defined.

Finally, for describing the service that the Web service agent order the book *BackStreetBoys* from the publisher, an atomic action named $order_{Bac}$ is defined as follows:

$$\begin{aligned} & order_{Bac} \\ & \equiv (\{(book \sqcup cd)(BackStreetBoys), \neg instore(BackStreetBoys)\}, \\ & \quad \{instore(BackStreetBoys)\}). \end{aligned}$$

The atomic actions $order_{Sch}$, $order_{Har}$, $order_{Gri}$ and $order_{Kin}$ are similarly defined.

The knowledge represented above can also be described with Baader et al.'s formalisms [4, 24, 30].

From the point of view of knowledge representation, a primary feature of $DDL(X^{@})$ is that many complex actions (or composed Web services) can be further specified.

For example, a composed Web service named $buyBook_{Har}$ might be represented as

$$buyBook_{Tom,Har} \cup buyBook_{Jack,Har}$$

This service is useful for both *Tom* and *Jack* to buy the book *HarryPott*.

As another example, a composed Web service named $VIPbuy_{Tom,Har}$ might be represented as

$$\begin{aligned} & VIPCust(Tom)?\ ;((instore(HarryPott)?\ ;\ buyBook_{Tom,Har}) \\ & \cup (\neg instore(HarryPott)?\ ;\ order_{Har}\ ;\ buyBook_{Tom,Har})) \end{aligned}$$

In this service, the test action "$VIPCust(Tom)?$" will be firstly executed to check whether *Tom* is a VIP customer; if the result is true, then the action "$buyBook_{Tom,Har}$" will be executed in the case that *HarryPott* is in store, and the actions "$order_{Har}$" and "$buyBook_{Tom,Har}$" will be executed sequentially in the case that *HarryPott* is not in store.

In fact, with the help of the sequence-, choice-, test- and iteration-constructors on actions, the "Sequence", "Choice", "Any-Order", "Iterate", "If-Then-Else", "Repeat-While" and "Repeat-Until" control structures adopted by the Web service ontology OWL-S [27] can be described in $DDL(X^{@})$ according to the following definitions:

$$\begin{aligned} sequence(\pi, \pi') &\triangleq \pi\ ;\pi' \\ choice(\pi, \pi') &\triangleq \pi \cup \pi' \\ any-order(\pi, \pi') &\triangleq (\pi\ ;\pi') \cup (\pi';\pi) \\ if\ \psi\ then\ \pi\ else\ \pi' &\triangleq (\psi?\ ;\pi) \cup ((\neg\psi)?\ ;\pi') \\ iterate(\pi) &\triangleq \pi^{*} \\ while\ \psi\ do\ \pi &\triangleq (\psi?\ ;\pi)^{*};(\neg\psi)? \\ do\ \pi\ until\ \psi &\triangleq \pi\ ;((\neg\psi)?\ ;\pi)^{*};\psi? \end{aligned}$$

From the point of view of knowledge representation, the second feature of $DDL(X^{@})$ is that many properties on actions (or Web services) can be stated explicitly by formulas.

On the one hand, necessary conditions for executing actions can be stated by diamond assertions. For example, the following formula states that a necessary condition for executing the action $VIPbuy_{Tom,Har}$ is that *Tom* is a VIP customer and *HarryPott* is a book:

$$< VIPbuy_{Tom,Har} > true \rightarrow (VIPCust(Tom) \wedge book(HarryPott))$$

As another example, the following formula states that the action "$buyCD_{Tom,Bac}$" and the action "$buyCD_{Jack,Bac}$" can never be executed sequentially:

$$\neg < buyCD_{Tom,Bac}; buyCD_{Jack,Bac} > true$$

On the other hand, results on the execution of actions can be stated with box assertions. For example, the following formula states that *Tom* will have the book *HarryPott* once the action $VIPbuy_{Tom,Har}$ is executed:

$$[VIPbuy_{Tom,Har}]has(Tom, HarryPott)$$

As another example, the following formula states that if *Tom* ever returned some CD, then he will become a captious customer once the action $returnCD_{Tom,Bac}$ or the action $returnCD_{Tom,Sch}$ is executed:

$$(\exists returned.CD)(Tom) \rightarrow [returnCD_{Tom,Bac} \cup returnCD_{Tom,Sch}]captiousCust(Tom)$$

As the last part of the Web service system, the current state is described by an ABox $\mathcal{A}_{shop}$ with the following ABox assertions:

$$person(Tom), \quad person(Jack), \quad creditCard(Mastercard), \quad creditCard(Visa),$$
$$book(HarryPotter), \quad book(KingLear), \quad book(GrimmsFairyTales),$$
$$cd(BackStreetBoys), \quad cd(SchubertSymphonien),$$
$$holds(Tom, Mastercard), \quad holds(Jack, Visa),$$
$$instore(KingLear), \quad instore(GrimmsFairyTales),$$
$$instore(BackStreetBoys), \quad \neg instore(HarryPotter).$$

In the next section, we will take the knowledge base $K_{shop} = (\mathcal{R}_{shop}, \mathcal{T}_{shop}, \mathcal{A}_{Cshop}, \mathcal{A}_{shop})$ as an example for investigating the inference problems of $DDL(X^{@})$.

4 Inference Problems in $DDL(X^{@})$

Inference problems in $DDL(X^{@})$ are divided into two groups. One is composed of those being studied in description logics, such as the consistency problem of ABoxes and the satisfiability problem of concepts. The other group deals with actions and is new introduced in $DDL(X^{@})$.

Inference problems of the first group can also be redefined with $DDL(X^{@})$-models. For example, the consistency of ABoxes can be defined as follows:

Definition 1 An ABox $\mathcal{A}$ is *consistent* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if there are some model $M = (W, T, \Delta, I)$ and some state $w \in W$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$ and $(M, w) \models \mathcal{A}$.

As another example, the satisfiability of concepts can be defined as follows:

Definition 2 A concept C is *satisfiable* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if there are some model $M = (W, T, \Delta, I)$ and some state $w \in W$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$ and $C^{I(w)} \neq \emptyset$.

It is obvious that the satisfiability problem of concepts can be reduced to the consistency problem of ABoxes; i.e.,

Theorem 2 *A concept C is* satisfiable *w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if the ABox $\{C(p)\}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, where p is an individual name does not occur in C, $\mathcal{R}$ and $\mathcal{T}$.*

Since actions do not occur in concepts, ABoxes, RBoxes and TBoxes of $DDL(X^{@})$, the consistency problem of ABoxes as well as the satisfiability problem of concepts can be decided with the help of standard reasoning mechanisms provided by the description logic $X^{@}$.

For the second group of inference problems, we firstly introduce the PE-consistency problem of atomic action definitions.

Definition 3 An atomic action definition $\alpha \equiv (P, E)$ is *PE-consistent* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if both the ABox P and the ABox E are consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

Definition 4 An ActBox $\mathcal{A}_C$ is *PE-consistent* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if every atomic action definition contained in $\mathcal{A}_C$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

Given a knowledge base $K = (\mathcal{R}, \mathcal{T}, \mathcal{A}_C, \mathcal{A})$ of $DDL(X^{@})$, it should be guaranteed that the ABox $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, and the ActBox $\mathcal{A}_C$ is PE-consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$. This is the premise for investigating other inference problems introduced in $DDL(X^{@})$. As an example, for the knowledge base presented in Section 3.3, it can be decided that $\mathcal{A}_{shop}$ is consistent w.r.t. $\mathcal{R}_{shop}$ and $\mathcal{T}_{shop}$, and $\mathcal{A}_{Cshop}$ is PE-consistent w.r.t. $\mathcal{R}_{shop}$ and $\mathcal{T}_{shop}$.

The second inference problem introduced in $DDL(X^{@})$ is the satisfiability/validity problem of formulas.

Definition 5 A formula φ is *satisfiable* w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$ if and only if there is a model $M = (W, T, \Delta, I)$ and a state $w \in W$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$ and $(M, w) \models \varphi$.

Definition 6 A formula φ is *valid* w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$ if and only if for any model $M = (W, T, \Delta, I)$ with $M \models \mathcal{R}$, $M \models \mathcal{T}$ and $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$, we have $(M, w) \models \varphi$ for every state $w \in W$.

It is obvious that the validity problem can be reduced to the satisfiability problem; i.e.,

Theorem 3 *A formula φ is valid w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$ if and only if the formula $\neg\varphi$ is unsatisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$.*

The third inference problem introduced in $DDL(X^@)$ is the *consistency problem* (also called *realizability problem* [26]) of actions. Intuition of this inference problem is to decide whether a given action makes sense with respect to the knowledge specified by an RBox, a TBox and an ActBox. With $DDL(X^@)$-models, it is formally defined as follows:

Definition 7 An action π is *consistent* (also called *realizable*) w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$ if and only if there is a model $M = (W, T, \Delta, I)$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$ and $T(\pi) \neq \emptyset$.

For example, considering the knowledge base presented in Section 3.3, the following sequential action is realizable w.r.t. $\mathcal{R}_{shop}$, $\mathcal{T}_{shop}$ and $\mathcal{A}_{Cshop}$:

$$buyBook_{Tom,Kin} \ ; \ returnBook_{Tom,Kin} \ ; \ buyBook_{Jack,Kin}$$

However, the action

$$buyBook_{Tom,Kin} \ ; \ buyBook_{Jack,Kin}$$

is not realizable w.r.t. $\mathcal{R}_{shop}$, $\mathcal{T}_{shop}$ and $\mathcal{A}_{Cshop}$.

According to the definition, it is obvious that the realizability problem of actions can be reduced to the satisfiability problem of formulas; i.e.:

Theorem 4 *An action π is realizable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$ if and only if the formula $< \pi >$ true is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$.*

The fourth inference problem introduced in $DDL(X^@)$ is the *executability problem* of actions [4]. Intuition of this inference problem is to decide whether a given action can be performed successfully starting from the states described by a given ABox. In order to define this inference problem with $DDL(X^@)$-models, we introduce some notations.

Let $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$ be an RBox, a TBox and an ActBox respectively; let $M = (W, T, \Delta, I)$ be a model with $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$. Then,

1. M is called *complete* w.r.t. the ActBox $\mathcal{A}_C$ if and only if for any state $w \in W$ and any atomic action definition $\alpha \equiv (P, E) \in \mathcal{A}_C$: if $(M, w) \models P$, then there must be some state $w' \in W$ with $(w, w') \in T(\alpha)$;
2. a state $w' \in W$ is called *connected with* a state w w.r.t. $\mathcal{A}_C$ if and only if
   - w' and w are the same state, or
   - there exist n ($n \geq 1$) atomic actions $\alpha_1, \ldots, \alpha_n$ defined in $\mathcal{A}_C$ and $n-1$ states $w_1, \ldots, w_{n-1} \in W$ such that $(w, w_1) \in T(\alpha_1)$, $(w_i, w_{i+1}) \in T(\alpha_{i+1})$ for every $1 \leq i \leq n-2$, and $(w_{n-1}, w') \in T(\alpha_n)$;
3. M is called *complete w.r.t. a state* w and the ActBox $\mathcal{A}_C$ if and only if for any state $w' \in W$ and any atomic action definition $\alpha \equiv (P, E) \in \mathcal{A}_C$: if $(M, w') \models P$ and w' is connected with w w.r.t. $\mathcal{A}_C$, then there must be some state $w'' \in W$ such that $(w', w'') \in T(\alpha)$.

Based on these notations, the executability of actions is defined as follows:

Definition 8 An action π is *executable* on states described by an ABox $\mathcal{A}$ w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$ if and only if for any model $M = (W, T, \Delta, I)$ and any state $w \in W$: if $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$, $(M, w) \models \mathcal{A}$, and M is complete w.r.t. w and $\mathcal{A}_C$, then there must be a state $w' \in W$ such that $(w, w') \in T(\pi)$.

The above definition is inspired by the PDL-based framework proposed by De Giacomo et al. for reasoning about actions [9]. In De Giacomo et al's framework, the executability of an action π on the states described by a formula S_0 is captured by a logical implication of the form $\Gamma \models S_0 \rightarrow < \pi > true$, where Γ is a finite set composed of precondition axioms, effect axioms and frame axioms, and the logical implication $\Gamma \models S_0 \rightarrow < \pi > true$ states that for any model M: if all the axioms contained in Γ hold in every state of M, then the action π will be performed in every state satisfying the formula S_0.

In $DDL(X^@)$, based on the syntax and semantics of atomic action definitions, all the knowledge described by effect axioms and frame axioms in De Giacomo et al's framework are captured here by an ActBox. Furthermore, for each precondition axiom of the form $< \alpha > true \leftrightarrow Pre$ in De Giacomo et al's framework (where α is an atomic action and Pre is a formula), the knowledge described by the formula $< \alpha > true \rightarrow Pre$ is also captured by the ActBox; what is left to deal with is the knowledge described by the formula $Pre \rightarrow < \alpha > true$, which states that the action α will be performed whenever the precondition Pre is satisfied. Therefore, in $DDL(X^@)$, we introduce the notation of complete model. Based on this notation and be similar with that stated by De Giacomo et al's formula, if an action π is executable on the states described by an ABox $\mathcal{A}$ w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$, then we can state that for any model M: if $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$, and M is complete w.r.t. $\mathcal{A}_C$, then the action π will be performed in every state satisfying the ABox $\mathcal{A}$. Moreover, since here we only care about each state w which satisfies the ABox $\mathcal{A}$, we can replace the premise "M is complete w.r.t. $\mathcal{A}_C$" with a premise "M is complete w.r.t. the state w and the ActBox $\mathcal{A}_C$", and then get Definition 8.

As an interesting example, for the knowledge base presented in Section 3.3, the following choice action is executable on the states described by $\mathcal{A}_{shop}$ w.r.t. $\mathcal{R}_{shop}$, $\mathcal{T}_{shop}$ and $\mathcal{A}_{Cshop}$:

$$buyCD_{Tom,Sch} \cup order_{Sch}$$

However, neither the action "$buyCD_{Tom,Sch}$" nor the action "$order_{Sch}$" is executable on the states described by $\mathcal{A}_{shop}$.

With the following theorem, the executability problem of actions can be reduced to the validity problem of formulas:

Theorem 5 *An action π is executable on the states described by an ABox $\mathcal{A}$ w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$ if and only if the following formula is valid w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$:*

$$[(\alpha_1 \cup ... \cup \alpha_n)^*]\Pi \rightarrow (Conj(\mathcal{A}) \rightarrow < \pi > true)$$

where α_1, ..., α_n are all the atomic actions defined in $\mathcal{A}_C$, and Π denotes the formula $(Conj(\boldsymbol{Pre}_{\alpha_1}) \rightarrow < \alpha_1 > true) \wedge ... \wedge (Conj(\boldsymbol{Pre}_{\alpha_n}) \rightarrow < \alpha_n > true)$.

In this theorem, the formula $[(\alpha_1 \cup ... \cup \alpha_n)^*]\Pi$ is constructed to guarantee that for any model $M = (W, T, \Delta, I)$ and any state $w \in W$ with $M \models \mathcal{R}$, $M \models \mathcal{T}$ and $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$: $(M, w) \models [(\alpha_1 \cup ... \cup \alpha_n)^*]\Pi$ if and only if M is complete w.r.t. w and $\mathcal{A}_C$. Based on this result, the correctness of Theorem 5 is an easy consequence of Definition 8.

If an action π is executable on the states described by an ABox $\mathcal{A}$, then we want to know whether applying it achieves some desired effect, i.e., whether some formula which we want to make true really holds after performing the action. Such an inference problem is called the *projection problem* [4, 36]. It is formally defined in $DDL(X^@)$ as follows:

Definition 9 With respect to an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$, a formula ψ is a *consequence of applying an action* π on the states described by an ABox $\mathcal{A}$ if and only if for any model $M = (W, T, \Delta, I)$ and any states $w, w' \in W$: if $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$, $(M, w) \models \mathcal{A}$ and $(w, w') \in T(\pi)$, then it must be $(M, w') \models \psi$.

For example, considering the knowledge base presented in Section 3.3 again, the formula "$VIPCust(Tom)$" is a consequence of applying the following sequential action on the states described by $\mathcal{A}_{shop}$:

$$buyCD_{Tom,Bac} \ ; \ buyBook_{Tom,Gri} \ ; \ buyBook_{Tom,Kin}$$

but it is not a consequence of applying a single action among $buyCD_{Tom,Bac}$, $buyBook_{Tom,Gri}$ and $buyBook_{Tom,Kin}$.

According to the definition, the projection problem of actions can also be reduced to the validity problem of formulas; i.e.:

Theorem 6 *A formula ψ is a consequence of applying an action π on the states described by an ABox $\mathcal{A}$ w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$ if and only if the formula $Conj(\mathcal{A}) \rightarrow [\pi]\psi$ is valid w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$.*

We conclude this section with the result that, in $DDL(X^@)$, the realizability problem, the executability problem and the projection problem of actions can all be reduced to the satisfiability problem of formulas.

5 A Satisfiability-Checking Algorithm for $DDL(X^@)$-Formulas

Let $\mathcal{R}$, $\mathcal{T}$ be an RBox and a TBox respectively; let $\mathcal{A}_C$ be an ActBox which is PE-consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$; and let ϕ be a $DDL(X^@)$-formula which is defined w.r.t. $\mathcal{A}_C$. In this section, we present an algorithm for deciding whether ϕ is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$.

For the ease of presentation, we firstly transform the formula ϕ into a normal form $nf(\phi)$ according to the following steps:

1. Replace each occurrence of atomic actions with their definitions; i.e., for any atomic action α occurring in ϕ, if it is defined by some atomic action definition $\alpha \equiv (P, E)$ in $\mathcal{A}_C$, then replace each occurrence of α in ϕ by the pair (P, E). Let ϕ' be the resulted formula.

2. Transform ϕ' into an equivalent one in negation normal form (i.e., negation signs occur only in front of concept assertions or role assertions), by pushing negations inwards according to the following equivalences:

$$\neg(<\pi>\varphi) = [\pi]\neg\varphi \qquad \neg([\pi]\varphi) = <\pi>\neg\varphi \qquad \neg\neg\varphi = \varphi$$
$$\neg(\varphi \wedge \varphi') = \neg\varphi \vee \neg\varphi' \qquad \neg(\varphi \vee \varphi') = \neg\varphi \wedge \neg\varphi'$$

In the rest of this paper, for each atomic action α defined by some atomic action definition $\alpha \equiv (P, E) \in \mathcal{A}_C$, we will use the pair (P, E) to denote the action α.

5.1 Algorithm Description

The algorithm presented here is in fact a combination of a tableau algorithm for the propositional dynamic logic PDL [10, 35], the procedure investigated in Section 2 for deciding the consistency of ABoxes of the description logic $X^@$, and a modification of the ABox updating algorithm proposed by Liu et al. [23].

The algorithm is based on the idea that ϕ is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$ if and only if we can construct a $DDL(X^@)$-model $M = (W, T, \Delta, I)$ and a state $w_0 \in W$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$ and $(M, w_0) \models \phi$. In order to denote and manipulate the models and states explicitly in the algorithm, we introduce the notations of prefixes, prefixed formulas, branches, and branch-model mappings. Each prefix is introduced to denote some state of a model. Each prefixed formula represents that the corresponding formula will hold on the state denoted by the corresponding prefix. A branch is a set composed of prefixed formulas as well as some auxiliary elements; by a branch-model mapping, each prefix occurring in the branch is mapped to some state of a model. As a result, with these notations, the process of constructing models for ϕ will be represented as a process of expanding branches according to some tableau expansion rules.

Definition 10 A *prefix* is of the form $\sigma.\varepsilon$ with σ an action and ε a set of primitive literals, and is constructed according to the following syntax rule:

$$\sigma.\varepsilon \quad ::= \quad (\emptyset, \emptyset).\emptyset \quad | \quad \sigma; (P, E).(\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$$

where $(\emptyset, \emptyset)$ and (P, E) are atomic actions,[1] $\sigma; (P, E)$ is a sequential action, and $(\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$ is a set composed of primitive literals. We also use $\sigma_0.\varepsilon_0$ to denote the prefix $(\emptyset, \emptyset).\emptyset$ and call it the *initial prefix*.

A *prefixed formula* is a pair $\sigma.\varepsilon : \varphi$, where $\sigma.\varepsilon$ is a prefix and φ is a formula.

In this definition, the prefixes are technically designed to guarantee that some function ι can be constructed to map each prefix $\sigma.\varepsilon$ to some state $\iota(\sigma.\varepsilon)$, satisfying that:

- if the formula ϕ is satisfiable, then the initial prefix $\sigma_0.\varepsilon_0$ is mapped to some state satisfying ϕ;
- the track of atomic actions executed from the state $\iota(\sigma_0.\varepsilon_0)$ to the state $\iota(\sigma.\varepsilon)$ is recorded by the sequential action σ; and

[1] Here $(\emptyset, \emptyset)$ is a special atomic action introduced temporarily by the satisfiability-checking algorithm.

– the accumulated post-conditions of the sequential action σ are captured by the set ε, so that the state $\iota(\sigma.\varepsilon)$ can be treated as the result of performing a special atomic action $(\emptyset, \varepsilon)$ on the state $\iota(\sigma_0.\varepsilon_0)$.

Definition 11 A *branch* $\mathcal{B}$ is a union of the following two sets:

– a set $\mathcal{B}_{PF}$ of prefixed formulas, and
– a set $\mathcal{B}_E$ of eventuality records, where each *eventuality record* is of the form $X \doteq < \pi^* > \varphi$, with X a character string and $< \pi^* > \varphi$ a formula.

For any branch $\mathcal{B}$, we use $IV_{\mathcal{B}}$ to denote the *initial view* of $\mathcal{B}$ and define it as follows:

$$IV_{\mathcal{B}} \triangleq \{\psi \mid \sigma_0.\varepsilon_0 : \psi \in \mathcal{B} \textit{ and } \psi \textit{ is an ABox assertion}\}.$$

Tableau expansion rules used in the algorithm are listed in Figs. 1, 2, 3, and 4.

Figure 1 presents tableau expansion rules on inverse roles, conjunction formulas and disjunction formulas. They are straightforward according to the corresponding semantic definitions.

Figure 2 presents tableau expansion rules for non-atomic actions. The $;_{[\,]}$-, $;_{<>}$-, $?_{[\,]}$-, $?_{<>}$-, $\cup_{[\,]}$- and $\cup_{<>}$-rules are straightforward according to the semantics of sequential actions, test actions and choice actions. The $*_{[\,]}$-, $*_{<>}$- and X-rules are designed for iterated actions; they are similar with the tableau rules introduced by De Giacomo [10] for dealing with iterated eventualities in the propositional dynamic logic. In our algorithm, if there is an iterated eventuality $< \pi^* > \varphi$ which is prefixed by some prefix and is not tagged, then a new character string X will be introduced by the $*_{<>}$-rule to tag this formula. This tag will be carried along with $< \pi^* > \varphi$ as while as this formula is propagated by the X-, $;_{<>}$-, $?_{<>}$-, $\cup_{<>}$- and $atom_{<>}$-rules, until either some prefix $\sigma'.\varepsilon'$ is reached with both $\sigma'.\varepsilon' : X \in \mathcal{B}$ and $\sigma'.\varepsilon' : \varphi \in \mathcal{B}$, or no more prefix can be introduced by the $atom_{<>}$-rule.

Figure 3 presents tableau expansion rules for atomic actions. Both of these rules are based on the intuition that two prefixes $\sigma.\varepsilon$ and $\sigma'.\varepsilon'$ denote the same state if and only if $\varepsilon = \varepsilon'$; this intuition will be demonstrated in the next subsection by Corollary 3. Therefore, according to the $atom_{<>}$-rule, if there is already some prefix $\sigma_i.\varepsilon_i$ with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$, then the branch will be expanded directly with the set $\{\sigma.\varepsilon : \phi \mid \phi \in P\} \cup \{\sigma_i.\varepsilon_i : \varphi\} \cup \{\sigma_i.\varepsilon_i : \psi \mid \psi \in \varepsilon_i\}$; otherwise, we should firstly introduce a new prefix $\sigma'.\varepsilon' := \sigma; (P, E).(\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$ before expanding the branch. Similarly, for

$\neg r^-$-rule:	If $\sigma.\varepsilon : \neg R(p,q) \in \mathcal{B}$, and $\sigma.\varepsilon : \neg Inv(R)(q,p) \notin \mathcal{B}$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma.\varepsilon : \neg Inv(R)(q,p)\}$.
r^--rule:	If $\sigma.\varepsilon : R(p,q) \in \mathcal{B}$, and $\sigma.\varepsilon : Inv(R)(q,p) \notin \mathcal{B}$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma.\varepsilon : Inv(R)(q,p)\}$.
$\wedge$-rule:	If $\sigma.\varepsilon : \varphi \wedge \psi \in \mathcal{B}$, and $\{\sigma.\varepsilon : \varphi, \sigma.\varepsilon : \psi\} \not\subseteq \mathcal{B}$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma.\varepsilon : \varphi, \sigma.\varepsilon : \psi\}$.
$\vee$-rule:	If $\sigma.\varepsilon : \varphi \vee \psi \in \mathcal{B}$, and $\{\sigma.\varepsilon : \varphi, \sigma.\varepsilon : \psi\} \cap \mathcal{B} = \emptyset$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma.\varepsilon : \phi\}$ for some $\phi \in \{\varphi, \psi\}$.

Fig. 1 Tableau expansion rules on inverse roles, conjunction formulas and disjunction formulas

$;_{[\,]}$-rule:	If $\sigma.\varepsilon : [\pi_1;\pi_2]\varphi \in \mathcal{B}$, and $\sigma.\varepsilon : [\pi_1][\pi_2]\varphi \notin \mathcal{B}$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : [\pi_1][\pi_2]\varphi\}$.
$;_{<>}$-rule:	If $\sigma.\varepsilon :< \pi_1;\pi_2 > \varphi \in \mathcal{B}$, and $\sigma.\varepsilon :< \pi_1 >< \pi_2 > \varphi \notin \mathcal{B}$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon :< \pi_1 >< \pi_2 > \varphi\}$.
$?_{[\,]}$-rule:	If $\sigma.\varepsilon : [\psi?]\varphi \in \mathcal{B}$, and $\{\sigma.\varepsilon : \neg\psi, \sigma.\varepsilon : \varphi\} \cap \mathcal{B} = \emptyset$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \phi\}$ for some $\phi \in \{\neg\psi, \varphi\}$.
$?_{<>}$-rule:	If $\sigma.\varepsilon :< \psi? > \varphi \in \mathcal{B}$, and $\{\sigma.\varepsilon : \psi, \sigma.\varepsilon : \varphi\} \not\subseteq \mathcal{B}$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \psi, \sigma.\varepsilon : \varphi\}$.
$\cup_{[\,]}$-rule:	If $\sigma.\varepsilon : [\pi_1 \cup \pi_2]\varphi \in \mathcal{B}$, and $\{\sigma.\varepsilon : [\pi_1]\varphi, \sigma.\varepsilon : [\pi_2]\varphi\} \not\subseteq \mathcal{B}$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : [\pi_1]\varphi, \sigma.\varepsilon : [\pi_2]\varphi\}$.
$\cup_{<>}$-rule:	If $\sigma.\varepsilon :< \pi_1 \cup \pi_2 > \varphi \in \mathcal{B}$, and $\{\sigma.\varepsilon :< \pi_1 > \varphi, \sigma.\varepsilon :< \pi_2 > \varphi\} \cap \mathcal{B} = \emptyset$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \psi\}$ for some $\psi \in \{< \pi_1 > \varphi, < \pi_2 > \varphi\}$.
$*_{[\,]}$-rule:	If $\sigma.\varepsilon : [\pi^*]\varphi \in \mathcal{B}$, and $\{\sigma.\varepsilon : \varphi, \sigma.\varepsilon : [\pi][\pi^*]\varphi\} \not\subseteq \mathcal{B}$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \varphi, \sigma.\varepsilon : [\pi][\pi^*]\varphi\}$.
$*_{<>}$-rule:	If $\sigma.\varepsilon :< \pi^* > \varphi \in \mathcal{B}$, and
	there is no character string X with $X \doteq< \pi^* > \varphi \in \mathcal{B}$ and $\sigma.\varepsilon : X \in \mathcal{B}$,
	then introduce a new character string X, and set $\mathcal{B}$:= $\mathcal{B} \cup \{X \doteq< \pi^* > \varphi, \sigma.\varepsilon : X\}$.
X-rule:	If $\sigma.\varepsilon : X \in \mathcal{B}$ with $X \doteq< \pi^* > \varphi \in \mathcal{B}$, and $\{\sigma.\varepsilon : \varphi, \sigma.\varepsilon :< \pi > X\} \cap \mathcal{B} = \emptyset$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \varphi\}$, or set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \neg\varphi, \sigma.\varepsilon :< \pi > X\}$.

Fig. 2 Tableau expansion rules on non-atomic actions

the $atom_{[\,]}$-rule, if some prefix $\sigma_i.\varepsilon_i$ exists with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$, then the branch will be expanded to guarantee that either $\sigma_i.\varepsilon_i : \varphi \in \mathcal{B}$ or $\{\sigma.\varepsilon : \phi^{\neg} \mid \phi \in P\} \cap \mathcal{B} \neq \emptyset$.

Tableau expansion rules presented in Fig. 4 are based on the intuition that, due to the minimal-change semantics of actions, the state denoted by the prefix $\sigma.\varepsilon$ can be treated as the result of executing a special atomic action $(\emptyset, \varepsilon)$ on the state denoted by the initial prefix $\sigma_0.\varepsilon_0$. Therefore, for any ABox assertion φ prefixed by $\sigma.\varepsilon$, if $\varphi \notin \varepsilon$, then an ABox assertion φ' can be constructed to guarantee that φ holds on the state denoted by $\sigma.\varepsilon$ if and only if φ' holds on the state denoted by $\sigma_0.\varepsilon_0$; as a result, we can

$atom_{<>}$-rule:	If $\sigma.\varepsilon :< (P, E) > \varphi \in \mathcal{B}$, and if $\{\sigma.\varepsilon : \phi \mid \phi \in P\} \not\subseteq \mathcal{B}$ or no prefix $\sigma_i.\varepsilon_i$ exists with both $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$ and $\sigma_i.\varepsilon_i : \varphi \in \mathcal{B}$,
	then: if there is no prefix $\sigma_i.\varepsilon_i$ with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$,
	then introduce a prefix $\sigma'.\varepsilon' := \sigma; (P, E).(\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$ and
	set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \phi \mid \phi \in P\} \cup \{\sigma'.\varepsilon' : \varphi\} \cup \{\sigma'.\varepsilon' : \psi \mid \psi \in \varepsilon'\}$,
	else find a prefix $\sigma_i.\varepsilon_i$ with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$ and
	set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \phi \mid \phi \in P\} \cup \{\sigma_i.\varepsilon_i : \varphi\} \cup \{\sigma_i.\varepsilon_i : \psi \mid \psi \in \varepsilon_i\}$.
$atom_{[\,]}$-rule:	If $\sigma.\varepsilon : [(P, E)]\varphi \in \mathcal{B}$, $\{\sigma.\varepsilon : \phi^{\neg} \mid \phi \in P\} \cap \mathcal{B} = \emptyset$, and there is a prefix $\sigma_i.\varepsilon_i$ with both $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$ and $\sigma_i.\varepsilon_i : \varphi \notin \mathcal{B}$,
	then set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma_i.\varepsilon_i : \varphi\}$, or set $\mathcal{B}$:= $\mathcal{B} \cup \{\sigma.\varepsilon : \phi^{\neg}\}$ for some $\phi \in P$.

Fig. 3 Tableau expansion rules on atomic actions

$Back_r$-rule:	If $\sigma.\varepsilon : \varphi \in \mathcal{B}$, φ is an ABox assertion of the form $R(p,q)$ or $\neg R(p,q)$, $\varphi \notin \varepsilon$, and $\sigma_0.\varepsilon_0 : \varphi \notin \mathcal{B}$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma_0.\varepsilon_0 : \varphi\}$.
$Back_c$-rule:	If $\sigma.\varepsilon : \varphi \in \mathcal{B}$, φ is an ABox assertion of the form $C(p)$, $\varphi \notin \varepsilon$, and $\sigma_0.\varepsilon_0 : C^{Regress(\varepsilon,\mathcal{T})}(p) \notin \mathcal{B}$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma_0.\varepsilon_0 : C^{Regress(\varepsilon,\mathcal{T})}(p)\}$.
$Back_{\neg c}$-rule:	If $\sigma.\varepsilon : \varphi \in \mathcal{B}$, φ is an ABox assertion of the form $\neg C(p)$, $\varphi \notin \varepsilon$, and $\sigma_0.\varepsilon_0 : \neg C^{Regress(\varepsilon,\mathcal{T})}(p) \notin \mathcal{B}$, then set $\mathcal{B} := \mathcal{B} \cup \{\sigma_0.\varepsilon_0 : \neg C^{Regress(\varepsilon,\mathcal{T})}(p)\}$.

Fig. 4 Tableau expansion rules on ABox assertions

expand the branch with the prefixed formula $\sigma_0.\varepsilon_0 : \varphi'$. More precisely, if φ is of the form $R(p,q)$ or $\neg R(p,q)$, then the corresponding ABox assertion φ' is equal with φ, and therefore the prefixed formula $\sigma_0.\varepsilon_0 : \varphi$ will be incorporated into the branch by the $Back_r$-rule. However, if φ is of the form $C(p)$ or $\neg C(p)$, then we need to construct some concept $C^{Regress(\varepsilon,\mathcal{T})}$ before incorporating the prefixed formula $\sigma_0.\varepsilon_0 : C^{Regress(\varepsilon,\mathcal{T})}(p)$ or $\sigma_0.\varepsilon_0 : \neg C^{Regress(\varepsilon,\mathcal{T})}(p)$ into the branch according to the $Back_c$- or $Back_{\neg c}$-rule, where the concept $C^{Regress(\varepsilon,\mathcal{T})}$ should satisfy that $(C^{Regress(\varepsilon,\mathcal{T})})^{I(w_0)} = C^{I(w)}$ for any two states w_0 and w denoted by the prefixes $\sigma_0.\varepsilon_0$ and $\sigma.\varepsilon$ respectively.

In order to construct such a concept, we adopt a process proposed by Liu et al. for constructing updated concepts in their ABox updating algorithm [23], and modify it to act as a regression operator. More precisely, for any prefix $\sigma.\varepsilon$ and any concept C, the concept $C^{Regress(\varepsilon,\mathcal{T})}$ is constructed according to the following steps:

1. Construct the expansion $C_{\mathcal{T}}$ of C w.r.t. $\mathcal{T}$.
2. Let $Obj(\varepsilon)$ be a set composed of all the individual names occurring in ε. Return the concept $(C_{\mathcal{T}})^{Regress(\varepsilon)}$ which is defined inductively as follows:

- $A^{Regress(\varepsilon)} := A \sqcup \left(\bigsqcup_{A(p)\in\varepsilon} \{p\}\right) \sqcap \left(\bigsqcap_{\neg A(p)\in\varepsilon} \neg\{p\}\right)$ for any concept name A;
- $\{p\}^{Regress(\varepsilon)} := \{p\}$;
- $(\neg D)^{Regress(\varepsilon)} := \neg D^{Regress(\varepsilon)}$.
- $(D \sqcup D')^{Regress(\varepsilon)} := D^{Regress(\varepsilon)} \sqcup D'^{Regress(\varepsilon)}$;
- $(D \sqcap D')^{Regress(\varepsilon)} := D^{Regress(\varepsilon)} \sqcap D'^{Regress(\varepsilon)}$;
- $$(\forall R.D)^{Regress(\varepsilon)} := \left(\left(\bigsqcup_{p\in Obj(\varepsilon)} \{p\}\right) \sqcup \forall R.D^{Regress(\varepsilon)}\right) \sqcap \left(\left(\bigsqcap_{p\in Obj(\varepsilon)} \neg\{p\}\right) \sqcup \forall R.\left(\left(\bigsqcup_{q\in Obj(\varepsilon)} \{q\}\right) \sqcup D^{Regress(\varepsilon)}\right)\right) \sqcap \bigsqcap_{p,q\in Obj(\varepsilon), R(p,q)\notin\varepsilon, \neg R(p,q)\notin\varepsilon} \left(\neg\{p\} \sqcup \forall R.\left(\neg\{q\} \sqcup D^{Regress(\varepsilon)}\right)\right) \sqcap \bigsqcap_{R(p,q)\in\varepsilon} \left(\neg\{p\} \sqcup @_q D^{Regress(\varepsilon)}\right);$$
- $$(\exists R.D)^{Regress(\varepsilon)} := \left(\left(\bigsqcap_{p\in Obj(\varepsilon)} \neg\{p\}\right) \sqcap \exists R.D^{Regress(\varepsilon)}\right) \sqcup \left(\left(\bigsqcup_{p\in Obj(\varepsilon)} \{p\}\right) \sqcap \exists R.\left(\left(\bigsqcap_{q\in Obj(\varepsilon)} \neg\{q\}\right) \sqcap D^{Regress(\varepsilon)}\right)\right)$$

$$\sqcup \bigsqcup_{p,q\in Obj(\varepsilon),R(p,q)\notin\varepsilon,\neg R(p,q)\notin\varepsilon}\left(\{p\}\sqcap\exists R.\left(\{q\}\sqcap D^{Regress(\varepsilon)}\right)\right)$$

$$\sqcup \bigsqcup_{R(p,q)\in\varepsilon}\left(\{p\}\sqcap @_q D^{Regress(\varepsilon)}\right);$$

– $(\leq nR.D)^{Regress(\varepsilon)} := \left(\left(\bigsqcap_{p\in Obj(\varepsilon)}\neg\{p\}\right)\sqcap\left(\leq nR.D^{Regress(\varepsilon)}\right)\right)$

$$\sqcup \bigsqcup_{p\in Obj(\varepsilon)}\left(\{p\}\sqcap \bigsqcup_{n_1+n_2+n_3=n}\left(\leq n_1 R.\left(\left(\bigsqcap_{q\in Obj(\varepsilon)}\neg\{q\}\right)\sqcap D^{Regress(\varepsilon)}\right)\right.\right.$$

$$\sqcap\leq n_2 R.\left(\left(\bigsqcup_{q\in Obj(\varepsilon),R(p,q)\notin\varepsilon,\neg R(p,q)\notin\varepsilon}\{q\}\right)\sqcap D^{Regress(\varepsilon)}\right)$$

$$\left.\left.\sqcap\bigsqcap_{O\subseteq\{q|R(p,q)\in\varepsilon\},\sharp O=n_3+1}\left(\bigsqcup_{q\in O}\neg @_q D^{Regress(\varepsilon)}\right)\right)\right),$$

where n_1, n_2, n_3 are positive integers;

– $(\geq nR.D)^{Regress(\varepsilon)} := \left(\left(\bigsqcap_{p\in Obj(\varepsilon)}\neg\{p\}\right)\sqcap\left(\geq nR.D^{Regress(\varepsilon)}\right)\right)$

$$\sqcup \bigsqcup_{p\in Obj(\varepsilon)}\left(\{p\}\sqcap \bigsqcup_{n_1+n_2+n_3=n}\left(\geq n_1 R.\left(\left(\bigsqcap_{q\in Obj(\varepsilon)}\neg\{q\}\right)\sqcap D^{Regress(\varepsilon)}\right)\right.\right.$$

$$\sqcap\geq n_2 R.\left(\left(\bigsqcup_{q\in Obj(\varepsilon),R(p,q)\notin\varepsilon,\neg R(p,q)\notin\varepsilon}\{q\}\right)\sqcap D^{Regress(\varepsilon)}\right)$$

$$\left.\left.\sqcap\bigsqcup_{O\subseteq\{q\mid R(p,q)\in\varepsilon\},\sharp O=n_3}\left(\bigsqcap_{q\in O}@_q D^{Regress(\varepsilon)}\right)\right)\right),$$

where n_1, n_2, n_3 are positive integers;

– $(@_p D)^{Regress(\varepsilon)} := @_p D^{Regress(\varepsilon)}$.

The property for which the concept $C^{Regress(\varepsilon,\mathcal{T})}$ is technically designed will be stated and proved in Lemma 2 of the next subsection.

Definition 12 A branch $\mathcal{B}$ is *contradictory* if and only if there is some prefix $\sigma.\varepsilon$ and some formula φ such that both $\sigma.\varepsilon : \varphi \in \mathcal{B}$ and $\sigma.\varepsilon : \neg\varphi \in \mathcal{B}$.

Definition 13 A branch $\mathcal{B}$ is *completed* if and only if it can not be expanded by any tableau expansion rules presented in Figs. 1, 2, 3 and 4.

Definition 14 An eventuality record $X \doteq < \pi^* > \varphi$ is *fulfilled* in a branch $\mathcal{B}$ if and only if there is a prefix $\sigma.\varepsilon$ such that both $\sigma.\varepsilon : X \in \mathcal{B}$, and $\sigma.\varepsilon : \varphi \in \mathcal{B}$.

Definition 15 A branch $\mathcal{B}$ is *ignorable* if and only if it is completed but contains some eventuality record $X \doteq < \pi^* > \varphi$ which is not fulfilled.

We are now ready to finish the description of *the satisfiability-checking algorithm*:

Algorithm 1 *The satisfiability of a formula ϕ w.r.t. an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ActBox $\mathcal{A}_C$ is decided according to the following steps:*

1. *Construct a branch $\mathcal{B} := \{\sigma_0.\varepsilon_0 : nf(\phi)\}$.*
2. *If tableau expansion rules in Figs. 1, 2, 3 and 4 can be applied to $\mathcal{B}$ in such a way that they yield a completed branch $\mathcal{B}'$, and*
   - *$\mathcal{B}'$ is neither contradictory nor ignorable, and*
   - *the initial view $IV_{\mathcal{B}'}$ of $\mathcal{B}'$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$,*

   then the algorithm returns "TRUE", else returns "FALSE".

In this algorithm, since the initial view $IV_{\mathcal{B}'}$ is an ABox of the description logic $X^@$, and the @ constructor doesn't occur in the TBox $\mathcal{T}$, the consistency of $IV_{\mathcal{B}'}$ w.r.t. $\mathcal{R}$ and $\mathcal{T}$ can be decided with the reasoning mechanisms investigated in Section 2.

5.2 Termination and Correctness of the Algorithm

Some notations are necessary for demonstrating the termination of the algorithm.

Firstly, for any role, concept, formula, action, role inclusion axiom, concept definition, atomic action definition, RBox, TBox, ABox or ActBox X, we use $|X|$ to denote the *size of* X and define it inductively as follows:

- $|X| = 1$ if $X \in N_R \cup N_C \cup N_I \cup N_A$;
- $|X| = |R| + 1$ if X is a role of the form R^-;
- $|X| = |C| + 1$ if X is a concept of the form $\neg C$;
- $|X| = |p|$ if X is a concept of the form $\{p\}$;
- $|X| = |p| + |C| + 1$ if X is a concept of the form $@_p C$;
- $|X| = |C| + |C'| + 1$ if X is a concept of the form $C \sqcup C'$ or $C \sqcap C'$;
- $|X| = |R| + |C| + 1$ if X is a concept of the form $\forall R.C$ or $\exists R.C$;
- $|X| = n + |R| + |C| + 1$ if X is a concept of the form $\leq nR.C$ or $\geq nR.C$;[2]
- $|X| = |C| + |p|$ if X is an ABox assertion of the form $C(p)$;
- $|X| = |R| + |p| + |q|$ if X is an ABox assertion of the form $R(p, q)$;
- $|X| = |\varphi| + 1$ if X is a formula of the form $\neg\varphi$;
- $|X| = |\pi| + |\varphi| + 1$ if X is a formula of the form $< \pi > \varphi$ or $[\pi]\varphi$;
- $|X| = |\varphi| + |\varphi'| + 1$ if X is a formula of the form $\varphi \wedge \varphi'$ or $\varphi \vee \varphi'$;
- $|X| = |\pi| + |\pi'| + 1$ if X is an action of the form $\pi \cup \pi'$ or $\pi; \pi'$;
- $|X| = |\varphi| + 1$ if X is an action of the form $\varphi?$;
- $|X| = |\pi| + 1$ if X is an action of the form π^*;

[2]Here the numbers inside qualified number restrictions are assumed to be written in unary [23].

- $|X| = |R| + |R'| + 1$ if X is a role inclusion axiom of the form $R \sqsubseteq R'$;
- $|X| = |A| + |C| + 1$ if X is a concept definition of the form $A \equiv C$;
- $|X| = |\alpha| + \sum_{\varphi \in P}(|\varphi|) + \sum_{\phi \in E}(|\phi|) + 1$ if X is an atomic action definition $\alpha \equiv (P, E)$;
- $|X| = \sum_{x \in X}(|x|)$ if X is an ABox, RBox, TBox or ActBox.

Secondly, for any concept C, we use $d(C)$ to denote the maximal nesting depth of existential restrictions, value restrictions and quantified number restrictions in C, and define it inductively as follows:

- $d(C) = 0$ if $C \in N_C$ or C is of the form $\{p\}$;
- $d(C) = d(C')$ if C is of the form $\neg C'$ or $@_p C'$;
- $d(C) = max(d(C'), d(C''))$ if C is of the form $C' \sqcup C''$ or $C' \sqcap C''$;
- $d(C) = d(C') + 1$ if C is of the form $\forall R.C'$, $\exists R.C'$, $\leq nR.C'$ or $\geq nR.C'$.

Thirdly, for any formula φ and any RBox $\mathcal{R}$, we use $cl_{\mathcal{R}}(\varphi)$ to denote the *relevant sub-formulas* of φ w.r.t. $\mathcal{R}$, and define it as the smallest set satisfying the following conditions:

- $nf(\varphi) \in cl_{\mathcal{R}}(\varphi)$;
- if $\psi \in cl_{\mathcal{R}}(\varphi)$ and ψ is not started with the negation sign, then $\neg\psi \in cl_{\mathcal{R}}(\varphi)$;
- if $\neg\psi \in cl_{\mathcal{R}}(\varphi)$, then $\psi \in cl_{\mathcal{R}}(\varphi)$;
- if $R(p, q) \in cl_{\mathcal{R}}(\varphi)$, then $Inv(R)(q, p) \in cl_{\mathcal{R}}(\varphi)$;
- if $R(p, q) \in cl_{\mathcal{R}}(\varphi)$ and $R \sqsubseteq R' \in \mathcal{R}$, then $R'(p, q) \in cl_{\mathcal{R}}(\varphi)$;
- if $\neg R'(p, q) \in cl_{\mathcal{R}}(\varphi)$ and $R \sqsubseteq R' \in \mathcal{R}$, then $\neg R(p, q) \in cl_{\mathcal{R}}(\varphi)$;
- if $\psi \in cl_{\mathcal{R}}(\varphi)$ and ψ is of the form $\psi_1 \vee \psi_2$, $\psi_1 \wedge \psi_2$, $<\psi_1?>\psi_2$ or $[\psi_1?]\psi_2$, then $\{\psi_1, \psi_2\} \subseteq cl_{\mathcal{R}}(\varphi)$;
- if $<\pi_1; \pi_2>\psi \in cl_{\mathcal{R}}(\varphi)$, then $<\pi_1><\pi_2>\psi \in cl_{\mathcal{R}}(\varphi)$;
- if $[\pi_1; \pi_2]\psi \in cl_{\mathcal{R}}(\varphi)$, then $[\pi_1][\pi_2]\psi \in cl_{\mathcal{R}}(\varphi)$;
- if $<\pi_1 \cup \pi_2>\psi \in cl_{\mathcal{R}}(\varphi)$, then $\{<\pi_1>\psi, <\pi_2>\psi\} \subseteq cl_{\mathcal{R}}(\varphi)$;
- if $[\pi_1 \cup \pi_2]\psi \in cl_{\mathcal{R}}(\varphi)$, then $\{[\pi_1]\psi, [\pi_2]\psi\} \subseteq cl_{\mathcal{R}}(\varphi)$;
- if $<\pi^*>\psi \in cl_{\mathcal{R}}(\varphi)$, then $\{\psi, <\pi><\pi^*>\psi\} \subseteq cl_{\mathcal{R}}(\varphi)$;
- if $[\pi^*]\psi \in cl_{\mathcal{R}}(\varphi)$, then $\{\psi, [\pi][\pi^*]\psi\} \subseteq cl_{\mathcal{R}}(\varphi)$;
- if $\psi \in cl_{\mathcal{R}}(\varphi)$ and ψ is of the form $<(P, E)>\psi_1$ or $[(P, E)]\psi_1$, then $P \cup E \cup \{\psi_1\} \subseteq cl_{\mathcal{R}}(\varphi)$.

Fourthly, we use $Ass_{\mathcal{R}}(\varphi)$ to denote the set of all the ABox assertions contained in $cl_{\mathcal{R}}(\varphi)$.

Finally, for any formula φ and any RBox $\mathcal{R}$, we use $AtomAct(\varphi)$ to denote the set of all the atomic actions occurring in $nf(\varphi)$, and use $Eff_{\mathcal{R}}(\varphi)$ to denote a set defined as follows:

$$Eff_{\mathcal{R}}(\varphi) \triangleq \bigcup_{(P,E) \in AtomAct(\varphi)} E^*_{\mathcal{R}}$$

where $E^*_{\mathcal{R}}$ is the closure of the ABox E w.r.t. the RBox $\mathcal{R}$.

It is obvious that the cardinalities $\sharp cl_{\mathcal{R}}(\varphi)$, $\sharp Ass_{\mathcal{R}}(\varphi)$ and $\sharp Eff_{\mathcal{R}}(\varphi)$ are all linearly bounded by $|nf(\varphi)| \times |\mathcal{R}|$.

Now we are ready to demonstrate the termination of the algorithm.

Theorem 7 *Algorithm* 1 *terminates.*

Proof Let f:= $|nf(\phi)|$, c:= $\sharp cl_{\mathcal{R}}(\phi)$, a:= $\sharp Ass_{\mathcal{R}}(\phi)$ and e:= $\sharp Eff_{\mathcal{R}}(\phi)$. Let m be the maximal one among $|\psi|$ for every $\psi \in Ass_{\mathcal{R}}(\phi)$. Then, it is obvious that $m \leq f$; the numbers c, a and e are all linearly bounded by $f \times |\mathcal{R}|$; and the number f is linearly bounded by $|\phi| \times |\mathcal{A}_C|$. Furthermore, the following properties are straightforward for the algorithm.

1. For each application of any tableau expansion rule on a branch $\mathcal{B}$, the number of possible expansions is finite, and the cardinality of the branch will be strictly increased for every expansion.
2. According to the definition of the $atom_{<>}$-rule, it must be $\varepsilon \subseteq Eff_{\mathcal{R}}(\phi)$, $\varepsilon' \subseteq Eff_{\mathcal{R}}(\phi)$ and $\varepsilon \neq \varepsilon'$ for any prefixes $\sigma.\varepsilon$ and $\sigma'.\varepsilon'$ occurring in the branch; therefore, the number of prefixes introduced during the execution of Algorithm 1 is bounded by 2^e.
3. For each non-initial prefix $\sigma.\varepsilon$, the number of formulas prefixed by it is bounded by c, and the number of ABox assertions prefixed by it is bounded by a.
4. The number of formulas prefixed by the initial prefix $\sigma_0.\varepsilon_0$ is bounded by $c + a \times (2^e - 1)$, where $a \times (2^e - 1)$ is the number of ABox assertions which might be introduced by applying the $Back_r$-, $Back_c$-, and $Back_{\neg c}$-rules.
5. For each prefixed formula $\sigma_0.\varepsilon_0 : C^{Regress(\varepsilon,\mathcal{T})}(p)$ (resp. $\sigma_0.\varepsilon_0 : \neg C^{Regress(\varepsilon,\mathcal{T})}(p)$) introduced by applying the $Back_c$-rule (resp. the $Back_{\neg c}$-rule), let $C_{\mathcal{T}}$ be the expansion of C w.r.t. $\mathcal{T}$. Then it must be $d(C_{\mathcal{T}}) \leq |C| + |\mathcal{T}|$ and $|C_{\mathcal{T}}| \leq |C| \times 2^{q_1(|\mathcal{T}|)}$ for some polynomial q_1. Furthermore, according to the construction of the concept $(C_{\mathcal{T}})^{Regress(\varepsilon)}$, and be similar with the result presented in Theorem 36 of [25] for the ABox updating algorithm, we can find some polynomial q_2 such that $|(C_{\mathcal{T}})^{Regress(\varepsilon)}| \leq |C_{\mathcal{T}}| \times (q_2(\sharp Obj(\varepsilon)))^{d(C_{\mathcal{T}})}$. At the same time, since $\varepsilon \subseteq Eff_{\mathcal{R}}(\phi)$, the number $\sharp Obj(\varepsilon)$ is linearly bounded by e. Therefore, to sum up, we can find two polynomials q_1 and q_2' such that $|C^{Regress(\varepsilon,\mathcal{T})}(p)| = |(C_{\mathcal{T}})^{Regress(\varepsilon)}(p)| \leq |C| \times 2^{q_1(|\mathcal{T}|)} \times (q_2'(e))^{|C|+|\mathcal{T}|} \leq m \times 2^{q_1(|\mathcal{T}|)} \times (q_2'(e))^{m+|\mathcal{T}|}$.

Now, according to Properties 2, 3 and 4 listed above, for any branch generated during the execution of Algorithm 1, the number of prefixed formulas contained in it is bounded by $(2^e - 1) \times c + (\, c + a \times (2^e - 1)\,)$. Therefore, together with Property 1, the number of branches which will be investigated by the algorithm is finite.

For each branch $\mathcal{B}$ investigated by the algorithm, the number of ABox assertions contained in the initial view $IV_{\mathcal{B}}$ is bounded by $a + a \times (2^e - 1) = a \times 2^e$. Therefore, according to Property 5, we can find two polynomials p_1 and p_2 such that $|IV_{\mathcal{B}}| \leq (a \times 2^e) \times (m \times 2^{p_1(|\mathcal{T}|)} \times (p_2(e))^{m+|\mathcal{T}|})$. So, the consistency of $IV_{\mathcal{B}}$ w.r.t. $\mathcal{R}$ and $\mathcal{T}$ can be decided with terminable procedures provided by the description logic $\mathcal{X}^@$.

To sum up, the algorithm terminates. □

Furthermore, according to the above proof and based on the result presented in Theorem 1, the following result is straightforward:

Corollary 1 *For the family $DDL(\mathcal{X}^@)$ of dynamic description logics, the complexity upper-bound of Algorithm 1 is EXPSPACE if $\mathcal{X} \in \{\mathcal{ALCO}, \mathcal{ALCHO}, \mathcal{ALCOQ}, \mathcal{ALCHOQ}\}$, and is N2EXPTIME if $\mathcal{X} \in \{\mathcal{ALCOI}, \mathcal{ALCHOI}, \mathcal{ALCOIQ}, \mathcal{ALCHOIQ}\}$.*

In order to demonstrate the correctness of the satisfiability-checking algorithm, we introduce three notations in the following paragraphs.

Firstly, we introduce branch-model mappings to act as bridges between branches and models:

Definition 16 Let $\mathcal{T}$, $\mathcal{B}$ and $M = (W, T, \Delta, I)$ be a TBox, a branch and a model respectively. A *branch-model mapping* ι w.r.t. $\mathcal{T}$, $\mathcal{B}$ and M is a function from prefixes occurring in $\mathcal{B}$ to states of M, satisfying that for each pair of prefixes $\sigma.\varepsilon$ and $\sigma'.\varepsilon'$ occurring in $\mathcal{B}$: if $\sigma' = \sigma; (P, E)$ and $\varepsilon' = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$, then:

- $A^{I(\iota(\sigma'.\varepsilon'))} = (A^{I(\iota(\sigma.\varepsilon))} \cup \{\, p^I \mid A(p) \in E^*_{\mathcal{R}} \,\}) \setminus \{\, p^I \mid \neg A(p) \in E^*_{\mathcal{R}} \,\}$ for each concept name A which is primitive w.r.t. $\mathcal{T}$, and
- $R^{I(\iota(\sigma'.\varepsilon'))} = (\, R^{I(\iota(\sigma.\varepsilon))} \cup \{\, (p^I, q^I) \mid R(p, q) \in E^*_{\mathcal{R}} \,\}) \setminus \{\, (p^I, q^I) \mid \neg R(p, q) \in E^*_{\mathcal{R}} \}$ for each role name R.

In this definition, it should be noted that the initial prefix $\sigma_0.\varepsilon_0$ is also mapped to some state of M, although the concrete state denoted by $\iota(\sigma_0.\varepsilon_0)$ is not specified here.

A branch-model mapping holds the following properties:

Lemma 1 *Let $\mathcal{T}$, $\mathcal{B}$ and $M = (W, T, \Delta, I)$ be a TBox, a branch and a model respectively; let ι be a function from prefixes occurring in $\mathcal{B}$ to states of M. Then, ι is a branch-model mapping w.r.t. $\mathcal{T}$, $\mathcal{B}$ and M if and only if the following statements hold for each prefix $\sigma.\varepsilon$ occurring in $\mathcal{B}$:*

- $A^{I(\iota(\sigma.\varepsilon))} = (A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{\, p^I \mid A(p) \in \varepsilon \,\}) \setminus \{\, p^I \mid \neg A(p) \in \varepsilon \,\}$ *for each concept name A which is primitive w.r.t. $\mathcal{T}$, and*
- $R^{I(\iota(\sigma.\varepsilon))} = (R^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{\, (p^I, q^I) \mid R(p, q) \in \varepsilon \,\}) \setminus \{\, (p^I, q^I) \mid \neg R(p, q) \in \varepsilon \,\}$ *for each role name R.*

Proof (*The Only-if direction*) The proof is by induction on the construction of prefixes. If $\sigma.\varepsilon$ is the initial prefix $\sigma_0.\varepsilon_0$, then the result is straightforward since $\varepsilon = \varepsilon_0 = \emptyset$.

Assume the result hold for some prefix $\sigma.\varepsilon$. Let $\sigma'.\varepsilon'$ be a prefix with $\sigma' = \sigma; (P, E)$ and $\varepsilon' = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. Then, for any concept name A which is primitive w.r.t. $\mathcal{T}$, we have

$$\left(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \left\{p^I \middle| A(p) \in \varepsilon'\right\}\right) \setminus \left\{p^I \middle| \neg A(p) \in \varepsilon'\right\} \tag{1}$$

$$= \left(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \left\{p^I \mid A(p) \in \left(\varepsilon \setminus \left(E^*_{\mathcal{R}}\right)^{\neg}\right) \cup E^*_{\mathcal{R}}\right\}\right) \setminus \left\{p^I \mid \neg A(p) \in \left(\varepsilon \setminus \left(E^*_{\mathcal{R}}\right)^{\neg}\right) \cup E^*_{\mathcal{R}}\right\} \tag{2}$$

$$= \left[A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \left(\left\{p^I \mid A(p) \in \varepsilon\right\} \setminus \left\{p^I \mid A(p) \in (E^*_{\mathcal{R}})^{\neg}\right\}\right) \cup \left\{p^I \mid A(p) \in E^*_{\mathcal{R}}\right\}\right] \setminus \left[\left(\left\{p^I \middle| \neg A(p) \in \varepsilon\right\} \setminus \left\{p^I \middle| \neg A(p) \in \left(E^*_{\mathcal{R}}\right)^{\neg}\right\}\right) \cup \left\{p^I \middle| \neg A(p) \in E^*_{\mathcal{R}}\right\}\right] \tag{3}$$

$$= \Big[\Big[A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \Big(\{p^I | A(p) \in \varepsilon\} \backslash \Big\{ p^I | A(p) \in \big(E^*_{\mathcal{K}}\big)^{\neg} \Big\} \Big) \cup \big\{ p^I | A(p) \in E^*_{\mathcal{K}} \big\} \Big]$$
$$\setminus \Big(\big\{ p^I \mid \neg A(p) \in \varepsilon \big\} \setminus \Big\{ p^I \mid \neg A(p) \in \big(E^*_{\mathcal{K}}\big)^{\neg} \Big\} \Big) \Big]$$
$$\setminus \big\{ p^I \mid \neg A(p) \in E^*_{\mathcal{K}} \big\} \tag{4}$$

$$= \Big[\Big[\Big(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \Big(\big\{ p^I \mid A(p) \in \varepsilon \big\} \setminus \{ p^I \mid A(p) \in \big(E^*_{\mathcal{K}}\big)^{\neg} \} \Big) \Big)$$
$$\setminus \Big(\big\{ p^I \mid \neg A(p) \in \varepsilon \big\} \setminus \Big\{ p^I \mid \neg A(p) \in \big(E^*_{\mathcal{K}}\big)^{\neg} \Big\} \Big) \Big] \cup \big\{ p^I \mid A(p) \in E^*_{\mathcal{K}} \big\} \Big]$$
$$\setminus \big\{ p^I \mid \neg A(p) \in E^*_{\mathcal{K}} \big\} \tag{5}$$

$$= \Big[\Big[\Big(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \Big(\big\{ p^I \mid A(p) \in \varepsilon \big\} \setminus \Big\{ p^I \mid A(p) \in \big(E^*_{\mathcal{K}}\big)^{\neg} \Big\} \Big) \Big)$$
$$\setminus \big\{ p^I \mid \neg A(p) \in \varepsilon \big\} \Big] \cup \big\{ p^I \mid A(p) \in E^*_{\mathcal{K}} \big\} \Big]$$
$$\setminus \big\{ p^I \mid \neg A(p) \in E^*_{\mathcal{K}} \big\} \tag{6}$$

$$= \Big[\Big[\big(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \big\{ p^I \mid A(p) \in \varepsilon \big\} \big)$$
$$\setminus \big\{ p^I \mid \neg A(p) \in \varepsilon \big\} \Big] \cup \big\{ p^I \mid A(p) \in E^*_{\mathcal{K}} \big\} \Big]$$
$$\setminus \big\{ p^I \mid \neg A(p) \in E^*_{\mathcal{K}} \big\} \tag{7}$$

$$= \Big[A^{I(\iota(\sigma.\varepsilon))} \cup \big\{ p^I \mid A(p) \in E^*_{\mathcal{K}} \big\} \Big] \setminus \big\{ p^I \mid \neg A(p) \in E^*_{\mathcal{K}} \big\} \tag{8}$$

$$= A^{I(\iota(\sigma'.\varepsilon'))} \tag{9}$$

The transformation from set (2) to set (3) is based on the unique name assumption on individual names, so that the set $\{p^I \mid A(p) \in (\varepsilon \backslash (E^*_{\mathcal{K}})^{\neg}) \cup E^*_{\mathcal{K}}\}$ and the set $\{p^I \mid \neg A(p) \in (\varepsilon \backslash (E^*_{\mathcal{K}})^{\neg}) \cup E^*_{\mathcal{K}}\}$ can be replaced by $(\{p^I \mid A(p) \in \varepsilon\} \setminus \{p^I \mid A(p) \in (E^*_{\mathcal{K}})^{\neg}\}) \cup \{p^I \mid A(p) \in E^*_{\mathcal{K}}\}$ and $(\{p^I \mid \neg A(p) \in \varepsilon\} \setminus \{p^I \mid \neg A(p) \in (E^*_{\mathcal{K}})^{\neg}\}) \cup \{p^I \mid \neg A(p) \in E^*_{\mathcal{K}}\}$ respectively. The transformation from set (4) to set (5) is based on the fact that the set $\{p^I \mid A(p) \in E^*_{\mathcal{K}}\}$ and the set $\{p^I \mid \neg A(p) \in \varepsilon\} \setminus \{p^I \mid \neg A(p) \in (E^*_{\mathcal{K}})^{\neg}\}$ are disjoint, so that the order of the union operation and the difference operation can be exchanged. The transformation from set (5) to set (6) is based on the fact that the set $\{p^I \mid \neg A(p) \in (E^*_{\mathcal{K}})^{\neg}\}$ is equal with the set $\{p^I \mid A(p) \in E^*_{\mathcal{K}}\}$ which will be combined with $(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup (\{p^I \mid A(p) \in \varepsilon\} \setminus \{p^I \mid A(p) \in (E^*_{\mathcal{K}})^{\neg}\})) \setminus (\{p^I \mid \neg A(p) \in \varepsilon\} \setminus \{p^I \mid \neg A(p) \in (E^*_{\mathcal{K}})^{\neg}\})$, therefore we can remove the set $\{p^I \mid \neg A(p) \in (E^*_{\mathcal{K}})^{\neg}\}$ from the expression $(\{p^I \mid \neg A(p) \in \varepsilon\} \setminus \{p^I \mid \neg A(p) \in (E^*_{\mathcal{K}})^{\neg}\})$. The transformation from set (6) to set (7) is based on the fact that the set $\{p^I \mid A(p) \in (E^*_{\mathcal{K}})^{\neg}\}$ is equal with the set $\{p^I \mid \neg A(p) \in E^*_{\mathcal{K}}\}$ and consequently is disjoint with set (6); therefore, we can remove the set $\{p^I \mid A(p) \in (E^*_{\mathcal{K}})^{\neg}\}$ from the expression $(\{p^I \mid A(p) \in \varepsilon\} \setminus \{p^I \mid A(p) \in (E^*_{\mathcal{K}})^{\neg}\})$. The transformation from set (7) to set (8) is based on the inductive hypothesis, and the transformation from set (8) to set (9) is based on the definition of branch-model mappings.

For any role name R, it can be similarly demonstrated that $(R^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{(p^I, q^I) \mid R(p, q) \in \varepsilon'\}) \setminus \{(p^I, q^I) \mid \neg R(p, q) \in \varepsilon'\} = R^{I(\iota(\sigma'.\varepsilon'))}$.

(*The If direction*) Let $\sigma.\varepsilon$ and $\sigma'.\varepsilon'$ be a pair of prefixes occurring in $\mathcal{B}$ with both $\sigma' = \sigma; (P, E)$ and $\varepsilon' = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. Then, for any concept name A which is primitive w.r.t. $\mathcal{T}$, we have both $A^{I(\iota(\sigma.\varepsilon))} = (A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{p^I \mid A(p) \in \varepsilon\}) \setminus \{p^I \mid \neg A(p) \in \varepsilon\}$ and $A^{I(\iota(\sigma'.\varepsilon'))} = (A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{p^I \mid A(p) \in \varepsilon'\}) \setminus \{p^I \mid \neg A(p) \in \varepsilon'\}$. At the same time, according to the transformation from set (1) to set (7) presented above for the proof of the Only-if direction, we have $(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{p^I \mid A(p) \in \varepsilon'\}) \setminus \{p^I \mid \neg A(p) \in \varepsilon'\} = [\,[(A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{p^I | A(p) \in \varepsilon\}) \setminus \{p^I \mid \neg A(p) \in \varepsilon\}] \cup \{p^I \mid A(p) \in E^*_{\mathcal{R}}\}\,] \setminus \{p^I \mid \neg A(p) \in E^*_{\mathcal{R}}\}$. Therefore, we have $A^{I(\iota(\sigma'.\varepsilon'))} = (\,A^{I(\iota(\sigma.\varepsilon))} \cup \{p^I \mid A(p) \in E^*_{\mathcal{R}}\}\,) \setminus \{p^I \mid \neg A(p) \in E^*_{\mathcal{R}}\}$.

For any role name R, it can be similarly demonstrated that $R^{I(\iota(\sigma'.\varepsilon'))} = (R^{I(\iota(\sigma.\varepsilon))} \cup \{(p^I, q^I) \mid R(p, q) \in E^*_{\mathcal{R}}\}) \setminus \{(p^I, q^I) \mid \neg R(p, q) \in E^*_{\mathcal{R}}\}$. □

Corollary 2 *Let ι be a branch-model mapping w.r.t. a TBox $\mathcal{T}$, a branch $\mathcal{B}$ and a model $M = (W, T, \Delta, I)$. Then, for any prefix $\sigma.\varepsilon$ occurring in $\mathcal{B}$ and any primitive literal $\psi \in \varepsilon$, it must be $(M, \iota(\sigma.\varepsilon)) \models \psi$.*

Proof If $\sigma.\varepsilon$ is the initial prefix $\sigma_0.\varepsilon_0$, then the result is straightforward since $\varepsilon = \varepsilon_0 = \emptyset$.

Now, let $\sigma.\varepsilon$ be a non-initial prefix. We demonstrate the result by investigating the forms of every $\psi \in \varepsilon$. Firstly, suppose ψ is of the form $A(p)$. Since every atomic action definition is assumed to be PE-consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$ at the beginning of Section 5, we have $\neg A(p) \notin \varepsilon$ according to the construction of prefixes. Therefore, by Lemma 1 and based on the unique name assumption on individual names, we have $p^I \in A^{I(\iota(\sigma.\varepsilon))}$ and consequently $(M, \iota(\sigma.\varepsilon)) \models A(p)$. Secondly, suppose ψ is of the form $\neg A(p)$. Then, by Lemma 1, we have $p^I \notin A^{I(\iota(\sigma.\varepsilon))}$ and consequently $(M, \iota(\sigma.\varepsilon)) \models \neg A(p)$. Finally, if ψ is of the form $R(p, q)$ or $\neg R(p, q)$, then the result can be similarly demonstrated. □

The following corollary is an easy consequence of Lemma 1:

Corollary 3 *Let ι be a branch-model mapping w.r.t. a TBox $\mathcal{T}$, a branch $\mathcal{B}$ and a model M; let $M \models \mathcal{T}$; let $\sigma.\varepsilon$ and $\sigma'.\varepsilon'$ be two prefixes occurring in $\mathcal{B}$. Then, $\iota(\sigma.\varepsilon)$ and $\iota(\sigma'.\varepsilon')$ are the same state if $\varepsilon = \varepsilon'$.*

With the help of branch-model mappings, we can present and demonstrate the following property for the regression operator introduced in the satisfiability-checking algorithm.

Lemma 2 *Let $C^{Regress(\varepsilon,\mathcal{T})}$ be a concept constructed by the regression operator in the Back$_c$- or Back$_{\neg c}$-rule; let $M = (W, T, \Delta, I)$ be a model with $M \models \mathcal{T}$. Then, for any branch-model mapping ι w.r.t. $\mathcal{T}$, $\mathcal{B}$ and M, we have $(C^{Regress(\varepsilon,\mathcal{T})})^{I(\iota(\sigma_0.\varepsilon_0))} = C^{I(\iota(\sigma.\varepsilon))}$.*

Proof Let $C_{\mathcal{T}}$ be the expansion of C w.r.t. $\mathcal{T}$. According to the construction of $C^{Regress(\varepsilon,\mathcal{T})}$, we have $(C^{Regress(\varepsilon,\mathcal{T})})^{I(\iota(\sigma_0.\varepsilon_0))} = (C_{\mathcal{T}}{}^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}$. Furthermore, since $M \models \mathcal{T}$, we have $C_{\mathcal{T}}{}^{I(\iota(\sigma.\varepsilon))} = C^{I(\iota(\sigma.\varepsilon))}$. Therefore, we just need to demonstrate $(C_{\mathcal{T}}{}^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} = C_{\mathcal{T}}{}^{I(\iota(\sigma.\varepsilon))}$.

In the following proof, we will use the result that for any individuals $x, y \in \Delta$ and any role name $R \in N_R$: the statement $(x, y) \in R^{I(\iota(\sigma.\varepsilon))}$ holds if and only if one of the following statements holds:

- $x \notin \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge (x, y) \in R^{I(\iota(\sigma_0.\varepsilon_0))}$,
- $x \in \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge y \notin \bigcup_{q \in Obj(\varepsilon)} \{q^I\} \wedge (x, y) \in R^{I(\iota(\sigma_0.\varepsilon_0))}$,
- $\exists p, q \in Obj(\varepsilon).\ (x = p^I \wedge y = q^I \wedge \neg R(p, q) \notin \varepsilon \wedge R(p, q) \notin \varepsilon \wedge (x, y) \in R^{I(\iota(\sigma_0.\varepsilon_0))})$,
- $\exists p, q \in Obj(\varepsilon).\ (x = p^I \wedge y = q^I \wedge R(p, q) \in \varepsilon)$.

This result is straightforward according to Lemma 1.

Now, we prove $(C_{\mathcal{T}}{}^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} = C_{\mathcal{T}}{}^{I(\iota(\sigma.\varepsilon))}$ by induction on the structure of $C_{\mathcal{T}}$.

Case 1. $C_{\mathcal{T}}$ is a concept name A. Then we have

$$\begin{aligned}
&(A^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} \\
&\quad = \left(A \sqcup \left(\bigsqcup_{A(p) \in \varepsilon} \{p\} \right) \sqcap \left(\bigsqcap_{\neg A(p) \in \varepsilon} \neg\{p\} \right) \right)^{I(\iota(\sigma_0.\varepsilon_0))} \\
&\quad = (\, A^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{p^I | A(p) \in \varepsilon\}\,) \backslash \{p^I | \neg A(p) \in \varepsilon\}.
\end{aligned}$$

Therefore, by Lemma 1, we have $(A^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} = A^{I(\iota(\sigma.\varepsilon))}$.

Case 2. C is of the form $\{p\}$, $@_p D$, $\neg D$, $D \sqcup D'$ or $D \sqcap D'$. The result is straightforward.

Case 3. C is of the form $\forall R.D$. Then, for any individual $x \in \Delta$, we have

$$\begin{aligned}
& x \in ((\forall R.D)^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} \\
\text{iff}\quad & x \in \left(\bigsqcup_{p \in Obj(\varepsilon)} \{p\} \sqcup \forall R.D^{Regress(\varepsilon)} \right)^{I(\iota(\sigma_0.\varepsilon_0))} \\
& \wedge\, x \in \left(\bigsqcap_{p \in Obj(\varepsilon)} \neg\{p\} \sqcup \forall R.\left(\bigsqcup_{q \in Obj(\varepsilon')} \{q\} \sqcup D^{Regress(\varepsilon)} \right) \right)^{I(\iota(\sigma_0.\varepsilon_0))} \\
& \wedge\, x \in \bigcap_{p,q \in Obj(\varepsilon), R(p,q) \notin \varepsilon, \neg R(p,q) \notin \varepsilon} (\neg\{p\} \sqcup \forall R.(\neg\{q\} \sqcup D^{Regress(\varepsilon)}))^{I(\iota(\sigma_0.\varepsilon_0))} \\
& \wedge\, x \in \bigcap_{R(p,q) \in \varepsilon} (\neg\{p\} \sqcup @_q D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} \\
\text{iff}\quad & \forall y. \Bigg(\left(\left(x \notin \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge (x, y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \right) \rightarrow y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} \right) \\
& \quad \wedge \left(\left(x \in \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge y \notin \bigcup_{q \in Obj(\varepsilon)} \{q^I\} \wedge (x, y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \right) \rightarrow y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} \right) \\
& \quad \wedge (\, \exists p, q \in Obj(\varepsilon).\ (x = p^I \wedge y = q^I \wedge \neg R(p, q) \notin \varepsilon \wedge R(p, q) \notin \varepsilon \wedge (x, y) \in R^{I(\iota(\sigma_0.\varepsilon_0))}) \\
& \qquad \rightarrow y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}\,) \Bigg)
\end{aligned}$$

$$\wedge \quad (\exists p, q \in Obj(\varepsilon).\quad (x = p^I \quad \wedge \quad y = q^I \quad \wedge \quad R(p,q) \in \varepsilon) \quad \rightarrow \quad y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))})\Big)\Big)$$

iff $\forall y.(\,(x,y) \in R^{I(\iota(\sigma.\varepsilon))} \rightarrow y \in D^{I(\iota(\sigma.\varepsilon))})$

iff $x \in (\forall R.D)^{I(\iota(\sigma.\varepsilon))}$.

Therefore, we have $((\forall R.D)^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} = (\forall R.D)^{I(\iota(\sigma.\varepsilon))}$.

Case 4. C is of the form $\exists R.D$. The proof is similar with the preceding case.

Case 5. C is of the form $\leq nR.D$. Then, for any individual $x \in \Delta$, we have

$$x \in ((\leq nR.D)^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}$$

iff $x \notin \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge \sharp\{y \mid (x,y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \wedge y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}\} \leq n$

or

$$\exists p \in Obj(\varepsilon).\Big(x = p^I \wedge \exists n_1, n_2, n_3 \geq 0.\Big(\, n_1 + n_2 + n_3 = n$$

$$\wedge \sharp\Big\{y \mid (x,y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \wedge y \notin \bigcup_{q \in Obj(\varepsilon)} \{q^I\} \wedge y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}\Big\} \leq n_1$$

$$\wedge \quad \sharp\Big\{y \quad | \quad (x,y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \quad \wedge \quad y \in \bigcup_{q \in Obj(\varepsilon) \wedge R(p,q) \notin \varepsilon \wedge \neg R(p,q) \notin \varepsilon} \{q^I\} \quad \wedge \quad y \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}\Big\}$$

$$\leq n_2$$

$$\wedge \sharp\{q \mid R(p,q) \in \varepsilon \wedge q^I \in (D^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))}\} \leq n_3\Big)\Big)$$

iff $x \notin \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge \sharp\{y \mid (x,y) \in R^{I(\iota(\sigma.\varepsilon))} \wedge y \in D^{I(\iota(\sigma.\varepsilon))}\} \leq n$

or

$$\exists p \in Obj(\varepsilon).\Big(x = p^I \wedge \exists n_1, n_2, n_3 \geq 0.\Big(\, n_1 + n_2 + n_3 = n$$

$$\wedge \sharp\Big\{y \mid y \notin \bigcup_{q \in Obj(\varepsilon)} \{q^I\} \wedge (x,y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \wedge y \in D^{I(\iota(\sigma.\varepsilon))}\Big\} \leq n_1$$

$$\wedge \quad \sharp\Big\{y \quad | \quad y \in \Big(\bigcup_{q \in Obj(\varepsilon) \wedge R(p,q) \notin \varepsilon \wedge \neg R(p,q) \notin \varepsilon} \{q^I\} \quad \wedge \quad (x,y) \in R^{I(\iota(\sigma_0.\varepsilon_0))} \quad \wedge \quad y \in D^{I(\iota(\sigma.\varepsilon))}\Big)\Big\} \leq n_2$$

$$\wedge \sharp\Big\{y \mid y \in \bigcup_{R(p,q) \in \varepsilon} \{q^I\} \wedge y \in D^{I(\iota(\sigma.\varepsilon))}\Big\} \leq n_3\Big)\Big)$$

iff $x \notin \bigcup_{p \in Obj(\varepsilon)} \{p^I\} \wedge \sharp\{y \mid (x,y) \in R^{I(\iota(\sigma.\varepsilon))} \wedge y \in D^{I(\iota(\sigma.\varepsilon))}\} \leq n$

or

$$x \in \bigcup_{p \in Obj(\varepsilon)} \{p\} \wedge \sharp\{y \mid (x,y) \in R^{I(\iota(\sigma.\varepsilon))} \wedge y \in D^{I(\iota(\sigma.\varepsilon))}\} \leq n$$

iff $x \in (\leq nR.D)^{I(\iota(\sigma.\varepsilon))}$.

Therefore, we have $((\leq nR.D)^{Regress(\varepsilon)})^{I(\iota(\sigma_0.\varepsilon_0))} = (\leq nR.D)^{I(\iota(\sigma.\varepsilon))}$.

Case 6. C is of the form $\geq nR.D$. The proof is similar with the preceding case. □

It should be noted that the proof of Lemma 2 is similar with the proof provided by Liu et al. [25] for their ABox updating algorithm, since the regression operator presented here is just a modification of Liu et al's process for constructing updated concepts.

The second notation introduced for demonstrating the correctness of Algorithm 1 is the satisfiability of branches.

Definition 17 Let $\mathcal{R}$, $\mathcal{T}$, $\mathcal{B}$, and M be an RBox, a TBox, a branch and a model respectively; let ι be a branch-model mapping w.r.t. $\mathcal{T}$, $\mathcal{B}$ and M. Then, $\mathcal{B}$ is called *satisfied by* M and ι w.r.t. $\mathcal{R}$ and $\mathcal{T}$, denoted by $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}$, if and only if $M \models \mathcal{R}$, $M \models \mathcal{T}$ and $(M, \iota(\sigma.\varepsilon)) \models \varphi$ for every $\sigma.\varepsilon : \varphi \in \mathcal{B}$.

If there is a model M and a branch-model mapping ι with $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}$, then we say that the branch $\mathcal{B}$ is *satisfiable w.r.t. both the RBox* $\mathcal{R}$ *and the TBox* $\mathcal{T}$, otherwise we say that $\mathcal{B}$ is *unsatisfiable w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$.

The third notation we will introduce is a partial order "$\preceq$" on prefixes.

Definition 18 Based on the definition of prefixes, the *partial order "*$\preceq$*" on prefixes* is defined inductively as follows:

- $\sigma.\varepsilon \preceq \sigma.\varepsilon$,
- $\sigma.\varepsilon \preceq \sigma; (P, E).(\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$,
- if $\sigma \preceq \sigma'$ and $\sigma' \preceq \sigma''$, then $\sigma \preceq \sigma''$.

With the help of the above notations, we can demonstrate the following properties for the satisfiability-checking algorithm.

Lemma 3 *For any branch* $\mathcal{B}$ *constructed in Algorithm* 1*, if it is contradictory or its initial view* $IV_{\mathcal{B}}$ *is inconsistent w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$*, then it is unsatisfiable w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$.

Proof According to Definition 12 and by the semantics of $DDL(X^{@})$-formulas, it is immediate that every contradictory branch is unsatisfiable.

If $IV_{\mathcal{B}}$ is inconsistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, then the branch $\mathcal{B}_0 := \{\sigma_0.\varepsilon_0 : \varphi \mid \varphi \in IV_{\mathcal{B}}\}$ is unsatisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$. Since $\mathcal{B}_0 \subseteq \mathcal{B}$, the branch $\mathcal{B}$ is unsatisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$. □

Lemma 4 *For each tableau expansion rule applied on some branch* $\mathcal{B}$*,* $\mathcal{B}$ *is satisfiable w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$ *if and only if this rule can be applied to* $\mathcal{B}$ *in such a way that it yields a branch which is satisfiable w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$.

Proof (*The If direction*) The result is immediate since every new generated branch subsumes the branch $\mathcal{B}$.

(*The Only-if direction*) Suppose there is a model $M = (W, T, \Delta, I)$ and a branch-model mapping ι such that $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}$. We demonstrate the result by investigating all the tableau expansion rules.

If a tableau expansion rule listed in Fig. 1 or Fig. 2 is applied on $\mathcal{B}$, then the result is obvious according to the semantics of roles, formulas and actions of $DDL(X^{@})$.

If the $atom_{<>}$-rule is applied on $\mathcal{B}$, then we have $(M, \iota(\sigma.\varepsilon)) \models < (P, E) > \varphi$, and therefore there is a state $w' \in W$ such that $(\iota(\sigma.\varepsilon), w') \in T((P, E))$, $(M, w') \models \varphi$ and $(M, \iota(\sigma.\varepsilon)) \models \phi$ for every $\phi \in P$. There are two cases to be investigated.

- Suppose no prefix $\sigma_i.\varepsilon_i$ occurring in $\mathcal{B}$ with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. Then, let $\sigma'.\varepsilon'$ be the prefix introduced by applying the $atom_{<>}$-rule and let $\mathcal{B}'$ be the resulted branch. Let ι' be a function constructed by extending the function ι with the map $\iota'(\sigma'.\varepsilon') = w'$. Then, based on Corollary 2, we have $(M, \iota') \models_{\mathcal{R},\mathcal{T}} \mathcal{B}'$.
- Suppose there is a prefix $\sigma_i.\varepsilon_i$ occurring in $\mathcal{B}$ with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. Then we have $\iota(\sigma_i.\varepsilon_i) = w'$ according to the definition of branch-model mappings and the semantics of atomic action definitions. Therefore, based on Corollary 2, we have $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}'$ for the resulted branch $\mathcal{B}'$.

If the $atom_{[\,]}$-rule is applied on $\mathcal{B}$, then we have $(M, \iota(\sigma.\varepsilon)) \models [(P, E)]\varphi$, and consequently there is not any state $w' \in W$ with both $(\iota(\sigma.\varepsilon), w') \in T((P, E))$ and $(M, w') \models \neg\varphi$. Let $\sigma_i.\varepsilon_i$ be a prefix with $\varepsilon_i = (\varepsilon \setminus (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. Then, according to the definition of branch-model mappings and the semantics of atomic action definitions, we have either $(M, \iota(\sigma_i.\varepsilon_i)) \models \varphi$ or $(M, \iota(\sigma.\varepsilon)) \models \phi^{\neg}$ for some $\phi \in P$. Therefore, among all the possible expansions there must be a branch $\mathcal{B}'$ with $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}'$.

If the $Back_r$-rule is applied on $\mathcal{B}$, then we have $(M, \iota(\sigma.\varepsilon)) \models \varphi$. Since $\varphi \notin \varepsilon$ and φ is of the form $R(p, q)$ or $\neg R(p, q)$, by Lemma 1 and based on the unique name assumption on individual names, we have $(p^I, q^I) \in R^{I(\iota(\sigma_0.\varepsilon_0))}$ if φ is of the form $R(p, q)$, and $(p^I, q^I) \notin R^{I(\iota(\sigma_0.\varepsilon_0))}$ if φ is of the form $\neg R(p, q)$. Therefore, we have $(M, \iota(\sigma_0.\varepsilon_0)) \models \varphi$ and consequently $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}'$ for the resulted branch $\mathcal{B}'$.

If the $Back_c$- or $Back_{\neg c}$-rule is applied on $\mathcal{B}$, then, by Lemma 2, the result is obvious. □

Lemma 5 *For any completed branch $\mathcal{B}$, if it is neither contradictory nor ignorable, and its initial view $IV_{\mathcal{B}}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, then $\mathcal{B}$ is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$.*

Proof Since $IV_{\mathcal{B}}$ is an ABox of the description logic $X^{@}$ and is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, there must be an interpretation $I_{IV_{\mathcal{B}}} = (\Delta^{I_{IV_{\mathcal{B}}}}, \cdot^{I_{IV_{\mathcal{B}}}})$ of $X^{@}$ such that $I_{IV_{\mathcal{B}}} \models \mathcal{R}$, $I_{IV_{\mathcal{B}}} \models \mathcal{T}$ and $I_{IV_{\mathcal{B}}} \models IV_{\mathcal{B}}$.

Let $\Sigma^{\mathcal{B}}$ be the set of all the prefixes occurring in $\mathcal{B}$; let N_C, N_R and N_I respectively be the sets of all the concept names, all the role names and all the individual names occurring in $\mathcal{B}$, $\mathcal{R}$ and $\mathcal{T}$. Furthermore, let $N_C = N^{\mathcal{T}}_{C_P} \cup N^{\mathcal{T}}_{C_D}$, where $N^{\mathcal{T}}_{C_P}$ is the set of primitive concept names w.r.t. $\mathcal{T}$, and $N^{\mathcal{T}}_{C_D}$ is the set of defined concept names. Construct a $DDL(X^{@})$-model $M = (W, T, \Delta, I)$ and a function $\iota : \Sigma^{\mathcal{B}} \to W$ according to the following steps.

1. Construct an interpretation $I_0 = (\Delta^{I_0}, \cdot^{I_0})$ according to the following steps:
   (a) set $\Delta^{I_0} := \Delta^{I_{IV_{\mathcal{B}}}}$;
   (b) for each individual name $p \in N_I$, if it is interpreted in $I_{IV_{\mathcal{B}}}$, then set $p^{I_0} := p^{I_{IV_{\mathcal{B}}}}$, otherwise introduce an individual $\tilde{\mathrm{p}}$, add $\tilde{\mathrm{p}}$ to the set Δ^{I_0}, and set $p^{I_0} := \tilde{\mathrm{p}}$;
   (c) for each concept name $C \in N_C$, if it is interpreted in $I_{IV_{\mathcal{B}}}$, then set $C^{I_0} := C^{I_{IV_{\mathcal{B}}}}$, otherwise set $C^{I_0} := \emptyset$;

(d) for each role name $R \in N_R$, if it is interpreted in $I_{IV_{\mathcal{B}}}$, then set $R^{I_0} := R^{I_{IV_{\mathcal{B}}}}$, otherwise set $R^{I_0} := \emptyset$.

2. Set the domain $\Delta := \Delta^{I_0}$.
3. Set $p^I := p^{I_0}$ for each individual name $p \in N_I$.
4. Introduce a state w_0 and construct an interpretation $I(w_0) = (\Delta, \cdot^{I(w_0)})$ as follows:

   – $R^{I(w_0)} := R^{I_0}$ for each role name $R \in N_R$, and
   – $C^{I(w_0)} := C^{I_0}$ for each concept name $C \in N_C$.

   Furthermore, set $\iota(\sigma_0.\varepsilon_0) := w_0$.
5. For each non-initial prefix $\sigma_i.\varepsilon_i \in \Sigma^{\mathcal{B}}$, introduce a state w_i and construct an interpretation $I(w_i) = (\Delta, \cdot^{I(w_i)})$ as follows:

   – $R^{I(w_i)} := (R^{I(w_0)} \cup \{ (p^I, q^I) \mid R(p,q) \in \varepsilon_i \}) \setminus \{ (p^I, q^I) \mid \neg R(p,q) \in \varepsilon_i \}$ for each role name $R \in N_R$.
   – $A^{I(w_i)} := (A^{I(w_0)} \cup \{ p^I \mid A(p) \in \varepsilon_i \}) \setminus \{ p^I \mid (\neg A)(p) \in \varepsilon_i \}$ for each primitive concept name $A \in N^{\mathcal{T}}_{C_P}$, and
   – $A^{I(w_i)} := C^{I(w_i)}$ for each concept name $A \in N^{\mathcal{T}}_{C_D}$ if A is defined by some concept definition $A \equiv C \in \mathcal{T}$.

   Furthermore, set $\iota(\sigma_i.\varepsilon_i) := w_i$.

Firstly, by Lemma 1, it is obvious that ι is a branch-model mapping w.r.t. $\mathcal{T}$, $\mathcal{B}$ and M.

Secondly, according to the above construction, it is obvious that $A^{I(w)} = C^{I(w)}$ for any concept definition $A \equiv C \in \mathcal{T}$ and any state $w \in W$. Therefore, we have $M \models \mathcal{T}$.

Thirdly, we demonstrate that $M \models \mathcal{R}$. Let $R \sqsubseteq R'$ be any role inclusion contained in $\mathcal{R}$; let $\sigma.\varepsilon$ be any prefix contained in $\Sigma^{\mathcal{B}}$; and let p, q be any individual names with $(p^I, q^I) \in R^{I(\iota(\sigma.\varepsilon))}$. Then, by Lemma 1 and based on the unique name assumption on individual names, we have $\neg R(p,q) \notin \varepsilon$, and either $(p^I, q^I) \in R^{I(\iota(\sigma_0.\varepsilon_0))}$ or $R(p,q) \in \varepsilon$. Therefore, according to the construction of prefixes and based on the fact that $I_{IV_{\mathcal{B}}} \models \mathcal{R}$, we have $\neg R'(p,q) \notin \varepsilon$, and either $(p^I, q^I) \in R'^{I(\iota(\sigma_0.\varepsilon_0))}$ or $R'(p,q) \in \varepsilon$. So, it must be $(p^I, q^I) \in R'^{I(\iota(\sigma.\varepsilon))}$.

Finally, we demonstrate that $(M, \iota(\sigma.\varepsilon)) \models \varphi$ for any prefixed formula $\sigma.\varepsilon : \varphi \in \mathcal{B}$. The proof is by induction on the structure of φ.

(*Base case*) φ is an ABox assertion. There are six cases to be investigated.

– $\sigma.\varepsilon$ is the initial prefix $\sigma_0.\varepsilon_0$. Then we have $\varphi \in IV_{\mathcal{B}}$ and consequently $I_{IV_{\mathcal{B}}} \models \varphi$. Therefore, according to the construction of M and ι, we have $(M, \iota(\sigma.\varepsilon)) \models \varphi$.
– $\sigma.\varepsilon$ is a non-initial prefix, and $\varphi \in \varepsilon$. Then, by Corollary 2, we have $(M, \iota(\sigma.\varepsilon)) \models \varphi$.
– $\sigma.\varepsilon$ is a non-initial prefix, $\varphi \notin \varepsilon$, and φ is of the form $R(p,q)$. Then, we have $\sigma_0.\varepsilon_0 : R(p,q) \in \mathcal{B}$ according to the $Back_r$-rule; so, we have $R(p,q) \in IV_{\mathcal{B}}$ and consequently $(p^I, q^I) \in R^{I(\iota(\sigma_0.\varepsilon_0))}$. At the same time, it must be $\neg R(p,q) \notin \varepsilon$; otherwise, according to the $atom_{<>}$-rule by which the prefix $\sigma.\varepsilon$ is introduced, we will get $\sigma.\varepsilon : \neg R(p,q) \in \mathcal{B}$ and make $\mathcal{B}$ a contradictory branch. Moreover, by Lemma 1 and based on the unique name assumption on individual names, we have $R^{I(\iota(\sigma.\varepsilon))} = (R^{I(\iota(\sigma_0.\varepsilon_0))} \cup \{ (p^I, q^I) \mid R(p,q) \in \varepsilon \}) \setminus \{ (p^I, q^I) \mid \neg R(p,q) \in \varepsilon \}$, no matter R is a role name or an inverse role. Therefore, we have $(M, \iota(\sigma.\varepsilon)) \models R(p,q)$.

- $\sigma.\varepsilon$ is a non-initial prefix, $\varphi \notin \varepsilon$, and φ is of the form $\neg R(p, q)$. Then the result can be demonstrated with a similar process of the preceding case.
- $\sigma.\varepsilon$ is a non-initial prefix, $\varphi \notin \varepsilon$, and φ is of the form $C(p)$. Then, we have $\sigma_0.\varepsilon_0 : C^{Regress(\varepsilon,\mathcal{T})}(p) \in \mathcal{B}$ according to the $Back_c$-rule; so, we have $C^{Regress(\varepsilon,\mathcal{T})}(p) \in IV_{\mathcal{B}}$ and consequently $(M, \iota(\sigma_0.\varepsilon_0)) \models C^{Regress(\varepsilon,\mathcal{T})}(p)$. By Lemma 2, we have $(M, \iota(\sigma.\varepsilon)) \models C(p)$.
- $\sigma.\varepsilon$ is a non-initial prefix, $\varphi \notin \varepsilon$, and φ is of the form $\neg C(p)$. Then the result can be demonstrated with a similar process of the preceding case.

(*Inductive step*) Since φ is a formula in negation normal form, we only need to investigate the following cases.

Case 1. φ is of the form $\phi \wedge \psi$ or $\phi \vee \psi$. Then, since the branch is completed, the result is straightforward according to the $\wedge$-rule, the $\vee$-rule and the inductive hypothesis.

Case 2. φ is of the form $< \pi > \phi$. By induction on the structure of π, we demonstrate that there is a state $w' \in W$ with both $(\iota(\sigma.\varepsilon), w') \in T(\pi)$ and $(M, w') \models \phi$.

- π is an atomic action (P, E). Then, according to the $atom_{<>}$-rule, we have $\{\sigma.\varepsilon : \psi_i \mid \psi_i \in P\} \subseteq \mathcal{B}$, and there must be a prefixed formula $\sigma_i.\varepsilon_i : \phi \in \mathcal{B}$ with $\varepsilon_i = (\varepsilon \backslash (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. By the inductive hypothesis, we have $(M, \iota(\sigma.\varepsilon)) \models P$ and $(M, \iota(\sigma_i.\varepsilon_i)) \models \phi$. Therefore, according to the construction of ι and the semantics of atomic action definitions, we have $(\iota(\sigma.\varepsilon), \iota(\sigma_i.\varepsilon_i)) \in T((P, E))$.
- π is of the form $\pi_1; \pi_2$, $\psi?$, or $\pi_1 \cup \pi_2$. Then the result is straightforward according to the inductive hypothesis and the $;_{<>}$-, $?_{<>}$- and $\cup_{<>}$-rules.
- π is of the form π_1^*. Then, according to the $*_{<>}$-rule, there must be a character string X with both $X \doteq < \pi_1^* > \phi \in \mathcal{B}$ and $\sigma.\varepsilon : X \in \mathcal{B}$. Let $path(X, \mathcal{B})$ be the set $\{\sigma_i.\varepsilon_i \mid \sigma_i.\varepsilon_i : X \in \mathcal{B}\}$. It is immediate that $path(X, \mathcal{B})$ is a finite set totally ordered by the partial order "$\preceq$", and we have $(\iota(\sigma.\varepsilon), \iota(\sigma_i.\varepsilon_i)) \in T(\pi_1^*)$ for every $\sigma_i.\varepsilon_i \in path(X, \mathcal{B})$. At the same time, since $\mathcal{B}$ is not ignorable, there must be a prefix $\sigma_m.\varepsilon_m \in path(X, \mathcal{B})$ with $\sigma_m.\varepsilon_m : \phi \in \mathcal{B}$, and consequently $(M, \iota(\sigma_m.\varepsilon_m)) \models \phi$ by the inductive hypothesis.

Case 3. φ is of the form $[\pi]\phi$. By induction on the structure of π, we demonstrate that no state $w' \in W$ exists with both $(\iota(\sigma.\varepsilon), w') \in T(\pi)$ and $(M, w') \models \neg\phi$.

- π is an atomic action (P, E). Then, according to the $atom_{[\,]}$-rule, we have either $\{\sigma.\varepsilon : \psi^{\neg} \mid \psi \in P\} \cap \mathcal{B} \neq \emptyset$, or $\sigma_i.\varepsilon_i : \phi \in \mathcal{B}$ for any prefix $\sigma_i.\varepsilon_i$ with $\varepsilon_i = (\varepsilon \backslash (E^*_{\mathcal{R}})^{\neg}) \cup E^*_{\mathcal{R}}$. Therefore, by the inductive hypothesis, the construction of M and the semantics of atomic action definitions, no state $w' \in W$ exists with both $(\iota(\sigma.\varepsilon), w') \in T((P, E))$ and $(M, w') \models \neg\phi$.
- π is of the form $\pi_1; \pi_2$, $\psi?$, or $\pi_1 \cup \pi_2$. Then the result is straightforward according to the inductive hypothesis and the $;_{[\,]}$-, $?_{[\,]}$- and $\cup_{[\,]}$-rules.
- π is of the form π_1^*. Then, according to the $*_{[\,]}$-rule, we have $\sigma.\varepsilon : \phi \in \mathcal{B}$ and $\sigma.\varepsilon : [\pi_1][\pi_1^*]\phi \in \mathcal{B}$. For any positive integer n and any prefix $\sigma'.\varepsilon'$ with $(\iota(\sigma.\varepsilon), \iota(\sigma'.\varepsilon')) \in (T(\pi_1))^n$, by double induction on n and on the structure of π_1, we have $\sigma'.\varepsilon' : [\pi_1^*]\phi \in \mathcal{B}$, $\sigma'.\varepsilon' : [\pi_1][\pi_1^*]\phi \in \mathcal{B}$, $\sigma'.\varepsilon' : \phi \in \mathcal{B}$, and consequently $(M, \iota(\sigma'.\varepsilon')) \models \phi$ by the inductive hypothesis. Therefore, according to the construction of M, there is not any state $w' \in W$ with both $(\iota(\sigma.\varepsilon), w') \in T(\pi_1^*)$ and $(M, w') \models \neg\phi$.

To sum up, we have $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}$, and $\mathcal{B}$ is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$. □

Lemma 6 *If the branch $\mathcal{B}_{init} = \{\sigma_0.\varepsilon_0 : nf(\phi)\}$ constructed in the first step of Algorithm 1 is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$, then, among all the branches generated by applying tableau expansion rules, there must be a completed branch which is not ignorable and is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$.*

Proof Suppose the contrary: $\mathcal{B}_{init}$ is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$, while every completed and satisfiable branch generated by applying tableau expansion rules is ignorable. It is worth noting that each branch can be ignorable due to a different unfulfilled eventuality record; however, without loss of generality, here we just investigate one of these unfulfilled eventuality record.

Let $\mathcal{B}'$ be a completed branch which is ignorable due to some eventuality record $X \doteq < \pi^* > \varphi$; let $\mathcal{B}'$ be satisfied by some model $M = (W, T, \Delta, I)$ and some branch-model mapping ι w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

Let $path(X, \mathcal{B}') = \{\sigma.\varepsilon \mid \sigma.\varepsilon : X \in \mathcal{B}'\}$. It is immediate that $path(X, \mathcal{B}')$ is a finite set totally ordered by the relation "$\preceq$". Let all the elements of $path(X, \mathcal{B}')$ be $\sigma_1.\varepsilon_1$, $\ldots$, $\sigma_m.\varepsilon_m$, with $\sigma_i.\varepsilon_i \preceq \sigma_j.\varepsilon_j$ for every $1 \leq i \leq j \leq m$. Then it is immediate that $\sigma_i.\varepsilon_i : \neg\varphi \in \mathcal{B}'$ for every $1 \leq i \leq m$; furthermore, for every $2 \leq j \leq m$, the prefix $\sigma_j.\varepsilon_j$ is introduced by reducing the prefixed formula $\sigma_{j-1}.\varepsilon_{j-1} : X$ according to the X-rule as well as the $;_{<>}$-, $?_{<>}$-, $\cup_{<>}$- and $atom_{<>}$-rules.

Let $\sigma_M.\varepsilon_M$ be the maximum prefix among all the prefixes preceded by $\sigma_m.\varepsilon_m$; i.e., let $\sigma.\varepsilon \preceq \sigma_M.\varepsilon_M$ for every prefix $\sigma.\varepsilon$ with $\sigma_m.\varepsilon_m \preceq \sigma.\varepsilon$. Furthermore, without loss of generality, let σ_M be of the form $\sigma_m; (P_1, E_1); \ldots; (P_k, E_k)$. Then, since $\mathcal{B}'$ is completed, the $atom_{<>}$-rule guarantees the existence of some prefixed formula $\sigma_M.\varepsilon_M : < (P_{k+1}, E_{k+1}) >< \pi' > X$ in $\mathcal{B}'$ with $T((P_1, E_1); \ldots; (P_k, E_k); (P_{k+1}, E_{k+1}); \pi') = T(\pi)$. At the same time, there must be some prefix $\sigma_n.\varepsilon_n \in path(X, \mathcal{B}')$ and some prefix $\sigma_N.\varepsilon_N$ occurring in $\mathcal{B}'$ such that $\sigma_n.\varepsilon_n \preceq \sigma_N.\varepsilon_N \preceq \sigma_m.\varepsilon_m$, $\varepsilon_N = (\varepsilon_M \setminus (E_{k+1}{}^*_{\mathcal{R}})^\neg) \cup E_{k+1}{}^*_{\mathcal{R}}$, and $\sigma_N.\varepsilon_N :< \pi' > X \in \mathcal{B}'$.

Since $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}'$ and $\sigma_n.\varepsilon_n : X \in \mathcal{B}'$, there must be an integer u as well as $u + 1$ states $w_n, w_{n+1}, \ldots, w_{n+u} \in W$ such that $\iota(\sigma_n.\varepsilon_n) = w_n$, $(M, w_{n+u}) \models \varphi$, and $(w_{n+i}, w_{n+1+i}) \in T(\pi)$ for every $0 \leq i < u$.

Let's first assume that $u \leq (m - n)$. Since $\sigma_M.\varepsilon_M : < (P_{k+1}, E_{k+1}) >< \pi' > X \in \mathcal{B}'$, we can remap the prefixes $\sigma_n.\varepsilon_n, \ldots, \sigma_m.\varepsilon_m$ so that $\iota(\sigma_{n+i}.\varepsilon_{n+i}) = w_{n+i}$ for every $0 \leq i \leq m - n$. Therefore, since $(M, w_{n+u}) \models \varphi$, we have $(M, \iota(\sigma_{n+u}.\varepsilon_{n+u})) \models \varphi$. At the same time, since $\sigma_{n+u}.\varepsilon_{n+u} : \neg\varphi \in \mathcal{B}'$ and $(M, \iota) \models_{\mathcal{R}} \mathcal{B}'$, we have $(M, \iota(\sigma_{n+u}.\varepsilon_{n+u})) \models \neg\varphi$. Hence we get a contradiction.

Assume that $u > (m - n)$. Since $\sigma_M.\varepsilon_M : < (P_{k+1}, E_{k+1}) >< \pi' > X \in \mathcal{B}'$, there must be one (P_{k+1}, E_{k+1})-step from the state $\iota(\sigma_M.\varepsilon_M)$ to the state $\iota(\sigma_N.\varepsilon_N)$, and one π'-step followed by $u - (m - n) - 2$ π-steps from the state $\iota(\sigma_N.\varepsilon_N)$ to the state w_{n+u}. Construct a new mapping $\jmath$ as follows: for every prefix $\sigma.\varepsilon$ occurring in $\mathcal{B}'$, if $\sigma.\varepsilon \preceq \sigma_N.\varepsilon_N$ or $\sigma.\varepsilon$ is unrelated with $\sigma_N.\varepsilon_N$, then map this prefix in the same way as ι does. By Lemma 4, there must be a completed branch $\mathcal{B}''$ which is satisfied by M and $\jmath$; furthermore, in this branch, the state $\jmath(\sigma_n.\varepsilon_n)$ fulfills the eventuality X in $u - (m - n) - 1$ π-steps. Now, since $u - (m - n) - 1 \leq u$, we can repeat the above process until reach an $u' \leq (m - n)$, getting a contradiction again. □

We are now ready to demonstrate the correctness of the satisfiability-checking algorithm.

Theorem 8 *Algorithm* 1 *returns "TRUE" if and only if* ϕ *is satisfiable w.r.t.* $\mathcal{R}$, $\mathcal{T}$ *and* $\mathcal{A}_\mathcal{C}$.

Proof Let $\mathcal{B}_{init} = \{\sigma_0.\varepsilon_0 : nf(\phi)\}$ be the branch constructed in the first step of Algorithm 1.

(*The If direction*) If ϕ is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_\mathcal{C}$, then there is a model $M = (W, T, \Delta, I)$ and a state $w \in W$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_\mathcal{C}$ and $(M, w) \models \phi$. Furthermore, according to the construction of $nf(\phi)$, we have $(M, w) \models nf(\phi)$.

Construct a function ι as $\iota(\sigma_0.\varepsilon_0) = w$. It is immediate that ι is a branch-model mapping w.r.t. $\mathcal{T}$, $\mathcal{B}_{init}$ and M, and we have $(M, \iota) \models_{\mathcal{R},\mathcal{T}} \mathcal{B}_{init}$. Therefore, $\mathcal{B}_{init}$ is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$. By Lemma 6, a completed branch $\mathcal{B}'$ which is not ignorable and is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$ will be generated by applying tableau expansion rules. Furthermore, by Lemma 3, the branch $\mathcal{B}'$ is not contradictory, and the initial view $IV'_\mathcal{B}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$. Therefore, the algorithm will return "TRUE".

(*The Only-if direction*) If the algorithm returns "TRUE", then there must be a completed branch $\mathcal{B}'$ which is neither contradictory nor ignorable, and its initial view $IV_{\mathcal{B}'}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$. By Lemma 5, $\mathcal{B}'$ is satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$, and consequently the branch $\mathcal{B}_{init}$ is also satisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$ by Lemma 4. So, there must be a model $M = (W, T, \Delta, I)$ and a branch-model mapping ι such that $M \models \mathcal{R}$, $M \models \mathcal{T}$ and $(M, \iota(\sigma_0.\varepsilon_0)) \models nf(\phi)$. Since there is not any action name occurring in $nf(\phi)$, we can modify the model M as follows: for every action name α defined by some atomic action definition $\alpha \equiv (P, E) \in \mathcal{A}_\mathcal{C}$, set

$$
\begin{aligned}
T(\alpha) := \{\ & (w, w') \in W \times W |\ (M, w) \models P, \\
& A^{I(w')} = (A^{I(w)} \cup \{p^I | A(p) \in E^*_\mathcal{R}\}) \setminus \{p^I | (\neg A)(p) \in E^*_\mathcal{R}\} \\
& \textit{for each concept name } A \textit{ which is primitive w.r.t. } \mathcal{T}, \textit{ and} \\
& R^{I(w')} = (R^{I(w)} \cup \{(p^I, q^I) | R(p, q) \in E^*_\mathcal{R}\}) \setminus \{(p^I, q^I) | \neg R(p, q) \in E^*_\mathcal{R}\} \\
& \textit{for each role name } R.\ \}.
\end{aligned}
$$

Let M' be the resulted model. Then it is immediate that $M' \models \mathcal{R}$, $M' \models \mathcal{T}$, $M' \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_\mathcal{C}$, and $(M', \iota(\sigma_0.\varepsilon_0)) \models \phi$. Therefore, ϕ is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_\mathcal{C}$. □

6 Extend Atomic Action Definitions of $DDL(X^@)$

In order to be compatible with atomic actions described by Baader et al.'s formalism [4], we extend atomic action definitions of $DDL(X^@)$ to include occlusions and conditional post-conditions.

To be distinguished from atomic action definitions discussed in previous sections, we call those extended by occlusions and conditional post-conditions as extended atomic action definitions.

With respect to a TBox $\mathcal{T}$, an *extended atomic action definition* of $DDL(X^{@})$ is of the form $\alpha \equiv (P, O, E)$, where

- $\alpha \in N_A$;
- P is a finite set of ABox assertions for describing the pre-conditions;
- O is a finite set of occlusions, where each occlusion is of the form $A(p)$ or $R(p, q)$, with A a primitive concept name, R a role name, and $p, q \in N_I$; and
- E is a finite set of conditional post-conditions, where each conditional post-condition is of the form φ/ψ with φ an ABox assertion and ψ a primitive literal.

In the above definition, the pre-conditions, occlusions and conditional post-conditions are the same with those introduced by Baader et al. [4] for describing atomic actions. The pre-conditions specify under which conditions the action is applicable. Each conditional post-condition φ/ψ says that, if φ is true before executing the action, then ψ should be true after the execution. The occlusions indicate those primitive literals that can change arbitrarily as while as the action is executed. With $DDL(X^{@})$-models, the semantics of extended atomic action definitions is strictly defined as follows.

With respect to an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$, a model $M = (W, T, \Delta, I)$ *satisfies* an extended atomic action definition $\alpha \equiv (P, O, E)$, in symbols $M \models_{\mathcal{R},\mathcal{T}} \alpha \equiv (P, O, E)$, if and only if $M \models \mathcal{R}$, $M \models \mathcal{T}$, and

$$
\begin{aligned}
T(\alpha) = \{\ (w, w') \in W \times W \mid\ & (M, w) \models P, \\
& \textit{both } A^{+}_{w} \cap A^{-}_{w} = \emptyset \textit{ and } A^{I(w')} \cap I^{w}_{A} \ = \ ((A^{I(w)} \cup A^{+}_{w}) \setminus A^{-}_{w}) \cap I^{w}_{A} \\
& \textit{for each concept name } A \textit{ which is primitive w.r.t. } \mathcal{T}, \textit{ and} \\
& \textit{both } R^{+}_{w} \cap R^{-}_{w} = \emptyset \textit{ and } R^{I(w')} \cap I^{w}_{R} \ = \ ((R^{I(w)} \cup R^{+}_{w}) \setminus R^{-}_{w}) \cap I^{w}_{R} \\
& \textit{for each role name } R.\ \},
\end{aligned}
$$

where, let $E_w := \{\ \psi \mid \varphi/\psi \in E \text{ and } (M, w) \models \varphi\ \}$, then A^{+}_{w}, A^{-}_{w}, I^{w}_{A}, R^{+}_{w}, R^{-}_{w} and I^{w}_{R} are some sets constructed as follows:

- $A^{+}_{w} := \{\ p^I \mid A(p) \in E_w\ \}$,
- $A^{-}_{w} := \{\ p^I \mid \neg A(p) \in E_w\ \}$,
- $I^{w}_{A} := (\ \Delta \setminus \{\ p^I \mid A(p) \in O\ \}\ ) \cup A^{+}_{w} \cup A^{-}_{w}$,
- $R^{+}_{w} := \{\ (p^I, q^I) \mid R(p, q) \in E^{*}_{w\mathcal{R}}\ \}$,
- $R^{-}_{w} := \{\ (p^I, q^I) \mid \neg R(p, q) \in E^{*}_{w\mathcal{R}}\ \}$,
- $I^{w}_{R} := (\ (\Delta \times \Delta) \setminus \{\ (p^I, q^I) \mid R(p, q) \in O\ \}\ ) \cup R^{+}_{w} \cup R^{-}_{w}$.

The above definition is compatible with the semantics of atomic action definitions. According to this definition, for any pair $(w, w') \in T(\alpha)$, any primitive concept name A and any role name R, the interpretations $A^{I(w)}$ and $A^{I(w')}$ should satisfy that $A^{+}_{w} \subseteq A^{I(w')}$, $A^{-}_{w} \cap A^{I(w')} = \emptyset$, and except these contained in $\{p^I \mid A(p) \in O\}$ might change arbitrarily, nothing else changes from $A^{I(w)}$ to $A^{I(w')}$; the interpretations $R^{I(w)}$ and $R^{I(w')}$ should satisfy that $R^{+}_{w} \subseteq R^{I(w')}$, $R^{-}_{w} \cap R^{I(w')} = \emptyset$, and except these contained in $\{(p^I, q^I) \mid R(p, q) \in O\}$ might change arbitrarily, nothing else changes from $R^{I(w)}$ to $R^{I(w')}$.

According to the definitions, the semantics of extended atomic action definitions are similar to the semantics of atomic actions defined in Baader et al.'s formalism [4], except that different DL-interpretations which are connected in [4] by the

interpretations of atomic actions are compressed here into a single $DDL(X^{@})$-model, and the condition $(M, w) \models P$ is introduced here to state explicitly that the pre-conditions should be satisfied.

As a result, atomic actions described by Baader et al.'s formalism can now be represented in $DDL(X^{@})$ with extended atomic action definitions. For example, the atomic action $buyBook_{a,b}$ presented in Section 1 of this paper can also be represented as an extended atomic action definition. As another example, for the Web service system discussed in Section 3.3, some Web service $BuyBookNotified_{Tom,Kin}$ might be described by the following extended atomic action definition:

$$
\begin{aligned}
&BuyBookNotified_{Tom,Kin}\\
&\quad \equiv \ (\{\, customer(Tom), book(KingLear)\,\}, \quad \{\,\},\\
&\qquad \{\, instore(KingLear)/bought(Tom, KingLear),\\
&\qquad\ \ instore(KingLear)/\neg instore(KingLear),\\
&\qquad\ \ instore(KingLear)/notify(Tom, NotifyOrderSucceed),\\
&\qquad\ \ \neg instore(KingLear)/notify(Tom, NotifyBookOutOfStock)\,\}\,)
\end{aligned}
$$

where *notify* is a new introduced role name, and both *NotifyOrderSucceed* and *NotifyBookOutOfStock* are new introduced individual names. According to this description, if the book *KingLear* is in store before executing the action, then the formulas $bought(Tom, KingLear)$, $\neg instore(KingLear)$ and $notify(Tom, NotifyOrderSucceed)$ will be true after the execution; otherwise, the formula $notify(Tom, NotifyBookOutOfStock)$ will be true after the execution, which means that Tom is notified that the book is out of stock.

In $DDL(X^{@})$, for each atomic action α defined by some extended atomic action definition $\alpha \equiv (P, O, E)$, we will introduce a procedure $Unfold(\alpha)$ to unfold it into some choice action $\alpha_1 \cup ... \cup \alpha_n$, where

- each α_i $(1 \leq i \leq n)$ is an atomic action defined by some atomic action definition; and
- let $\mathcal{A}_C$ be an ActBox in which every α_i $(1 \leq i \leq n)$ is defined, then it must be $T(\alpha) = T(\alpha_1 \cup ... \cup \alpha_n)$ for any model $M = (W, T, \Delta, I)$ with $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \alpha \equiv (P, O, E)$ and $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$.

Based on such a procedure, reasoning mechanisms presented in previous sections can be easily extended to support extended atomic action definitions.

First of all, we present the procedure $Unfold(\,)$ and demonstrate some properties for it.

Let α be an atomic action defined by some extended atomic action definition $\alpha \equiv (P, O, E)$ w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$; let $O = \{\, \phi_1, ..., \phi_m \,\}$ and $E = \{\, \varphi_1/\psi_1, ..., \varphi_k/\psi_k \,\}$. Then, the procedure $Unfold(\alpha)$ operates according to the following steps:

1. Construct two (initially empty) sets $\mathcal{A}_C'$ and $\mathcal{A}_C''$ of atomic action definitions.
2. Construct an ABox $P_{cond} := \{\, \varphi \mid \varphi/\psi \in E \,\}$.
3. For each set $\mathcal{A}_i \subseteq P_{cond} \cup \{\varphi^{\neg} \mid \varphi \in P_{cond}\}$, if either $\varphi \in \mathcal{A}_i$ or $\varphi^{\neg} \in \mathcal{A}_i$ for every $\varphi \in P_{cond}$, and the ABox $\mathcal{A}_i$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, then:
   (a) construct an atomic action definition $\beta_i \equiv (P \cup \mathcal{A}_i, \{\psi \mid \varphi/\psi \in E \text{ and } \varphi \in \mathcal{A}_i\})$, and

(b) put it into $\mathcal{A}_C'$ if it is PE-consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

4. For each atomic action definition $\beta_i \equiv (P_i, E_i) \in \mathcal{A}_C'$, do the following operations sequentially:
   (a) construct an ABox $O_i := \{ \phi \mid \phi \in O, \phi \notin E_i{}^*_{\mathcal{R}}$ and $\phi^\neg \notin E_i{}^*_{\mathcal{R}} \}$;
   (b) for each set $O_{i,j} \subseteq O_i \cup \{\phi^\neg \mid \phi \in O_i\}$, if either $\phi \in O_{i,j}$ or $\phi^\neg \in O_{i,j}$ for every $\phi \in O_i$, and the ABox $O_{i,j}$ is consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$, then :
      i. construct an atomic action definition $\beta_{i,j} \equiv (P_i, E_i \cup O_{i,j})$, and
      ii. put it into $\mathcal{A}_C''$ if it is PE-consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$.
5. If the set $\mathcal{A}_C''$ is empty, then construct an atomic action definition $\beta_0 \equiv (\{false\}, \emptyset)$ and put it into $\mathcal{A}_C''$.
6. Let $\alpha_1, ..., \alpha_n$ be all the atomic actions defined in $\mathcal{A}_C''$; construct a choice action $\alpha_1 \cup ... \cup \alpha_n$ and return it.

As an example, taking the atomic action $BuyBookNotified_{Tom,Kin}$ defined above as input, the procedure $Unfold(BuyBookNotified_{Tom,Kin})$ will return a choice action of the form

$$BuyBookSucceed_{Tom,Kin} \cup BuyBookFailed_{Tom,Kin}$$

where $BuyBookSucceed_{Tom,Kin}$ and $BuyBookFailed_{Tom,Kin}$ are two atomic actions defined by the following atomic action definitions:

$$\begin{aligned}
&BuyBookSucceed_{Tom,Kin}\\
&\quad \equiv \ (\ \{\ customer(Tom), book(KingLear), instore(KingLear)\ \},\\
&\qquad \{\ bought(Tom, KingLear), \neg instore(KingLear),\\
&\qquad\quad notify(Tom, NotifyOrderSucceed)\ \}\ )\\
&\qquad BuyBookFailed_{Tom,Kin}\\
&\quad \equiv \ (\ \{\ customer(Tom), book(KingLear), \neg instore(KingLear)\ \},\\
&\qquad \{\ notify(Tom, NotifyBookOutOfStock)\ \}\ )
\end{aligned}$$

Lemma 7 *For any atomic action α which is defined by some extended atomic action definition $\alpha \equiv (P, O, E)$ w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$, the procedure $Unfold(\alpha)$ terminates.*

Proof Let $m := \sharp O$ and $k := \sharp E$. Then, during the execution of the procedure $Unfold(\alpha)$, we have $\sharp P_{cond} \leq k$, and consequently the number of atomic action definitions put into $\mathcal{A}_C'$ is bounded by 2^k. At the same time, for each atomic action definition $\alpha_i \equiv (P_i, E_i) \in \mathcal{A}_C'$, we have $\sharp O_i \leq m$, and therefore the number of atomic action definitions constructed for it is bounded by 2^m. To sum up, the number of atomic action definitions which will be generated and put into $\mathcal{A}_C''$ is bounded by $2^k \times 2^m$. So, the procedure $Unfold(\alpha)$ terminates. □

Lemma 8 *Let α be an atomic action defined by some extended atomic action definition $\alpha \equiv (P, O, E)$ w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$; let $\alpha_1 \cup ... \cup \alpha_n$ be the action returned by the procedure $Unfold(\alpha)$; and let $\mathcal{A}_C$ be an ActBox in which each α_i $(1 \leq i \leq n)$ is defined by some atomic action definition. Then, for any model $M = (W, T, \Delta, I)$ with $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_C$ and $M \models_{\mathcal{R},\mathcal{T}} \alpha \equiv (P, O, E)$, it must be $T(\alpha_1 \cup ... \cup \alpha_n) = T(\alpha)$.*

Proof For any states $w, w' \in W$, we demonstrate that $(w, w') \in T(\alpha_1 \cup ... \cup \alpha_n)$ if and only if $(w, w') \in T(\alpha)$.

(*The If direction*) Let $(w, w') \in T(\alpha)$. By the semantics of extended atomic action definitions, we have $(M, w) \models P$, both $A_w^+ \cap A_w^- = \emptyset$ and $A^{I(w')} \cap I_A^w = ((A^{I(w)} \cup A_w^+) \backslash A_w^-) \cap I_A^w$ for each concept name A which is primitive w.r.t. $\mathcal{T}$, and both $R_w^+ \cap R_w^- = \emptyset$ and $R^{I(w')} \cap I_R^w = ((R^{I(w)} \cup R_w^+) \backslash R_w^-) \cap I_R^w$ for each role name R, where, let $E_w := \{\psi \mid \varphi/\psi \in E \text{ and } (M, w) \models \varphi\}$, then the sets A_w^+, A_w^-, I_A^w, R_w^+, R_w^- and I_R^w are constructed as those listed in the semantic definition of extended atomic action definitions.

Construct three ABoxes P_w, O' and O_w as follows:

$$P_w := \{ \varphi \mid \varphi/\psi \in E \text{ and } (M, w) \models \varphi \} \cup \{ \varphi^\neg \mid \varphi/\psi \in E \text{ and } (M, w) \models \neg\varphi \};$$

$$O' := \{ \phi \mid \phi \in O, \phi \notin E_w \overset{*}{\mathcal{R}} \text{ and } \phi^\neg \notin E_w \overset{*}{\mathcal{R}} \}.$$

$$O_w := \{ \phi \mid \phi \in O' \text{ and } (M, w') \models \phi \} \cup \{ \phi^\neg \mid \phi \in O' \text{ and } (M, w') \models \neg\phi \}.$$

Then, according to the operations of the procedure $Unfold(\alpha)$, there must be some atomic action α_i $(1 \leq i \leq n)$ such that $\mathbf{Pre}_{\alpha_i} = P \cup P_w$ and $\mathbf{Eff}_{\alpha_i} = E_w \cup O_w$. In the following paragraphs, we demonstrate $(w, w') \in T(\alpha_i)$ by the semantics of atomic action definitions.

Firstly, it is obvious that $(M, w) \models P \cup P_w$.

Secondly, for any concept name A which is primitive w.r.t. $\mathcal{T}$, since $\{p^I \mid A(p) \in E_w\} \cap \{p^I \mid \neg A(p) \in E_w\} = A_w^+ \cap A_w^- = \emptyset$, we have $\{p^I \mid A(p) \in E_w \cup O_w\} \cap \{p^I \mid \neg A(p) \in E_w \cup O_w\} = \emptyset$. In order to demonstrate $A^{I(w')} = (A^{I(w)} \cup \{p^I \mid A(p) \in E_w \cup O_w\}) \setminus \{p^I \mid \neg A(p) \in E_w \cup O_w\}$, there are two cases to be investigated for each $x \in \Delta$:

- $x \in I_A^w$. Then, since $A^{I(w')} \cap I_A^w = ((A^{I(w)} \cup A_w^+) \setminus A_w^-) \cap I_A^w$, we have $x \in A^{I(w')}$ if and only if $x \in (A^{I(w)} \cup \{ p^I \mid A(p) \in E_w \}) \setminus \{p^I \mid \neg A(p) \in E_w\}$. Now, we demonstrate the equation from two directions.
  - If $x \in A^{I(w')}$, then we have $x \in A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}$ and $x \notin \{p^I \mid \neg A(p) \in E_w\}$. At the same time, it must be $x \notin \{p^I \mid \neg A(p) \in O_w\}$; otherwise, according to the construction of the set O_w, we have $x \notin A^{I(w')}$ and get a contradiction. Therefore, we have $x \in (A^{I(w)} \cup \{p^I \mid A(p) \in E_w \cup O_w\}) \setminus \{p^I \mid \neg A(p) \in E_w \cup O_w\}$.
  - If $x \in (A^{I(w)} \cup \{p^I \mid A(p) \in E_w \cup O_w\}) \setminus \{ p^I \mid \neg A(p) \in E_w \cup O_w\}$, then we have $x \notin \{p^I \mid \neg A(p) \in E_w\}$, and either $x \in A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}$ or $x \in \{p^I \mid A(p) \in O_w\}$. In the case that $x \in A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}$, we have $x \in (A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}) \setminus \{p^I \mid \neg A(p) \in E_w\}$ and therefore $x \in A^{I(w')}$; in the case that $x \in \{p^I \mid A(p) \in O_w\}$, according to the construction of the set O_w, we have also $x \in A^{I(w')}$.

- $x \notin I_A^w$. Then, we have $x \in \{p^I \mid A(p) \in O\}$, $x \notin \{p^I \mid A(p) \in E_w\}$ and $x \notin \{p^I \mid \neg A(p) \in E_w\}$, and therefore there must be some individual name p_0 such that $x = p_0^I$ and $A(p_0) \in O'$. Now, we demonstrate the equation from two directions.
  - If $x \in A^{I(w')}$, then we have $A(p_0) \in O_w$ according to the construction of the set O_w, and consequently $x \in \{p^I \mid A(p) \in O_w\}$. At the same time, it must be $x \notin \{p^I \mid \neg A(p) \in O_w\}$; otherwise, according to the construction of the set O_w, we have $x \notin A^{I(w')}$ and get a contradiction. Therefore, we have $x \in (A^{I(w)} \cup \{p^I \mid A(p) \in E_w \cup O_w\}) \setminus \{p^I \mid \neg A(p) \in E_w \cup O_w\}$.
  - If $x \in (A^{I(w)} \cup \{p^I \mid A(p) \in E_w \cup O_w\}) \setminus \{p^I \mid \neg A(p) \in E_w \cup O_w\}$, then we have $x \notin \{p^I \mid \neg A(p) \in O_w\}$ and consequently $\neg A(p_0) \notin O_w$. So, according to the construction of the set O_w, we have $A(p_0) \in O_w$ and $(M, w') \models A(p_0)$. Therefore, we have $x \in A^{I(w')}$.

Thirdly, be similar with the above demonstration on primitive concept names, it can be proved that both $\{(p^I, q^I) \mid R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\} \cap \{(p^I, q^I) \mid \neg R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\} = \emptyset$ and $R^{I(w')} = (R^{I(w)} \cup \{(p^I, q^I) \mid R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\}) \setminus \{(p^I, q^I) \mid \neg R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\}$ for each role name R.

To sum up, we have $(w, w') \in T(\alpha_i)$ and consequently $(w, w') \in T(\alpha_1 \cup ... \cup \alpha_n)$.

(*The Only-if direction*) Let $(w, w') \in T(\alpha_1 \cup ... \cup \alpha_n)$. Then there must be some atomic action α_i $(1 \leq i \leq n)$ with $(w, w') \in T(\alpha_i)$. Furthermore, during the execution of the procedure $Unfold(\alpha)$, there must be some sets P_{cond}, $\mathcal{A}_i$, E_i, O_i and $O_{i,j}$ such that:

- $P_{cond} = \{\varphi \mid \varphi/\psi \in E\}$,
- $\mathcal{A}_i \subseteq P_{cond} \cup \{\varphi^\neg \mid \varphi \in P_{cond}\}$, and it is either $\varphi \in \mathcal{A}_i$ or $\varphi^\neg \in \mathcal{A}_i$ for every $\varphi \in P_{cond}$,
- $E_i = \{\psi \mid \varphi/\psi \in E$ and $\varphi \in \mathcal{A}_i\}$,
- $O_i = \{\phi \mid \phi \in O, \phi \notin E_i{}^*_{\mathcal{R}}$ and $\phi^\neg \notin E_i{}^*_{\mathcal{R}}\}$,
- $O_{i,j} \subseteq O_i \cup \{\phi^\neg \mid \phi \in O_i\}$, and it is either $\phi \in O_{i,j}$ or $\phi^\neg \in O_{i,j}$ for every $\phi \in O_i$,
- $\mathbf{Pre}_{\alpha_i} = P \cup \mathcal{A}_i$, and $\mathbf{Eff}_{\alpha_i} = E_i \cup O_{i,j}$.

By the semantics of atomic action definitions, we have $(M, w) \models P \cup \mathcal{A}_i$, both $\{p^I \mid A(p) \in E_i \cup O_{i,j}\} \cap \{p^I \mid \neg A(p) \in E_i \cup O_{i,j}\} = \emptyset$ and $A^{I(w')} = (A^{I(w)} \cup \{p^I \mid A(p) \in E_i \cup O_{i,j}\}) \setminus \{p^I \mid \neg A(p) \in E_i \cup O_{i,j}\}$ for each concept name A which is primitive w.r.t. $\mathcal{T}$, and both $\{(p^I, q^I) \mid R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\} \cap \{(p^I, q^I) \mid \neg R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\} = \emptyset$ and $R^{I(w')} = (R^{I(w)} \cup \{(p^I, q^I) \mid R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\}) \setminus \{(p^I, q^I) \mid \neg R(p,q) \in \mathbf{Eff}_{\alpha_i \mathcal{R}}^*\}$ for each role name R.

Let $E_w := \{\psi \mid \varphi/\psi \in E$ and $(M, w) \models \varphi\}$. Then it is obvious that $E_w = E_i$. In the following paragraphs, we demonstrate $(w, w') \in T(\alpha)$ by the semantics of extended atomic action definitions.

Firstly, it is straightforward that $(M, w) \models P$.

Secondly, let A be any concept name which is primitive w.r.t. $\mathcal{T}$, and let $I_A^w := (\Delta \setminus \{p^I \mid A(p) \in O\}) \cup \{p^I \mid A(p) \in E_w\} \cup \{p^I \mid \neg A(p) \in E_w\}$. Since $\{p^I \mid A(p) \in E_i \cup O_{i,j}\} \cap \{p^I \mid \neg A(p) \in E_i \cup O_{i,j}\} = \emptyset$, we have $\{p^I \mid A(p) \in E_w\} \cap \{p^I \mid \neg A(p) \in E_w\} = \emptyset$. Now, we demonstrate the equation $A^{I(w')} \cap I_A^w = ((A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}) \setminus \{p^I \mid \neg A(p) \in E_w\}) \cap I_A^w$ by investigating each individual $x \in \Delta$:

- If $x \in A^{I(w')} \cap I_A^w$, then we have both $x \in A^{I(w')}$ and $x \in I_A^w$, and therefore $x \in A^{I(w)} \cup \{p^I \mid A(p) \in E_i\} \cup \{p^I \mid A(p) \in O_{i,j}\}$, $x \notin \{p^I \mid \neg A(p) \in E_i\}$, and $x \notin \{p^I \mid \neg A(p) \in O_{i,j}\}$. Let's assume that $x \notin A^{I(w)} \cup \{p^I \mid A(p) \in E_i\}$, then it must

be $x \in \{p^I \mid A(p) \in O_{i,j}\}$, and therefore we have $x \in \{p^I \mid A(p) \in O\}$ according to the construction of the set $O_{i,j}$; at the same time, since $x \in I_A^w$ and $x \notin \{p^I \mid \neg A(p) \in E_i\}$, we have $x \in \Delta \setminus \{p^I \mid A(p) \in O\}$ and consequently $x \notin \{p^I \mid A(p) \in O\}$, and therefore get a contradiction. So, it must be $x \in A^{I(w)} \cup \{p^I \mid A(p) \in E_i\}$. Therefore, we have $x \in ((A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}) \setminus \{p^I \mid \neg A(p) \in E_w\}) \cap I_A^w$.

– If $x \in ((A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}) \setminus \{p^I \mid \neg A(p) \in E_w\}) \cap I_A^w$, then we have $x \in A^{I(w)} \cup \{p^I \mid A(p) \in E_w\}$, $x \notin \{p^I \mid \neg A(p) \in E_w\}$ and $x \in I_A^w$. Let's assume that $x \in \{p^I \mid \neg A(p) \in O_{i,j}\}$; then, according to the construction of the set $O_{i,j}$, we have $x \in \{p^I \mid A(p) \in O\}$ and consequently $x \notin \Delta \setminus \{p^I \mid A(p) \in O\}$; furthermore, since $x \in I_A^w$ and $x \notin \{p^I \mid \neg A(p) \in E_w\}$, we have $x \in \{p^I \mid A(p) \in E_w\}$ and consequently $x \in \{p^I \mid \neg A(p) \in O_{i,j}\} \cap \{p^I \mid A(p) \in E_w\}$, which is contradictory with the fact that $\{p^I \mid A(p) \in E_i \cup O_{i,j}\} \cap \{p^I \mid \neg A(p) \in E_i \cup O_{i,j}\} = \emptyset$. So, it must be $x \notin \{p^I \mid \neg A(p) \in O_{i,j}\}$. Therefore, we have $x \in (A^{I(w)} \cup \{p^I \mid A(p) \in E_i \cup O_{i,j}\}) \setminus \{p^I \mid \neg A(p) \in E_i \cup O_{i,j}\}$ and consequently $x \in A^{I(w')} \cap I_A^w$.

Thirdly, be similar with the above demonstration on primitive concept names, it can be proved that both $\{(p^I, q^I) \mid R(p,q) \in E_{w\mathcal{R}}^*\} \cap \{(p^I, q^I) \mid \neg R(p,q) \in E_{w\mathcal{R}}^*\} = \emptyset$ and $R^{I(w')} \cap I_R^w = ((R^{I(w)} \cup \{(p^I, q^I) \mid R(p,q) \in E_{w\mathcal{R}}^*\}) \setminus \{(p^I, q^I) \mid \neg R(p,q) \in E_{w\mathcal{R}}^*\}) \cap I_R^w$ for each role name R, where $I_R^w := ((\Delta \times \Delta) \setminus \{(p^I, q^I) \mid R(p,q) \in O\}) \cup \{(p^I, q^I) \mid R(p,q) \in E_{w\mathcal{R}}^*\} \cup \{(p^I, q^I) \mid \neg R(p,q) \in E_{w\mathcal{R}}^*\}$.

To sum up, we have $(w, w') \in T(\alpha)$. □

Lemma 9 *Let α be an atomic action defined by some extended atomic action definition $\alpha \equiv (P, O, E)$ w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$; let $\alpha_1 \cup ... \cup \alpha_n$ be the action returned by the procedure $Unfold(\alpha)$; let $\mathcal{A}_\mathcal{C}$ be an ActBox in which each α_i ($1 \le i \le n$) is defined by some atomic action definition; and for each atomic action α_i ($1 \le i \le n$), let $\phi_{i,1}, ..., \phi_{i,k_i}$ be all the ABox assertions contained in $\mathbf{Pre}_{\alpha_i}$. Then, for any model $M = (W, T, \Delta, I)$ with $M \models \mathcal{R}$, $M \models \mathcal{T}$, $M \models_{\mathcal{R},\mathcal{T}} \mathcal{A}_\mathcal{C}$ and $M \models_{\mathcal{R},\mathcal{T}} \alpha \equiv (P, O, E)$, it must be $T(\alpha_i) = T(\phi_{i,1}?; ...; \phi_{i,k_i}?; \alpha)$.*

Proof For any states $w, w' \in W$, we demonstrate that $(w, w') \in T(\alpha_i)$ if and only if $(w, w') \in T(\phi_{i,1}?; ...; \phi_{i,k_i}?; \alpha)$.

(*The If direction*) Let $(w, w') \in T(\phi_{i,1}?; ...; \phi_{i,k_i}?; \alpha)$. Then we have $(w, w) \in T(\phi_{i,1}?; ...; \phi_{i,k_i}?)$ and $(w, w') \in T(\alpha)$. Therefore, we have $(M, w) \models Conj(\mathbf{Pre}_{\alpha_i})$ since $\mathbf{Pre}_{\alpha_i} = \{\phi_{i,1}, ..., \phi_{i,k_i}\}$. Furthermore, for any atomic action α_j with $1 \le j \le n$ and $j \ne i$, by investigating the operations of the procedure $Unfold(\alpha)$, it is straightforward that the formula $Conj(\mathbf{Pre}_{\alpha_i}) \wedge Conj(\mathbf{Pre}_{\alpha_j})$ is unsatisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$; therefore, we have $(M, w) \models \neg Conj(\mathbf{Pre}_{\alpha_j})$ and consequently $(w, w') \notin T(\alpha_j)$. At the same time, since $(w, w') \in T(\alpha)$, we have $(w, w') \in T(\alpha_1 \cup ... \cup \alpha_n)$ by Lemma 8. So, it must be $(w, w') \in T(\alpha_i)$.

(*The Only-if direction*) Let $(w, w') \in T(\alpha_i)$. Then, we have $(w, w') \in T(\alpha_1 \cup ... \cup \alpha_n)$, $(M, w) \models Conj(\mathbf{Pre}_{\alpha_i})$ and consequently $(w, w) \in T(\phi_{i,1}?; ...; \phi_{i,k_i}?)$. Therefore, by Lemma 8, we have $(w, w') \in T(\alpha)$ and consequently $(w, w') \in T(\phi_{i,1}?; ...; \phi_{i,k_i}?; \alpha)$. □

Now, we can extend reasoning mechanisms discussed in previous sections to support extended atomic action definitions.

Firstly, we introduce the PE-consistency problem of extended atomic action definitions.

Definition 19 An extended atomic action definition $\alpha \equiv (P, O, E)$ is *PE-consistent* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if there is a model $M = (W, T, \Delta, I)$ and two states $w, w' \in W$ such that $M \models \mathcal{R}$, $M \models \mathcal{T}$, $(M, w) \models P$ and $(M, w') \models \{\psi \mid \varphi/\psi \in E \text{ and } (M, w) \models \varphi\}$.

With the help of the $Unfold()$ procedure, the PE-consistency problem of extended atomic action definitions can in fact be reduced to the PE-consistency problem of atomic action definitions; i.e.:

Theorem 9 *An extended atomic action definition $\alpha \equiv (P, O, E)$ is PE-consistent w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if the action returned by the procedure $Unfold(\alpha)$ is not the atomic action β_0 defined by $\beta_0 \equiv (\{false\}, \emptyset)$.*

Secondly, we extend the ActBox of $DDL(X^{@})$ to include extended atomic action definitions; i.e., for each finite set $\mathcal{A}_C$ of atomic action definitions and extended atomic action definitions, if no action name occurs on the left-hand sides for more than once, then call $\mathcal{A}_C$ an *ActBox* of $DDL(X^{@})$.

Correspondingly, the terms and notations related with ActBoxes should be extended. For example,

- an ActBox $\mathcal{A}_C$ is called *PE-consistent* w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if and only if all the elements of $\mathcal{A}_C$ are PE-consistent w.r.t. $\mathcal{R}$ and $\mathcal{T}$;
- an atomic action α is called *defined* in an ActBox $\mathcal{A}_C$ if and only if α occurs on the left-hand side of some element of $\mathcal{A}_C$.

Finally, by the following algorithm, we extend the satisfiability-checking algorithm presented in Section 5 to support extended atomic action definitions.

Algorithm 2 *Let $\mathcal{A}_C$ be an ActBox which is PE-consistent w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$. For any formula φ defined w.r.t. $\mathcal{A}_C$, we decide whether it is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C$ according to the following steps:*

1. *Construct an (initially empty) set $\mathcal{A}_C{}'$ of atomic action definitions.*
2. *Construct a formula $\varphi' := \varphi$.*
3. *For every atomic action α occurring in φ':*
   - *if α is defined by some atomic action definition $\alpha \equiv (P, E) \in \mathcal{A}_C$, then add $\alpha \equiv (P, E)$ into the set $\mathcal{A}_C{}'$;*
   - *if α is defined by some extended atomic action definition $\alpha \equiv (P, O, E) \in \mathcal{A}_C$, then do the following operations sequentially:*
     (a) *call the procedure $Unfold(\alpha)$ and let $\beta_1 \cup ... \cup \beta_n$ be the action returned by it;*
     (b) *add every atomic action definition of β_i $(1 \leq i \leq n)$ into the set $\mathcal{A}_C{}'$; and*
     (c) *replace each occurrence of α in the formula φ' by the action $\beta_1 \cup ... \cup \beta_n$.*
4. *If φ' is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C{}'$ according to Algorithm 1, then return "TRUE", else return "FALSE".*

Based on Lemma 7 and Theorem 7, it is obvious that this algorithm terminates. Furthermore, by Lemma 8, the following result is straightforward:

Theorem 10 *Algorithm* 2 *returns "TRUE" if and only if the formula* φ *is satisfiable w.r.t.* $\mathcal{R}$, $\mathcal{T}$ *and* $\mathcal{A}_C$.

In the following paragraphs, we investigate the complexity of Algorithm 2.

First of all, we extend the notation of size introduced in Section 5.2 to support extended atomic action definitions; i.e., for each extended atomic action definition $\alpha \equiv (P, O, E)$, its *size* is defined as $|\alpha \equiv (P, O, E)| = |\alpha| + \sum_{\varphi \in P}(|\varphi|) + \sum_{\phi \in O}(|\phi|) + \sum_{\varphi/\psi \in E}(|\varphi| + |\psi|) + 1$.

Next, we investigate the complexity of the $Unfold(\,)$ procedure. Let α be an atomic action defined by some extended atomic action definition $\alpha \equiv (P, O, E)$; and let $\alpha_1 \cup ... \cup \alpha_n$ be the action returned by the procedure $Unfold(\alpha)$. Then, according to the proof of Lemma 7, it is obvious that the size $|\alpha_1 \cup ... \cup \alpha_n|$ is linearly bounded by $|\alpha| \times 2^{\sharp O} \times 2^{\sharp E}$. Furthermore, based on the result presented in Theorem 1, it is straightforward to show that, for the family $DDL(X^@)$ of dynamic description logics, the complexity upper-bound of the $Unfold(\,)$ procedure is EXPTIME if $X \in \{\mathcal{ALCO}, \mathcal{ALCHO}, \mathcal{ALCOQ}, \mathcal{ALCHOQ}, \mathcal{ALCOI}, \mathcal{ALCHOI}\}$, and is NEXPTIME if $X \in \{\mathcal{ALCOIQ}, \mathcal{ALCHOIQ}\}$.

Based on the above results, it is obvious that the complexity upper-bound of Algorithm 2 is dependent on Step 4 of the algorithm.

Now, suppose Algorithm 1 is called by Step 4 to decide whether the formula φ' is satisfiable w.r.t. $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}_C{}'$. Let m_O (resp. m_E) be the maximal one among $\sharp O$ (resp. $\sharp E$) for every extended atomic action definition $\alpha \equiv (P, O, E) \in \mathcal{A}_C$. Then, it is obvious that the size $|\varphi'|$ is linearly bounded by $|\varphi| \times 2^{m_O} \times 2^{m_E}$, and the size $|\mathcal{A}_C{}'|$ is linearly bounded by $|\mathcal{A}_C| \times 2^{m_O} \times 2^{m_E}$. Therefore, it seems that the complexity of Algorithm 2 will be increased exponentially. But in fact, it is not the case.

Let $f := |nf(\varphi')|$, $c := \sharp cl_{\mathcal{R}}(\varphi')$, $a := \sharp Ass_{\mathcal{R}}(\varphi')$ and $e := \sharp Eff_{\mathcal{R}}(\varphi')$, where the sets $cl_{\mathcal{R}}(\varphi')$, $Ass_{\mathcal{R}}(\varphi')$ and $Eff_{\mathcal{R}}(\varphi')$ are constructed according to the definitions presented in Section 5.2. Let m be the maximal one among $|\psi|$ for every $\psi \in Ass_{\mathcal{R}}(\varphi')$. Then, according to the proof of Theorem 7, the complexity of Algorithm 1 is determined by the integers c, a, e and m, where the numbers c, a and e are linearly bounded by $f \times |\mathcal{R}|$, and the number m is bounded by f. At the same time, the number f is linearly bounded by $|\varphi| \times 2^{m_O} \times 2^{m_E} \times |\mathcal{A}_C|$.

In Algorithm 2, the number c is still linearly bounded by $f \times |\mathcal{R}|$ and consequently linearly bounded by $|\varphi| \times 2^{m_O} \times 2^{m_E} \times |\mathcal{A}_C| \times |\mathcal{R}|$. However, for the integers a, e and m, we can find some better upper bounds for them, and the exponential increase caused by the size of φ' can be avoided.

Firstly, let $\alpha_1, ..., \alpha_n$ be all the different atomic actions occurring in φ. For each α_i $(1 \le i \le n)$, let $E'_{\alpha_i} := E_i$ if α_i is defined by some atomic action definition $\alpha_i \equiv (P_i, E_i)$, and let $E'_{\alpha_i} := O_i \cup \{\psi \mid \varphi/\psi \in E_i\}$ if α_i is defined by some extended atomic action definition $\alpha_i \equiv (P_i, O_i, E_i)$. Based on these notations, construct a set $Eff^{ex}_{\mathcal{R}}(\varphi)$ as follows:

$$Eff^{ex}_{\mathcal{R}}(\varphi) := E'^{\,*}_{\alpha_1\,\mathcal{R}} \cup ... \cup E'^{\,*}_{\alpha_n\,\mathcal{R}}.$$

Then, it is obvious that $\sharp Eff^{ex}_{\mathcal{R}}(\varphi)$ is linearly bounded by $|\mathcal{A}_\mathcal{C}| \times |\mathcal{R}|$. Furthermore, we have $Eff_{\mathcal{R}}(\varphi') \subseteq Eff^{ex}_{\mathcal{R}}(\varphi)$. Therefore, the number e is linearly bounded by $|\mathcal{A}_\mathcal{C}| \times |\mathcal{R}|$.

Secondly, for any formula or action X, we use $Ass^{ex}(X)$ to denote an ABox defined inductively as follows:

- $Ass^{ex}(X) = \{X, X^{\neg}\}$ if X is a concept assertion or a role assertion;
- $Ass^{ex}(X) = Ass^{ex}(\psi)$ if X is a formula of the form $\neg\psi$;
- $Ass^{ex}(X) = Ass^{ex}(\pi) \cup Ass^{ex}(\psi)$ if X is a formula of the form $< \pi > \psi$ or $[\pi]\psi$;
- $Ass^{ex}(X) = Ass^{ex}(\psi_1) \cup Ass^{ex}(\psi_2)$ if X is a formula of the form $\psi_1 \vee \psi_2$ or $\psi_1 \wedge \psi_2$;
- $Ass^{ex}(X) = \bigcup_{\varphi \in P} Ass^{ex}(\varphi) \cup \bigcup_{\phi \in E} Ass^{ex}(\phi)$ if X is an atomic action defined by some atomic action definition $X \equiv (P, E)$;
- $Ass^{ex}(X) = \bigcup_{\varphi \in P} Ass^{ex}(\varphi) \cup \bigcup_{\phi \in O} Ass^{ex}(\phi) \cup \bigcup_{\varphi/\psi \in E} (Ass^{ex}(\varphi) \cup Ass^{ex}(\psi))$ if X is an atomic action defined by some extended atomic action definition $X \equiv (P, O, E)$;
- $Ass^{ex}(X) = Ass^{ex}(\varphi)$ if X is an action of the form $\varphi?$;
- $Ass^{ex}(X) = Ass^{ex}(\pi_1) \cup Ass^{ex}(\pi_2)$ if X is an action of the form $\pi_1 \cup \pi_2$ or $\pi_1; \pi_2$;
- $Ass^{ex}(X) = Ass^{ex}(\pi_1)$ if X is an action of the form π_1^*.

Furthermore, let $Ass^{ex}_{\mathcal{R}}(X)$ be the closure of $Ass^{ex}(X)$ w.r.t. $\mathcal{R}$. Then, it is obvious that $\sharp Ass^{ex}_{\mathcal{R}}(\varphi)$ is linearly bounded by $(|\varphi| + |\mathcal{A}_\mathcal{C}|) \times |\mathcal{R}|$. Furthermore, we have $Ass_{\mathcal{R}}(\varphi') \subseteq Ass^{ex}_{\mathcal{R}}(\varphi)$. Therefore, the number a is linearly bounded by $(|\varphi| + |\mathcal{A}_\mathcal{C}|) \times |\mathcal{R}|$.

Finally, Let m^{ex} be the maximal one among $|\psi|$ for every $\psi \in Ass^{ex}_{\mathcal{R}}(\varphi)$. Then, it is obvious that $m \leq m^{ex}$ and m^{ex} is linearly bounded by $|\varphi| + |\mathcal{A}_\mathcal{C}|$. Therefore, the number m is linearly bounded by $|\varphi| + |\mathcal{A}_\mathcal{C}|$.

According to the proof of Theorem 7, during the execution of Algorithm 1, the number of branches which will be generated is finite. Furthermore, for each branch $\mathcal{B}$ generated by the algorithm, the number of prefixed formulas contained in it is bounded by $(2^e - 1) \times c + (\ c + a \times (2^e - 1)\ )$; the number of elements contained in the ABox $IV_\mathcal{B}$ is bounded by $a \times 2^e$; and there exist some polynomials p_1 and p_2 such that $|IV_\mathcal{B}| \leq (a \times 2^e) \times (m \times 2^{p_1(|\mathcal{T}|)} \times (p_2(e))^{m+|\mathcal{T}|})$. Therefore, to sum up, we can get the following result on the complexity upper-bounds of Algorithm 2.

Theorem 11 *For the family $DDL(X^@)$ of dynamic description logics, the complexity upper-bound of Algorithm* 2 *is EXPSPACE if $X \in \{\mathcal{ALCO}, \mathcal{ALCHO}, \mathcal{ALCOQ}, \mathcal{ALCHOQ}\}$, and is N2EXPTIME if $X \in \{\mathcal{ALCOI}, \mathcal{ALCHOI}, \mathcal{ALCOIQ}, \mathcal{ALCHOIQ}\}$.*

We conclude this section with the result that both the domain constraints and the actions described by Baader et al.'s formalism [4] are now supported by $DDL(X^@)$.

7 Related Works

We have proposed a family of dynamic description logics for representing and reasoning about actions. Related works are organized as six groups.

7.1 Representing and Reasoning About Actions with Description Logics

The idea of adopting description logics for representing and reasoning about actions is not new. Two typical formalisms based on this idea are the RAT (Representation of Actions using Terminological logics) system [20] and the CLASP (CLassification of Scenarios and Plans) system [11]. In both systems, world states are represented as concept expressions of description logics; atomic actions are described in the STRIPS style [13] with pre-conditions, add-lists and delete-lists; and each finite sequence of atomic actions is treated as a plan individual. Moreover, the CLASP system introduces the notation of plan concepts. It defines each plan concept as a triple composed of INITIAL state, GOAL state and PLAN-EXPRESSION, where each PLAN-EXPRESSION is constructed from atomic actions with the help of the SEQUENCE, LOOP, REPEAT, TEST, OR, and SUBPLAN constructors. Based on standard reasoning mechanisms of description logics (and with the help of the finite automaton theory), reasoning tasks on actions and plans can be effectively carried out. A limitation of these two formalisms is that they work with the Closed World Assumption (CWA), and therefore require complete knowledge about the problem to be given.

Compared with the above systems, our formalism describes the world states and the pre- and post-conditions of atomic actions with ABox assertions; therefore, actions in our formalism are in fact represented "over" description logics. Moreover, the Open World Assumption (OWA) adopted in description logics is preserved in the satisfiability-checking algorithm of $DDL(X^{@})$, so that all the reasoning tasks introduced in our formalism can be carried out with incomplete knowledge about the world.

7.2 Reasoning About Actions with Temporal Description Logics

Based on a combination of interval-based temporal logics and feature description logics, Artale and Franconi [2] proposed a class of temporal description logics for reasoning about actions. With these logics, not only the world states but also the actions and the plans are represented as concepts. Each state describes a collection of properties of the world holding at certain time; each action is represented through temporal constraints on states, by describing what is true while the action itself is occurring; and each plan is constructed by temporally relating actions and states. Artale et al. provided sound and complete algorithms for deciding the subsumption relationship between concepts. Based on these algorithms, many reasoning tasks on actions and plans can be effectively carried out, such as deciding the subsumption between plans and deciding the instance relationship between individual plans and plan types.

Compared with Artale et al.'s formalisms, our work is characterized by integrating dynamic logics with description logics. Therefore, on the one hand, many complex temporal properties which are captured by Artale et al.'s formalisms can't be represented with our formalism; on the other hand, many complex control structures on actions that are supported by our formalism can't be described by Artale et al.'s formalisms. Additionally, the reasoning tasks investigated by Artale et al. are the action/plan taxonomy problem and the problem of recognizing plans with respect

to plan descriptions; our formalism, however, focus on the realizability problem, the executability problem and the projection problem of actions.

7.3 Action Formalisms Constructed Over Description Logics

A formalism in which actions are represented over description logics was firstly proposed by Lutz and Sattler [26]. In their formalism, each action was described as a triple consisting of a finite set of pre-conditions, a finite set of relaxations, and a finite set of conditional post-conditions, where each condition was an ABox assertion or an extended assertion of the form $\forall C$. Taking each finite sequence of actions as a service, inference problems such as the realizability of services, the subsumption between services and the projection problems were introduced. However, reasoning mechanisms for these inference problems were not provided.

A realizable action formalism constructed over description logics was presented by Baader et al. [4]. Based on this formalism, Liu et al. [24] investigated the ramification problem induced by general concept inclusions in the case that general TBoxes was incorporated; and Miličić [30] investigated the planning problem.

As discussed in the Introduction, a limitation of the above works is that they do not support complex control structures of actions. This limitation is overcome in our formalism by combining Baader et al.'s formalism with the propositional dynamic logic PDL.

7.4 DL-Based Restrictions of Classical Action Formalisms

Inspired by Baader et al.'s action formalism, Gu and Soutchanski [19] proposed a modified version of the Situation Calculus [36]. The modification is embodied in two aspects. Firstly, the first-order logic used in the Situation Calculus is restricted to be the C^2 logic [17, 34], so that the executability problem and the projection problem on actions are guaranteed to be decidable. Secondly, based on the fact that the C^2 logic and the description logic $\mathcal{ALCQI}(\sqcup, \sqcap, \neg, |, id)$ are equally expressive, the RBoxes and TBoxes of description logics are included explicitly in the action theory of Situation Calculus

With a similar motivation, Drescher and Thielscher [12] proposed to use ABoxes of description logics as decidable state descriptions in the basic Fluent Calculus.

Both Gu et al.'s work [19] and Drescher et al.'s work [12] can be treated as DL-based restrictions of classical action formalisms constructed over first- or higher-order logics. However, our work can be viewed as a DL-based extension of the action formalism constructed over propositional dynamic logics [9]. From the point of knowledge representation and reasoning, a feature of $DDL(X^{@})$ is that properties on actions can be explicitly stated by formulas, and consequently many inference problems on actions can be reduced to the satisfiability problem of $DDL(X^{@})$-formulas.

7.5 Dynamic Extensions of Description Logics

A dynamic description logic named $PDLC$ was constructed by Wolter et al. [43] as a combination of the propositional dynamic logic PDL and the description logic $\mathcal{ALC}$. A feature of $PDLC$ is that actions are used as model operators for constructing

not only the formulas but also the concepts. Therefore, concepts with dynamic meanings can be described by $PDLC$. For example, a concept *Easy_cured_child* can be specified by the following concept definition:

$$\begin{aligned} \mathit{Easy\_cured\_child} \equiv\ & \mathit{Child} \wedge \exists \mathit{has}.\mathit{Angina} \\ & \wedge < (\mathit{give\_honey} \cup \mathit{give\_aspirin})^* > \neg \exists \mathit{has}.\mathit{Angina}, \end{aligned}$$

which refers to the children suffering from angina that can be cured by using honey and aspirin [43]. Wolter et al. demonstrated that the logic $PDLC$ was still decidable; but the complexity of the decision problem and the development of efficient decision algorithms were left as open problems.

Compared with Wolter et al's work, the motivation of our work is to provide a kind of action formalisms for describing and reasoning about actions. Therefore, on the one hand, actions are not used as model operators for the construction of $DDL(X^{@})$-concepts. On the other hand, an action formalism constructed over description logics is incorporated in $DDL(X^{@})$; with this action formalism, each atomic action in $DDL(X^{@})$ is further specified by an atomic action definition or an extended atomic action definition, and is interpreted according to the minimal-change semantics [8, 38].

7.6 Updating Description Logic ABoxes

Taking ABoxes of description logics as a tool for describing the state of affairs in an application domain, Liu et al. [23] proposed a theory for updating ABoxes. In that theory, the initial state of an application domain was described by an ABox $\mathcal{A}$ of some description logic of the $\mathcal{ALCQIO}$ family; the update was specified by a restricted ABox $\mathcal{U}$ which was composed of primitive literals; and the semantics of updating $\mathcal{A}$ with $\mathcal{U}$ was defined according to the minimal-change semantics. Liu et al. firstly demonstrated that both the nominals and the "@" constructor were necessary for a description logic to represent the updated ABoxes. Then, based on a technically designed procedure for constructing updated concept $C^{\mathcal{U}}$ w.r.t. any concept C and any update $\mathcal{U}$, Liu et al. provided algorithms for computing updated ABoxes; the complexity of these algorithms were also investigated.

In our formalism, the regression operator $C^{Regress(\varepsilon,\mathcal{T})}$ presented in Section 5 is in fact an inverse operator of Liu et al's process for constructing the updated concept $C'^{\mathcal{U}}$. For any prefix $\sigma.\varepsilon$ introduced in our satisfiability-checking algorithm, the set ε is used to record the accumulated post-conditions for the sequential action σ. Therefore, the set ε in our regression operator can be treated as the update $\mathcal{U}$ in Liu et al.'s theory; the concept C can be treated as an updated concept $C'^{\mathcal{U}}$ for some concept C'; and the target of the regression operator $C^{Regress(\varepsilon,\mathcal{T})}$ is just to compute the concept C'.

8 Conclusion

In this paper we constructed a family of dynamic description logics named $DDL(X^{@})$, where X represents well-studied description logics ranging from the

$\mathcal{ALCO}$ to the $\mathcal{ALCHOIQ}$, and $X^@$ denotes the extension of X with the @ constructor. As a combination of the description logic $X^@$, the propositional dynamic logic PDL and an action formalism proposed by Baader et al. [4], the logic $DDL(X^@)$ offers considerable expressive power for the description of actions. We also developed a tableau algorithm for deciding the satisfiability of $DDL(X^@)$-formulas. Based on this algorithm, reasoning tasks on actions, such as the realizability problem, the executability problem and the projection problem, can all be effectively carried out.

The logic $DDL(X^@)$ provides an approach to bring the power and character of description logics into the description and reasoning of dynamic application domains. One of our future work is to study the planning problem based on $DDL(X^@)$. Another work is to represent and reason about semantic Web services with $DDL(X^@)$.

Acknowledgements The authors are very grateful to anonymous reviewers for their helpful comments and suggestions during the preparation of this paper.

References

1. Areces, C., Blackburn, P., Marx, M.: A roadmap on complexity for hybrid logics. In: Proc. of the 13th Int'l Workshop and 8th Annual Conf. of the EACSL on Computer Science Logic (CSL'05). LNCS, vol. 1683, pp. 307–321. Springer (1999)
2. Artale, A., Franconi, E.: A temporal description logic for reasoning about actions and plans. J. Artif. Intell. Res. **9**, 463–506 (1998)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
4. Baader, F., Lutz, C., Miličić, M., Sattler, U., Wolter., F.: Integrating description logics and action formalisms: first results. In: Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'05), pp. 572–577. AAAI Press / MIT Press (2005)
5. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. **284**(5), 34–43 (2001)
6. Calvanese, D., De Giacomo, G., Vardi, M.: Reasoning about actions and planning in LTL action theories. In: Proc. of the 8th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'02), pp. 593–602. AAAI Press (2002)
7. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Actions and programs over description logic ontologies. In: Proc. of the 20th Int'l Workshop on Description Logics (DL'07), pp. 29–40 (2007)
8. Chang, L., Shi, Z., Qiu, L., Lin, F.: Dynamic description logic: embracing actions into description logic. In: Proc. of the 20th Int'l Workshop on Description Logics (DL'07), pp. 243–250 (2007)
9. De Giacomo, G., Lenzerini, M.: PDL-based framework for reasoning about actions. In: Proc. of the 4th Congresss of the Italian Association for Artificial Intelligence (AI*IA'95), pp. 103–114. Springer, Berlin (1995)
10. De Giacomo, G., Massacci, F.: Tableaux and algorithms for propositional dynamic logic with converse. In: Proc. of the 13th Int'l Conf. on Automated Deduction (CADE'96), pp. 613–628. Springer, Berlin (1996)
11. Devanbu, P.T., Litman, D.J.: Taxonomic plan reasoning. Artif. Intell. **84**(1–2), 1–35 (1996)
12. Drescher, C., Thielscher, M.: Integrating action calculi and description logics. In: Proc. of the 30th Annual German Conf. on Artificial Intelligence (KI'07), pp. 68–83. Springer, Berlin (2007)
13. Fikes, R.E., Hart, P.E., Nilsson, N.J.: Learning and executing generalized robot plans. Artif. Intell. **3**, 251–288 (1972)
14. Fischer M.J., Ladner, R.E.: Propositional modal logic of programs: extended abstract. In: Proc. of the 9th Annual ACM Symposium on Theory of Computing (STOC'77), pp. 286–294 (1977)
15. Fischer M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (1979)

16. Giordano, L., Martelli, A., Schwind, C.: Reasoning about actions in dynamic linear time temporal logic. Logic J. IGPL **9**(3), 79–92 (2001)
17. Grädel E., Otto, M., Rosen, E.: Two-variable logic with counting is decidable. In: Proc. of the 12th Annual IEEE Symposium on Logic in Computer Science (LICS'97), pp. 306–317. IEEE Computer Society Press, Los Alamitos (1997)
18. Gu, Y., Soutchanski, M.: A logic for decidable reasoning about services. In: Proc. of AAAI-06 workshop on AI Driven Technologies for Services-Oriented Computing (2006)
19. Gu, Y., Soutchanski, M.: Decidable reasoning in a modified situation calculus. In: Proc. of the 12th Int'l Joint Conf. on Artificial Intelligence (IJCAI'07), pp. 1891–1897. Morgan Kaufmann Publishers, San Fransisco (2007)
20. Heinsohn, J., Kudenko, D., Nebel, B., Profitlich, H.J.: Rat—representation of actions using terminological logics. In: Proc. of the DFKI Workshop on Taxonomic Reasoning, Technical Report D-92-08, DFKI, Saarbrüken (1992)
21. Horrocks, I., Patel-Schneider, P.F., Harmelen, F.V.: From SHIQ and RDF to OWL: the making of a web ontology language. J. Web Semant. **1**(1), 7–26 (2003)
22. Horrocks, I., Sattler, U.: A tableaux decision procedure for SHOIQ. J. Autom. Reason. **39**(3), 249–276 (2007)
23. Liu, H., Lutz, C., Miličić, M., Wolter, F.: Updating description logic ABoxes. In: Proc. of the 10th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'06), pp. 46–56. AAAI Press, California (2006)
24. Liu, H., Lutz, C., Miličić, M., Wolter, F.: Reasoning about actions using description logics with general TBoxes. In: Proc. of the 10th European Conf. on Logics in Artificial Intelligence (JELIA'06), pp. 266–279. Springer (2006)
25. Liu, H.: Computing updates in description logics. Ph.D. thesis, Technischen Universität Dresden (2010)
26. Lutz, C., Sattler, U.: A proposal for describing services with DLs. In: Proc. of the 15th Int'l Workshop on Description Logics (DL'02) (2002)
27. Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D., Sirin, E., Srinivasan, N.: Bringing semantics to web services with OWL-S. World Wide Web J. **10**(3), 243–277 (2007)
28. McCarthy, J., Hayes, P.: Some philosophical problems form the standpoint of artificaial intelligence. Mach. Intell. **4**, 463–502 (1969)
29. McIlraith, S., Son, T., Zeng, H.: Semantic web services. IEEE Intell. Syst. **16**(2), 46–53 (2001)
30. Miličić, M.: Planning in action formalisms based on DLs: first results. In: Proc. of the 20th Int'l Workshop on Description Logics (DL'07), pp. 112–122 (2007)
31. Miličić, M: Complexity of planning in action formalisms based on description logics. In: Proc. of the 14th Int'l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07), pp. 408–422 (2007)
32. Myers, R., Pattinson, D., Schrŏder, L.: Coalgebraic hybrid logic. In: Proc. of the 12th Int'l Conf. on Foundations of Software Science and Computational Structures (FoSSaCS'09), pp. 137–151 (2009)
33. Narayanan, S., McIlraith, S.: Simulation, verification and automated composition of web services. In: Proc. of the 11th Int'l World Wide Web Conference (WWW'02), pp. 77–88 (2002)
34. Pacholski, L., Szwast, W., Tendera, L.: Complexity of two-variable logic with counting. In: Proc. of the 12th Annual IEEE Symposium on Logic in Computer Science (LICS'97), pp. 318–327. IEEE Computer Society Press, Los Alamitos (1997)
35. Pratt, V.R.: A practical decision method for propositional dynamic logic. In: Proc. of the 10th Annual ACM Symposium on Theory of Computing (STOC'78), pp. 326–337 (1978)
36. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge, MA (2001)
37. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. **48**(1), 1–26 (1991)
38. Shi, Z., Dong, M., Jiang, Y., Zhang, H.: A logic foundation for the semantic web. Sci. China Ser. F **48**(2), 161–178 (2005)
39. Smith, M.K., Welty, C., McGuinness D.L.: OWL web ontology language guide. W3C Recommendation, 10 Feb. 2004. http://www.w3.org/TR/owl-guide/. Accessed 3 Jan 2009
40. Thielscher, M.: From situation calculus to fluent calculus: state update axioms as a solution to the inferential frame problem. Artif. Intell. **111**(1–2), 277–299 (1999)
41. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. Artif. Intell. Res. **12**(1), 199–217 (2000)

42. Wolter, F., Zakharyaschev, M.: Satisfiability problem in description logics with modal operators. In: Proc. of the 6th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'98), pp. 512–523. Morgan Kaufmann Publishers, San Francisco (1998)
43. Wolter, F., Zakharyaschev, M.: Dynamic description logic. Adv. Modal Log. **2**, 449–463 (2000)
44. Zolin, E.: Complexity of reasoning in description logics. http://www.cs.man.ac.uk/~ezolin/dl/. Accessed 10 Jan 2010