Description Logics with Two Types of Definite Descriptions: Complexity, Expressiveness, and Automated Deduction

Michał Sochański*[1], Przemysław Andrzej Wałęga*[1,2], Michał Zawidzki*[1]

[1]Department of Logic, University of Łódź, Poland
[2]School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom
{michal.sochanski,przemyslaw.walega,michal.zawidzki}@filhist.uni.lodz.pl, p.walega@qmul.ac.uk

Abstract

Definite descriptions are expressions of the form "the unique x satisfying property C," which allow reference to objects through their distinguishing characteristics. They play a crucial role in ontology and query languages, offering an alternative to proper names (IDs), which lack semantic content and serve merely as placeholders.

In this paper, we introduce two extensions of the well-known description logic $\mathcal{ALC}$ with local and global definite descriptions, denoted $\mathcal{ALC}\iota_L$ and $\mathcal{ALC}\iota_G$, respectively. We define appropriate bisimulation notions for these logics, enabling an analysis of their expressiveness. We show that although both logics share the same tight ExpTime complexity bounds for concept and ontology satisfiability, $\mathcal{ALC}\iota_G$ is strictly more expressive than $\mathcal{ALC}\iota_L$. Moreover, we present tableau-based decision procedures for satisfiability in both logics, provide their implementation, and report on a series of experiments. The empirical results demonstrate the practical utility of the implementation and reveal interesting correlations between performance and structural properties of the input formulas.

1 Introduction

Definite descriptions (DDs)—expressions of the form "the unique x satisfying property C"—serve as complex terms identifying individuals via uniquely characterising properties. Their study originates in philosophical logic, where the main focus has been on semantics (Russell 1905; Pelletier and Linsky 2005; Hilbert and Bernays 1968; Rosser 1978; Lambert 2001). In recent decades, DDs have drawn renewed interest in formal logic, including classical, intuitionistic, and temporal systems (Fitting and Mendelsohn 2023; Indrzejczak 2023a,b; Indrzejczak and Kürbis 2023; Indrzejczak and Petrukhin 2024; Indrzejczak and Zawidzki 2021, 2023a,b; Kürbis 2019a,b; Kürbis 2025; Orlandelli 2021). This line of research has also led to reasoning systems supporting DDs, including KeYamera X (Bohrer, Fernández, and Platzer 2019), PROVER9 (Oppenheimer and and 2011), and Isabelle/HOL (Benzmüller and Scott 2020).

Of particular relevance is the application of DDs in *Knowledge Representation and Reasoning* (KRR), where they enable precise identification of individuals while encoding structural constraints—a capability surpassing that of non-descriptive names such as opaque IDs (Borgida, Toman, and Weddell 2016a,b, 2017; Toman and Weddell 2016, 2018, 2019a,b). Within *description logics* (DLs), this has motivated the introduction of DD operators into the logical language (Areces, Koller, and Striegnitz 2008; Ren, van Deemter, and Pan 2010; Neuhaus, Kutz, and Righetti 2020; Toman and Weddell 2019a). In particular, Artale et al. (2021) introduced concepts $\{\iota C\}$, denoting the singleton containing the unique individual satisfying concept C, or the empty set if no such individual exists. This allows for succinct representation of concepts such as:

$$\{\iota(\mathsf{building} \sqcap \forall \mathsf{tallThan}.\neg \mathsf{building})\},$$

which captures the expression "the tallest building." We refer to such constructs as *local* DDs.

We contrast *local* DDs with newly introduced *global* DDs of the form $\iota C.D$, expressing that "the individual satisfying C also satisfies D." For example, the concept

$$\iota(\mathsf{building} \sqcap \forall \mathsf{tallThan}.\neg \mathsf{building}).\exists \mathsf{locIn}.\{Dubai\}$$

formalises the statement "the tallest building is located in Dubai." This type of DDs builds on recent developments in modal and hybrid logics (Wałęga and Zawidzki 2023; Wałęga 2024; Indrzejczak and Zawidzki 2023a).

Despite growing interest in DDs within DLs, key foundational questions have remained open. In particular, the *computational complexity*, *expressive power*, and *reasoning procedures* for $\mathcal{ALC}$ extended with local or global DDs have not been systematically studied. While the complexity analysis proves relatively straightforward, characterising expressive power is significantly more challenging. Notably, devising suitable *bisimulations* for such logics is non-trivial. Existing bisimulations are for local DDs only and assume the presence of nominals and the universal role—features that simplify the task (Artale et al. 2021). To the best of our knowledge, bisimulations for $\mathcal{ALC}$ with DDs alone have not been proposed. Moreover, no existing DL reasoning system appears to support DDs, despite their practical motivation.

This paper aims to address these gaps. Our main contributions are as follows:

- We introduce three extensions of $\mathcal{ALC}$ with definite descriptions: $\mathcal{ALC}\iota_L$ (local), $\mathcal{ALC}\iota_G$ (global), and $\mathcal{ALC}\iota$ (both). These are formalised in Section 2.

*These authors contributed equally.

• To analyse expressivity, we define bisimulations for each logic in Section 3. They rely on novel, non-trivial conditions, which we show can be reduced to standard $\mathcal{ALC}$ bisimulation checks. We also provide algorithms for this reduction.

• Using these bisimulations, we study expressivity via equivalence-preserving concept translations. We show that $\mathcal{ALC}\iota_L$ is strictly less expressive than $\mathcal{ALC}\iota_G$, while $\mathcal{ALC}\iota_G$ and $\mathcal{ALC}\iota$ are equally expressive. All three are ExpTime-complete for concept and ontology satisfiability.

• In Section 4, we present tableau-based decision procedures for all three logics. Despite differing semantics of the two DD types, similar tableau rules suffice to handle both.

• We implement these procedures and evaluate them on custom benchmarks (Section 5). Results confirm their viability and show links between structural features of DDs and reasoning performance.

2 Description Logics with Definite Descriptions

Syntax Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite, pairwise disjoint sets of *atomic concept names*, *role names*, and *individual names*, respectively. $\mathcal{ALC}\iota$ *concepts* C are defined by the following grammar:

$$C \coloneqq A \mid \neg C \mid (C \sqcap C) \mid \exists r.C \mid \{\iota C\} \mid \iota C.C,$$

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. Standard abbreviations are used for other logical constructs: $\bot \coloneqq A \sqcap \neg A$, $\top \coloneqq \neg\bot$, $C \sqcup D \coloneqq \neg(\neg C \sqcap \neg D)$, and $\forall r.C \coloneqq \neg\exists r.\neg C$. An $\mathcal{ALC}\iota$ *concept inclusion* (CI) is a formula of the form $C \sqsubseteq D$, where C and D are $\mathcal{ALC}\iota$ concepts. We write $C \equiv D$ as shorthand for $C \sqsubseteq D$ and $D \sqsubseteq C$. An *assertion* is either $a : C$ or $r : (a_1, a_2)$, where $a, a_1, a_2 \in \mathsf{N_I}$, C is a concept, and $r \in \mathsf{N_R}$. An *ABox* $\mathcal{A}$ is a finite set of assertions; a *TBox* $\mathcal{T}$ is a finite set of concept inclusions. An *ontology* $\mathcal{O}$ consists of a TBox and an ABox. Where clear from context, we refer simply to *atomic concepts*, *roles* and *individuals* without mentioning names explicitly.

The description logic $\mathcal{ALC}\iota_L$ is the fragment of $\mathcal{ALC}\iota$ that includes *local definite descriptions* $\{\iota C\}$ but excludes *global definite descriptions* $\iota C.D$. Conversely, $\mathcal{ALC}\iota_G$ includes only global descriptions.

Semantics An *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty domain $\Delta^{\mathcal{I}}$ and a function that maps: (i) atomic concepts $A \in \mathsf{N_C}$ to subsets of $\Delta^{\mathcal{I}}$, (ii) roles $r \in \mathsf{N_R}$ to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, (iii) individuals $a \in \mathsf{N_I}$ to elements of $\Delta^{\mathcal{I}}$. This function extends to complex concepts:

$$(\neg C)^{\mathcal{I}} \coloneqq \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \qquad (C \sqcap D)^{\mathcal{I}} \coloneqq C^{\mathcal{I}} \cap D^{\mathcal{I}},$$

$$(\exists r.C)^{\mathcal{I}} \coloneqq \{d \in \Delta^{\mathcal{I}} \mid (d, e) \in r^{\mathcal{I}} \text{ for some } e \in C^{\mathcal{I}}\},$$

$$(\{\iota C\})^{\mathcal{I}} \coloneqq \begin{cases} \{d\}, & \text{if } C^{\mathcal{I}} = \{d\} \text{ for some } d \in \Delta^{\mathcal{I}}, \\ \emptyset, & \text{otherwise,} \end{cases}$$

$$(\iota C.D)^{\mathcal{I}} \coloneqq \begin{cases} \Delta^{\mathcal{I}}, & \text{if } C^{\mathcal{I}} = \{d\} \subseteq D^{\mathcal{I}} \text{ for some } d \in \Delta^{\mathcal{I}}, \\ \emptyset, & \text{otherwise.} \end{cases}$$

A concept C is *satisfied* in $\mathcal{I}$ if $C^{\mathcal{I}} \neq \emptyset$, and *satisfiable* if such $\mathcal{I}$ exists. A *pointed interpretation* is a pair $(\mathcal{I}, d)$ with $d \in \Delta^{\mathcal{I}}$. We write $(\mathcal{I}, d) \equiv_{\mathcal{L}} (\mathcal{J}, e)$ if for every concept C from a logic $\mathcal{L}$, we have $d \in C^{\mathcal{I}}$ iff $e \in C^{\mathcal{J}}$. Satisfaction of axioms of ontology $\mathcal{O}$ in $\mathcal{I}$, written $\mathcal{I} \models \alpha$, is defined as:

$$\mathcal{I} \models C \sqsubseteq D \quad \text{iff} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}},$$

$$\mathcal{I} \models a : C \quad \text{iff} \quad a^{\mathcal{I}} \in C^{\mathcal{I}},$$

$$\mathcal{I} \models r : (a_1, a_2) \quad \text{iff} \quad (a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in r^{\mathcal{I}}.$$

Interpretation $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ (written $\mathcal{I} \models \mathcal{O}$) if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{O}$. An ontology is *satisfiable* if it has a model, and a concept C is *satisfiable w.r.t. an ontology* $\mathcal{O}$ if C is satisfied in a model of $\mathcal{O}$.

We compare the expressive power of description logics via equivalence-preserving translations between their concepts. A logic $\mathcal{L}$ is *not more expressive* than $\mathcal{L}'$, denoted $\mathcal{L} \leq \mathcal{L}'$, if every $\mathcal{L}$-concept has an equivalent $\mathcal{L}'$-concept. It is *strictly less expressive* if $\mathcal{L} \leq \mathcal{L}'$ but $\mathcal{L}' \not\leq \mathcal{L}$, and *equally expressive* if both $\mathcal{L} \leq \mathcal{L}'$ and $\mathcal{L}' \leq \mathcal{L}$.

3 Expressiveness and Complexity

We analyse and compare the logics $\mathcal{ALC}\iota_L$, $\mathcal{ALC}\iota_G$, and $\mathcal{ALC}\iota$. Although their expressive power differs, all three have the same computational complexity.

Theorem 1. *In $\mathcal{ALC}\iota_L$, $\mathcal{ALC}\iota_G$, and $\mathcal{ALC}\iota$, both concept and ontology satisfiability are* ExpTime*-complete.*

Proof sketch. For upper bounds, we reduce $\mathcal{ALC}\iota$ ontology satisfiability to that of $\mathcal{ALCO}^{\iota}_{u}$, which is ExpTime-complete (Artale et al. 2021, Thm. 2). Local DDs $\{\iota C\}$ are allowed in $\mathcal{ALCO}^{\iota}_{u}$, and each global DD $\{\iota C.D\}$ is replaced with $\exists u.(\{\iota C\} \sqcap D)$, where u is the universal role. This polynomial reduction constructs an equivalent ontology.

For lower bounds, it suffices to show ExpTime-hardness for concept satisfiability in $\mathcal{ALC}\iota_L$ and $\mathcal{ALC}\iota_G$. We give logspace reductions from satisfiability of an $\mathcal{ALC}$ concept C w.r.t. a TBox $\mathcal{T}$, known to be ExpTime-complete (Baader 2003, Thm. 3.27). For $\mathcal{ALC}\iota_L$, we construct C' as:

$$C \sqcap \textstyle\bigsqcap_{(D \sqsubseteq E) \in \mathcal{T}} ((\neg D \sqcup E) \sqcap \{\iota\,(\neg(\neg D \sqcup E) \sqcup A_{D \sqsubseteq E})\}),$$

where each $A_{D \sqsubseteq E}$ is a fresh atomic concept. Then C' is satisfiable iff C is satisfiable w.r.t. $\mathcal{T}$. For $\mathcal{ALC}\iota_G$, we replace the above local definite descriptions with the following: $A_{D \sqsubseteq E} \sqcap \iota\,(\neg(\neg D \sqcup E) \sqcup A_{D \sqsubseteq E})\,.\top$. □

Despite having the same complexity, the logics differ in expressive power. We first observe that local descriptions can be encoded using global ones:

Proposition 2. *There is an exponential translation of $\mathcal{ALC}\iota_L$ concepts into equivalent $\mathcal{ALC}\iota_G$ concepts, and a polynomial translation of $\mathcal{ALC}\iota_L$ ontologies into conservative extensions in $\mathcal{ALC}\iota_G$.*

Proof sketch. The exponential translation replaces $\{\iota C\}$ with $C \sqcap \iota C.\top$. The polynomial version replaces $\{\iota C\}$ with $A_C \sqcap \iota A_C.\top$ and adds axioms $A_C \equiv C$. □

We conclude that $\mathcal{ALC}\iota_L \leq \mathcal{ALC}\iota_G = \mathcal{ALC}\iota$. Next, we will introduce bisimulations and show that $\mathcal{ALC}\iota_L < \mathcal{ALC}\iota_G$. For this, we will exploit the following notion of *names* of individuals in an interpretation:

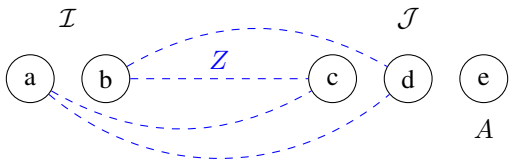

Figure 1: Maximal $\mathcal{ALC}\iota_L$ bisimulation between $\mathcal{I}$ and $\mathcal{J}$

Definition 3. *Let $\Delta' \subseteq \Delta^{\mathcal{I}}$. The set Names$(\Delta', \mathcal{I})$ consists of all $\mathcal{ALC}$ concepts C, with $C^{\mathcal{I}} = \{d\}$ for some $d \in \Delta'$.*

Example 4. *Let $\Delta^{\mathcal{I}} = \{a, b\}$ with $A^{\mathcal{I}} = \emptyset$, and $\Delta^{\mathcal{J}} = \{c, d, e\}$ with $A^{\mathcal{J}} = \{e\}$ (see Figure 1). Then Names$(\{a, b\}, \mathcal{I})$ = Names$(\{c, d\}, \mathcal{J}) = \emptyset$, since no $\mathcal{ALC}$ concept uniquely identifies an individual in either set. However, Names$(\{c, d, e\}, \mathcal{J})$ is non-empty, as it contains for example A, $\neg\neg A$, and $A \sqcup \exists r.\top$.*

We now define bisimulations for $\mathcal{ALC}\iota_L$ and $\mathcal{ALC}\iota_G$. Note that by Proposition 2, a separate definition for $\mathcal{ALC}\iota$ is unnecessary.

Definition 5. *An $\mathcal{ALC}\iota_L$ bisimulation between interpretations $\mathcal{I}$ and $\mathcal{J}$ is a relation $Z \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ such that $Z = \emptyset$ or, for all $(d, e) \in Z$, every atomic concept A, and role r:*

Atom *$d \in A^{\mathcal{I}}$ iff $e \in A^{\mathcal{J}}$,*

Forth *if $(d, d') \in r^{\mathcal{I}}$, then there exists e' such that $(e, e') \in r^{\mathcal{J}}$ and $(d', e') \in Z$,*

Back *if $(e, e') \in r^{\mathcal{J}}$, then there exists d' such that $(d, d') \in r^{\mathcal{I}}$ and $(d', e') \in Z$,*

Names$_L$ *Names$(Dom(Z), \mathcal{I})$ = Names$(Rng(Z), \mathcal{J})$.*

*An $\mathcal{ALC}\iota_G$ bisimulation is defined identically, except that **Names$_L$** is replaced with:*

Names$_G$ *Names$(\Delta^{\mathcal{I}}, \mathcal{I})$ = Names$(\Delta^{\mathcal{J}}, \mathcal{J})$.*

We write $(\mathcal{I}, d) \sim_{\mathcal{L}} (\mathcal{J}, e)$ if $(d, e) \in Z$ for some $\mathcal{L}$-bisimulation Z, where $\mathcal{L} \in \{\mathcal{ALC}\iota_L, \mathcal{ALC}\iota_G\}$.

Figure 1, presents the maximal $\mathcal{ALC}\iota_L$ bisimulation Z between $\mathcal{I}$ and $\mathcal{J}$ It satisfies **Names$_L$**, as Names$(\Delta^{\mathcal{I}}, \mathcal{I})$ = Names$(\Delta^{\mathcal{J}}, \mathcal{J}) = \emptyset$. However, there is no $\mathcal{ALC}\iota_G$ bisimulation betwee $\mathcal{I}$ and $\mathcal{J}$, as **Names$_G$** fails: $A \in$ Names$(\Delta^{\mathcal{J}}, \mathcal{J})$ but $A \notin$ Names$(\Delta^{\mathcal{I}}, \mathcal{I})$.

It is worth observing that our bisimulations differ from those used for $\mathcal{ALCO}^{\iota}_{u}$ by Artale et al. (2021), which rely on totality and "counting up to one" via the universal role and nominals. These conditions are too strong for $\mathcal{ALC}\iota_L$, as illustrated in Figure 1. While our **Names$_L$** and **Names$_G$** conditions are non-standard and appear to require quantification over all concepts, we show in Algorithms 1 and 2 how they can be verified procedurally. Before that, we prove that our bisimulations preserve concept satisfiability, and that the converse holds for ω-saturated[1] interpretations.

Theorem 6. *For all pointed interpretations $(\mathcal{I}, d)$ and $(\mathcal{J}, e)$, and both $\mathcal{L} \in \{\mathcal{ALC}\iota_L, \mathcal{ALC}\iota_G\}$ the following hold:*

1. *if $(\mathcal{I}, d) \sim_{\mathcal{L}} (\mathcal{J}, e)$, then $(\mathcal{I}, d) \equiv_{\mathcal{L}} (\mathcal{J}, e)$,*
2. *if $(\mathcal{I}, d) \equiv_{\mathcal{L}} (\mathcal{J}, e)$ and $\mathcal{I}, \mathcal{J}$ are ω-saturated, then $(\mathcal{I}, d) \sim_{\mathcal{L}} (\mathcal{J}, e)$.*

Proof sketch. We first prove Statement 1 by induction on the structure of $\mathcal{L}$-concepts C, showing $d \in C^{\mathcal{I}}$ if and only if $e \in C^{\mathcal{J}}$. If C is atomic, or of the form $\neg D$, $D \sqcap E$, or $\exists r.D$, the result follows from conditions **Atom**, **Forth**, and **Back**, as in the standard $\mathcal{ALC}$ case (Baader et al. 2017). If $C = \{\iota D\}$ and $d \in C^{\mathcal{I}}$, then $D^{\mathcal{I}} = \{d\}$, so $D \in$ Names$(Dom(Z), \mathcal{I})$. By the inductive hypothesis, $e \in D^{\mathcal{J}}$, and by **Names$_L$**, $D \in$ Names$(Rng(Z), \mathcal{J})$, hence $D^{\mathcal{J}} = \{e\}$, and $e \in (\{\iota D\})^{\mathcal{J}}$. The converse direction is analogous. If $C = \iota D.E$ and $d \in C^{\mathcal{I}}$, then $\{D, D \sqcap E\} \subseteq$ Names$(\Delta^{\mathcal{I}}, \mathcal{I})$. By **Names$_G$**, these concepts are also in Names$(\Delta^{\mathcal{J}}, \mathcal{J})$, implying $e \in (\iota D.E)^{\mathcal{J}}$. The converse again is similar.

For Statement 2, we begin with $\mathcal{L} = \mathcal{ALC}\iota_G$. Since $(\mathcal{I}, d) \equiv_{\mathcal{ALC}\iota_G} (\mathcal{J}, e)$ implies $(\mathcal{I}, d) \equiv_{\mathcal{ALC}} (\mathcal{J}, e)$, and $\mathcal{I}$, $\mathcal{J}$ are ω-saturated, standard results for $\mathcal{ALC}$ (Baader et al. 2017) imply that there exists an $\mathcal{ALC}$-bisimulation Z with $(d, e) \in Z$. To show that Z is an $\mathcal{ALC}\iota_G$-bisimulation, it remains to prove **Names$_G$**. If $C \in$ Names$(\Delta^{\mathcal{I}}, \mathcal{I})$, then $d \in (\iota C.\top)^{\mathcal{I}}$, hence $e \in (\iota C.\top)^{\mathcal{J}}$, so $C \in$ Names$(\Delta^{\mathcal{J}}, \mathcal{J})$. The reverse direction is symmetric. For $\mathcal{L} = \mathcal{ALC}\iota_L$, we define $Z := \{(k, l) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}} \mid (\mathcal{I}, k) \equiv_{\mathcal{ALC}\iota_L} (\mathcal{J}, l)\}$, so $(d, e) \in Z$. Conditions **Atom**, **Forth**, and **Back** hold by standard arguments for $\mathcal{ALC}$ bisimulations, extended with a translation of $\mathcal{ALC}\iota_L$ into first-order logic. To show **Names$_L$**, assume $C \in$ Names$(Dom(Z), \mathcal{I})$. Then some $d' \in Dom(Z)$ satisfies $d' \in (\{\iota C\})^{\mathcal{I}}$. Since $(d', e') \in Z$ for some $e' \in \Delta^{\mathcal{J}}$, we conclude $e' \in (\{\iota C\})^{\mathcal{J}}$, so $C \in$ Names$(Rng(Z), \mathcal{J})$. Thus, Names$(Dom(Z), \mathcal{I}) \subseteq$ Names$(Rng(Z), \mathcal{J})$. The reverse inclusion follows analogously, proving equality. □

We now apply our bisimulations to show that $\mathcal{ALC}\iota_L < \mathcal{ALC}\iota_G$. Combined with Proposition 2, this yields the following expressiveness result.

Theorem 7. *The following expressive power relations hold: $\mathcal{ALC}\iota_L < \mathcal{ALC}\iota_G = \mathcal{ALC}\iota$.*

Proof. By Proposition 2, it suffices to show $\mathcal{ALC}\iota_G \not\leq \mathcal{ALC}\iota_L$. Assume, for contradiction, that $\iota A.\top$ is equivalent to some $\mathcal{ALC}\iota_L$ concept C. Consider the interpretations $\mathcal{I}$ and $\mathcal{J}$ from Example 4, with the maximal $\mathcal{ALC}\iota_L$ bisimulation Z shown in Figure 1. Since $(\mathcal{I}, a) \sim_{\mathcal{ALC}\iota_L} (\mathcal{J}, c)$, Theorem 6 gives $a \in C^{\mathcal{I}}$ iff $c \in C^{\mathcal{J}}$. However, by construction, $\mathcal{I} \not\models \iota A.\top(a)$ and $\mathcal{J} \models \iota A.\top(c)$, i.e., $a \notin C^{\mathcal{I}}$ and $c \in C^{\mathcal{J}}$—a contradiction. □

While our bisimulations accurately characterise concept equivalence, we have not yet addressed how to decide bisimilarity between two pointed interpretations. We now present an algorithm for computing the maximal bisimulation between two finite interpretations, identifying all bisimilar pairs. The following notion plays a central role:

Definition 8. *Let $\Delta' \subseteq \Delta^{\mathcal{I}}$. The set NamedInd$(\Delta', \mathcal{I})$ of* named individuals *comprises all $d \in \Delta'$ such that $d \in C^{\mathcal{I}}$ for some $C \in$ Names$(\Delta', \mathcal{I})$.*

[1] See, e.g., Chang and Keisler (1992) for the definition.

For instance, in the interpretations $\mathcal{I}$ and $\mathcal{J}$ from Example 4, we have $\mathit{NamedInd}(\Delta^{\mathcal{I}}, \mathcal{I}) = \emptyset$ and $\mathit{NamedInd}(\Delta^{\mathcal{J}}, \mathcal{J}) = \{e\}$.

In contrast to computing names, identifying named individuals is straightforward: they are exactly those not $\mathcal{ALC}$-bisimilar to any other individual in the interpretation. The next theorem links named individuals to names, showing that once the named individuals are known, one can determine whether two interpretations have the same names. Below, we call relation Z *total* if it is both left- and right-total.

Theorem 9. *Let $\mathcal{I}$ and $\mathcal{J}$ be finite interpretations and let $\Delta' \subseteq \Delta^{\mathcal{I}}$ and $\Delta'' \subseteq \Delta^{\mathcal{J}}$. If $\mathit{Names}(\Delta', \mathcal{I}) \neq \emptyset \neq \mathit{Names}(\Delta'', \mathcal{J})$, then $\mathit{Names}(\Delta', \mathcal{I}) = \mathit{Names}(\Delta'', \mathcal{J})$ if and only if the maximal $\mathcal{ALC}$-bisimulation Z between $\mathcal{I}$ and $\mathcal{J}$ is a total relation and the restriction of Z to $\mathit{NamedInd}(\Delta', \mathcal{I}) \times \mathit{NamedInd}(\Delta'', \mathcal{J})$ is also total.*

Proof sketch. Assume that $\mathit{Names}(\Delta', \mathcal{I}) = \mathit{Names}(\Delta'', \mathcal{J}) \neq \emptyset$. To show that the restriction of Z to $\mathit{NamedInd}(\Delta', \mathcal{I}) \times \mathit{NamedInd}(\Delta'', \mathcal{J})$ is total, suppose for contradiction that some $d \in \mathit{NamedInd}(\Delta', \mathcal{I})$ has no Z-related counterpart in $\mathit{NamedInd}(\Delta'', \mathcal{J})$. Then there exists $C \in \mathit{Names}(\Delta', \mathcal{I})$ such that $C^{\mathcal{I}} = \{d\}$ and $C^{\mathcal{J}} = \{e\}$ for some $e \in \Delta''$. Since $(d, e) \notin Z$, there exists an $\mathcal{ALC}$ concept D with $d \in D^{\mathcal{I}}$ but $e \notin D^{\mathcal{J}}$, so $C \sqcap D \in \mathit{Names}(\Delta', \mathcal{I})$ but $C \sqcap D \notin \mathit{Names}(\Delta'', \mathcal{J})$—a contradiction. To show that Z is total, suppose some $d \in \Delta^{\mathcal{I}} \setminus \mathit{NamedInd}(\Delta', \mathcal{I})$ has no Z-related individual in $\Delta^{\mathcal{J}}$. As $\mathit{Names}(\Delta'', \mathcal{J}) \neq \emptyset$, there exists $e^* \in \mathit{NamedInd}(\Delta'', \mathcal{J})$ with $C^{\mathcal{J}} = \{e^*\}$ for some C. For each $e \in \Delta^{\mathcal{J}} \setminus \{e^*\}$, choose C_e with $e \in C_e^{\mathcal{J}}$ and $d \notin C_e^{\mathcal{I}}$, and let $D = C \sqcup \bigsqcap_{e \neq e^*} \neg C_e$. Then $D^{\mathcal{J}} = \{e^*\}$, so $D \in \mathit{Names}(\Delta'', \mathcal{J}) = \mathit{Names}(\Delta', \mathcal{I})$, and since $d \in D^{\mathcal{I}}$, we get $d \in \mathit{NamedInd}(\Delta', \mathcal{I})$—a contradiction.

Assume Z and its restriction are total. Suppose, for contradiction, that $C \in \mathit{Names}(\Delta', \mathcal{I})$ but $C \notin \mathit{Names}(\Delta'', \mathcal{J})$. Then $C^{\mathcal{I}} = \{d\}$ for some $d \in \mathit{NamedInd}(\Delta', \mathcal{I})$, and there exists $e \in \mathit{NamedInd}(\Delta'', \mathcal{J})$ with $(d, e) \in Z$ and $e \in C^{\mathcal{J}}$. Since $C \notin \mathit{Names}(\Delta'', \mathcal{J})$, there is $e' \neq e$ with $e' \in C^{\mathcal{J}}$. By totality, there exists $d' \in \Delta^{\mathcal{I}}$ with $(d', e') \in Z$. If $d' \neq d$, then $\{d, d'\} \subseteq C^{\mathcal{I}}$ contradicts uniqueness. If $d' = d$, and for $D \in \mathit{Names}(\Delta'', \mathcal{J})$, $D^{\mathcal{J}} = \{e\}$, then $d \in D^{\mathcal{I}}$ but $e \notin D^{\mathcal{J}}$, so $(d, e') \in Z$ contradicts that Z is a bisimulation. □

In Algorithm 1, we use Theorem 9 to compute the maximal $\mathcal{ALC}\iota$ bisimulation between finite interpretations $\mathcal{I}$ and $\mathcal{J}$. The algorithm first computes three maximal $\mathcal{ALC}$ bisimulations: Z, $Z_{\mathcal{I}}$, and $Z_{\mathcal{J}}$, which are used to determine the sets $N_{\mathcal{I}} = \mathit{NamedInd}(\Delta^{\mathcal{I}}, \mathcal{I})$ and $N_{\mathcal{J}} = \mathit{NamedInd}(\Delta^{\mathcal{J}}, \mathcal{J})$, as well as $N_{\mathcal{I}}^Z = \mathit{NamedInd}(\mathit{Dom}(Z), \mathcal{I})$ and $N_{\mathcal{J}}^Z = \mathit{NamedInd}(\mathit{Rng}(Z), \mathcal{J})$. If both $N_{\mathcal{I}}^Z$ and $N_{\mathcal{J}}^Z$ are empty, then $\mathbf{Names}_L$ holds and the algorithm returns Z. If only one is empty, then $\mathbf{Names}_L$ fails, and the algorithm returns $\emptyset$. Otherwise, Theorem 9 provides an equivalent condition for $\mathbf{Names}_L$: if it holds, the algorithm returns Z; otherwise, it returns $\emptyset$. For example, when applied to $\mathcal{I}$ and $\mathcal{J}$ from Figure 1, the algorithm returns the bisimulation Z shown therein.

Algorithm 1: Maximal $\mathcal{ALC}\iota_L$-bisimulation

Input: interpretations $\mathcal{I}$ and $\mathcal{J}$
Output: maximal $\mathcal{ALC}\iota_L$ bisimulation for $\mathcal{I}$ and $\mathcal{J}$

1 $Z \coloneqq \textsc{MaxBsimALC}(\mathcal{I}, \mathcal{J})$;
2 $Z_{\mathcal{I}} \coloneqq \textsc{MaxBsimALC}(\mathcal{I}, \mathcal{I})$;
3 $Z_{\mathcal{J}} \coloneqq \textsc{MaxBsimALC}(\mathcal{J}, \mathcal{J})$;
4 $N_{\mathcal{I}} \coloneqq \{d \in \Delta^{\mathcal{I}} \mid (d, e) \notin Z_{\mathcal{I}} \text{ for all } e \neq d \text{ in } \Delta^{\mathcal{I}}\}$;
5 $N_{\mathcal{J}} \coloneqq \{d \in \Delta^{\mathcal{J}} \mid (d, e) \notin Z_{\mathcal{J}} \text{ for all } e \neq d \text{ in } \Delta^{\mathcal{J}}\}$;
6 $N_{\mathcal{I}}^Z \coloneqq N_{\mathcal{I}} \cap \mathit{Dom}(Z)$;
7 $N_{\mathcal{J}}^Z \coloneqq N_{\mathcal{J}} \cap \mathit{Rng}(Z)$;
8 **if** $N_{\mathcal{I}}^Z = \emptyset = N_{\mathcal{J}}^Z$ **then return** Z;
9 **if** $N_{\mathcal{I}}^Z = \emptyset \neq N_{\mathcal{J}}^Z$ *or* $N_{\mathcal{I}}^Z \neq \emptyset = N_{\mathcal{J}}^Z$ **then return** $\emptyset$;
10 **if** Z *is total over* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ *and* $Z \cap (N_{\mathcal{I}}^Z \times N_{\mathcal{J}}^Z)$ *is total over* $N_{\mathcal{I}}^Z \times N_{\mathcal{J}}^Z$ **then return** Z;
11 **else return** $\emptyset$;

To compute the maximal $\mathcal{ALC}\iota$ bisimulation, we introduce Algorithm 2 which modifies Algorithm 1 by replacing Lines 6 and 7 with the following:

$$6\ N_{\mathcal{I}}^Z \coloneqq N_{\mathcal{I}}; \qquad 7\ N_{\mathcal{J}}^Z \coloneqq N_{\mathcal{J}};$$

We can show that both algorithms are correct:

Theorem 10. *Algorithms 1 and 2 return, respectively, the maximal $\mathcal{ALC}\iota_L$ and $\mathcal{ALC}\iota$ bisimulations for $\mathcal{I}$ and $\mathcal{J}$.*

Proof. The argumentation exploits Theorem 9, which provides the necessary condition for the bisimulation checks. What remains is to justify that $N_{\mathcal{I}} = \mathit{NamedInd}(\Delta^{\mathcal{I}}, \mathcal{I})$ and $N_{\mathcal{J}} = \mathit{NamedInd}(\Delta^{\mathcal{J}}, \mathcal{J})$. This follows from the Hennessy-Milner property for $\mathcal{ALC}$ bisimulations and the finiteness of $\mathcal{I}$ and $\mathcal{J}$. Specifically, if $d \in N_{\mathcal{I}}$, then for every $e \neq d$, there exists an $\mathcal{ALC}$ concept C_e such that $d \in C_e^{\mathcal{I}}$ and $e \notin C_e^{\mathcal{I}}$. Let $D = \bigsqcap_{e \neq d} C_e$; then $D^{\mathcal{I}} = \{d\}$, so $d \in \mathit{NamedInd}(\Delta^{\mathcal{I}}, \mathcal{I})$. Conversely, if $d \notin N_{\mathcal{I}}$, then there exists $e \neq d$ such that $(\mathcal{I}, d) \sim_{\mathcal{ALC}} (\mathcal{I}, e)$, and thus d satisfies exactly the same $\mathcal{ALC}$ concepts as e, implying $d \notin \mathit{NamedInd}(\Delta^{\mathcal{I}}, \mathcal{I})$. The argument for $N_{\mathcal{J}} = \mathit{NamedInd}(\Delta^{\mathcal{J}}, \mathcal{J})$ is analogous. □

4 Tableaux Systems

In this section, we present tableau calculi $\mathtt{TAB}_{\mathcal{ALC}\iota_L}$, $\mathtt{TAB}_{\mathcal{ALC}\iota_G}$, and $\mathtt{TAB}_{\mathcal{ALC}\iota}$, for all three logics. The first two share rules for common constructs, but differ in handling local and global definite descriptions, whereas $\mathtt{TAB}_{\mathcal{ALC}\iota}$ combines these systems.

Rules Given a concept C (optionally with ontology $\mathcal{O}$), our calculi return `sat` if C is satisfiable (w.r.t. $\mathcal{O}$), and `unsat` otherwise. Rules are applied to assertions to incrementally build a tableau tree. The root contains $a : C$, where a is fresh and does not occur in $\mathcal{O}$. A *branch* $\mathcal{B}$ is a path from root to leaf; if $\mathcal{B}'$ extends $\mathcal{B}$, we write $\mathcal{B} \subseteq \mathcal{B}'$. If it does not lead to confusion, we treat branches as sets of assertions. For an individual a and branch $\mathcal{B}$, the *theory* of a, written $\mathit{Th}_{\mathcal{B}}(a)$, is $\{C \mid a : C \in \mathcal{B}\}$; we say a *satisfies* C on

ABox rules:

$$(ABox_{\mathsf{l}})\ \frac{a : C \in ABox}{a : C} \qquad (ABox_r)\ \frac{r(a,a') \in ABox}{r(a,a')}$$

TBox rule: **Clash rule:**

$$(TBox)\ \frac{C \sqsubseteq D \in TBox, a : E}{a : \neg(C \sqcap \neg D)} \qquad (\bot)\ \frac{a : C, a : \neg C}{\bot}$$

Propositional rules:

$$(\neg\neg)\ \frac{a : \neg\neg C}{a : C} \qquad (\sqcap)\ \frac{a : C \sqcap D}{a : C, a : D} \qquad (\neg\sqcap)\ \frac{a : \neg(C \sqcap D)}{a : \neg C \mid a : \neg D}$$

Role rules:

$$(\exists r)\ \frac{a : \exists r.C}{b : C, r : (a,b)} \qquad (\neg\exists r)\ \frac{a : \neg\exists r.C, r : (a,a')}{a' : \neg C}$$

Global definite description rules:

$$(\iota_1^g)\ \frac{a : \iota C.D}{b : C, b : D} \qquad (\iota_2^g)\ \frac{a : \iota C.D, a' : C, a'' : C, a' : E}{a'' : E}$$

$$(\neg\iota^g)\ \frac{a : \neg\iota C.D, a' : E}{a' : \neg C \mid a' : \neg D \mid \begin{array}{c} b : C, b : A_C^g, \\ b' : C, b' : \neg A_C^g \end{array}} \qquad (cut_\iota^g)\ \frac{a : \iota C.D, a' : E}{a' : C \mid a' : \neg C}$$

Local definite description rules:

$$(\iota_1^\ell)\ \frac{a : \{\iota C\}}{a : C} \qquad (\iota_2^\ell)\ \frac{a : \{\iota C\}, a' : C, a'' : C, a' : D}{a'' : D}$$

$$(\neg\iota^\ell)\ \frac{a : \neg\{\iota C\}}{a : \neg C \mid \begin{array}{c} a : \neg A_C, \\ b : C, b : A_C \end{array}} \qquad (cut_\iota^\ell)\ \frac{a : \{\iota C\}, a' : D}{a' : C \mid a' : \neg C}$$

* b, b' occurring in the conclusion of a rule are fresh.

Figure 2: Rules of $\mathtt{TAB}_{\mathcal{ALC}\iota}$

$\mathcal{B}$ if $a : C \in \mathcal{B}$. A rule has the form $\frac{\mathcal{P}r}{\mathcal{C}on_1 | \cdots | \mathcal{C}on_m}$, with m the branching factor. Rules are *deterministic* if $m = 1$, and *branching* otherwise. The calculus's branching factor is the largest m among its rules. A rule (r) applies to premise $\mathcal{P}r$ if (i) it is not *blocked*, (ii) no conclusion already appears on the branch, and (iii) the branch is *open*. A branch is *closed* if a clash occurs; otherwise, it is open. It is *saturated* if open and no rules are applicable.

Figure 2 lists the rules of $\mathtt{TAB}_{\mathcal{ALC}\iota}$. Those for standard $\mathcal{ALC}$ constructs are omitted here. We highlight the handling of DDs and the blocking condition for $(\exists r)$.

Blocking. Rule $(\exists r)$ is blocked for $a : \exists r.C$ if:

($block_\exists$) There exists an individual a' on $\mathcal{B}$ such that $a' : C \in \mathcal{B}$ and $a' : \neg D \in \mathcal{B}$ for all D with $a : \neg\exists r.D \in \mathcal{B}$.

In that case, a' serves as a proxy r-successor of a. Blocking can be lifted if new $a : \neg\exists r.E$ assertions are added such that $a' : E \notin \mathcal{B}$. This is known as pattern-based blocking (Kaminski and Smolka 2009).

Global DDs. Rule (ι_1^g) introduces a fresh individual satisfying both C and D, unless an existing one already satisfies C, in which case only D is propagated (to the least such individual w.r.t. lexicographic order, if needed). Rule (ι_2^g) ensures uniqueness by merging the theories of any two individuals satisfying C. Rule $(\neg\iota^g)$ enforces non-uniqueness: for each individual a', at least one of the following holds: (i) $a' : \neg C$, (ii) $a' : \neg D$, or (iii) two distinct individuals satisfy C. In the third case, further applications to $a' : \neg\iota C.E$ are blocked. Rule (cut_ι^g) ensures decisiveness: every individual satisfies either C or $\neg C$ for any positive DD $\iota C.D$. The rule is crucial for the completeness of $\mathtt{TAB}_{\mathcal{ALC}\iota}$. For example, the tableau for the unsatisfiable concept $(\iota\neg(C \sqcap D).\top) \sqcap (\iota C.\neg D) \sqcap (\iota D.\neg C)$ fails to close without (cut_ι^g).

Local DDs. Local DDs are handled similarly, except for negation. While $\neg\iota C.D$ enforces non-uniqueness globally, $\neg\iota C$ does so locally, i.e., relative to the current individual. Rule $(\neg\iota^\ell)$ ensures either the current individual does not satisfy C or some other individual does. The right branch does not introduce new individuals when $(\neg\iota^\ell)$ is applied to $a' : \neg\{\iota C\}$—only $a' : \neg A_C$ is added in the right conclusion. In both negated-DD rules, A_C^g or A_C is a fresh atom (not in the input) determined by C.

Priorities and Confluence. Rules are applied in fixed order: $(\bot)$ has the highest, $(\exists)$ the lowest priority. As formulas are never removed, the calculus is *cumulative*; all rules are *invertible*, ensuring *confluence*: rule order may affect derivation length but not outcome.

Correctness We prove that $\mathtt{TAB}_{\mathcal{ALC}\iota}$—and so also $\mathtt{TAB}_{\mathcal{ALC}\iota_G}$ and $\mathtt{TAB}_{\mathcal{ALC}\iota_L}$—is sound, complete, and terminating. Recall that $\mathtt{TAB}_{\mathcal{ALC}\iota}$ is *sound* if, for any $\mathcal{ALC}\iota$-concept C (and ontology $\mathcal{O}$), it returns `unsat` only if C is unsatisfiable (w.r.t. $\mathcal{O}$). It is *complete* if, whenever it returns `sat`, C is indeed satisfiable (w.r.t. $\mathcal{O}$). To establish soundness, we show that rules of $\mathtt{TAB}_{\mathcal{ALC}\iota}$ *preserve satisfiability*:

Lemma 11. *Let* (r) *be a rule of* $\mathtt{TAB}_{\mathcal{ALC}\iota}$ *applied to a branch* $\mathcal{B}$*, and let* $\mathcal{B}_1, \ldots, \mathcal{B}_n \supseteq \mathcal{B}$ *be the resulting branches. If* $\mathcal{B}$ *is satisfiable, then so is some* $\mathcal{B}_i$ *for* $i \in \{1, \ldots, n\}$.

Proof sketch. The proof proceeds by case analysis on the rules. We illustrate it using the most complex case, $(\neg\iota^g)$. Let $\mathcal{P}r = \{a : \neg\iota C.D, a' : E\}$ be satisfiable. Then $a \in (\neg\iota C.D)^{\mathcal{I}}$ and $a' \in E^{\mathcal{I}}$, meaning no unique individual satisfies both C and D. We have three cases: (i) $a' \notin C^{\mathcal{I}}$, so $a' \in (\neg C)^{\mathcal{I}}$ and $\mathcal{P}r \cup \{a' : \neg C\}$ is satisfiable. (ii) $a' \notin D^{\mathcal{I}}$, so $a' \in (\neg D)^{\mathcal{I}}$ and $\mathcal{P}r \cup \{a' : \neg D\}$ is satisfiable. (iii) $a' \in C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and there exists $b' \in \Delta^{\mathcal{I}}$, $b' \neq a'$, such that $b' \in C^{\mathcal{I}}$. Extend $\mathcal{I}$ to $\mathcal{I}'$ so that $(A_C^g)^{\mathcal{I}'} = \{a'\}$. Then $a' \in (A_C^g)^{\mathcal{I}'}$, $b' \in (\neg A_C^g)^{\mathcal{I}'}$, hence $\mathcal{P}r \cup \{b : C, b : A_C^g, b' : C, b' : \neg A_C^g\}$ is satisfiable. □

Theorem 12. *For any* $\mathcal{ALC}\iota$*-concept* C *(and ontology* $\mathcal{O}$*), if* $\mathtt{TAB}_{\mathcal{ALC}\iota}$ *returns* `unsat`*, then* C *is unsatisfiable (w.r.t.* $\mathcal{O}$*).*

Proof. If $\mathtt{TAB}_{\mathcal{ALC}\iota}$ returns `unsat`, then all branches of the constructed tableau are closed—i.e., the clash rule has been applied to each, indicating unsatisfiability. Since every assertion in the tableau (except the root) results from rule applications, the contrapositive of Lemma 11 ensures that unsatisfiability propagates upward through the tableau, ultimately reaching $a : C$ at the root. This implies that C is unsatisfiable (w.r.t. $\mathcal{O}$ if ABox or TBox rules were applied). □

For completeness, we show that if $\mathtt{TAB}_{\mathcal{ALC}\iota}$ constructs a tableau with an open and saturated branch $\mathcal{B}$ for input concept C (and ontology $\mathcal{O}$), then C is satisfiable (w.r.t. $\mathcal{O}$),

as $\mathcal{B}$ provides sufficient information to construct a model $\mathcal{I}_\mathcal{B} = (\Delta^{\mathcal{I}_\mathcal{B}}, \cdot^{\mathcal{I}_\mathcal{B}})$ of C (and $\mathcal{O}$).

Let $\mathsf{DD}(\mathcal{B}) = \{C \mid a : \iota C.D \in \mathcal{B} \text{ or } a : \iota C \in \mathcal{B}\}$ denote the set of concepts that must have singleton extensions in $\mathcal{I}_\mathcal{B}$. For each individual a on $\mathcal{B}$, let the *representative of* a, $rep(a)$, be the least (w.r.t. lexicographic order) individual a' such that both a and a' satisfy some $C \in \mathsf{DD}(\mathcal{B})$ on $\mathcal{B}$, or a itself if no such C exists. If the application of $(\exists r)$ to an assertion $a : \exists r.C \in \mathcal{B}$ was blocked (and never unblocked), then any individual a' on $\mathcal{B}$ satisfying $\{C\} \cup \{\neg D \mid a : \neg\exists r.D \in \mathcal{B}\} \subseteq Th_\mathcal{B}(a')$ is called an (r, D)-*proxy successor* of a. We define $\mathcal{I}_\mathcal{B} = (\Delta^{\mathcal{I}_\mathcal{B}}, \cdot^{\mathcal{I}_\mathcal{B}})$ as follows:

- $\Delta^{\mathcal{I}_\mathcal{B}} = \{rep(a) \mid a \text{ occurs on } \mathcal{B}\}$,
- $a^{\mathcal{I}_\mathcal{B}} = rep(a)$ for each individual a on $\mathcal{B}$,
- $C^{\mathcal{I}_\mathcal{B}} = \{rep(a) \mid a : C \in \mathcal{B}\}$ for each concept C on $\mathcal{B}$,
- $r^{\mathcal{I}_\mathcal{B}} = \{(rep(a), rep(a')) \mid r : (a, a') \in \mathcal{B} \text{ or } a' \text{ is an } (r, D)\text{-proxy successor of } a \text{ for some } D\}$ for each role r.

Lemma 13. *For any assertion $a : C \in \mathcal{B}$, $rep(a) \in C^{\mathcal{I}_\mathcal{B}}$.*

Proof sketch. By structural induction on C. We illustrate the case $C = \exists r.D$. Assume $a : \exists r.D \in \mathcal{B}$. Since $\mathcal{B}$ is saturated, two cases arise: (i) If $(\exists r)$ was applied, then $\mathcal{B}$ contains $b : D$ and $r : (a, b)$. Thus $(rep(a), rep(b)) \in r^{\mathcal{I}_\mathcal{B}}$, and by the induction hypothesis, $rep(b) \in D^{\mathcal{I}_\mathcal{B}}$, hence $rep(a) \in (\exists r.D)^{\mathcal{I}_\mathcal{B}}$. (ii) If $(\exists r)$ was blocked, then ($block_\exists$) must be satisfied. Then there exists an (r, D)-proxy successor of a on $\mathcal{B}$, say a'. By definition, $a' : D \in \mathcal{B}$. Thus $(rep(a), rep(a')) \in r^{\mathcal{I}_\mathcal{B}}$ and by the induction hypothesis, $rep(a') \in D^{\mathcal{I}_\mathcal{B}}$, so $rep(a) \in (\exists r.D)^{\mathcal{I}_\mathcal{B}}$. □

Theorem 14. *If $\mathtt{TAB}_{\mathcal{ALC}\iota}$ returns `sat` for input C (and $\mathcal{O}$), then C is satisfiable (w.r.t. $\mathcal{O}$).*

Proof. A `sat` output implies the existence of an open, saturated branch $\mathcal{B}$. By Lemma 13, the constructed interpretation $\mathcal{I}_\mathcal{B}$ satisfies all $a : D \in \mathcal{B}$. Since $a : C \in \mathcal{B}$ for some a, we have $rep(a) \in C^{\mathcal{I}_\mathcal{B}}$, so C is satisfiable. If $\mathcal{O}$ is part of the input, saturation ensures all ABox rules were applied, so $\mathcal{I}_\mathcal{B}$ satisfies all ABox assertions. For TBox axioms $C \sqsubseteq D$, exhaustive application of $(TBox)$ and Lemma 13 assure that $rep(a) \in C^{\mathcal{I}_\mathcal{B}}$ implies $rep(a) \in D^{\mathcal{I}_\mathcal{B}}$ for each individual a on $\mathcal{B}$. Hence, $\mathcal{I}_\mathcal{B}$ satisfies $\mathcal{O}$, and C is satisfiable w.r.t. $\mathcal{O}$. □

To prove that $\mathtt{TAB}_{\mathcal{ALC}\iota}$ terminates, we show that for any input concept C (and ontology $\mathcal{O}$), the tableau $\mathcal{T}$ constructed by applying the $\mathtt{TAB}_{\mathcal{ALC}\iota}$ rules is finite. Each concept appearing in $\mathcal{T}$ is of one of the forms D, $\neg D$, A^g_E, $\neg A^g_E$, A_F, or $\neg A_F$, where D, $\iota E.G$ (for some concept G), and ιF are subconcepts of C or of concepts in $\mathcal{O}$. Hence, the number of distinct concepts in $\mathcal{T}$ is bounded by $4 \cdot |C|$ (or $4 \cdot (|C| + |\mathcal{O}|)$), where $|C|$ denotes the number of symbols in C (excluding parentheses), and $|\mathcal{O}|$ the total number of symbols in the concepts from $\mathcal{O}$ (also excluding parentheses). Consequently, each individual on a branch $\mathcal{B}$ of $\mathcal{T}$ can satisfy at most $4 \cdot |C|$ (or $4 \cdot (|C| + |\mathcal{O}|)$) distinct concepts.

Lemma 15. *Let $\mathcal{T}$ be a tableau for input C (and $\mathcal{O}$), and let $\mathcal{B}$ be a branch of $\mathcal{T}$. Then the number of distinct individuals on $\mathcal{B}$ is bounded by $2^{4\cdot|C|}$ (or $2^{4\cdot(|C|+|\mathcal{O}|)} + k$, where k is the number of individuals occurring in the ABox).*

Proof sketch. New individuals may be introduced by the rules $(ABox_1)$, $(ABox_r)$, $(\exists r)$, (ι^g_1), $(\neg\iota^g)$, and $(\neg\iota^\ell)$. Every non-root individual must be introduced by one of these rules. Define a function f that maps each individual a not introduced by any of the ABox rules to a set of concepts:

- $\{D\}$, if a is the root individual;
- $\{D, E\}$, if a is introduced by (ι^g_1) on $a' : \iota D.E$;
- $\{D, \pm A^g_D\}$, if introduced by $(\neg\iota^g)$ on $a' : \neg\iota D.E$ and $a : \pm A^g_D \in \mathcal{B}$;
- $\{D, A_D\}$, if introduced by $(\neg\iota^\ell)$ on $a' : \neg\{\iota D\}$;
- $\{D\} \cup \{\neg E \mid a' : \neg\exists r.E \in \mathcal{B} \text{ when } a \text{ was introduced}\}$, if introduced by $(\exists r)$ on $a' : \exists r.D$;

where $\pm A^g_E$ denotes either A^g_E or $\neg A^g_E$. By construction, $f(a) \subseteq Th_\mathcal{B}(a)$. Thus, the range of f is a subset of the power set of the set of concepts on $\mathcal{B}$, bounded by $2^{4\cdot|C|}$ (or $2^{4\cdot(|C|+|\mathcal{O}|)}$). We can show that f is injective, so the bound on the number of non-ABox individuals follows. Adding the k individuals from the ABox yields the claimed result. □

The bound in Lemma 15 allows us to prove termination:

Theorem 16. *The tableau system $\mathtt{TAB}_{\mathcal{ALC}\iota}$ is terminating.*

Proof. Let $\mathcal{T}$ be the tableau for C (and $\mathcal{O}$). Each branch $\mathcal{B}$ of $\mathcal{T}$ has at most $2^{4\cdot|C|}$ (or $2^{4\cdot(|C|+|\mathcal{O}|)} + k$, for k the number of individuals in ABox) individuals, each satisfying at most $4 \cdot |C|$ (or $4 \cdot (|C| + |\mathcal{O}|)$) concepts. Each rule application (except $(ABox_r)$) adds at least one concept to some individual's theory. $(ABox_r)$ applies at most as many times as there are role assertions in the ABox, say n. Thus the length of $\mathcal{B}$ is at most $4\cdot|C|\cdot 2^{4\cdot|C|}$ (or $4\cdot(|C|+|\mathcal{O}|)\cdot(2^{4\cdot(|C|+|\mathcal{O}|)}+k)+n$). As the branching factor of $\mathtt{TAB}_{\mathcal{ALC}\iota}$ is also finite, $\mathcal{T}$ is finite, so $\mathtt{TAB}_{\mathcal{ALC}\iota}$ terminates. □

We conclude this section with a corollary following from completeness and termination of $\mathtt{TAB}_{\mathcal{ALC}\iota}$:

Theorem 17. *Let C be a satisfiable $\mathcal{ALC}\iota$-concept. There exists a model $\mathcal{I}$ of C of size at most exponential in $|C|$.*

Proof. Let $\mathcal{T}$ be a tableau for input C. Since $\mathtt{TAB}_{\mathcal{ALC}\iota}$ is complete and C is satisfiable, $\mathcal{T}$ contains an open branch $\mathcal{B}$. Build an interpretation $\mathcal{I}_\mathcal{B} = (\Delta^{\mathcal{I}_\mathcal{B}}, \cdot^{\mathcal{I}_\mathcal{B}})$ as in the completeness proof. By Lemma 13, $rep(a) \in C^{\mathcal{I}_\mathcal{B}}$, where a is the root individual. The domain size is bounded by the number m of individuals on $\mathcal{B}$, which by Lemma 15 is exponential in the size of C (or C and $\mathcal{O}$). □

5 Implementation and Experiments

In this section, we describe our implementation of the three tableau calculi and present their evaluation. We have implemented all three calculi in a single prover written in Python. Our implementation takes as input an $\mathcal{ALC}\iota$ concept, possibly together with an ABox and a TBox, however our preliminary experiments consider satisfiability of concepts without ontologies. The prover parses the input concept using the Python library Lark, applies the tableau procedure, and outputs information about the satisfiability of the concept. The code and instructions for using the prover

No. DDs in a concept:		$0.1 \cdot k$	$0.3 \cdot k$	$0.5 \cdot k$
Global DDs	**runtime avg.:**	0.239s	0.750s	0.777s
	runtime std.dev.:	0.879s	2.16s	1.81s
	no. of time-outs:	21	31	42
Local DDs	**runtime avg.:**	0.371s	0.356s	0.411s
	runtime std.dev.:	1.31s	0.92s	1.59s
	no. of time-outs:	15	32	28

Table 1: Runtime for concepts with only GDs and only LDs; k represents the number of binary operators in a concept

are available on https://github.com/ExtenDD/two-types-of-DDs-AAAI-2026.

The primary aim of our experiment is to compare the efficiency of the prover depending on the amount of global descriptions (GDs) and local descriptions (LDs) in the input concepts. All the experiments were run on a machine with the processor AMD Ryzen 5 PRO 7540U and 16GB RAM under Windows 11.

Generator Our generator first builds a random binary syntax tree containing a predefined number of nodes, each corresponding to a subconcept; then, atomic concepts are randomly distributed among the leaves, binary operators (including GDs) among the inner nodes, and unary operators (including LDs) among all the nodes. The generator allows to customise the number of different atoms occurring in a concept, the number of GDs, LDs, and existential restrictions, as well as the chance for each subconcept to be preceded by negation.

Experiments and results In all experiments, the prover was run 5 times for each concept and the minimum was recorded as runtime, with the "time-out" limit set to 10s. The reported average runtimes do not consider the "time-outs", as well as concepts of the form $\neg\exists r.C$, which turn out satisfiable without applying any rule. To compare runtimes for GDs and LDs we generated six datasets, each with 150 concepts containing a random number of atoms between 10 and 200. First three datasets have concepts with GDs, but with no LDs. The latter three datasest have concepts with LDs, but with no GDs. For k being the number of binary operators in a concept, the first three datasets contain $0.1 \cdot k$, $0.3 \cdot k$, and $0.5 \cdot k$ GDs, respectively, whereas the next three datasets contain $0.1 \cdot k$, $0.3 \cdot k$, and $0.5 \cdot k$ LDs. Other parameters are kept the same in all datasets: the number of existential restrictions is 30% of the number of subconcepts, and the number of different atoms is 50% of the number of occurrences of atoms.

The runtimes for all six datasets are presented in Table 1. We do not include the time required for concept generation and for their parsing, as our focus was on the performance of the implemented tableau procedure.[2] Two observations can be made: runtime and the number of time-outs

[2]The runtime for parsing and concept generation grow linearly with concept size. For example, concepts with 100 atoms require approx. 0.003s to be generated and approx. 0.5s to parse and concepts with 200 atoms require 0.006s and 1s, respectively.

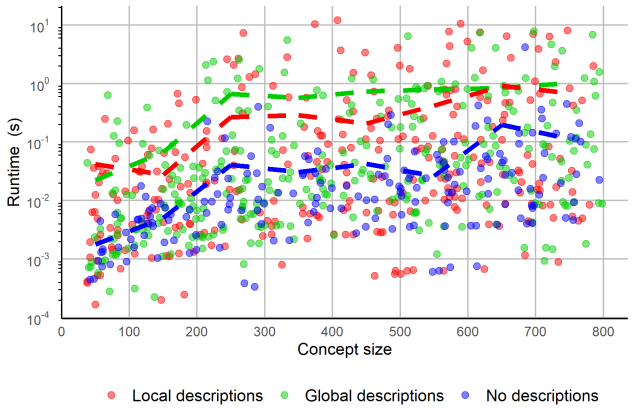

Figure 3: Runtimes for concepts with only GDs, only LDs, and without DDs; dashed lines represent average runtimes

are proportional to the number of descriptions in a concept (although this tendency is less visible for LDs), and greater for GDs than for LDs. The fact that GDs are more challenging aligns with our theoretical findings, whereas the fact that the growth of runtime is only linear, suggests practical feasibility of reasoning with DDs.

In the next experiment we analysed the scalability of the prover, by testing its performance on concepts of growing size. Recall that we identify the size of a concept with the number of symbols it uses, excluding parentheses. For this, we grouped the previously generated concepts into a set with LDs only and a set with GDs only. Additionally, we generated a set of 200 concepts without descriptions, leaving the other parameters the same as in the case of the other datasets. Results are depicted in Figure 3. We observe that in all three cases, the runtime seems to grow polynomially with concept size, even though computational complexity of the satisfiability problem is ExpTime-complete. This may be due to how we generate concepts or to insufficient data. Again, concepts with GDs have higher runtimes than those with LDs, whereas concepts without descriptions have clearly lower runtime, with only two time-outs across the whole dataset.

6 Conclusions

In this paper, we introduced three extensions of the standard description logic $\mathcal{ALC}$: $\mathcal{ALC}\iota_L$ with local definite descriptions $\{\iota C\}$, $\mathcal{ALC}\iota_G$ with global descriptions $\iota C.D$, and $\mathcal{ALC}\iota$, which supports both. We showed that all three logics are ExpTime-complete, but differ in expressive power: $\mathcal{ALC}\iota_L < \mathcal{ALC}\iota_G = \mathcal{ALC}\iota$. This expressiveness result is established via tailored bisimulations developed and analysed in the paper. We also proposed tableau-based decision procedures for all the logics and implemented them. Experimental results show that definite descriptions increase reasoning time, with global descriptions incurring a higher cost than local ones. Nonetheless, the overhead remains manageable, confirming the practical feasibility of our extensions. In future work, our aim is to optimise both the algorithms and their implementations.

Acknowledgements

This research is funded by the European Union (ERC, ExtenDD, project number: 101054714). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

Areces, C.; Koller, A.; and Striegnitz, K. 2008. Referring expressions as formulas of description logic. In *Proc. of INLG*, 42–49.

Artale, A.; Mazzullo, A.; Ozaki, A.; and Wolter, F. 2021. On Free Description Logics with Definite Descriptions. In *Proc. of KR*, 63–73.

Baader, F. 2003. *The description logic handbook: Theory, implementation and applications*. Cambridge: Cambridge University Press.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An introduction to description logic*. Cambridge University Press.

Benzmüller, C.; and Scott, D. S. 2020. Automating Free Logic in HOL, with an Experimental Application in Category Theory. *Journal of Automated Reasoning*, 64(1): 53–72.

Bohrer, R.; Fernández, M.; and Platzer, A. 2019. dL_ι: Definite Descriptions in Differential Dynamic Logic. In Fontaine, P., ed., *Proc. of CADE*, 94–110.

Borgida, A.; Toman, D.; and Weddell, G. 2016a. On referring expressions in information systems derived from conceptual modelling. In *Proc. of ER*, 183–197.

Borgida, A.; Toman, D.; and Weddell, G. 2016b. On referring expressions in query answering over first order knowledge bases. In *Proc. of KR*, 319–328.

Borgida, A.; Toman, D.; and Weddell, G. 2017. Concerning Referring Expressions in Query Answers. In *Proc. of IJCAI-17*, 4791–4795.

Chang, C. C.; and Keisler, H. J. 1992. *Model theory, Third Edition*, volume 73 of *Studies in logic and the foundations of mathematics*. North-Holland.

Fitting, M.; and Mendelsohn, R. L. 2023. *First-Order Modal Logic*, volume 480 of *Synthese Library*. Cham: Springer, 2 edition.

Hilbert, D.; and Bernays, P. 1968. *Grundlagen der Mathematik I*. Berlin, Heidelberg: Springer.

Indrzejczak, A. 2023a. Russellian Definite Description Theory – a Proof Theoretic Approach. *Review of Symbolic Logic*, 16(2): 624–649.

Indrzejczak, A. 2023b. Towards Proof-Theoretic Formulation of the General Theory of Term-Forming Operators. In *Proc. of TABLEAUX*, 131–149.

Indrzejczak, A.; and Kürbis, N. 2023. A Cut-Free, Sound and Complete Russellian Theory of Definite Descriptions. In *Proc. of TABLEAUX*, 112–130.

Indrzejczak, A.; and Petrukhin, Y. I. 2024. Bisequent Calculi for Neutral Free Logic with Definite Descriptions. In *Proc. of ARQNL*, 48–61.

Indrzejczak, A.; and Zawidzki, M. 2021. Tableaux for Free Logics with Descriptions. In *Proc. of TABLEAUX*, 56–73.

Indrzejczak, A.; and Zawidzki, M. 2023a. Definite descriptions and hybrid tense logic. *Synthese*, 202(3). Article number: 98.

Indrzejczak, A.; and Zawidzki, M. 2023b. When iota meets lambda. *Synthese*, 201(1). Article number: 72.

Kaminski, M.; and Smolka, G. 2009. Hybrid Tableaux for the Difference Modality. *Electronic Notes in Theoretical Computer Science*, 231: 241–257. Proc of. M4M5 2007.

Kürbis, N. 2019a. A Binary Quantifier for Definite Descriptions in Intuitionist Negative Free Logic: Natural Deduction and Normalisation. *Bulletin of the Section of Logic*, 48(2): 81–97.

Kürbis, N. 2019b. Two Treatments of Definite Descriptions in Intuitionist Negative Free Logic. *Bulletin of the Section of Logic*, 48(4): 299–317.

Kürbis, N. 2025. Normalisation for Negative Free Logics without and with Definite Descriptions. *Review of Symbolic Logic*, 18(1): 240–272.

Lambert, K. 2001. Free Logic and Definite Descriptions. In *New Essays in Free Logic*, volume 23, 37–48.

Neuhaus, F.; Kutz, O.; and Righetti, G. 2020. Free Description Logic for Ontologists. In *Proc. of JOWO*.

Oppenheimer, P. E.; and and, E. N. Z. 2011. A Computationally-Discovered Simplification of the Ontological Argument. *Australasian Journal of Philosophy*, 89(2): 333–349.

Orlandelli, E. 2021. Labelled calculi for quantified modal logics with definite descriptions. *Journal of Logic and Computation*, 31(3): 923–946.

Pelletier, F. J.; and Linsky, B. 2005. What is Frege's Theory of Descriptions. In *On Denoting: 1905–2005*, 195–250.

Ren, Y.; van Deemter, K.; and Pan, J. Z. 2010. Charting the Potential of Description Logic for the Generation of Referring Expressions. In *Proc. of INLG*, 115–123.

Rosser, J. B. 1978. *Logic for Mathematicians*. Dover: Dover Publications.

Russell, B. 1905. On denoting. *Mind*, 14(56): 479–493.

Toman, D.; and Weddell, G. 2016. Ontology Based Data Access with Referring Expressions for Logics with the Tree Model Property – (Extended Abstract). In *Proc. of AI*, 353–361.

Toman, D.; and Weddell, G. 2018. Identity Resolution in Conjunctive Querying over DL-Based Knowledge Bases. In *Proc. of DL*, 1–12.

Toman, D.; and Weddell, G. 2019a. Finding ALL answers to OBDA queries using referring expressions. In *Proc. of AI*, 117–129.

Toman, D.; and Weddell, G. 2019b. Identity resolution in ontology based data access to structured data sources. In *Proc. of PRICAI*, 473–485.

Wałęga, P. A. 2024. Expressive Power of Definite Descriptions in Modal Logics. In *Proc. of KR*, 687–696.

Wałęga, P. A.; and Zawidzki, M. 2023. Hybrid Modal Operators for Definite Descriptions. In *Proc. of JELIA*, 712–726.