A Tableau System for First-Order Logic with Standard Names

Jens Claßen[0000−0002−0395−1907] and Torben Braüner[0000−0003−4582−1702]

Institute for People and Technology, Roskilde University, Denmark
{classen,torben}@ruc.dk

Abstract. Levesque and Lakemeyer proposed a logic called $\mathcal{L}$ as a first-order logic for knowledge representation and reasoning in knowledge-based systems. A characteristic feature of this logic is that it uses a countably infinite set of what are called standard names, which are syntactically treated like constants, but which are also isomorphic to a fixed universe of discourse. Quantifiers in $\mathcal{L}$ are then given a substitutional interpretation. This non-standard semantics not only simplifies the proofs for certain meta-theoretic properties, but is also exploited in dedicated reasoning procedures for modal extensions of $\mathcal{L}$ that include notions of belief, actions, time, and more. However, the only sound and complete proof system provided for $\mathcal{L}$ so far is a Hilbert-style axiom system, as well as an iterative reasoning mechanism based on resolution and clause subsumption. In this paper, we present a tableau system for $\mathcal{L}$, and show its soundness and completeness. Completeness is proved first by reduction to the existing axiom system, and involves the cut rule, and then via Hintikka sets, which does not require the cut rule.

Keywords: Tableau · First-Order Logic · Standard Names · Substitutional Quantification

1 Introduction

The logic $\mathcal{L}$ has been developed and studied by Levesque and Lakemeyer (LL henceforth) as a formalism for knowledge-based systems (their book [21] gives a formal introduction and an overview of recent developments). It is a non-standard first-order logic where so-called *standard names* serve a dual purpose: Syntactically they are treated as a (countable infinite) supply of constants, while semantically they also constitute the fixed universe of discourse, where every name is interpreted by itself. Thus, standard names provide a well-defined notion of the identity of an individual, similar to unique object identifiers in database management. $\mathcal{L}$ forms the basis for a range of modal logics that extend it with notions of belief [21], non-monotonic inference [18], actions [17], programs and temporal specifications [6], and combinations thereof [7], to name but a few. Typically, this involves a type of possible-world semantics where standard names can be viewed as rigid designators in the sense of Kripke [16]. LL argue that this assumption is beneficial not only in terms of simplifying the proofs of certain

meta-theoretical properties, but also for defining specific reasoning procedures. For example, the logic $\mathcal{OL}$ [21] adds modalities for knowledge and "only-knowing". The question of whether only-knowing a knowledge base (KB) κ entails knowing a certain query α can then be reduced to logical entailment in the non-modal fragment $\mathcal{L}$, where quantified subformulas are handled by considering all standard names mentioned in κ and α, plus one extra name to represent all unmentioned names. In a similar way, $\mathcal{L}$ serves as a base logic in an ongoing line of research that is concerned with planning [5], verification [29], and synthesis [11].

Interestingly, to the best of our knowledge, so far there is no implementation of a sound and complete reasoner for $\mathcal{L}$. Moreover, the only sound and complete proof system that has been provided is a Hilbert-style axiom system, which is not suitable for actual reasoning. What comes closest is an implementation of a tractable, yet incomplete reasoner studied in [25], as well as a formal description of a sound and complete reasoning mechanism based on resolution presented in [19]. In principle, the latter could serve as the basis for an implementation, but arguably the method is not analytic. In particular, it involves guessing substitutions for free variables in query clauses, and then testing for subsumption against KB clauses, which is impractical.

In the present paper, we propose an alternative, tableau-based proof system for $\mathcal{L}$ that allows reasoning in a systematic way. To the best of our knowledge, no analytic proof system in the form of tableau-, sequent- or natural deduction systems for $\mathcal{L}$ has been published until now. We show our system to be sound, and give two different completeness results: The first one is via a reduction to LL's axiom system as presented in [21], and requires including the cut rule in the system. It is obviously desirable to avoid the cut rule since it blatantly violates analyticity, and at a more conceptual level, it violates a fundamental idea behind tableau systems, namely that tableau rules break down formulas into smaller formulas. Hence, the result is more of theoretical interest, as it explicates the relation between our new system and LL's existing proof theory. Our second completeness result goes via traditional Hintikka sets, and does not require the cut rule, so lends itself more towards an implementation.

The remainder of this paper is organized as follows. The following section introduces $\mathcal{L}$ formally and presents LL's axiom system for it. In Section 3, we then present our tableau system for $\mathcal{L}$ and show its soundness. Afterwards, in Section 4, we provide our two completeness results, one wrt. the axiom system and using cuts, and one based on Hintikka sets and not requiring cuts. The paper finishes with a discussion and concluding remarks.

2 The Logic $\mathcal{L}$

In this section, we give brief formal description of $\mathcal{L}$ and its proof theory, based on the presentation in [21].

2.1 Syntax

In terms of syntax, $\mathcal{L}$ is very similar to the standard first-order predicate calculus with equality and function symbols, but with the addition of standard names that can be used in the same way as constant symbols. Formally:

Definition 1 (Vocabulary). *Formulas are built from a set of symbols consisting of:*

1. *a countably infinite supply of* variables*, written as* x, y, z*;*
2. *a countably infinite supply of* standard names $\#1, \#2, \ldots$*, written schematically as* n, m*;*
3. *countably many* predicate symbols $P, Q, R, \ldots$ *with associated arity* $k \geq 0$*;*
4. *countably many* function symbols $f, g, h, \ldots$ *with associated arity* $k \geq 0$*;*
5. equality $=$ *and* logical connectives $\exists$, $\vee$, $\neg$.

Definition 2 (Terms). *The* terms *are the smallest set such that*

1. *every variable is a term;*
2. *every standard name is a term;*
3. *if* f *is a* k*-ary function symbol and* $t_1, \ldots, t_k$ *are terms, then* $f(t_1, \ldots, t_k)$ *is a term.*

A term without variables is a ground term*, and a ground term with exactly one function symbol is a* primitive term*.*

Definition 3 (Formulas). *The* formulas *are the smallest set such that*

1. *if* P *is a* k*-ary predicate symbol and* $t_1, \ldots, t_k$ *are terms, then* $P(t_1, \ldots, t_k)$ *is an (atomic) formula;*
2. *if* t_1, t_2 *are terms, then* $(t_1 = t_2)$ *is a formula;*
3. *if* ϕ, ψ *are formulas, then so are* $\neg\phi$ *and* $\phi \vee \psi$*;*
4. *if* x *is a variable and* ϕ *a formula, then* $\exists x\phi$ *is a formula.*

We treat $\phi \wedge \psi$*,* $\phi \supset \psi$*,* $\phi \equiv \psi$*, and* $\forall x\phi$ *as the usual abbreviations. The notions of bound and free variables as well as their scope are defined as usual. By* ϕ_t^x *we denote the result of simultaneously replacing all free occurrences of variable* x *by term* t*. A* sentence *is a formula without free variables. A* ground atom *is an atomic formula with no variables. A* primitive atom *is ground atom where every* t_i *is a standard name.*

2.2 Semantics

The only non-standard concepts introduced above are those of primitive terms and primitive atoms. They are used in the definition of a model-theoretic semantics as given below. Intuitively, an interpretation (called a *world*) is uniquely defined by means of which primitive atoms $P(n_1, \ldots, n_k)$ are true in it, and to which standard names it maps each primitive term $f(n_1, \ldots, n_k)$. Standard names n_i are always interpreted by themselves, and complex terms and formulas are then evaluated recursively:

Definition 4 (Semantics). *A* world w *is a mapping from primitive terms to standard names, and from primitive atoms to truth values* $\{0,1\}$*. Let* $\mathcal{W}$ *be the set of all worlds. The* value of a ground term t at a world w*, written as* $w(t)$*, is defined by*

1. $w(n) = n$ *for every standard name* n*;*
2. $w(f(t_1,\ldots,t_k)) = w[f(n_1,\ldots,n_k)]$*, where* $n_i = w(t_i)$*.*

The truth of a sentence ϕ in a world w*, written* $w \models \phi$*, is then given by*

1. $w \models P(t_1,\ldots,t_k)$ *iff* $w[P(n_1,\ldots,n_k)] = 1$*, where* $n_i = w(t_i)$*;*
2. $w \models (t_1 = t_2)$ *iff* $w(t_1)$ *is the same name as* $w(t_2)$*;*
3. $w \models \neg\phi$ *iff it is not the case that* $w \models \phi$*;*
4. $w \models \phi \vee \psi$ *iff* $w \models \phi$ *or* $w \models \psi$*;*
5. $w \models \exists x\phi$ *iff for some name* n*,* $w \models \phi^x_n$*.*

The notions of validity, satisfiability, and logical entailment are defined as usual.

As mentioned previously, epistemic extensions of $\mathcal{L}$ such as $\mathcal{OL}$ employ possible-world semantics, where an epistemic state is defined as a set of worlds, and a formula is known/believed if it holds in all accessible worlds. While the interpretation of $\mathcal{L}$ formulas does not require such epistemic states, but only single interpretations, we use the term "world" for an interpretation here to keep our terminology consistent with the existing literature.

At least two things here are noteworthy. First, from the second clause for sentences above it can be seen that equality is a built-in predicate of $\mathcal{L}$ with a fixed meaning, namely identity over co-referring standard names. Furthermore, the fifth clause defines quantification to be interpreted substitutionally: A quantified formula such as $\forall x P(x)$ is true in w just in case all of $P(\#1)$, $P(\#2)$,... are true in it. As a consequence, there is no need for valuations or variable maps, but, as we will see, properties about sentences can often be proved by a simple induction over their structure. In such a proof, the case for a quantified sentence $\exists x\phi$ can be reduced to that of its instances ϕ^x_n, i.e., sentences of smaller size.

2.3 Axiom System for $\mathcal{L}$

The Hilbert-style axiom system for $\mathcal{L}$ presented by LL is shown in Figure 1. Again, the two most unusual aspects relate to standard names: Axiom 6 formalizes equality as identity over standard names. Moreover, the rule for Universal Generalization only requires finitely many instances. Here the intuition is that if we can prove α^x_n for *every* standard name n, it is obviously sound to infer $\forall x\alpha$. However, α can only mention finitely many standard names $n_1,\ldots,n_{k-1}$. If we have a proof for every such name, together with one more proof for a single name n_k that is *not* mentioned in α, that is enough because the proof for any other unmentioned name n' can be the same as the one for n_k, just with all occurrences of n_k replaced by n'. The latter works because standard names have no internal structure, except of being distinct from one another. To illustrate these principles, Figure 2 shows an example axiomatic proof for the substitutivity property for functions. We note:

Theorem 1 ([21]). *The axiom system for* $\mathcal{L}$ *is sound and complete.*

Axioms:

1. $\alpha \supset (\beta \supset \alpha)$
2. $(\alpha \supset (\beta \supset \gamma)) \supset ((\alpha \supset \beta) \supset (\alpha \supset \gamma))$
3. $(\neg\beta \supset \neg\alpha) \supset ((\neg\beta \supset \alpha) \supset \beta)$
4. $\forall x(\alpha \supset \beta) \supset (\alpha \supset \forall x\beta)$, provided that x does not occur freely in α
5. $\forall x\alpha \supset \alpha^x_t$
6. $(n = n) \wedge (n \neq m)$ for any distinct n, m

Rules:

1. From α and $\alpha \supset \beta$, infer β (MP).
2. From $\alpha^x_{n_1}, \ldots, \alpha^x_{n_k}$, infer $\forall x\alpha$, provided the n_i range over all names in α and at least one not in α (UG).

Fig. 1. The axiom system for $\mathcal{L}$ [21].

1. $\#1 = \#1$	Ax
2. $\forall x(x = x)$	UG
3. $f(\#1) = f(\#1)$	MP
4. $\#1 = \#1 \supset f(\#1) = f(\#1)$	MP
5. $\#1 \neq \#2$	Ax
6. $\#1 = \#2 \supset f(\#1) = f(\#2)$	MP
7. $\forall y(\#1 = y \supset f(\#1) = f(y))$	UG
8. $\forall x \forall y(x = y \supset f(x) = f(y))$	UG

Fig. 2. Axiomatic proof of the substitutivity property for functions [21].

3 Sentence Tableaux for $\mathcal{L}$

Drawing inspiration from the axiom system for $\mathcal{L}$, we devised a tableau proof system as depicted in Figure 3. The first row are standard rules for the Boolean connectives. The second row contains rules for quantification, where the one for existential quantification is non-standard and incorporates the same idea underlying the Universal Generalization inference rule: In addition to considering all standard names that are mentioned on a branch, it suffices to include one additional unmentioned name as representative for all other unmentioned names. The third row lists rules for handling terms that include functions, one for making a case distinction over their possible values (again with the "trick" of using an extra name), and one for substituting a term for its value. The fourth row contains the standard rule for closing a branch with two complementary formulas, the standard rule for closing a branch using inequality, and a non-standard rule for closing a branch based on equality over standard names.

Figure 4 shows an example proof for the substitutivity property for functions to illustrate how existential quantifiers and equality are handled. As for dealing with terms using the (TCut) and (TSub) rules, Figure 5 depicts a tableau for

the set $\{(a = b), P(a, a), \neg P(b, b)\}$, also used in [2] to demonstrate how equality and terms are dealt with in Jeffrey's [12] and Reeves' [23] systems. Intuitively, our system avoids some of the problems that these early systems had in relation to symmetries and unrestricted substitution, as it deconstructs nested terms in a systematic fashion and only uses substitution to remove function symbols. Arguably, our approach aligns more with the ideal of a tableau proof being analytic.

$$\frac{\neg(\phi \vee \psi)}{\neg\phi, \neg\psi}\ (\neg\vee) \qquad \frac{\neg\neg\phi}{\phi}\ (\neg\neg) \qquad \frac{\phi \vee \psi}{\phi \mid \psi}\ (\vee)$$

$$\frac{\exists x\alpha}{\alpha^x_{n_1} \mid \cdots \mid \alpha^x_{n_k}}\ (\exists) \qquad \frac{\neg\exists x\alpha}{\neg\alpha^x_t}\ (\neg\exists)$$

$$\frac{}{(t = n_1) \mid \cdots \mid (t = n_k)}\ (\text{TCut}) \qquad \frac{\phi[t], (t = n)}{\phi[n]}\ (\text{TSub})$$

$$\frac{\phi, \neg\phi}{*}\ (\bot) \qquad \frac{\neg(t = t)}{*}\ (\neq) \qquad \frac{(n = m)}{*}\ (=)$$

Fig. 3. Tableau rules for $\mathcal{L}$. In rules ($\exists$) and (TCut), $n_1, \ldots, n_k$ range over all standard names in the branch, plus *one extra*. In rules ($\neg\exists$), (TSub), and ($\neq$), t can be *any* ground term. In rule (TSub), the notation $\phi[t]$ is used to express that ϕ mentions ground term t, and $\phi[n]$ then means ϕ with every occurrence of t simultaneously replaced by n. In rule (TCut), t can be any ground term occurring on the branch. In rule ($=$), n and m are *distinct* names.

3.1 Soundness

Theorem 2 (Soundness). *The system shown in Figure 3 is sound, i.e., if a sentence ϕ is satisfiable, then every tableau for ϕ has an open branch.*

Proof. As usual, we identify a branch in a tableau with the set of formulas it contains. Clearly, none of the closure rules are applicable in a satisfiable branch. We prove by induction that the expansion rules preserve the property that when being applied to a satisfiable branch, at least one of the resulting branches is satisfiable. Let B be a satisfiable branch. The claim is obvious for rules ($\neg\vee$), ($\neg\neg$), and ($\vee$), and easy to show for rules ($\neg\exists$) and (TSub). The most interesting cases are rules ($\exists$) and (TCut), whose proofs work similarly, so we only show the one for ($\exists$) below. Let $w \models \exists x\alpha$. By definition, this means there is a name n such that $w \models \alpha^x_n$. If n is any of the names $n_1, \ldots, n_k$, then by assumption $w \models B \cup \{\alpha^x_n\}$, and we are done. In case that n is a name that is *not* mentioned

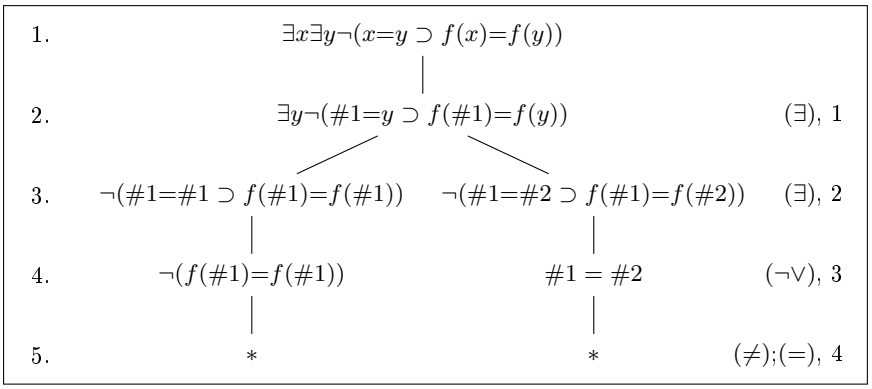

Fig. 4. Tableau proof of the substitutivity property for functions.

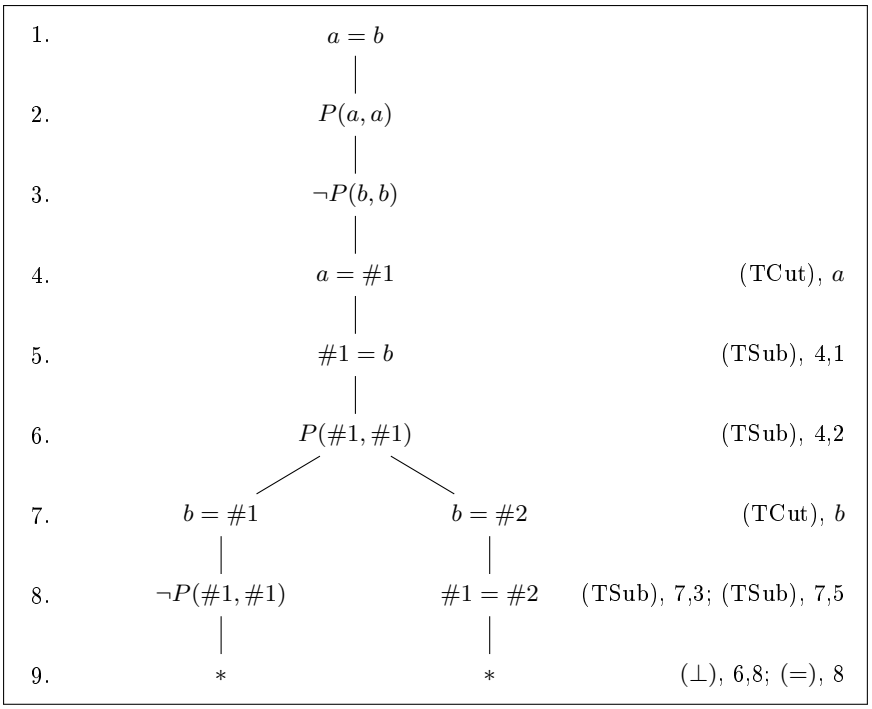

Fig. 5. Tableau proof for $\{(a = b), P(a, a), \neg P(b, b)\}$.

in B *and* distinct from the extra name (let's call it n'), then let $*$ be a bijection from standard names to standard names that swaps n with n' and leaves all other names unchanged. Similar as in the proof for Theorem 2.8.8 in [21], let w^* be the world such that $w^*[\alpha] = w[\alpha^*]$ for every primitive formula α, and $w^*[t] = w[t^*]$ for every primitive term t, where α^* and t^* denote the results of simultaneously replacing every name by its mapping under $*$ in α and t, respectively. It follows by induction that for any formula ϕ, $w^* \models \phi$ iff $w \models \phi^*$. Since B does not mention n', $B^* = B$. Hence $w^* \models B \cup \{\alpha^x_{n'}\}$.

The proof above uses a bijection to formalize the following intuition: Because standard names are interchangeable in the sense that their only internal property is being distinct from one another, one "extra" name used in the ($\exists$) and (TCut) rules is enough as it serves as a placeholder (representative) for all other unmentioned names.

4 Completeness

As mentioned earlier, we give two different completeness results: The first completeness result is via a reduction to LL's axiom system as presented in [21], and requires including the cut rule in the system. The second completeness result goes via traditional Hintikka sets, and does not require the cut rule. The two are independent from another in the sense that we do not need the first result in the completeness prove for the Cut-free system, but present it merely to explore the relation of our new system to LL's original axiom system.

4.1 Completeness via reduction to an axiom system (requiring cuts)

We take it that most readers of the present paper are familiar with the cut rule, but to make the paper self-contained, we display it anyway:

$$\frac{}{\phi \mid \neg\phi}\ (\text{Cut})$$

So at any stage in the tableau construction, a branch can be split into two branches, where an arbitrary formula ϕ is added to one of the branches and the negated formula $\neg\phi$ is added to the other. The cut rule is usually considered undesirable, one reason being that it blatantly violates the idea that tableau rules break down formulas into smaller formulas, but the rule also has its advocates, see for example the paper [8]. See also the discussion of cuts in tableau systems in [9], pages 107–108.

Theorem 3 (Completeness). *The system comprised of the rules shown in Figures 3 together with the rule (Cut) is complete, i.e., if a sentence ϕ is unsatisfiable, then there is a tableau for ϕ all of whose branches are closed.*

Proof. The proof is by means of the axiom system shown in Figure 1. Let ϕ be unsatisfiable. Then $\neg\phi$ is valid, and therefore there exists a derivation of $\neg\phi$ using

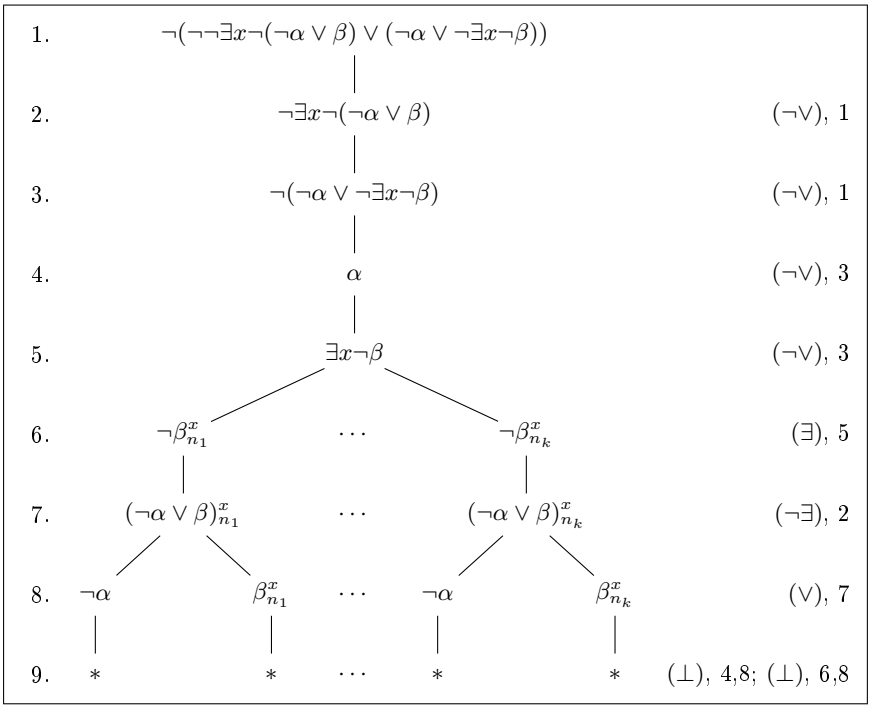

Fig. 6. Tableau proof for the negation of Axiom 4. Applications of the $(\neg\neg)$ rule are not shown. Step 8 exploits the fact that α does not contain x, hence $(\neg\alpha)^x_n = \neg\alpha$.

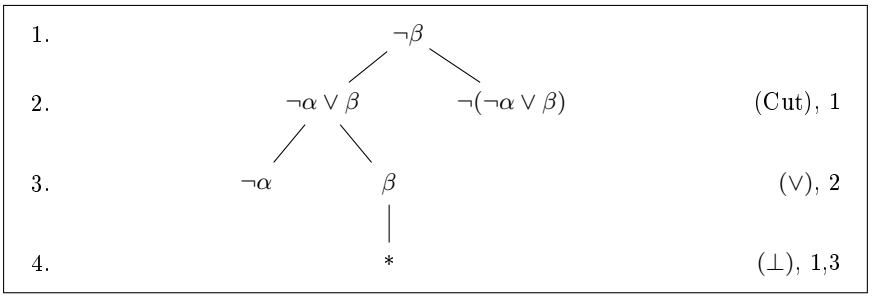

Fig. 7. Tableau construction corresponding to the (MP) rule.

the axioms and rules of Figure 1, cf. [21]. We show by induction on the structure of the derivation that for every derivable formula ψ, there is a corresponding tableau for $\neg\psi$, all of whose branches are closed.

For the base case, we show that this claim is true for the axioms. Axioms 1–3 represent propositional reasoning, and the corresponding tableaux are straightforward, using only the propositional rules $(\neg\vee)$, $(\neg\neg)$, $(\vee)$, and $(\bot)$. Axioms 5

and 6 are similarly easy, requiring an additional application of rules ($\neg\exists$) as well as ($\neq$) and ($=$), respectively. The tableau corresponding to Axiom 4 is shown in Figure 6.

For the inductive case, we show that the claim is preserved under the application of the (MP) and (UG) rules. In case of (MP), assume that there are closed tableaux for $\neg\alpha$ and $\neg(\neg\alpha \vee \beta)$. Then a closed tableau for $\neg\beta$ can be constructed as shown in Figure 7, where we can "glue" the given tableaux underneath the $\neg\alpha$ and $\neg(\neg\alpha \vee \beta)$ nodes, respectively. The case of (UG) is similar, requiring only one application each of ($\neg\neg$) and ($\exists$).

The overall claim now follows from the observation that if there is a closed tableau for $\neg\neg\phi$, there is also one for ϕ: Using (Cut), we generate one branch including $\neg\phi$, and one including $\neg\neg\phi$. The former can be closed immediately using ($\bot$), the latter like in the given closed tableau for $\neg\neg\phi$.

An interesting observation is that the above proof did not make use of the (TCut) and (TSub) rules, so the system is also complete when these rules are not included, as long as the cut rule is present.

4.2 Completeness via Hintikka sets (no cuts needed)

The overall structure of our completeness proof follows the one from [20], but we make use of a different tableau construction procedure, based on the lexicographic order given in Definition 7, and used throughout the remaining part of the completeness proof. Moreover, our proof covers a language that includes equality and function symbols.

In the following, when t is a primitive term and n a standard name, we call the formula $(t = n)$ a *primitive assignment*. In the semantics of $\mathcal{L}$, a world w can then be identified by (with) the set of primitive atoms *and* primitive assignments it satisfies. Obviously, for two distinct names n and m, it is impossible that both $(t = n)$ and $(t = m)$ are true simultaneously. For the proof, and also for practical reasons, it will be helpful to include the following admissible rule in our system:

$$\frac{(t = n), (t = m)}{*}\ (\overset{\circ}{=})$$

Proposition 1. *Rule* ($\overset{\circ}{=}$) *is admissible wrt the system given in Figure 3, i.e., for every tableau involving the rule, there is an equivalent one not including it.*

Proof. If a branch contains $(t = n)$ and $(t = m)$, then we can add $(n = m)$ by the (TSub) rule, and close it using the ($=$) rule.

Literals and their complement are defined as usual, and include equalities and inequalities. The subcategory of *flat* literals consists of all primitive atoms, negated primitive atoms, primitive assignments, and equalities and inequalities over standard names.

Definition 5 (Downward Saturated Set). *A set S of $\mathcal{L}$ sentences is called* downward saturated *if it satisfies the following conditions:*

1. *If S contains $\neg(\phi \vee \psi)$, then it contains $\neg\phi$ and $\neg\psi$.*
2. *If S contains $\neg\neg\phi$, then it contains ϕ.*
3. *If S contains $\phi \vee \psi$, then it contains ϕ or it contains ψ.*
4. *If S contains $\exists x\phi$, then it contains ϕ^x_n for some standard name n.*
5. *If S contains $\neg\exists x\phi$, then it contains $\neg\phi^x_n$ for all standard names n.*
6. *If S contains a non-flat literal $\phi[t]$ in which some primitive term t occurs, then for some standard name n, S contains $(t = n)$ and $\phi[n]$.*

Definition 6 (Hintikka Set). *A* Hintikka set *is a downward saturated set that does not contain a primitive atom and its negation, does not contain an assignment and its negation, does not contain two assignments $(t = n)$ and $(t = m)$ for the same t and distinct standard names n, m, does not contain $\neg(n = n)$ for any standard name, and does not contain $(n = m)$ for any two distinct standard names n, m.*

We remark that the definition above corresponds to what is called an *atomic* Hintikka set by Letz [20] and others, since only inconsistencies among or within literals are considered. Obviously, a set S that contains both ϕ and $\neg\phi$ for a non-literal formula is also unsatisfiable, which is why the $(\bot)$ rule allows to close corresponding branches. However, as we will see, for proving completeness, it suffices to restrict our attention to (flat) literals when closing branches in a tableau. For the sake of brevity, we will still just write "Hintikka set", "closed branch" etc. instead of "atomic Hintikka set", "atomically closed branch" etc.

The reader may have observed that there is a correspondence between most clauses listed in Definition 5 and the tableau expansion rules shown in Figure 3, except for clause 6, which corresponds to rules (TCut) and (TSub) combined. Indeed, these two rules are often applied directly after one another, and it would be possible to merge them so that branching over standard names and substituting them for t happens in a single step. However, in practice it makes sense to keep them separate since there may be branches that already contain an assignment $(t = n)$ for the primitive term t in question, and there we would only want to apply substitution. In such a case, the merged rule would first branch over all relevant standard names, but then immediately close all but one branches again.

Note that the restriction to primitive terms is sufficient here, as non-primitive ground terms are handled by unnesting them in a recursive fashion. For example, if a set S contains $P(f(a))$ and $f(a) = n$, then a is a primitive term occurring in a non-flat literal, and by clause 6, S has to include $(a = m)$, $P(f(m))$ and $f(m) = n$ for some name m. Then $f(m)$ is another primitive term occurring in a non-flat literal, and S subsequently has to include $P(n)$.

Lemma 1 (Hintikka's Lemma). *Every Hintikka set is satisfiable.*

Proof. Let S be a Hintikka set. The proof is similar to the one presented by Letz in [20], but simpler as it does not need to resort to Herbrand interpretations. In a manner of speaking, Herbrand interpretations are directly "built into" $\mathcal{L}$: A world w is characterized completely by which primitive atoms $P(n_1, \ldots, n_k)$ are true in it, and what names it assigns to primitive terms $f(n_1, \ldots, n_k)$. By

assumption, S does not contain any complementary literals over primitive atoms, or over assignments, or two assignments $(t = n)$, $(t = m)$ for distinct names, or unsatisfiable (in-)equalities over standard names. Note that this also excludes the case where S contains $(t \neq n)$ for *every* name n, as by clause 6 in Def. 5, S then needs to contain some $(t = n)$ as well. Therefore, there is a world w that satisfies all such literals in S.

With these literals as base cases, it then follows by induction on the structure of terms and formulas that w satisfies all sentences in S. For example, if $\exists x\phi \in S$, then by Def. 5, $\phi^x_n \in S$ for some standard name n. By induction, $w \models \phi^x_n$, and hence $w \models \exists x\phi$ by the substitutional semantics of $\mathcal{L}$. Also, if $\phi[t]$ is a non-flat literal in S with some primitive term t, then by Def. 5, $\phi[n] \in S$ for some standard name n. Then $w \models \phi[n]$ by induction, and since $w \models (t = n)$, we have $w(t) = w(n)$. Hence by the semantics of $\mathcal{L}$, $w \models \phi[t]$. The other cases are similar.

In what follows, we need total orders over the set of standard names and the set of primitive terms. For standard names, we can use the one where #1 is the first name, #2 the second, and so on. For the primitive terms, we can use any term order. One difference to the standard first-order case, cf. [20], is that we do not consider infinite sets of sentences as input, but only a single sentence (see discussion in the next section).

Definition 7 (Systematic Tableau Construction). *We assume that every node in a tableau T is optionally equipped with a natural number label, except for flat literals, which are never assigned a number label. Any such node is assigned a pair (d, m) where m is the number labeling the node in question and d is the depth of this node in the tableau tree. We order such pairs with the lexicographic order, that is, $(d, m) < (d', m')$ iff (1) $d < d'$ or (2) $d = d'$ and $m < m'$. Note that the reflexive closure of this order is a total order.*

Given a tableau T with at least one labeled node, let (d, m) be the lexicographically smallest pair assigned to a labeled node in the tableau tree (such a pair exists because the reflexive closure of the order is total). The leftmost node assigned the pair (d, m) will be called the usable node *of T (this is well-defined since all nodes assigned (d, m) have the same depth). The* systematic tableau sequence *$T_0, T_1, \ldots$ for a sentence ϕ is then given by:*

- *The tableau T_0 is a one-node tableau with root formula ϕ. If it is not a flat literal, it is assigned number label 1.*
- *Given tableau T_k containing usable node N with number label m, T_{k+1} is obtained by expanding each open branch B passing through N in T_k to:*
  1. *$B \oplus \neg\phi \oplus \neg\psi$, if the formula at N is $\neg(\phi \vee \psi)$;*
  2. *$B \oplus \phi$, if the formula at N is $\neg\neg\phi$;*
  3. *$B \oplus \phi \mid \psi$, if the formula at N is $\phi \vee \psi$;*
  4. *$B \oplus \phi^x_{n_1} \mid \cdots \mid \phi^x_{n_l} \mid \phi^x_{n'}$, if the formula at N is $\exists x\phi$, $n_1, \ldots, n_l$ is the ordered sequence of all standard names occurring in B, and n' is the least standard name not occurring in B;*
  5. *$B \oplus \neg\phi^x_n \oplus \neg\exists x\phi$, if the formula at N is $\neg\exists x\phi$ and n is the m-th standard name;*

6. *If the formula at N is a non-flat literal ϕ, where t is the least primitive term occurring in ϕ, we proceed in two steps:*
   (a) *First, every open branch B through N that does* not *contain an assignment for t is expanded to $B \oplus (t = n_1) \mid \cdots \mid (t = n_l) \mid (t = n')$, where $n_1, \ldots, n_l$ is the ordered sequence of all standard names occurring in B, and n' is the least standard name not occurring in B;*
   (b) *Afterwards, every open branch B through N containing an assignment $(t = n)$ is expanded to $B \oplus \phi[n]$.*

   The number label of the formula at N is removed. If the formula at N is of the form $\neg\exists x\phi$, every new node of the form $\neg\phi^x_n$ is assigned the label 1, if it is not a flat literal, and every new node of the form $\neg\exists x\phi$ is assigned the label $m+1$. If the formula at N is not of the form $\neg\exists x\phi$, every new node is assigned the number label 1, if it is not a flat literal. Finally, number labels of all nodes at all branches that have become closed (including equalities) by means of the latest expansion are removed.

– *If T_k has no usable node, it is the last tableau in the sequence.*

The notation $\oplus$ is taken from [20]: Expanding a branch B to $B \oplus \phi$ means extending B and adding the formula ϕ, and expanding B to $B \oplus \phi \mid \psi$ means splitting B into two branches, where ϕ is added to one of the obtained branches and ψ is added to the other.

In clause 5 of the definition above, we add a new copy of a formula $\neg\exists x\phi$ whenever we expand by one of its instances. While this technique may not be standard, it will be helpful in the following proofs, and is also used by Smullyan [26, pp. 58–59] in the classical first-order case. Moreover, some tableaux provers such as leanTAP [3] process universally quantified formulas in a similar fashion.

Definition 8 (Saturated Systematic Tableau). *Given a systematic tableau sequence $T_0, T_1, \ldots$ for formula ϕ, the* saturated systematic tableau *of ϕ, denoted T^*, is the smallest tree containing all T_i as initial segments. Note that T^* is guaranteed to exist because our tableau system is* non-destructive, *i.e., if we obtain T' by applying a rule to T, T is an initial segment of T'. Furthermore, T^* is infinite just in case the sequence is.*

Lemma 2. *Consider the saturated systematic tableau T^* for a systematic tableau sequence $T_0, T_1, \ldots$. Let N be a node on an open branch B in T^* containing a formula φ, which is not a flat literal, and let T_i be the first tableau in the sequence in which N occurs. Then N is the usable node of some tableau T_j, where $j \geq i$.*

Proof. The proof is based on the following four observations:

– A pair (d, m) is lexicographically smaller than any pair $(d+1, n)$.
– For any d, there are at most finitely many nodes in the tableau T^* with depth less than or equal to d.

- The label assigned to a node in a tableau T_k is either unchanged or removed in the embedding of T_k in T_{k+1}. Moreover, if the node in question is not selected as usable node in T_k, and furthermore the node is on an open branch in T^*, then its label will not be removed in the embedding of T_k in T_{k+1}.
- If N is not selected as usable node in a tableau T_k, then the usable node of T_k is assigned a pair smaller than or equal to the pair assigned to N in T_k. Moreover, the usable node of T_k has its number label removed in the embedding of T_k in T_{k+1}.

Since φ is not a flat literal, it is assigned a number label m, and hence some pair (d, m) in the lexicographic order. From the four observations above it follows that for every step succeeding T_i, if there is a strictly positive number of nodes assigned the pair (d, m) or a smaller pair, then a smaller number of nodes are assigned the pair (d, m) after the step. Therefore N is selected as the usable node of some tableau T_j with $j \geq i$.

Proposition 2. *If B is an open branch of a saturated systematic tableau T^*, then the set of formulas it contains is a Hintikka set.*

Proof. We first prove that the set of formulas at B is downwards saturated. In the proof we shall frequently make use of the assumption that B is open.

Assume that φ is a formula that is not a flat literal on the branch B and let T_k be the first tableau in the tableau sequence where φ occurs at the branch. By Lemma 2, the occurrence of φ is the usable node of some tableau T_j, where $j \geq k$.

If one of the first four clauses in Definition 5 apply, the appropriate nodes are added to the branch.

If the fifth clause in Definition 5 applies, then φ is of the form $\neg\exists x\phi$. It follows by inspection of the systematic tableau construction that the occurrence of φ in T_k, and hence in T_j, is labeled with the number 1, and hence in T_{j+1} the formula $\neg\phi^x_{\#1}$ is added to the branch, as well as a new instance of $\neg\exists x\phi$ with label 2. Now, by Lemma 2, the new occurrence of $\neg\exists x\phi$ is in turn the usable node of some tableau T_i, where $i \geq j + 1$, implying that $\neg\phi^x_{\#2}$ and $\neg\exists x\phi$ (now with label 3) are added to B, etc. By induction, it follows that B contains $\neg\phi^x_n$ for every name n.

If the sixth clause in Definition 5 applies, we observe that an open branch will never contain more than one assignment for any t, and that the number of function symbols occurring in the formula $\phi[n]$ added to the branch is reduced by at least one compared to ϕ, since $\phi[n]$ is obtained by replacing a primitive term t occurring in ϕ by n. This concludes the proof that B is downwards saturated.

Finally, note that since B is open, the set of formulas on B cannot contain both a primitive atom and its complement, or both a primitive assignment and its complement, or two different assignments for the same term, or $\neg(n = n)$ for any name n, or $(n = m)$ for any distinct names n, m. Hence the set of formulas at B is a Hintikka set.

Theorem 4 (Completeness). *The system shown in Figure 3 is complete, i.e., if a sentence ϕ is unsatisfiable, then there is a finite tableau for ϕ all of whose branches are closed.*

Proof. Let T be a saturated systematic tableau for ϕ. Then T must be closed: Suppose it is not, then there is an open branch B, whose formulas by Proposition 2 form a Hintikka set. By Lemma 1, this set would be satisfiable. Since the set includes ϕ, this would mean that ϕ is satisfiable, contradicting the assumption that this formula is unsatisfiable. Furthermore, T being closed implies it being finite: When closing a branch, it will not be extended anymore, so its length remains finite. Because every node has only finitely many successors, the tree T is also finite by König's Lemma.

5 Discussion and Concluding Remarks

5.1 Compactness

While tableau systems for classical first-order logic can work on infinite sets [20], our system only considers a single sentence as input. In case of the former, compactness of the logic is a corollary of the system's soundness and completeness: It is guaranteed that there exists a finite, closed tableau for any infinite unsatisfiable set. Since that tableau only involves a finite subset of formulas from the input, this subset is also unsatisfiable. So any infinite set is unsatisfiable just in case it has a finite unsatisfiable subset.

The restriction to a finite input is necessary in our case due to the fact that $\mathcal{L}$ is not compact, as observed by LL [21]. To see why, consider the infinite set of sentences $\{\exists x P(x), \neg P(\#1), \neg P(\#2), \neg P(\#3), \dots\}$. It is clearly unsatisfiable, yet every finite subset is satisfiable, as we can then set P to true for some name that is not mentioned. Intuitively, in order for our system to work on this set, the $(\exists)$ rule would need to branch over *all* standard names, since the initial branch enumerates all standard names. That is to say, extending our tableau system to handle infinite sets would require non-finite branching.

In a sense, $\mathcal{L}$ can be viewed as an *infinitary logic* [4] if we interpret $\exists x\alpha$ as the infinite disjunction $\bigvee_n \alpha^x_n$ and $\forall x\alpha$ as the infinite conjunction $\bigwedge_n \alpha^x_n$. Specifically, it bears close resemblance to ω-logic [13], which is defined over a countable sets of constants $n \in \omega$, and admits deriving $\forall x\varphi(x)$ from $\{\varphi(n) \mid n \in \omega\}$. Compactness in the classical sense obviously fails for infinitary logics, but there are adapted notions that involve changing the notion of "finite", including *Barwise Compactness* [1], *Kreisel Compactness* [14], and *Kreisel-Barwise Compactness* [13]. A deeper discussion of how these notions might be applicable to $\mathcal{L}$ is beyond the scope of this paper, and hence left for future work, but in simple terms we conjecture that it would involve identifying the infinite set $\{\neg P(\#1), \neg P(\#2), \dots\}$ with the equivalent, finite formula $\forall x \neg P(x)$. The set $\{\exists x P(x), \neg P(\#1), \neg P(\#2), \dots\}$ then coincides with $\{\exists x P(x), \forall x \neg P(x)\}$, and the latter can be proven unsatisfiable using a finite tableau.

5.2 Standard Names and Substitutional Quantification

LL [21] say that the original inspiration for standard names stems from "parameters" as described by Smullyan [26]. However, there are significant differences. Smullyan uses standard Tarskian semantics, where the universe can vary between interpretations, and in particular parameters are not identified with the domain of discourse, and are not necessarily distinct from one another. In more modern terminology, parameters are hence merely a countable infinite supply of *constant symbols*. Since Smullyan does not consider function symbols of higher arity, Herbrand bases are then constructed solely from predicates with parameters as arguments.

There has also been a philosophical discussion around substitutional quantification. Kripke [15] rejects earlier criticisms due to Wallace [28] and Tharp [27], among other things the application of criteria of ontological commitment resulting in "an ever-present fear that it will be shown, using substitutional quantification, that nothing exists." Formally, he sees little difference between a substitutional and a referential quantifier in the case that (a) the denotation function for terms is total and (b) all formulae are *transparent*, i.e., their truth value does not change due to substitution by equals. (Both conditions obviously hold for $\mathcal{L}$.) He concludes that there "never was any problem with substitutional quantification".

5.3 Implementation

We believe that the presented system lends itself well to an implementation, but clearly, the next step will be to validate this hypothesis through experimentation. Currently, we are working on developing a prototype in Prolog, taking inspiration from similar systems for standard first-order logic [3,10,22]. An interesting observation is that our ($\exists$) rule is very reminiscent of corresponding rules that are used for description logic tableaux [24], where they play a vital role in termination and loop detection, and it will be interesting to study whether similar results can be obtained for decidable fragments of $\mathcal{L}$.

Acknowledgements

We thank the anonymous reviewers for their helpful comments, in particular for pointing out an issue in an earlier version of the paper that has now been fixed.

References

1. Barwise, J.: Infinitary logic and admissible sets. Journal of Symbolic Logic **34**(2), 226–252 (1969). https://doi.org/10.2307/2271099
2. Beckert, B.: Equality and other theories. In: D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 197–254. Springer Netherlands, Dordrecht (1999). https://doi.org/10.1007/978-94-017-1754-0_4

3. Beckert, B., Posegga, J.: leanTAP: Lean tableau-based deduction. Journal of Automated Reasoning **15**(3), 339–358 (1995)
4. Bell, J.L.: Infinitary Logic. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Fall 2023 edn. (2023)
5. Claßen, J., Eyerich, P., Lakemeyer, G., Nebel, B.: Towards an integration of Golog and planning. In: Veloso, M.M. (ed.) Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007). pp. 1846–1851. AAAI Press (2007)
6. Claßen, J., Lakemeyer, G.: A logic for non-terminating Golog programs. In: Brewka, G., Lang, J. (eds.) Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008). pp. 589–599. AAAI Press (2008)
7. Claßen, J., Neuss, M.: Knowledge-based programs with defaults in a modal situation calculus. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) Proceedings of the Twenty-Second European Conference on Artificial Intelligence (ECAI 2016). pp. 1309–1317. IOS Press (2016). https://doi.org/10.3233/978-1-61499-672-9-1309
8. D'Agostino, M., Mondadori, M.: The taming of the cut. Classical refutations with analytical cut. Journal of Logic and Computation **4**, 285–319 (1994)
9. Fitting, M.: Modal proof theory. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 85–138. Elsevier (2007)
10. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer, 2nd edn. (1996)
11. Hofmann, T., Claßen, J.: LTLf synthesis on first-order agent programs in nondeterministic environments. In: Walsh, T., Shah, J., Kolter, Z. (eds.) Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2025). pp. 14976–14986. AAAI Press (2025). https://doi.org/10.1609/aaai.v39i14.33642
12. Jeffrey, R.: Formal Logic: Its Scope and Limits. McGraw Hill (1967)
13. Keisler, H.J., Knight, J.F.: Barwise: Infinitary logic and admissible sets. Bulletin of Symbolic Logic **10**(1), 4–36 (2004). https://doi.org/10.2178/bsl/1080330272
14. Kreisel, G.: Set theoretic problems suggested by the notion of potential totality. In: Infinitistic Methods (Proc. Sympos. Foundations of Math., Warsaw, 1959). pp. 103–140 (1961)
15. Kripke, S.A.: Is there a problem about substitutional quantification? In: Evans, G., McDowell, J. (eds.) Truth and meaning: essays in semantics, pp. 324–419. Clarendon Press (1976)
16. Kripke, S.A.: Naming and Necessity: Lectures Given to the Princeton University Philosophy Colloquium. Harvard University Press, Cambridge, MA (1980)
17. Lakemeyer, G.: The situation calculus: A case for modal logic. Journal of Logic, Language and Information **19**(4), 431–450 (2010). https://doi.org/10.1007/s10849-009-9117-6
18. Lakemeyer, G., Levesque, H.J.: Only-knowing meets nonmonotonic modal logic. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2012). AAAI Press (2012)
19. Lakemeyer, G., Levesque, H.J.: A tractable, expressive, and eventually complete first-order logic of limited belief. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019). pp. 1764–1771. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/244

20. Letz, R.: First-order tableau methods. In: D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 125–196. Springer Netherlands, Dordrecht (1999). https://doi.org/10.1007/978-94-017-1754-0_3
21. Levesque, H.J., Lakemeyer, G.: The Logic of Knowledge Bases. College Publications, second edn. (2022)
22. Posegga, J., Schmitt, P.H.: Implementing semantic tableaux. In: D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 581–629. Springer Netherlands, Dordrecht (1999). https://doi.org/10.1007/978-94-017-1754-0_10
23. Reeves, S.V.: Adding equality to semantic tableaux. J. Autom. Reason. **3**(3), 225–246 (1987). https://doi.org/10.1007/BF00243790
24. Schmidt, R.A., Tishkovsky, D.: Using tableau to decide description logics with full role negation and identity. ACM Transactions on Computational Logic **15**(1), 7:1–7:31 (2014). https://doi.org/10.1145/2559947
25. Schwering, C.: A reasoning system for a first-order logic of limited belief. In: Bacchus, F., Sierra, C. (eds.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017). pp. 5246–5248. AAAI Press (2017)
26. Smullyan, R.M.: First-Order Logic. Springer, New York [etc.] (1968)
27. Tharp, L.H.: Truth, quantification, and abstract objects. Noûs **5**(4), 363–372 (1971)
28. Wallace, J.: Convention T and substitutional quantification. Noûs **5**(2), 199–211 (1971)
29. Zarrieß, B., Claßen, J.: Decidable verification of Golog programs over non-local effect actions. In: Schuurmans, D., Wellman, M. (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016). pp. 1109–1115. AAAI Press (2016)