

# A Tableau System for First-Order Logic with Standard Names

Jens Claßen    Torben Braüner

Department of People and Technology  
Roskilde University, Denmark



TABLEAUX 2025  
Reykjavik, Iceland  
27–29 September 2025

# The Logic $\mathcal{L}$

(Levesque 1981; 1984), (Levesque & Lakemeyer 2022)

$\mathcal{L}$  is a first-order logic with functions and equality using **standard names**  $\mathcal{N} = \{\#1, \#2, \#3, \dots\}$ .

- ▶  $\mathcal{N}$  serves as fixed **universe of discourse**
- ▶  $\mathcal{N}$  is also **part of the language** (like constants)

$\mathcal{L}$  is basis for modal logics for notions of. . .

- ▶ belief and “only-knowing” (Levesque 1981; 1984), (Levesque & Lakemeyer 2022)
- ▶ non-monotonic inference (Lakemeyer & Levesque 2006; 2012)
- ▶ actions and change (Lakemeyer & Levesque 2004), (Lakemeyer 2010)
- ▶ agent programs and temporal specifications (Claßen & Lakemeyer 2008), (Zarriß & Claßen 2016)
- ▶ . . .

However, so far there is no implementation of a sound and complete reasoner for  $\mathcal{L}$ , only:

- ▶ a Hilbert-style axiom system **not suitable for actual reasoning** (Levesque & Lakemeyer 2022)
- ▶ an implementation of a tractable, but **incomplete** reasoner (Schwering 2017)
- ▶ a **non-analytic**, resolution-style reasoning mechanism (Lakemeyer & Levesque 2019)

📖 Here we present a **sound & complete tableau system** for  $\mathcal{L}$ , and also its first **analytic** proof system.

# The Logic $\mathcal{L}$

(Levesque 1981; 1984), (Levesque & Lakemeyer 2022)

**Syntax:** Standard names used like **constants**, e.g.,

$$\forall x. (x \neq \#3) \supset P(x, a), \quad \exists y. f(\#7) = g(y)$$

**Semantic model:** A **world**  $w$  is a mapping where

- ▶  $w[P(n_1, \dots, n_k)] \in \{0, 1\}$  for **primitive atoms**  $P(n_1, \dots, n_k)$  (all  $n_i$  are standard names)
- ▶  $w[f(n_1, \dots, n_k)] \in \mathcal{N}$  for **primitive terms**  $f(n_1, \dots, n_k)$  (all  $n_i$  are standard names)

**Value of ground terms:**

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

**Sentence satisfaction:**

1.  $w \models P(t_1, \dots, t_k)$  iff  $w[P(n_1, \dots, n_k)] = 1$ , where  $n_i = w(t_i)$ ;
2.  $w \models (t_1 = t_2)$  iff  $w(t_1)$  is the same name as  $w(t_2)$ ; “=” as identity of co-referring standard names
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  $n$ ,  $w \models \phi_n^x$ . substitutional interpretation of quantification

# Axiom System for $\mathcal{L}$

(Levesque & Lakemeyer 2022)

## 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  $\alpha$
5.  $\forall x\alpha \supset \alpha_t^x$
6.  $(n = n) \wedge (n \neq m)$  for any distinct  $n, m$

## Rules:

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

📌 Axiom 6 formalizes equality as **identity over standard names**.

📌 Universal Generalization only requires **finitely many instances**:

It is enough to prove  $\alpha_n^x$  for one unmentioned name  $n$ .

The same proof can be used for any other unmentioned name  $n'$  by substituting  $n$  with  $n'$ .

# Example Proof in $\mathcal{L}$

(Levesque & Lakemeyer 2022)

1. $\#1 = \#1$	Ax6
2. $\forall x(x = x)$	1, UG
3. $f(\#1) = f(\#1)$	MP
4. $\#1 = \#1 \supset f(\#1) = f(\#1)$	MP
5. $\#1 \neq \#2$	Ax6
6. $\#1 = \#2 \supset f(\#1) = f(\#2)$	MP
7. $\forall y(\#1 = y \supset f(\#1) = f(y))$	4,6, UG
8. $\forall x\forall y(x = y \supset f(x) = f(y))$	7, UG

# Tableau System for $\mathcal{L}$

$\frac{\neg(\phi \vee \psi)}{\neg\phi, \neg\psi} (\neg\vee)$	$\frac{\neg\neg\phi}{\phi} (\neg\neg)$	$\frac{\phi \vee \psi}{\phi \mid \psi} (\vee)$
$\frac{\exists x\alpha}{\alpha_{n_1}^x \mid \dots \mid \alpha_{n_k}^x} (\exists)$	$\frac{\neg\exists x\alpha}{\neg\alpha_t^x} (\neg\exists)$	
$\frac{}{(t = n_1) \mid \dots \mid (t = n_k)} (\text{TCut})$	$\frac{\phi[t], (t = n)}{\phi[n]} (\text{TSub})$	
$\frac{\phi, \neg\phi}{*} (\perp)$	$\frac{\neg(t = t)}{*} (\neq)$	$\frac{(n = m)}{*} (=)$

$(\exists)$ ,  $(\text{TCut})$ :  $n_1, \dots, n_k$  range over all standard names in the branch, plus one extra.

☞ Branches over **finitely many instances**.

$(=)$ :  $n$  and  $m$  are distinct standard names.

☞ Equality as **identity over standard names**.

## Example Derivation

1.  $\exists x \exists y \neg (x=y \supset f(x)=f(y))$

## Example Derivation

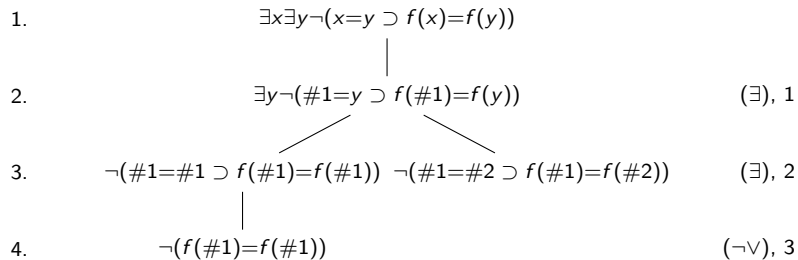
1.  $\exists x \exists y \neg (x=y \supset f(x)=f(y))$
2.  $\exists y \neg (\#1=y \supset f(\#1)=f(y))$  ( $\exists$ ), 1



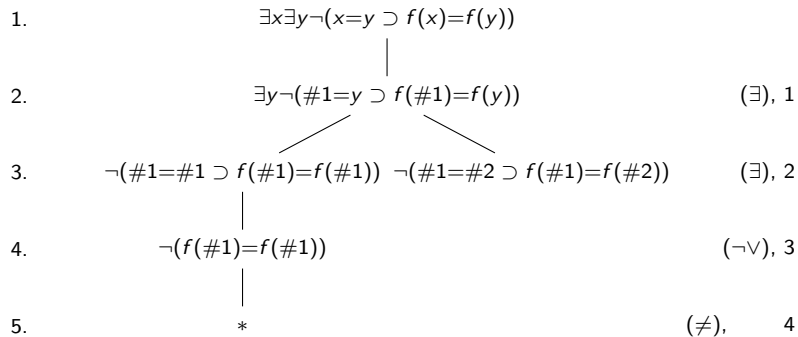
## Example Derivation

1.  $\exists x \exists y \neg(x=y \supset f(x)=f(y))$
2.  $\exists y \neg(\#1=y \supset f(\#1)=f(y))$  ( $\exists$ ), 1
3.  $\neg(\#1=\#1 \supset f(\#1)=f(\#1)) \quad \neg(\#1=\#2 \supset f(\#1)=f(\#2))$  ( $\exists$ ), 2

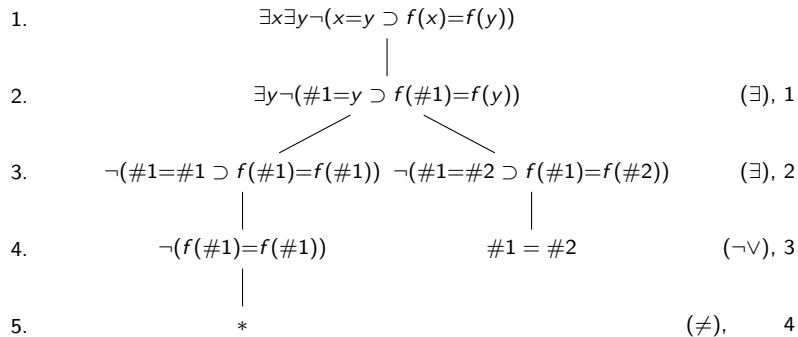
## Example Derivation



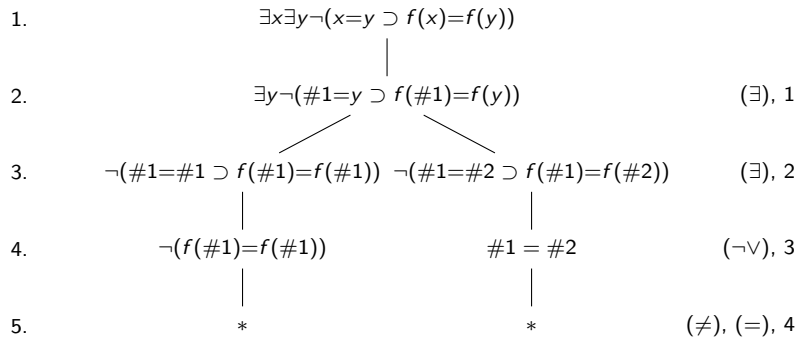
## Example Derivation



## Example Derivation



## Example Derivation



## Example Derivation

1.

$$a = b$$

|

2.

$$P(a, a)$$

|

3.

$$\neg P(b, b)$$

## Example Derivation

1.  $a = b$   
|
2.  $P(a, a)$   
|
3.  $\neg P(b, b)$   
|
4.  $a = \#1$  (TCut),  $a$

## Example Derivation

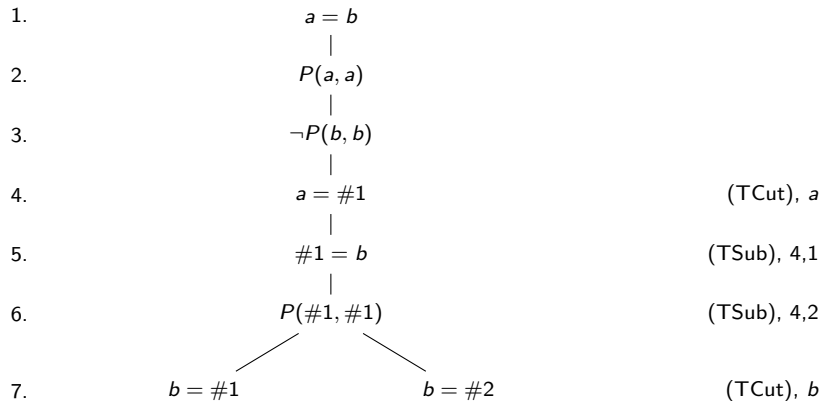
- |    |                |             |
|----|----------------|-------------|
| 1. | $a = b$        |             |
|    |                |             |
| 2. | $P(a, a)$      |             |
|    |                |             |
| 3. | $\neg P(b, b)$ |             |
|    |                |             |
| 4. | $a = \#1$      | (TCut), $a$ |
|    |                |             |
| 5. | $\#1 = b$      | (TSub), 4,1 |



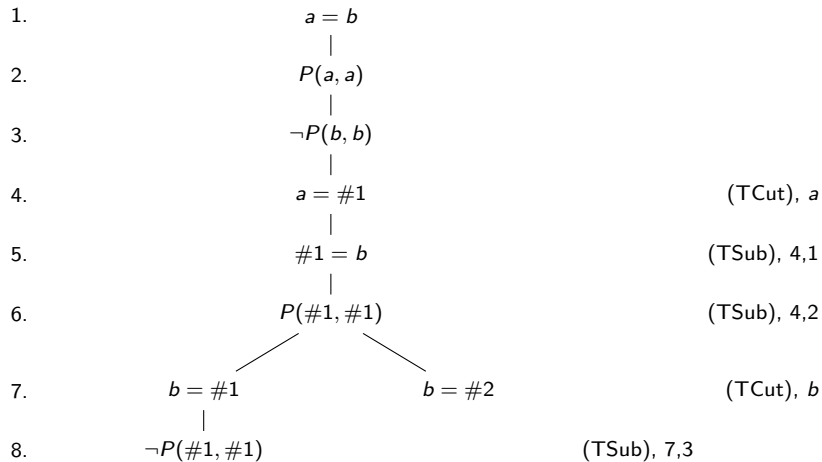
## Example Derivation

- |    |                |             |
|----|----------------|-------------|
| 1. | $a = b$        |             |
|    |                |             |
| 2. | $P(a, a)$      |             |
|    |                |             |
| 3. | $\neg P(b, b)$ |             |
|    |                |             |
| 4. | $a = \#1$      | (TCut), $a$ |
|    |                |             |
| 5. | $\#1 = b$      | (TSub), 4,1 |
|    |                |             |
| 6. | $P(\#1, \#1)$  | (TSub), 4,2 |

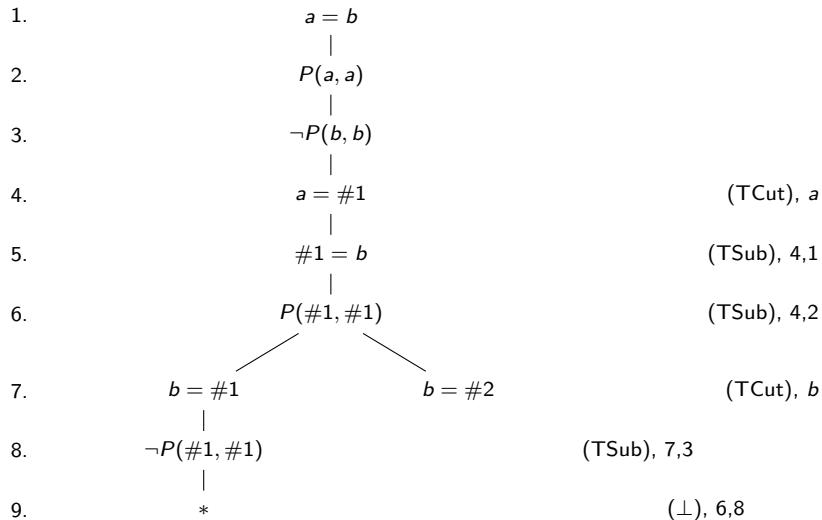
## Example Derivation



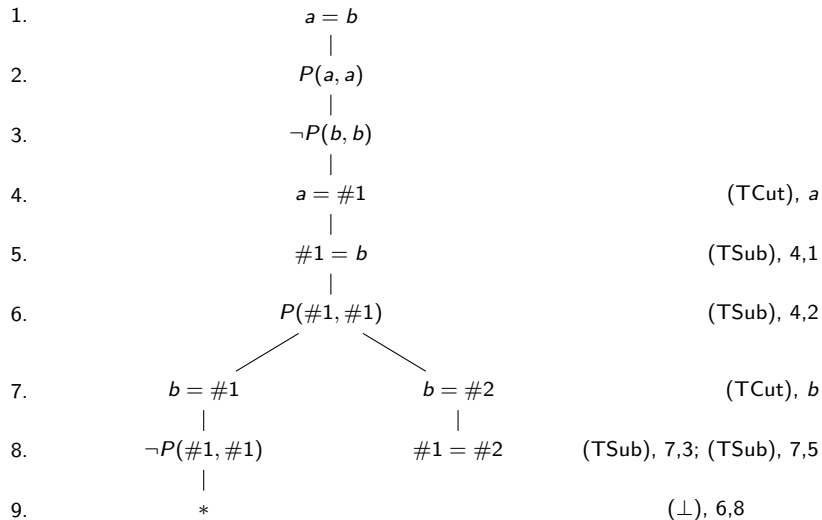
## Example Derivation



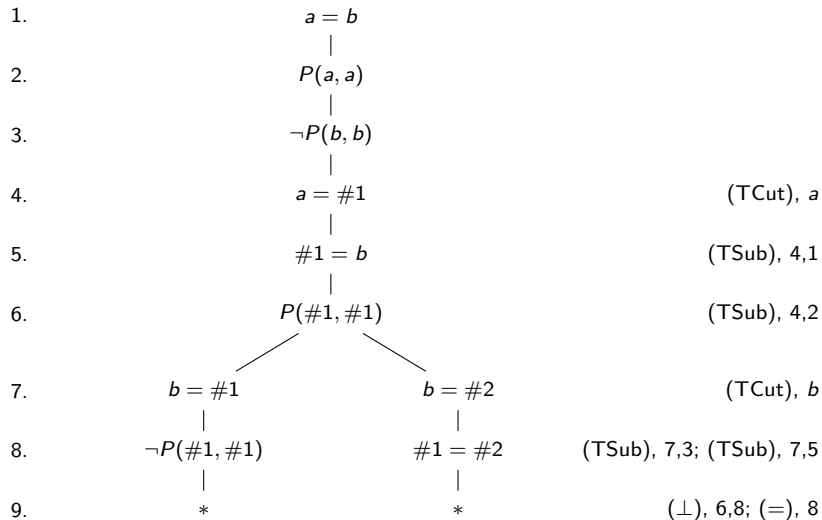
## Example Derivation



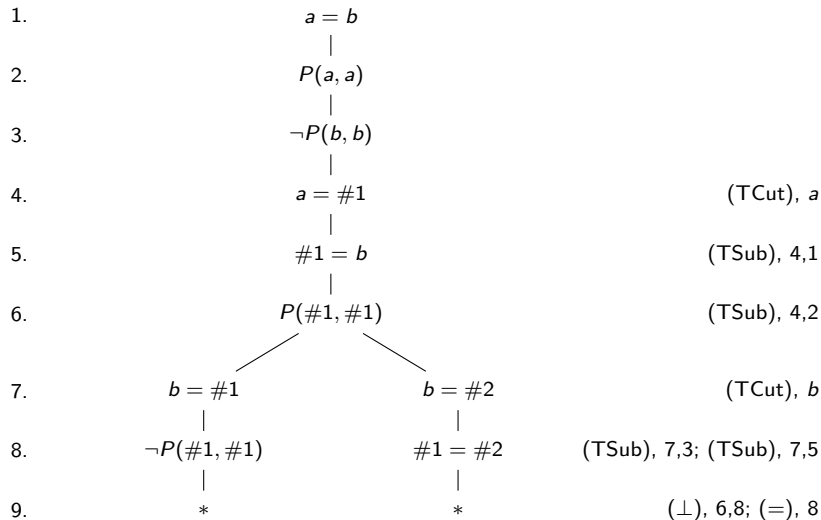
## Example Derivation



## Example Derivation



## Example Derivation



Function symbols are handled **analytically** by systematically substituting subterms by values (names).

# Soundness and Completeness

**Theorem.** The tableau system for  $\mathcal{L}$  is **sound**.

*Proof.* Use induction to show that expansion rules preserve satisfiability on a branch.

**Theorem.** The tableau system for  $\mathcal{L}$  is **complete**.

*Proof.* Following (Letz 1999) in overall structure:

1. Build a saturated systematic tableau.
2. Show that every open branch is satisfiable.

(uses specific **lexicographic order** for node selection)

(uses variant of **Hintikka set** and Hintikka's lemma)



# Discussion

Our tableau system only works on **finite inputs**, since  $\mathcal{L}$  is **not compact**:

$$\{\exists x P(x), \neg P(\#1), \neg P(\#2), \neg P(\#3), \dots\}$$

☞ extended notions of compactness for infinitary logics ( $\omega$ -logic)

There are philosophical reservations against **substitutional quantification** wrt lack of “ontological commitment”. Kripke (1976) argues that substitutional and referential quantification coincide if

- (a) denotation function for terms is total
- (b) all formulae are transparent

☞ Both conditions hold for  $\mathcal{L}$ . Standard names are rigid designators in the sense of (Kripke 1980).

We believe that our systems lends itself well to an **implementation**.

- ☞ We are developing a prototype in Prolog, similar in spirit to LeanTAP and other systems for FOL.
- ☞ In the long run, we are interested in guaranteeing **termination** for decidable fragments.

# Thank You!