ANITA: Analytic Tableau Proof Assistant

Davi Romero Vasconcelos
UFC
Quixadá, Brazil
School of Computer Science
Federal University of Ceará
Quixadá, Brazil
daviromero@ufc.br

This work presents the system ANITA (Analytic Tableau Proof Assistant) developed for teaching analytic tableaux to computer science students. The tool is written in Python and can be used as a desktop application, or in a web platform. This paper describes the logical system of the tool, explains how the tool is used and compares it to several similar tools. ANITA has already been used in logic courses and an evaluation of the tool is presented.

1 Introduction

Logic in Computer Science course is part of most Information and Communication Technology curricula, such as the curricula of the Information Systems, Software Engineering, Computer Science, and Computer Engineering of the Federal University of Ceará at Quixadá as mandatory components. The course has a high failure rate. For better assimilation of the contents, it is essential that the students exercise and that they get feedback on the correctness of their proofs.

Many deductive systems are used for teaching the formal reasoning of proofs, such as Axiomatic Systems (a la Hilbert), Natural Deduction System [7, 8], and Analytic Tableaux [12, 2].

The Analytic Tableau system is widely used for teaching proofs and appears in many Logic textbooks such as [12, 11, 13, 4]. This work presents a proof assistant, ANITA (Analytic Tableau Proof Assistant), in order to assist in the teaching-learning of undergraduate and graduate students. For the purpose of teaching deduction systems, we take into account in ANITA the following features: the students should write their proofs as similar as possible to what is available in the textbooks and to what the students would usually write on paper; the tool should be easy-to-use and reduce the number of clicks since mouse-clicking can be tedious; the tool should allow the student to make mistakes and point out errors on the proofs.

The rest of the paper is organized as follows. We provide a concise definition of Analytic Tableau system in Section 2; We propose to write proofs in Analytic Tableaux in Fitch-style in Section 3. Section 4 describes ANITA. Section 5 compares ANITA to other proof assistants. The evaluation of ANITA is presented in Section 6. And, we conclude this work in Section 7.

2 Analytic Tableaux

We now describe Analytic Tableaux for propositional logic which we will subsequently extend to first-order logic.

Analytic Tableaux is an inference method based on *refutation*: to prove $\Gamma \vdash \varphi$, we assert that each formula of Γ is *true* and φ is *false*, in order to derive a *contradiction*. On the other hand, if no contra-

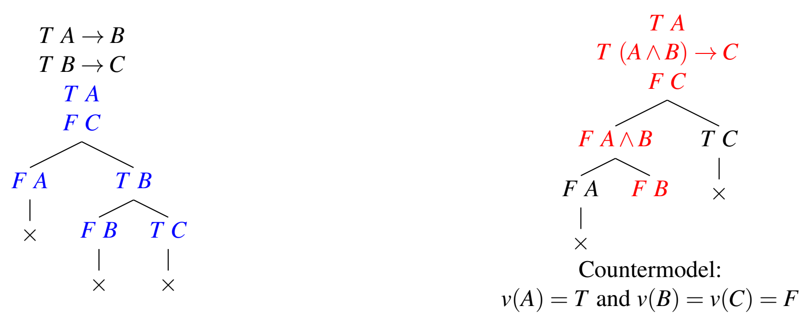
J. Marcos, W. Neuper and P. Quaresma (Eds.): Theorem Proving Components for Educational Software 2022 (ThEdu'22)
EPTCS 375, 2023, pp. 38–53, doi:10.4204/EPTCS.375.4

diction is obtained, then we construct a *countermodel*, that is, a valuation[1] that satisfies Γ and does not satisfy φ.

In the method of analytic tableaux, we define $T\ \varphi$ and $F\ \varphi$ as signed formulas to stand that φ is true and φ is false. The first step to constructing a tableau is to label all formulas in Γ with T and the formula φ with F. Starting from the initial tableau, tableau expansion rules can be used to: add new formulas to the end of a branch (α-type rules); or split a branch into two branches (β-type rules). The rules for the construction of tableaux are as follows:

α rule	$T\ \varphi\wedge\psi$ | $T\ \varphi$ $T\ \psi$	$F\ \varphi\vee\psi$ | $F\ \varphi$ $F\ \psi$	$F\ \varphi\rightarrow\psi$ | $T\ \varphi$ $F\ \psi$	$T\ \neg\varphi$ | $F\ \varphi$ $F\ \neg\varphi$ | $T\ \varphi$
β rule	$F\ \varphi\wedge\psi$ $F\ \varphi$ $F\ \psi$	$T\ \varphi\vee\psi$ $T\ \varphi$ $T\ \psi$	$T\ \varphi\rightarrow\psi$ $F\ \varphi$ $T\ \psi$	

In each branch, a formula can only be expanded once. A branch that has no more formulas to expand is said to be **saturated**. A branch that has a pair of formulas $T\ \varphi$ and $F\ \varphi$ is said to be **closed**. A closed branch no longer needs to be expanded. A tableau is said to be closed whether it has all its branches closed, i.e., $\Gamma\vdash\varphi$. A saturated and unclosed branch provides a *countermodel*, i.e., $\Gamma\nvdash\varphi$. Figure 1a shows that $A\rightarrow B, B\rightarrow C, A$ entails C as an analytic tableau proof. We use $\times$ as a symbol to close a branch by the signed formulas in the blue nodes. Figure 1b shows a proof, in which we have one of the branches (red nodes) that is saturated. So, we can extract a countermodel from the truth values of the atoms in the branch.



(a) $A\rightarrow B, B\rightarrow C, A\vdash C$ (b) $A, A\wedge B\rightarrow C\nvdash C$

Figure 1: Examples of proofs in Analytic Tableau

We extend the analytic tableau system in order to include proofs of first-order logic, in which we have all the rules of propositional logic and add the following rules:

γ rule	$T\ \forall x\varphi$ | $T\ \varphi^x_t$ t is substitutable for x in φ	$F\ \exists x\varphi$ | $F\ \varphi^x_t$ t is substitutable for x in φ
δ rule	$F\ \forall x\varphi$ | $F\ \varphi^x_a$ a is a new variable	$T\ \exists x\varphi$ | $T\ \varphi^x_a$ a is a new variable

[1]A valuation function v is a mapping from the atoms to the set $\{T,F\}$.

Here φ_t^x is the expression obtained from the formula φ by replacing the variable x, whenever it occurs free in φ, by the term t. For instance, $(H(x) \to \forall x M(x))_y^x = (H(y) \to \forall x M(x))$. We can say that a term t is substitutable for x in φ if there is no variable y in t that is captured by a $\forall y$ (or $\exists y$) quantifier of φ_t^x. For example, term z is substitutable for y in $\forall x P(x,y)$. On the other hand, term x is not substitutable for y in $\forall x P(x,y)$.

The above rules can occur more than once in each branch, as we can make arbitrary substitutions of variables. Thus, in the general case, we will not be able to generate a countermodel. Figure 2 shows examples of proofs in Analytic Tableaux.

(a) $\forall x(H(x) \to M(x)), \forall x H(x) \vdash \forall x M(x)$ (b) $\forall x(H(x) \to M(x)), \exists x H(x) \vdash \exists x M(x)$

Figure 2: Examples of Proofs in Analytic Tableaux

3 Analytic Tableaux in Fitch-Style

A (signed) tableau is a certain kind of binary, labeled ordered tree where each node is labeled by a signed formula. However, we can present a version of the analytic tableaux in Fitch-style. The proof is written in a linear and sequential order, in which we number all the lines, and write a statement (signed formula) with its justification which can be a premise, the conclusion of the proof, or the application of one of the inference rules. Each split of a branch is delimited by $\{$ and $\}$. A formula can only be used in a proof at a given point if that formula happened previously and within that branch. In the sequel, we will present all the rules in Fitch-style.

Initial Tableau Rule: A proof of $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ starts with initial tableau as shown in Figure 3a, where:

- The premises $\varphi_1, \varphi_2, \ldots, \varphi_n$ are represented in one line each, following a sequential numbering, labeled as T (True) and as justification "Premise".
- The conclusion ψ is defined on the line after the last premise, labeled by F (False) and with the justification "Conclusion".

Closed Branch Rule: We say that a **branch is closed** (contains a contradiction $\perp$) in line p if a formula φ is labeled in one line m with T and in another line n with F (both before p). A closed branch can no longer be expanded. Figure 4a presents the scheme of this rule. Figure 4b shows the proof of $A \vdash A$, in

1.	$T\ \varphi_1$	Premise
2.	$T\ \varphi_2$	Premise
⋮	⋮	⋮
n.	$T\ \varphi_n$	Premise
n+1.	$F\ \psi$	Conclusion
⋮	⋮	⋮

(a) Initial Tableau

1.	$T\ A$	Premise
2.	$F\ A$	Conclusion
3.	⋮	⋮

(b) Sample $A \vdash A$

Figure 3: Initial Tableau Rule

which we close the (single) branch in line 3 from the formula A referenced in lines 1 and 2 as T and F, respectively.

⋮	⋮	⋮
m.	$T\ \varphi$	
⋮	⋮	⋮
n.	$F\ \varphi$	
⋮	⋮	⋮
p.	$\perp$	m,n

(a) Closed Branch

1.	$T\ A$	Premise
2.	$F\ A$	Conclusion
3.	$\perp$	1,2

(b) $A \vdash A$

Figure 4: Closed Branch Rule

The **negation-true rule** ($\neg T$) is shown in Figure 5a, where the signed formula $F\ \varphi$ can be obtained in line n by the signed formula $T\ \neg\varphi$ in line m. In a similar way, the **negation-false rule** ($\neg F$), see Figure 5b, can be used to show $T\ \varphi$ in line n by $F\ \neg\varphi$ in line m. As we can see, in Figure 5c, we conclude $T\ \neg A$ in line 3 by the negation-false rule in line 2. So, we apply $\neg T$ rule and get $F\ A$ in line 4. Thus, we close the branch in line 5 (a contradiction), because we have $T\ A$ in line 1 and $F\ A$ in line 4.

⋮	⋮	⋮
m.	$T\ \neg\varphi$	
⋮	⋮	⋮
n.	$F\ \varphi$	m

(a) Negation-True ($\neg T$)

⋮	⋮	⋮
m.	$F\ \neg\varphi$	
⋮	⋮	⋮
n.	$T\ \varphi$	m

(b) Negation-False ($\neg F$)

1.	$T\ A$	Premise
2.	$F\ \neg\neg A$	Conclusion
3.	$T\ \neg A$	2
4.	$F\ A$	3
5.	$\perp$	1,4

(c) $A \vdash \neg\neg A$

Figure 5: Negation Rules

The **and-true rule** ($\wedge T$) is shown in Figure 6a, in which the signed formulas $T\ \varphi$ and $T\ \psi$ in lines n and $n+1$, respectively, are obtained by the signed formula $T\ \varphi \wedge \psi$ in line m. For example, in Figure 6b, we apply rule $\wedge T$ to $T\ A \wedge B$ in line 1 and derive $T\ A$ and $T\ B$, in lines 3 and 4.

The **and-false rule** ($\wedge F$) is shown in Figure 7a. This rule is applied to $F\ \varphi \wedge \psi$ in line m and splits

⋮	⋮	⋮
m.	$T\ \varphi \wedge \psi$	
⋮	⋮	⋮
n.	$T\ \varphi$	m
n+1.	$T\ \psi$	m

(a) And-True ($\wedge T$)

1.	$T\ A \wedge B$	Premise
2.	$F\ A$	Conclusion
3.	$T\ A$	1
4.	$T\ B$	1
5.	$\bot$	2,3

(b) $A \wedge B \vdash A$

Figure 6: And-True Rule

this branch into two: one that starts in line n with $F\ \varphi$; and the other in line p with $F\ \psi$. To delimit the respective branches, we use the symbols { and }. For instance, in Figure 7b, we apply $\wedge F$ rule to $F\ A \wedge B$ in line 3 and split this branch:

1. In the branch starting in line 4 with $F\ A$ which is used with $T\ A$ from line 1 to close this branch in line 5.
2. In the branch starting in line 6 with $F\ B$ which is used with $T\ B$ from line 2 to close this branch in line 7.

It is worth noting that we can only reference the formulas in the same branch. So, for example, the formula $F\ A$ in line 4 could not be referenced in the branch starting in line 6.

⋮	⋮		⋮
m.	$F\ \varphi \wedge \psi$		
⋮	⋮		⋮
n.	{	$F\ \varphi$	m
⋮		⋮	⋮
	}		
p.	{	$F\ \psi$	m
⋮		⋮	⋮
	}		

(a) And-False ($\wedge F$)

1.	$T\ A$		Premise
2.	$T\ B$		Premise
3.	$F\ A \wedge B$		Conclusion
4.	{	$F\ A$	3
5.		$\bot$	1,4 }
6.	{	$F\ B$	3
7.		$\bot$	2,6 }

(b) $A, B \vdash A \wedge B$

Figure 7: And-False Rule

The **or-true rule** ($\vee T$) is presented in Figure 8a. We apply this rule to $T\ \varphi \vee \psi$ in line m and we split this branch into two new branches: one that starts in line n with $T\ \varphi$; and the other in line p with $T\ \psi$. For example, in Figure 8b, the rule $\vee T$ is applied to $T\ A \vee B$ in line 1 and we split this branch:

1. In the branch that starts in line 5 with $T\ A$ which is used with $F\ A$ in line 3 to close this branch in line 6.
2. In the branch starting in line 7 with $T\ B$ which is used with $F\ B$ in line 4 to close this branch in line 8.

The **or-false rule** ($\vee F$) is shown in Figure 9a, in which the signed formulas $F\ \varphi$ and $F\ \psi$ in lines n and $n+1$, respectively, are obtained by the signed formula $F\ \varphi \vee \psi$. For example, in Figure 9b, we apply rule $\vee F$ to $F\ A \vee B$ in line 2 and derive $F\ A$ and $F\ B$, in lines 3 and 4.

⋮	⋮		⋮
m.	$T\ \varphi \vee \psi$		
⋮	⋮		⋮
n.	{	$T\ \varphi$	m
⋮		⋮	⋮
	}		
p.	{	$T\ \psi$	m
⋮		⋮	⋮
	}		

(a) Or-True ($\vee T$)

1.	$T\ A \vee B$		Premise	
2.	$T\ \neg B$		Premise	
3.	$F\ A$		Conclusion	
4.	$F\ B$		2	
5.	{	$T\ A$	1	
6.		$\bot$	3,5	}
7.	{	$T\ B$	1	
8.		$\bot$	4,7	}

(b) $A \vee B, \neg B \vdash A$

Figure 8: Or-True Rule

⋮	⋮	⋮
m.	$F\ \varphi \vee \psi$	
⋮	⋮	⋮
n.	$F\ \varphi$	m
n+1.	$F\ \psi$	m

(a) Or-False ($\vee F$)

1.	$T\ A$	Premise
2.	$F\ A \vee B$	Conclusion
3.	$F\ A$	2
4.	$F\ B$	2
5.	$\bot$	1,3

(b) $A \vdash A \vee B$

Figure 9: Or-False Rule

The **implication-true rule ($\rightarrow T$)** is presented in Figure 10a. We apply this rule to $T\ \varphi \rightarrow \psi$ in line m and we split this branch into two new branches: one that starts in line n with $F\ \varphi$; and the other in line p with $T\ \psi$. For example, in Figure 10b, the rule $\rightarrow T$ is applied to $T\ A \rightarrow B$ in line 1 and we split this branch:

1. In the branch that starts in line 5 with $F\ \neg A$, in which we apply $\neg F$ to obtain $T\ A$ in line 6 and, then, we use $F\ A$ in line 3 to close this branch in line 7.
2. In the branch starting in line 8 with $T\ B$ which is used with $F\ B$ in line 4 to close this branch in line 9.

The **implication-false rule ($\rightarrow F$)** is shown in Figure 11a, in which the signed formulas $T\ \varphi$ and $F\ \psi$ in lines n and $n+1$, respectively, are obtained by the signed formula $F\ \varphi \rightarrow \psi$. For example, in Figure 11b, we apply rule $\rightarrow F$ to $F\ A \rightarrow B$ in line 2 and derive $T\ A$ and $F\ B$, in lines 3 and 4.

The **universal-true rule ($\forall T$)** is shown in Figure 12a, in which we apply $\forall T$-rule to signed formula $T\ \forall x \varphi(x)$ in line m and obtain, in line n, a signed formula $T\ \varphi_t^x$, where t is substitutable for x in φ. Figure 12b shows the use of this rule to $T\ \forall x(H(x) \rightarrow M(x))$ in line 1 to get $T\ H(s) \rightarrow M(s)$ in line 4.

The **universal-false rule ($\forall F$)** is shown in Figure 13a, in which we apply $\forall F$-rule to signed formula $F\ \forall x \varphi(x)$ in line m and obtain, in line n, a signed formula $F\ \varphi_a^x$, where a is a new variable. Figure 13b illustrates the use of this rule to $F\ \forall x M(x)$ in line 3 to conclude $F\ M(a)$ in line 4.

The **existential-true rule ($\exists T$)** is shown in Figure 14a, in which we apply $\exists T$-rule to signed formula $T\ \exists x \varphi(x)$ in line m and obtain, in line n, a signed formula $T\ \varphi_a^x$, where a is a new variable. Figure 14b illustrates the use of this rule to $T\ \exists x\, H(x)$ in line 2 to get $T\ H(a)$ in line 4.

⋮	⋮	⋮
m.	$T\ \varphi \to \psi$	
⋮	⋮	⋮
n.	{ $F\ \varphi$	m
⋮	⋮	⋮
	}	
p.	{ $T\ \psi$	m
⋮	⋮	⋮
	}	

(a) Implication-True ($\to T$)

1.	$T\ \neg A \to B$	Premise
2.	$F\ A \vee B$	Conclusion
3.	$F\ A$	2
4.	$F\ B$	2
5.	{ $F\ \neg A$	1
6.	$T\ A$	5
7.	$\bot$	6,3 }
8.	{ $T\ B$	1
9.	$\bot$	8,4 }

(b) $\neg A \to B \vdash A \vee B$

Figure 10: Implication-True Rule

⋮	⋮	⋮
m.	$F\ \varphi \to \psi$	
⋮	⋮	⋮
n.	$T\ \varphi$	m
n+1.	$F\ \psi$	m

(a) Implication-False ($\to F$)

1.	$T\ B$	Premise
2.	$F\ A \to B$	Conclusion
3.	$T\ A$	2
4.	$F\ B$	2
5.	$\bot$	1,4

(b) $B \vdash A \to B$

Figure 11: Implication-False Rule

⋮	⋮	⋮
m.	$T\ \forall x \varphi$	
⋮	⋮	⋮
n.	$T\ \varphi^x_t$	m

t is substitutable for x in φ

(a) Universal-True ($\forall T$)

1.	$T\ \forall x(H(x) \to M(x))$	Premise
2.	$T\ H(s)$	Premise
3.	$F\ M(s)$	Conclusion
4.	$T\ H(s) \to M(s)$	1
5.	{ $F\ H(s)$	4
6.	$\bot$	2,5 }
7.	{ $T\ M(s)$	4
8.	$\bot$	7,3 }

(b) $\forall x(H(x) \to M(x)), H(s) \vdash M(s)$

Figure 12: Universal-True Rule

The **existential-false rule ($\exists F$)** is shown in Figure 15a, in which we apply $\exists F$-rule to signed formula $F\ \exists x \varphi(x)$ in line m and obtain, in line n, a signed formula $F\ \varphi^x_t$, where t is substitutable for x in φ. Figure 15b illustrates the use of this rule to $F\ \exists x P(x)$ in line 4 to conclude $P(a)$ in line 5.

⋮	⋮	⋮
m.	$F\ \forall x\varphi$	
⋮	⋮	⋮
n.	$F\ \varphi^x_a$	m

a is new variable

(a) Universal-False ($\forall F$)

1.	$T\ \forall x(H(x) \to M(x))$	Premise
2.	$T\ \forall x H(x)$	Premise
3.	$F\ \forall x M(x)$	Conclusion
4.	$F\ M(a)$	3
5.	$T\ H(a)$	2
6.	$T\ H(a) \to M(a)$	1
7.	{ $F\ H(a)$	6
8.	$\bot$	5,7 }
9.	{ $T\ M(a)$	6
10.	$\bot$	9,4 }

(b) $\forall x(H(x) \to M(x)), \forall x H(x) \vdash \forall x M(x)$

Figure 13: Universal-False Rule

⋮	⋮	⋮
m.	$T\ \exists x\varphi$	
⋮	⋮	⋮
n.	$T\ \varphi^x_a$	m

a is new variable

(a) Existential-True ($\exists T$)

1.	$T\ \forall x(H(x) \to M(x))$	Premise
2.	$T\ \exists x H(x)$	Premise
3.	$F\ \exists x M(x)$	Conclusion
4.	$T\ H(a)$	2
5.	$F\ M(a)$	3
6.	$T\ H(a) \to M(a)$	1
7.	{ $F\ H(a)$	6
8.	$\bot$	4,7 }
9.	{ $T\ M(a)$	6
10.	$\bot$	5,9 }

(b) $\forall x(H(x) \to M(x)), \exists x H(x) \vdash \exists x M(x)$

Figure 14: Existential-True Rule

4 Analytic Tableau Proof Assistant (ANITA)

The ANITA[2] proof assistant, *Analytic Tableau Proof Assistant*, is a tool written in Python that can be used as a *desktop* application, or in a web platform[3]. The main idea is that the students can write their proofs as similar as possible to what is available in the textbooks and to what the students would usually write on paper. ANITA allows the students to automatically check whether a proof in the analytic tableaux is valid. If the proof is not correct, the tool will display the errors on the proof. So, the students may make mistakes and learn from the errors. The web interface is very easy-to-use and has:

- An area for editing the proof in plain text. The students should write a proof in Fitch-style pre-

[2]ANITA source-code is available at `https://github.com/daviromero/anita` under a MIT License.
[3]ANITA is available at: `https://sistemas.quixada.ufc.br/anita/en/`

⋮	⋮	⋮
m.	$F\ \exists x\varphi$	
⋮	⋮	⋮
n.	$F\ \varphi^x_t$	m

t is substitutable for x in φ

(a) Existential-False ($\exists F$)

1.	$T\ P(a)$	Premise
2.	$T\ \exists x P(x) \to B$	Premise
3.	$F\ B$	Conclusion
4.	{ $F\ \exists x P(x)$	2
5.	$F\ P(a)$	4
6.	$\perp$	1,5 }
7.	{ $T\ B$	2
8.	$\perp$	7,3 }

(b) $P(a), \exists x P(x) \to B \vdash B$

Figure 15: Existential-False Rule

sented in Section 3.

- A message area to display whether the proof is valid, the countermodel, or the errors on the proof.
- And the following *links*: *Check*, to check the correctness of the proof; *Manual*, to view a document with the inference rules and examples; *LaTeX*, to generate the LaTeX code[4] of the trees from a valid proof; *Latex in Overleaf* to open the proof source code directly in Overleaf[5].

To facilitate the writing of the proofs, we made the following conventions in ANITA:

- The Atoms[6] are written in capital letters (e.g. A, B, H(x));
- Variables are written with the first letter in lowercase, followed by letters and numbers (e.g. x, x0);
- Formulas with $\forall x$ and $\exists x$ are represented by *Ax* and *Ex* ('A' and 'E' followed by the variable x). For instance, Ax (H(x)−>M(x)) represents $\forall x\ (H(x) \to M(x))$.
- Figure 16 shows the equivalence of logic symbols and those used in ANITA.
- The order of precedence of quantifiers and logical connectives is defined by $\neg, \forall, \exists, \wedge, \vee, \to$ with right alignment. For example:
  - Formula ∼A&B−>C represents formula $(((\neg A) \wedge B) \to C)$;
  - The theorem ∼A|B |− A−>C represents $((\neg A) \vee B) \vdash (A \to C)$.
- Each inference rule will be named by its respective connective and the truth value of the signed formula. For example, &T represents the and-true rule. Optionally, the rule name can be omitted.
- The justifications for the premises and the conclusion use the reserved words pre and conclusion, respectively.

Symbol	$\neg$	$\wedge$	$\vee$	$\to$	$\forall x$	$\exists x$	$\perp$	branch	$\vdash$
LaTeX	\lnot	\land	\lor	\rightarrow	\forall x	\exists x	\bot	[.]	\vdash
ANITA	∼	&	|	−>	Ax	Ex	@	{}	|−

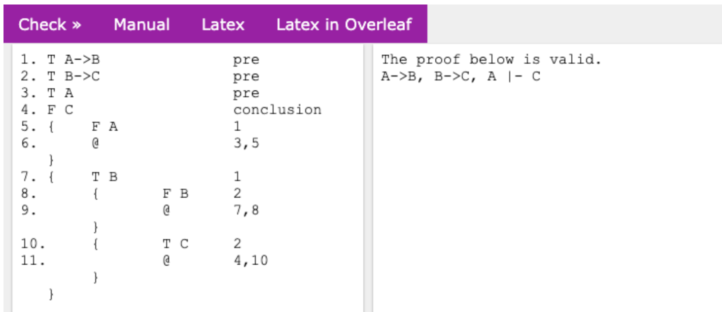
Figure 16: Equivalence between the symbols of logic, ANITA and LaTeX

[4]Use the *qtree* package in your LaTeX code.

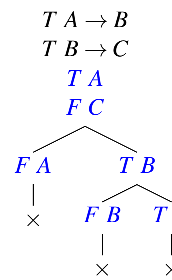
[5]Overleaf is a collaborative platform for editing LaTeX. Available at: `http://overleaf.com/`

[6]An atomic formula or atom is simply a predicate applied to a tuple of terms; that is, an atomic formula is a formula of the form $P(t_1, \ldots, t_n)$ for P a predicate, and the t_n terms.

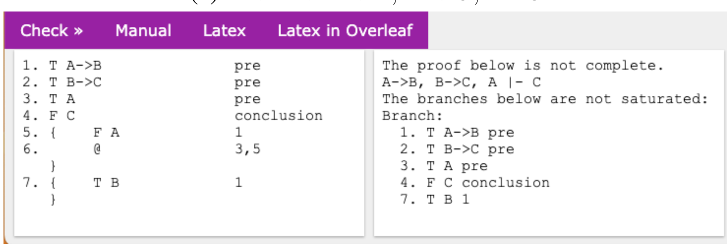
Figure 17a shows a valid proof of $A \to B, B \to C, A \vdash C$ in ANITA, and Figure 17b shows the tree generated by ANITA, where the blue nodes (signed formulas) point out the closed branches. Figure 17c illustrates an example of an incomplete proof of $A \to B, B \to C, A \vdash C$ in ANITA, whereas Figure 17d displays the open branch in red of the analytic tableau that was generated by ANITA.



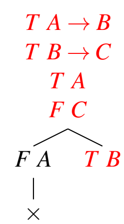
(a) ANITA: $A \to B, B \to C, A \vdash C$



(b) Analytic Tableau Proof



(c) Open Branch in ANITA: $A \to B, B \to C, A \vdash C$



(d) Analytic Tableau Proof

Figure 17: ANITA: $A \to B, B \to C, A \vdash C$

Figure 18 shows a message that the existential-true rule is not applied correctly to signed formula $T\ Ex\ H(x)$ in line 1 to obtain $T\ H(a)$ in line 4, because the term a is not a new variable (see line 3).

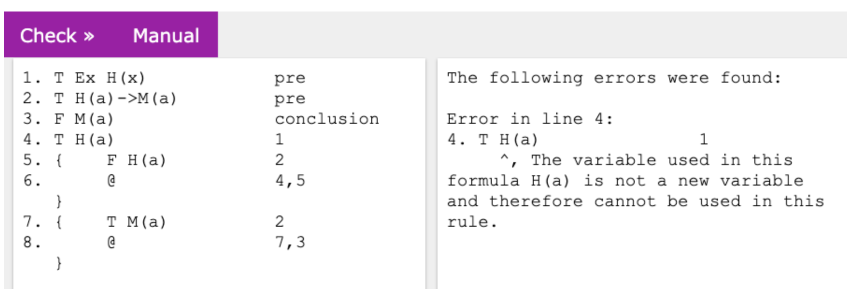
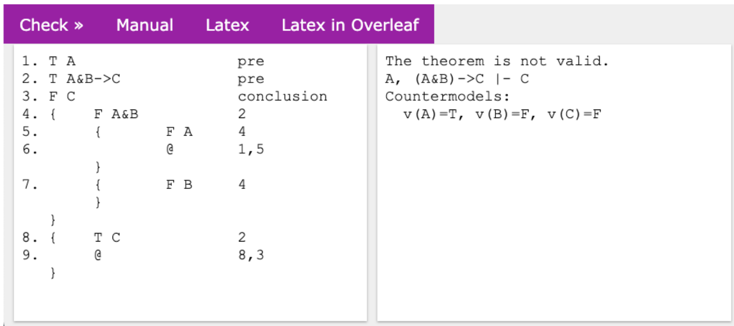
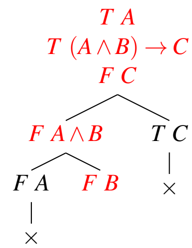


Figure 18: ANITA: An error in a proof

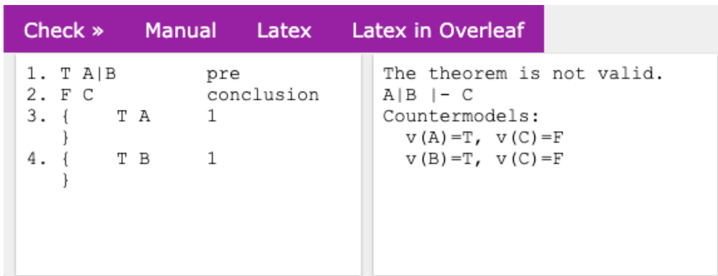
Figure 19a presents a proof that $A, A \wedge B \to C$ does not entail C, and ANITA displays the countermodel of the proof. Figure 19b displays the saturated branch in red of the analytic tableau that provides a countermodel that was generated by ANITA. Figure 19d displays two saturated branches in red of the analytic tableau that provide countermodels of the proof $A \vee B \nvdash C$ that were generated by ANITA, see Figure 19c. Note that in the open branch (lines 1, 2 and 3) the atomic formula B does not occur, then $v(B)$ can be T or false F, and the countermodel is displayed by $v(A) = T, v(C) = F$.



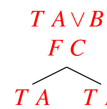
(a) ANITA: $A, A \wedge B \rightarrow C \nvdash C$



(b) Analytic Tableau Proof



(c) ANITA: $A \vee B \nvdash C$



(d) Analytic Tableau Proof

Figure 19: ANITA: Sample of countermodel

5 Related Work

In this article, we focus on Natural Deduction and Analytic Tableau proof assistants. Although, there are proof assistants for other systems, such as SeCaV [5]. We summarize the features of proof assistants, as well as highlight the similarities and differences between these tools and the proposal in this work. We also provide the proof of $A \rightarrow B, B \rightarrow C, A \vdash C$ in each proof assistant.

- The Jape[7] [1] is a desktop proof assistant to write proofs in Fitch-style in Natural Deduction. The proofs are performed by inserting the inference rules through its GUI.
- The ProofWeb [10] is a web interface that intends to be an evolution of JAPE and uses Coq[8] that is state-of-art proof assistant for writing mathematical proofs. The user must write the proofs in a text area or use the GUI to add the inference rules. The ProofWeb can display proofs in Fitch or Gentzen-styles.
- The Panda [6] is also a desktop proof assistant which differs from the previous ones by allowing the writing of proofs in Gentzen-style from its GUI.
- The NaDeA[9] [15] is a web proof assistant for Natural Deduction with a formalization in Isabelle. The user must write the proofs through its user interface which is based on clicking.
- The NADIA[10][11] [14] is a web proof assistant for Natural Deduction, in Fitch-style. NADIA allows

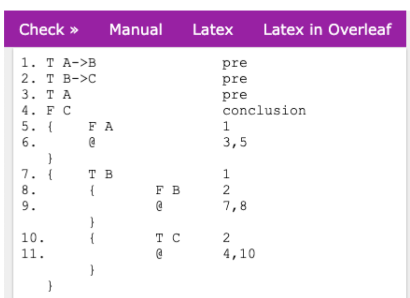
[7] Jape source-code is available at `https://github.com/RBornat/jape/`
[8] Coq is available at `https://coq.inria.fr/`
[9] Available at `https://nadea.compute.dtu.dk/`
[10] Available at `https://sistemas.quixada.ufc.br/nadia/`
[11] NADIA source-code is available at `https://github.com/daviromero/nadia` under a MIT License.

students to write their proofs as closely as possible to the proofs they take on paper, by using an input syntax code similar to [8]. NADIA displays proofs in Fitch or Gentzen-style.

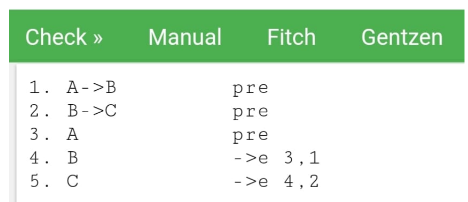
- The Carnap.io[12] [9] is a free and open-source Haskell framework for creating and exploring formal languages, logics, and semantics. A web proof assistant for Analytic Tableaux[13] is available and can be used to construct proofs by using the GUI interface.

- The Tree Proof Generator Tableau[14] is a tableau prover for classical propositional and first-order logic, as well as some modal logics. The prover is written in Javascript and runs entirely in the browser. The user can enter a formula of standard propositional, predicate, or modal logic and the prover will automatically try to find either a countermodel or a tree proof.

ANITA is very similar to NADIA. Both systems receive as input a text with a proof of a theorem and check whether the proof is correct or not. If not, the tools display the errors found. The main difference between the proof assistants is that ANITA accepts proofs in Analytic Tableaux (see Figure 20a) and NADIA in Natural Deduction (see Figure 20b). The parser of the proofs in ANITA is completely different from the NADIA parser, as each implements a very different set of inference rules.



(a) ANITA: $A \rightarrow B, B \rightarrow C, A \vdash C$

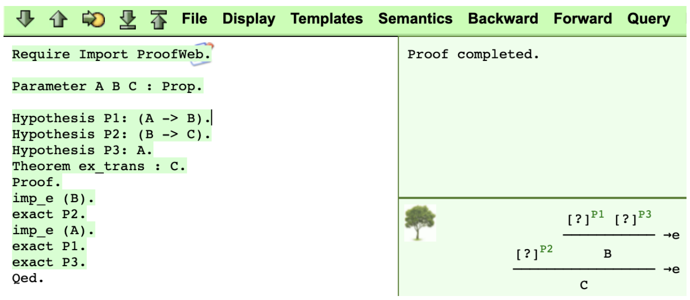


(b) NADIA: $A \rightarrow B, B \rightarrow C, A \vdash C$

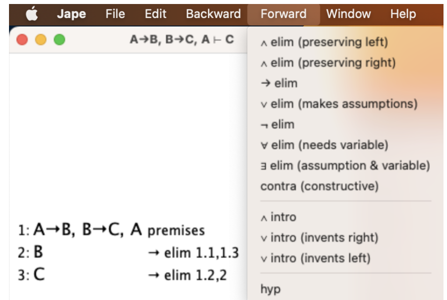
Figure 20: ANITA and NADIA Proof Assistants

Figure 21a presents a proof in ProofWeb. Note that the student has to learn a new syntax that differs a lot from what the student would write on paper. On the other hand, the proofs in Jape (Figure 21b), Panda (Figure 21c), NaDeA (Figure 21d), and Carnap.io (Figure 22a) are carried out by the GUI and the user should click on the menu to add each inference rule. The proof generator, in fact, is a prover instead of proof assistant. So, the user can only interact with the tool to enter the theorem to be get either a countermodel or a tree proof that it is not very useful in order to teach how to use the inference rules.

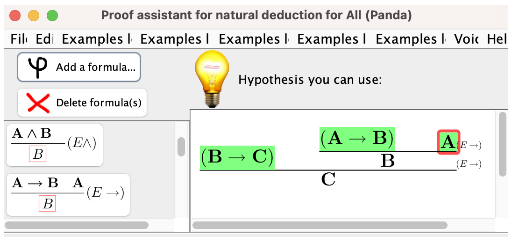
[12] Available at `https://carnap.io`
[13] Available at `https://carnap.io/srv/doc/truth-tree.md`
[14] Available at `https://www.umsu.de/trees`



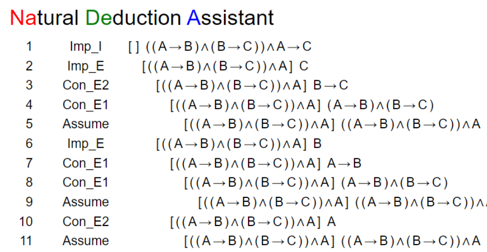
(a) ProofWeb: $A \to B, B \to C, A \vdash C$



(b) JAPE: $A \to B, B \to C, A \vdash C$

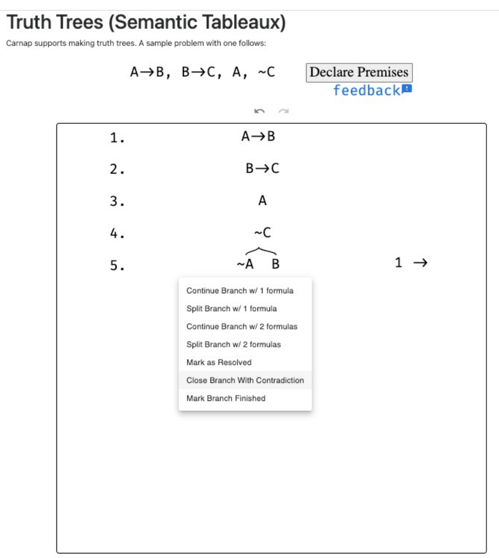


(c) Panda: $A \to B, B \to C, A \vdash C$

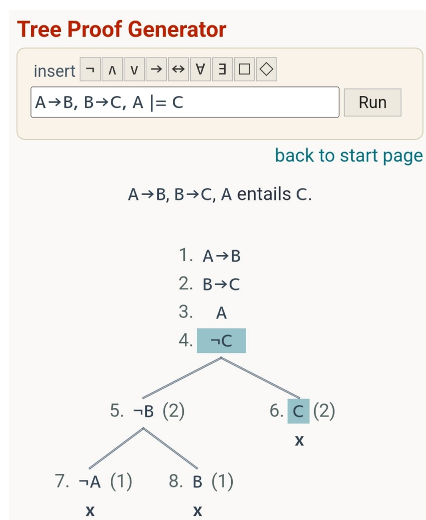


(d) NaDeA: $A \to B, B \to C, A \vdash C$

Figure 21: Natural Deduction Tableau Proof Assistants



(a) Carnap.io: $A \to B, B \to C, A \vdash C$



(b) Proof Generator: $A \to B, B \to C, A \vdash C$

Figure 22: Analytic Tableau Proof Assistants

Below we summarize the assistant proofs regarding to: the deductive systems (ND for Natural Deduction, AT for Analytic Tableaux); the display of the proof-style (F for Fitch-style, G for Gentzen-style); The input proof-writing (GUI for based on clicking in the GUI interface, PT for plain text).

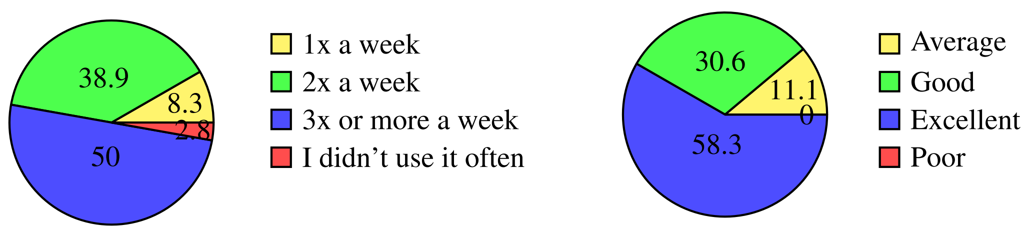
	ProofWeb	Jape	Panda	NaDeA	NADIA	ANITA	Carnap.io	Proof Gen.
Deductive Systems	ND	ND	ND	ND	ND	AT	AT	AT
Display Proof-Style	F, G	F	G	F	F, G	F	T	T
Input Proof-Writing	GUI, PT	GUI	GUI	GUI	PT	PT	GUI	GUI

6 Evaluation of ANITA

In this section, we present the results of the evaluation of ANITA that were carried out in two classes of Logic in Computer Science in 2022 at the Federal University of Ceará at Quixadá Campus. Each class has 4 hours of class per week and a total of 16 weeks. The classes had a total of 74 students enrolled.

6.1 Student Evaluations of ANITA

In total 36 out of 74 registered students answered the anonymous online form (49%). 100% of the students stated that they used ANITA as a study tool and considered that ANITA helped to exercise the content. 91.7% considered ANITA very easy-to-use. Figure 23a shows how often did the students use ANITA and Figure 23b shows how they rate ANITA error messages.



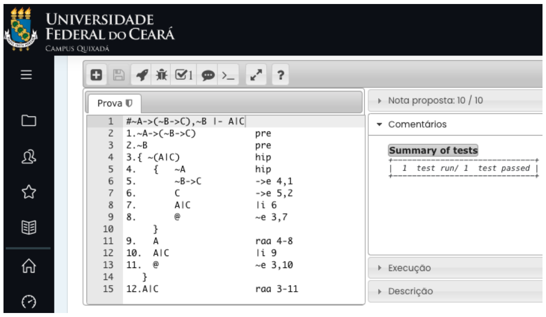
(a) How often did you use ANITA? (b) How do you rate ANITA error messages?

Figure 23: Evaluation

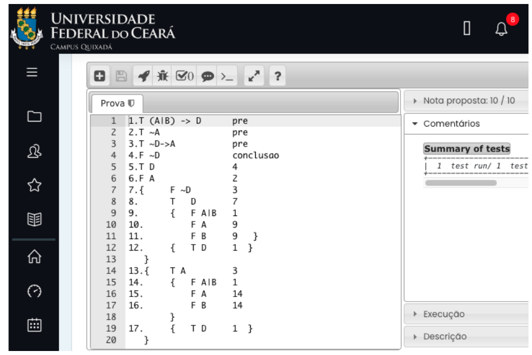
6.2 Evaluation

We used NADIA and ANITA, integrated in the Moodle platform[15], in the second partial evaluation (AP2), which was applied in the laboratory and had four theorems to be proved in Natural Deduction (ND) and four in analytic Tableaux (AT), each item was worth 1.25. The students wrote down the proofs of each theorem in the Moodle platform and checked automatically, by ANITA and NADIA, whether each proof was correct. For instance, Figure 24a displays the answer of a student of Question 1 of ND in the Moodle platform. Figure 24b displays the answer of a student of Question 4 of AT in the Moodle platform.

[15]Moodle is available at `https://moodle.com/`



(a) Moodle (NADIA) - Answer of Question 1: $\neg A \to (\neg B \to C), \neg B \vdash A \vee C$



(b) Moodle (ANITA) - Answer of Question 4: $(A \vee B) \to D, \neg A, \neg D \to A \nvdash \neg D$

Figure 24: AP2 - Samples of questions of NADIA and ANITA in the Moodle Platform

In total 48 out of 74 registered students did the test (65%). The students got a mean (M) of 6.80 with standard deviation (SD) of 3.28. For ND questions, they got 2.76 (MND) and 4.04 (MAT) for AT. 63% (AND) answered the ND questions, of which 87% (RND) got the questions right; 92% (AAT) answered from AT and 88% (RAT) of those answered the questions correctly. Table 1 presents the results by class.

Class	Students	M	SD	MND	SDND	AND	RND	MAT	SDAT	AAT	RAT
A	20	7.31	2.93	2.94	2.08	65%	90%	4.38	1.43	94%	93%
B	28	6.44	3.53	2.64	2.03	62%	85%	3.80	1.82	91%	84%
A+B	**48**	**6.80**	3.28	**2.76**	2,05	63%	87%	**4.04**	1.66	92%	88%

Table 1: Results by Class

7 Conclusion and Future Work

ANITA has been used for teaching analytic tableaux to computer science students. We have compared ANITA as a tool for teaching logic to other tools. From the evaluation point of view, ANITA has been a success in our courses. 49% of the students answered an anonymous online form, in which: 100% consider that the tool helped to exercise the content; 91% consider the tool easy-to-use (excellent or good); 90% used the tool two or more times a week; and 90% considered the understanding of messages as Excellent or Good. We used ANITA, integrated in the Moodle platform, in the partial evaluation. In total 48 out of 74 registered students did the test (65%). 92% of the students submitted their proofs to 4 theorems and of these 88% got the questions right.

As future work, we consider developing more teaching materials for ANITA and making further evaluations of ANITA as a tool for teaching logic.

Acknowledgements: This work is partially supported by the project 04772314/2020/FUNCAP.

References

[1] Richard Bornat & Bernard Sufrin (1996): *Jape's quiet interface*. *User Interfaces for Theorem Provers UITP'98.*

[2] M. D'Agostino, D.M. Gabbay, R. Hähnle & J. Posegga (1999): *Handbook of Tableau Methods*. Springer Netherlands, doi:10.1007/978-94-017-1754-0.

[3] Dirk van Dalen (2013): *Logic and Structure (5th Ed.)*. Springer London, London, doi:10.1007/978-1-4471-4558-5.

[4] Melvin Fitting (1996): *First-Order Logic and Automated Theorem Proving (2nd Ed.)*. Springer-Verlag, Berlin, Heidelberg, doi:10.1007/978-1-4612-2360-3.

[5] Asta Halkjær From, Frederik Krogsdal Jacobsen & Jørgen Villadsen (2022): *SeCaV: A Sequent Calculus Verifier in Isabelle/HOL*. In Mauricio Ayala-Rincon & Eduardo Bonelli, editors: Proceedings 16th *Logical and Semantic Frameworks with Applications*, Buenos Aires, Argentina (Online), 23rd - 24th July, 2021, *Electronic Proceedings in Theoretical Computer Science* 357, Open Publishing Association, pp. 38–55, doi:10.4204/EPTCS.357.4.

[6] Olivier Gasquet, François Schwarzentruber & Martin Strecker (2011): *Panda: A Proof Assistant in Natural Deduction for All. A Gentzen Style Proof Assistant for Undergraduate Students*. In Patrick Blackburn, Hans van Ditmarsch, María Manzano & Fernando Soler-Toscano, editors: *Tools for Teaching Logic*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 85–92, doi:10.1007/978-3-642-21350-2_11.

[7] Gerhard Gentzen (1969): *The collected papers*. North-Holland Publishing Company, doi:10.2307/2272429.

[8] Michael Huth & Mark Ryan (2004): *Logic in Computer Science: Modelling and Reasoning about Systems (2nd Ed.)*. Cambridge University Press, doi:10.1017/CBO9780511810275.

[9] Graham Leach-Krouse (2018): *Carnap: An Open Framework for Formal Reasoning in the Browser*. In Pedro Quaresma & Walther Neuper, editors: Proceedings 6th International Workshop on *Theorem proving components for Educational software*, Gothenburg, Sweden, 6 Aug 2017, *Electronic Proceedings in Theoretical Computer Science* 267, Open Publishing Association, pp. 70–88, doi:10.4204/EPTCS.267.5.

[10] Hendriks Maxim, Cezary Kaliszyk, Femke van Raamsdonk & Freek Wiedijk (2010): *Teaching logic using a state-of-art proof assistant*. *Acta Didactica Napocensia* 3. Available at `https://files.eric.ed.gov/fulltext/EJ1056118.pdf`.

[11] F.S.C. da Silva, M. Finger & A.C.V. de Melo (2006): *Lógica para computação*. Cengage Learning. Available at `https://books.google.com.br/books?id=w27uOgAACAAJ`.

[12] R.M. Smullyan (1995): *First-order Logic*. Dover books on advanced mathematics, Dover, doi:10.1007/978-3-642-86718-7.

[13] J.N. de Souza (2008): *Logica Para Ciencia da Computação*. Campus SBC, Elsevier. Available at `https://books.google.com.br/books?id=Y8GEsUoRKiEC`.

[14] D. R. Vasconcelos, R. Paula & M. V. Menezes (2022): *NADIA - Natural DeductIon proof Assistant*. In: *Anais do XXX Workshop sobre Educação em Computação*, SBC, Porto Alegre, RS, Brasil, pp. 427–438, doi:10.5753/wei.2022.222875.

[15] Jørgen Villadsen, Andreas Halkjær From & Anders Schlichtkrull (2018): *Natural Deduction and the Isabelle Proof Assistant*. In Pedro Quaresma & Walther Neuper, editors: Proceedings 6th International Workshop on *Theorem proving components for Educational software*, Gothenburg, Sweden, 6 Aug 2017, *Electronic Proceedings in Theoretical Computer Science* 267, Open Publishing Association, pp. 140–155, doi:10.4204/EPTCS.267.9.