Reasoning with NL

Gianzina Alexandra i6294354
Grot Leon i6282244
Rahimi Ali i6288408
Vico Gianluca i6183186

Supervisors: Nico Roos and Tjitze Rienstra
Coordinator: Gijs Schoenmakers

Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands

I. Abstract

We address the problem of reasoning with natural language in a way that humans can directly understand. This project is based on the work of Inglot et al. [1]. We extend their controlled natural language by adding new types of expressions. The reasoning is performed with a semantic tableau. The tableau is adapted to work with this controlled natural language and to derive an argument for the conclusion that the tableau reached. Moreover, both the controlled natural language and the tableau can handle defeasible information. A web user interface is used to show that result of the proof to the user. In this way, the reasoning is transparent to the user.

II. Introduction

A. Motivation

Reasoning has always been part of the human mentality and during their lifetime there are many instances that require their inference skills. They use inferential skills in many occasions in life such as making decisions, evaluating hypotheses and concepts, arguing, solving problems and contributing in technological and scientific progress. "An inference is a process of thought that leads from one set of propositions to another" [2]. In all logical models, a conclusion is valid if there is no counter-example to it i.e., an inference is valid if there is no interpretation of the premises that is consistent with negation of the conclusion [2]. The semantic tableau method works in this exact manner [3]. There exist logical computer models such as *ACE* [4] that work with logical representations and symbols. These logical systems provide accurate reasoning, however, they are not explainable to humans without background information in logic. The goal for using natural language directly with semantic tableau method is to provide an explainable reasoning tool that is more comprehensible to humans without knowledge of logic.

B. Background information

1) First order logic and propositional logic: There exist several types of logic, but those relevant for this work are propositional logic and first order logic (FOL).

Propositional logic works with propositions and the relations between them. The propositions are treated as atomic objects and they can be combined with logical operators, such as **and** ($\wedge$), **or** ($\vee$), **not** ($\neg$), **if** ($\rightarrow$) and **if and only if** ($\leftrightarrow$).

FOL can reason about objects. It supports the same operators from propositional logic, but it also includes the **existential** ($\exists$) and **universal** ($\forall$) quantifiers.

2) Controlled natural language: The English language is expressive, but it does not have a fixed structure and it can be ambiguous. This makes interpreting the meaning of a sentence more complicated. Therefore, a controlled natural language (CNL) is a subset of the English language and it used to limit its complexity. The advantage is that an automatic system can more easily work with a CNL than with the English language.

3) Semantic tableaux and argumentation tableaux: A semantic tableau [3] is a graph-like structure, often a tree, used to as proof system for different kind of logic [5], including proposition and predicate logic [3]. Each node of the tree contains a set of logical propositions. The tableau is expanded by removing an operator from one of the propositions and by generating the child nodes according to the specific rules [3]. This method can be used to prove a consequence of the given information by refutation. A limitation of this method is that is does not automatically provide an explicit argument for the conclusion other than the tree itself, and that it does not directly work on natural languages.

Roos proposes an argumentation tableau [6], i.e. a semantic tableau capable of deriving an argument for the claim being proved. This tableau can work with propositional and predicate logic, but it can also handle defeasible information.

C. Problem statement and research questions

The problem that this projects addresses is to do reasoning as close as possible to natural reasoning using the semantic

tableau method on natural language. Our focus is not only to present a reasoning result but to make the reasoning process comprehensible for humans by providing an argument for the proof.

This leads to the following research questions:

1) **In which ways can the controlled natural language implemented by Inglot et al. [1] be extended? How can the different language structures (e.g., conjunctions) be handled?**

   The CNL is constrained to the abilities of the application to parse and apply reasoning to natural language. Our work extends the work of our predecessor project [1] and, by following their approach, we can use regular expressions to detect grammatical patterns and structures. We show that by hard coding these patterns with their following logical rules, more structures could be supported.
2) **Can we extend the semantic tableau method in a way that it gives natural language arguments about why it came to a conclusion? Can we draw conclusions and derive arguments with defeasible information?** By using Roos' argumentation tableau[6], it is possible to derive a supporting argument for each proposition in the tableau. The CNL has been extended to include sentences about uncertainty (e.g. "Birds usually fly"). Then, the argumentation tableau can handle defeasible logic.
3) **How can we visualise the proofs generated by the system to make the reasoning transparent to the user?** Inglot et al. [1] use a tree to display their semantic tableaux for (controlled) natural language. This can be extended by showing the arguments supporting each statement. Moreover, when the proof involves multiple tableaux, it should be possible to visualise all of them in the correct order and with the intermediate conclusions and arguments.

D. Related Work

The project Attempto ACE [4] is language and a reasoner able to test if a statement is entailed by the given information and answer queries about the given information. It defines a language similar to the English language. However, in contrast to our project, this proving system is not based on the semantic tableau method. Moreover, it works on the parsed sentences, instead directly using the CNL and it has a fixed vocabulary. ACE can work with words that are not in its vocabulary, but in this case the grammatical category of the word needs to be explicitly given to the reasoner, making the language less accessible to people with no background knowledge. Other limitations of original Attempto ACE is that it is not able to handle defeasible logic, however, there are extensions to work with defeasible information [7].

Another work similar to ours has been conducted by Lash Abzianidze [8]. They extends the work of Muskens [9] on a tableau for natural languages and proposes LangPro [10]. It consists of a parser for the language and a reasoner, similarly to Attempo ACE [4]. This tools uses a parser for Combinatory Categorial Grammar (CCG) to interpret the structure of a sentence. Then, the reasoner uses the parsed sentences to prove whether a given conclusion holds. The proof method that the reasoner uses is an adaptation of the semantic tableau-

Inglot et al. developed a natural language reasoning tool [1]. In order to implement this tool, they constructed a controlled natural language to express FOL statements. More specifically, they identified 6 abstract models of propositions in CNL that they called expressions. Some of their expressions contain complex information, as they may carry another expression internally and link different sub-sentences together. Other expressions, with a simpler structure, only check whether the sentence or sub-sentence has a subject, verb and object. In Figure 1 you can find an example of expression that the research group created.

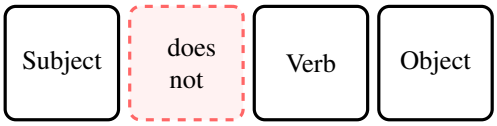

Fig. 1: The structure of Base Expression that the previous group implemented. The block "does not" is optional and it can be used to express a negation of a statement.

Then, they formulated a number of logical rules for a semantic tableau that can be applied to their CNL, using as model the rules for the semantic tableaux for propositional logic and FOL.

They then proceeded to implement a web application to use their tableau interactively. When entering input sentences in the user interface of the application, a parser is responsible for the tokenization of the sentences. Each expression contains specific keywords, which the parser can recognise with the help of the regular expressions. Afterwards, the tableau reasoner takes the parsed sentences and creates a semantic tableau tree [3]. The reasoner tries to close all the branches of the tree (i.e. it finds a contradiction in all the branches), so as to prove or refute the conclusion. Finally, it displays the result and the user can visually see the tree and extract information about the rules that were applied. Figure 2 demonstrates the Inglot et al. Application Workflow.

The advantage of this system is that the nodes of the tableau contain a list of natural langugae (NL) expressions, which are directly interpretable by the user. Although, the system shows whether the conclusion is entailed by the given information and provide a semantic tableau supporting or disproving the statement, it is not understandable for people not familiar with the semantic tableau method. Moreover, this system does not handle defeasible rules and due to the limitations of the CNL some expressions need to be rephrased in order to be usable.

III. Methodology

As already mentioned, we are extending the implementation of the Inglot et al. research group [1]. First we are going to

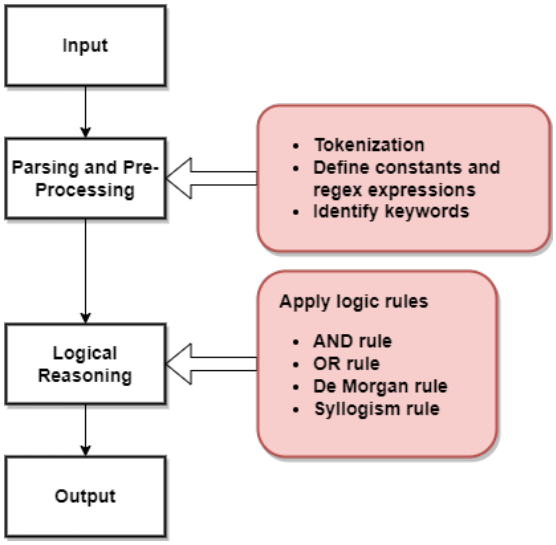

Fig. 2: Inglot et al. [1] Application Workflow

describe how the sentences are parsed and processed. To begin with, the application starts by tokenizing the input sentences. Some special keywords and sentences are predefined in order to detect the structure (e.g. and, or, if, then, are, is, unless, if and only if, to, of, doesn't, does not, it, etc...). The system does not handle temporal logic, therefore verb tenses are ignored. Any sentence that contains from three words and up to five words ,except from the special keywords that are not counted, is considered a valid sentence. For instance if we have a three word sentence, "Mary loves dogs" and the same sentence in Latin "Maria amat canibus", then both are valid sentences made up of a subject, a verb, and an object. In case of a four-word sentence and a five-word sentence the requirement is the sentence to have a preposition. Keywords such as "does not", "do not" and "it is not the case that" has been used to define negation. It is necessary that any sentence can be negated, since negation is a necessary operation in the semantic tableaux method. The word "not" has been used for this purpose. The grammar is simplified in this case, so the system can generate, for instance, the sentence "Mary loves not dogs" when negating "Mary loves dogs". After tokenization, we apply logical rules on sentences and generate an argumentation tableaux tree, as described later in Section III-C, and present the tree in a web interface.

A. Semantic Tableaux

A semantic tableau [3] is a tree-like structure used to as proof system for different kind of logic [5], including proposition and predicate logic [3]. Each node of the tree contains a set of logical propositions. The tableau is expanded by removing an operator from one of the propositions and by generating the child nodes according to the specific rules [3]. When working with First Order Logic, additional rules are needed to handle the universal and the existential quantifiers. This method can be used to prove a consequence of the given information by refutation. Our reasoner uses an extension of semantic tableaux method that works directly on controlled natural language sentences. Figure 3 show an example of a semantic tableau for propositional logic.

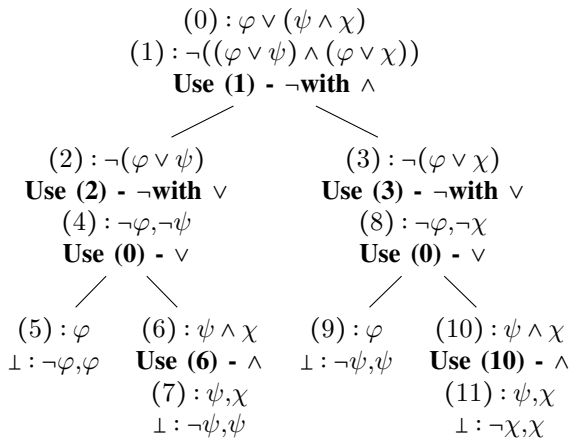

Fig. 3: Examples of a semantic tableau for propositional logic. The information is $\varphi \vee (\psi \wedge \chi)$ and the goal is to prove that $(\varphi \vee \psi) \wedge (\varphi \vee \chi)$ holds. This is done by showing that the information and the negation of what we are trying to prove lead to a contradiction.

B. Controlled Natural Language

We limit the language by allowing only predefined structures for each sentence. We identified the following classes of expressions, based also on the work of Inglot et al. [1]:

- **Base Expressions** [1]: Tim plays tennis
- **Base Expressions with Prepositions**: You play tennis with Tim
- **Connecting Expressions** [1]: I read a book and Tim play chess
- **Implicit Connecting Expressions**: Tim plays chess, tennis and football
- **Function Expressions** [1]: Tim is a boy
- **Conditional Expressions** [1]: When it is sunny, Tim plays outside
- **Unless Expressions**: Unless it is raining, Tim plays outside
- **If-and-only-if Expressions** : Tim plays outside if and only if it is not raining
- **Syllogism Expressions** [1]: All horses are animals, All animals are mortal, Therefore all horses are mortal
- **Quantified Expressions** [1]: for all cats, for all mouses, cats are bigger than mouses

The expressions are identified via regular expressions and multiple expressions can be combined. Moreover, every class of expression can be marked as "defeasible" by adding keywords like "usually" and "normally" at the beginning of the sentence. In this way it is possible to express statements that are generally true, but that allow exceptions. For examples, the sentence "Usually, Tim plays outside" is a defeasible **Base Expression**: it is generally the case the "Tim plays outside" but there can be exceptions. Section IV-A shows in details the characteristics of the different expressions.

C. Argumentation tableaux and defeasible logic

An argumentation tableaux is an extension of semantic tableaux that adds support for the propositions and can be used to derive arguments for a conclusion. The support generated in an argumentation tableau is the only information we need to prove the conclusion. For instance, when we have a large database of information, and only five sentences have been used to prove a conclusion, the support will be exactly those five sentences that we needed to prove the conclusion. Moreover, the support allows us to handle defeasible logic, and it could be further used in rebuttal and undercutting attacks in context of multi agent systems. The tableau rules of an argumentation tableaux are the same rules of a traditional semantic tableau, with the difference being the added support. The rules of argumentation tableaux are presented at Figure 4. In each rule, the first line represent a statement in the tableau where S is the support (represented as a set of expression). The second line contains the expressions that are derived from the first line. The "|" symbol indicates that the proof is split in two branches. In the root of the tableau, the support for propositions is the propositions itself unless it is a defeasible information with a support S. When the tableau closes, it is possible to use the union of the support of the leaf nodes as support for the claim being proved.

$$\frac{(\mathcal{S}, \varphi \wedge \psi)}{(\mathcal{S}, \varphi), (\mathcal{S}, \psi)} \qquad \frac{(\mathcal{S}, \varphi \vee \psi)}{(\mathcal{S}, \varphi) \mid (\mathcal{S}, \psi)}$$

$$\frac{(\mathcal{S}, \varphi \rightarrow \psi)}{(\mathcal{S}, \neg\varphi) \mid (\mathcal{S}, \psi)} \qquad \frac{(\mathcal{S}, \varphi \leftrightarrow \psi)}{(\mathcal{S}, \varphi \rightarrow \psi), (\mathcal{S}, \psi \rightarrow \varphi)}$$

$$\frac{(\mathcal{S}, \neg(\varphi \vee \psi))}{(\mathcal{S}, \neg\varphi), (\mathcal{S}, \neg\psi)} \qquad \frac{(\mathcal{S}, \neg(\varphi \wedge \psi))}{(\mathcal{S}, \neg\varphi) \mid (\mathcal{S}, \neg\psi)}$$

$$\frac{(\mathcal{S}, \neg(\varphi \rightarrow \psi))}{(\mathcal{S}, \varphi), (\mathcal{S}, \neg\psi)} \qquad \frac{(\mathcal{S}, \neg(\varphi \leftrightarrow \psi))}{(\mathcal{S}, \neg(\varphi \rightarrow \psi)) \mid (\mathcal{S}, \neg(\psi \rightarrow \varphi))}$$

$$\frac{(\mathcal{S}, \neg\neg\varphi)}{(\mathcal{S}, \varphi)} \qquad \frac{(\mathcal{S}, \varphi), (\mathcal{S}', \neg\varphi)}{(\mathcal{S} \cup \mathcal{S}', \bot)}$$

Fig. 4: The tableau rules of an argumentation tableau for propositional logic [6]. The support of the child nodes is the same of the parent. With the exception of the support, this is identical to the classical semantic tableau.

Defeasible Logic is necessary when we deal with uncertainty. It works with information that is usually *True* and we accept it unless there is a counter argument against it. In this work we distinguish two kind of defeasible statements. The first case is for **Conditional Defeasible Expression** i.e. in formal notation $p \rightsquigarrow q$ (e.g., "Usually if it is raining then the sun is not visible"). The second case is for all the remaining types of expressions (e.g., "Normally it is raining", "usually it is raining or cloudy"). For handling defeasible information, we have explored two approaches, using multiple proofs and reasoning by cases [6]. They are not complete, but some problem can be proved by one method but not by the other (e.g., see Appendix VI-E).

1) Defeasible logic with multiple proofs: In this approach, we first try to eliminate uncertainly from defeasible information. In case of uncertainly in statements that are not **Conditional Expressions**, we try to remove the uncertainly by proving that it cannot be the case that the given proposition does not hold. If it is the case, we accept the proposition as certain information. For example, when we have the defeasible information "usually it is raining", we try to prove "it is not raining" and we accept that "it is raining" only if we cannot prove that "it is not raining". For handling defeasible conditional statements, i.e, "usually if *antecedent* then *conclusion*", first we try to prove that it is not the case that ¬*antecedent* holds, i.e. *antecedent* holds, and if it is the case, we accept *conclusion* with the arguments that has been generated in the argumentation tableaux that we used to prove the *antecedent*. Furthermore, we add the *conclusion* with its support S to the root of a new tableau, and we do this step recursively for all defeasible information until no further defeasible information can be accepted.

2) Defeasible logic with reasoning by cases: In reasoning by cases, conditional defeasible rules are handled inside the tableau as an additional tableau rule presented in Figure 5.

Defeasible Rule

$$\frac{(\{S'\}, \varphi \rightsquigarrow \psi), (\{S\}, \varphi)}{(\{S' \cup S\}, \psi)}$$

Fig. 5: Defeasible Rule for Reasoning by cases. When the premise of a defeasible expression is proved inside a particular node, the conclusion is immediately added. The difference between a normal $p \rightarrow q$ and a defeasible $p \rightsquigarrow q$ rule is the Support.

A key difference between this method and multiple proofs is that in this method we can use and apply a conditional defeasible expression only in some branches, while in multiple proof method the *antecedent* of the conditional rule is proved therefore it holds in all of the cases. A limitation of the current implementation is that for **Conditional Defeasible Expressions** ($p \rightsquigarrow q$) we need the proposition p in the node to apply the rule, while the consensus in defeasible logic is that when we do not have $\neg p$ in a node, we should be able to use the defeasible rule. However, as the tableau tree expands, we might derive $\neg p$ in a node, thus, we only accept the rule when $\neg p$ does not holds i.e. p holds.

D. Inconsistencies

1) Inconsistencies in the information: Inconsistencies in the Information I happens when the tableau with Information I closes all the branches and thus any claim can by valid. In this case, we present an inconsistency claim S with a closed argumentation tableau to the user. The claim S is a subset of

I and in fact is a Support for the claim *True*, thus it can be used to generate a closed tableau and prove a tautology. For instance, when we have "it is raining", and "it is not raining", we have a contradiction that needs to be resolved. In case of inconsistencies in certain information, we simply ask the user to resolve the inconsistencies. An example of a inconsitency is illustrated at Appendix VI-H.

2) Inconsistencies in defeasible information: There are different approaches to handle inconsistencies in defeasible logic. Here we have investigated the following cases. To handle inconsistencies in reasoning by case, we introduce **Defeated defeasible propositions**. A defeated defeasible proposition p is a defeasible proposition that can be disproved with a set of certain information I. In other words, I is the support for the claim $\neg p$, thus the information I defeats the defeasible proposition p (e.g., see Appendix VI-I). Furthermore, after identifying defeated defeasible propositions, we simply remove the defeated defeasible proposition from the information and continue to try to prove the conclusion.

Moreover, multiple defeasible statements can lead to contradiction when combined (e.g., "it is usually raining", "usually, if it is sunny, it is not raining", and "it is sunny"). Depending on the order in which the statements are inserted in the reasoner, it is possible to obtain different results. To avoid ambiguities, the statements are given to the reasoner in the same order used by the user. In this way, the deafeasible statements are sorted from the "strongest" to the "weakest", and a stronger statement can cancel a weakest one if inserting both in the tableau leads to a contradiction. For example, the fist statement is "Usually, it is raining" and the second is "Usually, it is not raining". The first statement is inserted in the tableau (assuming that "it is not raining" does not hold), but the second one is not inserted, since it is contradicted by a stronger statement.

E. User Interface

One of the main goals of this and the predecessor project [1] is to make the natural language reasoning easy to use and understand by humans. In order to support this, a web interface was developed by the predecessor project to allow users to easily use the tool with as few technical barriers as possible. We evaluated that decision and came to the conclusion the we were going to continue using that web interface, do some necessary major technical refactoring and then adapt it to the new functionalities that we added to the reasoner.

The basic features of the web interface are checking whether a sentence matches our CNL and doing reasoning by entering a set of premises and a conclusion to check. The system does the proof and then shows the result and gives all the available and useful information explaining it. A more detailed description of the different features accompanied by some screenshots can be found as part of the results section at IV-C.

IV. Results

A. CNL

Inglot et al. [1] constructed a controlled natural language in order to express FOL statements. They identified a number of abstract models of propositions in CNL and called them expressions. Each expression is characterised by a strict structure and by a set of rules to derive new information in the semantic tableaux. In this section we illustrate the extentions on the CNL carried out by our group.

1) Base Expressions with Prepositions: This is an extension of the **Base Expression** [1] (i.e. an expression with subject, verb and object as indicated in Figure 1). The **Base Expression with Prepositions** consists of subject, verb and object, as the **Base Expression**, but it also adds a preposition and a second object. Figure 6 shows the structure of a **Base Expression with Prepositions**.

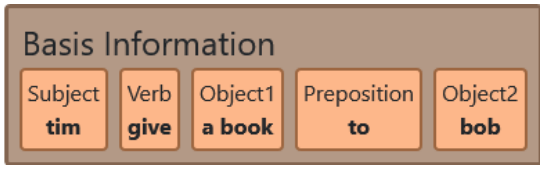

Fig. 6: Structure of a Base Expression with Prepositions. The original sentence, "Tim gives a book to Bob", is split in subject, verb, object, preposition and another object.

The negation can be expressed by using a negated verb or by adding "it is not the case that" at the beginning of the sentence [1]. Moreover in the tableaux, this kind of expression behave exactly the same as a **Base Expression**, and it is treated like an atomic proposition in propositional logic. Furthermore in some cases, it is possible to derive a generalised **Base Expression** from a **Base Expression with Preposition**. For example, "Alice gives a book to Bob" implies that "Alice gives a book", but "it is rainy in the Netherlands" does not imply that "it is rainy" in general. Therefore, this kind of reasoning needs a deeper understanding of the meaning of a sentence, and it cannot be directly used by our semantic tableaux.

2) Implicit Connecting Expressions: When multiple phrases are connected with the same coordinating conjunction (e.g., "and" and "or"), it is possible to use a comma instead of the conjunction. In this ways, the CNL includes sentences like "Amanda plays the violin , guitar and piano". Figure 7 and 8 shows how the application handles these input sentences and transforms them into new sentences. In the tableau, these kind of expressions are treated as multiple **Connecting Expressions**, i.e. as conjunction or disjunction of multiple propositions.

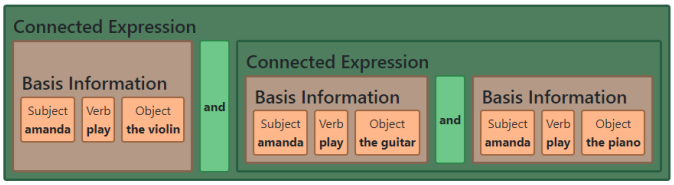

Fig. 7: Structure of a Connected Expression. The original sentence is "Amanda plays the violin , guitar and piano". As we can see, this expression has changed to "Amanda plays the violin and Amanda plays the guitar and Amanda plays the piano".

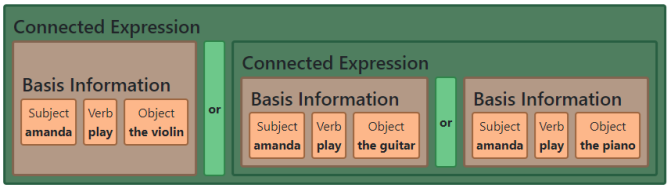

Fig. 8: Structure of a Connected Expression. The original sentence is "Amanda plays the violin , guitar or piano". As we can see, this expression has changed to "Amanda plays the violin or Amanda plays the guitar or Amanda plays the piano".

3) Unless Expressions: The **Unless Expression** is related to the **Conditional Expression** [1] (e.g., "When it is sunny, Tim plays outside"). This kind of expression can be use to describe a statement that holds when another statement does not hold. The unless statement consist of "unless *Expression_1*, *Expression_2*" or "*Expression_1* unless *Expression_2*". Figure 9 shows the structure of this kind of expressions.

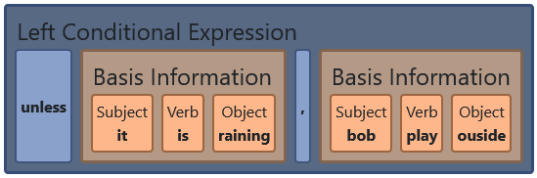

Fig. 9: Structure of an Unless Expression. The original sentence is "Unless it is raining, Bob plays outside". As we can see, this kind of expression combine multiple base expressions ("It is raining" and "Bob plays outside").

In propositional logic, this is interpreted as $\neg\varphi \rightarrow \psi$. The rule for the semantic tableaux for this expression is shown in Figure 10. The **Unless Expression** is decomposed in two expressions. In one case, the condition (i.e. the "unless" part) holds; in the other case, the conclusion holds.

Unless Rule

$$\frac{\textbf{unless } \textit{Expression\_1}\textbf{, } \textit{Expression\_2}}{\textit{Expression\_1} \mid \textit{Expression\_2}}$$

Fig. 10: Rule for the tableau for the Unless Expressions. The first line contains the expressions with the unless keyword. The second line contains the "|" symbol which means that the expressions will be split in two branches. Each branch has a derived expression that will be recursively simplified by the tableau solver.

4) If-and-only-if Expressions: The statement $\varphi \leftrightarrow \psi$ can be expressed in natural language as "*Expression_1* if and only if *Expression_2*" or shorter as "*Expression_1* iff *Expression_2*". This expression is handled by the following rule in Figure 11. In a similar way than in propositional logic, the **If-and-Only-If Expression** is decomposed in two **Conditional Expressions**.

If-and-Only-If Rule

$$\frac{\textit{Exp\_1} \textbf{ if and only if } \textit{Exp\_2}}{\textbf{when } \textit{Exp\_1}\textbf{, then } \textit{Exp\_2}\textbf{, when } \textit{Exp\_2}\textbf{, then } \textit{Exp\_1}}$$

Fig. 11: Rule for the tableaux for the **If-and-Only-If Expression**. The first line contains the expressions with the **if and only if** keyword. The second line shows that the expressions are split in two **Conditional Expressions**. The left **Conditional Expression** consists of "when *Exp_1*, then *Exp_2*" and the right **Conditional Expression** consists of "when *Exp_2*, then *Exp_1*". Exp is referred to the Expression.

5) Syllogism Expressions: Reasoning with natural language was already studied back in 384-322 B.C. with Aristotle's introducing the syllogistic reasoning [11]. Syllogistic reasoning is a logical argument where deductive reasoning is used to reach a conclusion in which two or more premises are claimed to be true. An example of an Aristotelian syllogism is: All X are Y, All Y are Z, Therefore all X are Z. Inglot et al. [1] implemented some rules for the syllogisms that only support "to be" verbs. Our intention is to extend the syllogisms so that we can support various arbitrary verbs. The rules the preceding group applied have been changed and feature the structure as shown in Figure 12. Moreover, in addition to arbitrary verbs, the input can also have prepositions.

Syllogism Rules

$$\frac{\textbf{all } \textit{Sub\_X Verb [Obj] [Prep] Obj\_1}\textbf{ , } \textit{Sub\_N } \textbf{is a(n) } \textit{Sub\_X}}{\textit{Sub\_N Verb [Obj] [Prep] Obj\_1}}$$

$$\frac{\textbf{no } \textit{Sub\_X Verb [Obj] [Prep] Obj\_1}\textbf{ , } \textit{Sub\_N } \textbf{is a(n) } \textit{Sub\_X}}{\textit{Sub\_N } \textbf{do not } \textit{Verb [Obj] [Prep] Obj\_1}}$$

$$\frac{\textbf{some } \textit{Sub\_X Verb [Obj] [Prep] Obj\_1}}{\textit{Sub\_N } \textbf{is a(n) } \textit{Sub\_X}\textbf{ , } \textit{Sub\_N Verb [Obj] [Prep] Obj\_1}}$$

$$\frac{\textbf{some } \textit{Sub\_X } \textbf{do not } \textit{Verb [Obj] [Prep] Obj\_1}}{\textit{Sub\_N } \textbf{is a } \textit{Sub\_X}\textbf{, } \textit{Sub\_N } \textbf{do not } \textit{Verb [Obj] [Prep] Obj\_1}}$$

Fig. 12: Examples of rules for the tableau for the syllogistic reasoning. Specifically, the first rule connects two facts (*Sub_X Verb [Obj] [Prep] Obj_1* **,** *Sub_N* **is a(n)** *Sub_X*) and assigns the Subject of the right fact to the left fact (*Sub_N Verb [Obj] [Prep] Obj_1*). The same logic applies to the other rules. Sub is referred to the Subject. Obj is referred to the Object. Prep is referred to the Preposition. The square brackets indicate that the part of speech is not mandatory. In the following example this applies to the Preposition and the Object.

B. Argumentation tableaux

The argumentation tableau takes a list of sentences as input and a conclusion that need to be proved. Since it is a refutation proof, the conclusion is negated. Then, the list of sentences is split in two separated lists: a list of defeasible sentences and a list of non-defeasible sentences. This is needed, since the two kind of expressions are handled differently.

With the multiple tableaux approach, the tableau solver tries to add the defeasible expressions one by one to the set of non-defeasible information: for the **Conditional Expressions** this is done by proving the antecedents, for the other expressions it is done by showing that the negation of the expression does not hold.

Algorithm 1 Expand defeasible information

```
Input: DefInfo, NonDefInfo
Output: NonDefInfo
  function EXPAND(DefInfo, NonDefInfo)
    for all D ∈ DefInfo do
      if D is WhenExpression then
        Premise, Conclusion = D
        if PROVE(NonDefInfo, Premise) then
          NonDefInfo ← NonDefInfo ∪ Conclusion
          DefInfo ← DefInfo ∖ D
          EXPAND(DefInfo, NonDefInfo)
        end if
      else
        if ¬ PROVE(NonDefInfo, ¬ D) then
          NonDefInfo ← NonDefInfo ∪ D
          DefInfo ← DefInfo ∖ D
          EXPAND(DefInfo, NonDefInfo)
        end if
      end if
    end for
  end function
```

Statements from the defeasible information (DefInfo) are added to the non defeasible information (NonDefInfo) when needed. The method PROVE checks if the given information entails the conclusion.

With this method, multiple tableaux are generated and displayed to the user. Each tableau works like a normal semantic tableau, but it keep track of the support of each statement. When an intermediate tableau used to prove a defeasible statement closes, an argument is generated by the union of the support of the leaf nodes. This argument is used as support for the defeasible statement in the next tableaux. When the last tableau closes, the argument generated supports the statement being proved. In case a tableau does not close, no argument is generated

In the reasoning by cases method only a tableau is created. As mentioned before, **Conditional Defeasible Expressions** and the other defeasible expressions are handled differently. For **Conditional Defeasible Expressions**, when a node is added to the tableaux, the tableaux solver check if the antecedent holds in that particular node. If it holds, then the consequence is added to the node with its proper support S. The other expressions are added directly to the tableau. However, in case of inconsistencies, the tableau tries to remove some of them to solve the inconsistencies. In this way, different branches can use different defeasible statements and the antecedent of the **Conditional Defeasible Expressions** do not need to hold in all the cases.

1) Comparison of two methods: In the first method, by using multiple proofs we try to completely eliminate uncertainty, and in the reasoning by cases method we allow defeasible rules in branches of a tableau node. Furthermore when using reasoning by cases we can handle cases that are not provable by using multiple proof method; for instance, when we have $p \rightsquigarrow t$, $q \rightsquigarrow t$ and $p \vee q$, we can prove t only by the reasoning by cases method (e.g., see Appendix VI-E). Furthermore the multiple proof method can handle complex propositions for antecedent of **Conditional Defeasible Expressions** while the current implementation of the reasoning by cases method only supports atomic propositions for the antecedent. As discussed in this text, both methods have different limitations and none of them are complete.

C. User interface

The natural language reasoner that this project is extending has to be easy to use and provide explanations about the given conclusions in a simple and understandable way. In the previous project [1], a web interface was developed for this purpose using the JavaScript framework Vue.js. We decided to keep the existing web interface in principal, but the underlying technical structure needed a major refactoring in order to be able to keep the system extendable and maintainable with the extensions of the argumentation tableau and the CNL.

1) Language checker: The natural language reasoner user interface provides two main pages and functionalities. The first functionality is the language checker. The language checker is used to verify whether a sentence that is given to the program via an input field is part of the implemented controlled natural language and can therefore be processed. In case of success, the program also splits up the given sentence into the recognised sub-sentences or words. Besides giving the ability to check sentences, the language checker can also be used to better understand how the program parses sentences and consequently performs the reasoning. The language checker was taken over from the predecessor project and only adjusted to the new language aspects and other changes.

2) Natural language reasoner: The main page of the web interface is the natural language reasoner. The reasoner provides a set of input fields to input the sentences that the reasoning should deal with. Premises can dynamically be added and removed in order to support an arbitrary number of premises. Furthermore, there is a field for the sentence to be proven and a selection control to select the proof algorithm. There is also an information card which displays a brief description of the selected proof-algorithm. After proving the given problem, the web interface provides a variety of features to make the reasoning process as transparent as possible.

a) Contradictions in Information: It is possible that the user inputs information that is inconsistent. That occurs when

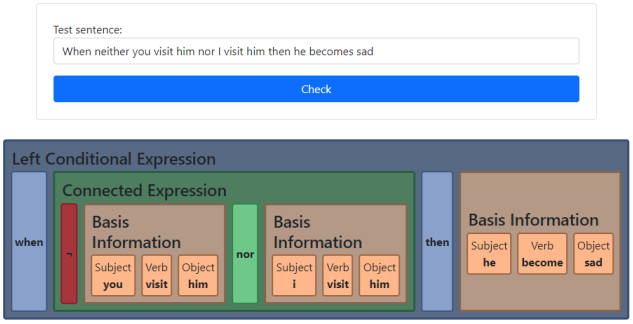

Fig. 13: Screenshot of the language checker with a rather complex recursively nested sentence to check

at least two indefeasible sentences directly contradict each other. In that case, proving is impossible and an error message is displayed. In Figure 14 you can see an example of the web interface output in that case. The web interface displays the statements that are contradicting each other and there is a tableau-proof proving the contradiction.

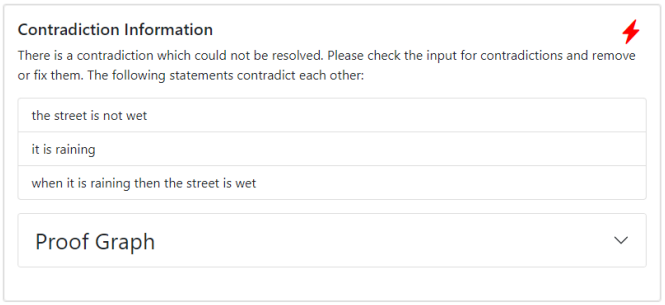

Fig. 14: Card in the web interface displaying that unsolvable inconsistent information was entered

As we added support for defeasible information, contradictions don't always have to be unsolvable. When one of the statements is defeasible and there is a contradiction, that contradiction is allowed as the defeasible expression is meant to be overridden by non-defeasible expressions. In that case of defeated defeasible propositions, the web interface displays an information card with details about the inconsistency claim. The information card consists of the expressions that cause the contradiction and a tableau tree that proves it. In Figure 15 you can see an example of how the web interface deals with inconsistency claims. When expanding the tree graph, an argumentation-tableau-proof for this inconsistency claim is shown. Contrary to the unsolvable contradictions, in this case the reasoning can be executed and a proof for the task is also shown.

b) Conclusion support: This card shows the conclusion support which is additional information that we gained by implementing the argumentation tableau reasoning technique. For content-related insight please use section III-C.

c) Applied rules: The applied rules card gives a detailed insight into the logical rules that were applied during the reasoning process. A rule like that is used to split a rather complex sentence into one or multiple simpler sentences by

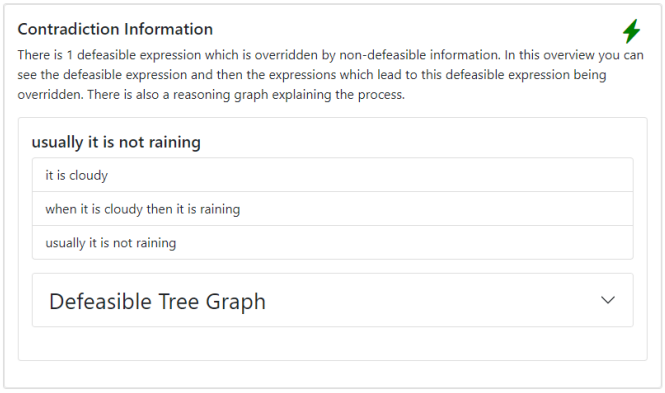

Fig. 15: Card in the web interface displaying an example of defeasible information which was overridden by non-defeasible information

applying a logical rule. The name of the rule and its internal parameters and the resulting expressions which were created by that rule are displayed for every applied rule. This card was taken over from the predecessor project.

Fig. 16: Screenshot of the applied rules card with an and-rule as example

d) Proof-tree: The proof-tree is used in order to give full insight into the used reasoning algorithms by graphically displaying all steps the reasoner takes. While this feature was mostly taken over from the previous project, due to the advanced proof algorithms used in this project, the tree needed to be extended in order to display a support phrase for every statement in the tree. This is indicated by a phrase indicated with an S after every sentence in the nodes.

V. Discussion

A. Addressing the research questions

1) In which ways can the implemented controlled natural language be extended? How can the different language structures can be handled?: We propose five additional class of expressions and an improvement of the **Syllogism Expressions**. Moreover, the CNL can now express defeability, although it has to be mentioned explicitly in a sentence. The tableau solver is able to use the new expressions and handle the

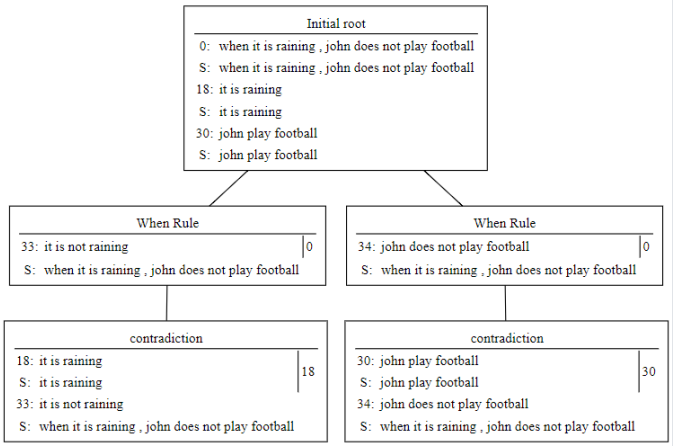

Fig. 17: Example of the tree-visualisation of an argumentation-tableau based proof

defeabile logic. The language is still limited, but it can model propositional logic and partly first order logic. At the moment, the language is still recognised with regular expressions, but it is possible other tools, such a context free grammar to define the language.

2) Can we extend the semantic tableau method in a way that it gives natural language arguments about why it came to a conclusion? Can we draw conclusions and derive arguments with defeasible rules?: We show that it is possible to extend the semantic tableau for natural language of Inglot et al. in order to derive arguments and the system is able to model defeasible information. However, this proof system is not complete, as shown in the examples' section (see Section VI).

3) How can we visualise the proof to make the reasoning transparent to the user?: The algorithmic extensions that were conducted in this project also had an effect on the web interface. The advanced algorithms provide more information to be shown to user on the interface. Mainly, this is about the conclusion support which was added to every expression within a node in the solution tree, but also a combined visualisation of the support sentences that led to the result. While the benefit of displaying the combined support is limited for simple problems in which every statement of the premises is used in the reasoning process, in cases in which a larger number of premises that are not all relevant to the actual proof exist, the conclusion support provides more useful information. In that case, it displays the subset of premises that was actually used to prove the expression to be proven. When thinking about cases such as reasoning based on laws, this could be very useful as it acts as a filter for relevant information when the number of premises is potentially huge. Having a look at one of our research questions which was about improving the explainability of the reasoning result, this feature can be seen as a major advantage for this class of problems as it clearly shows based on which subset of the input it came to a decision (e.g., see Appendix VI-F). Furthermore, in the process of adding defeasible information it became much more common to have contradictions which can be resolved in the premises. To show the user the progress of how these inconsistencies relate to each other, we added a card to the web interface that describes this process by listing only the expressions that led to that conclusion (see section IV-C2a).

B. Limitations and future work

Using this approach, we have found two kind of limitations. The first is the limitation of the CNL and understanding of natural language. The second is a limitation of the logical rules. The CNL is still limited in the number of expressions it can handle. The way it has been extended involves hard coding all the types of expression that should be included. However, this method it is not feasible for complex languages structures and a context free grammar could provide a better structure for a CNL. Furthermore, the negation of sentences are not natural, for instance, "John play not football" is not grammatically correct. However, for handling negation, understanding of temporal logic and different verb tenses is required. For instance, the negation of "John played football" and "John plays football" is different in natural language. Moreover, the parser can sometimes fail to recognise a sentence if there are articles or adjectives. Understanding pronouns or determiners and what they refers to is a hard task and perhaps using this method a insolvable task. For instance, in the sentence "A horse went in to a barn and its head was wet", "its" refers to the horse, however, in the sentence "A house went into a barn and its roof was wet", "its" refers to the barn.

As mentioned above, the reasoner is also limited to the logical rules it can understand. Further work could be conducted to support more logical rules e.g. temporal and modal logic. In this work, we explored the possibilities of handling defeasible logic, and we presented two limited methods. The first method, using multiple proofs is limited since it cannot handle all the cases (e.g., see Appendix VI-E). Furthermore the reasoning by cases method is limited since for antecedent of the **Conditional Defeasible Expressions** it only supports atomic propositions. Moreover, our reasoning by cases method lacks the implementation of mutually exclusive cases tableau rules presented in Figure 18. Future work could be done to address these limitation of the reasoning by cases method. Furthermore in our implementation the information needed for defeating a defeasible proposition is not the minimal set of propositions needed to defeat the defeasible proposition (e.g., see Figure 33).

Moreover, future work could be conducted on handling of a large database of information. However, this is not a simple task due to branching factor of the tableaux tree; special measures such as using heuristics for order of applying logical rules on sentences is required for handling a large database of information.

C. Social impact

The social impact of our project is hard to determine at the current state of development. In general, reasoning tools for natural language have the potential to drastically change some areas of work. For example, in the judicial branch

$$\frac{(\mathcal{S}, \varphi \vee \psi)}{(\mathcal{S}, \varphi \wedge \neg\psi) \mid (\mathcal{S}, \varphi \wedge \psi) \mid (\mathcal{S}, \neg\varphi \wedge \psi)}$$

$$\frac{(\mathcal{S}, \neg(\varphi \wedge \psi))}{(\mathcal{S}, \neg\varphi \wedge \psi) \mid (\mathcal{S}, \neg\varphi \wedge \neg\psi) \mid (\mathcal{S}, \varphi \wedge \neg\psi)}$$

$$\frac{(\mathcal{S}, \varphi \rightarrow \psi)}{(\mathcal{S}, \neg\varphi \wedge \neg\psi) \mid (\mathcal{S}, \neg\varphi \wedge \psi) \mid (\mathcal{S}, \varphi \wedge \psi)}$$

Fig. 18: The argumentation tableaux rules necessary for handling mutually exclusive cases [6]. These expressions can be decomposed in three mutually exclusive cases

reasoning systems can be used to automate or at least support processes such as law research. In case reasoning tools once get powerful enough to do those kinds of work, this obviously has an impact on people working in this sector. As in many other cases, that potential of automation can have different consequences. While new technologies could be used in order to increase productivity, they could also be used to pare down personnel which of course might have some negative impact on society.

When and whether this case might occur also depends on the progress that can be made on reasoning techniques like the one this project is about. To be used in sensitive areas such as law, the reasoning tools have to improve in some aspects. For example, the AI's decisions need to be explainable and trustworthy. While explainability has been a focus of this research, the standard needed might be even higher.

In addition to progress which needs to be made on the reasoning tools themselves, there might also be work to be done on the laws or texts that the reasoning tools have to work with. Our reasoning tool relies on a controlled natural language, that means that in case that similar tools which also rely on a CNL should be used on laws, these laws first have to be checked or rewritten so that they match the CNL's rules. The Dutch government took first steps in this field by rewriting the tax laws in a controlled natural language called Regelspraak [12] so that they can be understood by both humans and computer programs. This project can be interesting to observe as many challenges from the perspective of social impact that this project faces might also need to be adressed when developing tools like this further.

VI. Conclusion

In this report we illustrated the improvements of the CNL and semantic tableau proposed by Inglot et al. [1]. We found that the CNL is limited and it needs the sentences to be structured in a straightforward and computer-friendly way to avoid ambiguity. Moreover, we found that using regular expression limiting to model more complex language structures. Furthermore, by using argumentation tableau method, we showed that the support could be further used for handling inconsistencies and defeasible logic. We found implementation of logical rules a hard task; and our implementation flawed. There are cases in which our program does not produce the correct results. Moreover, our program is deterministic, meaning with a given input, it will produce the same set of outputs. Finally, the user interface has been adapted to changes in the CNL and tableau.

List of Figures

References

[1] K. Inglot et al. *Explanaible AI: reasong with natural language*. 2021.

[2] Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*. 6. Harvard University Press, 1983.

[3] E.W. Beth. *Formal Methods: An Introduction to Symbolic Logic and to the Study of Effective Operations in Arithmetic and Logic*. Synthese Library. Springer Netherlands, 2012. ISBN: 9789401032698.

[4] Norbert E. Fuchs and Rolf Schwitter. "Attempto Controlled English (ACE)". In: *CoRR* cmp-lg/9603003 (1996). URL: http://arxiv.org/abs/cmp-lg/9603003.

[5] Fabio Massacci. "Single step tableaux for modal logics: Computational properties, complexity and methodology". In: *JOURNAL OF AUTOMATED REASONING* 24 (2000), p. 2000.

[6] Nico Roos. "A Semantic Tableau Method for Argument Construction". In: *Artificial Intelligence and Machine Learning*. Ed. by Mitra Baratchi et al. Cham: Springer International Publishing, 2021, pp. 122–140. ISBN: 978-3-030-76640-5.

[7] Norbert E. Fuchs. "Reasoning in Attempto Controlled English: Non-monotonicity". In: *Proceedings of the 5th International Workshop on Controlled Natural Language (CNL 2016)*. Ed. by Brian Davis, Gordon J. Pace, and Adam Wyner. Vol. 9767. Lecture Notes in Computer Science. Springer, 2016, pp. 13–24. ISBN: 978-3-319-41497-3.

[8] Lasha Abzianidze. "A Tableau Prover for Natural Logic and Language". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 2492–2502. DOI: 10.18653/v1/D15-1296. URL: https://aclanthology.org/D15-1296.

[9] Reinhard Muskens. "An Analytic Tableau System for Natural Logic". In: *Logic, Language and Meaning*. Ed. by Maria Aloni et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 104–113. ISBN: 978-3-642-14287-1.

[10] Lasha Abzianidze. "LangPro: Natural Language Theorem Prover". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 115–120. DOI: 10.18653/v1/D17-2020. URL: https://aclanthology.org/D17-2020.

[11] Günther Patzig. "What is an Aristotelian Syllogism?" In: *Aristotle's Theory of the Syllogism: A Logico-Philological Study of Book A of the Prior Analytics*. Dordrecht: Springer Netherlands, 1968, pp. 1–15. ISBN: 978-94-017-0787-9. DOI: 10.1007/978-94-017-0787-9_1. URL: https://doi.org/10.1007/978-94-017-0787-9_1.

[12] Mariette Lokin et al. "RegelSpraak: een brug tussen wetgeving en ICT". Dutch. In: *RegelMaat* 35.1 (Jan. 2020), pp. 28–47. ISSN: 0920-055X. DOI: 10.5553/RM/0920055X2020035001003.

APPENDIX

Appendix A: Examples

A. Simple example

Given information:

1) Unless I study AI, I am chilling
2) I do not study AI
3) I am chilling

To be shown: I am chilling

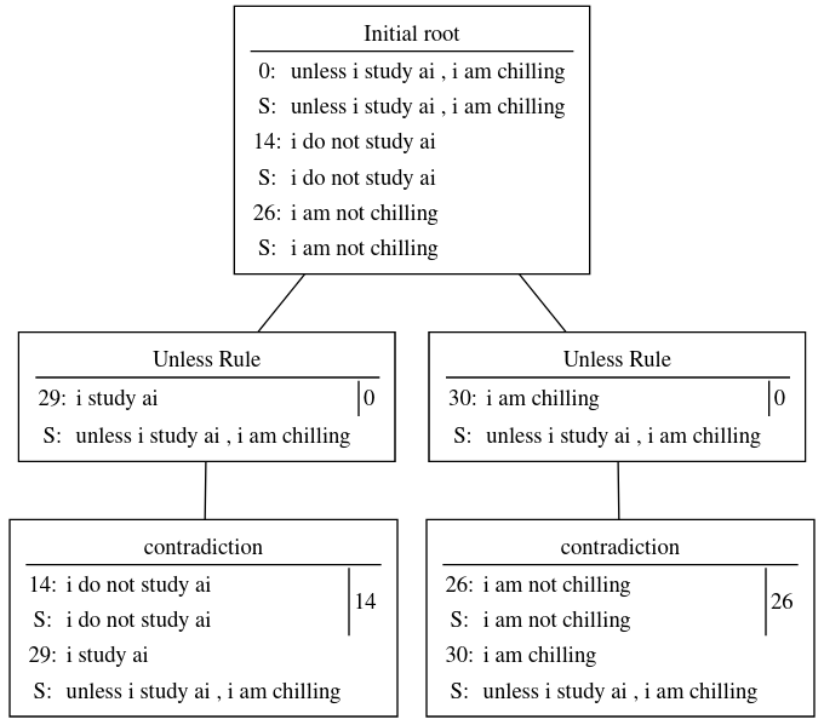

Fig. 19: Semantic tableau generated for this example. Each node contains a set of expressions (each of them identified by a unique number) with their support (marked with ”S”)

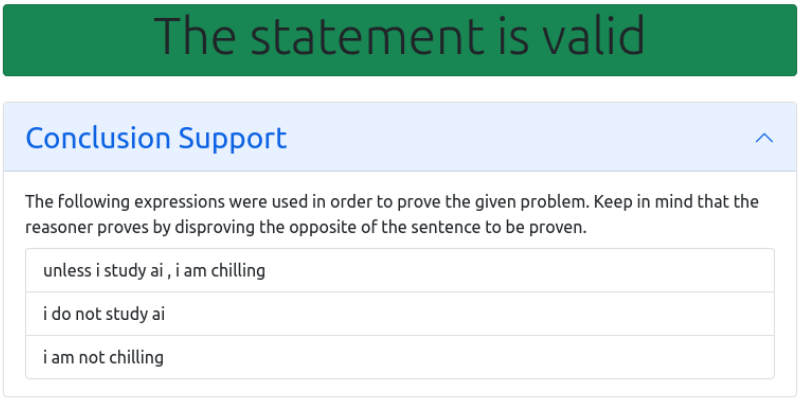

Fig. 20: There is a contradiction in each leaf node, therefore the tableau is closed. All input statements have been used for the proof, so they are inserted in the support for the conclusion.

B. Simple example, but the tableau does not close

Given information:

1) I play football or you play football

2) I play football

To be shown: you play football

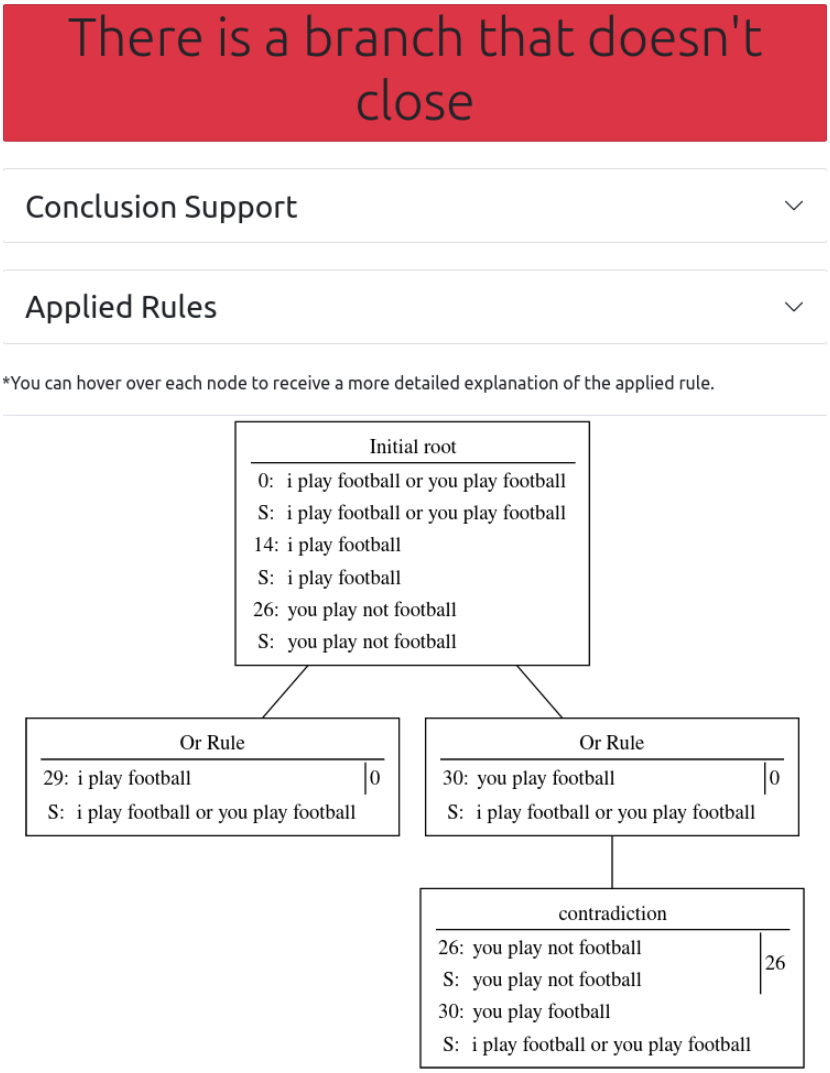

Fig. 21: In this case it is not possible to prove the sentence ”you play football”. We can see that there is a leaf nodes that does not contain a contradiction.

C. Defeasible rules, multiple tableaux approach

Given information:

1) Usually I play volleyball
2) Usually If I play volleyball then I am happy

To be shown: I am happy

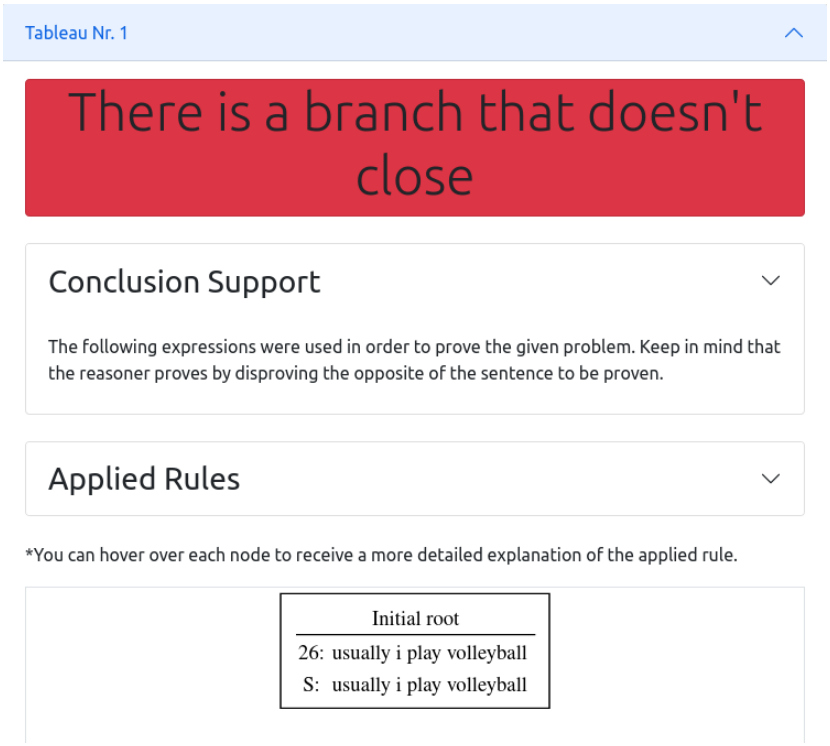

Fig. 22: The first tableau is used to add the defeasible expression ”Usually I play volleyball”. There are no contradiction between this sentence and the set of non-defeasible sentences (none in this case), therefore we can assume that this sentence is true.

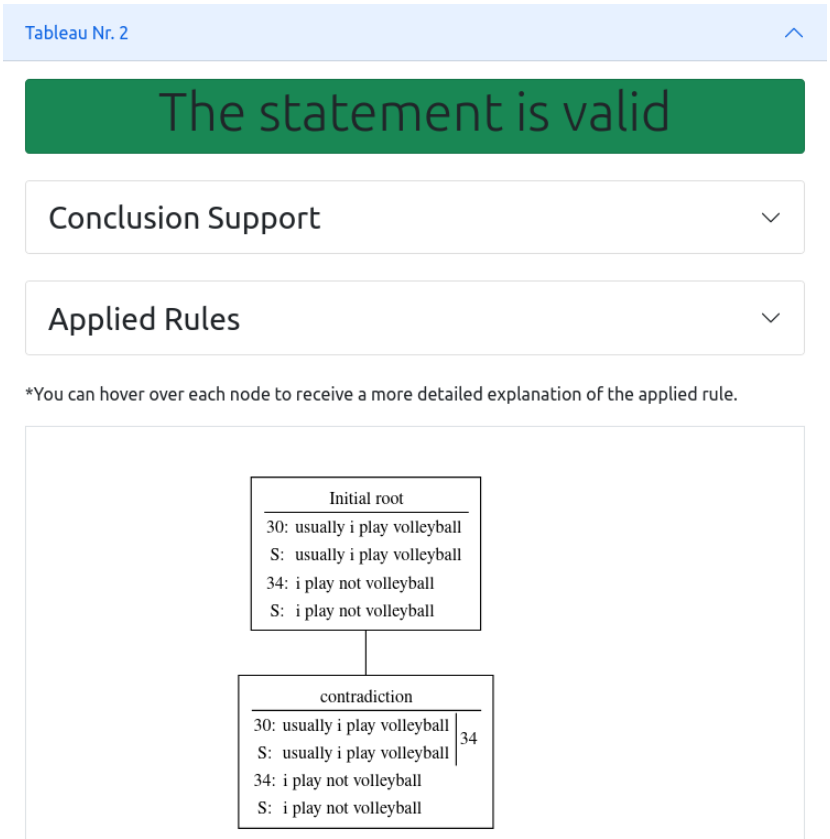

Fig. 23: The second tableau proves the premise of the second defeasible expression (”I play volleyball”). This can be proves, so the conclusion (”I am happy”) can be used in the next tableaux.

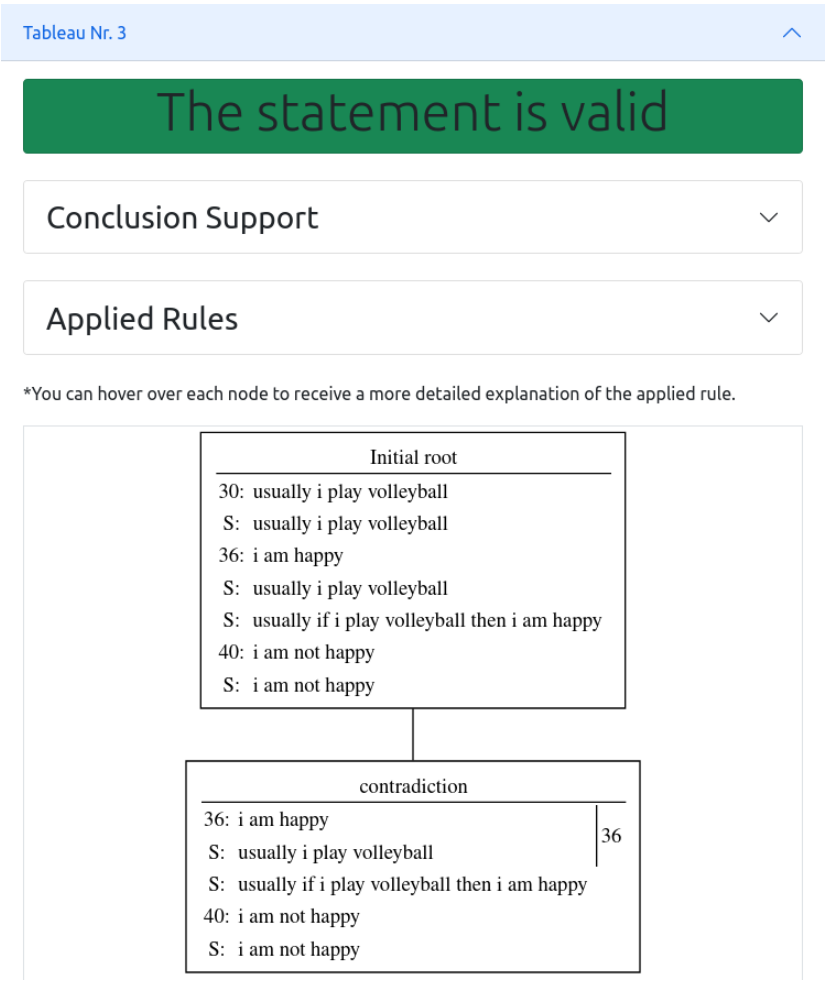

Fig. 24: In the third tableau, there are no defeasible statement left. So, this tableau tries to prove the conclusion of this example (”I am happy”), which can be immediately proved. Note the support of ”I am happy”: this was derived in the second tableau and it is made up ”Usually I play volleyball” and ”Usually if I play volleyball then I am happy”.

D. Defeasible rules, reasoning by cases approach

Given information:

1) I play volleyball or I read a book
2) Usually If I play volleyball then I am happy
3) Usually I am happy if I read a book

To be shown: I am happy

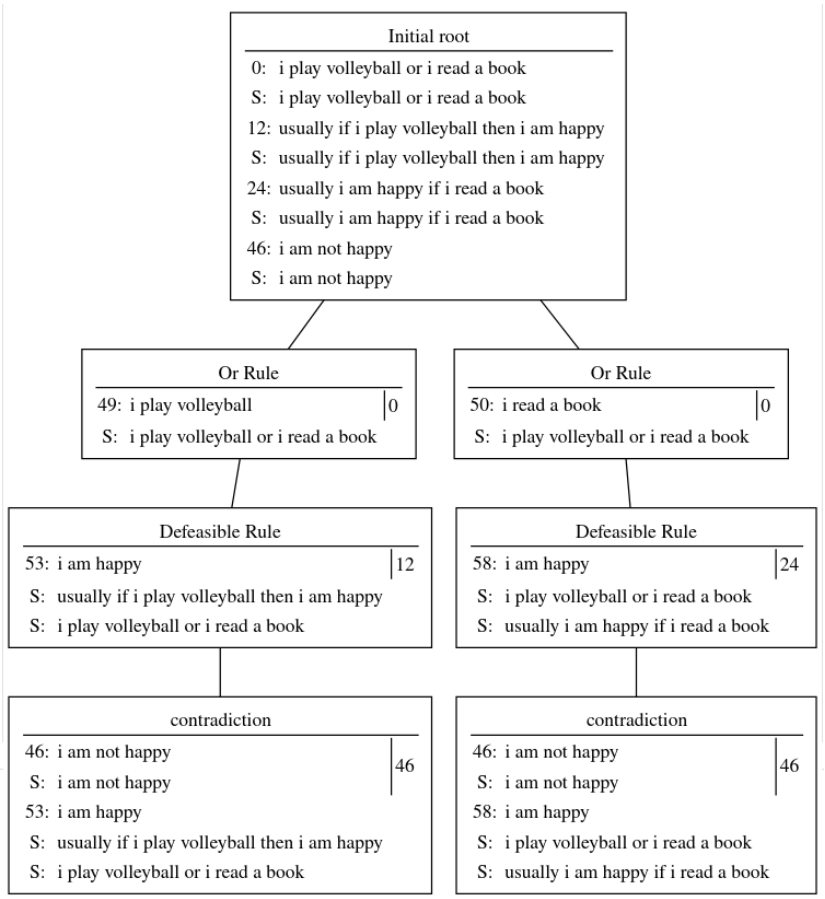

Fig. 25: When reasoning by cases only one tableau is generated. There are multiple branches in the tableau and some defeasible expressions can be applied only in some branches. However, all the branches generate a constradition, therefore they reach the same conclusion.

E. Defeasible rules, limitations of the multiple tableaux and reasoning by cases approaches

Given information:

1) I play volleyball or I read a book
2) Usually If I play volleyball then I am happy
3) Usually I am happy if I read a book

To be shown: I am happy

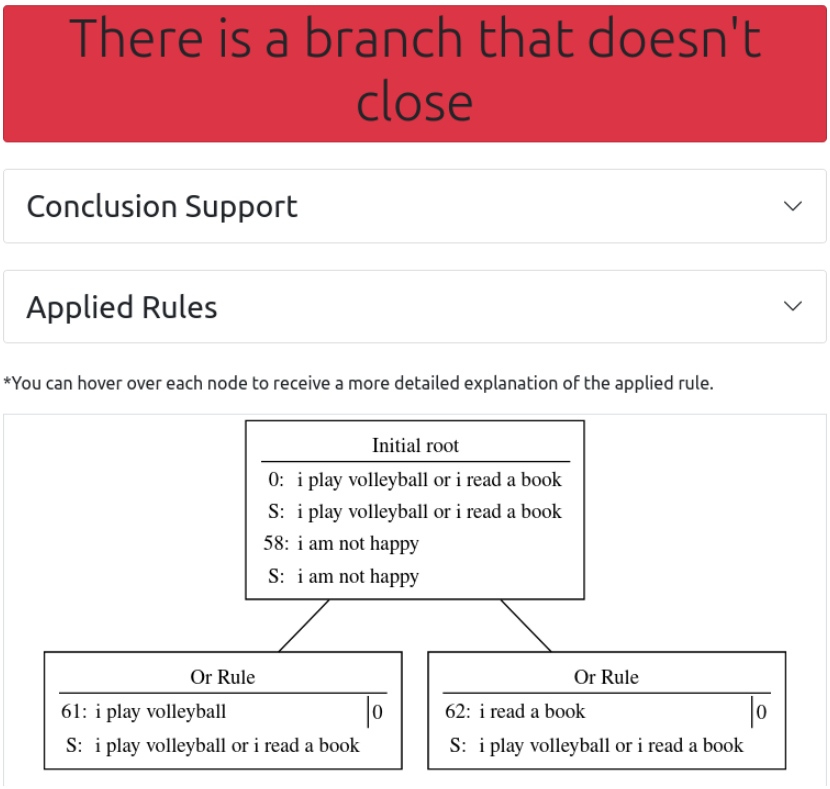

Fig. 26: The multiple tableau approach fails with this example, because no defeasible expression can be added to the table given the initial non-defeasible information. However, the reasoning by cases approach can prove this. This example shows that our system is not complete, since there are statements that can be proved but this method is not able to do it.

F. An example of support in case of irrelevant information

The following example contains seven premises of which only three are needed to prove the statement to be proven. This is shown in Figure 27. As you can see in Figure 28, the conclusion support contains the three sentences that contain the relevant information plus the negation of the sentence to be proven.

Fig. 27: Input of the conclusion support visualisation with seven premises of which only three are actually relevant to proof the statement

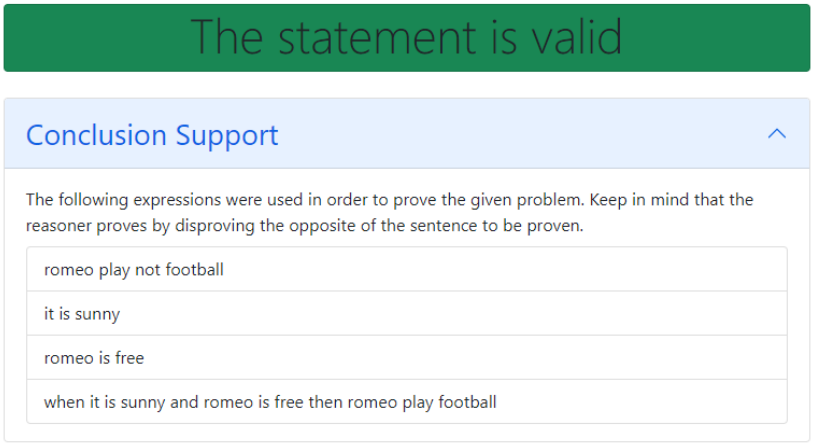

Fig. 28: Result and conclusion support card of the example of 27; the conclusion support contains only the relevant premises plus the negated hypothesis

G. Language checker

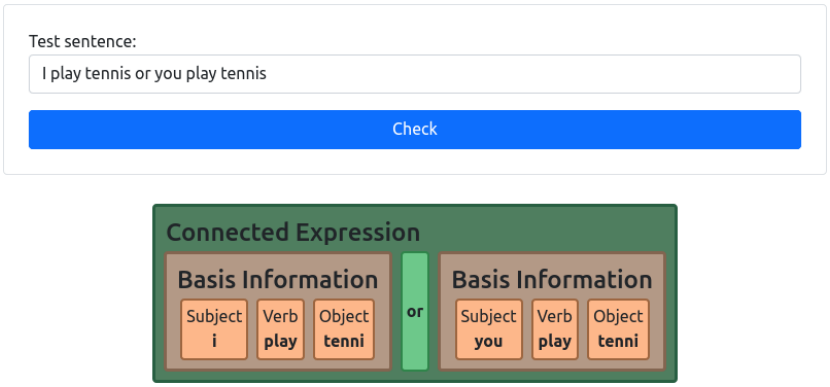

Fig. 29: Parsing a valid sentence. The web interface shows the structure of the parsed sentence.

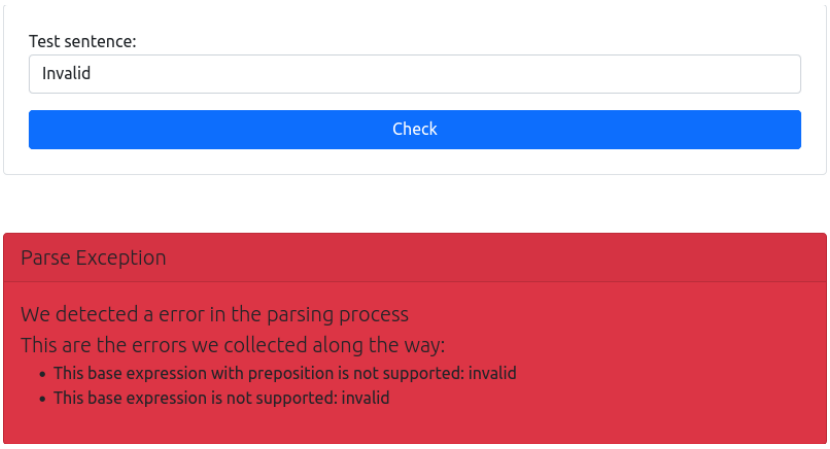

Fig. 30: Attempt of parsing an invalid sentence.

H. An example of inconsistency in the information

Given information:

1) When it is raining, John does not play football
2) It is raining
3) John plays football

4) John plays football or chess
5) John is a boy
6) John is happy

The following information, form a tautology:

1) When it is raining, John does not play football
2) It is raining
3) John plays football

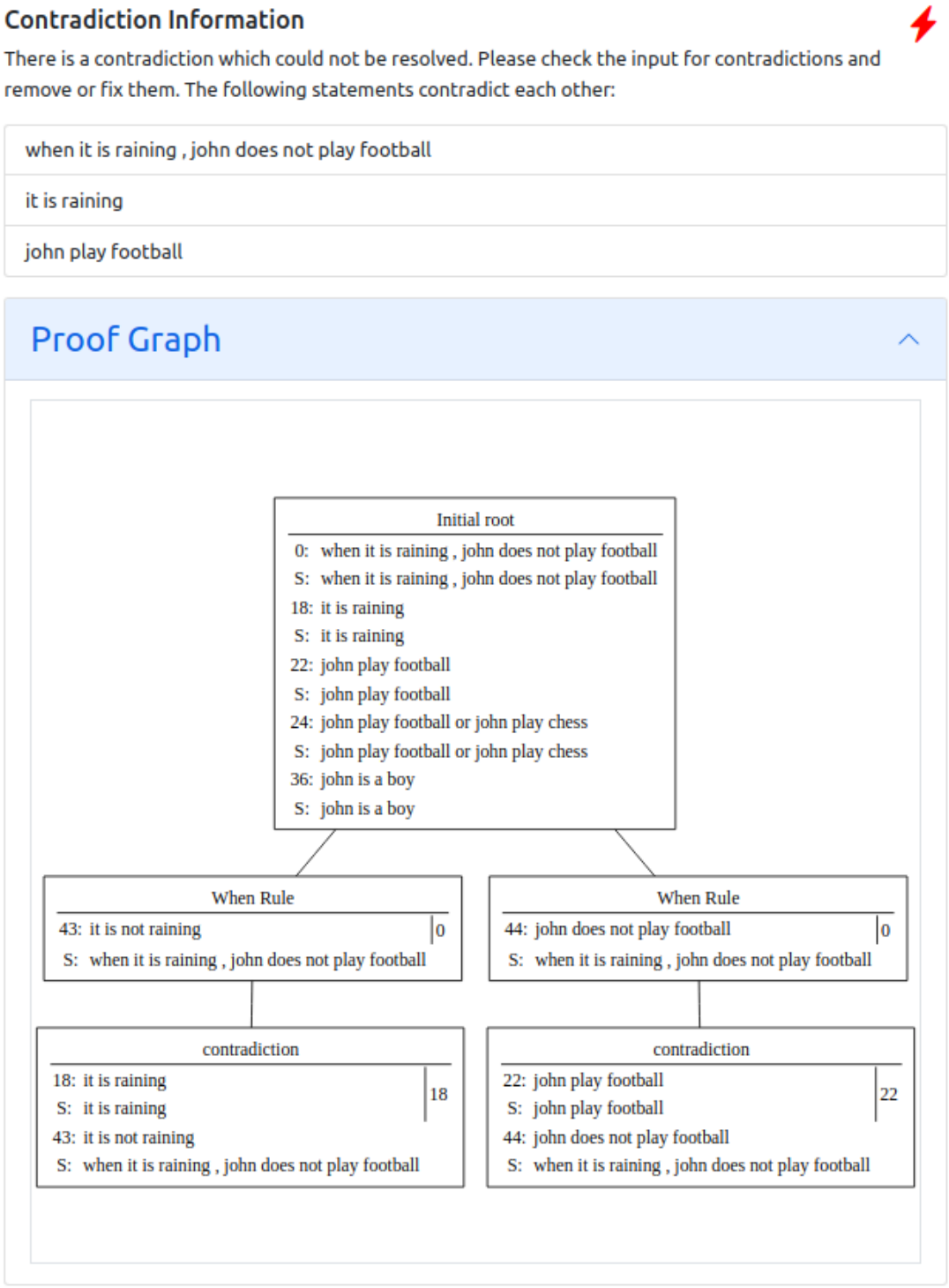

Fig. 31: A contradiction found in the information, presented with a set of claims and a tableaux

I. An example of defeated defeasible expressions

Given Information:

1) Usually it is not raining
2) When it is cloudy then it is raining
3) it is cloudy
4) Usually it is sunny
5) when it is cloudy then it is not sunny

Claim: it is raining and it is not sunny

In this example, both reasoning by cases and multiple tableau proof generate the same result of the claim being True with the same set of Support. When using the reasoning by cases method, we defeat the defeasible propositions ”usually it is sunny” and ”usually it is not raining” from the information and prove the claim from the accepted information. 34)

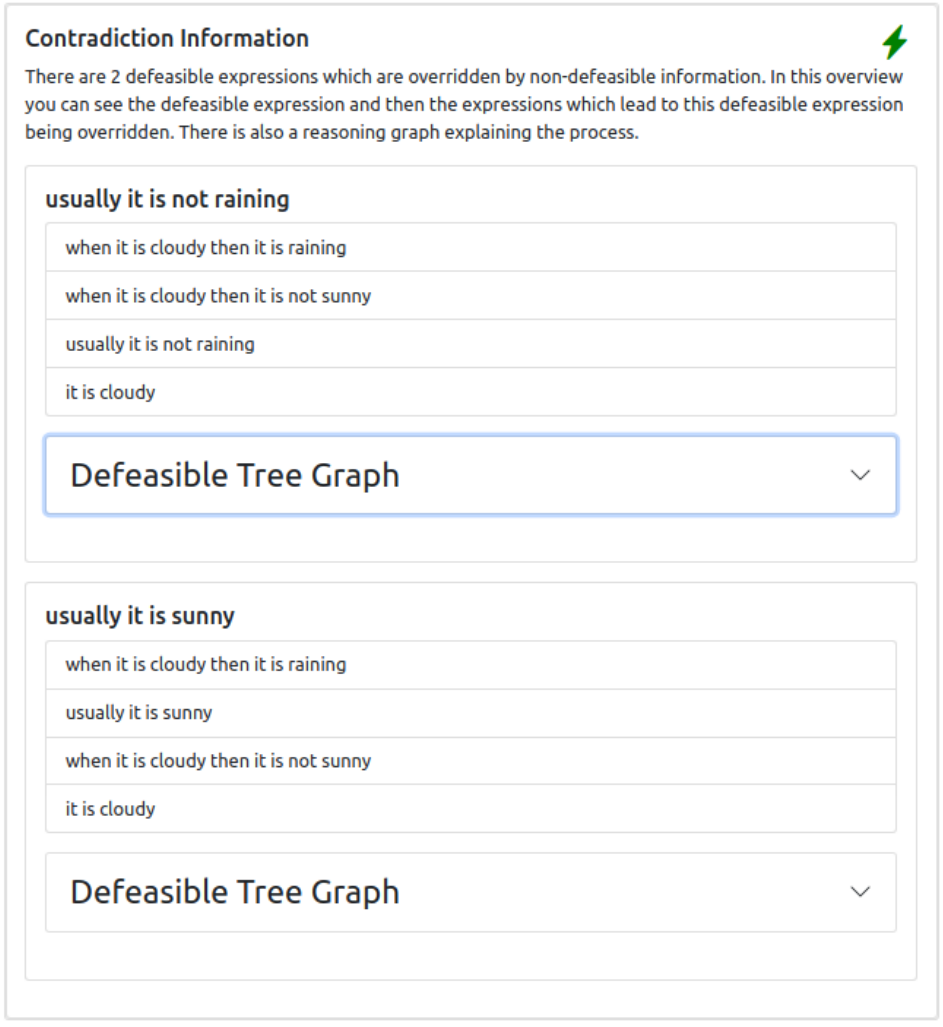

Fig. 32: Here we have two defeasible expressions defeated with the presented set of Information and its closed tableau (see Figure 33).

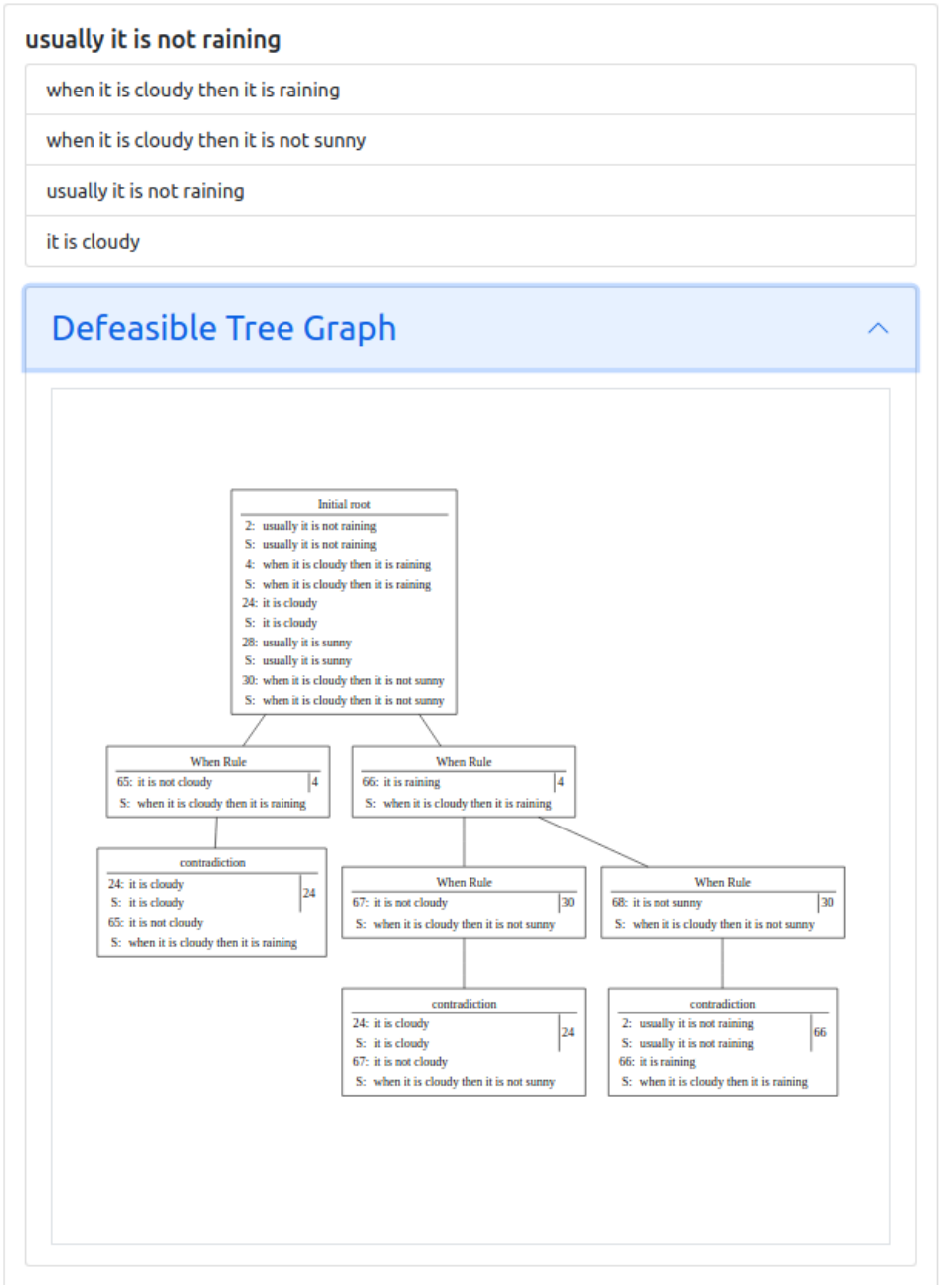

Fig. 33: The tableau tree used to disprove the defeasible information. Here the claim against the case that ”usually it is not raining” is not the minimal, since the second information i.e, ”when it is cloudy then it is not sunny” is not needed to create the contradictions.

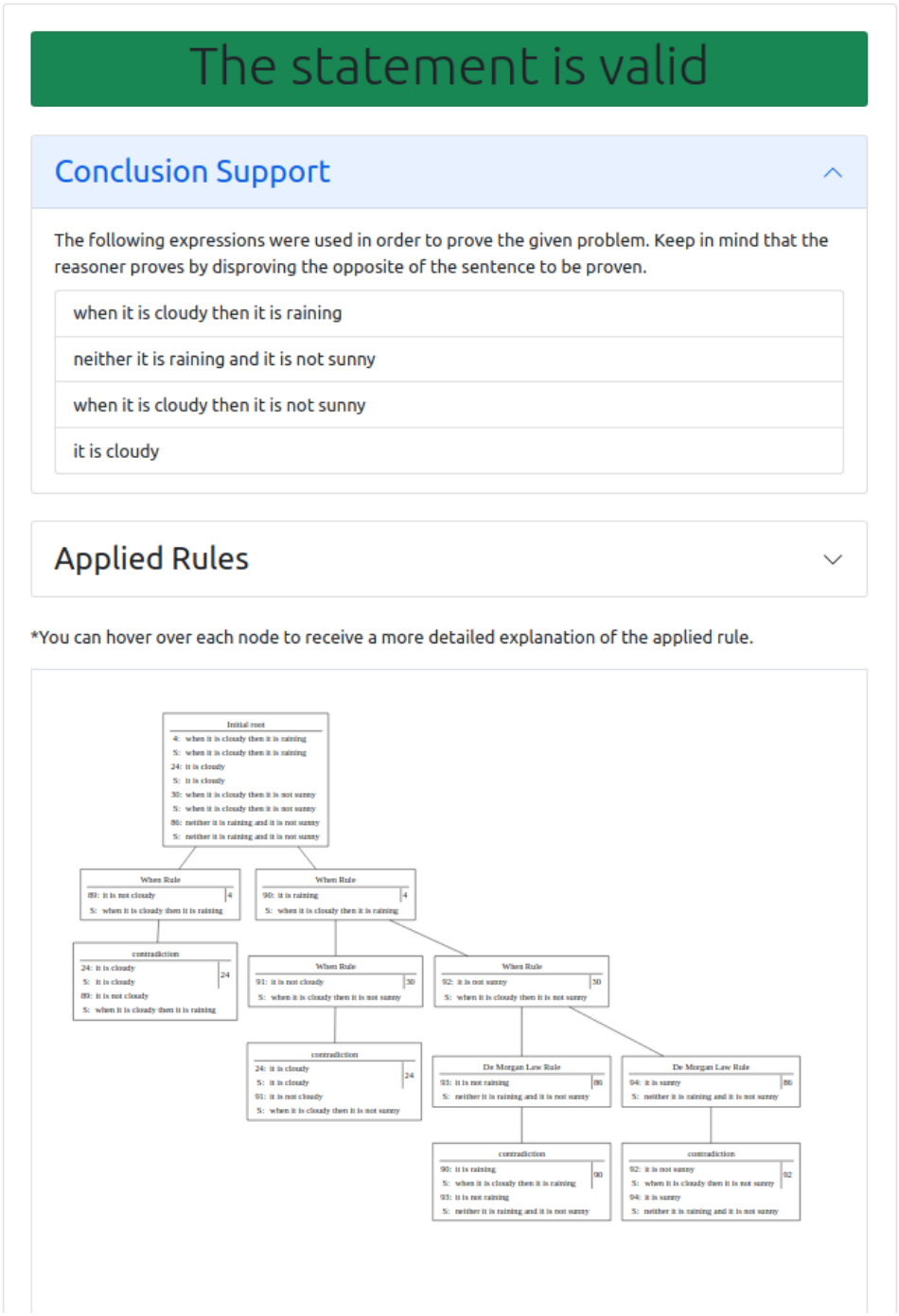

Fig. 34: The tableau used here to prove the conclusion after rejecting defeasible information.