university of groningen / faculty of science and engineering

Automatic Tableau Builder for Grzegorczyk Logic

Bachelor's Project Thesis

Nina Schoeber, s2689162, n.schoeber@student.rug.nl,
Supervisor: Prof. Dr. L.C. Verbrugge

Abstract: This research aims to build an automated tableau builder for Grzegorczyk logic and to check whether its logic is sound and complete. This is done by creating a tableau system using the rules and characteristics of Grzegorcyk logic. These new tableau rules are then directly implemented in an algorithm that tries to create a tableau for every inference. This algorithm is able to prove the Grzegorczyk axiom and its set of rules is proven to be sound and complete.

1 Introduction

In this research, the goal is to create an automated tableau builder to check whether formulas are consistent with Grzegorczyk logic and determine whether the logic used by this program is sound and complete. If a formula is inconsistent with a logic, its negation is proven for that logic. Therefore, this program can also be described as an automated theorem prover. The research question in this thesis is: Is an adaptation of Grzegorczyk logic for an automated theorem prover sound and complete?

Rautenberg (1983) describes Grzegorczyk logic as a type of modal logic. Its rules and axiom are described along with other types of logic. Dyckhoff and Negri (2013) discuss the characteristics that these rules govern. Grzegorczyk logic follows the axiom $\Box(\Box(p \to \Box p) \to p) \to p$. It is characterized by reflexive, transitive and weakly conversely well-founded frames, as is explained further in Section 1.3.

Grzegorczyk logic is related to the more well-known Gödel-Löb logic (GL) by the translation $\Box A$ (in GL) $\Rightarrow \Box A \wedge A$ (in Grz) (Boolos, 1995). GL is a logic characterized by irreflexive, transitive and conversely well-founded frames. This translation adds the reflexive property and thereby makes it weakly conversely well-founded. In Peano Arithmetic (PA), GL's $\Box$ is translated as the provability predicate $Prov$ (Verbrugge, 2017). This means that $\Box A$ (in GL) becomes $Prov(\ulcorner A \urcorner)$ (in PA), which means that A is provable. Applying this to the relation between Grzegorczyk logic and GL, gives the following relation between Peano Arithmetic and Grzegorczyk logic: $\Box A$ (in Grz) becomes $Prov(\ulcorner A \urcorner) \wedge A$ (in PA). This means that A is both provable and true.

In order to create a tableau builder for Grzegorczyk logic, a suitable tableau system had to be selected. Priest (2008) describes a tableau system for K, which is the smallest modal logic. This tableau system is deemed most suitable for use in automated tableau building. The rules described by Rautenberg (1983) are in a different tableau system, but are formed as an extension to K and therefore can be adapted to the structure of the Priest (2008) tableau system, extending its system for K. This results in a new set of rules in the Priest (2008) tableau system.

This set of rules then has to be implemented in an automated theorem prover. In order to answer the research question, the soundness and completeness of the new set of rules is attempted to be proven. The program also generates the tableau for the negation of the Grzegorczyk axiom to check whether the program is able to prove it.

The hypothesis is that the new set of rules is sound and complete and able to prove the Grzegorczyk axiom. It can be implemented directly into an automated theorem prover and therefore the research question would be answered in the affirmative.

1.1 Tableau Systems and Types of Tableau Rules

As mentioned by Goré (1999), a tableau system (or tableau calculus) is a finite collection of tableau rules. A tableau for a formula X is a finite tree where X is the root and whose nodes have finite formula sets. A tableau rule in both Goré (1999) and Rautenberg (1983) is of the form $\frac{N}{D_1|D_2|...|D_k}$, where N is the numerator and each D is a denominator. The numerator and all denominators are finite sets of formulas. The tableau rule is read as "if the numerator is L-satisfiable, then so is at least one of the denominators" for an axiomatically formulated logic L (Goré, 1999). L-satisfiable means here that there is an L-model for the set of formulas. An L-model for a set of formulas is a model of type $< W, R, V >$ (where $< W, R >$ is an L-frame) for which there exists some world w_0 in W such that for all formulas A in the set $w_0 \models A$. An L-frame is characterized by specific properties for the type of logic. The properties of a Grz-frame are explained in Section 1.3.

In Priest's (2008, Chapter 2.4) tableau system, the rules have a different structure. Every rule looks like a part of the final tableau and therefore has a tree structure in itself. Each node contains one formula and the index of the world in which this formula is true. If there are multiple child nodes connected to one parent node, at least one of these child nodes has to be true. An example is the following rule for disjunction:

$$\begin{array}{c} A \vee B \,.\, i \\ \diagup \quad \diagdown \\ A \,.\, i \qquad B \,.\, i \end{array}$$

This rule shows that if the formula $A \vee B$ is true in world i, then A or B has to be true in world i, which is shown by the branch splitting. The Priest tableau system has a set of rules for the smallest modal logic K. In this logic, a tableau branch is closed when both A, i and $\neg A, i$ appear on the same branch. A tableau is closed if all of its branches are closed. A tableau is complete when no more rules can be applied. If the tableau is complete but not closed, it is L-consistent (Goré, 1999). To prove a theorem (for example the formula P) in a logic, a tableau is made with its negation $(\neg P, 0)$ as starting point. If the tableau closes, that means that $\neg P$ is L-inconsistent, which means that P is proven in L.

1.2 Soundness and completeness from Priest in K

Soundness and completeness are two important characteristics of a logic. The soundness theorem is defined as follows: "For finite Σ, if $\Sigma \vdash A$ then $\Sigma \models A$" (Priest, 2008). This means that if a formula A is provable from the set of rules Σ, it must also be a logical consequence of Σ. Therefore, a logic is sound if and only if every theorem is a tautology.

The opposite is true for completeness. A logic is complete if and only if all of its tautologies are theorems. The completeness theorem is defined as follows: "For finite Σ, if $\Sigma \models A$ then $\Sigma \vdash A$" (Priest, 2008). This means that every formula A that is a logical consequence of Σ, must also be provable from Σ.

1.3 Grzegorczyk Logic Properties

As mentioned by Dyckhoff and Negri (2013), Grzegorczyk logic is characterized by reflexive (as opposed to the more well-known Gödel-Löb logic (GL) which is characterized by irreflexive frames), transitive and weakly conversely well-founded (Noetherian) frames. They define a weakly conversely well-founded relation as follows: "Let R be a relation on a set S.

1. An R-sequence is a sequence of elements of S such that for any two successive elements x_i and x_{i+1}, $x_i R x_{i+1}$ holds
2. An R-sequence is convergent (or stabilises) iff, for some i, for all $j > i$ for which x_j is defined, $x_i = x_j$
3. R is weakly conversely well-founded iff every R-sequence is convergent."

This shows that weakly conversely well-founded relations satisfy the ascending chain condition. The ascending chain condition means that every strictly ascending sequence of elements eventually terminates. Because step 2 shows that every R-sequence must stabilize or converge, there can be no infinite

ascending sequence. Therefore, these relations satisfy the ascending chain condition.
That Grzegorczyk logic is characterized by reflexive, transitive and weakly conversely well-founded frames is proven by showing that the axiom Grz is always true in these frames and that there is always a possible countermodel in frames where any of these properties is missing. This proof can be found in Appendix A.

1.4 Tableau Calculus Rules for Grz

Rautenberg (1983) defines a set of rules for several types of modal logics. Their derivability relation $\vdash$ must satisfy the normality rule and a set of formal properties for the classical propositional consequence. These properties are defined by the following rules:

- ($\Box$) (normality rule) $\dfrac{X \vdash P}{\Box X \vdash \Box P}$
- (Dt) (Deduction theorem) $X; P \vdash Q \Leftrightarrow X \vdash P \rightarrow Q$

Classical rules:

- (o) $\dfrac{P; \neg P}{0}$
- ($\neg$) $\dfrac{X; \neg\neg P}{X; P}$
- ($\wedge$) $\dfrac{X; P \wedge Q}{X; P; Q}$
- ($\vee$) $\dfrac{X; \neg(P \wedge Q)}{X; \neg P | X; \neg Q}$
- (θ) $\dfrac{X; Y}{X}$

Specific modal rules:
For K:

- (K) $\dfrac{\Box X; \neg\Box P}{X; \neg P}$

For Gr (Grzegorczyk) = K($\Box(\Box(p \rightarrow \Box p) \rightarrow p) \rightarrow p$) (which means that Gr is the normal extension of K by the extra axiom $\Box(\Box(p \rightarrow \Box p) \rightarrow p) \rightarrow p$):

- (M) $\dfrac{X; \Box P}{X; P}$
- (Go) $\dfrac{\Box X; \neg\Box P}{X; \Box X; \neg P; \Box(P \rightarrow \Box P)}$

All of the rules above hold for Grzegorczyk logic. In the tableau system by Priest, the rules for K are already present. Because extending K to Gr adds the rules M and Go, these have to be added to the existing rules in the Priest tableau system.

1.5 Tableau Rules for K

K is the weak modal logic that is used as a foundation for a lot of modal logics (Garson, 2016). It uses the propositional set of logical and non-logical symbols and extends these with $\Box$ and $\diamond$ which mean "it is necessary that" and "it is possible that", respectively. It is itself created by adding the necessitation rule and the distribution axiom to the principles of propositional logic. The necessitation rule states that for every formula A, if A is a theorem of K, then so is $\Box A$. This states that every theorem in the logic is provably necessary. The distribution axiom is defined as follows: $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$ for all formulas A and B.

Priest (2008) defines a tableau system for K using a set of worlds with relations between them. Each tableau node consists of a formula and a number i that corresponds to the world w_i in which it is true. The relations are shown by nodes of the type irj, which shows the relation between worlds w_i and w_j. This corresponds with the notation $w_i R w_j$ in a Kripke model. If a formula $\Box A$ is true in a world w, A must be true in every world x for which wRx. If a formula $\diamond A$ is true in a world w, A must be true in at least one world x for which wRx.
A list of all rules of the tableau system for K defined by Priest (2008) are in Appendix B.

1.6 Adapting the Priest Tableau System to Grzegorczyk Logic

In order to adapt the rules from Grzegorczyk logic to the Priest tableau system, the differences between the tableau systems have to be identified and interpreted.

1.6.1 Tableau Calculus rules

The main difference between the two types of tableau rules is that Rautenberg's (1983) rules do not relate the calculus to specific worlds, while

Priest's (2008) rules do. It seems that as a way of translation, a Rautenberg rule of the form $\frac{\Box X; A}{X; B}$ can be read as "If A is true in world w_i, B is true in at least one world w_j where $w_i R w_j$".
For example, looking at rule (K) $\frac{\Box X; \neg\Box P}{X; \neg P}$, it is clear that if $\Box X, i$ and $\neg\Box P, i$ are true, that the combination of X, i and $\neg P, i$ has to be true in at least one world related to w_i. This is the case because $\Box X, i$ mandates that X is true in every world related to w_i and $\neg\Box P, i$ states that P is not necessarily true in every world related to w_i, which means that there is at least one accessible world in which P is false (or $\neg P$ is true).
In Priest's tableau system, this is arranged by the following four rules:

1.
$$\begin{array}{c} \Box A \,.\, i \\ irj \\ | \\ A \,.\, j \end{array}$$
for all existing j

2.
$$\begin{array}{c} \neg\Box A \,.\, i \\ | \\ \Diamond\neg A \,.\, i \end{array}$$

3.
$$\begin{array}{c} \Diamond A \,.\, i \\ | \\ irj \\ A \,.\, j \end{array}$$
for a new j

4.
$$\begin{array}{c} \neg \Diamond A \,.\, i \\ | \\ \Box\neg A \,.\, i \end{array}$$

From rule 1 follows that if $\Box X, i$, then X is true in every world related to w_i and from a combination of rules 2 and 3 follows that if $\neg\Box P, i$ is true, then $\Diamond\neg P, i$ is true and therefore P is false (or $\neg P$ is true) in at least one world related to w_i.
Rules of the form $\frac{X; A}{X; B}$ can be read as "If A is true in world w_i, B is true in world w_i".
To adapt the Priest tableau system to create tableaux in Grzegorczyk Logic, a similar analysis has to be done on rules (M) and (Go) as mentioned in Section 1.4 and new tableau rules in the Priest system have to be created for these rules.

1.6.2 Other Properties

As mentioned in Section 1.3, Grzegorczyk logic is characterized by reflexive, transitive and Noetherian frames. This means that a few additional rules have to be added to the Priest tableau system. In order to make the relations reflexive, for every occurence of i in the tree, iri has to appear in the tree. In order to make the relations transitive, whenever the relations irj and jrz occur on a branch, the relation irz is introduced as well.

2 Method

In order to build an automated tableau builder for Grzegorczyk logic, the rules from Rautenberg (1983) first have to be adapted to work with the Priest (2008) tableau system and then these rules (combined with Priest's rules for K) have to be implemented.

2.1 Rules

As mentioned in section 1.6, the rules (M) and (Go) have to adapted to work with the Priest tableau system and rules for reflexivity and transitivity have to be added.

2.1.1 Extensions to K

First, we check if there are existing extensions to K that already use some of these new properties. The extension $K_{\rho\tau}$ includes the rules:

$$\begin{array}{c} \cdot \\ | \\ iri \end{array}$$
for all i on the branch

for the reflexive property ρ and

$$\begin{array}{c} irj \\ jrk \\ | \\ irk \end{array}$$

for the transitive property τ. These rules introduce the reflexive and transitive properties and therefore we use this extension to add the other rules to.

2.1.2 (M)

In Rautenberg's (1983) article, the rule (M) is defined as follows: $\frac{X; \Box P}{X; P}$
This rule is interpreted as "If $\Box P$ is true in world

w_i, then P is also true in world w_i" as explained in Section 1.6. In this interpretation, the rule defines the reflexive property of Grzegorczyk logic. In $K_{\rho\tau}$ this ρ property is already included by the rule mentioned in the previous section. This rule adds the relation between a world and itself, which by definition of $\Box$ has the same effect as rule (M). Therefore no extra rule has to be added to the Priest system for (M).

2.1.3 (Go)

In Rautenberg's (1983) article, the rule (Go) is defined as follows: $\dfrac{\Box X; \neg\Box P}{X; \Box X; \neg P; \Box(P \to \Box P)}$
This rule is interpreted as "If $\Box X; \neg\Box P$ is true in world w_i, then $X; \Box X; \neg P; \Box(P \to \Box P)$ is true in at least one world w_j where $w_i R w_j$". This introduces two properties:

1. A necessity symbol in a world i does not only mandate the (set of) formula(s) being true in the worlds related to i, but also in the worlds related to those worlds and so on. This introduces the transitivity property of Grzegorczyk logic. In $K_{\rho\tau}$ this τ property is already included by the rule mentioned in Section 2.1.1. Therefore no extra rule has to be added for this property.

2. When a formula P is not necessarily true ($\neg\Box P$) it has to be false in at least one accessible world. In the world w_i where P is false, $\Box(P \to \Box P)$ is true. This means that for every world w_j for which $w_i R w_j$, if P is true in that world, it is also true in every world w_k for which $w_j R w_k$. Because of transitivity, it is in turn also true in every world related to k and so on. This property can be added to the Priest system using the following rule:

$$\begin{array}{c} \neg\Box A \,.\, i \\ | \\ irj \\ \neg A \,.\, j \\ \Box(A \to \Box A) \,.\, j \\ \text{for a new } j \end{array}$$

   This rule defines the weakly conversely well-foundedness of the Grz-frames, because it delimits the sequence. It makes sure that there can never be an infinite sequence. The reflexive property combined with $A \to \Box A$ lets the last world only relate to itself, thereby ending the sequence.

2.2 Program Design

The program uses an object-oriented approach. The tableau is an object of the type Tree and can contain up to two child nodes which are each also of the type Tree. Apart from its child nodes, each Tree object contains the following information:

- Formula f
- World w
- Constants c
  A set of constants that are shared by all nodes to keep track of the present worlds
- *applied*
  A boolean that represents whether f has already been applied
- *closed*
  A boolean that represents whether this node is closed

Each formula is an object of the class of its main connective or of the class Atom, WorldRelation or Absurd. These classes are all subclass of the class Formula. Every subclass contains its own functions Apply and Negate and contain a number of formulas based on the arity of the connective. This dictates which effect the application of that formula has on the tree. For the program's functionality, world relations are encoded as a formula of the class WorldRelation. The classes and their arity are as follows:

- Absurd (0)
- WorldRelation (0)
- Atom (1)
- Negation (1)
- Necessity (1)
- Possibility (1)
- Disjunction (2)

- Conjunction (2)
- Conditional (2)
- Biconditional (2)

This setup allows the main program to simply call the Apply function on the object of the type formula, without needing extra knowledge of the specific type of formula. The Apply function of the class Negation calls the Negate function of its subformula.
The Apply and Negate functions directly implement the rules as defined in Section 2.1 and Appendix B. For example, the Apply function of the class Conjunction adds both subformulas to the current branch if they do not yet occur on the branch, whereas the Negate function splits the branch and adds each subformula to their separate subbranches. A formula is only added to the branch if it does not yet occur on it. Whether or not the formula was added is returned in the form of a boolean. This is done by a function in the class Tree, which can be seen in algorithm 2.1.
This function is called on the tree object whose formula is being applied. This algorithm searches the tree from the top using breadth-first search. Whenever it encounters a closed subtree or a subtree with the same formula and world as the node that has to be added, that subtree is ignored. If the current node is found, the queue is cleared and $foundNode$ is set to true. This causes the node to be added to every leaf below the current node, except for subbranches that are closed or are below a node with the same formula and world.
The method to split a branch is much simpler. The algorithm does not have to check if the nodes already occur on the branch and can therefore simply add them to every leaf below the current node that is not part of a closed subbranch.

The functions Apply and Negate then generally become very simple. Apart from the Apply function in the class Necessity and the Negate function in the class Possibility, they all follow a similar structure. The branch is extended exactly as in the rules, creating small subtrees that have the exact same structure as the rules. The only main difference is whether a branch is only extended, or also split. The functions return a boolean which represents whether the application of the formula made any changes to the tree. When the branch is extended,

Algorithm 2.1 The method that adds a node to a tree if it is not yet present in that branch of the entire tree

Input: The node that has to be added *newNode*, the *entireTree* and the tree object on which this function is called *this*
Output: A boolean whether the *newNode* was added to *this*

```
added ← FALSE
foundNode ← FALSE
toCheck ← new list
toCheck ← entireTree
while toCheck is not empty do
  tChecking ← first item from toCheck
  if tChecking is a closed node or tChecking
  has the same formula as newNode then
    skip this node and all nodes below it
  end if
  if tChecking == this then
    toCheck ← empty
    foundNode ← TRUE
  end if
  if tChecking has a left child node then
    add left child node to toCheck
    if tChecking has a right child node then
      add right child node to toCheck
    end if
  else
    if foundNode then
      added ← TRUE
      tChecking left child ← newNode
    end if
  end if
end while
return added
```

this boolean is determined by a disjunction of the results of the *addNode* calls. When the branch is split, there is always a change in the branch, therefore simply true is returned.

The Apply function for the Necessity and the Negate function for the Possibility are more complicated, because they have to check for relations between worlds. We first focus on the Apply function for the Necessity. The formula can occur both before and after the world relations in the tree and can be applied to multiple relations. Therefore, there are two separate searches in the algorithm. The first search starts at the node of the Necessity at world index i and searches the nodes below it using breadth-first search. For every node that contains a world relation that defines a relation between world w_i and another world, *addNode* is called and a node containing the subformula of the Necessity at the new world is added below the world relation. The second search does exactly the opposite. It searches the entire tree up to the node of the Necessity for relations with w_i and for every world relation it finds in the same branch, *addNode* is called and a node containing the subformula of the Necessity at the new world is added below the node of the Necessity.

Because a Possibility or $\Diamond$ is defined as $\Diamond A \equiv \neg\Box\neg A$ (Garson, 2016), both its Apply and Negate function are a special case of those of the class Necessity. The Apply function of the class Possibility creates a new object of the class Necessity with as formula a negation of its own formula. Then the Negate function of the Necessity is called on this. The Negate function of the class Possibility also creates a new object of the class Necessity with as formula a negation of its own formula. Then the aforementioned Apply function of the Necessity class is called on this.

Every iteration of the application, the following steps are taken:

1. A formula is selected and applied.
2. All old necessities and negations of possibilities are checked to see if they can be applied again.
3. The tableau is checked to see if there are any recurring sequences that would create an infinite tree.
4. The tableau is checked for contradictions and (partly) closed when they are found.
5. The transitivity is updated.

This continues until the tableau is closed, stopped because of an infinite sequence or the tableau is complete. After this the tableau is printed to a PDF using LaTeX together with information about whether it is closed and if not why it was stopped. If the tableau is open and complete, a countermodel is created by applying depth-first search on the tableau to find any open branch.

2.2.1 Formula selection

The order in which the separate formulas are applied is not bound to specific rules. The order that was used is based on the 'human strategy'. When creating a tableau on paper, you first apply the formulas that do not split the branch before you apply the splitting formulas and you apply the possibilities (and negations of necessities) before the necessities (and negations of possibilities) However, there is no optimal solution for this. The ideal order is inference-dependent. For example, in some situations it is more efficient to apply necessities before splitting formulas. When testing, the following order seemed to be most efficient:

1. Non-splitting formulas in the same world
   - Conjunction
   - Negation of a disjunction
   - Negation of a conditional
2. Possibilities and negations of necessities
3. Splitting formulas in the same world
   - Disjunction
   - Conditional
   - (Negation of a) Biconditional
   - Negation of a conjunction
4. Necessities and negations of possibilities

The program selects which formula to apply by traversing the tree using breadth-first search and determining the score of each formula based on the order above if it has not yet been applied. The node

with the current highest score is recorded. Whenever a new node with the same or a higher score is found, it overwrites the previous best node. After the entire tree has been checked, the node with the highest score is applied. Whenever a necessity or negation of a possibility is found, the node is also added to a list. This is done because these types of formulas must be applied again when new worlds are added. Every iteration, this entire list is checked and applied when possible. When a node closes, it is removed from the list.

2.2.2 Transitivity

Every iteration, the transitivity is updated. This is done by double breadth-first search. The algorithm traverses the entire tree. Whenever it finds a world relation irj, it calls a different function that searches every node below this world relation to find every formula of the form jrx. Using the *addNode* function described above, it then adds a new world relation of the form irx for each of them.

2.2.3 Closing the tableau

Every iteration, the tableau is checked to see if it can be (partially) closed. This is done by breadth-first search over the tree. For every combination of formula and world index in the tree, the subtree below it is searched for a contradicting formula with the same world index. This occurs when one of the two formulas is a negation of the other. When a contradiction is found, the formula of the two that is lowest in the tree is set to closed and an absurd is added to all leaves below it.
After all contradictions have been found, the tree is traversed again to close every node whose child nodes are both closed. When the root of the tableau is closed, the entire tableau is closed and therefore the program is stopped.

2.2.4 Checking for infinite tableaux

In some cases, tableau branches can become infinite. There are two ways to make sure that the algorithm recognizes this and stops in a timely fashion: setting a maximum tree depth or a method that checks for patterns. Because a maximum depth is arbitrary and can produce problems with large tableaux, the pattern detection was implemented. In order for a tableau to become infinite, it has to contain related worlds with the same set of formulas, thus creating a repeating pattern. To detect this pattern, the algorithm checks for each world if it contains only all the same formulas as a previous world it is related to. If this is the case, a future world will have the same pattern, thus creating an infinite tableau. In this case, the program is stopped and the tableau remains open. The infinite pattern is translated into a corresponding countermodel. Because a repetition of the entire set of formulas will always cause an infinite tableau, this detection method does not get any false positives. Therefore this method should detect all infinite sequences without stopping tableaux that would not otherwise get infinite.

2.2.5 Generation of countermodels

When a tableau is open and complete or an infinite branch is found, a countermodel has to be given. In an open and complete tableau, this countermodel is found by using depth-first search through the tree. The first open branch that is found is chosen. Then the worlds, relations and atoms are recorded to create a model of the form $< W, R, V >$. When a tableau is stopped because of an infinite branch, the other branches of the tableau are not necessarily complete. Therefore, the infinite pattern is selected as the countermodel and its worlds, relations and atoms are recorded.

3 Results

3.1 Soundness and Completeness

Priest (2008) has proven the soundness and completeness of K. Therefore, in this proof only the added rules/characteristics are discussed. These are the reflexivity, transitivity and the weak converse well-foundedness (or Noetherian property) of R. These are mandated by the following rules:

1. Reflexivity
ρ
$\vdots$
iri
for all i on the branch

2. Transitivity
τ
irj
jrk
$\vdots$
irk

3. Noetherian
$\neg\Box A\,.\,i$
$\vdots$
irj
$\neg A\,.\,j$
$\Box(A \to \Box A)\,.\,j$
for a new j

The rule for the Noetherian property replaces the rule in K for $\neg\Box A$.

3.1.1 Soundness

Lemma
Priest (2008, Chapter 2.9) uses the following definition of faithfulness in his proof of soundness for K: "Let $I =< W, R, v >$ be any modal interpretation, and b any branch of a tableau. Then I is faithful to b iff there is a map f from the natural numbers to W such that

- For every node A, i on b, A is true at $f(i)$ in I
- If irj is on b, $f(i)Rf(j)$ in I"

The soundness lemma is defined as follows: "Let b be any branch of a tableau, and $I =< W, R, v >$ be any interpretation. If I is faithful to b, and a tableau rule is applied to it, then it produces at least one extension, b', such that I is faithful to b'." This is proven by a case by case consideration of the rules. Because Priest has proven the soundness of K, only the new rules mentioned in Section 1 are discussed. Because the weak converse well-foundedness is mandated by the rules for $\neg\Box A$ and $\diamond A$ in the exact same way, only one of them is discussed. For all of these situations it is supposed that I is faithful to the branch b.

1. **Reflexivity** ρ
   The rule ρ causes iri to be on the direct extension b' of b for any world i. Since R is reflexive, $f(i)Rf(i)$. Therefore I is faithful to the extension of b' of b.

2. **Transitivity** τ
   Suppose irj and jrk are on b and the rule τ is applied. This gives an extended branch b' which contains irk. Since I is faithful to b, $f(i)Rf(j)$ and $f(j)Rf(k)$. Hence, $f(i)Rf(k)$ because R is transitive. Therefore I is faithful to the extension b' of b.

3. **Weak converse well-foundedness**
   Suppose $\neg\Box A, i$ is on b and the new rule for $\neg\Box A$ is applied. Then the branch is extended to b' which contains irj with a new j and $\neg A, j$ and $\Box(A \to \Box A), j$. Because I is faithful to b, $\neg\Box A$ is true at $f(i)$. Hence, for some $w \in W$, $f(i)Rw$ and $\neg A$ is true at w. Let f' be the same as f except for $f(j) = w$. Hence $f'(i)Rf'(j)$ and $\neg A$ is true at $f'(j)$.
   Because R is Noetherian, there cannot be any infinitely ascending sequences of worlds. This means that when $\neg\Box A$ is true at $f(i)$, there must also be some formula that delimits the sequence. The transitive nature of R could potentially make it infinite otherwise. For example, when $\neg\Box A, i$ was added to the branch due to an application of the formula $\Box\neg\Box A, h$ for which hri appears on the branch, $\neg\Box A, w$ would also appear on the branch for every world w created by application of the new rule for $\neg\Box A$, because of the transitive nature of R. This would cause an infinite sequence. To avoid this, for every world $w \in W$ such that $f(j)Rw$, if A is true at w, it must be true in all related words ($\Box(A \to \Box A)$ must be true at world w). This is delimiting, because the branch closes whenever $\neg\Box A, w$ and $\Box A, w$ are on the same branch for the same world w, thus removing the possibility for $\neg\Box A, w$ appearing on the branch for every new related world and thus avoiding infinite sequences in R. It does not cause any infinite sequences in itself, as it does not cause the creation of any new worlds and does not cause any conflicts in situations that would not otherwise introduce an infinite sequence. Because of the reflexive nature of R, the formula $\Box(A \to \Box A), j$ allows a world w_j to at some point only access itself, thereby stopping the sequence.
   Therefore, $\Box(A \to \Box A)$ must be true at $f(j)$ which means that I is faithful to the extension b' of b.

Theorem
The soundness theorem is defined as follows: For finite Σ, if $\Sigma \vdash_{Grz} A$ then $\Sigma \models_{Grz} A$

This is proven by contraposition. Suppose $\Sigma \nvDash_{Grz} A$. This means that there is a valuation v such that for every $B \in \Sigma$, B is true and A is false at some world w. Let f be any function such that $f(0) = w$. This shows that I is faithful to the initial list. For $\Sigma \vdash A$ to be true, the tableau starting with the premises (Σ) and the negation of the conclusion (A) must be closed. This means that both X, w and $\neg X, w$ must appear on a branch b. Because I is faithful to the initial list b, by the soundness lemma it is faithful at least one extension b' of b. Because there is only one extension b', I should be faithful to this extension of b. This means that both $v_w(X) = 1$ and $v_w(X) = 0$, which is impossible. Therefore $\Sigma \nvdash_{Grz} A$ can be concluded.
Therefore, if $\Sigma \vdash_{Grz} A$ then $\Sigma \models_{Grz} A$. Hence, this set of rules is sound.

3.1.2 Completeness

Lemma
"Let b be any open complete branch of a tableau, let $I =< W, R, v >$ be the interpretation induced by b. Then:

- If A, i is on b then A is true at w_i
- If $\neg A, i$ is on b then A is false at w_i"

An interpretation is induced by a branch b if:

- $\mathrm{W} = \{w_i : i \text{ occurs on } b\}$
- If p, i occurs on b, then $v_{w_i}(P) = 1$
  If $\neg p, i$ occurs on b, then $v_{w_i}(P) = 0$
  If neither occurs on b, then you can choose arbitrarily between $v_{w_i}(P) = 1$ and $v_{w_i}(P) = 0$.

This is proven by induction on the complexity of A, where the induction hypothesis is as follows:

- If B, i occurs on b, then $v_{w_i}(B) = 1$
- If $\neg B, i$ occurs on b, then $v_{w_i}(B) = 0$
- If neither occurs on b, then you can choose arbitrarily between $v_{w_i}(B) = 1$ and $v_{w_i}(B) = 0$.

Because Priest has proven the completeness of K, only the new rules mentioned in Section 1 are discussed. Again, because the weak converse well-foundedness is mandated by the rules for $\neg\Box A$ and $\diamond A$ in the exact same way, only one of them is discussed.

1. **Reflexivity** ρ
   By the ρ rule, for every $w_i \in W$, iri occurs on b. Hence, by definition of R, $w_i R w_i$ as required. This shows that the model is reflexive, because for every world w, wRw.

2. **Transitivity** τ
   For $w_i, w_j, w_k \in W$, suppose $w_i R w_j$ and $w_j R w_k$. Then irj and jrk occur on b and by the τ rule irk also occurs on b. Hence, $w_i R w_k$ as required. This shows that the model is transitive, because for all relations $w_i R w_j$ and $w_j R w_k$, $w_i R w_k$.

3. **Weak converse well-foundedness**
   If $\neg\Box B, i$ is on b, then the new rule for $\neg\Box A$ has been applied. Thus for some new j, irj, $\neg B, j$ and $\Box(B \to \Box B), j$ are on b. By the induction hypothesis, $w_i R w_j$ by definition and B is false at w_j by the induction hypothesis. If $\Box(B \to \Box B), j$ is on b, then by the rule for necessity for all k such that jrk is on b, $B \to \Box B, k$ is on b. By the rule for the conditional, either $\neg B, k$ or $\Box B, k$ is on b. If $\Box B, k$ is on b, then for all l such that krl is on b, B, l is on b. By the induction hypothesis, this means that B is true at w_l hence $\Box B$ is true at w_k or B is false at w_k, because by definition $w_k R w_l$. Hence $\Box(B \to \Box B)$ is true at w_k. Hence $\neg\Box B$ is true at w_i. As explained in the proof for the soundness, this formula is delimiting. This means that there cannot be an infinite ascending sequence in the set of relations R. Therefore, the model is weakly conversely well-founded.

Theorem The completeness theorem is defined as follows: For finite Σ, if $\Sigma \models_{Grz} A$ then $\Sigma \vdash_{Grz} A$. This is also proven by contraposition. The following proof is from Priest (2008): "Suppose that $\Sigma \nvdash_{Grz} A$. Given an open complete branch of the tablau, the interpretation that this induces has the right characteristics and makes all premises true at w_0 and A false at w_0, hence $\Sigma \nvDash_{Grz} A$"
Therefore, if $\Sigma \models_{Grz} A$ then $\Sigma \vdash_{Grz} A$

3.2 Grzegorczyk Axiom Proof

In Figure 3.1, the tableau for the negation of the Grzegorczyk axiom is shown. Because this tableau

closes, the Grzegorczyk axiom is proven in this program and its corresponding set of rules.

3.3 Program functionality

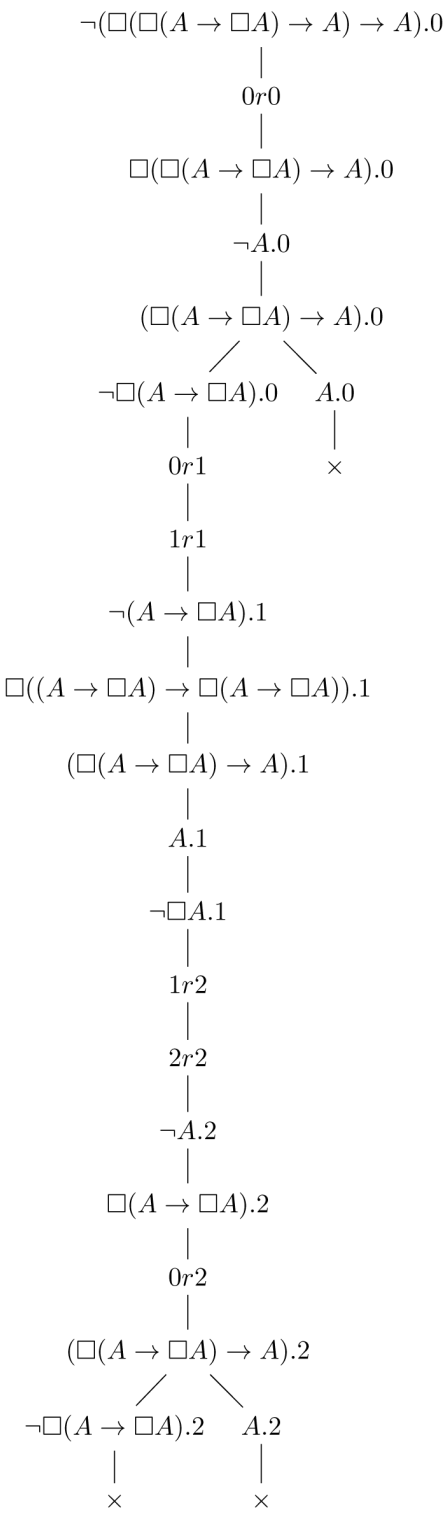
Generally, the program functioned well. Whenever a closed tableau was generated, this was the correct solution. The program did not close the tableaux too soon or too late. Whenever a infinite pattern was detected or the tableau was open and complete, the generated tableau and countermodel were also correct. This countermodel was not necessarily the shortest possible countermodel. As the chosen detection method for infinite tableaux was rather strict, the program did not stop these tableaux too soon. However, because the order of application was not ideal for every formula, the program did not always produce the most efficient tableau.

4 Discussion

As shown in the previous section, the new set of rules is sound and complete. Because the program uses the rules directly (as shown in Section 2.2), it can be concluded that the logic used by the program is sound and complete.
The research question was: Is an adaptation of Grzegorczyk logic for an automated theorem prover sound and complete? This functioning automated theorem prover directly uses the new set of rules. This new set of rules is sound and complete and has all of the properties that characterize Grzegorczyk logic. The program also efficiently proves the Grzegorczyk axioma Grz. Therefore the research question can be answered in the affirmative. This adaptation of Grzegorczyk logic for an automated theorem prover is sound and complete.

4.1 Potential problems

While the program uses the rules directly, the application order is not mandated by any rules. The current order was chosen for efficiency and likeness to the human strategy. The ideal order is inference-dependent. For example, while it is often most efficient to apply splitting formulas before necessities, there are also situations in which it is more efficient to apply necessities first. For example, for the formula $(\Box\neg(A \vee B) \wedge (A \vee B))$,



This tableau is closed.

Figure 3.1: The generated proof of the Grzegorczyk axiom $\Box(\Box(p \to \Box p) \to p) \to p$

the tableau closes sooner when necessities are applied first, as can be seen in Figure C.1 and Figure C.2 in Appendix C. While the order of application only has a small influence on the tableau for this inference, it shows that a different order or strategy might produce vastly different tableaus. In some cases, these changes might make the difference between a tableau that continues forever (and is therefore cut off) and a tableau that closes early in the process. While this was not encountered during testing, situations like this are theoretically possible. As there is no universal optimal strategy, this might cause a difference in performance between several types of formulas.

4.2 Further research

For future research, the influence of different application orders on different types of formulas could be examined. This could provide more insight in which application order is optimal for which type of formula. This insight could then be used to create an adaptive application order which changes its order based on the type of formula it encounters. This could solve the problems mentioned above and could create a more universal tableau builder that is less inference-dependent.

References

George Boolos. *The Logic of Provability*. Cambridge University Press, 1995.

Roy Dyckhoff and Sara Negri. A cut-free sequent system for Grzegorczyk logic, with an application to the Gödel−McKinsey−Tarski embedding. *Journal of Logic and Computation*, page ext036, 2013.

James Garson. Modal logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2016 edition, 2016.

Rajeev Goré. Tableau methods for modal and temporal logics. In *Handbook of Tableau Methods*, pages 297–396. Springer, 1999.

Larisa Maksimova. On modal Grzegorczyk logic. *Fundamenta Informaticae*, 81(1-3):203–210, 2007.

Graham Priest. *An Introduction to Non-Classical Logic: From If to Is*. Cambridge University Press, 2008.

Wolfgang Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12 (4):403–423, 1983.

Rineke (L.C.) Verbrugge. Provability logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition, 2017.

A Appendix A - Characterisations Grz

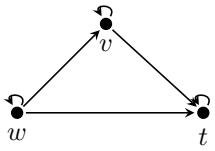
For all frames $F =< W, R >$:
$F \models \Box(\Box(p \to \Box p) \to p) \to p \iff R$ is reflexive, transitive and Noetherian

A.1 Reflexive, transitive and Noetherian

Suppose R is reflexive, transitive and Noetherian and $M =< W, R, v >$ is an arbitrary model on F where w is an arbitrary world such that $w \in W$.

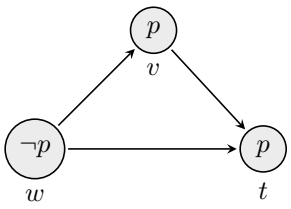
1. Suppose $M, w \models \Box(\Box(p \to \Box p) \to p)$ and $M, w \not\models p$.
2. Because of the reflexivity of R: $M, w \models \Box(p \to \Box p) \to p$
3. Therefore either:
   (a) $M, w \models p$ (leads to $\bot$ because of 1)
   or
   (b) $M, w \not\models \Box(p \to \Box p)$
4. There exists a world v with wRv for which both
   (a) $M, v \not\models p \to \Box p$ (because of 3b)
   (b) $M, v \models \Box(p \to \Box p) \to p$ (because of 1)
5. Because of 4a:
   (a) $M, v \models p$
   (b) $M, v \not\models \Box p$
6. Because of 4b:
   (a) $M, v \models p$ (already shown in 5a)
   or
   (b) $M, v \not\models \Box(p \to \Box p)$
7. There exists a world t such that vRt for which
   (a) $M, t \not\models p$ (because of 5b)
   (b) $M, t \models \Box(p \to \Box p) \to p$ (because of 1 and transitivity)
8. Because of 7b:
   (a) $M, t \not\models \Box(p \to \Box p)$
   or
   (b) $M, t \models p$ (leads to $\bot$ because of 7a)
9. Therefore there exists a world u such that tru for which
   (a) $M, u \not\models p \to \Box p$ (because of 8a)
   (b) $M, u \models \Box(p \to \Box p) \to p$ (because of 1 and transitivity)

This world u has the exact same set of starting formulas as world v. Therefore, this sequence would become infinite. Because the R is Noetherian, there can be no infinitely ascending sequences. The formula present in 8a states that $M, u \not\models p \to \Box p$ for any world u such that tru. In order to avoid an infinite sequence, at some point $u = t$ has to be true. This is possible due to the reflexivity of R. This leads to $M, t \models p$ and $M, t \not\models \Box p$. $M, t \models p$ leads to $\bot$ because of 7a and therefore it is not possible that $M, w \models \Box(\Box(p \to \Box p) \to p)$ and $M, w \not\models p$ in a reflexive, transitive and Noetherian frame, thus $F \models \Box(\Box(p \to \Box p) \to p) \to p$



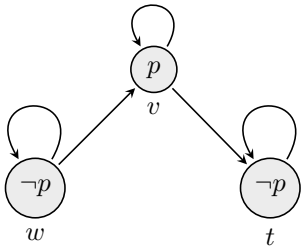
A.2 Not reflexive

Suppose R is not reflexive. Then there is a $w \in W$ such that not wRw. Now we can define a v on $< W, R >$ such that $M, w \models \Box(\Box(p \to \Box p) \to p)$ and $M, w \not\models p$. This can be done by making p true in every world related to w and false in every other world (such as w itself).
The valuation is therefore as follows: $\forall x \in W(V_x(p) = 1 \iff wRx$. Therefore, $F \not\models \Box(\Box(p \to \Box p) \to p) \to p$.



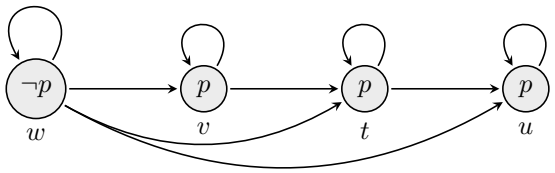
A.3 Not transitive

Suppose R is reflexive but not transitive. Then there is a combination of $w_i, w_j, w_k \in W$ such that w_iRw_j and w_jRw_k but not w_iRw_k. Now we can define a v on $< W, R >$ such that $M, w \models \Box(\Box(p \rightarrow \Box p) \rightarrow p)$ and $M, w \not\models p$. This is the case when p is only true in every world that is reachable in one step, except itself. This means that in every world related to w, p and $\neg\Box p$ are true.
This means that the valuation is as follows: $\forall x \in W(V_x(p) = 1 \iff wRx \wedge w \neq x$. Therefore, $F \not\models \Box(\Box(p \rightarrow \Box p) \rightarrow p) \rightarrow p$.



A.4 Not Noetherian

In the situation where all properties are present, it is shown that without the Noetherian property, an infinite sequence of worlds is produced. Every world except for starting world w then contains a combination of $M, v \not\models p \rightarrow \Box p$ and $M, v \models \Box(p \rightarrow \Box p) \rightarrow p$, because of the transitive and reflexive properties. This causes p to be true in every world except the starting world.
Therefore the valuation is as follows: $\forall x \in W(V_x(p) = 1 \iff x \neq w \wedge wRx)$. Therefore, $F \not\models \Box(\Box(p \rightarrow \Box p) \rightarrow p) \rightarrow p$.



A.5 Conclusion

Therefore, for all frames $F =< W, R >$:
$F \models \Box(\Box(p \rightarrow \Box p) \rightarrow p) \rightarrow p \iff R$ is reflexive, transitive and Noetherian

B Appendix B - Tableau Rules in K from Priest

- Negation

$$\neg\neg A\,.\,i \quad\Rightarrow\quad A\,.\,i$$

- Disjunction

$$\neg(A \vee B)\,.\,i \quad\Rightarrow\quad \neg A\,.\,i \;\;\neg B\,.\,i$$

$$A \vee B\,.\,i \quad\Rightarrow\quad A\,.\,i \;\mid\; B\,.\,i$$

- Conjunction

$$\neg(A \wedge B)\,.\,i \quad\Rightarrow\quad \neg A\,.\,i \;\mid\; \neg B\,.\,i$$

$$A \wedge B\,.\,i \quad\Rightarrow\quad A\,.\,i \;\;B\,.\,i$$

- Conditional

$$\neg(A \rightarrow B)\,.\,i \quad\Rightarrow\quad A\,.\,i \;\;\neg B\,.\,i$$

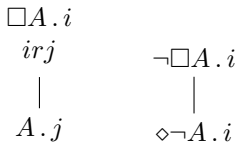
$$A \rightarrow B\,.\,i \quad\Rightarrow\quad \neg A\,.\,i \;\mid\; B\,.\,i$$

- Biconditional

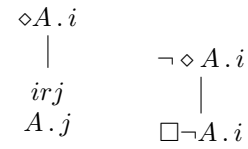
$$\neg(A \leftrightarrow B)\,.\,i \quad\Rightarrow\quad A\,.\,i,\ \neg B\,.\,i \;\mid\; \neg A\,.\,i,\ B\,.\,i$$

$$A \leftrightarrow B\,.\,i \quad\Rightarrow\quad A\,.\,i,\ B\,.\,i \;\mid\; \neg A\,.\,i,\ \neg B\,.\,i$$

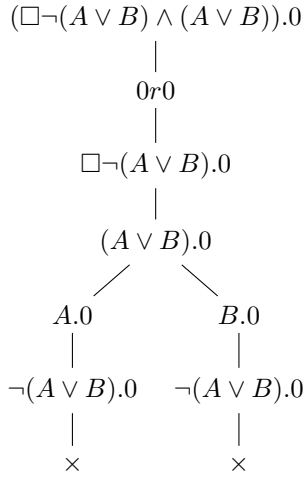
- Necessity



- Possibility

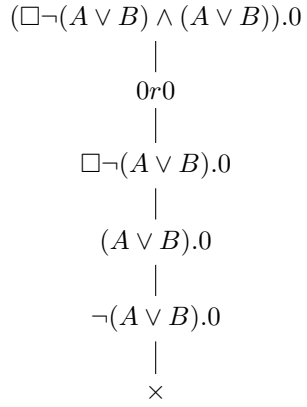


C Appendix C - Differences in application order



This tableau is closed.

Figure C.1: The tableau for $(\Box\neg(A \vee B) \wedge (A \vee B))$ when applying splitting formulas before necessities



This tableau is closed.

Figure C.2: The tableau for $(\Box\neg(A \vee B) \wedge (A \vee B))$ when applying necessities before splitting formulas