UNIVERSITÄT LEIPZIG
Fakultät für Mathematik und Informatik
Institut für Informatik

**A 3-Valued Approach to Disbelief**

# Diplomarbeit (englisch)

Leipzig, 19.3.03

vorgelegt von
Alexander Nittka
geb. am 13. September 1977
Studiengang Informatik

# Contents

# 1 Introduction and theoretical background

Before we can motivate the topic of our thesis, we need to sketch the general setting. Artificial intelligence, AI for short, is one branch of applied computer science. Simply put, it is concerned with modelling intelligent behaviour on machines – computers and robots in particular. Besides problem solving, planning, learning, reasoning etc. the representation of knowledge is one aspect of AI.

How should information, rules etc. be encoded so that (correct) conclusions can be drawn automatically? In order to model intelligent behaviour, this question must be answered – if not in general, then at least within a certain scope. One question follows directly. How can this be achieved in a way that modifications can be made?

## 1.1 Belief Revision

The core aspect of belief revision is the investigation of how a rational agent should modify its beliefs or knowledge – generally called the epistemic state – in the light of new information. There are several approaches to that question and they are based on different assumptions about how the knowledge of the agent is represented, what it is about, etc.

As our work is merely motivated by a belief revision framework, it will suffice to present some of the basic ideas of belief revision without examining them in detail.

The two main approaches to belief revision are foundationalism and coherentism. Whereas the former distinguishes a particular set of beliefs as the basic beliefs from which all others can be derived, the latter does not. So in the foundationalist approach a belief is justified if it can in some way be derived from the basic beliefs which are self-justifying. In the coherentist one it is the relationship to (all) other beliefs which determines whether a belief is justified.

This has a direct influence on whether revision is done on a belief base or a belief set. A belief set is understood to be a set of propositions closed under consequence, i.e. if $K$ is a belief set and $\varphi$ follows from $K$ then $\varphi \in K$. A belief base, on the other hand, is not necessarily closed under consequence. It just is a set of propositions.

The epistemic state of an agent could be represented in terms of a belief set or a belief base. If the latter is the case, the knowledge of the agent would still be seen as everything that can be derived from the belief base. It should be clear that different belief bases can

lead to the same set of believed propositions.

Common assumptions are that the epistemic state of an agent is consistent, i.e. that it does not allow a contradiction to be derived, and that after incorporating new information the epistemic state remains consistent. It should be clear that new information could contradict current beliefs. This means that some old or the new information must be modified or discarded in order to ensure a consistent state. The attempt to keep the loss of information as small as possible is known as information economy and it is a further main aspect of many of the revision approaches.

If the new information happens to be consistent with the current beliefs it suffices to insert it. This simple operation is known as expansion. If the new information $\varphi$ contradicts the current knowledge, however, a more complicated mechanism has to be used. One of the standard approaches is to remove the contradicting part from the epistemic state, i.e. contract $\neg\varphi$, and then expand with $\varphi$. The *Levi identity* formalizes this by stating that revision is the composed operation of first contracting by the negation and then expanding.

Here contraction is an intermediate step for revision and often contraction is seen as having this purpose only. In the next section we will argue against this notion.

As the modelling of expansion seems obvious, the various belief revision approaches deal with the question of how to model contraction in a rational way. The question is how to decide which beliefs should be eliminated. If, for example, the belief set contains $\{\varphi, \varphi \rightarrow \psi\}$, and $\psi$ is to be contracted, one of the two propositions has to be removed, otherwise $\psi$ could still be inferred. A choice between $\varphi$ and $\varphi \rightarrow \psi$ must be made. So one of the central points of all belief revision frameworks is how to make choices of that kind.

Probably the most fundamental results are the AGM rationality postulates for belief revision. They constitute properties that rational contraction and revision operations should satisfy. These postulates or criticism concerning them and resulting suggestions for modifications often are the starting point for belief revision approaches.

There are also subtle differences in the notion of what is represented in the knowledge of an agent and therefore of what the new information represents. Generally, belief revision is viewed in the light of an agent receiving information about a static world. The goal is that after each revision step the agent's representation of the world is more correct than before. New contradicting information means that previous assumptions about the world

were wrong and have to be revised.

A different notion – termed belief update – assumes the world to be dynamic. That is, contradicting information represents changes that have occured in the world. So while belief revision tries to decide which beliefs should be contracted to allow a new one to be consistently inserted, belief update tries to determine what changes in the world have caused the new observation.

## 1.2 Motivation

Consider the following belief revision framework [1][1]. The epistemic state of an agent is represented by a sequence of propositions $\sigma = (\varphi_1, ..., \varphi_n)$ of a language. For now, let that language be the usual propositional language $L_p$, consisting of propositional letters and being closed under the classical connectives. $\sigma$ could be interpreted as the sequence of inputs the agent received, considering later propositions more reliable. The belief set $Bel(\sigma)$ corresponding to this epistemic state is calculated iteratively, starting with the set of tautologies and beginning at the back of the sequence adding one formula at a time – if this can be done consistently.

In other words, propositions from the sequence that would produce an inconsistent belief set are simply left out. It is obvious that the sequence could also be interpreted as a linear ordering regarding importance of the propositions.

**Definition 1.1.**

$$Bel_i(\sigma) \stackrel{\text{def}}{=} \begin{cases} Cn_{L_p}(\emptyset), i = 0 \\ BT_i \stackrel{\text{def}}{=} Cn_{L_p}(Bel_{i-1}(\sigma) \cup \{\varphi_{n+1-i}\}), 0 < i \leq n \, and \perp \notin BT_i \\ Bel_{i-1}(\sigma), otherwise \end{cases}$$

$$Bel(\sigma) \stackrel{\text{def}}{=} \lim_i Bel_i(\sigma)$$

**Remark 1.** $Bel(\sigma) = Bel_n(\sigma)$

The equation in Remark 1 follows immediately from the definition of $Bel$. As there are no further proposition left in $\sigma$ after n steps of the calculation, $Bel_i$ would not change any more.

[1] Notation and definitions are our own. If they do not correctly reflect the intentions of the originators of the framework, we are to blame.

### Revision

Revision, i.e. the consistent incorporation of information, of $\sigma$ by a proposition $\psi$ is realized by appending it to the sequence.

**Definition 1.2.** $(\varphi_1, \ldots, \varphi_n) * \psi \stackrel{\text{def}}{=} (\varphi_1, \ldots, \varphi_n, \psi)$

**Example 2.**

- $\sigma_1 = (a, \neg a)$:
  $Bel((a, \neg a)) = Cn_{L_p}(\{\neg a\})$
  $Bel((a, \neg a) * a) = Cn_{L_p}(\{a\})$
  At first $a$ is left out, because adding it to the belief set would conflict with $\neg a$ which has higher priority. After the revision, $\neg a$ is left out, because the new information conflicting with it has higher priority.

- $\sigma_2 = (a, a \to b, \neg b)$:
  $Bel((a, a \to b, \neg b)) = Cn_{L_p}(\{a \to b, \neg b\})$
  $Bel((a, a \to b, \neg b) * c) = Cn_{L_p}(\{a \to b, \neg b, c\})$
  Together with the implication, $a$ would create an inconsistency. Consequently, it is left out. Revision with $c$ has no modifying impact on the previous belief set, as that proposition does not interact with anything earlier in the sequence, so it is just added.

- $\sigma_3 = (a, a \to b, \neg b)$:
  $Bel((a, a \to b, \neg b)) = Cn_{L_p}(\{a \to b, \neg b\})$
  $Bel((a, a \to b, \neg b) * a) = Cn_{L_p}(\{a, \neg b\})$
  First $a$ is left out as in the previous example, but after revision the implication is neglected as *now* it is the weakest link in the chain .

### Contraction

**Example 3.** An agent $A$ whose current beliefs consist of $\{\varphi\}$ only, learns from another agent $B$ that $\{\neg\varphi, \psi\}$ hold and accordingly changes its belief state to $\{\neg\varphi, \psi\}$. Afterwards a more reliable source informs $A$ that information obtained from $B$ cannot be trusted.

It would be rational to expect that agent $A$ now revises his beliefs to the old state. Without a contraction operation, however, there is no possibility of getting rid of $\psi$ other than revising by $\neg\psi$. But the result of this would not be the old belief state. This should

illustrate that contraction must be viewed as a separate operation and not just as an intermediate step for revision.

So, how can contraction of a proposition $\psi$ be modelled in our belief revision framework? One way of achieving it would be to remove all propositions that imply $\psi$ from the sequence.

**Example 4.** $(\varphi, \varphi \to \psi, \psi) - \psi = (\varphi \to \psi)$

This is not satisfactory, because information might be lost if the contraction turns out to have been unjustified. In the example the information that $\varphi$ was once part of the knowledge has disappeared. So the question is not just how to achieve contraction but also how to be able to undo contraction if necessary.

Of course, the agent could store a sequence of all prior sequences in order to regain information, but the resulting belief revision procedure would be very complicated. In addition, this method bears the risk of wasting a lot of memory.

We do not consider the contraction of $\psi$ by the introduction of $\neg\psi$ to be a desirable approach, either. There is an essential difference between removing knowledge and replacing it by its opposite, as could be seen in Example 3.

It would be nice to be able to just block $\psi$ from being introduced into the belief set. Therefore, we suggest to enrich the usual language of propositional logic $L_p$ by a means to denote disbeliefs which create an inconsistency without implying the negation.

## 1.3 Goals of this work

If we had that language and the corresponding logic, we could achieve contraction of a proposition $\varphi$ in terms of revision by the disbelief in $\varphi$. That is, we would not modify the belief revision framework, but use it with a different underlying logic which allows us to express both, revision and contraction, by a single operation.

The main goal of this work is to present a language that is capable of achieving the above mentioned and to develop a logic based on this language. We will prove soundness and completeness of the logic using a semantic approach. Although we will not provide an axiomatization of a deductive system, we give some results obtained in the attempt to find one.

We will also present a Prolog implementation of the logic developed. It will be based on the semantic approach. This implementation will allow to test properties of the logic

as well as mechanisms of the belief revision framework.

Viewing things from different angles often allows deeper insight into the mechanisms that lie behind them. That is why we will investigate the belief revision framework introduced above from the perspective of an argumentation framework.

We hope that this work will contribute to the research field of knowledge representation and especially to the relatively young research in rejection and disbelief.

## 1.4 Structure of this work

The remainder of the thesis is structured as follows:

In this section, we continue to introduce some the main fields of research our paper touches briefly. Section 2 introduces the extended language and lists desirable properties of the logic. Soundness and completeness of the proof theory are shown in Section 3. Section 4 provides information about the two implementation approaches. In Section 5 we continue to illustrate properties of the logic of disbelief and take some steps towards its axiomatization . A transformation of the belief revision framework introduced in the motivation into an argumentation framework is given in Section 6. Section 7 concludes with a summary of the results and with aspects of further work.

## 1.5 Logics of Disbelief

Many logics have been developed in order to increase the expressibility of classical propositional logic which does not suffice for some purposes, or to capture decidable fragments of first or second order logic which in many cases are too general. Logics have been devised to capture notions of modalities like "it is known", "it is necessary", "it is possible", or notions of time. Other logics have been developed in order to be able to talk about objects and their relationships, to reason about points or regions in space, to reason about actions, etc.

Usually these logics are based on the idea that a set of propositions is assumed to be true and that the question is what can be inferred from them, i.e. what else is true if some premises hold. The question of what is not believed to hold and of inferring from that what else should be rejected is mostly neglected or reduced to classical negation. Like contraction in belief revision, rejection in logic has seldom been seen as a primitive concept.

However, there are some approaches which attempt to overcome this shortcoming. They usually expand the formalism by a set of propositions dual to the classical approaches. Besides a set of propositions that are assumed to be true, i.e. believed, they consider another one with propositions that are disbelieved or rejected. A consequence relation is then defined which does not only handle inferences from the set of beliefs but also inferences from the set of disbeliefs or even from both sets. This can result in new beliefs and new disbeliefs.

In some cases the two sets are only connected via a notion of inconsistency, i.e. inference is done separately on every set and then a consistency condition is checked – for example, that no proposition is included in both sets. In other cases, the consequence relation considers both sets, i.e. there is interaction between beliefs and disbeliefs.

Another issue that motivated the consideration of disbeliefs was the desire to model contraction explicitly. To illustrate this we want to adopt an example given in [2].

**Example 5.** An ignorant agent revises its belief base with $a \to b$, then contracts $b$, then revises with $a$. If the revision framework does not allow the explicit representation of disbeliefs there is *only* one possible result – the agent's belief set contains $\{a, a \to b\}$. As $b$ was not in the belief set when the contraction was executed, nothing changed and then the contraction was forgotten.

If contraction was modelled in terms of disbeliefs denoted $\overline{b}$ for the disbelief in $b$, other beliefs sets would be possible. Depending on the order on the propositions, $\left\{\overline{b}, a \to b\right\}$ and $\left\{a, \overline{b}\right\}$ would be acceptable outcomes of the above revision sequence as well.

## 1.6 Many-valued Logics

Considerations about logics with more than two truth values go back at least to Aristotle who discussed how to evaluate propositions talking about the future. The answer to this is closely related to the philosophical question of whether future events are already determined. One possible but not inescapable way of dealing logically with nondeterminism is to discard the principle of the excluded middle and interpret a third value as "possible" or "undefined".

The works of Lukasiewicz and Post in the 1920s stand at the beginning of many-valued logics as a separate research area. Important early results were the axiomatization of Lukasiewicz' three-valued logic and its expansion to a truth functional complete logic.

However, numerous other logics have been devised and investigated.

In general, many-valued logics are similar to classical propositional logic in that they obey the principle of truth functionality, i.e. the value of a proposition is uniquely determined by the value of its components. There is no standard interpretation for the truth values, but usually there are values 0 and 1 that correspond to $true$ and $false$ in standard propositional logic. If reduced to these two values, most many-valued logics behave like the classical one. The interpretation of the truth values depends on the domain for which the logic is used.

In many cases the truth values are a finite or infinite set of rational or real numbers in the interval $[0, 1]$ and the connectives are defined in terms of mathematical functions usually including the minimum and maximum of a set.

Many-valued logics are applied in linguistics, logic, philosophy, hardware design, and mathematics. Probably most successful in application is the theory of fuzzy sets and the resulting fuzzy logic, which is often used for reasoning with uncertainty, i.e. vague information.

# 2 The language of disbelief and basics of its logic

## 2.1 Syntax of the language

We extend the language of propositional logic $L_p$ by introducing the unary connective *bar*, which denotes the disbelief in the proposition below it, i.e. $\overline{\varphi}$ denotes the disbelief in $\varphi$.

**Definition 2.1.** A set $L$ is a *language of disbelief* if it is the smallest set satisfying the following properties.

(i) The propositional letters $a$, $a_1$, $a_2, \ldots$, $b$, $b_1, \ldots$ are elements of $L$.

(ii) If $\varphi \in L$ and $\psi \in L$, then $(\varphi \wedge \psi) \in L$, $(\varphi \vee \psi) \in L$, $(\varphi \to \psi) \in L$, $(\neg\varphi) \in L$ and $\overline{\varphi} \in L$.

**Example 6.**

- $(a \wedge b)$, $\left(a \vee \overline{b}\right)$, $\overline{(\overline{a} \to b)}$ are elements of $L$.
- $(a \wedge \vee b)$, $\overline{(a \to} b)$, $(a\,(\neg\wedge)\,b)$ are not elements of $L$.

In the remainder of this paper, we will often simplify notation by omitting parentheses if doing so does not cause ambiguity.

**Remark 7.** In contrast to the logics introduced in [2], we allow the nesting of the disbelief bar. The language considered in this paper is restricted to elements $\varphi$ and $\overline{\varphi}$, where $\varphi$ is a classical proposition. As seen in Example 6, our language contains sentences of a more general structure.

It is possible to think of the connective bar as a second type of negation. However, it will be seen that its behavior makes the logic a lot more complicated.

**Definition 2.2.** Let $L$ be a language of disbelief. An element $\varphi$ of $L$ is called a *disbelief* if its main connective is the unary connective bar. Otherwise it is called a *belief*. All elements of $L$ are called *propositions*.

## 2.2 Remarks on desirable properties of the logic

We want to keep as many properties of classical logic as possible. That is why we do not want any proposition of the form $\varphi \wedge \neg\varphi$ to have a model. That is, any superset of

$\{\varphi, \neg\varphi\}$ is inconsistent. The same should hold for a proposition and the disbelief in it, i.e. $\varphi \wedge \overline{\varphi}$ and therefore $\{\varphi, \overline{\varphi}\}$ is not to be satisfiable. This corresponds to the intuition that believing and disbelieving in the same thing at the same time is inconsistent.

In order to be able to express agnosticism, we want our framework to have models for propositions that denote the disbelief in a proposition and at the same time the disbelief in its negation. That is, in contrast to $\neg\varphi \wedge \neg\neg\varphi$, $\overline{\varphi} \wedge \overline{\neg\varphi}$ should be consistent and therefore should have a model. The intuitive reading is that information concerning $\varphi$ is rejected. If that was not possible, disbelief would break down to something very similar to classical negation.

In order to capture this notion with models satisfying a set of propositions including disbeliefs, we propose to use a three-valued logic with the values *true* denoted by 1 (proposition believed), *false* denoted by 0 (negation of proposition believed), *undefined* denoted by $u$.

We want to back up this proposal with the following example. Consider the proposition $\overline{\varphi} \wedge \overline{\neg\varphi}$ and a two-valued logic, i.e. one in which every propositional variable is assigned either 0 or 1. As required above, the proposition should have a model, i.e. there should be an assignment that evaluates it to 1. For now, we further assume that an intuitive conjunction is evaluated to 1 if and only if all its conjuncts are evaluated to 1 as well. This assumption is all the more natural because we consider a two-valued logic. Since $\overline{\varphi} \wedge \overline{\neg\varphi}$ is to be true $\overline{\varphi}$ and $\overline{\neg\varphi}$ must take the value 1. Now it becomes obvious that any assignment to $\varphi$ causes a problem. Assigning 1 to $\varphi$ results in trying to satisfy $\{\varphi, \overline{\varphi}\}$, which we demanded not to be possible. The same problem arises with the assignment of 0 to $\varphi$, which is equivalent to assigning 1 to $\neg\varphi$. Now we would be forced to satisfy $\{\neg\varphi, \overline{\neg\varphi}\}$. Consequently, a third value appears necessary which $\varphi$ has to take if $\overline{\varphi} \wedge \overline{\neg\varphi}$ is to be evaluated to true.

It seems to be a reasonable demand that $\overline{\neg\varphi}$ be a logical consequence of $\varphi$, i.e. that an assignment of 1 to $\varphi$ implies an assignment of 1 to $\overline{\neg\varphi}$. This would correspond to the intuition that if $\varphi$ is believed its negation *cannot* be believed, so it is disbelieved. Using substitution, contraposition, and elimination of double negation, we get the following implications:

- $\varphi$ implies $\overline{\neg\varphi}$
- $\neg\varphi$ implies $\overline{\varphi}$

- $\neg\overline{\neg\varphi}$ implies $\neg\varphi$
- $\neg\overline{\varphi}$ implies $\varphi$

Thinking back to the motivation, we did not just want to be able to model contraction, but also to undo it again if necessary. It is not enough to have blockers that keep propositions from being introduced into the belief set, we also need blockers for the blockers.

It is natural that if $\overline{\varphi}$ is a blocker for $\varphi$, $\overline{\overline{\varphi}}$ should be the one for $\overline{\varphi}$.

**Example 8.**

$$\sigma_0 = (\ldots)$$

$$\sigma_1 = (\ldots) * \overline{\varphi} \wedge \overline{\neg\varphi}$$

$$\sigma_2 = (\ldots, \overline{\varphi} \wedge \overline{\neg\varphi}) * \overline{\overline{\varphi}} \wedge \overline{\overline{\neg\varphi}}$$

$$\sigma_2 = \left(\ldots, \overline{\varphi} \wedge \overline{\neg\varphi}, \overline{\overline{\varphi}} \wedge \overline{\overline{\neg\varphi}}\right)$$

Now consider Example 8 . It sketches parts of a possible development in the belief revision framework. Obviously, the underlying logic is a logic of disbelief.

We have an initial belief state $\sigma_0$. Then $\varphi$ and its negation are simultaneously contracted by introducing their corresponding disbeliefs. This could have been done in two separate revision steps as well. $\varphi$ as well as its negation are now blocked from being introduced into the belief set, i.e. an assignment of the truth value $u$ to $\varphi$ is forced.

Then this contraction of information about $\varphi$ is undone by blocking it via $\overline{\overline{\varphi}} \wedge \overline{\overline{\neg\varphi}}$[2]. Intuitively, whatever the value of $\varphi$ was in $\sigma_0$, it should be the same as induced by $\sigma_2$. However, this is not the case. Believing $\varphi$ conflicts with $\overline{\overline{\neg\varphi}}$. That is because $\varphi$ implies $\overline{\neg\varphi}$ which is inconsistent with $\overline{\overline{\neg\varphi}}$. Likewise, $\neg\varphi$ conflicts with $\overline{\overline{\varphi}}$, as $\neg\varphi$ implies $\overline{\varphi}$. In consequence, $\varphi$ is still restricted to the value $u$, which is counterintuitive. This undesirable property is caused by the fact that the implication "$\varphi$ implies $\overline{\neg\varphi}$" creates interaction between propositions and the blockers of their blockers. Thereby, we have shown that the demand for $\overline{\neg\varphi}$ to be a logical consequence of $\varphi$ is not reasonable considering our purposes. Interaction should be restricted to propositions and disbeliefs one order higher.

[2]Note that $\overline{\overline{\varphi} \wedge \overline{\neg\varphi}}$ would not necessarily achieve the same, as this would demand only one of the two blockers to be removed.

Demanding, in contrast, that if $\varphi$ is assigned 1, $\overline{\neg\varphi}$ is to be assigned $u$, solves the problem only partially. This would mean that the intuitively rational proposition $\neg\varphi \wedge \overline{\varphi}$ would not be satisfiable, which is just as counterintuitive.

The implication "if $\varphi$ is assigned 1, then $\neg\overline{\varphi}$ is assigned 1 as well" is not desirable, either. Consider $\left(\varphi \rightarrow \overline{\psi}\right) \wedge \psi$. This proposition obviously implies $\neg\varphi$. Certainly, if $\psi$ was true, $\varphi$ should not be, but jumping to the conclusion that its negation must hold is too strong a demand – assigning $u$ to $\varphi$ intuitively is just as right.

The way out of this dilemma is to refrain from restricting $\overline{\varphi}$ to one single value if one value is given for $\varphi$. This breaks the principle of extensionality. The value of a proposition is no longer determined by the value assigned to the propositional variables it contains. The disbelief operator is not truth functional.

## 2.3 Truth Tables for the classical connectives

As noted above, our logic will be a three-valued one. So we have to extend the truth tables for all the connectives to the third value *undefined*. Again, we will give an intuitive justification for our choice rather than a strictly formal one.

Interpreting $u$ as: there is no information about the value, it might be $true$, it might be $false$, it might be neither; we arrive at the following definitions for the connectives.

The truth value of a conjunction $\varphi \wedge \psi$ should be the same as in the classical case when the only values involved are 0 and 1. If $\varphi$ is 0, the conjunction can only be 0 too. Even if, later on, the value of $\psi$ turns out to be true, this does not change anything. If $\varphi$ is 1, the value of the conjunction depends heavily on $\psi$. For 0 and 1, the case is clear, as mentioned above; but what if $\psi$ is $u$? We think that in this case the conjunction should be evaluated to $u$ as well. Depending on the future value of $\psi$, it can change to 0 or 1, so it is undefined until the value is known. These considerations are symmetric, of course.

Similar considerations apply to the disjunction $\varphi \vee \psi$. If $\varphi$ is 1, the disjunction is evaluated to 1 too. If $\varphi$ it is 0, however, its value depends on $\psi$ only. And in case of $\psi$ being $u$, the value of the disjunction should be undefined as well, since a future value of $\psi$ might make it true or false.

By intuitive considerations, we have thus arrived at Lukasiewicz's definitions for weak conjunction and disjunction of a three-valued logic. The negation is quite obvious, as well. Using similar argumentation as above, the value of a negated $u$ remains undefined.

| $\varphi \wedge \psi$ | 0 | $u$ | 1 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| $u$ | 0 | $u$ | $u$ |
| 1 | 0 | $u$ | 1 |

| $\varphi$ | $\neg\varphi$ |
|---|---|
| 0 | 1 |
| $u$ | $u$ |
| 1 | 0 |

| $\varphi \vee \psi$ | 0 | $u$ | 1 |
|---|---|---|---|
| 0 | 0 | $u$ | 1 |
| $u$ | $u$ | $u$ | 1 |
| 1 | 1 | 1 | 1 |

| $\varphi \rightarrow \psi$ | 0 | $u$ | 1 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| $u$ | $u$ | 1 | 1 |
| 1 | 0 | $u$ | 1 |

| $\varphi \leftrightarrow \psi$ | 0 | $u$ | 1 |
|---|---|---|---|
| 0 | 1 | $u$ | 0 |
| $u$ | $u$ | 1 | $u$ |
| 1 | 0 | $u$ | 1 |

Table 1: Truth Tables for classical connectives

The definition for the implication $\varphi \rightarrow \psi$ is not quite as straightforward, though. If it is defined in terms of negation and disjunction, as in the classical case, its value would be $u$ given that $\varphi$ and $\psi$ are evaluated to $u$. Intuitively, that makes perfect sense, because, depending on the future values of the propositions, the implication could take any value.

If we do that, however, there will be no tautologies in our logic, which can easily be seen by assigning $u$ to every propositional variable and disbelief – which would cause every proposition to be evaluated to $u$. Therefore, we use a separate definition for implication. It differs from the intuitively right one in that it evaluates the implication not to $u$ but to 1 if $\varphi$ and $\psi$ are assigned $u$. This means that implication and disjunction are no longer interdefinable via negation.

We further define $\varphi \leftrightarrow \psi$ as $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. This corresponds to the notion that two propositions are equivalent if they are evaluated to the same value under all truth assignments.

Table 1 summarizes our definition of the classical connectives.

**Remark 9.** We are aware of the fact that the implication as we defined it has semantical shortcomings. When using the logic in an application, though, it will always be possible to fall back on negation and disjunction to capture the other notion of implication.

## 2.4 Laws of the classical connectives

Using the above definitions, many of the properties of the connectives still hold. Two of the rules that do *not* hold any more are $\varphi \wedge \neg\varphi = \bot$ and $\varphi \vee \neg\varphi = \top$. They basically represent the principle of the excluded middle, so it is not surprising that we cannot use

them any longer as we explicitly allow a third value.

The following rules can be used to transform propositions syntactically without changing their truth value. Consequently, these rules can be used to characterize the equivalence relation between propositions later on.

**Proposition 2.3.** *The following equations hold.*

| | | |
|---|---|---|
| *Associativity* | $(\varphi \wedge \psi) \wedge \chi = \varphi \wedge (\psi \wedge \chi)$ | $(\varphi \vee \psi) \vee \chi = \varphi \vee (\psi \vee \chi)$ |
| *Commutativity* | $\varphi \wedge \psi = \psi \wedge \varphi$ | $\varphi \vee \psi = \psi \vee \varphi$ |
| *Distributivity* | $(\varphi \vee \psi) \wedge \chi = \varphi \wedge \chi \vee \psi \wedge \chi$ | $(\varphi \wedge \psi) \vee \chi = (\varphi \vee \chi) \wedge (\psi \vee \chi)$ |
| *Absorption* | $\varphi \wedge (\varphi \vee \psi) = \varphi$ | $\varphi \vee (\varphi \wedge \psi) = \varphi$ |
| *Idempotency* | $\varphi \wedge \varphi = \varphi$ | $\varphi \vee \varphi = \varphi$ |
| *de Morgan* | $\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$ | $\neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$ |
| *Double negation* | $\neg\neg\varphi = \varphi$ | |
| *Laws for $\bot$ and $\top$* | $\varphi \wedge \top = \varphi$ | $\varphi \vee \top = \top$ |
| | $\varphi \wedge \bot = \bot$ | $\varphi \vee \bot = \varphi$ |

*Proof.* We show the validity of two of these equations by calculating the truth value of the proposition given assignments of their atomic components. In order to save space, we write all values of $\chi$ in one row.

| $\varphi$ | $\psi$ | $\chi$ | $(\varphi \vee \psi) \wedge \chi$ | $\varphi \wedge \chi \vee \psi \wedge \chi$ | $\neg(\varphi \wedge \psi)$ | $\neg\varphi \vee \neg\psi$ |
|---|---|---|---|---|---|---|
| 0 | 0 | $0u1$ | 000 | 000 | 1 | 1 |
| 0 | $u$ | $0u1$ | $0uu$ | $0uu$ | 1 | 1 |
| 0 | 1 | $0u1$ | $0u1$ | $0u1$ | 1 | 1 |
| $u$ | 0 | $0u1$ | $0uu$ | $0uu$ | 1 | 1 |
| $u$ | $u$ | $0u1$ | $0uu$ | $0uu$ | $u$ | $u$ |
| $u$ | 1 | $0u1$ | $0u1$ | $0u1$ | $u$ | $u$ |
| 1 | 0 | $0u1$ | $0u1$ | $0u1$ | 1 | 1 |
| 1 | $u$ | $0u1$ | $0u1$ | $0u1$ | $u$ | $u$ |
| 1 | 1 | $0u1$ | $0u1$ | $0u1$ | 0 | 0 |

The rest of the equations can be verified analogously using truth tables. □

Using the de Morgan rules, conjunction and disjunction are interdefinable. So we would not restrict the expressibility of our language by omitting either conjunction or disjunction.

| | | $\varphi$ | | |
|---|---|---|---|---|
| | | 0 | $u$ | 1 |
| | 0 | $a$ | $b$ | $c$ |
| $\overline{\varphi}$ | $u$ | $d$ | $e$ | $f$ |
| | 1 | $g$ | $h$ | $i$ |

Table 2: Possible constellations of a proposition and the disbelief in it

## 2.5 Restrictions to the connective *bar*

In Section 2.2, we have seen that determining the value of a disbelief $\overline{\varphi}$ by the value assigned to $\varphi$ does not work, but still a proposition and the disbelief in it are not independent of each other. What then is their relationship - which constellations of truth assignments should be possible?

Table 2 shows all the combinations between a proposition and the disbelief in it. It is clear that the combination $i$ must not be allowed because belief in a proposition and, at the same time, disbelief in it would contradict the purpose of the disbelief framework. In fact, it was our starting point in Section 2.2 that this combination was to be inconsistent.

Combinations $c$ and $g$ should be allowed as they correspond to our intuition. Combinations $d$ and $f$ must be allowed, in order not to cause the counterintuitive results of the implications discussed in the context of Example 8.

The combinations $b$, $e$ and $h$ should be allowed as well. Whenever we have no definite information about one side, all should be possible for the other.

Combination $a$ is a difficult case, though. We propose not to allow it. On the one hand, allowing it would mean that there is no distinction between assigning 0 or $u$ to a disbelief. On the other, not allowing it would mean that $\neg\overline{\varphi} \wedge \neg\overline{\neg\varphi}$ implies that $\varphi$ is assigned $u$ which corresponds to the intuition, because the proposition reads: it is not the case that $\varphi$ is disbelieved (evaluation of $\varphi$ to true is favoured) and it is not the case that $\neg\varphi$ is disbelieved (evaluation of $\varphi$ to false is favoured), so an assignment to $u$ is the only reasonable compromise.

**Remark 10.** The table for $\varphi$ and $\overline{\neg\varphi}$ can be constructed from this one by substituting $\neg\varphi$ for $\varphi$ and then interchanging the left and the right column, thereby undoing the negation of $\varphi$.

| | | $\varphi$ | | |
|---|---|---|---|---|
| | | 0 | $u$ | 1 |
| | 0 | - | + | + |
| $\overline{\varphi}$ | $u$ | + | + | + |
| | 1 | + | + | - |

| | | $\varphi$ | | |
|---|---|---|---|---|
| | | 0 | $u$ | 1 |
| | 0 | + | + | - |
| $\overline{\neg\varphi}$ | $u$ | + | + | + |
| | 1 | - | + | + |

Table 3: Valid constellations for a proposition and the disbelief in it

Table 3 shows the possible constellations of $\varphi$ and $\overline{\varphi}$ – "+" denoting allowed, "-" denoting not allowed.

**Remark 11.** Note that $\varphi$ and $\overline{\overline{\varphi}}$ are independent of each other. All constellations between the two are possible as $u$ can be assigned to $\overline{\varphi}$.

## 2.6 Truth assignment and valuation

There have been several approaches to a definition of truth assignment and valuation. One of the goals was to keep the structure of the models as simple as possible. Obviously it will not work to assign values to the propositional letters alone, since the connective "disbelief" is not truth functional.

One approach was based on the idea of simplifying all propositions via de Morgan-like rules. There would have been a definition that $\overline{\varphi \wedge \psi}$ is equivalent to $\overline{\varphi} \vee \overline{\psi}$ and $\overline{\varphi \vee \psi}$ to $\overline{\varphi} \wedge \overline{\psi}$. This would have allowed a simplification to *atomic* disbeliefs. These rules would have been sound as the propositions would allow the same truth values for $\overline{\varphi \wedge \psi}$ and $\overline{\varphi} \vee \overline{\psi}$ etc. given assignments to $\varphi$ and $\psi$[3]. There is no justified simplification for the disbelief in an implication, however, which is why this approach was abandoned.

The approach we finally chose combines an assignment with a consistency condition.

**Definition 2.4.** Let $L$ be a language of disbelief. A *truth assignment* $A$ is a function that assigns a unique truth value $A(\alpha), A(\overline{\varphi}) \in \{0, u, 1\}$ to each propositional letter $\alpha \in L$ and to each disbelief $\overline{\varphi} \in L$.

Let $A$ be a truth assignment. A *truth valuation* $V_A$ is a function that assigns a unique truth value $V_A(\varphi)$ to each proposition $\varphi \in L$, such that $V_A(\psi) = A(\psi)$ for all propositional letters and disbeliefs $\psi$ and the value of a compound proposition is determined in

[3]But only when using the combinations defined in Table 3. Other constellations would not have had this property.

accordance with the truth tables (Table 1) from section 2.3.

**Proposition 2.5.** *Applying Definintion 2.4, a truth valuation $V_A$ is uniquely determined by the truth assignment A, i.e. there are no $V_A^1$ and $V_A^2$ such that $V_A^1(\varphi) \neq V_A^2(\varphi)$ for any $\varphi$.*

*Proof.* Assume there is a truth assignment $A$ that gives rise to two truth valuations $V_A^1$ and $V_A^2$ such that $V_A^1(\varphi) \neq V_A^2(\varphi)$ for a particular $\varphi$. By induction on the structure of $\varphi$, we continue this argument, ending in a contradiction and thereby proving the proposition.

- $\varphi$ is a propositional letter. $V_A^1(\varphi) \neq V_A^2(\varphi)$. Consequently $A(\varphi) = v_1$ and $A(\varphi) = v_2$ and $v_1 \neq v_2$. $A$ is not a truth assignment.
- $\varphi$ is a disbelief. $V_A^1(\varphi) \neq V_A^2(\varphi)$. Consequently $A(\varphi) = v_1$ and $A(\varphi) = v_2$ and $v_1 \neq v_2$. $A$ is not a truth assignment.

These two are the initial cases that end an induction and, as can easily be seen, they contradict the assumption that $A$ is a truth assignment.

- $\varphi$ is syntactically equivalent to $\neg\psi$. $V_A^1(\neg\psi) \neq V_A^2(\neg\psi)$. Consequently $V_A^1(\psi) \neq V_A^2(\psi)$. Inductively apply the proof.
- $\varphi$ is syntactically equivalent to $\varphi_1 \circ \varphi_2$, where $\circ \in \{\wedge, \vee, \rightarrow\}$.

  $V_A^1(\varphi_1 \circ \varphi_2) \neq V_A^2(\varphi_1 \circ \varphi_2)$. Consequently $V_A^1(\varphi_1) \neq V_A^2(\varphi_1)$ or $V_A^1(\varphi_2) \neq V_A^2(\varphi_2)$. Inductively apply the proof.

□

**Example 12.** (i) $A(a) = 1, A(\overline{a}) = 1$ is a truth assignment, but it contradicts our notion of disbelief.

(ii) $A\left(\overline{a \rightarrow b}\right) = 1, A(a) = u, A(b) = u$ like (i).

(iii) $A(\overline{a}) = 1, A(\overline{a \wedge a}) = u$ is a truth assignment, but our intention demands that "equivalent" formulae should be evaluated to the same truth value.

Example 12 illustrates that Definition 2.4 is not yet sufficient. It is shown that some truth assignments – and the truth valuations they give rise to – satisfy the definition but are still inconsistent with respect to the intuition we are trying to capture. This is due to the fact that the definition considers the truth tables for the classical connectives, but the value restrictions that come with the disbelief bar are not yet taken care of.

Example 12 (ii) makes clear why dealing with the value restrictions at the level of defining the truth assignment is very hard. In order to realize that the assignment is in some way inconsistent, the value for compound propositions has to be calculated.

Example 12 (iii) causes an additional dilemma. Usually equivalence of propositions is defined in terms of truth valuations, but now we need equivalence in order to define truth valuations. Tableau proofs introduced in Section 3 will provide a tool for determining which propositions are equivalent.

Our suggestion for dealing with these problems is to calculate the truth valuations of all possible truth assignments. We can then check which truth assignments give rise to truth valuations containing inconsistencies shown above and leave those out of further consideration.

**Definition 2.6.** A truth valuation $V_A$ as defined in definition 2.4 is *valid* if there is no proposition $\varphi$ such that one of the following holds:

(1) $V_A(\varphi) = 1$ and $V_A(\overline{\varphi}) = 1$

(2) $V_A(\varphi) = 0$ and $V_A(\overline{\varphi}) = 0$

(3) $V_A(\varphi) = 1$ and $V_A(\overline{\neg\varphi}) = 0$

(4) $V_A(\varphi) = 0$ and $V_A(\overline{\neg\varphi}) = 1$

(5) $\psi \equiv_L \varphi$ and $V_A\left(\overline{\psi}\right) \neq V_A(\overline{\varphi})$

A truth assignment $A$ is valid only if it gives rise to a valid truth valuation.

(1-4) make sure that a valid truth valuation is in accordance with the restrictions outlined in Section 2.5, (5) captures our intuition that "equivalent" propositions should be treated in the same way. Note that using equivalence for the definition of truth valuations leads to a somewhat circular definition of both. However, as will be seen later, this circle is broken by the use of tableaux.

**Remark 13.** There is no valid truth assignment such that $V\left(\overline{\top}\right) = 1$. Follows immediately from Definition 2.6 (property 1).

Remark 13 shows that disbelieving in a tautology is not consistently possible in our framework. We think this to be a nice property. A similar one holds for contradictions: $V\left(\overline{\bot}\right) = 0$ is not permitted.

The distinction between truth assignment and valid truth assignment is essential. Only valid truth assignments give rise to what we want to be models of a proposition. Nonvalid

truth assignments have an inconsistency built in and are therefore of no interest. In the remainder we will consider only valid truth assignments and omit the word valid.

We define the consequence relation between a set of propositions and a proposition in accordance with the classical notion.

**Definition 2.7.** Let $\Sigma$ be a set of propositions. We say that $\varphi$ is a *consequence of* $\Sigma$, denoted $\Sigma \models \varphi$, if for any truth valuation $V$,

$\forall \psi \, (\psi \in \Sigma \wedge V(\psi) = 1)$ implies $V(\varphi) = 1$.

We also define equivalence in the traditional way.

**Definition 2.8.** Two propositions $\varphi$ and $\psi$ are equivalent, $\varphi \equiv_L \psi$, if and only if for all truth assignments $A$: $V_A(\varphi) = V_A(\psi)$.

**Proposition 2.9.** *If the proposition $\psi$ does not contain the disbelief operator, $\overline{\varphi} \not\equiv \psi$ for any $\varphi$.*

*Proof.* (Sketch) It suffices to show the existence of a truth assignment that leads to an evaluation of $\psi$ and $\overline{\varphi}$ to different truth values.

As $\psi$ does not contain a disbelief bar, its value is determined by the assignment to the propositional letters alone. As a starting point, we chose the assignment that assigns the value $u$ to every propositional letter and disbelief. Further, we we leave the assignment fixed for the propositional letters, i.e. the value of $\psi$ is not going to change during the following considerations.

If the value of $\psi$ is 1 or 0, we already have the desired assignment. So the only case that remains to be considered is when $\psi$ takes the value $u$. And depending on the value of $\varphi$, the desired assignment can be constructed by chosing $\overline{\varphi}$ to be 1 or 0. □

# 3 Tableau proofs in the logic of disbelief

Axiomatizations for many-valued logics are often difficult. Semantic characterizations of the relations that are of interest are usually easier to find. The same holds for soundness and completeness proofs, in many cases.

The main aspect and a source of complexity of our disbelief logic lies in the compatibility of several truth assignments to $\varphi$ and $\overline{\varphi}$, e.g. the possibility of believing $\neg\varphi$ and at the same time not committing to disbelief in $\varphi$ (evaluating $\overline{\varphi}$ to $u$), as well as in the nesting of the disbelief bar. However, the specifications made in Sections 2.3, 2.5 can be captured elegantly with tableaux and a model-checking framework using tableau rules, which we want to present in the following. Definitions, lemmas and theorems 3.1 - 3.8 are adapted from [9].

## 3.1 Atomic tableaux and definitions

First we explain our notation. An atomic tableau is a tree with a root entry $\varphi : v_0$ and n leaves with the entries $c_1 \ldots c_n$. In our case, n will be 1, 2 or 3. Every $c_i$ is a possibly empty list of signed propositions $(\psi_{i1} : v_{i1}\,, ..., \psi_{in} : v_{in})$. $\varphi$, $\psi_{i1}, ..., \psi_{in}$ are propositions of a language of disbelief and $v_0, ..., v_{in}$ are members of $\{0, u, 1, <1, >0\}$ which represent value restrictions. $<1$ and $>0$ are abbreviations for $\{0, u\}$ and $\{u, 1\}$ as will become clear from the rules (t1) and (t2) in Figure 2.

| $\varphi : 1$ | $\varphi : u$ | $\varphi : 0$ | $\varphi :< 1$ | $\varphi :> 0$ |
|---|---|---|---|---|
| (r1) | (r2) | (r3) | (r4) | (r5) |

Figure 1: Valid roots of a tableau

Figure 1 shows which entries are allowed at the root of a tableau.

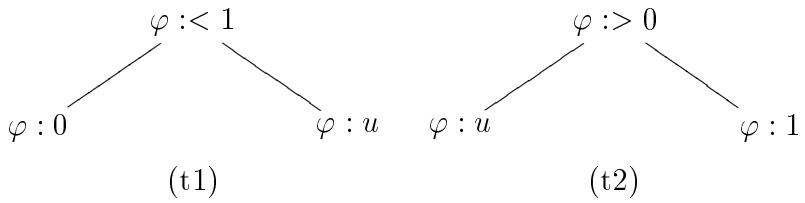



Figure 2: Abbreviations

The atomic tableaux (t1) and (t2) in Figure 2 define the meaning of the abbreviations. If $\varphi$ is restricted to a value less than 1, it can take the values 0 or $u$. The case of a restriction

to a value greater than 0 is analogous.

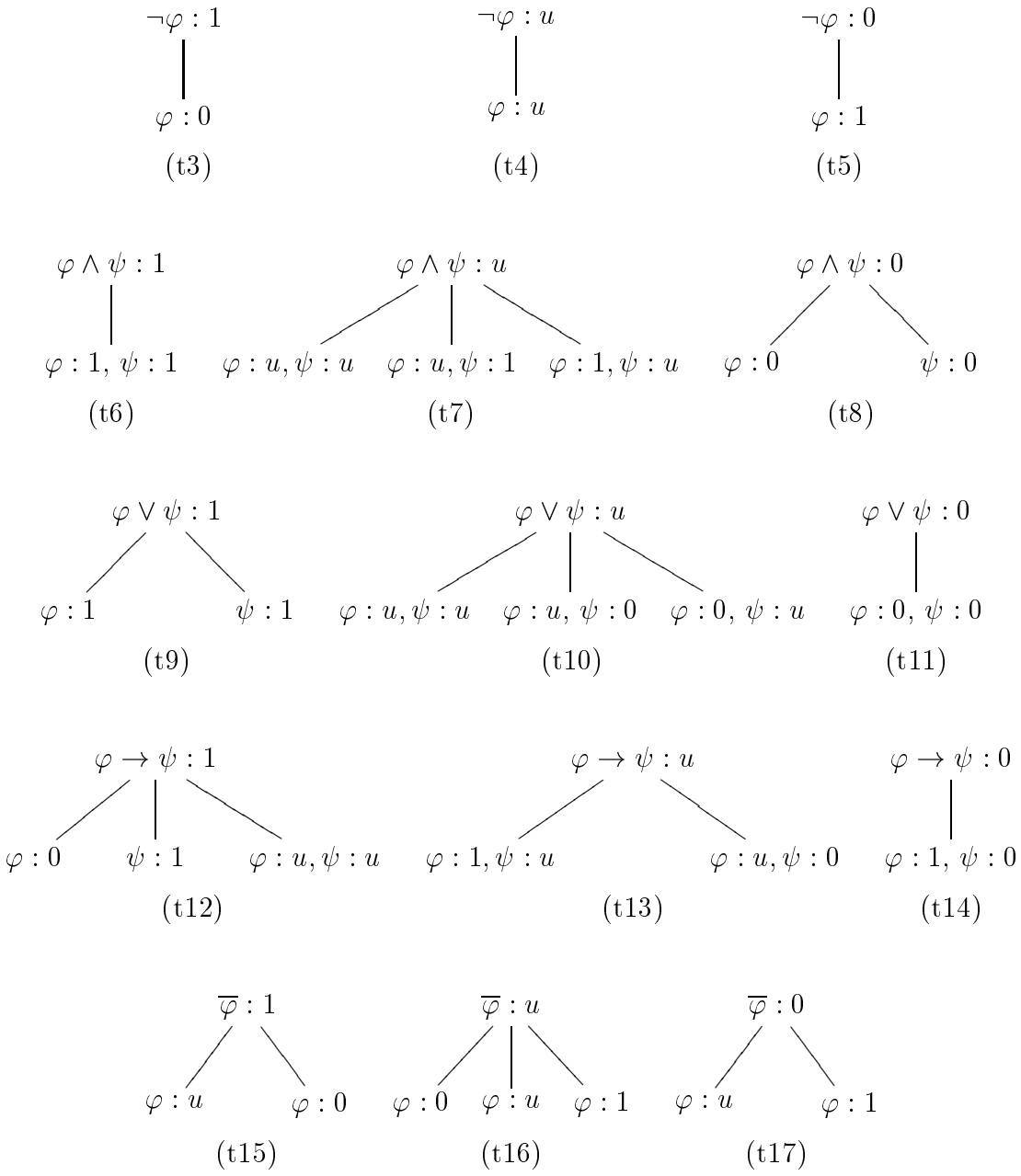



Figure 3: Atomic tableaux for the connectives

The atomic tableaux in Figure 3 precisely encode the truth tables presented in Section 2.3 and the restrictions caused by the disbelief bar presented in Section 2.5. (t1-17) represent the atomic tableaux.

The basic idea behind the tableau approach is to create a model – or to show that no model exists. Starting with an initial goal, all possible paths of achieving this goal are tried. Sometimes there is only one path, like in (t6). Sometimes there are different possibilities, like in (t9). In these cases all of the paths have to be investigated separately.

Arriving at an inconsistency on one path tells us that chosing it was not successful. If none of the paths are successful, the initial goal cannot be achieved. A consistent path that cannot be expanded further gives us a solution for the initial goal.

Now the notions of what a path is, of how we define inconsistency, of when a path cannot be expanded further have to be defined more formally.

**Definition 3.1.** A *finite tableau* is a binary tree, labelled with signed propositions called entries, that satisfies the following inductive definition:

(i) All atomic tableaux are finite tableaux.

(ii) If $T$ is a finite tableau, $P$ a path on $T$, $E$ an entry on $P$ and $T'$ is obtained from $T$ by adjoining the unique atomic tableau with root entry $E$ at the end of the path $P$, then $T'$ is also a finite tableau.

If $T_0, T_1, ..., T_n, ...$ is a sequence of finite tableaux, such that, for each $n \geq 0$, $T_{n+1}$ is constructed from $T_n$ by an application of (ii), then $T = \bigcup T_n$ is a tableau.

**Remark 14.** Some of the atomic tableaux are in fact not binary but ternary. This is to represent the third value more naturally and improve readability. It should be clear, though, that those can be easily transformed to be binary trees.

**Definition 3.2.** Let $T$ be a tableau, $P$ a path on $T$ and $E$ an entry occuring on $P$.

(i) $E$ has been *reduced* on $P$ if all entries on one path through the atomic tableau with root $E$ occur on $P$.

(ii) $P$ is *contradictory* [4] if

(1) for some proposition $\varphi$,

$\varphi : 0$ and $\varphi : 1$,

$\varphi : 0$ and $\varphi : u$, or

$\varphi : u$ and $\varphi : 1$ are both entries on $P$, or

(2) for two propositions $\overline{\varphi}$ and $\overline{\psi}$ with $\varphi \equiv_L \psi$, $\overline{\varphi} : v_1$ and $\overline{\psi} : v_2$ are entries on $P$ such that $v_1 \neq v_2$.

(iii)$P$ is *finished* if it is contradictory or every entry on $P$ is reduced on $P$.

(iv) $T$ is *finished* if every path through $T$ is finished.

(v) $T$ is *contradictory* if every path through $T$ is contradictory.

[4] We will sometimes say "the branch is closed" instead of "the branch is contradictory" and "open" instead of "noncontradictory".

**Remark 15.** Condition (2) of Definition 3.2 is an essential difference compared to a normal tableau. See section 3.3 for an explanation that equivalence can be defined in terms of tableau proofs in spite of this condition, which seems to make the definition circular.

**Example 16.**

- $\varphi \wedge (\psi \vee \chi) : 1$
  |
  $\varphi : 1, \psi \vee \chi : 1$

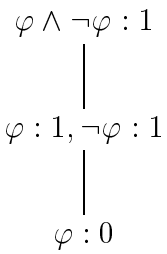
  $\psi \vee \chi : 1$ is not reduced on the only path $P$ in this tableau, as not all entries of a path through the atomic tableau with root entry $\psi \vee \chi : 1$ occur on $P$. So the tableau is not finished.

- $\varphi \wedge (\psi \vee \chi) : 1$
  |
  $\varphi : 1, \psi \vee \chi : 1$
  |
  $\psi : 1$

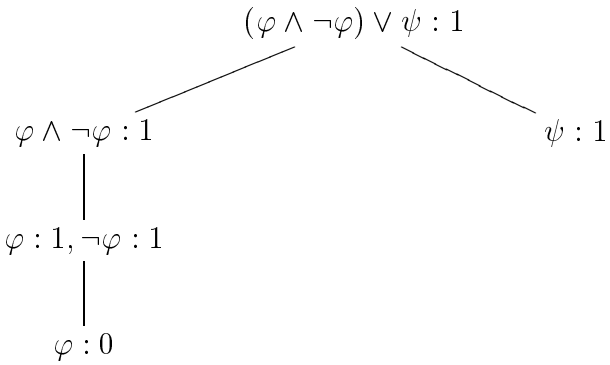
  $\psi \vee \chi : 1$ is reduced on the only path $P$ in this tableau, as all entries of a path through the atomic tableau with root entry $\psi \vee \chi : 1$ occur on $P$. So this tableau is finished.

- $(\varphi_1 \wedge \varphi_2) \wedge (\psi \vee \chi) : 1$
  |
  $\varphi_1 \wedge \varphi_2 : 1, \psi \vee \chi : 1$
  / \
  $\psi : 1$ $\chi : 1$
  |
  $\varphi_1 : 1, \varphi_2 : 1$

  $\psi \vee \chi : 1$ is reduced in the tableau, as on every single path, we find all the entries of a path through the atomic tableau with root entry $\psi \vee \chi : 1$. $\varphi_1 \wedge \varphi_2 : 1$ is reduced on the left path of the tableau, but not on the right one. Therefore the tableau is not finished.

- 


  This tableau is finished, as all entries are reduced on all paths. Furthermore it is contradictory, as its only path is contradictory. The path is contradictory, because there are two entries that demand one and the same proposition to have different values.

- 


  This tableau is finished but not contradictory, as it has a nonclosed branch.

**Definition 3.3.** A *tableau proof* of a proposition $\varphi$ is a contradictory tableau with root entry $\varphi :< 1$. A proposition is *tableau provable*, written $\vdash_t \varphi$, if it has a tableau proof.

A *tableau refutation* for a proposition $\varphi$ is a contradictory tableau starting with $\varphi :> 0$. A proposition is *tableau refutable* if it has a tableau refutation.

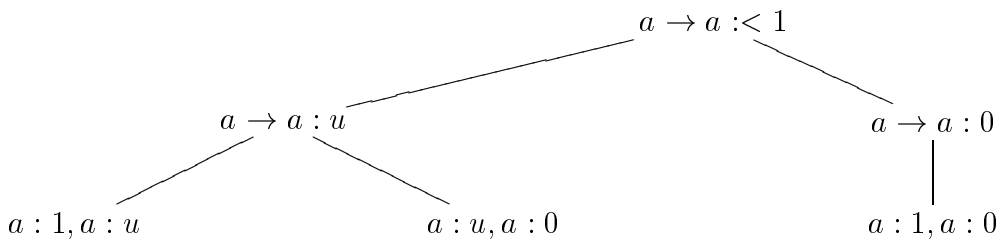



Figure 4: Example Proof

**Example 17.** Figure 4 is a tableau proof for $a \to a$.

It is clear that open branches in a finished tableau represent possible models for the propositions at the root of the tableau. Hence satisfiability of $\varphi$ can be checked with a

tableau with root entry $\varphi : 1$. If all branches are closed it is not satisfiable, if there is an open branch, it is.

It can be shown, that the proof for a proposition is always finite.

**Theorem 3.4.** *There is a finished finite tableau $T$ for each possible root entry $\varphi :< 1$.*

*Proof.* In fact we prove that any finished tableau with a root entry corresponding to (r1) - (r3) is finite. The theorem then follows, because corresponding to (t1) we can combine the finite finished tableaux $\varphi : 0$ and $\varphi : u$ with the root entry $\varphi :< 1$ to get a finished tableau that is still finite.

We proceed by induction on the structure of the given proposition $\varphi$. If $\varphi$ is a propositional letter, then the tableau consisting of just the signed propositional letter is finished. Note that signed propositional letters are themselves atomic tableaux.

For the inductive case consider (t8) $\varphi \wedge \psi : 0$. By induction there are finished finite tableaux $T_\varphi$ and $T_\psi$ with root entries $\varphi : 0$ and $\psi : 0$ respectively. We form the desired tableau with root entry $\varphi \wedge \psi : 0$ by beginning with the corresponding tableau (t8) and then appending copies of $T_\varphi$ and $T_\psi$ below the entries $\varphi : 0$ and $\psi : 0$ respectively. It is immediate from the definition of a finished tableau that this gives us the desired result.

Next consider (t6) $\varphi \wedge \psi : 1$. As before, we have finished finite tableaux $T_\varphi$ and $T_\psi$. We begin our desired tableau $T$ with the appropriate atomic tableau (t6). To the end of this tableau we add a copy of $T_\varphi$ to get a tableau $T'$ in which the only possible occurence of an entry is that of $\psi : 1$ in the original atomic tableau. We now add a copy of $T_\psi$ to the end of every noncontradictory path in $T'$ to get our desired $T$.

The other cases are treated similarily. □

## 3.2 Soundness and completeness

**Theorem 3.5.** *(Soundness) If $\varphi$ is tableau provable, then $\varphi$ is valid, i.e. $\vdash_t \varphi$ implies $\models_L \varphi$.*

*Proof.* Applying contraposition, suppose $\varphi$ is not valid. By definition there is a truth valuation $V$ assigning 0 or $u$ to $\varphi$. We say that the valuation $V$ agrees with a signed proposition $E$, if

(1) $E$ is $\varphi : 1$ and $V(\varphi) = 1$,

(2) $E$ is $\varphi : u$ and $V(\varphi) = u$, or

(3) $E$ is $\varphi : 0$ and $V(\varphi) = 0$.

We show (Lemma 3.6) that if any valuation $V$ agrees with the root node of a tableau, then there is a path $P$ in the tableau such that $V$ agrees with every entry on $P$. As no valuation can agree with any path on a contradictory tableau, there can be no tableau proof for $\varphi$. □

**Lemma 3.6.** *If $V$ is a valuation that agrees with the root entry of a given tableau $T$, then $T$ has a path $P$ every entry of which agrees with $V$.*

*Proof.* We prove by induction that there is a sequence $(P_n)$ such that, for every $n$, $P_n$ is contained in $P_{n+1}$ and $P_n$ is a path through $T_n$ such that $V$ agrees with every entry on $P_n$. The desired path $P$ through $T$ will then simply be the union of the $P_n$. The initial case of the induction is easily seen to be true by the assumption that $V$ agrees with the root of $T$. It is easy to verify that the atomic tableaux precisely capture the truth tables and value restrictions for the disbelief bar, so that $V$ has to agree with one of the paths the tableau branches into (otherwise it would not be a valid truth valuation).

For the induction step, suppose we have constructed a path $P_n$ in $T_n$ every entry of which agrees with $V$. If we get $T_{n+1}$ from $T_n$ without extending $P_n$, then we let $P_{n+1} = P_n$. If $P_n$ is extended in $T_{n+1}$, then it is extended by adding to its end an atomic tableau with root $E$ for some entry $E$ appearing on $P_n$. As it is known by induction that $V$ agrees with E, the same analysis as used in the initial case shows that $V$ agrees with one ot the extensions of $P_n$ to a path $P_{n+1}$ in $T_{n+1}$. □

**Lemma 3.7.** *Let $P$ be a noncontradictory path of a finished tableau $T$. Define a truth assignment $A$ on all propositional letters $a$ and disbeliefs $\overline{\varphi}$[5] as follows:*

*If $a : 1$ is an entry on $P$ then $A(a) \stackrel{\text{def}}{=} 1$.*

*If $a : u$ is an entry on $P$ then $A(a) \stackrel{\text{def}}{=} u$.*

*If $a : 0$ is an entry on $P$ then $A(a) \stackrel{\text{def}}{=} 0$.*

*If $\overline{\varphi} : 1$ is an entry on $P$ then $A(\overline{\varphi}) \stackrel{\text{def}}{=} 1$.*

*If $\overline{\varphi} : u$ is an entry on $P$ then $A(\overline{\varphi}) \stackrel{\text{def}}{=} u$.*

*If $\overline{\varphi} : 0$ is an entry on $P$ then $A(\overline{\varphi}) \stackrel{\text{def}}{=} 0$.*

*For all other propositional letters and disbeliefs define $A(\cdot) \stackrel{\text{def}}{=} u$.*

[5] As all paths where disbeliefs in equivalent propositions are evaluated differently are closed, this truth assignment will be valid.

*Let $V_A$ be the valuation the truth assignment $A$ gives rise to (Definitions 2.4, 2.6), then $V_A$ agrees with all entries of $P$.*

*Proof.* We proceed by induction on the structure of propositions on $P$.

(i) If $\varphi$ is a propositional letter occuring as (the proposition in) an entry, $V_A$ agrees by definition.

(ii) If $\varphi$ is a disbelief, likewise.

(iii) Suppose $(\varphi \wedge \psi) : 1$ occurs on the noncontradictory path $P$. Since $T$ is finished, both $\varphi : 1$ and $\psi : 1$ occur on $P$. By the induction hypothesis $V_A(\varphi) = V_A(\psi) = 1$, and so $V_A(\varphi \wedge \psi) = 1$ as required.

The remaining connectives and truth values are treated likewise. □

**Theorem 3.8.** *(Completeness) If $\varphi$ is valid, then $\varphi$ is tableau provable, i.e. $\models_L \varphi$ implies $\vdash_t \varphi$. In fact, any contradictory tableau with root entry $\varphi :< 1$ is a proof of $\varphi$.*

*Proof.* Suppose that $\varphi$ is valid and so $V(\varphi) = 1$ for every valuation $V$. Consider any finished tableau $T$ with root $\varphi :< 1$. If $T$ has a noncontradictory path $P$, there would be, by Lemma 3.7, a valuation $V$ that agrees with all its entries and so in particular with $\varphi : u$ or $\varphi : 0$. This would give us a valuation with $V(\varphi) \neq 1$ contradicting the the validity of $\varphi$. Thus every path on $T$ is contradictory and $T$ is a tableau proof of $\varphi$. □

## 3.3 Remarks

The equivalence between two sentences $\varphi$ and $\psi$ can now be checked with a tableau proof for $(\varphi \to \psi) \wedge (\psi \to \varphi)$. For all propositions that do not contain a disbelief bar, condition (2) of Definition 3.2(ii) does not come into play, so the tableau will work just like a classical one. If there are entries containing disbeliefs that are assigned different values, this condition triggers other equivalence proofs with propositions that contain one disbelief bar less. As propositions are finite this procedure will terminate and determine if equivalence holds or not.

As we are in a three-valued logic, there are propositions that are not satisfiable without being contradictions, i.e. they are never evaluated to 1 without always being false. A set of propositions containing one of these does not have a model. We want to be able to make a distinction between nonsatisfiable propositions and contradictions as both play an

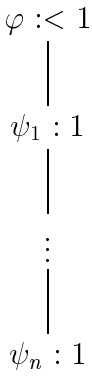



Figure 5: Tableau from premises

important role in matters of equivalence and inference. We therefore introduce a symbol denoting nonsatisfiability: $\perp_w$

Unlike contradictions, nonsatisfiable propositions may behave differently under different truth assignments, i.e. although they never take the value 1, they may take the value $u$ at different times.

**Example 18.** $a \wedge \neg a$ and $a \wedge \overline{a}$ are both nonsatisfiable as can be easily verified, but under the truth assignment $A(a) = u$ $A(\overline{a}) = 0$ the first is evaluated to $u$ but the second to 0.

That is why we cannot write $\varphi \equiv \perp_w$ for a nonsatisfiable proposition $\varphi$. If we did, equivalence would not be transitive any more. So we write $\varphi \vdash_t \perp_w$ instead. Without explicit definition, we assume that this symbol like $\top$ and $\perp$ is element of the language and that it is element of $Cn(\Sigma)$ whenever a nonsatisfiable proposition is element of $Cn(\Sigma)$.

The tableau scheme provides us with a notion of consequence/inference, as well. Figure 5 illustrates the basic idea. Instead of proving $\varphi$ to be true under all truth assignments, it is proved to be true under the assignments that make a set of premises true which captures the definition of $\varphi$ being a consequence of these premises. $\psi_i$ are the propositions occuring in $\Sigma$.

If the tableau in Figure 5 is contradictory, $\Sigma \vdash_t \varphi$ holds, because it means that in any open path of a finished tableau for which $\psi_1 : 1, ..., \psi_n : 1$ hold, $\varphi : 1$ has to hold as well. Soundness and completeness results can be adapted to tableaux with premises.

**Proposition 3.9.** *The deduction theorem*

$\{\varphi\} \models \psi$ *if and only if* $\models \varphi \rightarrow \psi$

*does not hold.*

*Proof.* It suffices to give a counterexample.

Consider $\varphi \wedge (\varphi \rightarrow \psi) \models \psi$. Obviously, this holds. Whenever $\varphi$ and $\varphi \rightarrow \psi$ are evaluated to true, $\psi$ must be true as well. This follows directly from the definition of the implication.

Under the assignment $A(\varphi) = u$, $A(\psi) = 0$, $(\varphi \wedge (\varphi \rightarrow \psi)) \rightarrow \psi$ evaluates to $u$ showing that it is not a tautology. Consequently $\models (\varphi \wedge (\varphi \rightarrow \psi)) \rightarrow \psi$ does not hold. □

**Proposition 3.10.** *If* $\models \varphi \rightarrow \psi$, *then* $\{\varphi\} \models \psi$ *holds.*

*Proof.* Follows from the definition of implication. A necessary condition for $\varphi \rightarrow \psi$ being a tautology is that whenever $\varphi$ takes the truth value 1, so does $\psi$. Which is what $\{\varphi\} \models \psi$ states. □

There is an intuitive explanation why the deduction theorem does not hold. The left hand side and the right hand side express different things. While the left hand side only demands that whenever $\varphi$ is evaluated to 1, so is $\psi$, the right hand side requires that the value of $\psi$ always be greater than or equal to that of $\varphi$. In a two-valued logic these two amount to the same thing, but in a three-valued one, they do not.

Returning to the counterexample used for the proof of Proposition 3.9, the following can be noted. If the conjunction is split up and the propositions are brought to the right side as illustrated in Example 19, the resulting implication holds. This can be easily verified using truth tables.

**Example 19.**

$$\begin{aligned} \{\varphi, (\varphi \rightarrow \psi)\} &\models \psi \\ \{\varphi\} &\models (\varphi \rightarrow \psi) \rightarrow \psi \\ &\models \varphi \rightarrow ((\varphi \rightarrow \psi) \rightarrow \psi) \end{aligned}$$

This principle cannot be used in general, however, as Example 20 illustrates.

**Example 20.**

$$\begin{aligned} \{\varphi \vee \psi, \chi \vee \overline{\psi}\} &\models \varphi \vee \chi \\ \{\chi \vee \overline{\psi}\} &\not\models (\varphi \vee \psi) \rightarrow (\varphi \vee \chi) \end{aligned}$$

The assignment $A(\varphi) = 0, A(\chi) = 0, A\left(\overline{\psi}\right) = 1, A(\psi) = u$ makes the first disjunction true whithout satisfying the implication. So the consequence relation does not hold. Taking $\chi \vee \overline{\psi}$ instead of $\varphi \vee \psi$ to the other side first does not lead to a valid consequence either. The truth assignment leading to the invalidity is the one given above with the values of $\psi$ and $\overline{\psi}$ exchangend.

The essence of the above paragraphs is that we can use tautologies that are made up of implications to derive inference rules, but that this will not give us all necessary rules. A further investigation of the above relationship seems worthwile and will be necessary if an axiomatization for the inference relation is to be found.

It was one of our conjectures that

$$\varphi \vee \psi \equiv \top \text{ if and only if } \varphi \equiv \top \text{ or } \psi \equiv \top,$$

i.e. that combining two nontautologies via disjunction cannot produce a tautology. However, this turned out not to be the case. One possible counterexample is $\varphi \vee \left(\left(\psi \wedge \overline{\psi}\right) \rightarrow \neg\varphi\right)$.

This is a slight modification of the tautology $\varphi \vee \neg\varphi$ in the two-valued case, which of course is not a tautology in our logic as an assignment of $u$ to $\varphi$ reveals. But actually this is the only case for which the disjunction is not evaluated to true. So the idea is to make sure that in this case one of the disjuncts is true without trivially letting it be a tautology. Once more nonsatisfiable propositions play an important role. Using one as the premise in an implication does the trick. As the value of a nonsatisfiable proposition never is 1, the second disjunct will be true not only if $\varphi$ is evaluated to 0 but also for the truth value $u$.

The fact that there are nontrivial disjunctions, i.e. disjunctions that cannot be simplified via $\varphi \vee \top \equiv \top$, which are tautologies indicates that an axiomatization of tautologies might be very hard, at least harder than if the above conjecture had held. It also illustrates the importance of nonsatisfiable propositions which are not contradictions.

Consequence in our three-valued logic is arguing across branches. Consider the following example: Given is the sentence $\neg\varphi \vee \overline{\varphi}$. The (partial) tableau it causes is shown in Figure 6. It does not allow to draw a definite conclusion about $\varphi$ as there are open paths in which $\varphi$ is assigned 0 but also one where it is assigned $u$. But it is obvious that any branch where $\neg\varphi \vee \overline{\varphi} : 1$ and $\varphi : 1$ occur will be closed.

So together with $\varphi \vee \psi : 1$, for example, it would be possible to infer that $\psi$ holds. In

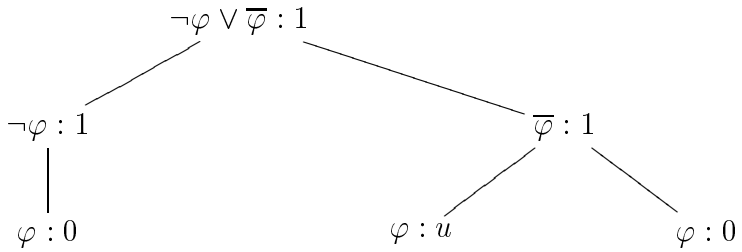



Figure 6: Tableau illustrating inference

the tableau framework, this is easy, but a syntactic axiomatization of inference must be capable of dealing with such cases as well. An inference across the branches in Figure 6 allows the conclusion that $\varphi$ cannot be evaluated to true, i.e. that it is nonsatisfiable. We think that this nonsatisfiability must be syntactically expressible in order to fully capture the inference relation. We present our approach to that in Section 5.

## 3.4 Truthfunctional completeness

One property that is often of interest is whether all truth assignments can be forced by a proposition. This property is known as truthfunctional completeness.

It was our conjecture that our logic is not truth functional. The intuitive explanation is the following. The values true and false do not cause problems, as writing $\varphi$ or $\neg\varphi$ does the job, but in order to force $\varphi$ to take the value $u$, $\overline{\varphi}$ and $\overline{\neg\varphi}$ are needed. Obviously, this would make it impossible to characterize the assignment $A(\varphi) = u$, $A(\overline{\varphi}) = u$.

But there is a way of forcing $\varphi$ to take the value $u$ without having to use disbeliefs – $(\varphi \rightarrow \neg\varphi) \wedge (\neg\varphi \rightarrow \varphi)$. This looks a bit awkward, but it works. The only assignment that satisfies this proposition is $A(\varphi) = u$.

Of course $\varphi$ could be a disbelief as well. This should illustrate that the logic of disbelief allows every truth assignment to be expressed in terms of a proposition.

At this point the question might arise why disbeliefs are necessary at all. Apparently, $a \rightarrow \neg a$ has the same effect as $\overline{a}$. Both prevent $a$ from taking the value 1. So why not encode every disbelief in this type of implication? But again, the problems arise when thinking of disbeliefs in disbeliefs. Using the above scheme $\overline{\overline{a}}$ would be $(a \rightarrow \neg a) \rightarrow \neg(a \rightarrow \neg a)$. This proposition does not have the desired property of being consistent with the evaluation of $a$ to neither 0 nor $u$. This notion of disbelief would imply $\overline{\overline{a}} \equiv a$.

# 4 Implementations of the logic

## 4.1 General notes

We present two attempts to implement the logic and its application in the belief revision framework as Prolog programs[6]. After briefly presenting the first and illustrating why it failed, we will go more into detail with the second.

The language of disbelief is transferred into Prolog via predicates. Valid propositional letters are all strings that satisfy the Prolog predicate *atomic()*. In particular, $a, b, c, \ldots$ can be used as propositional letters for which syntactic and semantic equivalence coincide. That is, two propositional letters are considered identical if and only if they have the same syntactic structure.

Recursively defined, valid propositions P are:

P= a, b, c ... | nott(P) | bar(P) | imp(P,P) | con([P,P$^+$]) | dis([P,P$^+$]

As conjunction and disjunction are associative, we chose not to restrict the corresponding predicates to be binary. Instead, we represented them by a unary predicate that takes a list of a length greater than one as argument.

**Example 21.** The following are examples of how elements of the language of disbelief are translated into the Prolog predicate notation.

- $\neg a$ becomes nott(a)
- $\overline{b}$ becomes bar(b)
- $a \to b$ becomes imp(a,b)
- $a \wedge c$ becomes con([a,c])
- $b \vee a$ becomes dis([b,a])
- $(a \wedge b) \wedge c$ becomes con([a,b,c])
- $(a \wedge (b \to \overline{c})) \vee \overline{\neg a \vee c}$ becomes dis([con([a,imp(b,bar(c))]),bar(dis([nott(a),c]))])

[6]Both programs are written for SWI-Prolog. They do not use any fancy features. One of the things that might cause problems with other Prolog versions is the relation *writeln* which is used for showing results to the user. Its occurences would have to be replaced by a corresponding predicate of the Prolog used.

As in the framework we base our work on, revision is just appending a proposition to the sequence, we did not implement revision itself but the identification of the propositions that make up the belief base. That is, we calculate the consistent subsequence from which the belief set is built. The procedure which does this reflects the definition given in Section 1.2.

**Algorithm 4.1.**

1. Base=$\emptyset$, Seq=Input Sequence
2. El=last element of Seq
3. TempBase=Base $\cup$ El
4. if TempBase consistent then Base=TempBase
5. RemSeq = Seq without the last element
6. if RemSeq empty then goto 7, else Seq = RemSeq and goto 2
7. output TempBase

Starting at the end of the sequence, it is tested whether incorporating the current proposition into the intermediate belief base creates an inconsistency. If so, the proposition is discarded, otherwise it is included into the new intermediate belief base.

Of course, the Prolog version of this algorithm looks somewhat different, but as both approaches differ in how the consistency calculation is done, we leave the details to the respective sections.

## 4.2 First approach

The main idea of the first approach was to label all propositions with a number representing the values the proposition could possibly take. In fact this number represents one of the subsets of the truth values $\{0, u, 1\}$. Using single numbers seemed easier than labelling with sets.

We used the following mapping: 0=$\{\}$=inconsistent, 1=$\{0\}$, 2=$\{u\}$, 3=$\{1\}$, 4=$\{0, u\}$, 5=$\{0, 1\}$, 6=$\{u, 1\}$, 7=$\{0, u, 1\}$.

The numbers are interpreted as value restrictions and these are propagated to the subpropositions. As propositions can occur more than once, all their value restrictions

are then combined. As the numbers represent sets, the resulting label corresponds to the intersection of the respective sets. So for example, the result of combining 4 and 5 leaves 1. Furthermore, information contained in the interaction between different propositions is used. This procedure continues until no further changes in the value restrictions occur.

**Example 22.** We assume that the propositions in the examples are the last ones in the sequence, so we do not have to consider possible interactions with propositions that have already been incorporated in the belief base.

- con([a,b]), i.e. $a \wedge b$, is the last proposition of the sequence. Consequently, it is labelled with 3. As a conjunction can be true only if all its components are true, this value is propagated to $a$ and $b$.

- dis([a,b]), i.e. $a \vee b$, is the last proposition of the sequence. Consequently, it is labelled with 3. As a disjunction can be true if any one of its components is true, no definite information can be derived about $a$ or $b$, so both of them are labelled with 7.

- con([bar(a),dis([b,a])]), i.e. $\overline{a} \wedge (b \vee a)$, is the last proposition of the sequence, so $\overline{a}$ and $b \vee a$ are labelled with 3. From the definition of disbelief it is clear that $a$ can now only take the truth values $u$ and 0, so $a$ is labelled with 4. As above, the value restriction of each component of the disjunction is 7.

  Obviously, we now have two different labels for $a$: 4 and 7. These are combined to a single label 4, which represents that $a$ cannot be evaluated to true any more. Seeing that the disjunction has to be satisfied and one of its disjuncts cannot be satisfied any more, the program knows that the "reduced" disjunction must be satisfied. In this case, that is $b$ which consequently is labelled with 3.

  Now we have two labels for $b$ – 7 and 3 – which are combined to 3. This concludes the calculation.

The investigation of possible interactions between propositions was done using a set of rules. Example 23 illustrates how this works. In the program, a labelled proposition is represented by a list that contains the proposition and its label. The propagation of value restrictions is done on a list of these lists.

**Example 23.**

```
(member([con(X0),1],List1),
 member([Y0,Val0],List1),
 isrestof(con(X0),Y0),
 (Val0=2;Val0=3;Val0=6)),!
                            -> (getrest(c,X0,Y0,Z0),
                                dupup([Z0,1],List1,M1),
                                mtb([con(X0),1],M1,I2));List1=I2)
```

Assume the list contains $a \wedge b \wedge c : 1$ and $c \wedge a : 6$. The first line checks whether there is a conjunction in the list that is evaluated to false. This will identify $a \wedge b \wedge c : 1$. The next three lines try to find a proposition that is a part of this conjunction and that is restricted to 2, 3 or 6. This will identify $c \wedge a : 6$.

We now have a conjunction that is to be evaluated to false and a part of the conjunction that cannot be evaluated to false. This means that the remaining part of the first conjunction must be false. This is realized in the next two lines. *getrest* calculates the rest of the conjunction $a \wedge b \wedge c$ when $c \wedge a$ is removed. The result is $b$. And then the list of value restrictions is updated (*dupup*) with the new information that $b$ must be false.

The next predicate causes the initial conjunction to be moved to the back of the list. This is done because the rule might be applicable to other conjunctions in the list. But in each run, each rule can be applied only once. The cut forces the rule to be applied in the first possible case, so the first conjunction is moved to give the other candidates a chance.

The cut is necessary for the following reason. Assume one rule is applicable only once and assume the next rule is not applicable at all. Since Prolog tries to satisfy as many predicates as possible it will do backtracking and try another version of the first rule, but since there is none and there is no undoing of the backtracking, it will now be assumed that no rule was applicable. In this case, the value restrictions would not be propagated correctly.

From the above explanation, it should become obvious how awkward this procedure really is. The first intention was not to try out every single truth assignment, but rather infer the right value restriction based on a set of rules. But the price for preventing this brute force approach is high. The set of rules that would be necessary is very big and

complicated – and it is hard to be sure that it can handle every possible case correctly.

Example 24 illustrates a family of the problem cases.

**Example 24.** Consider the following entry: $(a \wedge \overline{a}) \vee \left(b \wedge \overline{b}\right)$

The program does not infer that this disjunction cannot be satisfied. In order to do so, it would have to be capable of recognizing the conjunctions to be nonsatisfiable. But as it is not forced to try to label one with 3 – the value propagated to the conjunctions is 7 – it allows the disjunction in the belief base. There are two ways out of this problem: syntactically recognizing nonsatisfiability or trying possible truth evaluations.

The first amounts to an axiomatization of nonsatisfiable propositions which we have not been able to give. The second is in opposition to the intention of this approach to infer rather than to try out.

This is why work on this approach was abandoned. Rather than trying to fix the problems by introducing satisfiability checks to the already quite complicated procedure, it seemed reasonable to implement the tableau approach[7].

## 4.3 Second approach – Tableau

The second program was written after the tableau idea had matured. Kernel of the program is the implementation of the tableau procedures introduced in Section 3. As they cannot deal with general conjunctions and disjunctions, i.e. those with more than two components, they are normalized. In other words, while the user can omit parentheses for the two connectives, the program reintroduces them.

The tableaux themselves are not implemented as trees. They are represented as lists of complete paths. Therefore entries are stored more than once, but checking for clashes within a list seemed easier than searching within a tree structure.

A tableau entry is a list consisting of a proposition, a truth value, and a flag representing whether the entry has been reduced. A path is a list of entries and a tableau a list of paths. There are three special entries – the strings *fin*, *open*, and *closed*. They are used as tags for paths, as to speed up checks, if a path is finished, open, or closed. That

[7] In fact, when work on the first implementation was started, the tableau idea had not been developed yet. By the time the problems became pressing, work on the tableau definition presented in Section 3 had begun and was quite promising. So a reimplementation seemed more favourable.

is, once marked as closed, entries on the path are not reduced any more, as this would not help in creating a consistent model.

When reducing an entry, the respective path is temporarily removed from the tableau. The atomic tableau that belongs to the entry is identified. If it branches, the path is duplicated and one of the nodes is inserted into one copy. Finally, the entry is marked as reduced and the new paths are reinserted into the tableau.

In each run, one entry of every path in the tableau is reduced. The program then tests whether paths are closed and removes them from the tableau. Finished open paths are also marked. In some cases, calculation is terminated when an open finished path is found.

The tableau kernel provides some basic predicates for the user:

- $prove(P)$ returns "yes" if $P$ is a tautology, "no" otherwise
- $satisfiable(P)$ returns "yes" if $P$ is satisfiable, "no" otherwise
- $notsatisfiable(P)$ returns "yes" if $P$ is nonsatisfiable, "no" otherwise
- $eq(P,Q)$ returns "yes" if $P$ and $Q$ are equivalent, "no" otherwise
- $isconsequenceof(P,[Q^*])$ returns "yes" if $P$ is a consequence of the set of propositions

The first three can have the string *show* as second argument. In this case, an open finished path is returned as well. Since a tableau proof for a proposition is a closed tableau, the open path then gives a model that causes the proposition not to be evaluated to true. In the case of satisfiability a model of the proposition is produced. In the case of nonsatisfiability, a model is produced as well – proving that there is one.

**Algorithm 4.2.**

```
prove(X):-        prove(X,A,noshow).
prove(X,show):-   prove(X,A,show).
prove(X,A,S):-    normalizelist([[X,1]],Y),
                  maketab(Y,Z),
                  finishopenfail(Z,A),!,
                  (S=show -> write\_ln(A);true),
                  closed(A).
```

All the above predicates are similar in appearance to the code shown in Algorithm 4.2. The proof procedure starts by normalizing the proposition. The main task here is to transform all disjunctions and conjunctions to have exactly two components. The resulting list is then turned into an initial tableau. The predicate *finishopenfail* reduces entries until one open finished path is found or all paths are closed. The cut is necessary to prevent backtracking. The last line then checks whether the resulting tableau was closed or contained an open path.

Recall Definition 3.2 of a closed path. The test whether there is a proposition that is assigned two different values is easily realized with a membership check. But to check the other possibility – that there are two disbeliefs in equivalent propositions with different values – is a little more complicated.

Equivalence checks are expensive because they require a tableau proof for mutual implication. If there are many disbeliefs in the propositions and if the complete equivalence check was triggered after every reduction of an entry in the tableau, a lot of time would be wasted on what was known before. That is why equivalence and nonequivalence results are stored in dynamic predicates that could be interpreted as a database. And before the tableau proof of equivalence is started, this database is searched, to see whether the check was run before. This causes a considerable speedup.

**Algorithm 4.3.**

```
equivalent(X,X).
eq(X,Y):-  equivalent(X,Y),!.
eq(X,Y):-  (notequivalent(X,Y),!)->
               fail;
               ((prove(con(imp(X,Y),imp(Y,X))),!) ->
                     (asserta(equivalent(X,Y)),
                      asserta(equivalent(Y,X)),
                      true);
                     (asserta(notequivalent(X,Y)),
                      asserta(notequivalent(Y,X)),
                      fail)
               ).
```

### Notes on the revision part of the tableau implementation

In this implementation, there is a clearer separation between the identification of the belief base and the calculation of the (partial) belief set.

**Algorithm 4.4.** [8]

```
dorevision(X,R):-belbase(X,R),
                 createbeliefset(R,S).

belbase(X,Y):-validlist(X),calculate(X,[],Y).

calculate([], Intermediate, Intermediate).
calculate([H|Rest], Intermediate,Y):-
                          calculate(Rest, Intermediate, Finalresult),
                          (satisfiablelist([H|Finalresult])->
                                      Y=[H|Finalresult];
                                      Y=Finalresult).
```

*dorevision* gets a list of propositions, identifies the belief base and creates the belief set. At first we are going to look at how the belief base is identified. This is essentially managed by the predicate *calculate*.

Its first argument is the initial sequence of propositions, the second is the intermediate belief base, the last accumulates the result. *calculate([], Intermediate, Intermediate).* defines that in case the sequence is empty, the resulting belief base is equivalent to the intermediate one.

The first line in the definition for the nontrivial case causes a recursive call of *calculate*. In every recursive step, the first element of the sequence is cut off, so in the end the list is empty and the initial case is triggered. Afterwards, one proposition after the other is appended to the intermediate belief base and it is tested if this list is satisfiable. If so, the proposition is inserted into the intermediate belief base, otherwise the latter is left unchanged. So in the end there is a consistent sequence of propositions.

*satisfiablelist* is based on the predicate *satisfiable*. In fact the former creates the conjunction of all the propositions in the list and then triggers the latter.

[8] The source code is slightly modified in order to increase readability

The calculation of the belief set is more interesting than the identification of the belief base – which should be quite clear after the last section. In fact, only a partial belief set is calculated as most of the proposition in the infinite set do not provide information.

If $\neg a$ is in the belief set, so is $a \rightarrow \varphi$ for any $\varphi$. But knowing this does not help with anything. It is our notion that the aim is to derive something about the subpropositions contained in the belief base. If $a \vee b$ is in the belief base, information about which of the two is true is far more valuable than the fact that any disjunction containing this one is still true.

So the naive inference engine we implemented does nothing except expanding the propositions in the belief base to all their subpropositions and checking whether these are consequences of the base. As the belief base does not change during this procedure, the finished tableau of the belief base is calculated and reused for the individual checks, rather than recalculating the entire tableau.

That is, not the predicate *isconsequenceof*, which has a list of propositions as argument, is used, but a different one that takes a tableau as argument, adds the proposition to be tested to every nonclosed branch and then continues the reduction of entries.

The very naive way of just considering (all) the subpropositions has some shortcomings. On the one hand some propositions may occur more than once – not necessarily in the same syntactic structure. So some checks will be done twice ore even more. Checking the entire set of subpropositions for duplicates will take much time, as equivalence checks are involved. This is one point where performance could still be improved.

Disjunctions are another issue. Long disjunctions create many subdisjunctions – two to the power of the number of disjuncts, to be exact. If one of the short subdisjunctions turns out to be a consequence of the belief base, all of its longer versions will be in the belief base, too. But as mentioned above $a \vee b$ is not interesting if we already know $a$. This could be solved by sorting the subpropositions by length, i.e. by the complexity of their structure, and test for each disjunction whether one of its disjuncts is already in the belief base.

One interesting case where the inference "fails" is general resolution. If the propositions $a \vee b$ and $c \vee \neg b$ are given, $a \vee c$ is not identified as a consequence, simply because it is not among the subpropositions of the two and therefore not tested. When it is asked explicitly if $a \vee c$ is a consequence, the answer is of course affirmative. A naive way to overcome this problem is to create the big disjunction of all the disjunctions in the set of

base beliefs and consider its subpropositions, which then adds to the problems mentioned above.

A number of examples illustrating aspects of the usage of the tableau implementation are found in Appendix C.

Comparing the implementations in size reveals that the (kernel) tableau version is smaller by far than the other one, which is mainly due to the fact that many inference rules are necessary to find out what the tableau reveals by just trying out all the possibilities, while they still miss to identify certain nonsatisfiable propositions. The difference in size is decreased by the part that does the inference.

# 5 Towards an axiomatization

## 5.1 Equivalence relation $\equiv_L$

$\top$ denotes the tautology, i.e. a proposition that is evaluated to 1 by every truth valuation, $\bot$ the contradiction, which is always evaluated to 0. As we are in a three-valued logic, though, there are propositions that are never evaluated to 1 without being contradictions. However, this property cannot be used for defining equivalence.

The equivalence relation $\equiv_L$ for the language of disbelief has many similarities to the classical one $\equiv$. A major difference is the set of tautologies. $\equiv_L$ is a symmetric, transitive and reflexive relation that satisfies the following properties. The rules for implication are necessary as it is not definable in terms of disjunction and negation.

$\varphi \equiv_L \neg\neg\varphi$

$\neg\varphi \equiv_L \neg\psi$ if and only if $\varphi \equiv_L \psi$

$\varphi_1 \equiv_L \varphi_2$ and $\psi_1 \equiv_L \psi_2$ implies $\varphi_1 \to \psi_1 \equiv_L \varphi_2 \to \psi_2$

$\varphi \to \psi \equiv_L \neg\psi \to \neg\varphi$

$\varphi \vee \top \equiv_L \top$

$\varphi \wedge \top \equiv_L \varphi$

$\varphi \vee \bot \equiv_L \varphi$

$\varphi \wedge \bot \equiv_L \bot$

$\varphi \to \top \equiv_L \top$

$\top \to \varphi \equiv_L \varphi$

$\bot \to \varphi \equiv_L \top$

$\varphi \to \bot \equiv_L \neg\varphi$

$\neg\top \equiv_L \bot$

Furthermore, it satisfies idempotence, commutativity, associativity and distributivity of conjunction and disjunction as well as the validity of the de Morgan laws. The above equivalences directly follow from the properties of the classical connectives given in Section 2.4.

The case when the disbelief bar is involved as the main connective is more interesting. The fundamental property that should hold in our logic is the following:

$\overline{\varphi} \equiv_L \overline{\psi}$ if and only if $\varphi \equiv_L \psi$

The essential definition for this property is Definition 2.6 which characterizes valid truth assignments. The right to left direction directly follows from the definition, as it

states that disbeliefs in equivalent propositions have to behave alike, i.e. be equivalent.

The left to right direction is also ensured. To see that, let us examine the identification of a valid truth assignmnent which could be interpreted as an iterative construction – iterative in the sense that we layer the propositions, starting with no disbelief bar at all and then adding one at a time, after it is clear what happens with the propositions of the previous layer. Formally, the language of disbelief could be constructed the following way.

**Definition 5.1.** Let $CL\,(\Sigma)$ be the (syntactic) closure of $\Sigma$ under the classical connectives.

$L'_0 = \{\text{propositional letters}\}$, $L_0 = CL\,(L'_0)$

$L'_i = \{\overline{\varphi} | \varphi \in L_{i-1}\}$

$L_i = CL\,(L'_i \cup L_{i-1})$

Obviously, the language of disbelief $L = L_\infty$. A truth assignment assigns values to the elements of $\bigcup L'_i$ in such a way that the evaluation of the elements of $L$ is consistent with Definition 2.6. We will now investigate a bit more in detail how valid truth assignments develop.

We start with assigning values to the elements of $L'_0$, i.e. to the propositional letters. It should be clear that any assignment of truth values $v \in \{0, u, 1\}$ to the propositional letters is valid. Of course these are not complete truth assignments as the values for disbeliefs are not yet considered, but they could be completed by assigning $u$ to all the disbeliefs, i.e. to the $\varphi \in L'_i, i > 0$. Obviously, all of the assignments are valid.

**Definition 5.2.** $A'_{L'_i}$ is a function assigning truth values $v \in \{0, u, 1\}$ to the elements of $L'_i$.

A truth assignment $A_{L'_i} = \bigcup A'_{L'_j}, j \leq i$. That is, it is the collection of all the assignments up to the $i$th layer.

**Remark 25.** $A_{L'_0} = A'_{L'_0}$

The truth assignments $A_{L'_i}$ naturally extend to $L_i$. Moreover, the value of an element of $L_i$ is uniquely determined by $A_{L'_i}$. That is, an assignment $A'_{L'_{i+1}}$ of a value to a disbelief of a higher layer has no influence on the *value*, but only on the question whether the combined assignment $A_{L'_i} \cup A'_{L'_{i+1}}$ is *valid*. This is an essential point and we want to illustrate it with a simple example.

**Example 26.** Let us restrict the language to two propositional letters $a$ and $b$. So $L'_0 = \{a, b\}$, $L_0 = \{a, b, a \vee b, a \wedge b, \dots\}$,
$L'_1 = \{\overline{a}, \overline{b}, \overline{a \vee b}, \overline{a \wedge b}, \dots\}$, $L_1 = \{a, b, b \vee \overline{a}, a \to \overline{b}, \overline{a \vee b} \wedge \overline{a \wedge b}, \dots\}$.

Now an assignment $A_{L'_0}(a) = v_a$, $A_{L'_0}(b) = v_b$ uniquely determines the values of all the elements of $L_0$, i.e. from the value of the propositional letters we can calculate the value of their combinations via the classical connectives. For example $A_{L'_0}(a) = 1$ allows the conclusion that the under that assignment the value of $a \vee b$ is 1, too. If we additionally have $A'_{L'_1}(\overline{a}) = 1$, neiter the value of $a$ nor that of $a \vee b$ would be different *but* as this combination is in conflict with the value restrictions of Table 3 $A_{L'_1} = A_{L'_0} \cup A'_{L'_1}$ is an *invalid* truth assignment.

The assignments $A_{L'_0}$ allow us to determine which propositions, constructed of propositional letters and the classical connectives only, are equivalent. We only have to check which of them behave alike under all the assignments $A_{L'_0}$. As the example illustrates, possible assignments to disbeliefs have no influence on that.

In other words, even if we expand the truth assignment giving values to disbeliefs, previously equivalent propositions stay equivalent and propositions that were not equivalent cannot suddenly become so, because at least one assignment remains under which the propositions behave differently – the one where all disbeliefs (of the higher layers) are assigned $u$.

Continuing this line of thought provides a way to construct all valid truth assignments. As mentioned above, all assignments $A_{L'_0}$ are valid. Next, we create all assignments $A'_{L'_1}$, i.e. all combinations of assignments of the truth values to all elements in $L'_1$. As we have an initial set of equivalent propositions in $L_0$, we can throw away those assignments that give different values to equivalent propositions.

Now we can combine every single valid truth assignment $A_{L'_0}$ with every single assignment of the the remaining $A'_{L'_1}$. Of these combinations, we throw away those which do not meet the value restrictions of Table 3. This is easily checked, as the $A_{L'_0}$ component provides us with the value of the propositions in question. The remaining combinations are the valid $A_{L'_1}$.

The next step is to expand the equivalence relation to $L_1$. This is easily done by checking which elements of $L_1$ behave alike under all valid assignments $A_{L'_1}$. And so on...

The following observations are essential:

- Two propositions which were not equivalent on a previous layer cannot become so after introducing a new layer. This is immediate, because we already have at least one valid truth assignment for the previous layer under which the propositions behave differently. Of course, the assumption is that both propositions were already defined on the previous layer, i.e. that they were elements of the fragment of the language.

- Two propositions which were equivalent on a previous layer cannot become inequivalent after introducing a new layer. This is because the value of the disbelief has no influence on the value of the proposition. As illustrated above, it can only cause a potential combined assignment to be invalid.

- A disbelief $\varphi$ introduced at layer $i$ cannot be equivalent with any element $\psi$ of the language fragment $L_{i-1}$. To see that, take an arbitrary valid truth assignment for the level $i-1$. This uniquely defines the value for any element in $L_{i-1}$, in particular the one for $\psi$. By the value restrictions in Table 3, $\varphi$ can take at least two different truth values, i.e. there is at least one valid truth assignment for which the values of $\varphi$ and $\psi$ differ.

Coming back to our goal of showing that $\overline{\varphi} \equiv_L \overline{\psi}$ implies $\varphi \equiv_L \psi$ holds as well, we can conclude from the last observation that $\overline{\varphi}$ and $\overline{\psi}$ must have been introduced at the same layer, i.e. that they are element of the same $L'_i$. We now assume the two disbeliefs to be equivalent but $\varphi$ and $\psi$ not to be equivalent. The latter means that there is a valid truth assignment $A_{L_{i-1}}$ that assigns different truth values to the two. But by the value restrictions of Table 3 for two different truth values, there are differing sets of truth values that the disbeliefs could take. So there are valid truth assignments that assign different truth values to the disbeliefs, contradicting the assumption.

Whether these properties suffice to characterize equivalence remains to be investigated.

**Proposition 5.3.** *There is no $\overline{\varphi}$ such that $\overline{\varphi} \equiv \top$.*

*Proof.* By the definition of the value restrictions in Table 3, independent of the value of $\varphi$, $\overline{\varphi}$ can take the value $u$, i.e. there is a truth assignment evaluating the disbelief to $u$. Consequently, $\overline{\varphi}$ cannot be a tautology. □

**Proposition 5.4.** *There is no $\overline{\varphi}$ such that $\overline{\varphi} \equiv \bot$.*

*Proof.* By the definition of the value restrictions in Table 3, independent of the value of $\varphi$, $\overline{\varphi}$ can take the value $u$, i.e. there is a truth assignment evaluating the disbelief to $u$. Consequently, $\overline{\varphi}$ cannot be a contradiction. □

## 5.2 A metalogic for inference

In Section 3.3 we gave a short explanation why the language of disbelief may not be rich enough to allow to write down all the inference rules necessary to capture all valid inferences. Again, nonsatisfiability played an important rule. In two-valued logic, nonsatisfiability is equivalent to contradiction, which can be captured syntactically by equivalence definitions using the symbol $\bot$.

Contradictory propositions are defined equivalent to this symbol. Then the equivalence definitions can be used for simplifications and inference. This can in fact be transferred to our logic, but just adding a symbol $\bot_w$ is not enough to solve our problem with nonsatisfiable propositions. They are not necessarily equivalent and may behave differently given a truth assignment, as Example 27 illustrates.

**Example 27.** $a \wedge \neg a$, $b \wedge \neg b, A(a) = 0, A(b) = u$

Clearly, both conjunctions are nonsatisfiable and have different truth values given the assignment. The first conjunction evaluates to 0, the second to $u$.

But it is not the pure nonsatisfiability alone that causes the problems. Satisfiable propositions which cause other propositions not to be able to take the value true or false are just as important. The last example in Section 3.3 illustrates just that. There, we considered the proposition $\neg\varphi \vee \overline{\varphi}$. This case may seem trivial, but there are examples where neither $\neg\varphi$ nor $\overline{\varphi}$ need to be necessarily true to imply the nonsatisfiability of $\varphi$. To see this consider $\varphi \rightarrow \overline{\varphi}$.

Having this example in mind, it becomes clear that there is no upper bound to the number of propositions that cause others not to be satisfiable. The main idea is illustrated in Example 28.

**Example 28.** $a \rightarrow b_1, b_1 \rightarrow b_2, \ldots, b_n \rightarrow \overline{a}$

Resolution-like inferences demand the capability of expressing the nonsatisfiability of a proposition – be it because of the structure of the proposition itself or because a set of

propositions implies it. Assume $a \vee b$ is in the belief set. Knowing $a$ to be nonsatisfiable clearly allows the inference of $b$, but, as illustrated by Example 28, there is an infinite number of sets of propositions that cause $a$ to be nonsatisfiable.

It is not desirable to have a rule for every one of these sets. It would be more convenient to have a rule like: If $a \vee b$ and $a$ is nonsatisfiable then $b$. Rules for $\neg a$ and $\overline{a}$ are just not enough, as Example 29 shows. Even though the two propositions are inconsistent with the set presented, $a$ is nonsatisfiable.

**Example 29.** $\left\{\varphi, a \rightarrow \overline{\varphi}, \overline{\overline{a}}, \overline{\neg a}\right\}$

That is why we think a further extension of the language is necessary, or at least helpful, to infer propositions that the tableau scheme would allow successfully. This extension, however, will only be used by the inference mechanism and is not available to the "user" of this logic. So it could be seen as a metalogic for inference. The extended language will contain two further symbols + and - to denote that a proposition cannot be evaluated to 1 and 0 respectively. Inference is then done on this extended language and afterwards all nonstandard sentences are removed. More formally:

**Definition 5.5.** Let $L$ be the language of disbelief as in Definition 2.1, $L' \stackrel{\text{def}}{=} \{x^+ | x \in L\} \cup \{x^- | x \in L\}$.

$L_\pm$is the language containing $L$ and $L'$ and being closed under negation, conjunction, disjunction and implication.

**Remark 30.** Expressions of the form $x^+$or $x^-$, $x$ cannot contain a further + or $-$.

Truth assignment and truth valuation for $L$ can be extended to $L_\pm$ in a straightforward way. The only additional rules are

$$V_A\left(\varphi^-\right) \stackrel{\text{def}}{=} \begin{cases} 0, V_A(\varphi) = 1 \\ 1, otherwise \end{cases}$$

and

$$V_A\left(\varphi^+\right) \stackrel{\text{def}}{=} \begin{cases} 0, V_A(\varphi) = 0 \\ 1, otherwise \end{cases}$$

which corresponds to the intuition behind the symbols.

**Definition 5.6.** Let $\vdash_\pm$be an inference relation on $L_\pm$.

$\Sigma \vdash_L \varphi$ if and only if $\varphi \in Cn_\pm(\Sigma) \cap L$, where $Cn_\pm(\Sigma) = \{\psi | \Sigma \vdash_\pm \psi\}$[9].

[9]We will simplify notation, writing $\varphi \vdash \psi$ instead of $\{\varphi\} \vdash \psi$.

In fact we are only interested in the inference relation on our language of disbelief, but we have problems to formulate rules restricted to that language, so what Definition 5.6 expresses is the following. We do the inference in the extended language $L_{\pm}$ and then take the intersection with our language of disbelief.

Our task is now to characterize $\vdash_{\pm}$such that $\vdash_L = \vdash_t$, in order to have an entailment relation that is not dependent on the tableau. The idea is to do so by mutual simulation, i.e. showing that any tableau rule can be expressed in terms of a set of inferences on $L_{\pm}$ and that any inference on $L_{\pm}$ can be expressed by a tableau. Inference on $L_{\pm}$ can then be interpreted as linearisation of argumentation across open branches in the tableau.

We have not fully solved this task, but wewill give some intermediate results.

## 5.3 Nonsatisfiability

If a proposition $\varphi$ cannot be satisfied, ($\varphi \vdash_{\pm} \bot_w$), $\varphi^-$ should follow from the empty set, i.e. $\vdash_{\pm} \varphi^-$ should hold, as $\varphi^-$is true whatever the evaluation of $\varphi$.

This is a natural demand. The above $\varphi^-$ could be interpreted as a new type of tautologies that occurs in the extended language. These tautologies need to be axiomatized, as well.

Some immediate properties are:

- $\vdash_{\pm} \varphi^-$ for all contradictions $\varphi$
- $\vdash_{\pm} \overline{\varphi}^-$ for all tautologies $\varphi$
- $\vdash_{\pm} (\varphi \wedge \neg\varphi)^-$ for all $\varphi$
- $\vdash_{\pm} (\varphi \wedge \overline{\varphi})^-$ for all $\varphi$
- $\vdash_{\pm} (\varphi \wedge \neg\overline{\neg\varphi})^-$ for all $\varphi$

It is our conjecture that these are the basic cases to which all nonsatisfiable propositions can be reduced, but neither do we have a proof for it nor a counterexample. We were not able to find an axiomatic or constructive characterization of (all) nonsatisfiable propositions, but we will give some of their properties.

**Proposition 5.7.** *$\varphi \vee \psi \vdash_{\pm} \bot_w$ if and only if $\varphi \vdash_{\pm} \bot_w$ and $\psi \vdash_{\pm} \bot_w$.*

*Proof.* ($\Rightarrow$) Assume $\varphi \vee \psi$ is nonsatisfiable, but either $\varphi$ or $\psi$ is satisfiable, i.e. there is a truth assignment under which $\varphi$ or $\psi$ are evaluated to true. By the definition of disjunction, this would give us a model for $\varphi \vee \psi$ contradicting the assumption.

($\Leftarrow$) Assume both $\varphi$ and $\psi$ are nonsatisfiable but $\varphi \vee \psi$ has a model. That is, there is a truth assignment under which the disjunction evaluates to true. By the definition of disjunction, this assignment must be a model for $\varphi$ or $\psi$ contradicting the assumption. □

Proposition 5.7 suggests that some kind of disjunctive normal form would be helpful for characterizing nonsatisfiable propositions, as then only the disjuncts have to be tested for nonsatisfiability.

**Proposition 5.8.** *$\varphi \vdash_{\pm} \bot_w$ implies $\varphi \wedge \psi \vdash_{\pm} \bot_w$ for any $\psi$.*

*Proof.* Assume $\varphi$ is nonsatisfiable but $\varphi \wedge \psi$ is satisfiable. That is, there is a truth assignment that evaluates the conjunction to true. By the definition of conjunction, this is only possible if both conjuncts are evaluated to true which contradicts the assumption. □

**Proposition 5.9.** *If $\varphi$ and $\psi$ do not share propositional letters, then*

$$\varphi \wedge \psi \vdash_{\pm} \bot_w \text{ if and only if } \varphi \vdash_{\pm} \bot_w \text{ or } \psi \vdash_{\pm} \bot_w.$$

*Proof.* ($\Leftarrow$) follows immediately from Proposition 5.8.

($\Rightarrow$) Assume $\varphi \wedge \psi \vdash_{\pm} \bot_w$, but $\varphi \nvdash_{\pm} \bot_w$ and $\psi \nvdash_{\pm} \bot_w$, i.e. $\varphi$ and $\psi$ are satisfiable. Therefore the finished tableau with the root entry $\varphi : 1$ has an open path $P_1$, and the one for $\psi : 1$ an open path $P_2$. If these two paths are appended, the resulting path cannot contain a contradiction, as $\varphi$ and $\psi$ do not share variables, so the tableau with the root entry $\varphi \wedge \psi : 1$ has an open path as well, which contradicts our assumption that $\varphi \wedge \psi$ is nonsatisfiable. □

It follows that the only possibility for $\varphi \wedge \psi$ to be nonsatisfiable in spite of both $\varphi$ and $\psi$ being satisfiable is for the two conjuncts to share variables, i.e. to be in some way related. This relationship is to be investigated further.

**Proposition 5.10.** *$\varphi \to \psi \vdash_{\pm} \bot_w$ implies $\neg\varphi \vdash_{\pm} \bot_w$.*

*Proof.* Assume $\varphi \to \psi$ to be nonsatisfiable and $\neg\varphi$ to be satisfiable. The latter means that there is a truth assignment evaluating $\neg\varphi$ to true. Consequently, the same assignment evaluates $\varphi$ to false. By the definition of implication, this assignment evaluates $\varphi \to \psi$ to true contradicting the assumption. □

**Proposition 5.11.** $\varphi \to \psi \vdash_{\pm} \bot_w$ *implies* $\psi \vdash_{\pm} \bot_w$.

*Proof.* Assume $\varphi \to \psi$ to be nonsatisfiable and $\psi$ to be satisfiable, that is, there is a truth assignment evaluating $\psi$ to true. By the definition of implication, this truth assignment evaluates $\varphi \to \psi$ to true, as well. This contradicts the assumption. □

The simplest cases would be those where in addition to these properties $\varphi$ is a tautology or $\psi$ is a contradiction (or both). Then, however, the implication could be simplified to either $\varphi$ or $\psi$ using the equivalence relation. In the more interesting case, the condition is that whenever $\varphi$ is evaluated to $u$, $\psi$ has to be evaluated to 0. Once more, the two are not independent of each other.

**Proposition 5.12.** $\overline{\varphi} \vdash_{\pm} \bot_w$ *if and only if* $\varphi \equiv_L \top$.

*Proof.* ($\Rightarrow$) Assume $\overline{\varphi}$ to be nonsatisfiable and $\varphi$ not to be a tautology. That is, there is a truth assignment which evaluates $\varphi$ to $u$ or 0. By the value restrictions of Table 3, this would allow an assignment of 1 to $\overline{\varphi}$.

($\Leftarrow$) Assume $\varphi$ to be a tautology and $\overline{\varphi}$ to be satisfiable. That is, there is a truth assignment that evaluates the disbelief to true. By the value restrictions of Table 3, this assignment cannot evaluate $\varphi$ to true, which contradicts the assumption. □

**Proposition 5.13.** $\varphi \to \overline{\psi} \vdash_{\pm} \bot_w$ *if and only if* $\varphi \equiv_L \top$ *and* $\psi \equiv_L \top$.

*Proof.* ($\Rightarrow$) Assume $\varphi \to \overline{\psi}$ to be nonsatisfiable and $\varphi \not\equiv_L \top$ or $\psi \not\equiv_L \top$.

- $\varphi \not\equiv_L \top$, i.e. there is at least one truth assignment that evaluates the proposition to $u$ or 0. By the value restrictions of Table 3, one of these truth assignments evaluates $\overline{\psi}$ to $u$, as well. By the definition of implication, this assignment evaluates $\varphi \to \overline{\psi}$ to true, contradicting the assumption.

- $\psi \not\equiv_L \top$, i.e. there is a truth assignment evaluating $\psi$ to 0 or $u$. By the value restrictions of Table 3, there is a truth assignment evaluating $\overline{\psi}$ to true, contradicting the assumption, as this assignment satisfies $\varphi \to \overline{\psi}$.

($\Leftarrow$) Assume $\varphi \equiv_L \top$ and $\psi \equiv_L \top$. $\varphi \to \overline{\psi}$ be satisfiable, i.e. there is a truth assignment evaluating the proposition to true. By the definition of implication, $\varphi \to \overline{\psi}$ can be true only if one of the following holds:

- The truth assignments evaluates $\varphi$ to false, but this contradicts the assumption of $\varphi$ being a tautology.
- The truth assignment evaluates $\overline{\psi}$ to true, but this contradicts the assumption of $\psi$ being a tautology (see Proposition 5.12).
- The truth assignment evaluates both $\varphi$ and $\overline{\psi}$ to $u$, but this contradicts the assumption of $\varphi$ being a tautology. □

**Proposition 5.14.** $\overline{\varphi} \to \psi \vdash_{\pm} \bot_w$ *if and only if* $\varphi \equiv_L \bot$ *and* $\psi \equiv_L \bot$.

*Proof.* ($\Rightarrow$) Assume $\overline{\varphi} \to \psi$ to be nonsatisfiable and $\varphi \not\equiv_L \bot$ or $\psi \not\equiv_L \bot$.

- $\varphi \not\equiv_L \bot$, i.e. there is at least one truth assignment evaluating $\varphi$ to $u$ or 1. By the value restrictions of Table 3, there is one evaluating $\overline{\varphi}$ to 0. By the definition of implication, this will evaluate $\overline{\varphi} \to \psi$ to 1, contradicting the assumption.
- $\psi \not\equiv_L \bot$, i.e. there is a truth assignment evaluating $\psi$ to $u$ or 1. The latter would already give us a model for $\overline{\varphi} \to \psi$ which would contradict the assumption. So only the case of an evaluation of $\psi$ to $u$ remains to be considered. By the value restrictions of Table 3, irrespective of the value of $\varphi$, $\overline{\varphi}$ can be evaluated to $u$, i.e. there is a truth assignment doing so. It should be clear that a corresponding truth assignment can be constructed.

($\Leftarrow$) Assume $\varphi \equiv_L \bot$ and $\psi \equiv_L \bot$. $\overline{\varphi} \to \psi$ be satisfiable, i.e. there is a truth assignment evaluating the proposition to true. By the definition of implication, $\overline{\varphi} \to \psi$ can be true only if

- the truth assignment evaluates $\overline{\varphi}$ to false. By the value restrictions of Table 3, the truth assignment then must evaluate $\varphi$ to $u$ or 1, contradicting the assumption of $\varphi$ being a contradiction.
- the truth assignment evaluates $\psi$ to true, contradicting the assumption of $\psi$ being a contradiction.
- the truth assignment evaluates both $\overline{\varphi}$ and $\psi$ to $u$, contradicting the assumption of $\psi$ being a contradiction. □

These properties inspire some of the inference rules given in Section 5.5. Example 31 shows some nonsatisfiable propositions.

**Example 31.**

- $\varphi \wedge \overline{\varphi}$
- $\varphi \wedge \psi \wedge (\psi \rightarrow \neg\varphi)$
- $\overline{\top}$
- $\overline{\varphi} \wedge \overline{\psi} \wedge (\varphi \vee \psi)$

## 5.4 Tautologies

Many of the tautologies we know from classical logic still hold in our logic of disbelief. This was already implied in Section 5.1. There are some notable exceptions, however.

- $((\varphi \wedge \psi) \rightarrow \chi) \rightarrow (\varphi \rightarrow (\psi \rightarrow \chi)) \equiv_L \top$ still holds, but the reverse direction $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \wedge \psi) \rightarrow \chi) \not\equiv_L \top$, which can be easily seen by the valuation $V(\varphi) = V(\psi) = u$, $V(\chi) = 0$.
- $((\varphi \rightarrow \psi) \rightarrow \chi) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi)) \not\equiv_L \top$ using the same valuation.
- $(\neg\varphi \vee \psi) \rightarrow (\varphi \rightarrow \psi) \equiv_L \top$ still holds, but the reverse direction $(\varphi \rightarrow \psi) \rightarrow (\neg\varphi \vee \psi) \not\equiv_L \top$, also evaluating $\varphi$ and $\psi$ to $u$. This example illustrates once more that implication and disjunction are not interdefinable.

As remarked in Section 3.3, nonsatisfiable propositions also play an important role. All this means that an axiomatization of the tautologies in our logic will not simply be an extension of the axioms used for classical logic.

## 5.5 Inference rules

In the following, we want to present some of the rules for the inference relation $\vdash_\pm$. In fact it is a relation between a set of propositions and a proposition, $\Sigma \vdash_\pm \varphi$. In order to simplify notation in case $\Sigma$ contains only one element, we write $\varphi \vdash_\pm \psi$ instead of $\{\varphi\} \vdash_\pm \psi$.

Naturally, $\vdash_\pm \varphi$ for all tautologies $\varphi$. Further, we assume that all nonsatisfiable propositions $\varphi$ are given and therefore $\vdash_\pm \varphi^-$ holds. Consequently, for all propositions $\psi$ that cannot be evaluated to false, $\vdash_\pm \psi^+$ holds. In the following, we will give rules of

| $\varphi$ | $\neg\varphi$ | $\varphi^+$ | $(\neg\varphi)^-$ | $\varphi^-$ | $(\neg\varphi)^+$ |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 |
| $u$ | $u$ | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |

Table 4: Truth table for the language extension

$\vdash_\pm$and show that they correspond to the tableau introduced. $\varphi$ and $\psi$ are propositions of $L$, i.e. they do not contain further $^+$ or $^-$.

$\varphi \vdash_\pm \varphi$

$\varphi \wedge \varphi^- \vdash_\pm \bot_w$

$(\neg\varphi)^+ \vdash_\pm \varphi^-$and $(\neg\varphi)^- \vdash_\pm \varphi^+$, so the two are equivalent. Table 4 illustrates this equivalence, as well.

$\neg(\varphi^-) \vdash_\pm \varphi$ and $\neg(\varphi^+) \vdash_\pm \neg\varphi$

$\varphi^- \vdash_\pm (\varphi \wedge \psi)^-$

$\varphi \vee \psi \vdash_\pm \varphi^- \to \psi$

$\varphi \vee \psi \vdash_\pm \psi^- \to \varphi$

$\varphi^- \to \psi \vdash_\pm \varphi \vee \psi$

$\psi^- \to \varphi \vdash_\pm \varphi \vee \psi$

$(\varphi \vee \psi)^+ \wedge \neg\varphi \vdash_\pm \psi^+$

$(\varphi \vee \psi)^+ \wedge \neg\psi \vdash_\pm \varphi^+$

$(\varphi \vee \psi)^+ \vdash_\pm \varphi^+ \vee \psi^+$

$\varphi^+ \vee \psi^+ \vdash_\pm (\varphi \vee \psi)^+$

$(\varphi \vee \psi)^- \vdash_\pm \varphi^-$, analogous for $\psi$

$(\varphi^- \vee \psi^-) \wedge \varphi \vdash_\pm \psi^-$

$(\varphi^- \vee \psi^-) \wedge \psi \vdash_\pm \varphi^-$

$\varphi^+ \wedge \psi^+ \vdash_\pm (\varphi \wedge \psi)^+$

$\varphi^- \wedge \psi^- \vdash_\pm (\varphi \vee \psi)^-$

$\varphi \to \psi \vdash_\pm \psi^- \to \varphi^-$

$\varphi \to \psi \vdash_\pm \varphi^+ \to \psi^+$

$(\varphi \to \psi)^+ \wedge \varphi \vdash_\pm \psi^+$

$(\varphi \to \psi)^+ \wedge \neg\psi \vdash_\pm \varphi^-$

$(\varphi \to \psi)^- \vdash_\pm \psi^-$

$(\varphi \to \psi)^- \vdash_\pm \varphi^+$

$(\varphi \to \psi)^- \vdash_\pm \varphi^- \to \neg\psi$

$\neg(\varphi \to \psi) \vdash_\pm \varphi$

$\neg(\varphi \to \psi) \vdash_\pm \neg\psi$

Furthermore, $\vdash_\pm$ takes equivalence of propositions into account, satisfies modus ponens, contraposition, and so on. If $\Sigma \vdash_\pm \varphi$ then $\Delta \vdash_\pm \varphi$ for any $\Delta \supseteq \Sigma$.

**Proposition 5.15.** *If $\Sigma \vdash_\pm \varphi$ then $\Delta \vdash_\pm \varphi$ for any $\Delta \supseteq \Sigma$.*

*Proof.* Let $T$ be the finished tableau with premises $\Sigma$. $\Sigma \vdash_\pm \varphi$ means that on every open branch of $T$, $\varphi : 1$ holds.

As $\Delta \supseteq \Sigma$, the tableau with premises $\Delta$ can be constructed from $T$ by adding the propositions $\Delta \setminus \Sigma$ to its open branches. Clearly, after finishing the resulting tableau, $\varphi : 1$ still holds on all the open branches. □

**Proposition 5.16.** *If $\Sigma \vdash_\pm \varphi$ and $\Sigma \vdash_\pm \psi$ then $\Sigma \vdash_\pm \varphi \wedge \psi$.*

*Proof.* Let $T$ be the finished tableau with premises $\Sigma$. $\Sigma \vdash_\pm \varphi$ means that on every open branch of $T$, $\varphi : 1$ holds. Analogously, on every open branch of $T$, $\psi : 1$ holds.

Now, assume $\Sigma \vdash_\pm \varphi \wedge \psi$ does not hold. This means that $T$ contains an open branch that is compatible with an evaluation of $\varphi \wedge \psi$ to less than 1. This can be tested by appending $\varphi \wedge \psi :< 1$ to every open branch of $T$. Finishing the resulting tableau reveals that now all branches are closed. This is obvious from the atomic tableaux (t1), (t7), and (t8) given in Section 3.1.

Consequently, $\varphi \wedge \psi : 1$ holds on all open branches of $T$ which corresponds to $\Sigma \vdash_\pm \varphi \wedge \psi$, contradicting the assumption. □

$Cn_\pm(S)$ is consistent if it does not contain $\varphi$ and $\varphi^-$ for any proposition $\varphi$, otherwise it is inconsistent (and therefore contains $\bot_w$).

The rule $\varphi^- \vdash_\pm (\varphi \wedge \psi)^-$ is not necessary for nonsatisfiable propositions $\varphi$, as they are given already (see Proposition 5.8). However, if $\varphi$ is not satisfiable for some other reason, e.g. because $\overline{\varphi}$ is in the set of propositions, being able to infer all other conjunctions that are not satisfiable any more might be helpful. By equivalence $\varphi^+ \equiv (\neg\varphi)^-$, the same holds for $\varphi^+ \vdash_\pm (\varphi \vee \psi)^+$. Furthermore, the rules for $(\varphi \wedge \psi)^\pm$ can be obtained by reformulating those for $(\varphi \vee \psi)^\mp$. Example 32 illustrates one case.

**Example 32.** $(\varphi \wedge \psi)^- \vdash_\pm (\neg(\varphi \wedge \psi))^+ \vdash_\pm (\neg\varphi \vee \neg\psi)^+ \vdash_\pm (\neg\varphi)^+ \vee (\neg\psi)^+ \vdash_\pm \varphi^- \vee \psi^-$, i.e. $(\varphi \wedge \psi)^- \vdash_\pm \varphi^- \vee \psi^-$.

Other rules can be derived as well. For example, $\varphi \to \overline{\varphi} \vdash_{\pm} \varphi^-$ is a special case of the rule $\left(\psi_1^- \vee \psi_2^-\right) \wedge \psi_1 \vdash_{\pm} \psi_2^-$. As $((\varphi \to \overline{\varphi}) \wedge \varphi)$ is not satisfiable, $\vdash_{\pm} ((\varphi \to \overline{\varphi}) \wedge \varphi)^-$ holds, i.e. $((\varphi \to \overline{\varphi}) \wedge \varphi)^-$ is a tautology. Using the rule from Example 32, it follows that $\vdash_{\pm} (\varphi \to \overline{\varphi})^- \vee \varphi^-$.

$(\varphi \to \overline{\varphi}) \vdash_{\pm} \left((\varphi \to \overline{\varphi})^- \vee \varphi^-\right) \wedge (\varphi \to \overline{\varphi})$. This step is justified by Proposition 5.16 and the fact that the first conjunct is a tautology which can be inferred from any set of propositions. Now using the inference rule $\left(\psi_1^- \vee \psi_2^-\right) \wedge \psi_1 \vdash_{\pm} \psi_2^-$, substituting $\psi_1$ with $(\varphi \to \overline{\varphi})$ and $\psi_2$ with $\varphi$, we get $\left((\varphi \to \overline{\varphi})^- \vee \varphi^-\right) \wedge (\varphi \to \overline{\varphi}) \vdash_{\pm} \varphi^-$.

Analogous argumentations apply to

$\varphi \to \neg\varphi \vdash_{\pm} \varphi^-$,

$\varphi \vdash_{\pm} \overline{\varphi}^-$,

$\neg\varphi \vdash_{\pm} (\neg\overline{\varphi})^-$,

$\overline{\varphi} \vdash_{\pm} \varphi^-$,

$\neg\overline{\varphi} \vdash_{\pm} \varphi^+$,

$\neg\varphi \vee \overline{\varphi} \vdash_{\pm} \varphi^-$,

$\neg\varphi \vdash_{\pm} \varphi^-$,

$\varphi \vdash_{\pm} \varphi^+$.

In fact, this pattern creates rules for all cases where a (finite) set of propositions $\Sigma$ implies that another proposition $\varphi$ cannot be evaluated to true. $\vdash_{\pm} (\bigwedge \Sigma \wedge \varphi)^-$ and substituting $\psi_1$ with $\bigwedge \Sigma$ and $\psi_2$ with $\varphi$ lets this become obvious.

The question why the rule is $(\varphi \vee \psi)^+ \wedge \neg\varphi \vdash_{\pm} \psi^+$ and not $(\varphi \vee \psi)^+ \vdash_{\pm} \neg\varphi \to \psi^+$ might arise. There is one case for which the second version is not justified. An evaluation of $\varphi$ to $u$ and $\psi$ to 0 clearly satisfies the first part, but the implication will not be evaluated to 1, so the more conservative first rule is necessary. There are more rules of the first type. Similar examples exist for them.

The list of rules we have given might not be complete. Appendix B gives an elaborate argumentation for the soundness of the rules. A complete axiomatization of the inference relation is subject to future work.

# 6 Conversion of the belief revision framework into an argumentation framework

## 6.1 Introduction

During the discussion of the belief revision framework [1] the question arose, whether the framework could be translated into an argumentation framework and how it would look like. In this section we want to answer this question partially.

The procedure that calculates the belief set from a given sequence can be divided into two steps – identifying a belief base, i.e. the correct subset of propositions in the sequence, and then applying the closure under consequence to that set. So the task of calculating the belief set can be reformulated to the task: Given a sequence $\sigma=(\varphi_1, \ldots, \varphi_n)$, calculate $Cn\,(BelBase\,(\sigma))$.

In the following, we need the elements of $\sigma$ to satisfy the following property: $\varphi_i \neq \varphi_j$ for $i \neq j$. That is, propositions are not allowed to appear twice in the sequence. This seems to be a strong restriction, but in fact every sequence can be transformed into one with this property without affecting the belief base or the belief set the sequence produces. This is done by keeping the latest appearance of the proposition and deleting the other ones.

**Example 33.** $(a, b, c, a \vee b, b, d, c \wedge d, d)$ becomes $(a, c, a \vee b, b, c \wedge d, d)$

The reason why this transformation leaves the belief base unchanged is that an earlier appearance of the proposition in the sequence has no effect. In the process of calculation the latest occurence has already been considered. If it was consistent then, it was included and already is part of the belief base. If it was inconsistent then, it still is.

**Definition 6.1.** Let $\sigma = (\varphi_1, \ldots, \varphi_n)$ be a sequence of propositions.

$pos_\sigma\,(\varphi_i) \stackrel{\text{def}}{=} i$

$maxp_\sigma\,(\{\varphi_1, \ldots, \varphi_m\}) \stackrel{\text{def}}{=} \max_{i=1\ldots}\,(pos_\sigma\,(\varphi_i))\,;\; maxp_\sigma\,(\emptyset) \stackrel{\text{def}}{=} 0$

$Set\,(\sigma) \stackrel{\text{def}}{=} \{\varphi_1, \ldots, \varphi_n\}$

$BelBase\,(\sigma) \stackrel{\text{def}}{=} B$ such that

(1) $B \subseteq Set\,(\sigma)$

(2) $\bot_w \notin Cn\,(B)$ and

(3) $\neg\exists S(S \subseteq Set\,(\sigma) \wedge \bot_w \notin Cn\,(S) \wedge maxp_\sigma\,(S \setminus B) > maxp_\sigma\,(B \setminus S))$.

*pos* gives the position of a proposition within a sequence. This is the reason why the proposition should only appear once. If it did not, the number would not be unique and the definition for the function would have to be more complicated. For the sequence in Example 33 we have $pos\,(a \vee b) = 3$, $pos\,(d) = 6$.

*maxp* calculates the positions of all the elements of a set of propositions in a sequence and returns the greatest number. For the sequence in Example 33 we have $maxp\,(\{a, b, c\}) = 4$.

*Set* simply transforms a sequence into a set.

The first two conditions of the definition of $BelBase$ are immediate. It has to be a consistent subset of the propositions appearing in the sequence. The definition is for the logic of disbelief. This is why the notion of inconsistency is not restricted to contradictions.

The third condition is more complicated so we will present it more in detail. It states that $B$ can only be the belief base if $B$ beats all other candidate $S$ except itself. $S$ is a candidate if it is a set that satisfies the first two conditions. $B$ beats $S$ if $B$ contains a proposition $\varphi$ which is not an element of $S$ and $\varphi$ appears later in the sequence than any $\psi$ that is element of $S$ but not of $B$.

**Example 34.**

- Consider $\sigma = (a, b, \neg a)$.

  $\{a, b, c\}$ does not satisfy condition 1, so it is not a candidate for being the belief base.

  $\{a, \neg a\}$ does not satisfy condition 2, so it can be disregarded as well.

  $S_1 = \{b\}$, $S_2 = \{a, b\}$, $S_3 = \{\neg a, b\}$ all satisfy the first two conditions.

  $S_1 \setminus S_2 = \emptyset$, $S_2 \setminus S_1 = \{a\}$. Obviously, $0 = maxp_\sigma\,(\emptyset) < maxp_\sigma\,(\{a\}) = 1$, so $S_1$ is not a candidate.

  $S_2 \setminus S_3 = \{a\}$, $S_3 \setminus S_2 = \{\neg a\}$. $1 = maxp_\sigma\,(\{a\}) < maxp_\sigma\,(\{\neg a\}) = 3$, so, as expected, $S_3$ wins.

- One possibility to present the the mechanism in a simplified way is not to compare the candidates but sets of natural numbers. Numbers that correspond to the position of the propositions in the sequence are substituted for the propositions. After elimitating the common subset, the set with the greatest number wins. An empty set loses agains any nonempty set.

The example should illustrate that a consistent superset will always beat the subset. This means that as many propositions as possible are incorporated into the belief base. Furthermore, later propositions have higher priority. If the common subset is disregarded, of two consistent sets, the one that has the latest proposition wins.

So calculating $BelBase(\sigma)$ essentially means: identify the subset of propositions that has a conflict-free closure under inference and that incorporates those propositions as late in the sequence as possible as well as a maximal number of them. This corresponds to the selection procedure described in Section 1.2. That is, $BelBase(\sigma)$ is the set of propositions from $\sigma$ which were accepted during the calculation of $Bel(\sigma)$.

We will show that this selection procedure can be translated into an argumentation framework. We first introduce the parts of the argumentation framework that are necessary for our purpose and then prove the correctness of our translation.

## 6.2 Vreeswijk's Abstract Argumentation System

The elements of a belief set could be seen as conclusions of reasoning processes. In the belief set, the information about why the propositions are contained in it is lost, however. Argumentation frameworks can provide this information because they say something about the interaction between propositions – which propositions support others, which attack others.

The identification of the belief base can be interpreted as such an argumentation process: as these propositions are believed to be true and their conclusions are incompatible with this proposition, it cannot be incorporated. The task now is to formalize this.

For this purpose, Vreeswijk's Abstract Argumentation System appealed to us. Definitions 6.2-6.13 are from [11].

**Definition 6.2.** An *abstract argumentation system* is a triple $(L, R, \leq)$, where $L$ is a language (containing an element $\perp$ representing a contradiction), $R$ is a set of rules of inference , and $\leq$ is a reflexive and transitive order on arguments.

**Definition 6.3.** Let $L$ be a language.

1. A *strict rule of inference* is a formula of the form $\phi_1, \ldots, \phi_n \rightarrow \phi$, where $\phi_1, \ldots, \phi_n$ is a finite, possibly empty, sequence in $L$ and $\phi$ is a member of $L$.

2. A *defeasible rule of inference* is a formula of the form $\phi_1, \ldots, \phi_n \Rightarrow \phi$, where $\phi_1, \ldots, \phi_n$ is a finite, possibly empty, sequence in $L$ and $\phi$ is a member of $L$.

A *rule of inference* is a strict or a defeasible rule of inference.

**Remark 35.** $\rightarrow$ and $\Rightarrow$ must not be part of the language, i.e. inference rules cannot be elements of the language. Of course, implication is allowed, but the arrows must be distinguished. The arrow of the inference rules could be thought of as $\vdash$.

**Definition 6.4.** An argument $\delta$ is

1. a member of $L$. Then $prem(\delta) = \{\delta\}$, $conc(\delta) = \delta$, $sent(\delta) = \{\delta\}$, $sub(\delta) = \{\delta\}$.

2. a formula of the form $\delta_1, \ldots, \delta_n \rightarrow \phi$, where $\delta_1, \ldots, \delta_n$ is a finite, possibly empty, sequence of arguments, such that $conc(\delta_1) = \phi_1, \ldots, conc(\delta_n) = \phi_n$ for some rule $\phi_1, \ldots, \phi_n \rightarrow \phi$ in $R$, and $\phi \notin \bigcup sent(\delta_i)$.
   Then $prem(\delta) = \bigcup prem(\delta_i)$, $conc(\delta) = \phi$, $sent(\delta) = \bigcup sent(\delta_i) \cup \{\phi\}$, $sub(\delta) = \bigcup sub(\delta_i) \cup \{\delta\}$ .

3. a formula of the form $\delta_1, \ldots, \delta_n \Rightarrow \phi$, where $\delta_1, \ldots, \delta_n$ is a finite, possibly empty, sequence of arguments, such that $conc(\delta_1) = \phi_1, \ldots, conc(\delta_n) = \phi_n$ for some rule $\phi_1, \ldots, \phi_n \Rightarrow \phi$ in $R$, and $\phi \notin \bigcup sent(\delta_i)$.
   Then $prem(\delta) = \bigcup prem(\delta_i)$, $conc(\delta) = \phi$, $sent(\delta) = \bigcup sent(\delta_i) \cup \{\phi\}$, $sub(\delta) = \bigcup sub(\delta_i) \cup \{\delta\}$ .

So combining elements of the language and rules of inference into trees creates an argument.

**Example 36.** Figure 7 shows arguments formed of an argumentation system with the language $\{a, b, c, d, e, \bot\}$ and the inference rules $d \Rightarrow a; c, e \rightarrow \bot; a, b \rightarrow c$.

The set of premises of the first argument is $\{b, d\}$, its conclusion is $c$, the sentences occuring are $\{d, a, b, c\}$, and its subarguments are given in Figure 8.

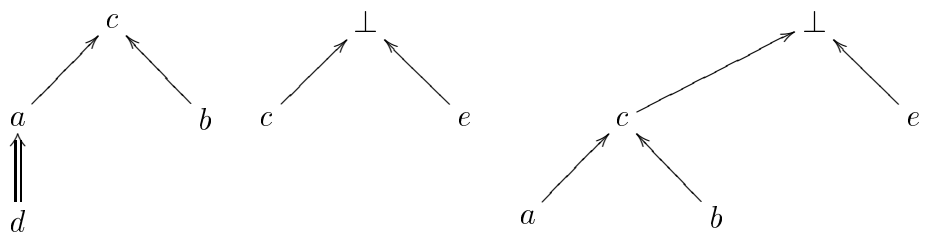



Figure 7: Argument examples

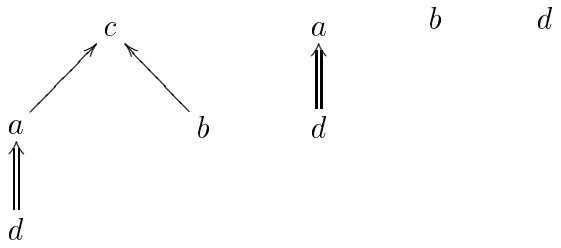



Figure 8: Subarguments

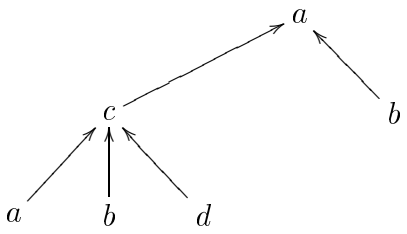



Figure 9: Not an argument

**Example 37.** The tree in Figure 9 does not constitute an argument. The sentence $a$ appears twice on one branch. This is not allowed by Definition 6.4, which prevents a sentence from being one of its own premises in an argument.

**Definition 6.5.** An argument $\delta$ is strict if $\delta \in L$, or $\delta_1, \ldots, \delta_n \to \delta$, where all $\delta_i$ are strict arguments.

**Example 38.** The left argument in Figure 10 is strict as all subarguments are strict. The right is not strict as there is a subargument whose top rule is defeasible.

**Definition 6.6.** Let $L$ be a language and let $P \subseteq L$.

An argument is *based on* $P$ if the premises of that argument constitute a subset of $P$; a set of arguments is based on $P$ if all its members are based on $P$.

A member of $L$ is based on $P$ if it is the conclusion of an argument that is based on $P$.

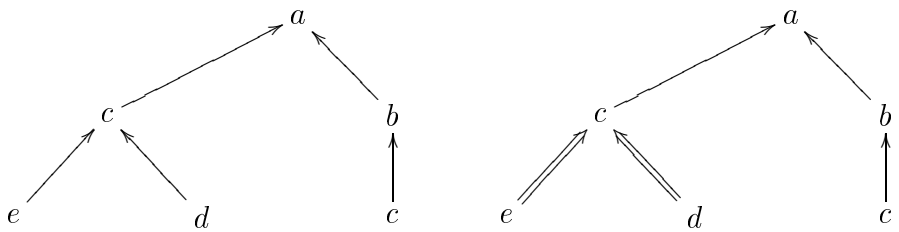



Figure 10: Strict and nonstrict argument

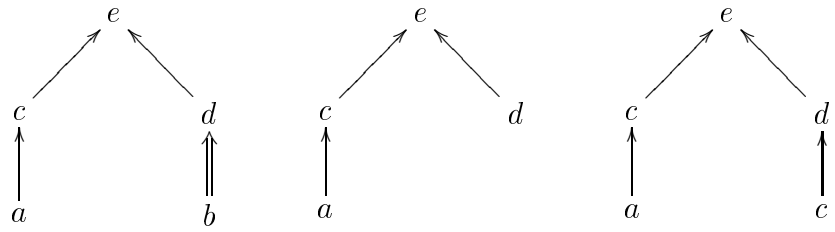



Figure 11: Arguments

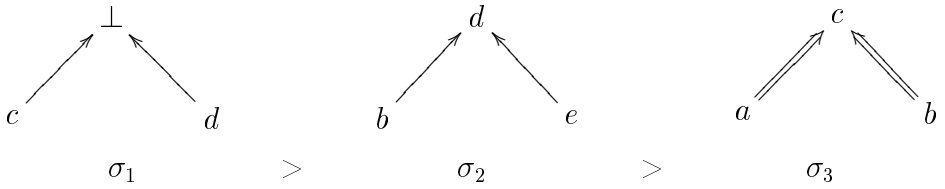



Figure 12: Compatibility

**Example 39.** While the left argument in Figure 11 is based on $\{a, b\}$, the middle one is not. The right argument is not based on the set either, as it does not contain $c$. The fact that $a$ is element of the set and there is a rule $a \to c$ does not change the matter.

**Definition 6.7.** An argument $\tau$ is an *underminer* of a set of arguments $\Sigma$, if $\delta < \tau$ for some $\delta \in \Sigma$. In this case, $\Sigma$ is *undermined* by $\tau$.

**Definition 6.8.** An argument $\delta$ is *in contradiction* if $conc\,(\delta) = \bot$.

**Definition 6.9.** A subset $P$ of $L$ is *incompatible* if there exists a strict argument in contradiction that is based on $P$. A subset of $L$ is *compatible* if it is not incompatible.

Compatibility naturally extends to arguments. Thus, a set of arguments $\Sigma$ is compatible if $conc\,(\Sigma)$ is compatible.

**Definition 6.10.** A *base set* is a finite compatible subset of $L$.

**Example 40.** Consider Figure 12. $\sigma_2$ is an underminer of the set of arguments $\{\sigma_1, \sigma_3\}$, as $\sigma_2 > \sigma_3$. $\sigma_3$ is not an underminer of the set of arguments $\{\sigma_1, \sigma_2\}$, as $\sigma_3 < \sigma_1$ and $\sigma_3 < \sigma_2$.

The argument $\sigma_1$ is in contradiction, as its conclusion is $\bot$.

The sets $\{c, d\}$, $\{c, b, e\}$ are incompatible, as there exist strict arguments in contradiction that are based on these sets.

The set $\{a, b, d\}$ is not incompatible. There exists an argument in contradiction based on the set, but it is not strict. However, the set of arguments $\{\sigma_2, \sigma_3\}$ is incompatible, as the set of their conclusions is incompatible.

The sets $\{c, d\}$, $\{c, b, e\}$ are not base sets, as they are incompatible. The sets $\{b, c\}$, $\{a, b, e\}$ and $\emptyset$ are base sets.

**Remark 41.** If there exists a strict argument in contradiction based on the empty set, then there are no base sets for the language.

**Definition 6.11.** Let $P$ be a base set and let $\delta$ be an argument. A set of arguments $\Sigma$ is a *defeater* of $\delta$ if it is incompatible with this argument and not undermined by it.

**Example 42.** Consider Figure 12 again. The argument $\sigma_3$ and the set of arguments $\{\sigma_2\}$ are incompatible, and as the set is not undermined by $\sigma_3$, it is a defeater of the argument.

The argument $\sigma_2$ and the set of arguments $\{\sigma_3\}$ are incompatible, but as the set is undermined by $\sigma_2$, it is not a defeater of the argument.

**Definition 6.12.** Let $P$ be a base set. An argument $\sigma$ is in force at level 1 on the basis of $P$ if it is based on $P$. Let n>1. An argument $\delta$ is in force at level n on the basis of $P$, written $P \mid\sim_n \delta$, if

1. The set $P$ contains $\delta$; or
2. For some arguments $\delta_1, \ldots, \delta_n$ we have $P \mid\sim_n \delta_1, \ldots, \delta_n$ and $\delta_1, \ldots, \delta_n \rightarrow \delta$; or
3. For some arguments $\delta_1, \ldots, \delta_n$ we have $P \mid\sim_n \delta_1, \ldots, \delta_n$ and $\delta_1, \ldots, \delta_n \Rightarrow \delta$ and no set of arguments $\Sigma$ that is in force at level n-1 on the basis of $P$ is a defeater of $\delta$.

**Remark 43.** Notation is simplified in the following way: $\delta_1, \ldots, \delta_n \rightarrow \delta$ denotes that the argument $\delta$ is composed of the arguments $\delta_1, \ldots, \delta_n$ via the rule $\phi_1, \ldots, \phi_n \rightarrow \phi$, with $conc(\delta_1) = \phi_1, \ldots, conc(\delta_n) = \phi_n, conc(\delta) = \phi$.

The set of rules should not include a set that would allow to infer $\perp$ from an empty set of premises, e.g. $\rightarrow \perp$, because otherwise the empty set would be a defeater for all defeasible arguments as it cannot be undermined by any rule. This is, of course, a reasonable demand. A logic should not be inherently inconsistent.

**Definition 6.13.** The expression $info_n(P)$ denotes the set of arguments $\{\delta \mid P \mid\sim_n \delta\}$.

An argument $\delta$ is ultimately undefeated if and only if, for some $n \geq 1$, we have $P \mid\sim_{n+k} \delta$, for every $k \geq 1$.

An argument $\delta$ is provisionally undefeated if and only if, for every $n \geq 1$, we have $P \mid\sim_{n+k} \delta$, for some $k \geq 1$, and not $P \mid\sim_{n+l} \delta$, for some $l \geq 1$.

An argument $\delta$ is ultimately defeated if and only if, for some $n \geq 1$, the entailmant $P \mid\sim_{n+k} \delta$ does no longer hold, for every $k \geq 1$.

**Example 44.** Consider the following simple argumentation system: $L = \{a, b, c, d, \bot\}$, $R = \{\Rightarrow a, \Rightarrow b, \Rightarrow c, \Rightarrow d, (a, b \rightarrow \bot), (c, d \rightarrow \bot)\}$. Concerning the order on the arguments, we have $\Rightarrow a = \Rightarrow b$ and $\Rightarrow c > \Rightarrow d$. We let the base set $P$ be the empty set. Consequently, no arguments are in force at level 1. As the premises of the defeasible rules are empty, we only have to check whether the previous level contains defeaters.

1. $\emptyset$

2. $\Rightarrow a, \Rightarrow b, \Rightarrow c, \Rightarrow d, (a, b \rightarrow \bot), (c, d \rightarrow \bot)$

   As there are no arguments in force at the first level, there cannot be defeaters, so all defeasible rules are in force at the second level. The two strict arguments are in force as well, as their premises are conclusions from arguments in force at the level.

3. $\Rightarrow c$

   $\Rightarrow a$ is not in force at this level, as the previous one now contains a defeater – the set $\{\Rightarrow b\}$. They are incompatible and $\Rightarrow a$ does not undermine the set, as it is not stronger than any of the arguments contained in it. $\Rightarrow b$ is not in force for the same reason. Its defeater is $\{\Rightarrow a\}$.

   For $\Rightarrow c$ there is a potential defeater as well – $\{\Rightarrow d\}$, but the set is undermined, because $\Rightarrow c$ is stronger than $\Rightarrow d$, so $\Rightarrow c$ remains undefeated. $\Rightarrow d$, in turn, is defeated by $\{\Rightarrow c\}$.

   For obvious reasons, the arguments with conclusion $\bot$ are not in force.

4. $\Rightarrow a, \Rightarrow b, \Rightarrow c, (a, b \rightarrow \bot)$

   $\Rightarrow d$ remains defeated by $\{\Rightarrow c\}$, and $\Rightarrow c$ remains undefeated.

   The defeaters for $\Rightarrow a$ and $\Rightarrow b$ were not among the arguments in force at the previous level, so the two are in force again.

5. $\Rightarrow c$

   The constellation at this level is the similar to that at level 3. $\Rightarrow d$ remains defeated and the defeaters for $\Rightarrow a$ and $\Rightarrow b$ have reappeared.

6. $\Rightarrow a, \Rightarrow b, \Rightarrow c, (a, b \rightarrow \bot)$

7. ...

It is obvious that the sets of arguments in force continue to alternate. $\Rightarrow c$ is ultimately undefeated. Starting at level 2 it is in force at all levels. $\Rightarrow d$ is ultimately defeated. Starting at level 3 it is not in force at any level. Both $\Rightarrow a$ and $\Rightarrow b$ are provisionally defeated. They keep disappearing and reappearing.

Example 44 is a particularily simple example to illustrate the dynamics of the process, showing how arguments can be provisionally defeated etc. Convergence is not necessarily reached so soon and alternation need not only comprise two steps.

## 6.3 The AAS for belief revision

We now define the components of the argumentation system, so that the procedure given in Definition 6.12 identifies the belief base. The basic idea is to make sure that the propositions neglected in the original approach are the ones defeated in the argumentation approach.

Let $L$ be a language and $\sigma=(\varphi_1, \ldots \varphi_n)$ , with $\varphi_i \neq \varphi_j$ for $i \neq j$, a sequence as above.

**Definition 6.14.** $R_\sigma \stackrel{\text{def}}{=} \{\Rightarrow \varphi_i \mid \varphi_i \in \sigma\} \cup$

$\{all\ instantiations\ of\ inference\ rules\ formulated\ as\ strict\ rules\}$.

Note that we do not restrict ourselves to propositional logic or the logic of disbelief introduced. That is, the results given in the following are of amore general nature. Of course, we think of applying the results to these twologics.

The propositions from the sequence are added to the inference rules the logic provides as defeasible arguments with empty premises. It is natural that these inference rules are strict rules. As the logic does not prohibit inferences that lead to inconsistent sets, the argumentation framework should not concern itself with this, either. Consistency will be ensured by the defeasibility of the "premises".

Propositional logic and the logic of disbelief obviously have valid inference rules like $a \rightarrow a$ or $a, b \vee c \rightarrow a$. These can be neglected because they cannot be used to form arguments, as can be seen from Definition 6.4. This is not a restriction, however, as these rules do not enrich the possibilities. This can be seen in Figure 13. The same conclusion can be inferred from a subset of the premises of rules like that.

It should be noted that there is an essential difference between the inference rules $a, b \rightarrow a$ and $a \wedge b \rightarrow a$. In the first, the conclusion is syntactically equivalent to one of the premises while in the second, premise and conclusion are completely different.

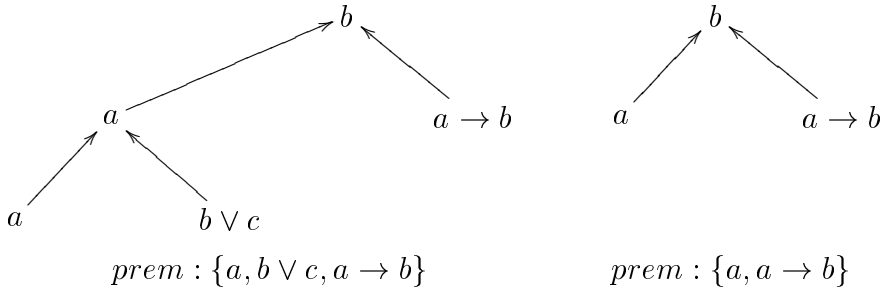



Figure 13: "Invalid" inference rule vs. valid ones

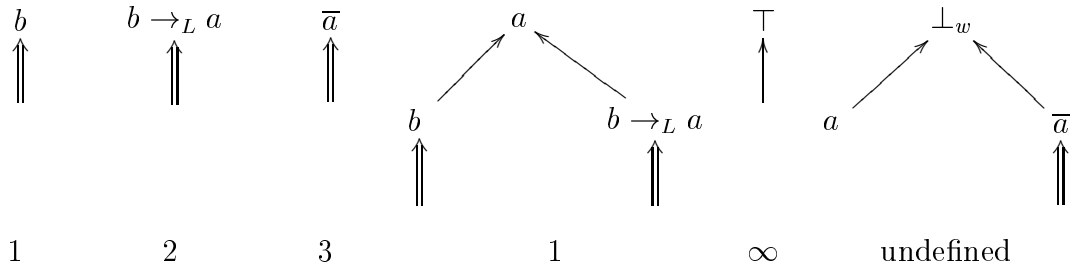



Figure 14: Arguments and their strength

**Definition 6.15.** Let $Arg$ be the set of all arguments created by $R_\sigma$. *strength* is a function that assigns a natural number or $undefined$ to every argument of $Arg$.

$$strength\,(\delta) \stackrel{\text{def}}{=} \begin{cases} \infty,\ \delta = \text{empty premise} \\ i,\ \delta = \Rightarrow \varphi_i \\ undefined, \delta \in L \\ \min_{\tau \in sub(\delta)\setminus\{\delta\}} \left(strenght\,(\tau)\right),\ otherwise \end{cases}$$

$\min(\Sigma)$ is defined such that it takes the the value $undefined$ if one of the elements of $\Sigma$ is undefined.

**Definition 6.16.** The order on arguments $\delta_i$ is defined by

$\delta_1 \leq \delta_2$ if and only if $strength\,(\delta_1) \leq strength\,(\delta_2)$[10].

**Remark 45.** The strength of an argument is determined by its weakest "premise". It should be noted that this definition does not allow to compare all possible arguments, but as we will always let the base set $P$ be empty, we actually only have arguments with empty premises, i.e. all arguments start with defeasible rules of the form $\Rightarrow \varphi_i$ which we will therefore call premises. The definition allows to compare all of these arguments.

[10]The first $\leq$ denoting the order to be defined, the second the common order on natural numbers. We think this definition to be so intuitive that the ambiguity in notation does not create confusion.

**Example 46.** Consider the sequence $\sigma = (b, b \to_L a, \overline{a})$, $\to_L$ denoting implication. $R_\sigma$ would be $\{\Rightarrow b, \Rightarrow b \to_L a, \Rightarrow \overline{a}, (b \to_L a, b \to a), (a, \overline{a} \to \bot_w), \dots\}$. Figure 14 shows some arguments with their respective strength. The strength of the rightmost argument is undefined because the strength of the subargument $a$ is not defined.

We now choose $(L, R, \leq)$ as follows: Let $L$ be the language used in the belief revision framework, i.e. propositional language or the language of disbelief. In the first case $\bot$ coincides with the symbol denoting the contradiction, in the latter we chose $\bot_w$ to be the symbol for the contradictory conclusion. Further $R$ and $\leq$ are chosen as in Definitions 6.14-6.16. Using this argumentation system, the above set $BelBase$ of basic beliefs can be identified by calculating the limit of the inductive warrant $(\lim_n info_n(P))$ where the base set $P$ is empty. The result of the calculation will contain the "premises" that correspond to the propositions of the sequence that are included in the belief base.

The proof for that will be given in the next two sections, but first a more complex example.

**Example 47.** Consider the sequence $\sigma = \big(e, a \vee b, b, c \wedge \overline{c}, b \to \overline{d}, d, \overline{e}, \overline{\overline{e}}, a \to c\big)$. We will illustrate the calculation of the belief base using the approach introduced in Section 1.2 and the inductive warrant of the argumentation system that is created by the sequence and its implied set of rules and order on arguments.

- The calculation starts with the empty set. $a \to c$ and $\overline{\overline{e}}$ are inserted as they do not introduce an inconsistency. $\overline{e}$ is neglected, because it it is not compatible with $\overline{\overline{e}}$. $d$ and $b \to \overline{d}$ do not cause an inconsistency and are included. $c \wedge \overline{c}$ is inconsistent in itself, so it is left out. $b$ cannot be included. Together with $b \to \overline{d}$ it implies $\overline{d}$, which is inconsistent with $d$, but the latter isalready included. $a \vee b$ is no problem, however, just like $e$.

  So the belief base is $\big\{a \to c, \overline{\overline{e}}, d, b \to \overline{d}, a \vee b, e\big\}$

- As the base set is empty, no arguments are in force at level 1.

  1. $\emptyset$
  2. $\big\{\Rightarrow e, \Rightarrow a \vee b, \Rightarrow b, \Rightarrow b \to \overline{d}, \Rightarrow d, \Rightarrow \overline{e}, \Rightarrow \overline{\overline{e}}, \Rightarrow a \to c, \dots\big\}$

     $\Rightarrow c \wedge \overline{c}$ is not in force as the empty set is a defeater of the argument. This means that the argument is ultimately defeated, because the empty set is subset of

any set, so this defeater will appear at all levels. For all the others, the previous level does not contain defeaters.

There are some further arguments in force at this level. These are strict continuations of the arguments given, e.g. $\big(b \to \overline{d}, b\big) \to \overline{d}$.

3. $\big\{\Rightarrow a \vee b, \Rightarrow b \to \overline{d}, \Rightarrow d, \Rightarrow \overline{\overline{e}}, \Rightarrow a \to c, \dots\big\}$

   $\Rightarrow e$ is not in force as it is defeated by $\{\overline{e}\}$ on the previous one.

   $\Rightarrow \overline{e}$ is not in force as it is defeated by $\big\{\overline{\overline{e}}\big\}$ on the previous one.

   $\Rightarrow b$ is not in force as it is defeated by $\big\{\Rightarrow b \to \overline{d}, \Rightarrow d\big\}$ on the previous one.

4. $\big\{\Rightarrow e, \Rightarrow a \vee b, \Rightarrow b \to \overline{d}, \Rightarrow d, \Rightarrow \overline{\overline{e}}, \Rightarrow a \to c, \dots\big\}$

   $\Rightarrow \overline{e}$ and $\Rightarrow b$ remain defeated as their defeaters are still among the arguments in force at the previous level. $\Rightarrow e$ reappears because its defeater was defeated at the previous level.

5. The fifth level is like the 4th.

So the argumentation system identifies the same proposition from the sequence.

We want to show that this works in general and not only in this example. Our first task in doing this is to prove that the limit $\lim_n info_n(P)$ exists, i.e. that (1) $P = \emptyset$ has a unique extension. Then we have to prove that (2) the "premises" in this extension correspond to the base beliefs in the above set $BelBase\,(\sigma)$.

## 6.4 Unique extension of the system

In order to prove (1), it is sufficient to show that on the basis of $P$ there are no provisionally defeated arguments (see Conjecture 4.5.6. in [11]). If there were provisionally defeated arguments, it would not be clear which propositions from the sequence would appear in the belief set.

The only arguments whose top rule is defeasible are the ones of the form $\Rightarrow \varphi_i$, i.e. the ones that encode the elements of the sequence. Simply put, an argument is defeated provisionally because it is either based on (provisionally defeated) arguments of the above form or it is one of that form itself. That is, several extensions could be interpreted as several possible belief bases, but, as we know, the belief base is uniquely determined.

The basic idea of the main proof is to show that, assuming an argument $\Rightarrow \varphi$ were provisionally defeated, its defeater would have the argument itself among its premises.

It is then possible to transform the defeater with the purpose of removing the argument from the premises which results in a defeater that contains only ultimately undefeated arguments leading to a contradiction to the assumption.

**Lemma 6.17.** *If an argument $\tau : \delta_1, \ldots, \delta_n \rightarrow \delta$ is provisionally defeated, then there is a subargument $\delta_i$ that is provisionally defeated.*

*Proof.* Assume all subarguments $\delta_1, \ldots, \delta_n$ are ultimately undefeated, then the argument $\tau$ is ultimately undefeated as well (follows from Definition 6.12), leading to a contradiction.

Assume any subargument $\delta_j$ is ultimately defeated, then $\tau$ is ultimately defeated as well, leading to a contradiction.

Which shows, that there must be a subargument $\delta_i$ that is provisionally defeated. □

**Lemma 6.18.** *An argument $\delta_1, \ldots, \delta_n \rightarrow \delta$ has the same strength as $\Rightarrow \varphi$ if and only if its weakest premise is $\Rightarrow \varphi$.*

*Proof.* Follows immediately from the definition of *strength*. □

**Lemma 6.19.** *Let $\delta = \delta_1, \ldots, \delta_n \rightarrow \delta$ and $\tau$ be arguments.*

*If $\Sigma \cup \{\delta\}$ is a defeater of $\tau$ then $\Sigma \cup \{\delta_1, \ldots, \delta_n\}$ is a defeater of $\tau$.*

*Proof.* Assume $\Sigma \cup \{\delta\}$ to be a defeater and $\Sigma \cup \{\delta_1, \ldots, \delta_n\}$ not to be a defeater of $\tau$. The latter means that either $\tau$ undermines $\Sigma \cup \{\delta_1, \ldots, \delta_n\}$ or $\tau$ is not incompatible with $\Sigma \cup \{\delta_1, \ldots, \delta_n\}$.

- $\tau$ undermines $\Sigma \cup \{\delta_1, \ldots, \delta_n\}$. Either $\tau$ undermines $\Sigma$, which contradicts the assumption of $\Sigma \cup \{\delta\}$ being a defeater of $\tau$, or $\tau$ undermines $\{\delta_1, \ldots, \delta_n\}$. This would mean that $\tau > \delta_i$ for some $0 < i < n$. By Lemma 6.18, this would imply that $\tau > \delta$, contradicting the assumption as well.

- $\tau$ is not incompatible with $\Sigma \cup \{\delta_1, \ldots, \delta_n\}$. But clearly, the set ofarguments that can be constructed by applying strict rules starting with the set $\Sigma \cup \{\delta\} \cup \{\tau\}$ is asubset of the arguments constructed by starting with $\Sigma \cup \{\delta_1, \ldots, \delta_n\} \cup \{\tau\}$. If$\perp_w$ is not among the conclusions of the latter, it cannot be amongthe conclusions of the former. This contradicts the assumption that $\Sigma \cup \{\delta\}$ is a defeater of $\tau$.

□

**Definition 6.20.** A set of arguments $\Sigma$ is a *minimal defeater* of an argument $\tau$ if $\Sigma$ is a defeater, but no subset of $\Sigma$ is a defeater of $\tau$.

**Proposition 6.21.** *Let $(L, R, \leq)$ be defined as above and let $P$ be the empty base set. Then there is no provisionally defeated argument based on $P$.*

*Proof.* From Lemma 6.17 immediately follows that it is sufficient to show that there are no provisionally defeated arguments of the form $\Rightarrow \varphi$.

Assume there is a nonempty set of provisionally defeated arguments of the form $\Rightarrow \varphi$. Let $\tau$ be the strongest argument among them.

In the following, we consider all minimal defeaters of $\tau$. They can be partitioned into three groups.

1. Defeaters containing ultimately undefeated arguments only.
2. Defeaters containing at least one ultimately defeated argument.
3. Defeaters containing no ultimately defeated argument and at least one provisionally defeated argument.

There cannot be a defeater of the first type. This would lead to a contradiction with the assumption that $\tau$ is provisionally defeated.If there was one of the first type, $\tau$ would be ultimately defeated.Not all of the defeaters can be of the second type. If they were, $\tau$ would be ultimately undefeated, contradicting the assumption. In ourconsiderations we can neglect the defeaters of the second type – starting ata particular level, they are not among the defeaters of $\tau$ any more.

So we can restrict our attention to the defeaters of the third type which obviously have to exist. Let $\Sigma$ be an arbitrary minimal defeater ofthe third type of $\tau$.

All the arguments in $\Sigma$ are of equal orgreater strength than $\tau$. Otherwise, $\tau$ would undermine$\Sigma$ and $\Sigma$ would not be a defeater. In particular, allprovisionally defeated arguments in $\Sigma$ are of equal strength with$\tau$. They cannot be of greater strength. If there was one, by Lemma6.17 and Lemma 6.18 there would exist a provisionallydefeated argument of the form $\Rightarrow \varphi$ with greater strengththan $\tau$ which contradicts the assumption that $\tau$ is the strongestargument of that form.

Repeating the essential point: all provisionally defeated arguments in $\Sigma$ are of equal strength with $\tau$. That is, $\tau$ is among the premises of all the provisionally defeated argu-

ments. Furthermore, $\tau$ is the weakest premise of every provisionally defeated argument in $\Sigma$.

With this knowledge, we can construct a defeater that contains only ultimately undefeated arguments leading to a contradiction with the assumption that $\tau$ is provisionally defeated. So, the set of provisionally defeated arguments is smaller. Then we apply the proof to the next strongest argument in the now smaller set and continue doing so until the set is empty. This leads to a contradiction to the assumption that there is a nonempty set of provisionally defeated arguments of the form $\Rightarrow \varphi$.

The construction of the defeater is based on the iterated application of Lemma 6.19. An argument $\delta = \delta_1, \ldots, \delta_n \rightarrow \delta$ is broken down into its components if $\tau \in sub(\delta_i)$ for some $i$. $\delta$ is then removed from $\Sigma$ and all $\delta_i$ are added to $\Sigma$. The new $\Sigma$ is still a defeater for $\tau$. This is continued until there is no argument with a strict top rule left that has $\tau$ as a subargument.

Now all arguments in $\Sigma \setminus \{\tau\}$ are ultimately undefeated. If this was not the case, by Lemma 6.17 a provisionally defeated argument $\hat{\tau}$ of the form $\Rightarrow \varphi$ with $\hat{\tau} > \tau$ would exist, which contradicts the assumption that $\tau$ is the strongest of these arguments. Since $\Sigma \cup \{\tau\} = \Sigma$, it is clear that $\Sigma \setminus \{\tau\}$ is incompatible with $\tau$. Consequently, $\Sigma \setminus \{\tau\}$ is a defeater of $\tau$ which contains only ultimately undefeated arguments, leading to the desired contradiction. □

## 6.5 Correspondence between $BelBase(\sigma)$ and the premises

In order to show (2) – that the "premises" in this extension correspond to the base beliefs in the above set $BelBase(\sigma)$ – we introduce the following notation.

**Definition 6.22.** Let $\sigma = (\varphi_1, \ldots, \varphi_n)$.

$res_i(\sigma) \stackrel{\text{def}}{=} (\varphi_{n-i}, \ldots, \varphi_n)$

$R_i \stackrel{\text{def}}{=} \{\Rightarrow \varphi_j \mid \varphi_j \in res_i(\sigma)\} \cup \{inference\ rules\ as\ strict\ rules\}$

$getf(X) \stackrel{\text{def}}{=} \{\varphi \mid \Rightarrow \varphi \in X\}$

$\lim_n info_n^{R_i}(X)$ is the extension of the argumentation system $(L, R_i, \leq)$ with the base set $X$.

Let $\sigma$ be a sequence of sentences of $L$ with definitions for $pos, \ldots$ as above. $res_i(\sigma)$ represents the tail part of $\sigma$ with length $i+1$. $(L, R_i, \leq)$ is the restriction of the argumentation system to the shortened sequence. Note, however, that all calculations of

*pos*, *strength*, ... are still executed with respect to the entire sequence $\sigma$. $getf\,(X)$ is used to extract the premises from a set, i.e. the propositions of the sequence. $\lim_n info_n^{R_i}\,(X)$ is the limit of the inductive warrant of the argumentation system that is restricted to the tail of the sequence.

**Example 48.** Consider the sequence $\sigma = \left(e, a \vee b, b, c \wedge \overline{c}, b \to \overline{d}, d, \overline{e}, \overline{\overline{e}}, a \to c\right)$.

$res_0 = (a \to c)$, $res_1 = \left(\overline{\overline{e}}, a \to c\right)$, $res_5 = \left(c \wedge \overline{c}, b \to \overline{d}, d, \overline{e}, \overline{\overline{e}}, a \to c\right)$

$R_0 = \{\Rightarrow a \to c, strict\ rules\}$, $R_5 = \left\{\Rightarrow c \wedge \overline{c}, \Rightarrow b \to \overline{d}, \Rightarrow d, \Rightarrow \overline{e}, \Rightarrow \overline{\overline{e}}, \Rightarrow a \to c, \ldots\right\}$

$getf\left(\left\{\Rightarrow c \wedge \overline{c}, b \to \overline{d}, \Rightarrow d, \overline{e} \vee c, \Rightarrow \overline{\overline{e}}\right\}\right) = \left\{c \wedge \overline{c}, d, \overline{\overline{e}}\right\}$

From $res_{n-1}\,(\sigma) = \sigma$ immediately follows $BelBase(res_{n-1}\,(\sigma)) = BelBase(\sigma)$ and $getf\left(\lim_m info_m^{R_{n-1}}\,(\emptyset)\right) = getf\left(\lim_m info_m^{R}\,(\emptyset)\right)$. This means that it suffices to show that $BelBase(res_i\,(\sigma)) = getf\left(\lim_m info_m^{R_i}\,(\emptyset)\right)$, which we will do inductively.

**Proposition 6.23.** $BelBase(res_i\,(\sigma)) = getf\left(\lim_m info_m^{R_i}\,(\emptyset)\right)$.

*Proof.* We first show that $BelBase(res_0\,(\sigma)) = getf\left(\lim_m info_m^{R_0}\,(\emptyset)\right)$.

$res_0\,(\sigma) = (\varphi_n)$

(a) If $\varphi_n$ is contradictory:

- $BelBase(res_0\,(\sigma)) = \emptyset$, as $\bot_w \in Cn\,(\{\varphi_n\})$
- $getf\left(\lim_m info_m^{R_0}\,(\emptyset)\right) = \emptyset$, as $\Rightarrow \varphi_n$ is incompatible with $\emptyset$ and therefore ultimately defeated

(b) If $\varphi_n$ is not contradictory:

- $BelBase(res_0\,(\sigma)) = \{\varphi_n\}$
- $getf\left(\lim_m info_m^{R_0}\,(\emptyset)\right) = \{\varphi_n\}$ (as $\Rightarrow \varphi_n$ is the only premise for all possible arguments and is not contradictory)

From (a) and (b) follows $BelBase(res_0\,(\sigma)) = getf\left(\lim_m info_m^{R_0}\,(\emptyset)\right)$.

Assume $BelBase(res_i\,(\sigma)) = getf\left(\lim_m info_m^{R_i}\,(\emptyset)\right)$, we then show

$BelBase(res_{i+1}\,(\sigma)) = getf\left(\lim_m info_m^{R_{i+1}}\,(\emptyset)\right)$.

(a) $\bot_w \in Cn\left(\left\{\varphi_{n-(i+1)}\right\} \cup BelBase(res_i\,(\sigma))\right)$

- $BelBase(res_{i+1}\,(\sigma)) = BelBase(res_i\,(\sigma))$

- As $\Rightarrow \varphi_{n-(i+1)}$ has lower rank than all other defeasible rules, it cannot contribute to a defeat, but as it is incompatible with ultimately undefeated premises, it is ultimately defeated.
  So $getf\left(\lim_m info_m^{R_{i+1}}(\emptyset)\right) = getf\left(\lim_m info_m^{R_i}(\emptyset)\right)$.

(b) $\perp_w \notin Cn\left(\left\{\varphi_{n-(i+1)}\right\} \cup BelBase(res_i(\sigma))\right)$

- $BelBase(res_{i+1}(\sigma)) = BelBase(res_i(\sigma)) \cup \left\{\varphi_{n-(i+1)}\right\}$

- As $\Rightarrow \varphi_{n-(i+1)}$ is compatible with everything ultimately undefeated so far, it is not defeated and therefore ultimately undefeated as well.
  So $getf\left(\lim_m info_m^{R_{i+1}}(\emptyset)\right) = getf\left(\lim_m info_m^{R_i}(\emptyset)\right) \cup \left\{\varphi_{n-(i+1)}\right\}$.

From (a) and (b) follows $BelBase(res_{i+1}(\sigma)) = getf\left(\lim_m info_m^{R_{i+1}}(\emptyset)\right)$, which together with $BelBase(res_0(\sigma)) = getf\left(\lim_m info_m^{R_0}(\emptyset)\right)$ implies
$BelBase(\sigma) = getf\left(\lim_m info_m^{R}(\emptyset)\right)$. □

We have thereby shown that the identification of the base beliefs can be done within an argumentation framework. Belief revision would then be accomplished by adding another defeasible rule of the form $\Rightarrow \varphi$ with a strength greater than the previous ones. This result does not provide any insight in the inference mechanism, though.

# 7 Conclusion and future work

Motivated by the desire to model contraction within a particular belief revision framework, we have started to developed a full-blown logic of disbelief. That is, we allowed the nesting of the disbelief connective. In our logic, this came for the price of a third truth value and a quite complicated structure of models. We formulated desirable properties of the logic and especially of the behaviour of the disbelief connective.

The modification of the classical tableau approach allowed to achieve one of the major results of the paper – the proof of soundness and completeness of the logic developed. We showed that the language introduced might not be rich enough to arrive at an axiomatization of the logic. We proposed a further extension of the language and proved some results obtained in the attempt to find an axiomatization.

We showed that the deduction theorem does not hold. This implies that an axiomatization of equivalence and inference will not be a mere extension of the classical relations.

The implementation we developed during our research is briefly presented. As it directly implements the tableau approach, it allows to try out simple examples and to test propositions, which would be much more time consuming if the truth tables had to be developed by hand.

We proved that the belief revision framework can be embedded into an argumentation framework. The translation illustrates how the propositions in the sequence defeat one another during the calculation of the belief base.

Applied in the belief revision framework, our logic not only models contraction but allows greater expressiveness, e.g. with propositions like $\overline{\varphi} \rightarrow \psi$ formalizing that an express disbelief in one proposition implies belief in another.

### Remarks on "Some Logics of Belief and Disbelief" [2]

Although our notion of disbelief differs slightly from that implied by examples 2 and 3 in that paper, our framework is able to handle them as demanded there. The examples are basically expressing that it should be possible to believe in disjunction $\varphi_1 \vee \ldots \vee \varphi_n$, while disbelieving in each individual $\varphi_i$. This problem is known as the lottery paradox – while disbelieving that a particular ticket will win, it is believed that one will win. It is easily seen that the (weak) disjunction as defined for our logic does not allow that, but

the strong one which can be defined in terms of implication and negation[11] can meet this demand. In our opinion, what is called disbelief in those examples is merely a strong tendency towards negation.

Our logic does not have the property (D$\vee$) – if $\Gamma \vdash \overline{\phi}$ and $\Gamma \vdash \overline{\psi}$, then $\Gamma \vdash \overline{\phi \vee \psi}$ – satisfied by the logic GBD. This property was argued against in the paper, but we want to note that this property does not necessarily force disbelief in tautologies.

Like the authors of [2], we think that agnosticism should be possible, i.e. that $\{\overline{\varphi}, \overline{\neg\varphi}\}$ should be satisfiable. By (D$\vee$), now $\overline{\varphi \vee \neg\varphi}$ holds as well, i.e. disbelief in a tautology. But is $\varphi \vee \neg\varphi$ still a tautology? In a two-valued logic it surely is, but looking at it with a three-valued logic in mind... At least (D$\vee$) does not look as strange any more. It is obvious that a logic satisfying (D$\vee$) cannot solve the lottery paradox as the two express opposites. While the one demands that $\overline{\phi \vee \psi}$ is a consequence of $\{\overline{\phi}, \overline{\psi}\}$, the other demands $\{\overline{\phi}, \overline{\psi}, \phi \vee \psi\}$ to be consistent.

An intuition for the disbeliefs provided in [2] is that the beliefs are information acquired by the agent itself and disbeliefs are information from a different source about what does not hold. The logic BD was said to be pitched at the right level, but we find the restriction that

> the worlds that the sources of an agent may regard as impossible have to be worlds that the agent itself regards as possible

counterintuitive. It is not clear to us why not both agent and source should be allowed to regard a certain world as impossible.

The approaches in [2] have a quite simple structure. They separate beliefs and disbeliefs into different sets and connect them by a set of rules. As mentioned above, our approach does not distinguish between the two. The price for that is having to allow a third truth value and a more complex inference scheme. When trying to generalize the logics in [2] by allowing the nesting of the disbelief bar, the structure of sets will become more complicated and so will the rules necessary.

[11] $\varphi \vee_s \psi \stackrel{\text{def}}{=} \neg\varphi \rightarrow \psi$. The difference with respect to the weak disjunction is, that if both disjuncts are evaluated to $u$, their disjunction is evaluated to 1.

**Future work**

There is much potential for future work. It is worthwhile to investigate the relationship between our logic of disbelief and modal logic. There are modal logics that allow to express the state of an agent's knowledge. Maybe a similar approach works for beliefs and disbeliefs, i.e. modal operators could model the disbelief connective.

An investigation into the relationship between our disbelief approach and other existing approaches would be of interest. As the latter are restricted to propositions and the disbelief in those and do not allow a nesting of the disbelief bar up to now, it is the question whether they can be embedded in our approach. Further, it is open if and how they can be generalized to more complicated structures of disbelief.

It would be nice to find axiomatizations of the inference and equivalence relations, as well as for nonsatisfiable propositions and tautologies. This would make the picture of our logic, which is dominated by semantic characterization up to now, more complete.

It might be interesting to investigate fragments of the logic. Possible candidates are the restriction to classical propositions and disbeliefs in them, the restriction to classical propositions and disbeliefs of higher order, i.e. allowing only purley classical propositions or disbeliefs below a disbelief bar. Furthermore, certain connectives could be removed, e.g. allowing only implication and negation or not allowing implication.

We have not attached a particular interpretation to the third truth value $u$ in our logic other than its not being $true$ or $false$. Dunn/Belnap's 4-valued system provides a very appealing notion for the two extra values – no information and contradicting information. It might be interesting how to interpret a disbelief operator in this setting. In the light of the belief revision framework introduced, this does not make sense, as the state of contradicting information will not be arrived at, but for other applications this might be different. Again, the essential point will be how to define the interaction between a proposition and the disbelief in it.

The implementation could be optimized. For examples with a greater number of propositions, it shows weaknesses in running time and memory problems. Right now, branches are checked for clashes after every step of expansion. This is very expensive, though, because it involves equivalence checks that require tableau proofs themselves. An expansion step, on the other hand, is a simple procedure on lists. So doing the check for clashes less frequently could speed up the program considerably. Also, reorganizing the

program so that intermediate results which are not needed any more are thrown away to clear the stack could allow the program to handle more complicated examples.

### Weak version of the logic

In the logic we have investigated so far, a set of propositions was inconsistent if its closure under consequence forced an evaluation of one proposition to 1, as well as to $u$ or 0 and so on, i.e. to two different truth values. A version of the logic might be considered, where an inconsistency only occurs if a proposition is forced to take the values 1 and 0, so the notion of inconsistency would be weaker.

However, the result is not very appealing, as it would mean that $\varphi \wedge \overline{\varphi}$ would not be inconsistent since there would be an open branch with evaluations of $\varphi$ to 1 and $u$. This would defeat the purpose of the logic. Therefore, we believe a further investigation not to be fruitful.

# References

[1] Booth, R., *personal communication,* April-August 2002.

[2] Chopra, S., Heidema, J., Meyer,T., *Some logics of belief and disbelief,* in: Proceedings of the International Workshop on Non-Monotonic Reasoning, 2002.

[3] Foo, N., Pagnucco, M., *personal communication,* October & November 2002.

[4] Friedman, N., Halpern, J. Y., *A knowledge-based framework for belief change: Part II: Revision and update,* in: Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94), pp. 190–200, 1994.
http://citeseer.nj.nec.com/friedman94knowledgebased.html

[5] Gärdenfors, P., *Belief revision: An introduction,* in Belief Revision, pp. 1-20, Cambridge University Press, Gärdenfors (ed.).

[6] Gomolinska, A., Pearce, D., *Disbelief Change* in: Electronic Essays on the occasion of the fiftieth birthday of Peter Gärdenfors, 2001

[7] Gottwald, S., *Many-Valued Logic,* The Stanford Encyclopedia of Philosophy (Winter 2001 Edition), Edward N. Zalta (ed.).
http://plato.stanford.edu/archives/win2001/entries/logic-manyvalued/

[8] Gottwald, S., *Mehrwertige Logik: Eine Einführung in Theorie und Anwendungen,* Akademie-Verlag Berlin, 1989.

[9] Nerode, A., Shore, R.A., *Logic for Applications,* 2nd Edition, Springer-Verlag New York, 1997.

[10] Rose, K. H., *XY-Pic User's Guide,* Version 3.7.

[11] Vreeswijk, G.A.W., *Abstract Argumentation Systems,* in: Artificial Intelligence, Vol. 90, pp. 225-279.

[12] Weinelt, J., *Der Latex-Index: Eine Befehlsübersicht im Internet.*
http://www.weinelt.de/latex/index.html

[13] Zhang, D., *personal communication,* January 2003.

# List of Figures

# List of Tables

# A Notes on notation

- $\neg$, $\vee$, $\wedge$, $\rightarrow$ connectives negation, disjunction, conjunction, implication. Their meaning depends on the logic used and should be clear from the context.
- $\rightarrow$ in Section 6 to denote strict rules, see Definition 6.3
- $\overline{\varphi}$ connective bar to denote disbelief in a proposition $\varphi$
- $\equiv$ equivalence, a subscript may indicate the underlying logic
- $\bot$ bottom, contradiction, proposition evaluated to 0 under all valuations
- $\bot_w$ inconsistency, not evaluated to 1 by any valuation, in classical propositional logic the same as $\bot$, but not so in the logic of disbelief
- $\models$ consequence relation between a set of propositions and a proposition, a subscript may indicate the underlying logic
- $\vdash$ entailment relation between a set of propositions and a proposition, a subscript may indicate the underlying logic or method
- $\emptyset$ empty set
- * operator for revision of an epistemic state by new information
- -,+ as superscripts of a proposition denote its nonsatisfiability / the nonsatisfiability of its negation
- $\sigma$ sequence
- $\varphi$, $\phi$, $\psi$ propositions, sentences of propositional logic or logic of disbelief, depending on the context
- $\Sigma$ a set, usually of propositions
- 0, $u$, 1 truth values: false, undefined, true
- $a$, $b$, ... propositional letters; in Section 6 also to denote arguments
- $A$ truth assignment function, see Definition 2.4
- $A_{L_i'}$, $A'_{L_i'}$ truth assignment function for language fragments, see Section 5.1

- bar see $\overline{\varphi}$
- $Bel$ Belief set, see Section 1.2
- $CL(\Sigma)$ syntactic closure under classical connectives, see Section 5.1
- $Cn(\bullet)$ closure under consequence, a subscript may indicate the underlying logic
- $E$ entry in a tableau
- $\lim_i X$ the limes of $X$ for $i$ approaching infinity
- $L$ language (without subscript: language of disbelief, see Definition 2.1; otherwise as indicated in the text)
- $L_i$, $L_i'$ language fragments, see Section 5.1
- $P$ in Section 3 (possibly with a subscript) to denote a path/branch in a tableau; in Section 6 used to denote a subset of a language
- $T$ (possibly with a subscript) tableau
- $v$ (possibly with a subscript) denotes a truth value (in the context of tableaux also $<1$ or $>0$, see Section 3.1)
- $V$ truth valuation function, see Definition 2.4, a subscript may indicate the truth assignment function that gives rise to it

# B Soundness of the inference rules

By using truth tables and the definition of truth valuations for the extended language, it can be easily seen that the rules from Section 5.5 are sensible. The remarks below show that they are in correspondence with the tableau introduced in Section 3, as well[12].

- $\varphi \vdash_{\pm} \varphi$ trivially, on every open path where $\varphi : 1$ holds, $\varphi : 1$ holds
- $\varphi \wedge \varphi^{-} \vdash_{\pm} \perp_{w}$: no open branch with $\varphi : 1$ and $\varphi :< 1$
- $(\neg\varphi)^{+} \vdash_{\pm} \varphi^{-}$: in an open branch $(\neg\varphi)^{+} : 1$ if $\neg\varphi : 1$ or $\neg\varphi : u$ , so $\varphi : 0$ or $\varphi : u$, i.e. $\varphi^{-} : 1$ holds
- $(\neg\varphi)^{-} \vdash_{\pm} \varphi^{+}$: analogous
- $\neg(\varphi^{-}) \vdash_{\pm} \varphi$: $\neg(\varphi^{-}) : 1$, if $\varphi^{-} : 0$ if and only if $\varphi : 1$
- $\neg(\varphi^{+}) \vdash_{\pm} \neg\varphi$: analogous
- $\varphi^{-} \vdash_{\pm} (\varphi \wedge \psi)^{-}$: if $\varphi^{-} : 1$, (t6) will not have an open branch
- $\varphi \vee \psi \vdash_{\pm} \varphi^{-} \rightarrow \psi$: by (t9) in any open branch where not $\varphi : 1$, $\psi : 1$ holds
- $\varphi \vee \psi \vdash_{\pm} \psi^{-} \rightarrow \varphi$: analogous
- $\varphi^{-} \rightarrow \psi \vdash_{\pm} \varphi \vee \psi$: using the definition of a truth assignment for an open branch, if there is no entry with $\varphi$, it is assigned $u$, so $\varphi^{-}$ is assigned 1, i.e. the implication is true. Only if $\psi$ is assigned 1, so that there has to be an entry $\psi : 1$ on the path, then the disjunction is assigned 1 as well. If $\varphi : 1$ is an entry on the path, the disjunction holds, too, and otherwise, the implication holds only if $\psi$ is evaluated to true satisfying the disjunction.
- $\psi^{-} \rightarrow \varphi \vdash_{\pm} \varphi \vee \psi$: analogous
- $(\varphi \vee \psi)^{+} \wedge \neg\varphi \vdash_{\pm} \psi^{+}$: (t11) excluded, by (t9) - left branch excluded, in right one $\psi^{+}$ holds, by (t10) - left branch excluded, in right one $\psi^{+}$ holds
- $(\varphi \vee \psi)^{+} \wedge \neg\psi \vdash_{\pm} \varphi^{+}$: analogous

[12] In connection with tableaux, $\varphi \vdash_{\pm} \psi$ reads: in any open tableau branch where $\varphi$ holds, $\psi$ holds as well. $\varphi^{+}$holds if and only if $\varphi : 1$ or $\varphi : u$; a branch in which $\varphi^{+}$ holds can be open only if $\neg\varphi : 1$ is not an entry on that path, analogous for $\varphi^{-}$.

- $(\varphi \vee \psi)^+ \vdash_\pm \varphi^+ \vee \psi^+$: (t11) excluded, by (t9),(t10), one of $\varphi^+$ and $\psi^+$ holds in every branch, so $\varphi^+ \vee \psi^+$ holds in every one
- $\varphi^+ \vee \psi^+ \vdash_\pm (\varphi \vee \psi)^+$: analogous
- $(\varphi \vee \psi)^- \vdash_\pm \varphi^-$: (t9) excluded, by (t10), (t11), (t1) $\varphi^-$ holds in every branch
- $(\varphi \vee \psi)^- \vdash_\pm \psi^-$: analogous
- $(\varphi^- \vee \psi^-) \wedge \varphi \vdash_\pm \psi^-$: in an open branch $\varphi$ and $\varphi^-$ cannot both hold, so $\psi^-$ has to be true if the disjunction is true
- $(\varphi^- \vee \psi^-) \wedge \psi \vdash_\pm \varphi^-$: analogous
- $\varphi^+ \wedge \psi^+ \vdash_\pm (\varphi \wedge \psi)^+$: obvious
- $\varphi^- \wedge \psi^- \vdash_\pm (\varphi \vee \psi)^-$: likewise
- $\varphi \to \psi \vdash_\pm \psi^- \to \varphi^-$: by (t12), it is easily checked that the implication holds in every branch
- $\varphi \to \psi \vdash_\pm \varphi^+ \to \psi^+$: likewise
- $(\varphi \to \psi)^+ \wedge \varphi \vdash_\pm \psi^+$: (t14) excluded, by (t13) - right branch excluded, in left one $\psi^+$ holds, by (t12) - left and right branch excluded, in middle one $\psi^+$ holds
- $(\varphi \to \psi)^+ \wedge \neg\psi \vdash_\pm \varphi^-$: (t14) excluded, by (t13) - left branch excluded, in right one $\varphi^-$ holds, by (t12) - middle and right branch excluded, in left one $\varphi^-$ holds
- $(\varphi \to \psi)^- \vdash_\pm \psi^-$: (t12) excluded, by (t13), (t14) $\psi^-$ in all open branches
- $(\varphi \to \psi)^- \vdash_\pm \varphi^+$: (t12) excluded, by (t13), (t14) $\varphi^+$ in all open branches
- $(\varphi \to \psi)^- \vdash_\pm \varphi^- \to \neg\psi$: (t12) excluded, by (t13) and (t14) implication holds in every branch
- $\neg(\varphi \to \psi) \vdash_\pm \varphi$: by (t3),(t14)
- $\neg(\varphi \to \psi) \vdash_\pm \neg\psi$: by (t3),(t14)

# C Examples for the tableau implementation

We present some simple examples to illustrate how the tableau implementation of the logic can be used. In some cases, we only give a sample predicate to show its usage. In others, we reproduce the output of the program, as well.

$dr(X)$ is the abbreviated form of the predicate $dorevision(X, R)$. Basically, its argument is the sequence from which the belief base and then a partial belief set is calculated.

- The first example is particularily simple. Here the sequence is $\sigma = (a)$.

  dr([a]). The output of the program is:

  ```
  The base belief set is: [a]
  preparing the base set
  starting inference
  [polishing the belief set, [a]]
  [a]
  is the belief set created by the base belief set
  [a]
  ```

  The second, third and fourth line are only status information telling the user what the program is currently doing. Information of this kind is notinteresting now, so we will omit it, in the following.

- The second example is more interesting. The corresponding sequence is $\sigma = (a, \neg a)$.

  dr([a, nott(a)]). Output:

  ```
  Integration of a leads to inconsistency.
  The base belief set is: [nott(a)]
  [nott(a)]
  is the belief set created by the base belief set
  [nott(a)]
  ```

  The first line tells the user that $a$ could not be integrated into the belief base as this would have introduced an inconsistency. But like in the first example the belief set is not particularily interesting.

- dr([dis([a, b]), bar(b)]), i.e. $\sigma = \left(a \vee b, \overline{b}\right)$. Output:

```
The base belief set is: [dis([a, b]), bar(b)]
[dis([a, b]), a, bar(b)]
is the belief set created by the base belief set
[dis([a, b]), bar(b)]
```

  Here the belief set is more interesting. As mentioned in Section 4.3, the propositions of the belief base are expanded to all its subpropositions which are then tested. Here $a$ is found to be a consequence of the belief base.

- dr([dis([a, b, c, d]), dis([a, b, con([nott(c), nott(d)])])]), i.e.
  $\sigma = (a \vee b \vee c \vee d, a \vee b\,(\neg c \wedge \neg d))$. Output:

```
The base belief set is: [dis([a, b, c, d]),
 dis([a, b, con([nott(c), nott(d)])])]
[dis([a, b, c, d]), dis([a, b, c]), dis([a, b, d]),
 dis([a, b, con([nott(c), nott(d)])]), dis([a, b])]
is the belief set created by the base belief set
[dis([a, b, c, d]), dis([a, b, con([nott(c), nott(d)])])]
```

  This example illustrates one of the possibilities of improving the program. $a \vee b \vee c$ and $a \vee b \vee d$ are mentioned in the belief set although it is more than clear that they hold, because $a \vee b$ is contained as well. Leaving propositions like that out of the belief base makes the output more readable – especially if the belief sets are bigger.

- dr([a, imp(a, dis([b, c])), bar(b), bar(c)]), i.e. $\sigma = \left(a, a \to \vee c, \overline{b}, \overline{c}\right)$. Output:

```
Integration of a leads to inconsistency.
```

- dr([a, bar(a), bar(nott(a)), bar(bar(a)), bar(bar(nott(a))), imp(a, bar(nott(a)))]), i.e. $\sigma = \left(a, \overline{a}, \overline{\neg a}, \overline{\overline{a}}, \overline{\overline{\neg a}}, a \to \overline{\neg a}\right)$. Output:

```
Integration of bar(nott(a)) leads to inconsistency.
Integration of bar(a) leads to inconsistency.
Integration of a leads to inconsistency.
The base belief set is: [bar(bar(a)), bar(bar(nott(a))),
 imp(a, bar(nott(a)))]
```

Apart from the calculation of the belief base and partial belief set that are induced by a given sequence, the tableau implementation can be used to check whether propositions are tautologies, equivalent, satisfiable, etc.

*prove*$(X)$ checks whether a proposition $X$ is a tautology.

- Is $(a \to b) \to ((b \to c) \to (a \to c))$ a tautology?

  prove(imp(imp(a, b),imp(imp(b, c), imp(a, c)))). The answer is

  ```
  Yes
  ```

- prove(imp(a, b), show). Output:

  ```
  [fin, open, [a, 1], [b, u], [imp(a, b), l], [imp(a, b), u]]
  No
  ```

  Of course, $a \to b$ is not a tautology. The optional second argument *show* forces the output of an open branch, i.e. of a noncontradictory assignment that evaluates the proposition to a value other than 1. Entries of a branch can be *open* or *fin* to indicate that the branch is open and finished or entries of the form $[\varphi, value]$. The value $l$ means *less than 1*, i.e. it corresponds to $< 1$ from the definition of atomic tableaux.

- An example from Section 5.4: $((\varphi \wedge \psi) \to \chi) \to (\varphi \to (\psi \to \chi))$ is a tautology. prove(imp(imp(con(a, b), c), imp(a, imp(b, c)))).

  But the reverse implication $(\varphi \to (\psi \to \chi)) \to ((\varphi \wedge \psi) \to \chi)$?

  prove(imp(imp(a, imp(b, c)), imp(con(a, b), c)), show). Output:

  ```
  [fin, open, [a, u], [b, u], [c, 0], [con(a, b), u],
   [imp(a, imp(b, c)), 1], [imp(b, c), u], [imp(con(a, b), c), u],
   [imp(imp(a, imp(b, c)), imp(con(a, b), c)),1],
   [imp(imp(a, imp(b, c)), imp(con(a, b), c)), u]]
  No
  ```

*eq*$(X, Y)$ checks the equivalence of two propositions $X$ and $Y$.

- eq(a, a).

- eq(a, nott(nott(a))).
- $\neg\neg a \equiv \neg\,(a \rightarrow \neg\,(b \rightarrow (c \rightarrow b)))$?

  eq(nott(nott(a)), nott(imp(a, nott(imp(b, imp(c, b)))))). Output:

  ```
  Yes
  ```

The predicate $satisfiable(X)$ tests whether the proposition $X$as a model.

- satisfiable(con(a, bar(a))).
- satisfiable(con(a, b), show). Output:

  ```
  [fin, open, [a, 1], [b, 1], [con(a, b), 1]]
  Yes
  ```

  Again, an optional second argument *show* causes the program to output an open branch – if there is one – representing a possible assignment.

- Does $(a \rightarrow (b \rightarrow a)) \rightarrow (a \wedge \overline{a})$ have a model?

  satisfiable(imp(imp(a, imp(b, a)), con(a, bar(a)))). Output:

  ```
  No
  ```

The last example illustrates a weakness of the simple inference approach of just checking the subpropositions. $createbeliefset([con(dis(a,b), dis(c, bar(b)))])$. triggers the construction of the belief set of $(a \vee b) \wedge \left(c \vee \overline{b}\right)$. The result given is

```
[con([dis([a, b]), dis([c, bar(b)])]), dis([a, b]), dis([c, bar(b)])]
```

However, it should be clear that this is an example of a general case of resolution. As not both $b$ and $\overline{b}$ can be true, the disjunction $a \vee c$ holds as well. Explicitly checking that by using the predicate $isconsequenceof(X, Y)$, where $X$ is a proposition and $Y$ a list of propositions, confirms this.

isconsequenceof(dis(a, c), [con(dis(a, b), dis(c, bar(b)))]).

But as $a \vee c$ is not among the subpropositions, it is not tested by the inference engine.