Quasi-Realization

Melvin Fitting
e-mail: melvin.fitting@gmail.com
web page: melvinfitting.org

City University of New York, emeritus

Abstract. Justification logics connect with modal logics via *Realization Theorems*. The first such theorem was proved constructively by Artemov, [1]. It showed how to translate an S4 sequent proof, as a whole, into an LP proof. We present a different algorithmic Realization proof for LP/S4, proceeding step by step instead of working on the entire proof, dividing the argument into two natural parts, one specific to LP/S4, the other widely applicable to justification/modal pairs. This structure makes an implementation easier, and we provide a link to one in Prolog.

Keywords: justification logic, modal logic, realization, quasi-realization, tableau, Prolog

1 Introduction

Justification logics are similar to modal logics, but with modal operators replaced by an infinite family of *justifications*. The first justification logic, LP (logic of proofs), was introduced by Artemov [1]. It played an essential role in Artemov's arithmetic completeness result for intuitionistic logic, finishing a line of research that began with Gödel, [16]. A step in that work shows that LP has a direct connection with modal S4, via a *Realization Theorem*. This says that every S4 theorem has a Realization, a replacement of modal operators by justification terms, that is a theorem of LP. In a sense, a Realization represents the flow of information hidden in the modal operators. One sometimes sees references to S4 as a logic of *implicit* knowledge, while LP *explicitly* represents that knowledge.

Since Artemov's work many justification logics have been created, and many proofs of Realization have been developed. It is now known that the family of modal/justification pairs is infinite, and much work has gone into the investigation of justification counterparts of familiar of modal logics such as K, T, K4, S5, and so on. See [2,12] for a discussion of the family of justification logics. Recently quantification has been added, but this is another story, see [3,6].

The first proof of a Realization theorem can be found in [1]. It is constructive. Constructive proofs commonly use cut-free Gentzen sequent systems but prefixed tableaus/nested sequents have also been used, providing a modular approach applicable to a basic family of modal logics, [17]. There are non-constructive proofs based on semantics, [11]. Recently there is a non-constructive proof using the model existence theorem, [15].

Here we give a new constructive proof of Realization. We present it for LP and S4, but the argument is clearly more general. We use semantic tableaus rather than a sequent calculus, but there is a well-known connection between them. The central fact is that our Realization algorithm proceeds step by step, rather than working with a tableau proof as a whole, and this makes implementation easier. A link to a Prolog impmentation can be found at [8]. In addition, our algorithm divides into two parts. First a *Quasi-Realization Theorem* is shown. This extracts a Quasi-Realization from a modal tableau proof—a simpler thing to do than getting a Realization proper. This part depends on details of S4, and needs modification for other modal logics. Algorithmic conversion from Quasi-Realization to Realization is independent of the particular logic, and can be found in [7]. This part is omitted here, because of space limitations.

2 The Logic LP

This section contains a brief formulation of LP axiomatically, which comes from [1]. A semantics will not be needed in this paper.

Justification terms are built up from justification variables, v_1, v_2, ..., and justification constants, c_1, c_2, ..., using the function symbols $\cdot$, $+$, and !. If t is a justification term, so is $!t$, and if t and u are justification terms, so are $(t + u)$ and $(t \cdot u)$. (We may omit some parenthesis when no harm is done.) *Ground* justification terms are those without variables.

Formulas are built up from propositional variables, P, Q, ..., and the propositional constant $\bot$ using $\supset$ (with other connectives defined in the usual way), and an extra rule of formation: if t is a justification term and X is a formula then $t{:}X$ is a formula.

The formula $t{:}X$ can be read: "t is a justification of X." Justification constants represent justifications of basic, assumed truths—axioms. Justification variables are thought of as implicitly universally quantified over justifications. If t is a justification of $X \supset Y$ and u is a justification of X, think of $t \cdot u$ as a justification of Y. The operation ! is a checker: if t is a justification of X then $!t$ is a verification that t is such a justification. The operation $+$ combines justifications, in that $t + u$ justifies all the things that t justifies plus all the things that u justifies.

The following axiom system for LP is from [1]. Axioms are specified by giving axiom schemas and rules, and these are:

A0.	**Classical**	Enough classical propositional axiom schemes
A1.	**Application**	$t{:}(X \supset Y) \supset (s{:}X \supset (t{\cdot}s){:}Y)$
A2.	**Factivity**	$t{:}X \supset X$
A3.	**Justification Checker**	$t{:}X \supset !t{:}(t{:}X)$
A4.	**Weakening**	$s{:}X \supset (s{+}t){:}X$
		$t{:}X \supset (s{+}t){:}X$
R1.	**Modus Ponens**	$\vdash Y$ provided $\vdash X$ and $\vdash X \supset Y$
R2.	**Axiom Necessitation**	$\vdash c{:}X$ where X is an axiom $A0 - A4$ and c is a justification constant.

A proof is a finite sequence of formulas each of which is an axiom or comes from earlier terms by one of the rules of inference. *Derivations* can be introduced either directly, or indirectly by defining $\Gamma \vdash_{\mathsf{LP}} X$ to mean that $(G_1 \wedge \ldots \wedge G_n) \supset X$ is a theorem for some finite subset $\{G_1, \ldots, G_n\}$ of Γ.

Which constants are associated with which axioms for rule *R2* applications is called a *constant specification*. More formally, a constant specification is a set $\mathcal{C}$ whose members are of the form $c{:}A$ where c is a justification constant and A is an axiom. A proof uses constant specification $\mathcal{C}$ if each instance of Axiom Necessitation is in $\mathcal{C}$. A constant specification can be given ahead of time, or created during the course of a proof. We will assume all constant specifications are *axiomatically appropriate*: every axiom is assigned at least one constant. In addition, all such assignments will be *injective*, no justification constant is used for more than one axiom. Many other conditions have been investigated, but we are not interested in constant specification details here.

If Z is any theorem of LP, and we replace every proof polynomial by $\Box$ (the *forgetful* projection), the result is a theorem of $\mathsf{S4}$. This is easy to see: it is the case for each axiom of LP, and is preserved by the LP rules of derivation. The Artemov Realization Theorem, from [1], is a converse to this.

Theorem 1 (Realization Theorem). *If Z is a theorem of $\mathsf{S4}$, there is some replacement of $\Box$ symbols with justification terms to produce a theorem of LP, provable using an injective, axiomatically appropriate constant specification). This can be done so that negative occurrences of $\Box$ in Z are replaced with distinct justification variables, and positive occurrences by justification terms that may involve those variables.*

Negative occurrences of justification variables can be thought of as inputs, and positive justification terms as outputs. Thus theorems of $\mathsf{S4}$ carry implicit constructive functional content which their LP Realizations make explicit.

A fundamental result is the Lifting Lemma, from [1,2], which says that proofs and derivations in LP can be internalized.

Theorem 2 (Lifting Lemma). *Assume we have an axiomatically appropriate constant specification. Suppose $s_1{:}X_1, \ldots, s_n{:}X_n, Y_1, \ldots, Y_k \vdash_{\mathsf{LP}} Z$. Then there is a justification term $t(s_1, \ldots, s_n, y_1, \ldots, y_k)$ (where the y_i are justification variables) such that $s_1{:}X_1, \ldots, s_n{:}X_n, y_1{:}Y_1, \ldots, y_k{:}Y_k \vdash_{\mathsf{LP}} t(s_1, \ldots, s_n, y_1, \ldots, y_k){:}Z$.*

Corollary 1. *With an axiomatically appropriate constant specification, if Z has an LP proof, then for some ground proof polynomial t, $t{:}Z$ will have an LP proof.*

3 Tableaus

Tableaus are refutation proof systems. Informally, one assumes a formula X could be false under some circumstances and derives a syntactic contradiction. Classical formulas are built up from propositional letters and $\bot$ using $\wedge$, $\vee$, $\supset$, and $\neg$, though other binary connectives could also be admitted. Smullyan's *uniform notation* is useful here, [18,19], both for theoretical purposes and to simplify

tableau implementations. We use *signed* formulas. Two special symbols, T and F, are introduced and $T\,X$ and $F\,X$ are signed formulas if X is a formula. The intended reading is that X is true, or false respectively. Signed formulas involving binary connectives divide into α cases, conjunctive, and β cases, disjunctive. For each case, two components are also specified. This is given in Figure 1.

Conjunctive			Disjunctive		
α	α_1	α_2	β	β_1	β_2
$T\,X \wedge Y$	$T\,X$	$T\,Y$	$F\,X \wedge Y$	$F\,X$	$F\,Y$
$F\,X \vee Y$	$F\,X$	$F\,Y$	$T\,X \vee Y$	$T\,X$	$T\,Y$
$F\,X \supset Y$	$T\,X$	$F\,Y$	$T\,X \supset Y$	$F\,X$	$T\,Y$

Fig. 1. α- and β-Formulas and Components

A tableau proof is a special labeled binary tree. A proof of X begins with a tree having only a root node, labeled $F\,X$. Then a tree is 'grown' using the *branch extension* rules, given in Figure 2. All trees produced this way are tableaus.

$$\frac{T\,\neg X}{F\,X} \qquad \frac{F\,\neg X}{T\,X} \qquad \begin{array}{c}\alpha\\ \hline \alpha_1\\ \alpha_2\end{array} \qquad \begin{array}{c}\beta\\ \hline \beta_1 \mid \beta_2\end{array}$$

Fig. 2. Classical Branch Extension Rules

Tableaus are displayed as downward branching trees. Think of a tree as representing the disjunction of its branches, and a branch as representing the conjunction of the signed formulas on it. The members of a tableau branch can be thought of as constituting a *set*, or a *multi-set*, or even a *sequence*. We treat branches as sets. Tableau rules are *non-deterministic*. At each stage we choose a signed formula occurrence on a branch and apply a rule to it. Since the order of choice is arbitrary, there can be many tableaus for a single signed formula. A tableau branch is *closed* if it contains $T\,A$ and $F\,A$ for some formula A, or if it contains $T\,\bot$. If each branch is closed, the tableau is *closed*. A closed tableau for $F\,X$ is a tableau proof of X. A branch is *atomically* closed if it contains $T\,P$ and $F\,P$ where P is atomic. We require atomic closure, which still gives us completeness. Classical branch extension rules can be restricted to *single use*: a classical tableau rule is never applied to a signed formula occurrence on a branch more than once. (This does not work for all logics, however.)

An example of a classical tableau proof is given in Figure 3. Numbers are for reference purposes only. In it, 2 and 3 are from 1 by α; 4 and 5 are from 3 by α; 6 and 7 are from 5 by α; 8 and 9 are from 2 by β. 10 and 11 are from 4 by β.

Reading from left to right, the branches are closed because of 8 and 10, 7 and 11, and 6 and 9. Notice that on one of the branches closure is on a non-atomic formula. This branch can be continued to yield atomic closure.

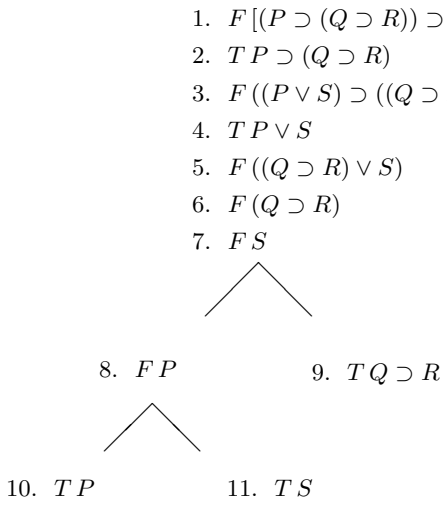

Fig. 3. Classical Proof of $(P \supset (Q \supset R)) \supset ((P \vee S) \supset ((Q \supset R) \vee S))$

Justification logics make little use of possibility, so there is no advantage now to modal uniform notation, and we do not use it. Syntactically $\Box$, but not $\Diamond$, is added to the classical language. We present what are called *destructive* tableaus. The name comes from the fact that certain modal rules cause branch information to disappear. Such tableaus exist for K, T, D, $\mathsf{D4}$, $\mathsf{K4}$, $\mathsf{S4}$, among others, but not for $\mathsf{S5}$. Here we only give rules for $\mathsf{S4}$.

Definition 1. *Let S be a set of signed formulas. $S^\sharp = \{T\,\Box X \mid T\,\Box X \in S\}$.*

The destructive tableau rules for $\mathsf{S4}$ are the classical tableau rules together with those in Figure 4. The first rule embodies reflexivity in an obvious way. The second rule is different, and is destructive, indicated by the double line. Suppose we have a branch containing $F\,\Box X$, with S being the set of other formulas on the branch. *The entire branch can be replaced with a new branch consisting of the members of $S^\sharp$, and $F\,X$.* Note that information is lost passing from S to $S^\sharp$, hence the name *destructive*.

$$\frac{T\,\Box X}{T\,X} \qquad \frac{S, F\,\Box X}{\overline{\overline{S^\sharp, F\,X}}}$$

Fig. 4. $\mathsf{S4}$ Branch Extension Rules

With classical propositional tableaus, *any* order of rule application is acceptable. This is not the case for S4. If both $F\,\Box X$ and $F\,\Box Y$ are present, applying a rule to one eliminates the other, and it may be that only one of the two possibilities will lead to a proof. Now *backtracking* becomes critical to proof search.

Figure 5 shows a proof, using the S4 rules, of $\Box X \supset \Box(\Box X \vee Y)$. We have indicated branch replacement witn horizontal lines. Lines 2 and 3 are from 1 by α. Next an S4 modal rule is applied to $F\,\Box(\Box X \vee Y)$, adding 4 while replacing S by $S^\sharp$ eliminates 1 and 3. Now an α-rule application to 4 adds 5 and 6, and produces a closed tableau, though not an atomically closed one. Continuing, we apply an S4 modal rule again, to 5, adding 7 while eliminating 4, 5, and 6. Applying the second S4 modal rule to 2 adds 8, and we have atomic closure.

1. $F\,\Box X \supset \Box(\Box X \vee Y)$
2. $T\,\Box X$
3. $F\,\Box(\Box X \vee Y)$

---

2. $T\,\Box X$
4. $F\,\Box X \vee Y$
5. $F\,\Box X$
6. $F\,Y$

---

2. $T\,\Box X$
7. $F\,X$
8. $T\,X$

Fig. 5. S4 proof of $\Box X \supset \Box(\Box X \vee Y)$

The rule $S, F\,\Box X \Rightarrow S^\sharp, F\,X$ is automatically single usage since applying it with $F\,\Box X$ eliminates the formula. The rule $T\,\Box X \Rightarrow T\,X$ is trickier. For S4, if $T\,\Box X \Rightarrow T\,X$ is applied to a signed formula occurrence it need not be applied again, *until the rule* $S, F\,\Box X \Rightarrow S^\sharp, F\,X$ *has been applied*. The intuition is simple: the destructive rule might eliminate the conclusion of $T\,\Box X \Rightarrow T\,X$ but for S4 it will not eliminate the premise, so a new application may be useful.

4 Annotated Formulas and Tableaus

Mapping modal formulas to formulas of justification logic requires that we keep track of the *occurrences* of $\Box$. In [5] we introduced *annotated formulas* for this; we use a simpler version here.

Definition 2. *An* annotated modal formula *is like a standard modal formula, except that instead of a single modal operator* $\Box$ *there is an infinite family,* $\Box_1$, $\Box_2$, *...,* *of* indexed *modal operators. In an annotated formula,* no index may occur twice.

If A *is an annotated formula and* A' *is the result of replacing all indexed modal operators,* $\Box_n$, *with* $\Box$, *regardless of index, we say* A *is an* annotated version *of* A', *and* A' *is an* unannotated version *of* A.

Annotations are purely for bookkeeping purposes. The α/β classification is exactly as with unannotated formulas, as is the definition of components. For instance, $T\,\Box_1 P \wedge \Box_2 Q$ counts as an α, with $\alpha_1 = T\,\Box_1 P$ and $\alpha_2 = T\,\Box_2 Q$. In tableau constructions, branch extension rules apply to annotated formulas exactly as to unannotated ones. The annotated version of the $\sharp$ operation is $S^\sharp = \{T\,\Box_i X \mid T\,\Box_i X \in S\}$. Since we are requiring atomic closure, closure conditions are not affected by annotations.

Figure 6 is an annotated version of the proof shown in Figure 5. Every S4 tableau proof can be turned into an annotated proof by annotating the modal operators appearing in the root, and then propagating these annotations downward through the tree.

1. $F\,\Box_1 X \supset \Box_2(\Box_3 X \vee Y)$
2. $T\,\Box_1 X$
3. $F\,\Box_2(\Box_3 X \vee Y)$

---

2. $T\,\Box_1 X$
4. $F\,\Box_3 X \vee Y$
5. $F\,\Box_3 X$
6. $F\,Y$

---

2. $T\,\Box_1 X$
7. $F\,X$
8. $T\,X$

Fig. 6. Annotated S4 proof of $\Box_1 X \supset \Box_2(\Box_3 X \vee Y)$

5 Changing the Tableau Representation

So far tableaus have been trees, and formula occurrences could be common to multiple branches. While this has advantages for some purposes, it does not when our Quasi-Realization algorithm is introduced. We will be associating a set of Quasi-Realizers with each signed formula occurrence in a tableau. How that is done depends on the history of the branch containing a given occurrence. If an occurrence is common to more than one branch, it is part of more than one history and things become ambiguous. Our solution is to change the way tableaus are represented, something that also brings us much closer to the data structure used in our Prolog implementation.

From now on a tableau is not a tree, but instead it is the set of its branches, where each branch is the set of signed formulas on it. When we write a branch as $\mathcal{B}, Z$, or more graphically $\overset{\mathcal{B}}{Z}$, we mean it is the set whose members are those of $\mathcal{B}$, together with signed formula Z. *This notation assumes that Z is not part of $\mathcal{B}$.* We reformulate the S4 tableau rules in this style, building in the notion of single-usage for tableau rules. Here are the formal details, which apply equally well to tableaus of signed formulas or of annotated signed formulas.

Definition 3 (Classical Tableau Revised). *A classical tableau is a finite set of finite sets (called branches) of signed formulas. A branch is closed if $T\,P$ and $F\,P$ are members for some atomic P, or if $T\,\bot$ is a member. A tableau is closed if each of its branches is closed. We say a signed formula is on a branch if it is a member of it, and a branch is in a tableau if it is a member of it. A* tableau proof *of X is a sequence of tableaus, beginning with a single branch tableau where that branch contains only $F\,X$, continuing using the Branch Extension Rules given in Figures 7 and 8, and ending with a closed tableau.*

$$\frac{\mathcal{B}\quad T\,\neg X}{\mathcal{B}\quad F\,X} \qquad \frac{\mathcal{B}\quad F\,\neg X}{\mathcal{B}\quad T\,X} \qquad \frac{\mathcal{B}\quad \alpha}{\mathcal{B}\quad \alpha_1\quad \alpha_2} \qquad \frac{\mathcal{B}\quad \beta}{\mathcal{B},\beta_1 \mid \mathcal{B},\beta_2}$$

Fig. 7. Classical Branch Extension Rules Revised

As an example, the β rule in Figure 7 is to be read as follows. If a tableau has $\mathcal{B},\beta$ as a branch, then the result of removing the branch from the tableau and replacing it with two branches, $\mathcal{B},\beta_1$ and $\mathcal{B},\beta_2$ is another tableau, which we call a *successor* of the original tableau. Similarly for the other rules. Note that the new branches do not contain β, but have β_1 and β_2 instead. This is our general strategy for enforcing single usage. That branches do not share any common parts is essential here.

There is one misleading aspect to the notation above. In the rule for $T\,\neg$ for instance, it may happen that $F\,X$ already occurs in $\mathcal{B}$, in which case the display of $\mathcal{B}, F\,X$ below the line is not correct—it should be simply $\mathcal{B}$. We allow this mild abuse, rather than complicating notation.

$$\frac{\mathcal{B}\quad T\,\Box X}{\mathcal{B}\quad \cancel{T\,\Box X}\quad T\,X} \qquad \frac{\mathcal{B}\quad F\,\Box X}{\mathcal{B}^\sharp\quad F\,X}$$

where $\mathcal{B}^\sharp = \{T\,\Box X \mid T\,\Box X \in \mathcal{B} \text{ or } \cancel{T\,\Box X} \in \mathcal{B}\}$

Fig. 8. S4 Modal Branch Extension Rules Revised

Single usage is trickier for modal rules. As noted earlier, single usage for the $F\,\Box$ rule is automatic, but for the $T\,\Box$ rule of S4 single usage only applies until the next application of the $F\,\Box$ rule. We build this into Figure 8 by crossing off an occurrence of $T\,\Box X$ when a rule has been applied to it, and providing no rule

that has a crossed off signed formula as a trigger. A cross off mark is removed, as part of the definition of $\mathcal{B}^\sharp$, when an $F\,\Box$ rule is applied.

Figure 9 shows a revised tableau proof of $\Box(P \supset Q) \supset (\Box P \supset \Box Q)$. It is not the shortest, by the way.

1. $\{\{F\,\Box(P \supset Q) \supset (\Box P \supset \Box Q)\}\}$
2. $\{\{T\,\Box(P \supset Q), F\,\Box P \supset \Box Q\}\}$
3. $\{\{\underline{T\,\Box(\cancel{P \supset Q})}, T\,P \supset Q, F\,\Box P \supset \Box Q\}\}$
4. $\{\{\underline{T\,\Box(\cancel{P \supset Q})}, T\,P \supset Q, T\,\Box P, F\,\Box Q\}\}$
5. $\{\{T\,\Box(P \supset Q), T\,\Box P, F\,Q\}\}$
6. $\{\{\underline{T\,\Box(\cancel{P \supset Q})}, T\,P \supset Q, T\,\Box P, F\,Q\}\}$
7. $\{\{\underline{T\,\Box(\cancel{P \supset Q})}, F\,P, T\,\Box P, F\,Q\}, \{\underline{T\,\Box(\cancel{P \supset Q})}, T\,Q, T\,\Box P, F\,Q\}\}$
8. $\{\{\underline{T\,\Box(\cancel{P \supset Q})}, F\,P, \underline{\cancel{T\,\Box P}}, T\,P, F\,Q\}, \{\underline{T\,\Box(\cancel{P \supset Q})}, T\,Q, T\,\Box P, F\,Q\}\}$

Fig. 9. Tableau As Set Of Sets

6 Quasi-Realizations

Our algorithm for computing Realizations divides into two halves. The first half constructs an intermediate object, a *Quasi-Realization*, from a tableau proof. The second half converts a Quasi-Realization into a proper Realization. The construction of Quasi-Realizations is logic dependent—we present an algorithm for S4 only. Input to this algorithm is a tableau proof. The input to the Quasi-Realization to Realization algorithm is a Quasi-Realization, not a formal proof, and the construction is independent of the particular logic involved. Because space is limited, the Quasi-Realization to Realization algorithm is not given here. It can be found in both [14,7].

Informally the goal is to associate a Quasi-Realization with each signed formula occurrence in a tableau proof. This quasi-Realization is constructed according to how the branch on which the signed formula appears is continued to closure. A problem is that a particular signed formula occurrence can be on more than one branch, appearing before the branch splits. A Quasi-Realization computed on one branch might be different than a Quasi-Realization computed on another. If we were dealing with Realizations, a merging solution to this problem would involve the + operation and substitution, and would be of some complexity since the entire nested structure of the realizing formulas would need to be taken into consideration. Our approach here bypasses this problem by allowing a *set* of Quasi-Realizations rather than insisting on a single one. In fact + does not appear until we reach the Quasi-Realization to Realization algorithm.

From now on we assume that $v_1, v_2, \ldots$ is an enumeration of all justification variables of LP with no variable repeated, fixed once and for all. Case 4 of the definition below always uses v_k in Quasi-Realizations where the sign is T and $\Box_k$ is involved. In case 2 two conjunctive, or α, signed formulas are mentioned.

For one we use α with α_1 and α_2 as components. For the other we use α' with α'_1 and α'_2 as components. Similarly for disjunctive, or β, signed formulas.

Definition 4 (Quasi-Realization Function). *The mapping $\langle\!\langle \cdot \rangle\!\rangle$ is defined recursively on the set of signed annotated modal formulas.*

1. *If A is atomic, $\langle\!\langle T\,A \rangle\!\rangle = \{T\,A\}$ and $\langle\!\langle F\,A \rangle\!\rangle = \{F\,A\}$.*
2. $\langle\!\langle T\,\neg A \rangle\!\rangle = \{T\,\neg U \mid F\,U \in \langle\!\langle F\,A \rangle\!\rangle\}$.
   $\langle\!\langle F\,\neg A \rangle\!\rangle = \{F\,\neg U \mid T\,U \in \langle\!\langle T\,A \rangle\!\rangle\}$.
3. $\langle\!\langle \alpha \rangle\!\rangle = \{\alpha' \mid \alpha'_1 \in \langle\!\langle \alpha_1 \rangle\!\rangle$ *and* $\alpha'_2 \in \langle\!\langle \alpha_2 \rangle\!\rangle\}$.
   $\langle\!\langle \beta \rangle\!\rangle = \{\beta' \mid \beta'_1 \in \langle\!\langle \beta_1 \rangle\!\rangle$ *and* $\beta'_2 \in \langle\!\langle \beta_2 \rangle\!\rangle\}$.
4. $\langle\!\langle T\,\Box_n A \rangle\!\rangle = \{T\,v_n{:}U \mid T\,U \in \langle\!\langle T\,A \rangle\!\rangle\}$.
   $\langle\!\langle F\,\Box_n A \rangle\!\rangle = \{F\,t{:}(U_1 \vee \ldots \vee U_k) \mid F\,U_1, \ldots, F\,U_k \in \langle\!\langle F\,A \rangle\!\rangle$ *and*
   t is any justification term$\}$.
5. *The mapping is extended to* sets *of signed annotated formulas by letting* $\langle\!\langle S \rangle\!\rangle = \cup\{\langle\!\langle Z \rangle\!\rangle \mid Z \in S\}$.

Members of $\langle\!\langle Z \rangle\!\rangle$ are called Quasi-Realizers *of Z.*

As an example, suppose t, u, and w are justification terms and P and Q are atomic formulas. Here are some Quasi-Realization calculations, leading up to $F\,\Box_1(\Box_2 P \vee \neg\Box_3 Q)$. We do not produce *all* Quasi-Realizations, an infinite set. Here's the reasoning for one case. $F\,\Box_2 P \vee \neg\Box_3 Q$, in item 5, is an α, with $\alpha_1 = F\,\Box_2 P$ and $\alpha_2 = F\,\neg\Box_3 Q$. By items 2 and 3, we can take $\alpha'_1 = F\,t{:}P$ and $\alpha'_2 = F\,\neg v_3{:}Q$, and then $\alpha' = F\,t{:}P \vee \neg v_3{:}Q$, which is taken to be one of the members of $\langle\!\langle F\,\Box_2 P \vee \neg\Box_3 Q \rangle\!\rangle$.

1. $\{F\,P\} = \langle\!\langle F\,P \rangle\!\rangle$ and $\{T\,Q\} = \langle\!\langle T\,Q \rangle\!\rangle$
2. $\{F\,t{:}P, F\,u{:}P\} \subseteq \langle\!\langle F\,\Box_2 P \rangle\!\rangle$
3. $\{T\,v_3{:}Q\} = \langle\!\langle T\,\Box_3 Q \rangle\!\rangle$
4. $\{F\,\neg v_3{:}Q\} = \langle\!\langle F\,\neg\Box_3 Q \rangle\!\rangle$
5. $\{F\,t{:}P \vee \neg v_3{:}Q, F\,u{:}P \vee \neg v_3{:}Q\} \subseteq \langle\!\langle F\,\Box_2 P \vee \neg\Box_3 Q \rangle\!\rangle$
6. $\{F\,t{:}((t{:}P \vee \neg v_3{:}Q) \vee (u{:}P \vee \neg v_3{:}Q)), F\,w{:}(u{:}(P \vee \neg v_3{:}Q)\} \subseteq \langle\!\langle F\,\Box_1(\Box_2 P \vee \neg\Box_3 Q) \rangle\!\rangle$

In Section 8 we give an algorithm which will establish the following.

Theorem 3. *Let X be an annotated modal formula. Given a tableau proof of X in* S4*, a finite set $\{F\,Q_1, \ldots, F\,Q_k\}$ of quasi-realizers for $F\,X$ can be constructed so that $Q_1 \vee \ldots \vee Q_k$ is a theorem of* LP*.*

7 Mixed Tableaus

We now introduce what we call mixed tableaus, which unite modal features with justification logic features. They are based on tableaus as defined in Section 5, using a set of sets representation. Informally, a mixed tableau expands an S4 tableau by associating a set of Quasi-Realizers to each signed annotated modal formula appearing in it.

Definition 5 (Mixed Tableau). *A* mixed S4 tableau *is like a tableau except that members of branches are pairs (M, S) where M is a signed annotated modal formula and S is a finite set of signed justification formulas, meeting the following requirements.*

1. *If (M, S) occurs in a mixed tableau, it is required that $S \subseteq \langle\!\langle M \rangle\!\rangle$.*
2. *If, in a mixed tableau, we replace each entry (M, S) by just M, the result must be an annotated S4 tableau.*

In a mixed tableau, we refer to M as the modal part *of (M, S), and to S as the* justification part *of (M, S). We say a mixed tableau $\mathcal{T}^{mix}$ is an* expansion *of an S4 tableau $\mathcal{T}$ if $\mathcal{T}$ results from $\mathcal{T}^{mix}$ by eliminating the justification parts of node labels, as in item 2 of Definition 5.*

Definition 6 (Justification Sound). *Let $\mathcal{B}$ be a branch of a mixed tableau. By the* associated justification formula *for $\mathcal{B}$ we mean $\bigwedge \mathcal{B}_T^{just} \supset \bigvee \mathcal{B}_F^{just}$ where $\mathcal{B}_T^{just}$ is the set of all justification formulas X such that $T\,X$ occurs in the justification part of some member of $\mathcal{B}$ and $\mathcal{B}_F^{just}$ is the set of X such that $F\,X$ occurs in the justification part of some member of $\mathcal{B}$.*

We say a mixed S4 tableau branch is justification sound *provided that its associated justification formula is provable in axiomatic LP. We say a mixed S4 tableau is justification sound if each branch is.*

The heart of our Quasi-Realization work is the following, which immediately gives us a proof of Theorem 3.

Theorem 4. *Let $\mathcal{T}$ be an annotated S4 tableau that can be continued to one that is closed (or is closed already). Then $\mathcal{T}$ has a mixed tableau expansion $\mathcal{T}^{mix}$ that is justification sound, where $\mathcal{T}^{mix}$ can be algorithmically constructed from any closed modal tableau extending $\mathcal{T}$.*

Proof (of Theorem 3). Suppose X is an annotated modal formula, and we have a closed S4 tableau proof for X. The construction of that proof begins with the single-branch modal tableau consisting of just a root node, labeled $F\,X$. Since this trivial tableau can be continued to a closed tableau, by Theorem 4 it can be expanded to a mixed tableau that is justification sound. Such a mixed tableau must consist of just a root node, labeled $(F\,X, \{F\,Q_1, \ldots, F\,Q_k\})$, where $\{F\,Q_1, \ldots, F\,Q_k\} \subseteq \langle\!\langle F\,X \rangle\!\rangle$. Since this expanded tableau is justification sound, the formula $\bigwedge \emptyset \supset \bigvee \{Q_1, \ldots, Q_k\}$ is axiomatically LP provable. That is, $Q_1 \vee \ldots \vee Q_k$ is provable, where $Q_1, \ldots, Q_k$ are quasi-realizers for $F\,X$.

8 The Quasi-Realization Algorithm

Figure 10 contains an algorithm for constructing justification sound mixed tableaus from closed S4 tableaus, followed by an example in Figures 11 and 12. In Section 9 a proof of the correctness of the algorithm is given, and this establishes Theorem 4. The construction is a kind of 'backward induction'. Suppose $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$

is a sequence of annotated S4 tableaus, in which each arises from the preceding by a single application of an S4 branch extension rule, as given in Section 5. Suppose also that $\mathcal{T}_k$ is closed. We show $\mathcal{T}_k$ has a mixed tableau expansion that is justification sound. Then, using this, we show the same for $\mathcal{T}_{k-1}$, then for $\mathcal{T}_{k-2}$, and so on back to $\mathcal{T}_1$. A bit more properly, the algorithm produces a mixed tableau expansion for each $\mathcal{T}_i$; the correctness proof in the following section shows that it must be justification sound.

A branch extension rule application modifies only one branch—all others remain unchanged. Consequently the algorithm is stated in terms of branch extension rules applied to single branches. The rest of the mixed tableau being constructed does not change, so unaffected branches are not explicitly displayed.

S4 tableaus are understood as sets of branches, with branches being sets of signed annotated formulas, as in Section 5. We make use of the notion convention introduced there, where $\mathcal{B}, Z$, or $\overset{\mathcal{B}}{Z}$, is a branch consisting of the members of $\mathcal{B}$, and Z (which is understood not to occur in $\mathcal{B}$).

The idea is to *expand* branches of an annotated S4 tableau so they become branches of a mixed tableau. If $\mathcal{B}$ is an S4 tableau branch, we will write $\mathcal{B}^E$ to denote an expansion of it to a mixed tableau branch. Each signed annotated formula M in $\mathcal{B}$ is transformed into a pair (M, S) in $\mathcal{B}^E$ so that $S \subseteq \langle\!\langle M \rangle\!\rangle$. Of course $\mathcal{B}^E$ is not unique—it is simply some expansion. In one case of the algorithm more than one branch expansion must be referenced, and we use $\mathcal{B}^{E_1}$ and $\mathcal{B}^{E_2}$ as notation. We write $\mathcal{B} \xrightarrow{exp} \mathcal{B}^E$ to indicate that annotated S4 tableau branch $\mathcal{B}$ expands to mixed tableau branch $\mathcal{B}^E$.

If $\mathcal{B}^{E_1}$ and $\mathcal{B}^{E_2}$ are both expansions of the same branch, $\mathcal{B}$, by $\mathcal{B}^{E_1} \dot{\cup} \mathcal{B}^{E_2}$ we mean the mixed tableau branch consisting of all $(M, S_1 \cup S_2)$ where $(M, S_1) \in \mathcal{B}^{E_1}$ and $(M, S_2) \in \mathcal{B}^{E_2}$.

In a few of the algorithm cases we refer to a *trivial expansion*. A trivial expansion of a signed formula M is (M, S) where S is *any* finite set such that $S \subseteq \langle\!\langle M \rangle\!\rangle$. A trivial expansion of an S4 branch replaces each member with a trivial expansion. In our Prolog implementation a particular easily computed trivial expansion is used, but the details don't matter here.

The algorithm is stated schematically below. We give a reading of the α Case as a representative example of how the algorithm notation should be understood. The idea is, we say how to expand the S4 tableau branch $\mathcal{B}, \alpha$ provided we already know how to expand $\mathcal{B}, \alpha_1, \alpha_2$. So, assume we have an expansion for S4 tableau branch $\mathcal{B}, \alpha_1, \alpha_2$, where $\mathcal{B}$ expands to $\mathcal{B}^E$, α_1 expands to (α_1, S_1), and α_2 expands to (α_2, S_2). Then S4 tableau branch $\mathcal{B}, \alpha$ expands to $\mathcal{B}^E, (\alpha, S)$, where S consists of all α signed formulas for which $\alpha_1 \in S_1$ and $\alpha_2 \in S_2$. (In the schematic we used α', α'_1, and α'_2 in characterizing S, simply because α, α_1, and α_2 were already in use to designate members of S4 tableau branches.)

Recall that v_1, v_2, ... is a fixed enumeration of all justification variables of LP with no variable repeated.

In the $F\,\Box$ case of Figure 10, $\bigwedge \mathcal{A} \supset \bigvee S$ appears as the associated justification formula for the branch $((\mathcal{B}^\sharp)^E, (F\,X, S_0)$. In fact, $\mathcal{A} = ((\mathcal{B}^\sharp)^E)^{just}_T$, using the notation of Definition 6, but such detailed notation distracts from the ba-

Atomic Cases

$$\begin{array}{c} \mathcal{B} \\ T\,P \\ F\,P \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (T\,P, \{T\,P\}) \\ (F\,P, \{F\,P\}) \end{array} \text{ where } \mathcal{B}^E \text{ trivially expands } \mathcal{B}$$

$$\begin{array}{c} \mathcal{B} \\ T\,\bot \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (T\,\bot, \{T\,\bot\}) \end{array} \text{ where } \mathcal{B}^E \text{ trivially expands } \mathcal{B}$$

α **Cases**

$$\dfrac{\begin{array}{c} \mathcal{B} \\ \alpha_1 \\ \alpha_2 \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (\alpha_1, S_1) \\ (\alpha_2, S_2) \end{array} \text{ where } S = \{\alpha' \mid \alpha'_1 \in S_1 \text{ and } \alpha'_2 \in S_2\}}{\begin{array}{c} \mathcal{B} \\ \alpha \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (\alpha, S) \end{array}}$$

β **Cases**

$$\dfrac{\begin{array}{c} \mathcal{B} \\ \beta_1 \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^{E_1} \\ (\beta_1, S_1) \end{array} \quad \begin{array}{c} \mathcal{B} \\ \beta_2 \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^{E_2} \\ (\beta_2, S_2) \end{array} \text{ where } S = \{\beta' \mid \beta'_1 \in S_1 \text{ and } \beta'_2 \in S_2\} \text{ and } \mathcal{B}^E = \mathcal{B}^{E_1} \dot{\cup} \mathcal{B}^{E_2}}{\begin{array}{c} \mathcal{B} \\ \beta \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (\beta, S) \end{array}}$$

Negation Cases

$$\dfrac{\begin{array}{c} \mathcal{B} \\ F\,X \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (F\,X, S_0) \end{array} \text{ where } S = \{T\,\neg Z \mid F\,Z \in S_0\}}{\begin{array}{c} \mathcal{B} \\ T\,\neg X \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (T\,\neg X, S) \end{array}}$$

$$\dfrac{\begin{array}{c} \mathcal{B} \\ T\,X \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (T\,X, S_0) \end{array} \text{ where } S = \{F\,\neg Z \mid T\,Z \in S_0\}}{\begin{array}{c} \mathcal{B} \\ F\,\neg X \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (F\,\neg X, S) \end{array}}$$

$T\,\Box$ **Case**

$$\dfrac{\begin{array}{c} \mathcal{B} \\ \cancel{T\,\Box_k X} \\ T\,X \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (\cancel{T\,\Box_k X}, S_0) \\ (T\,X, S_1) \end{array} \text{ where } S = S_0 \cup \{T\,v_k{:}Z \mid T\,Z \in S_1\}}{\begin{array}{c} \mathcal{B} \\ T\,\Box_k X \end{array} \xrightarrow{exp} \begin{array}{c} \mathcal{B}^E \\ (T\,\Box_k X, S) \end{array}}$$

$F\,\Box$ **Case**

$$\dfrac{\begin{array}{c} \mathcal{B}^\sharp \\ F\,X \end{array} \xrightarrow{exp} \begin{array}{c} (\mathcal{B}^\sharp)^E \\ (F\,X, S_0) \end{array}}{\begin{array}{c} \mathcal{B}^\sharp \\ \mathcal{B} - \mathcal{B}^\sharp \\ F\,\Box_n X \end{array} \xrightarrow{exp} \begin{array}{c} (\mathcal{B}^\sharp)^E \\ (\mathcal{B} - \mathcal{B}^\sharp)^E \\ (F\,\Box_n X, \{F\,t{:}\bigvee S\}) \end{array}}$$

where $(\mathcal{B} - \mathcal{B}^\sharp)^E$ trivially expands $\mathcal{B} - \mathcal{B}^\sharp$, $S = \{Z \mid F\,Z \in S_0\}$, $\bigwedge \mathcal{A} \supset \bigvee S$ is the associated justification formula for branch $((\mathcal{B}^\sharp)^E, (F\,X, S_0)$ and $\vdash_{\mathsf{LP}} \bigwedge \mathcal{A} \supset t{:}\bigvee S$

Fig. 10. Quasi-Realization Algorithm

sic idea and we have suppressed it here. The existence of a term t such that $\vdash_{\mathsf{LP}} \bigwedge \mathcal{A} \supset t{:}\bigvee S$ will be guaranteed by the Lifting Lemma. Also note that the combination $\mathcal{B}^\sharp$ and $\mathcal{B} - \mathcal{B}^\sharp$ below the line simply amounts to $\mathcal{B}$, though the separation is useful for our purposes.

We give an example to illustrate how the Quasi-Realization Algorithm works. Figure 11 shows an S4 proof of the annotated formula $(\Box_1 A \vee \Box_2 B) \supset \Box_3(A \vee B)$, using the representation of tableaus described in Section 5. Each numbered item should be thought of as the set of signed formulas making up a tableau branch. A detailed description follows. 1 is the initial single branch tableau. Single branch tableau 2 follows from 1 by α. A β rule application creates a tableau with two branches, 3 and 4. Modal rule applications on $F\,\Box_3(A \vee B)$ in 3 and 4 produce the two-branched tableau having branches 5 and 6. Modal rule applications on $T\,\Box_1 A$ and $T\,\Box_2 B$ in these give the two branches 7 and 8. Finally, α rule applications give the two branches 9 and 10, both of which are atomically closed.

1. $F\,(\Box_1 A \vee \Box_2 B) \supset \Box_3(A \vee B)$

2. $T\,\Box_1 A \vee \Box_2 B$
   $F\,\Box_3(A \vee B)$

3. $T\,\Box_1 A$
   $F\,\Box_3(A \vee B)$

4. $T\,\Box_2 B$
   $F\,\Box_3(A \vee B)$

5. $T\,\Box_1 A$
   $F\,A \vee B$

6. $T\,\Box_2 B$
   $F\,A \vee B$

7. ~~$T\,\Box_1 A$~~
   $F\,A \vee B$
   $T\,A$

8. ~~$T\,\Box_2 B$~~
   $F\,A \vee B$
   $T\,B$

9. ~~$T\,\Box_1 A$~~
   $T\,A$
   $F\,A$
   $F\,B$

10. ~~$T\,\Box_2 B$~~
    $T\,B$
    $F\,A$
    $F\,B$

Fig. 11. S4 Tableau Proof (to be expanded)

Next, the proof created in Figure 11 is converted to a mixed tableau, displayed in Figure 12. The work is from bottom up. In Figure 11, 9 is an atomically closed branch. For this, the algorithm makes use of a *trivial* expansion, giving the corresponding 9 of Figure 12, and similarly for 10. Branch 7 in Figure 11 yields branch 9 by an α rule. Since 9 in Figure 11 expands to 9 in Figure 12, 7 of Figure 11 converts to 7 of Figure 12 by the α case of the Algorithm. Similarly for 8 and 10. Then branch 5 of Figure 11 converts to 5 of Figure 12 because of the 7 conversion, and the $T\Box$ case of the Algorithm, and similarly for 6 and 8. Branch 3 of Figure 11 yields branch 5 by the $F\Box$ rule. The associated

justification formula for branch 5 is $v_1{:}A \supset (A \vee B)$. Justification term t, in 3, is such that $v_1{:}A \supset t{:}(A \vee B)$ is provable in LP. Existence is guaranteed by the Lifting Lemma 2. Similarly u in branch 4 is such that $v_2{:}B \supset u{:}(A \vee B)$. Branch 2 yields branches 3 and 4 using the β rule. Note that in branch 2 in Figure 12, the justification part associated with $F\,\Box_3(A \vee B)$ is the union of those parts from branches 3 and 5. Finally 1 is a straightforward application of the α rule.

Then, according to the algorithm, $\{F\,(v_1{:}A \vee v_2{:}B) \supset t{:}(A \vee B), F\,(v_1{:}A \vee v_2{:}B) \supset u{:}(A \vee B)\}$ is a set of quasi-realizers for $F\,(\Box_1 A \vee \Box_2 B) \supset \Box_3(A \vee B)$. In fact, the following is provable in LP.

$$\bigvee\{(v_1{:}A \vee v_2{:}B) \supset t{:}(A \vee B), (v_1{:}A \vee v_2{:}B) \supset u{:}(A \vee B)\}$$

1. $(F\,(\Box_1 A \vee \Box_2 B) \supset \Box_3(A \vee B), \{F\,(v_1{:}A \vee v_2{:}B) \supset t{:}(A \vee B)$
   $F\,(v_1{:}A \vee v_2{:}B) \supset u{:}(A \vee B)\})$
2. $(T\,\Box_1 A \vee \Box_2 B, \{T\,v_1{:}A \vee v_2{:}B\})$
   $(F\,\Box_3(A \vee B), \{F\,t{:}(A \vee B), F\,u{:}(A \vee B)\})$
3. $(T\,\Box_1 A, \{T\,v_1{:}A\})$
   $(F\,\Box_3(A \vee B), \{F\,t{:}(A \vee B)\})$
4. $(T\,\Box_2 B, \{T\,v_2{:}B\})$
   $(F\,\Box_3(A \vee B), \{F\,u{:}(A \vee B)\})$
5. $(T\,\Box_1 A, \{T\,v_1{:}A\})$
   $(F\,A \vee B, \{F\,A \vee B\})$
6. $(T\,\Box_2 B, \{T\,v_2{:}B\})$
   $(F\,A \vee B, \{F\,A \vee B\})$
7. $(\cancel{T\,\Box_1 A}, \{T\,v_1{:}A\})$
   $(F\,A \vee B, \{F\,A \vee B\})$
   $(T\,A, \{T\,A\})$
8. $(\cancel{T\,\Box_2 B}, \{T\,v_2{:}B\})$
   $(F\,A \vee B, \{F\,A \vee B\})$
   $(T\,B, \{T\,B\})$
9. $(\cancel{T\,\Box_1 A}, \{T\,v_1{:}A\})$
   $(T\,A, \{T\,A\})$
   $(F\,A, \{F\,A\})$
   $(F\,B, \{F\,B\})$
10. $(\cancel{T\,\Box_2 B}, \{T\,v_2{:}B\})$
    $(T\,B, \{T\,B\})$
    $(F\,A, \{F\,A\})$
    $(F\,B, \{F\,B\})$

Justification term t, in 3, is such that $v_1{:}A \supset t{:}(A \vee B)$ is provable in LP. Similarly u in 4 is such that $v_2{:}B \supset u{:}(A \vee B)$ is LP provable.

Fig. 12. S4 Tableau Proof (expanded)

9 Quasi-Realization Algorithm Correctness Proof

This section is devoted to showing the correctness of the Quasi-Realization Algorithm, and hence proving Theorem 4. It is straightforward that the algorithm produces a mixed tableau expansion. We concentrate on showing the resulting mixed tableau must be justification sound, Definition 6. To do this, we show it for the Atomic Cases, and show that each rule of the algorithm preserves justification soundness.

Proof (Correctness for Quasi-Realization Algorithm).

Atomic Cases Consider the first of the two atomic cases—the second is similar. The mixed tableau branch produced in this case is $\mathcal{B}^E, T\,P, F\,P$. The associated justification formula is $[\bigwedge(\mathcal{B}^E)^{just}_T \wedge P] \supset [\bigvee(\mathcal{B}^E)^{just}_F \vee P]$, and this is trivially an LP theorem, so the branch is justification sound.

α **Case** Assume that $\mathcal{B}^E, (\alpha_1, S_1), (\alpha_2, S_2)$ is a mixed tableau branch that is justification sound. We must show the same for $\mathcal{B}^E, (\alpha, S)$ where $S = \{\alpha' \mid \alpha'_1 \in S_1 \text{ and } \alpha'_2 \in S_2\}$. Since $S_1 \subseteq \langle\!\langle \alpha_1 \rangle\!\rangle$ and $S_2 \subseteq \langle\!\langle \alpha_2 \rangle\!\rangle$, it is easy to see from Definition 4 that $S \subseteq \langle\!\langle \alpha \rangle\!\rangle$. After this case we leave such arguments to the reader. We must show the branch is justification sound.

Since we only consider $\wedge$, $\vee$, and $\supset$, there are three possibilities for α. We look at one of them, with $\alpha = F\,A \supset B$; the other two cases are similar. All three could be condensed into a single argument by making use of uniform notation, but this would be a bit of a diversion just now. So, assume $\mathcal{B}^E, (T\,A, S_1), (F\,B, S_2)$ is justification sound; we show the same for $\mathcal{B}^E, (F\,A \supset B, S)$.

Let us say $S_1 = \{T\,A_1, \ldots, T\,A_m\}$ and $S_2 = \{F\,B_1, \ldots, F\,B_n\}$. Then the associated justification formula for $\mathcal{B}^E, (T\,A, S_1), (F\,B, S_2)$ is the following.

$$\Big[\bigwedge(\mathcal{B}^E)^{just}_T \wedge \bigwedge\{A_1, \ldots, A_m\}\Big] \supset \Big[\bigvee(\mathcal{B}^E)^{just}_F \vee \bigvee\{B_1, \ldots, B_n\}\Big]$$

By classical logic we also have provability of the following, where i ranges over $1, \ldots, m$ and j ranges over $1, \ldots, n$.

$$\bigwedge(\mathcal{B}^E)^{just}_T \supset \left[\bigvee(\mathcal{B}^E)^{just}_F \vee \bigvee_{i,j}(A_i \supset B_j)\right]$$

Thus the associated justification formula for $\mathcal{B}, (F\,A \supset B, S)$ is provable.

β **Case** Assume that $\mathcal{B}^{E_1}, (\beta_1, S_1)$ and $\mathcal{B}^{E_2}, (\beta_2, S_2)$ are justification sound. We show this also the case for $\beta^E, (\beta, S)$, where $S = \{\beta' \mid \beta'_1 \in S_1 \text{ and } \beta'_2 \in S_2\}$ and $\mathcal{B}^E = \mathcal{B}^{E_1} \dot\cup \mathcal{B}^{E_2}$. As with α there are three cases, and we only consider one of them, where $\beta = T\,A \supset B$. So, assume that $\mathcal{B}^{E_1}, (F\,A, S_1)$ and $\mathcal{B}^{E_2}, (T\,B, S_2)$ are justification sound.

Suppose $S_1 = \{F\,A_1, \ldots, F\,A_m\}$ and $S_2 = \{T\,B_1, \ldots, T\,B_n\}$. Then the provable associated justification formulas for $\mathcal{B}^{E_1}, (F\,A, S_1)$ and $\mathcal{B}^{E_2}, (T\,B, S_2)$ are the following.

$$\bigwedge(\mathcal{B}^{E_1})^{just}_T \supset \Big[\bigvee(\mathcal{B}^{E_1})^{just}_F \vee \bigvee\{A_1, \ldots, A_m\}\Big]$$

$$\Big[\bigwedge(\mathcal{B}^{E_2})^{just}_T \wedge \bigwedge\{B_1 \wedge \ldots \wedge B_n\}\Big] \supset \bigvee(\mathcal{B}^{E_2})^{just}_F$$

$\mathcal{B}^E = \mathcal{B}^{E_1} \dot\cup \mathcal{B}^{E_2}$, and it follows easily that $(\mathcal{B}^E)^{just}_T = (\mathcal{B}^{E_1})^{just}_T \cup (\mathcal{B}^{E_2})^{just}_T$ and $(\mathcal{B}^E)^{just}_F = (\mathcal{B}^{E_1})^{just}_F \cup (\mathcal{B}^{E_2})^{just}_F$. Then we have provability of the following.

$$\bigwedge(\mathcal{B}^E)^{just}_T \supset \Big[\bigvee(\mathcal{B}^E)^{just}_F \vee \bigvee\{A_1, \ldots, A_m\}\Big]$$

$$\Big[\bigwedge(\mathcal{B}^E)^{just}_T \wedge \bigwedge\{B_1 \wedge \ldots \wedge B_n\}\Big] \supset \bigvee(\mathcal{B}^E)^{just}_F$$

By classical logic this gives provability of the following, where i ranges over $1, 2, \ldots, m$ and j ranges over $1, 2, \ldots, n$.

$$\left[\bigwedge(\mathcal{B}^E)^{just}_T \wedge \bigwedge_{i,j}(A_i \supset B_j)\right] \supset \bigvee(\mathcal{B}^E)^{just}_F$$

Thus the associated justification formula for $\mathcal{B}, (T\,A \supset B, S)$ is provable.

Negation Cases These cases are similar to the α and β cases, but are simpler and are left to the reader.

$T\Box$ **Case** Assume that $\mathcal{B}^E, (\cancel{T\Box_k X}, S_0), (T\,X, S_1)$ is a justification sound mixed tableau branch. Then $\mathcal{B}^E, (T\,\Box_k X, S)$ is a mixed tableau branch, where $S = S_0 \cup \{T\,v_k{:}Z \mid T\,Z \in S_1\}$. We show it is justification sound.

Suppose $S_0 = \{T\,v_k{:}W_1, \ldots, T\,v_k{:}W_m\}$ and $S_1 = \{T\,Z_1, \ldots, T\,Z_k\}$. Then the provable associated justification formula for $\mathcal{B}^E, (\cancel{T\Box_k X}, S_0), (T\,X, S_1)$ is the following.

$$\Big[\bigwedge(\mathcal{B}^E)^{just}_T \wedge \bigwedge\{v_k{:}W_1, \ldots, v_k{:}W_m\} \wedge \bigwedge\{Z_1, \ldots, Z_k\}\Big] \supset \bigvee(\mathcal{B}^E)^{just}_F$$

Using Factivity, Axiom A2, we have LP provability of the following.

$$\Big[\bigwedge(\mathcal{B}^E)^{just}_T \wedge \bigwedge\{v_k{:}W_1, \ldots, v_k{:}W_m, v_k{:}Z_1, \ldots, v_k{:}Z_k\}\Big] \supset \bigvee(\mathcal{B}^E)^{just}_F$$

This is the associated justification formula for $\mathcal{B}^E, (T\,\Box_k X, S)$.

$F\Box$ **Case** Assume that $(\mathcal{B}^\sharp)^E, (F\,X, S_0)$ is a justification sound mixed tableau branch. Then $(\mathcal{B}^\sharp)^E, (\mathcal{B} - \mathcal{B}^\sharp)^E, (F\,\Box_n X, \{F\,t : \bigvee S\})$ is a mixed tableau branch, where $(\mathcal{B} - \mathcal{B}^\sharp)^E$ trivially expands $\mathcal{B} - \mathcal{B}^\sharp$, $S = \{Z \mid F\,Z \in S_0\}$, and t is any justification term. We show $(\mathcal{B}^\sharp)^E, (\mathcal{B} - \mathcal{B}^\sharp)^E, (F\,\Box_n X, \{F\,t{:}\bigvee S\})$ is justification sound, given the right choice of t.

Note that since all members of $\mathcal{B}^\sharp$ are T-signed, the LP-provable associated justification formula for $(\mathcal{B}^\sharp)^E, (F\,X, S_0)$ is simply $\bigwedge((\mathcal{B}^\sharp)^E)^{just}_T \supset \bigvee S$, where $S = \{Z \mid F\,Z \in S_0\}$. Also members of $\mathcal{B}^\sharp$ are necessitated, so by the Lifting Lemma 2, for some justification term t, $\vdash_{\mathsf{LP}} \bigwedge((\mathcal{B}^\sharp)^E)^{just}_T \supset t{:}\bigvee S$. Then, trivially, the following is also LP-provable

$$\Big[\bigwedge((\mathcal{B}^\sharp)^E)^{just}_T \wedge \bigwedge((\mathcal{B} - \mathcal{B}^\sharp)^E)^{just}_T\Big] \supset \Big[t{:}\bigvee S \vee \bigvee((\mathcal{B} - \mathcal{B}^\sharp)^E)^{just}_F\Big]$$

and this is the associated justification formula for $(\mathcal{B}^\sharp)^E, (\mathcal{B} - \mathcal{B}^\sharp)^E, (F\,\Box_n X, \{F\,t{:}\bigvee S\})$ as specified by the algorithm.

10 Realizations

Quasi-Realizations convert to Realizations. There is an algorithm for doing this in [7] that does not depend on tableau proofs, but only on the structure of a

Quasi-Realization formula. It applies uniformly to a wide range of justification logics, not just to LP. Because of space limitations we omit the algorithm, and just state what it gives us.

We begin with a definition of Realization equivalent to the usual one, but following the lines of Definition 4. Differences are confined to case 4 where disjunction appearing in the definition of quasi-realizer is folded into a justification term by using the + operator. We still assume that $v_1, v_2, \ldots$ is an enumeration of all justification variables of LP, with no justification variable repeated.

Definition 7. *The mapping $[\![\cdot]\!]$ is defined recursively on the set of signed annotated modal formulas.*

1. *If A is atomic, $[\![T\,A]\!] = \{T\,A\}$ and $[\![F\,A]\!] = \{F\,A\}$.*
2. $[\![T\,\neg A]\!] = \{T\,\neg U \mid F\,U \in [\![F\,A]\!]\}$.
   $[\![F\,\neg A]\!] = \{F\,\neg U \mid T\,U \in [\![T\,A]\!]\}$.
3. $[\![\alpha]\!] = \{\alpha' \mid \alpha'_1 \in [\![\alpha_1]\!]$ *and* $\alpha'_2 \in [\![\alpha_2]\!]\}$.
   $[\![\beta]\!] = \{\beta' \mid \beta'_1 \in [\![\beta_1]\!]$ *and* $\beta'_2 \in [\![\beta_2]\!]\}$.
4. $[\![T\,\Box_n A]\!] = \{T\,v_n{:}U \mid T\,U \in [\![T\,A]\!]\}$.
   $[\![F\,\Box_n A]\!] = \{F\,t{:}U \mid F\,U \in [\![F\,A]\!]$ *and t is any justification term*$\}$.
5. *The mapping is extended to* sets *of signed annotated formulas by letting* $[\![S]\!] = \cup\{[\![Z]\!] \mid Z \in S\}$.

Members of $[\![Z]\!]$ are Realizers *of Z, where Z is a signed, annotated modal formula. A* normal Realization *of annotated modal A is any justification formula U where $F\,U \in [\![F\,A]\!]$. For a modal formula A without annotations, a normal Realization for A is any normal Realization for A', where A' is an annotated version of A.*

Substitutions are fundamental. A substitution σ replaces justification variables with justification terms. For a justification formula A the result of applying a substitution σ is denoted $A\sigma$. It is easy to show that substitutions turn LP theorems into LP theorems, though generally the constant specification will change. Substitution σ meets the *no new variable* condition if, for every v_k in the domain of σ, the justification term $v_k\sigma$ contains no variables other than v_k. *σ lives on* A if, for every justification variable v_k in the domain of σ, $\Box_k$ occurs in A,

Definition 8. *Let A be an annotated modal formula, $\mathcal{A}$ be a set of justification formulas, A' be a single justification formula, and σ be a substitution.*

1. $\mathcal{A} \xrightarrow{T\,A} (A', \sigma)$ *means: σ lives on A and meets the no new variable condition; $T\,\mathcal{A} \subseteq \langle\!\langle T\,A \rangle\!\rangle$; $T\,A' \in [\![T\,A]\!]$; and $\vdash_{LP} A' \supset (\bigwedge \mathcal{A})\sigma$.*
2. $\mathcal{A} \xrightarrow{F\,A} (A', \sigma)$ *means: σ lives on A and meets the no new variable condition; $F\,\mathcal{A} \subseteq \langle\!\langle F\,A \rangle\!\rangle$; $F\,A' \in [\![F\,A]\!]$; and $\vdash_{LP} (\bigvee \mathcal{A})\sigma \supset A'$.*

One can read $\mathcal{A} \xrightarrow{T\,A} (A', \sigma)$ as saying that the set of quasi-realizers $\mathcal{A}$ for $T\,A$ *condenses* to the single realizer $T\,A'$ using substitution σ, and similarly for $\mathcal{A} \xrightarrow{F\,A} (A', \sigma)$.

Theorem 5 (Condensing). *Let A be an annotated modal formula. For each finite set $\mathcal{A}$ of justification formulas:*

1. *If $T\,\mathcal{A} \subseteq \langle\!\langle T\,A \rangle\!\rangle$ then there are A' and σ so that $\mathcal{A} \xrightarrow{T\,A} (A', \sigma)$.*
2. *If $F\,\mathcal{A} \subseteq \langle\!\langle F\,A \rangle\!\rangle$ then there are A' and σ so that $\mathcal{A} \xrightarrow{F\,A} (A', \sigma)$.*

As has been said several times, the proof of Theorem 5 is algorithmic, and [7] can be consulted for details.

Corollary 2 (Realization). *Every formula provable in S4 has a normal Realization that is provable in LP.*

Proof. Suppose X is a theorem of S4. Let A be an annotated version of X, any one will do. Then from Theorem 3, proved using the algorithm given in Figure 10, there are $Q_1, \ldots, Q_k$ with $\{F\,Q_1, \ldots, F\,Q_k\} \subseteq \langle\!\langle F\,A \rangle\!\rangle$ such that $Q_1 \vee \ldots \vee Q_k$ is a theorem of LP. By part 2 of Theorem 5 there is a substitution σ and a formula A' with $F\,A' \in [\![F\,A]\!]$ such that $(Q_1 \vee \ldots \vee Q_k)\sigma \supset A'$ is a theorem of LP. Since $(Q_1 \vee \ldots \vee Q_k)\sigma$ must also be provable in LP, so is A', and this is a normal Realization of A, and hence of X.

At the end of Section 8 we presented an example showing that the annotated modal formula $(\Box_1 A \vee \Box_2 B) \supset \Box_3(A \vee B)$, provable in S4, has the following Quasi-Realization set, $\{(v_1 : A \vee v_2 : B) \supset t : (A \vee B), (v_1 : A \vee v_2 : B) \supset u : (A \vee B)\}$, where $\vdash_{\mathsf{LP}} v_1 : A \supset t : (A \vee B)$ and $\vdash_{\mathsf{LP}} v_2 : B \supset u : (A \vee B)$. Then $\vdash_{\mathsf{LP}} [(v_1 : A \vee v_2 : B) \supset t : (A \vee B)] \vee [(v_1 : A \vee v_2 : B) \supset u : (A \vee B)]$. Applying the (unstated) algorithm converting Quasi-Realizers to Realizers, we obtain that, $(v_1 : A \vee v_2 : B) \supset (c \cdot t + c \cdot u) : (A \vee B)$ is a provable normal Realization of $(\Box_1 A \vee \Box_2 B) \supset \Box_3(A \vee B)$, where c internalizes a proof of $(A \vee B) \supset (A \vee B)$.

11 What Next?

Here is a brief summary of work that remains undone. It is extensive.

We have given a constructive proof of Quasi-Realization from modal S4 to justification LP. As noted several times, a uniform algorithmic conversion from Quasi-Realizers to Realizers is available, [7]. The ideas extend directly to any modal logic having a similar destructive tableau system. Other cut-free proof methods extend things to a still richer variety of logics. In [7] it is shown that the family of modal logics having justification logic counterparts is infinite, which is somewhat surprising. But the proof is non-constructive. It is not known whether something similar can be shown constructively.

Recently LP has been extended to admit quantifiers, [3,6], with a Realization theorem connecting it with first-order S4. But a *monotinicity* condition is assumed. Work is in progress on a constant domain version, but this is incomplete. Varying domain assumptions have not yet been considered. No modal logic except S4 has been examined.

Hybrid logic and paraconsistent modal logics are largely unexplored as far as justification counterparts are concerrned.

There are still things to do.

References

1. S. N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, Mar. 2001. 1, 2, 3
2. S. N. Artemov. The logic of justification. *The Review of Symbolic Logic*, 1(4):477–513, Dec. 2008. 1, 3
3. S. N. Artemov and T. Yavorskaya (Sidon). First-order logic of proofs. Technical Report TR–2011005, CUNY Ph.D. Program in Computer Science, May 2011. 1, 19
4. S. Feferman, J. W. Dawson Jr., S. C. Kleene, G. H. Moore, R. M. Solovay, J. van Heijenoort, W. D. Goldfarb, C. Parsons, and W. Sieg, editors. *Kurt Gödel Collected Works*. Oxford, 1986-2003. Five volumes. 20
5. M. Fitting. Realizations and LP. *Annals of Pure and Applied Logic*, 161(3):368–387, Dec. 2009. Published online August 2009. 6
6. M. Fitting. Possible world semantics for first order LP. Technical Report TR–2011010, CUNY Ph.D. Program in Computer Science, Sept. 2011. 1, 19
7. M. C. Fitting. Modal logics, justification logics, and realization. Submitted October 6, 2015. 2, 9, 17, 19
8. M. C. Fitting. Prolog code for S4 realization. See *Realization Implemented* at `http://melvinfitting.org/bookspapers/techreports.html`. 2
9. M. C. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Co., Dordrecht, 1983.
10. M. C. Fitting. First-order modal tableaux. *Journal of Automated Reasoning*, 4:191–213, 1988.
11. M. C. Fitting. The logic of proofs, semantically. *Annals of Pure and Applied Logic*, 132:1–25, 2005. 1
12. M. C. Fitting. Reasoning with justifications. In D. Makinson, J. Malinowski, and H. Wansing, editors, *Towards Mathematical Philosophy*, number 28 in Trends in Logic, chapter 6, pages 107 – 123. Springer, 2009. 1
13. M. C. Fitting. Prefixed tableaus and nested sequents. *Annals of Pure and Applied Logic*, 163:291–313, 2012. Available on-line at `http://dx.doi.org/10.1016/j.apal.2011.09.004`.
14. M. C. Fitting. Realization implemented. Technical Report TR-2013005, CUNY Ph.D. Program in Computer Science, May 2013. `http://www.cs.gc.cuny.edu/tr/`. 9
15. M. C. Fitting. Realization using the Model Existence Theorem. *Journal of Logic and Computation*, July 16, 2013. Published online. 1
16. K. Gödel. Eine Interpretation des intuistionistischen Aussagenkalkuls. *Ergebnisse eines mathematischen Kolloquiums*, 4:39–40, 1933. Translated as *An interpretation of the intuitionistic propositional calculus* in [4] I, 296-301. 1
17. R. Goetschi and R. Kuznets. Realization for justification logics via nested sequents: Modularity through embedding. *Annals of Pure and Applied Logic*, 163(9):1271–1298, Sept. 2012. Published online March 2012. 1
18. R. M. Smullyan. A unifying principle in quantification theory. *Proceedings of the National Academy of Sciences*, 49(6):828–832, June 1963. 3
19. R. M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, 1968. Revised Edition, Dover Press, New York, 1994. 3