# Rule Refinement for Semantic Tableau Calculi

Dmitry Tishkovsky and Renate A. Schmidt[⋆]

School of Computer Science, The University of Manchester, Manchester, UK

**Abstract.** This paper investigates refinement techniques for semantic tableau calculi. The focus is on techniques to reduce branching in inference rules and thus allow more effective ways of carrying out deductions. We introduce an easy to apply, general principle of atomic rule refinement, which depends on a purely syntactic condition that can be easily verified. The refinement has a wide scope, for example, it is immediately applicable to inference rules associated with frame conditions of modal logics, or declarations of role properties in description logics, and it allows for routine development of hypertableau-like calculi for logics with disjunction and negation. The techniques are illustrated on Humberstone's modal logic $\mathrm{K}_m(\neg)$ with modal operators defined with respect to both accessibility and inaccessibility, for which two refined calculi are given.

## 1 Introduction

The tableau method is a popular deduction method in automated reasoning. Tableau methods in various forms are successfully used and applied for many non-classical logics and are especially apt for new application domains to develop new deduction systems. Of all the different forms, semantic tableau calculi in the styles of Smullyan and Fitting [22, 8] are widely used and widely taught in logic courses, because the rules of inference are easily explained and understood, and deductions are carried out in a completely goal-directed way. In explicit semantic tableau approaches the application of the inference rules is order independent (because these approaches are proof confluent), which avoids the overhead and complication associated with handling don't know non-determinism of non-invertible rules in direct methods [1] (see also the discussion in [13]). Because semantic tableau approaches construct and return models, they are suitable for fault diagnosis and debugging, which is useful in areas such as ontology development, theory creation and multi-agent systems.

We are interested in refinements of semantic tableau calculi that lead to improvements in carrying out deductions. When carrying out deductions by hand a natural inclination is to delay the application of branching rules as much as possible because these are cumbersome. When it can no longer be delayed, we tend to apply rules creating fewer branches earlier than those creating more branches, unless looking ahead allows us to see that several branches created by an inference step can be closed quickly. In a prover, where everything is

[⋆] This research was supported by UK EPSRC research grant EP/H043748/1.

automated, the overhead of branching is high as well, so that similar strategies are useful and have been shown to give significant speed-ups, as we found for example in the evaluations undertaken in [14, 23]. Similar considerations and better performance have motivated the development and use of hypertableau, hyperresolution or selection-based resolution methods [3, 4, 7, 16].

It is therefore natural to ask whether there are general principles, which achieve these kinds of refinements in semantic tableau calculi. In [19] we described a general condition for reducing the branching in inference rules without loosing completeness of calculi devised in the tableau synthesis framework. Because this condition is inductive, at present it needs to be checked manually and it is open whether it can be checked automatically.

In this paper we extend the possibilities of refining inference rules, thereby making progress toward the aim of automating rule refinement in the tableau synthesis framework. We describe two new approaches to satisfy the general rule refinement condition of the tableau synthesis framework. For the first approach we introduce *atomic rule refinement* as a specialisation of the general rule refinement technique with the advantage that it is syntactic and therefore automatic. This guarantees that atomic rule refinement preserves constructive completeness of a tableau calculus. In the second approach we show how to extend a set of non-refinable rules by altering the semantic specification of the logic and obtain a modified set of rules which can be refined. The approaches are illustrated on first-order frame conditions of modal logics, and a tableau calculus for the modal logic $\mathrm{K}_m(\neg)$ of 'some', 'all' and 'only' [12]. This logic is an extension of basic multi-modal logic $\mathrm{K}_m$ allowing negation on accessibility relations.

The paper is structured as follows. In the next two sections we sketch the main ideas of the tableau synthesis framework [19] and two existing refinements: general rule refinement and internalisation refinement. In Sect. 4 we introduce and investigate *atomic rule refinement*, which we show preserves constructive completeness and illustrate its usefulness in several examples. In Sect. 5 we show how atomic rule refinement can be used to construct hypertableau-like calculi. In the final section we apply the presented techniques to the extended modal logic $\mathrm{K}_m(\neg)$.

## 2 The Tableau Synthesis Framework

The tableau synthesis framework provides a method for systematically deriving a tableau calculus for a propositional logic $L$ [19]. In the following we give a minimal description of the main ingredients, the tableau language and tableau formulae in the calculi obtained using the method.

As the generated calculi are designed to construct models, the formulae in them are expressed in a *meta-language* $\mathsf{FO}(L)$ that extends the *object language* $\mathcal{L}$ of the logic with extra symbols sufficient to define models and truth valuations of formulas. Consider, for example, the basic modal logic $\mathrm{K}_m$ with multiple modalities. The *object language* is a two-sorted language in which the formulae are defined by the BNF $\phi \stackrel{\mathsf{def}}{=} p \mid \neg\phi \mid \phi \vee \phi \mid [r]\phi$ in sort $\mathsf{f}$, where $r$ is a variable or

$$\frac{\nu_{\mathsf{f}}(\neg p, x)}{\neg\nu_{\mathsf{f}}(p, x)} \quad \frac{\neg\nu_{\mathsf{f}}(\neg p, x)}{\nu_{\mathsf{f}}(p, x)} \quad \frac{\nu_{\mathsf{f}}(p \vee q, x)}{\nu_{\mathsf{f}}(p, x) \mid \nu_{\mathsf{f}}(q, x)} \quad \frac{\neg\nu_{\mathsf{f}}(p \vee q, x)}{\neg\nu_{\mathsf{f}}(p, x),\ \neg\nu_{\mathsf{f}}(q, x)} \quad \frac{\nu_{\mathsf{f}}([r]p, x)}{\neg\nu_{\mathsf{r}}(r, x, y) \mid \nu_{\mathsf{f}}(p, y)}$$

$$\frac{\neg\nu_{\mathsf{f}}([r]p, x)}{\nu_{\mathsf{r}}(r, x, f(r, p, x)),\ \neg\nu_{\mathsf{f}}(p, f(r, p, x))} \quad \frac{\nu_{\mathsf{f}}(p, x),\ \neg\nu_{\mathsf{f}}(p, x)}{\bot} \quad \frac{\nu_{\mathsf{r}}(r, x, y),\ \neg\nu_{\mathsf{r}}(r, x, y)}{\bot}$$

**Fig. 1.** Generated tableau calculus $T_{\mathrm{K}_m}$ for $\mathrm{K}_m$

constant over the second sort $\mathsf{r}$. The *meta-language* extends the object language with a domain sort $\mathsf{D}$ (for the domain of interpretation), and two designated predicate symbols $\nu_{\mathsf{f}}$ and $\nu_{\mathsf{r}}$ (the holds predicates) plus the connectives of first-order logic and the equality predicate $\approx$. The language is expressive enough to define the semantics of basic modal logic $\mathrm{K}_m$ as follows.

$$\forall x\ (\nu_{\mathsf{f}}(\neg p, x) \leftrightarrow \neg\nu_{\mathsf{f}}(p, x)) \qquad \forall x\ (\nu_{\mathsf{f}}(p \vee q, x) \leftrightarrow \nu_{\mathsf{f}}(p, x) \vee \nu_{\mathsf{f}}(q, x))$$

$$(1) \qquad \forall x\ (\nu_{\mathsf{f}}([r]p, x) \leftrightarrow \forall y\ (\nu_{\mathsf{r}}(r, x, y) \rightarrow \nu_{\mathsf{f}}(p, y)))$$

Intuitively, $\nu_{\mathsf{f}}(p, x)$ can be read as '$p$ is true in the world $x$' (or formally $x \in p^{\mathcal{I}}$), and $\nu_{\mathsf{r}}(r, x, y)$ as '$y$ is an $r$-successor of $x$' (or $(x, y) \in r^{\mathcal{I}}$). Thus we can read (1) as saying: $[r]p$ is true in $x$ iff for any $r$-successor $y$ of $x$, $p$ is true in $y$.

The stages in the tableau synthesis method are *synthesis*, *refinement* and *blocking*. The synthesis stage will transform the semantic specification of a logic such as the above into a tableau calculus $T_L$. The tableau calculus $T_{\mathrm{K}_m}$ produced for modal logic $\mathrm{K}_m$ is given in Fig. 1. This calculus allows reasoning in the semantics of the logic and we can use it for testing the (un)satisfiability of $\mathrm{K}_m$-formulae, and for model building.

The actual creation of the tableau calculus is not important for this paper, only that we have a sound and complete semantic tableau calculus at hand. When certain well-definedness conditions are true for the semantic definition of a logic the generated tableau calculus $T_L$ is sound and constructively complete [19]. A tableau calculus is *sound* when for a satisfiable set of tableau formulae any fully expanded tableau derivation has an open branch. A tableau calculus is *constructively complete*, if from every open fully expanded branch an interpretation can be constructed that validates all formulae on the branch. This interpretation is the *canonical interpretation* denoted by $\mathcal{I}(\mathcal{B})$.

Because the rule language and the initial calculus is heavily laden with meta-language notation, a crucial second stage in the method is the *refinement* stage. This is described in the next section. The paper is a contribution to this stage.

The third stage involves adding some form of *blocking* or *loop checking* mechanism to ensure termination or find models for finitely satisfiable input. For different modal and description logics various standard blocking mechanisms have been developed. In the tableau synthesis framework, blocking is realised by the use of equality-based blocking, which can be incorporated through additional inference rules, and is independent of the tableau calculus or the logic. Refinements of equality reasoning and equality-based blocking in semantic tableau-like approaches have been studied in [14, 4, 18, 21].

## 3 Refinement Techniques

The refinement stage of the tableau synthesis method involves two refinements: *rule refinement* and *internalisation.*

*Rule refinement* addresses the problem of reducing branching in inference rules by turning conclusions into premises [19]. Suppose $T_L$ is a sound and constructively complete tableau calculus for a logic $L$ and suppose $\rho$ is a tableau rule in $T_L$. Suppose $\rho$ has the form $\rho \overset{\mathsf{def}}{=} X_0/X_1 \mid \cdots \mid X_m$, where each $X_i$ is a set $\{\psi_1, \ldots, \psi_k\}$ of formulae. For simplicity and without loss of generality, we assume the aim is to refine away the first denominator $X_1$.

Let the rules $\rho_j$ with $j = 1, \ldots, k$ be defined by

$$\rho_j \overset{\mathsf{def}}{=} \frac{X_0 \cup \{\sim\psi_j\}}{X_2 \mid \cdots \mid X_m},$$

where $\sim$ denotes complementation, i.e., $\sim\phi = \psi$, if $\phi = \neg\psi$, and $\sim\phi = \neg\psi$, otherwise. Each rule $\rho_j$ is obtained from the rule $\rho$ by removing the first denominator $X_1$ and adding the complement of one of the formulae in $X_1$ as a premise. Intuitively, we may think of the refined rules as incorporating a look-ahead and branch closure, since when $\sim\psi_j$ is on the branch then the branch $X_1$ can be immediately closed. Note that there is however no guarantee that the formulae $\sim\psi_j$ are actually on the branch, even though there may be enough information so that $\mathcal{I}(\mathcal{B}) \not\models \psi_j$ and thus $\mathcal{I}(\mathcal{B}) \not\models X_1$ for particular instances, where $\mathcal{I}(\mathcal{B})$ is the canonical interpretation associated with the current (partial) branch $\mathcal{B}$. We say a branch $\mathcal{B}$ is *reflected* in the canonical interpretation $\mathcal{I}(\mathcal{B})$, if $\mathcal{I}(\mathcal{B})$ validates all formulae occurring on the branch $\mathcal{B}$.

Let $\mathsf{Ref}(\rho, T_L)$ denote the *refined tableau calculus* obtained from $T_L$ by replacing the rule $\rho$ with the rules $\rho_1, \ldots, \rho_k$. We say that $\mathsf{Ref}(\rho, T_L)$ is the *($\rho$-)rule refinement* of $T_L$. One can show that each rule $\rho_j$ is derivable [10] in $T_L$ and this implies that the calculus $\mathsf{Ref}(\rho, T_L)$ is sound. In general, $\mathsf{Ref}(\rho, T_L)$ is neither constructively complete nor complete. Nevertheless, the following theorem holds [19].

**Theorem 1.** *Let $T_L$ be a tableau calculus which is sound and constructively complete for the logic $L$. Let $\rho$ be the rule $X_0/X_1 \mid \cdots \mid X_m$ in $T_L$ and suppose $\mathsf{Ref}(\rho, T_L)$ is the rule refinement of $T_L$. Further, suppose $\mathcal{B}$ is an open branch in a $\mathsf{Ref}(\rho, T_L)$-tableau derivation and for every set $Y$ of $\mathcal{L}$-formulae from $\mathcal{B}$ the following holds. Then, $\mathcal{B}$ is reflected in the interpretation $\mathcal{I}(\mathcal{B})$ induced by $\mathcal{B}$.*

**General rule refinement condition:** *If all formulae in $Y$ are reflected in $\mathcal{I}(\mathcal{B})$ then for any $E_1, \ldots, E_l \in Y$ and any domain terms $t_1, \ldots, t_n$*

*if $X_0(\overline{E}, t_1, \ldots, t_n) \subseteq \mathcal{B}$ and $\mathcal{I}(\mathcal{B}) \not\models X_1(\overline{E}, \|t_1\|, \ldots, \|t_n\|)$*
*then $X_i(\overline{E}, t_1, \ldots, t_n) \subseteq \mathcal{B}$ for some $i = 2, \ldots, m$.*

*$X_i(\overline{E}, t_1, \ldots, t_n)$ denotes the set of instances of the formulae in $X_i$ under uniform substitution of $E_1, \ldots, E_l$ and $t_1, \ldots, t_n$ for $p_1, \ldots, p_l$ and $x_1, \ldots, x_n$, respectively, where $p_1, \ldots, p_l$ and $x_1, \ldots, x_n$ are respectively all the $\mathcal{L}$-variables and all the domain variables occurring in the rule $\rho$. The notation $\|t_i\|$ denotes the equivalence classes of terms modulo the equational theory defined by the term equalities encountered on the branch.*

The general rule refinement condition states that if there is information in the branch $\mathcal{B}$ to exclude $X_1(\overline{E}, t_1, \ldots, t_n)$ from holding in the model $\mathcal{I}(\mathcal{B})$ constructed from $\mathcal{B}$ then all the formulae of at least one of the other denominators of the rule are on the branch $\mathcal{B}$. In [19] a weaker condition for rule refinement is given,[1] but the condition of Theorem 1 is sufficient for the results of this paper. A consequence of the theorem is the following.

**Corollary 1.** *If the general rule refinement condition of Theorem 1 holds for every open branch $\mathcal{B}$ of any fully expanded $\mathsf{Ref}(\rho, T_L)$-tableau then the refined calculus $\mathsf{Ref}(\rho, T_L)$ is constructively complete for the logic $L$.*

The generalisation of this refinement which turns more than one denominator of a rule into premises is not difficult.

As an example of rule refinement let us consider the $(\mathsf{box})$ rule obtained from (1) in the tableau synthesis framework. Rule refinement gives (something close to) the usual box rule $(\Box)$.

$$(\mathsf{box})\ \frac{\nu_{\mathsf{f}}([r]p, x)}{\neg\nu_{\mathsf{r}}(r, x, y) \mid \nu_{\mathsf{f}}(p, y)} \qquad\qquad (\Box)\ \frac{\nu_{\mathsf{f}}([r]p, x),\ \nu_{\mathsf{r}}(r, x, y)}{\nu_{\mathsf{f}}(p, y)}$$

It can be proved directly that the general rule refinement condition is true in any branch of the refined calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ of the generated calculus $T_{\mathrm{K}_m}$ of basic modal logic $\mathrm{K}_m$. By Corollary 1 the refined calculus is therefore constructively complete.

**Theorem 2.** *The tableau calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ is sound and constructively complete for basic multi-modal logic $\mathrm{K}_m$.*

In Sect. 6 we give an example where this rule refinement is not possible in an extension of the logic $\mathrm{K}_m$. In fact, the general rule refinement condition is too strong to hold generally, because tableau calculi do not include introduction rules (just elimination rules), but we give examples in Sects 4 and 6 where the refinement condition does hold.

*Internalisation refinement* in the tableau synthesis process involves eliminating some of the extra-logical notation, by expressing the rules in a tableau language as close as possible to the language of the logic. In particular, the internalisation involves reduction of the calculus to one where the holds predicates $\nu_{\mathsf{s}}$ have been eliminated and the domain sort symbols are expressed in the language of the logic, provided this is possible. The idea is that each atomic

[1] The general rule refinement condition given here corresponds to condition (‡) in [19].

$$\frac{@_i p,\ @_i\neg p}{\bot} \qquad \frac{@_i\neg\neg p}{@_i p} \qquad \frac{@_i(p \vee q)}{@_i p \mid @_i q} \qquad \frac{@_i\neg(p \vee q)}{@_i\neg p,\ @_i\neg q}$$

$$\frac{@_i[r]p,\ @_i\neg[r]\neg j}{@_j p} \qquad \frac{@_i\neg[r]p}{@_i\neg[r]\neg f(r,p,i),\ @_{f(r,p,i)}\neg p}$$

**Fig. 2.** Refined tableau calculus $T^{\mathsf{ref}}_{\mathrm{K}_m}$.

formula $\nu_{\mathsf{s}}(E, \overline{a})$ in the tableau calculus is replaced by a suitable formula of the logic, and all syntactically redundant rules are removed from the calculus.

For example, if the logic $L$ contains nominals and the @ connective of hybrid logic [5] then the elements of the domain sort $\mathsf{D}$ can be identified with nominals and the formulae $\nu_{\mathsf{f}}(\phi, v)$ and $\nu_{\mathsf{r}}(\alpha, v, w)$ can be internalised as the formulae $@_v\phi$ and $@_v\neg[\alpha]\neg w$ respectively, where $v$ and $w$ have become nominals. The refined and internalised calculus for basic modal logic $\mathrm{K}_m$ is given in Fig. 2.

If the logic is not expressive enough then an option is to simplify the notation of the rules by reformulating them using labels and the ':' connective (of varying arity), to rephrase the rules in notation more familiar from the literature (alternative notations also exist).

The internalisation refinement simplifies the tableau language and, in many cases, reduces the number of the rules in the tableau calculus. In our experience it is easiest and produces better results, if rule refinement is performed first, followed by the internalisation refinement.

## 4 Atomic Rule Refinement

In this section we introduce the technique of *atomic rule refinement*. Under this refinement, formulae in the conclusions are only moved upwards to premise positions if *the formulae are negated $\mathcal{L}$-atomic formulae* in the language $\mathsf{FO}(\mathrm{L})$.

By definition, a $\mathsf{FO}(L)$-formula $\phi$ is *$\mathcal{L}$-atomic* if it is an atomic formula of $\mathsf{FO}(L)$ and all occurrences of $\mathcal{L}$-formulae in $\phi$ are also atomic. Thus, $\nu_{\mathsf{s}}(E, \overline{t})$ is $\mathcal{L}$-atomic only if $E$ is an atomic formula of $\mathcal{L}^{\mathsf{s}}$. For example, the formulae $\nu_{\mathsf{f}}(p, x)$ and $\nu_{\mathsf{r}}(r, f(r, p, x), x)$ are $\mathcal{L}(\mathrm{K}_m)$-atomic, but the formulae $\neg\nu_{\mathsf{f}}(p, x)$, $\nu_{\mathsf{f}}(\neg p, x)$ and $\nu_{\mathsf{f}}(p \vee q, x)$ are not. The respective reasons are that $\neg\nu_f(p, x)$ is a negated $\mathcal{L}(\mathrm{K}_m)$-atomic formula and $\nu_{\mathsf{f}}(\neg p, x)$ and $\nu_{\mathsf{f}}(p \vee q, x)$ are not $\mathcal{L}$-atomic in $\mathsf{FO}(\mathrm{K}_m)$.

Using the notation and assumptions of Theorem 1, we can prove:

**Theorem 3.** *Assume that for an open branch $\mathcal{B}$ of the refined tableau calculus $\mathsf{Ref}(\rho, T_L)$ and for every set $Y$ of $\mathcal{L}$-formulae from $\mathcal{B}$ the following holds. Then, $\mathcal{B}$ is reflected in $\mathcal{I}(\mathcal{B})$.*

**Atomic rule refinement condition:** *If all formulae in $Y$ are reflected in $\mathcal{I}(\mathcal{B})$ then for any $E_1, \ldots, E_l \in Y$ and any domain terms $t_1, \ldots, t_n$,*

$$X_0(\overline{E}, t_1, \ldots, t_n) \subseteq \mathcal{B} \text{ implies that}$$

$$X_1(\overline{E}, t_1, \ldots, t_n) = \{\neg\xi_1, \ldots, \neg\xi_k\} \text{ and all } \xi_1, \ldots, \xi_k \text{ are } \mathcal{L}\text{-atomic.}$$

Unlike the general rule refinement condition, the atomic rule refinement condition is purely syntactic and, thus, can be automatically checked against each given open branch $\mathcal{B}$. However, even if all the formulae from $X_1$ are negated $\mathcal{L}$-atomic formulae, their instantiation within a branch of a tableau derivation can, in general, produce a formula which is not a negated $\mathcal{L}$-atom. Therefore, similar to Corollary 1, by Theorem 3, in order to preserve constructive completeness we need to make sure that the atomic rule refinement condition holds for every branch of any derivation in the refined calculus.

**Corollary 2.** *If the assumptions and condition of Theorem 3 holds for every open branch $\mathcal{B}$ of any fully expanded $\mathsf{Ref}(\rho, T_L)$-tableau then the refined calculus $\mathsf{Ref}(\rho, T_L)$ is constructively complete for the logic $L$.*

In the following we give several examples of atomic rule refinement.

**Example 1.** The refinement $(\Box)$ of the rule $(\mathsf{box})$ mentioned in the previous section is an example of an atomic rule refinement. Because any instantiation of $\nu_{\mathsf{r}}(r, x, y)$ in the language of $\mathrm{K}_m$ is an $\mathcal{L}(\mathrm{K}_m)$-atomic formula, constructive completeness of the refined calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ follows from Corollary 2.

**Example 2.** Suppose we wish to impose that *one* accessibility relation $r$ of our modal logic is irreflexive, i.e., we specify that $\forall x\, \neg\nu_{\mathsf{r}}(r, x, x)$. This generates the rule $/\neg\nu_{\mathsf{r}}(r, x, x)$.[2] Using atomic rule refinement the rule can be refined to the following closure rule

$$(\mathsf{irr})\ \frac{\nu_{\mathsf{r}}(r, x, x)}{\bot}, \qquad \text{or in internalised form the rule} \qquad \frac{@_i\neg[r]\neg i}{\bot}\ .$$

**Example 3.** If we wish to specify that *all* relations are irreflexive, atomic rule refinement allows us to use the rule $\nu_{\mathsf{r}}(r, x, x)/\bot$. Because the language of $\mathrm{K}_m$ contains only atomic relations $r_1, \ldots, r_m$ and no relational connectives, any instantiation of $r$ and variable $x$ in $\nu_{\mathsf{r}}(r, x, x)$ produces only $\mathcal{L}(\mathrm{K}_m)$-atomic formulae of the form $\nu_{\mathsf{r}}(r_i, t, t)$ (where $t$ is a term of the domain sort). Therefore, the atomic rule refinement condition is true for any branch of any tableau derivation in the calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ extended with the $(\mathsf{irr})$ rule. Thus, by Corollary 2, the calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ extended with the $(\mathsf{irr})$ rule is sound and constructively complete for the logic $\mathrm{K}_m$ with irreflexive relations.

Applying the internalisation refinement we obtain the following theorem for the labelled tableau calculus.

**Theorem 4.** *$T^{\mathsf{ref}}_{\mathrm{K}_m}$ extended with the rule $@_i\neg[r]\neg i/\bot$ for each irreflexive relation $r$ in $\mathrm{K}_m$ is sound and constructively complete (or, where $r$ denotes a variable, for the case that each relation in the logic is irreflexive).*

[2] In the framework the rule would have a premise involving domain predication, but in this paper we silently assume domain predication without making it explicit in the interest of simplicity of presentation, see [19] for details.

**Example 4.** The following frame condition from [2] states the *existence of an immediate predecessor* for every element in a model.

$$\forall x \exists y \forall z \Big( \nu_{\mathsf{r}}(r, y, x) \wedge x \not\approx y \wedge \big( (\nu_{\mathsf{r}}(r, y, z) \wedge \nu_{\mathsf{r}}(r, z, x)) \rightarrow (z \approx x \vee z \approx y) \big) \Big)$$

We first reduce the formula to a form acceptable in the tableau synthesis framework. Let $g$ be a new Skolem function which depends on two arguments, one of the sort $\mathsf{r}$ and one from the domain sort. The existential quantifier is eliminated from the frame conditions and decomposed to give three formulae (see [19]):

$$\forall x\ \nu_{\mathsf{r}}(r, g(r,x), x), \qquad \forall x\ (x \not\approx g(r,x)),$$
$$\forall x \forall z\ \big( (\nu_{\mathsf{r}}(r, g(r,x), z) \wedge \nu_{\mathsf{r}}(r, z, x)) \rightarrow (g(r,x) \approx z \vee z \approx x) \big) \ .$$

From these formulae three rules are generated:

$$\frac{}{\nu_{\mathsf{r}}(r, g(r,x), x)}, \qquad \frac{}{x \not\approx g(r,x)},$$
$$\frac{}{\neg\nu_{\mathsf{r}}(r, g(r,x), z) \mid \neg\nu_{\mathsf{r}}(r, z, x) \mid g(r,x) \approx z \mid z \approx x} \ .$$

Atomic rule refinement is not applicable to the first rule since the conclusion is not negated. Consider the second and third rules. Applying the same argument as in Example 3 above we find that no instantiation of $x \approx g(r,x)$, $\nu_{\mathsf{r}}(r, g(r,x), z)$, and $\nu_{\mathsf{r}}(r, z, x)$ within the language $\mathsf{FO}(\mathrm{K}_m)$ produces a formula which is not $\mathcal{L}(\mathrm{K}_m)$-atomic. This means the atomic rule refinement condition holds for these rules. Refining the second rule once and the third rule twice, the rules

$$\frac{x \approx g(r,x)}{\perp} \qquad \text{and} \qquad \frac{\nu_{\mathsf{r}}(r, g(r,x), z),\ \nu_{\mathsf{r}}(r, z, x)}{g(r,x) \approx z \mid z \approx x}$$

are obtained. By Corollary 2, constructive completeness of any tableau calculus in the language $\mathsf{FO}(\mathrm{K}_m)$ is preserved under these refinements. Internalising $\mathsf{FO}(\mathrm{K}_m)$ in the hybrid logic extension of $\mathrm{K}_m$ we obtain the following theorem.

**Theorem 5.** *$T^{\mathit{ref}}_{\mathrm{K}_m}$ extended with the rules*

$$\frac{}{@_{g(r,i)}\neg[r]\neg i}, \qquad \frac{@_i g(r,i)}{\perp} \qquad \textit{and} \qquad \frac{@_{g(r,i)}\neg[r]\neg j,\ @_j\neg[r]\neg i}{@_{g(r,i)} j \mid @_j i}$$

*is sound and constructively complete for $\mathrm{K}_m$ over the class of models satisfying the frame condition of existence of an immediate predecessor.*

The use of Skolem terms is not in agreement with common, present practice in the area, but they provide a useful technical device to enhance the scope of semantic tableau approaches by accommodating properties and specifications with negative occurrences of existential quantification, which produce rules where these occurrences appear in premise positions, cf. Example 4. This easily accommodates non-geometric theories. Skolem terms also allow for effective implementation of blocking and equality reasoning, since, e.g., no inference steps need to be recomputed when blocking occurs (cf. the comments in [14]).

Examples 3 and 4 are important because they show that atomic rule refinement allows *automatic refinement of tableau rules generated from frame conditions of modal logics*. Furthermore, the case of the last rule in Example 4 is a particularly clear illustration of the benefits of rule refinement. In that case the unrefined rule is applicable for every pair of domain elements and creates four branches on application, whereas the refined rule replacing it, is only applied to formulae matching two premises, and then creates only two branches. This *constraining effect on the search space* is an important benefit of rule refinement. In general, using the fairness requirements for tableau derivations, it is possible to map each refined derivation to its unrefined counterpart where each rule application is either mapped to itself or to the application of the corresponding unrefined rule. Since more premises need to be satisfied, the refined rule will be applied less often and each of its applications produces fewer branches than the corresponding point of the unrefined tableau. Thus, each refined derivation (in other words, the search space) is smaller than its unrefined counterpart.

Another important point is the *incrementality* of the technique: the rules can be refined one by one without affecting the refinability of other rules. It is therefore *more flexible*, *robust* and *useful* than general rule refinement, of which it is a special case.

Because of these attractive features we have used atomic rule refinement in other recent work. In [23] we applied the tableau synthesis framework and atomic rule refinement in the creation of terminating tableau calculi for a bi-intuitionistic logic with interacting modal operators, called BISKT. This logic can be equivalently embedded into a tense logic $Kt(H, R)$ with several interacting modalities [18] via an extension of the standard embedding of intuitionistic propositional logic into modal logic S4. $Kt(H, R)$ was subject to an investigation of the numerous possibilities of defining tableau calculi for modal logics, and their relative efficiency [18]. Interestingly, we found that the tableau calculi of BISKT [23] exhibited better performance than those of $Kt(H, R)$ [18], which we attribute to the greater constraining power of atomic rule refinement in the style of calculus used for BISKT.

In [14] we used atomic rule refinement to obtain a tableau calculus with dynamically generated hypertableau-like inference rules for description logic ontologies. In particular, the standard inference rule $/@_i\alpha$ for TBox statements $\alpha$, which hold universally, is replaced by a set of refined rules for each statement in the TBox. E.g., for the statement $A_1 \sqcap A_2 \sqsubseteq B$ the specifically generated rule is $@_iA_1, @_iA_2/@_iB$, where $A_1, A_2, B$ are atomic concepts. For satisfiable and unsatisfiable inputs, the evaluation results showed improved performance for this refinement. The speed-up was particularly marked for unsatisfiable inputs (2.5–6 times faster on average), which was found to be mainly due to the presence of additional closure rules such as $@_iA, @_iB/\bot$ generated from the disjointness statement $A \sqcap B \sqsubseteq \bot$, where $A$ and $B$ are atomic concepts. The results also showed a 22% (and 74%) drop in memory use for satisfiable (and unsatisfiable) inputs when using refined rules. The essential idea in this work is generalised in the next section.

## 5 Hypertableau

Let the given logic $L$ have disjunction-like connectives $\vee$ and negation-like connectives $\neg$ for some sort $\mathsf{s}$ of the logic. Assume $T_L$ is a tableau calculus sound and constructively complete for $L$ and contains the rules

$$\frac{\nu_{\mathsf{s}}(p \vee q, \overline{x})}{\nu_{\mathsf{s}}(p, \overline{x}) \mid \nu_{\mathsf{s}}(q, \overline{x})} \quad \text{and} \quad \frac{\nu_{\mathsf{s}}(\neg p, \overline{x})}{\neg \nu_{\mathsf{s}}(p, \overline{x})},$$

which are the usual rules for disjunction and negation. We transform the synthesised calculus $T_L$ into a new calculus $T_L^{\mathsf{hyp}}$ in three steps. For simplicity we assume that disjunction in $L$ is associative and commutative with respect to satisfiability, that is, the following statements are entailed by the semantic specification of $L$.

$$\nu_{\mathsf{s}}(p \vee q, \overline{x}) \leftrightarrow \nu_{\mathsf{s}}(q \vee p, \overline{x}) \qquad \nu_{\mathsf{s}}((p \vee q) \vee r, \overline{x}) \leftrightarrow \nu_{\mathsf{s}}(p \vee (q \vee r), \overline{x})$$

This assumption is not essential for the transformation but allows us to flatten disjunctions and avoid a combinatorial blow-up.

In the first step of the transformation, the usual disjunction rule $\nu_{\mathsf{s}}(p \vee q, \overline{x})/\nu_{\mathsf{s}}(p, \overline{x}) \mid \nu_{\mathsf{s}}(q, \overline{x})$ is replaced by the set of the rules (for $k > 1$):

$$(\mathsf{split}_k) \ \frac{\nu_{\mathsf{s}}(p_1 \vee \cdots \vee p_k, \overline{x})}{\nu_{\mathsf{s}}(p_1, \overline{x}) \mid \cdots \mid \nu_{\mathsf{s}}(p_k, \overline{x})} \ .$$

We denote by $T_L^{\mathsf{sp}}$ a tableau calculus obtained from $T_L$ by replacing the usual disjunction rule by the rules $(\mathsf{split}_k)$. The $(\mathsf{split}_k)$ rules and the usual disjunction rule are derivable from each other. Therefore, the transformed calculus $T_L^{\mathsf{sp}}$ is sound and constructively complete.

For the second step consider the following rules (for $m + n > 1$)

$$(\mathsf{split}^+_{mn}) \ \frac{\nu_{\mathsf{s}}(\neg p_1 \vee \cdots \vee \neg p_m \vee q_1 \vee \cdots \vee q_n, \overline{x})}{\neg\nu_{\mathsf{s}}(p_1, \overline{x}) \mid \cdots \mid \neg\nu_{\mathsf{s}}(p_m, \overline{x}) \mid \nu_{\mathsf{s}}(q_1, \overline{x}) \mid \cdots \mid \nu_{\mathsf{s}}(q_n, \overline{x})}$$

with the side-condition that only atomic substitutions are allowed for $p_1, \ldots, p_m$. Note, $m$ is the maximal number of negated atoms in the disjunction which match the premise. That is, the rules are applicable only to formulae of the shape $\nu_{\mathsf{s}}(\neg E_1 \vee \cdots \vee \neg E_m \vee F_1 \vee \cdots \vee F_n, \overline{x})$, where all $E_1, \ldots, E_m$ are *atomic* formulae of the logic $L$ and no $F_1, \ldots, F_n$ is a negated atomic formula of $L$.

Let $T_L^{\mathsf{sp}+}$ be a tableau calculus obtained from $T_L^{\mathsf{sp}}$ by replacing the rules $(\mathsf{split}_k)$ by the rules $(\mathsf{split}^+_{mn})$. The rules $(\mathsf{split}_k)$ and $(\mathsf{split}^+_{mn})$ are derivable from each other and, thus, the following theorem holds.

**Theorem 6.** *$T_L^{\mathsf{sp}+}$ is sound and constructively complete for the logic $L$.*

Now we are in a position to use atomic rule refinement to refine the rules $(\mathsf{split}^+_{mn})$ to the rules $(m + n > 1)$

$$(\mathsf{hyp}_{mn}) \ \frac{\nu_{\mathsf{s}}(\neg p_1 \vee \cdots \vee \neg p_m \vee q_1 \vee \cdots \vee q_n, \overline{x}),\ \nu_{\mathsf{s}}(p_1, \overline{x}),\ \cdots,\ \nu_{\mathsf{s}}(p_m, \overline{x})}{\nu_{\mathsf{s}}(q_1, \overline{x}) \mid \cdots \mid \nu_{\mathsf{s}}(q_n, \overline{x})}$$

with the restriction that only atomic substitutions are allowed for $p_1, \ldots, p_m$. These are hypertableau-like rules. Similarly to the rules in the previous step, an application of the rule $(\mathsf{hyp}_{mn})$ is allowed only to formulae of the shape $\nu_{\mathsf{s}}(\neg E_1 \vee \cdots \vee \neg E_m \vee F_1 \vee \cdots \vee F_n)$, where all $E_1, \ldots, E_m$ are atomic formulae and no $F_1, \ldots, F_n$ are negated atomic formulae of the logic $L$. Notice that in the case of $n = 0$ the rules $(\mathsf{hyp}_{mn})$ are atomic closure rules.

Let $T_L^{\mathsf{hyp}}$ be the calculus obtained from $T_L^{\mathsf{sp}}$ by adding the $(\mathsf{hyp}_{mn})$ rules. By Corollary 2 and Theorem 6 we obtain constructive completeness of $T_L^{\mathsf{hyp}}$.

**Theorem 7.** *$T_L^{\mathsf{hyp}}$ is sound and constructively complete for the logic $L$.*

Thus, for any (propositional) logic $L$ with disjunction and negation connectives and any sound and constructive complete calculus for $L$ with the usual disjunction and negation rules, it is possible to devise a hypertableau-like calculus that is sound and constructively complete for the logic $L$.

Derivations in $T_L^{\mathsf{hyp}}$ can be done more efficiently if the logic $L$ has additional properties. We already assume associativity and commutativity of disjunction. Suppose now that the satisfiability of formulae in a subset of the language $\mathcal{L}$ is reducible to the satisfiability of formulae in conjunctive normal form:

$$\nu_{\mathsf{s}}(E, \overline{x}) \leftrightarrow \bigwedge_{i=1}^{I} \nu_{\mathsf{s}_{ij}}(\bigvee_{j=1}^{J_i} E_{ij}, \overline{x}) \qquad \text{where } \mathsf{s} \text{ and } \mathsf{s}_{ij} \text{ are sorts of the logic } .$$

Thus, every formula $E$ has an equi-satisfiable clausal representation as a set of clauses $C_1, \ldots, C_I$, where $C_i = E_{i1} \vee \cdots \vee E_{iJ_i}$ for each $i = 1, \ldots, I$. Since disjunction is associative and commutative, we can assume that, in every clause, all negated atomic formulae (negative literals) of the logic appear before all other formulae. Let $\mathcal{A}$ be a reduction algorithm which transforms any formula $E$ into such equi-satisfiable clausal normal form.

Cases of logics become interesting when there are many clauses with negated atomic formulae, because then the $(\mathsf{hyp}_{mn})$ rules with $m > 0$ are applied more frequently in derivations in $T_L^{\mathsf{hyp}}$. Since the $(\mathsf{hyp}_{mn})$ rules with $m > 0$ create fewer branching points in derivations than the $(\mathsf{hyp}_{mn})$ rules with $m = 0$, derivations in $T_L^{\mathsf{hyp}}$ will have fewer branches and therefore performance is enhanced.

The conclusions of the $(\mathsf{hyp}_{mn})$ rules are allowed to contain non-atomic $\mathcal{L}$-formulae which have to be decomposed further by other rules of the calculus. For the conclusions of other rules, we have two alternatives. One is to use the rules of the tableau calculus to decompose their formulae up to atomic components. The other alternative is to apply the reduction algorithm $\mathcal{A}$ to every new conclusion of any rule different from the $(\mathsf{hyp}_{mn})$ rules. The first alternative uses the decomposition rules of the tableau calculus (assuming it includes rules for conjunction and disjunction) and the second one uses the algorithm $\mathcal{A}$. In the implementation of a prover these two alternatives have to be carefully balanced, depending on the complexity of the algorithm $\mathcal{A}$ and how efficiently it is implemented. There is an efficient clausification algorithm for Boolean parts which runs in polynomial time on the length of the input [17]. Thus, we can assume that every conclusion of a rule is immediately transformed into a set of clauses. This allows to omit all the rules for Boolean connectives except the hypertableau rules. We give an example of a hypertableau-style calculus in the next section.

## 6 Case Study: The Modal Logic of 'Some', 'All' and 'Only'

As an illustration of the usefulness and generality of the refinement techniques investigated in this paper, we apply them to the modal logic $\mathrm{K}_m(\neg)$ of 'some', 'all' and 'only' [12]. $\mathrm{K}_m(\neg)$ is the extension of the basic multi-modal logic $\mathrm{K}_m$ with the relational negation.

Following the tableau synthesis framework [19] the language $\mathcal{L}$ has two sorts: a sort $\mathsf{f}$ for formulae and a sort $\mathsf{r}$ for relations. Assuming the sort $\mathsf{r}$ is formed over a set of relational constants $\{a_1, \ldots, a_m\}$, in $\mathcal{L}$ every *relation* $\alpha$ is defined by the BNF $\alpha \stackrel{\mathsf{def}}{=} a_1 \mid \cdots \mid a_m \mid \neg\alpha$, where $\neg$ is a relational connective. The sort $\mathsf{f}$ is formed over a set of propositional variables $\{p, q, \ldots\}$ and every *formula* $\phi$ is defined by the BNF $\phi \stackrel{\mathsf{def}}{=} p \mid \neg\phi \mid \phi \vee \phi \mid [\alpha]\phi$, where $\alpha$ ranges over all relations in the language.

The semantic specification language $\mathsf{FO}(\mathrm{K}_m(\neg))$ for $\mathrm{K}_m(\neg)$ is a first-order language over the sorts $\mathsf{f}$ and $\mathsf{r}$ and an additional *domain sort* $\mathsf{D}$. Formulae of $\mathcal{L}$ are encoded in the obvious way as terms of the appropriate sorts in $\mathsf{FO}(\mathrm{K}_m(\neg))$. That is, every logical connective of $\mathcal{L}$ is represented by a function in $\mathsf{FO}(\mathrm{K}_m(\neg))$. Every propositional variable of $\mathcal{L}$ is an individual variable of the sort $\mathsf{f}$ in $\mathsf{FO}(\mathrm{K}_m(\neg))$. Besides the individual constants $a_1, \ldots, a_m$ for relations, the language $\mathsf{FO}(\mathrm{K}_m(\neg))$ has a countable set of relation variables $r, r', \ldots$. The additional sort $\mathsf{D}$ has a countable set of individual variables $x, y, z, \ldots$. Furthermore, the semantic specification language has two predicate symbols $\nu_\mathsf{f}$ and $\nu_\mathsf{r}$ of sort $(\mathsf{f}, \mathsf{D})$ and $(\mathsf{r}, \mathsf{D}, \mathsf{D})$, respectively, to encode satisfiability. The meaning of these symbols can be understood from the definitions given next. The semantic specification consists of the following formulae, one for each of the logical connectives of $\mathrm{K}_m(\neg)$.

$$\forall x\ (\nu_\mathsf{f}(\neg p, x) \leftrightarrow \neg\nu_\mathsf{f}(p, x)) \qquad \forall x\ (\nu_\mathsf{f}(p \vee q, x) \leftrightarrow \nu_\mathsf{f}(p, x) \vee \nu_\mathsf{f}(q, x))$$
$$\forall x \forall y\ (\nu_\mathsf{r}(\neg r, x, y) \leftrightarrow \neg\nu_\mathsf{r}(r, x, y)) \qquad \forall x\ (\nu_\mathsf{f}([r]p, x) \leftrightarrow \forall y\ (\nu_\mathsf{r}(r, x, y) \rightarrow \nu_\mathsf{f}(p, y)))$$

Compared to the specification of $\mathrm{K}_m$, the specification of $\mathrm{K}_m(\neg)$ is extended with the second clause in the left column, which defines relational negation.

The logic $\mathrm{K}_m(\neg)$ is interesting because of the presence of three quantifier operators. These are the necessity operator $[\alpha]$, the possibility operator $\neg[\alpha]\neg$ and a third operator, the sufficiency operator $[\neg\alpha]\neg$, sometimes referred to as the window operator. $\nu_\mathsf{f}([\alpha]\phi, v)$ can be read as saying '$\phi$ is true in *all* $\alpha$-successors', $\nu_\mathsf{f}(\neg[\alpha]\neg\phi, v)$ as '$\phi$ is true in *some* $\alpha$-successor', and $\nu_\mathsf{f}([\neg\alpha]\neg\phi, v)$ as '$\phi$ is true in *only* $\alpha$-successors of $v$'. $\mathrm{K}_m(\neg)$ is a sublogic of Boolean modal logic [9] and the description logics $\mathcal{ALBO}$ and $\mathcal{ALBO}^{\mathsf{id}}$ [20]. $\mathrm{K}_m(\neg)$ has the finite model property [9] but the tree model property fails for the logic (e.g. [15]). The results of [15] imply that the satisfiability problem in $\mathrm{K}_m(\neg)$ is $\mathsf{ExpTime}$-complete.

$$\frac{\nu_{\mathsf{f}}(\neg p, x)}{\neg\nu_{\mathsf{f}}(p, x)} \qquad \frac{\neg\nu_{\mathsf{f}}(\neg p, x)}{\nu_{\mathsf{f}}(p, x)} \qquad \frac{\nu_{\mathsf{f}}(p \vee q, x)}{\nu_{\mathsf{f}}(p, x) \mid \nu_{\mathsf{f}}(q, x)} \qquad \frac{\neg\nu_{\mathsf{f}}(p \vee q, x)}{\neg\nu_{\mathsf{f}}(p, x),\ \neg\nu_{\mathsf{f}}(q, x)}$$

$$\frac{\nu_{\mathsf{f}}([r]p, x)}{\neg\nu_{\mathsf{r}}(r, x, y) \mid \nu_{\mathsf{f}}(p, y)} \qquad \frac{\neg\nu_{\mathsf{f}}([r]p, x)}{\nu_{\mathsf{r}}(r, x, f(r, p, x)),\ \neg\nu_{\mathsf{f}}(p, f(r, p, x))}$$

$$\frac{\nu_{\mathsf{f}}(p, x),\ \neg\nu_{\mathsf{f}}(p, x)}{\bot} \qquad \frac{\nu_{\mathsf{r}}(r, x, y),\ \neg\nu_{\mathsf{r}}(r, x, y)}{\bot} \qquad \frac{\nu_{\mathsf{r}}(\neg r, x, y)}{\neg\nu_{\mathsf{r}}(r, x, y)} \qquad \frac{\neg\nu_{\mathsf{r}}(\neg r, x, y)}{\nu_{\mathsf{r}}(r, x, y)}$$

**Fig. 3.** Generated tableau calculus $T_{\mathrm{K}_m(\neg)}$ for $\mathrm{K}_m(\neg)$

The tableau calculus $T_{\mathrm{K}_m(\neg)}$ obtained from the semantic specification of $\mathrm{K}_m(\neg)$ in the tableau synthesis framework is given in Fig. 3. New compared to the generated tableau calculus for the basic modal logic $\mathrm{K}_m$ in Fig. 1 are the last two rules for relational negation. Because the semantic specification of $\mathrm{K}_m(\neg)$ is well-defined in the sense of [19], from Theorems 5.1 and 5.6 in that work, we immediately obtain the following result.

**Theorem 8 (Soundness and constructive completeness).** *The calculus $T_{\mathrm{K}_m(\neg)}$ is sound and constructively complete for the logic $\mathrm{K}_m(\neg)$.*

However, none of the rules of the tableau calculus for $\mathrm{K}_m(\neg)$ from Fig. 3 are refinable. In particular, the $(\mathsf{box})$ rule cannot be refined to the $(\Box)$ rule (as discussed in Sect. 4) without loosing constructive completeness. Take for instance the set of formulae $\{\nu_{\mathsf{f}}([\neg\neg r]p, a), \nu_{\mathsf{r}}(r, a, b), \neg\nu_{\mathsf{f}}(p, b)\}$. The set is not $\mathrm{K}_m(\neg)$-satisfiable but none of the rules of the refined calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m(\neg)})$ are applicable to the set.

A possibility for refinement is the atomic refinement of *instances* of rules. Atomic rule refinement would allow us to use the rule $(\Box)$ on formulae $[r]\phi$, where $r$ is bound to a relational constant. We would still need to use the rule $(\mathsf{box})$ when $r$ is bound to a complex relational formula (in this case a negated relational formula). This kind of refinement is generally possible, and will be useful in practice, but leads to an uneven treatment of box formulae. Better would be if all instances of a rule can be refined.

In fact, by a small amendment of the semantic specification it *is* possible to refine the $(\mathsf{box})$ rule generally, for *all* instances. Observe that the semantic specification of $\mathrm{K}_m(\neg)$ entails the following formula.

$$\forall x\ (\nu_{\mathsf{f}}([\neg r]p, x) \rightarrow \forall y\ (\neg\nu_{\mathsf{r}}(r, x, y) \rightarrow \nu_{\mathsf{f}}(p, y)))$$

This means the formula can be added to the semantic specification of $\mathrm{K}_m(\neg)$ without changing the class of models of the logic. We use the notation $T^{+}_{\mathrm{K}_m(\neg)}$ to refer to the tableau calculus generated from the semantic specification extended with this formula. $T^{+}_{\mathrm{K}_m(\neg)}$ consists of the rules listed in Fig. 3 and the rule:

$$([\neg])\ \frac{\nu_{\mathsf{f}}([\neg r]p, x)}{\nu_{\mathsf{r}}(r, x, y) \mid \nu_{\mathsf{f}}(p, y)}\ .$$

$$\frac{@_i p,\ @_i \neg p}{\bot} \qquad \frac{@_i \neg\neg p}{@_i p} \qquad \frac{@_i (p \vee q)}{@_i p \mid @_i q} \qquad \frac{@_i \neg(p \vee q)}{@_i \neg p,\ @_i \neg q} \qquad \frac{@_i [r]p,\ @_i \neg[r]\neg j}{@_j p}$$

$$\frac{@_i \neg[r]p}{@_i \neg[r]\neg f(r,p,i),\ @_{f(r,p,i)} \neg p} \qquad \frac{@_i \neg[\neg r]\neg j}{@_i [r]\neg j} \qquad \frac{@_i [\neg r]\neg j}{@_i \neg[r]\neg j} \qquad \frac{@_i [\neg r]p}{@_i \neg[r]\neg j \mid @_j p}$$

**Fig. 4.** Refined tableau calculus $T^{\mathsf{ref}}_{\mathrm{K}_m(\neg)}$.

We can check that the well-definedness conditions from [19] are satisfied for the extended semantic specification of $\mathrm{K}_m(\neg)$. Therefore, by the results of the tableau synthesis framework, the extended calculus $T^{+}_{\mathrm{K}_m(\neg)}$ is sound and constructively complete for $\mathrm{K}_m(\neg)$. Note, the rule $([\neg])$ is a derived rule in the calculus $T_{\mathrm{K}_m(\neg)}$.

While the rule $([\neg])$ neither satisfies the atomic nor the general rule refinement condition, the *general* rule refinement condition is now satisfied for the $(\mathsf{box})$ rule, and, thus, as a consequence of Corollary 1 we get:

**Theorem 9.** *The tableau calculus* $\mathsf{Ref}((\mathsf{box}), T^{+}_{\mathrm{K}_m(\neg)})$ *using the* $(\Box)$ *rule instead of the* $(\mathsf{box})$ *rule is sound and constructively complete for the logic* $\mathrm{K}_m(\neg)$.

The internalisation refinement is possible for the new calculus if nominals and the @ operator of hybrid logic [5] are introduced to the tableau language of $\mathrm{K}_m(\neg)$. This significantly strengthens the tableau language and allows all formulae $\nu_{\mathsf{f}}(\phi, a)$ and $\neg\nu_{\mathsf{f}}(\phi, a)$ to be replaced by the formulae $@_a\phi$ and $@_a\neg\phi$, respectively, and the formulae $\nu_{\mathsf{r}}(\alpha, a, b)$ and $\neg\nu_{\mathsf{r}}(\alpha, a, b)$ can be replaced respectively by the formulae $@_a\neg[\alpha]\neg b$ and $@_a[\alpha]\neg b$ (the latter is equivalent to $@_a\neg\langle\alpha\rangle b$). In this case the result of the refinement is a significantly simplified calculus, reminiscent of standard labelled tableau calculi. The obtained rules are listed in Fig. 4. We denote this calculus by $T^{\mathsf{ref}}_{\mathrm{K}_m(\neg)}$. By the results of this paper and [19] it is sound and constructively complete for $\mathrm{K}_m(\neg)$.

Because disjunction and negation in $\mathrm{K}_m(\neg)$ are Boolean, it is possible to devise a hypertableau calculus for $\mathrm{K}_m(\neg)$, see Fig. 5. By Theorem 7, this calculus is sound and constructively complete for $\mathrm{K}_m(\neg)$. In summary, we have:

**Theorem 10.** *The refined tableau calculi* $T^{\mathit{ref}}_{\mathrm{K}_m(\neg)}$ *and* $T^{\mathit{hyp}}_{\mathrm{K}_m(\neg)}$ *(of Figs. 4 and 5) are sound and constructively complete for the logic* $\mathrm{K}_m(\neg)$.

A further example of systematic rule refinement using the ideas of this paper is the description logic $\mathcal{ALBO}^{\mathsf{id}}$, for which we presented a tableau calculus in [20]. $\mathcal{ALBO}^{\mathsf{id}}$ is an extension of the description logic $\mathcal{ALC}$ with individuals, the inverse role (relation) connective, Boolean connectives on roles and the identity role. Although that work predates the work in [19] and the present work, the tableau calculus in [20] of $\mathcal{ALBO}^{\mathsf{id}}$ can be in fact synthesised by altering the semantic specification similar as described for $\mathrm{K}_m(\neg)$ in this section. Using the results of the previous section a hypertableau calculus can be defined for $\mathcal{ALBO}^{\mathsf{id}}$.

$$\frac{@_i\neg p_1 \vee \cdots \vee \neg p_m \vee q_1 \vee \cdots \vee q_n,\ @_i p_1,\ \ldots,\ @_i p_m}{@_i q_1 \mid \cdots \mid @_i q_n} \left(\begin{array}{c} m+n>1 \\ p_1,\ldots,p_m \text{ are atomic}\end{array}\right)$$

$$\frac{@_i p,\ @_i\neg p}{\bot} \qquad \frac{@_i[r]p,\ @_i\neg[r]\neg j}{@_j p} \qquad \frac{@_i\neg[r]p}{@_i\neg[r]\neg f(r,p,i),\ @_{f(r,p,i)}\neg p}$$

$$\frac{@_i\neg[\neg r]\neg j}{@_i[r]\neg j} \qquad \frac{@_i[\neg r]\neg j}{@_i\neg[r]\neg j} \qquad \frac{@_i[\neg r]p}{@_i\neg[r]\neg j \mid @_j p}$$

**Fig. 5.** Hybrid hypertableau calculus $T^{\mathsf{hyp}}_{\mathrm{K}_m(\neg)}$ for $\mathrm{K}_m(\neg)$

## 7 Concluding Remarks

The paper has investigated refinement of inference rules for semantic tableau calculi in the setting of the tableau synthesis framework. We introduced atomic rule refinement as a general principle to reduce branching and simplify the way deductions are carried out with disjunctive formulae. A distinctive feature of the refinement is that it is syntactic and can be automated. As we have shown the approach covers two important cases: refinement of inference rules generated from frame conditions and systematically developing hypertableau-like calculi. In both cases, properties of the language of the logic are exploited. In the first case, because frame conditions are properties on atomic relations, the condition for atomic rule refinement trivially holds. In the second case, formulae of the logic were transformed into a normal form and the hypertableau rule was defined by constraining disjunctive splitting rules with atomic premises.

In the case study of $\mathrm{K}_m(\neg)$ we showed that even if none of the rules of the initially generated calculus are refinable (without loss of completeness) there may be ways to modify the semantic specification for the logic and extend the calculus by additional rules in order to achieve refinability. In this case the addition of derivable rule enabled the refinement of other rules in the calculus.

Adding analytic cut rules [6] to the calculus is another approach to make rule refinement possible. This allows KE tableau calculi to be systematically derived in the framework. Due to space limitation we do not elaborate on this case.

We have considered rule refinement in the tableau synthesis framework. Since its rule language gives full freedom to generate sets of inference rules for any logic, where the semantics can be expressed in a first-order language, the results of the paper apply to all calculi that can be described in the framework. The refinements and essential ideas are however more general and can be applied to other types of deduction calculi, which deserves to be investigated. Further work will include the investigation of other refinements and reduction of the search space, such as ordering restrictions [11].

## References

1. P. Abate and R. Goré. The Tableau Workbench. *Electr. Notes Theoret. Comput. Sci.*, 231:55–67, 2009.

2. S. Babenyshev, V. Rybakov, R. A. Schmidt, and D. Tishkovsky. A tableau method for checking rule admissibility in S4. *Electr. Notes Theoret. Comput. Sci.*, 262:17–32, 2010.
3. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper tableaux. In *Proc. JELIA'96*, vol. 1126 of *LNAI*, pp. 1–17. Springer, 1996.
4. P. Baumgartner and R. A. Schmidt. Blocking and other enhancements for bottom-up model generation methods, 2016. arXiv e-Print 1611.09014 [cs.AI].
5. P. Blackburn and J. Seligman. What are hybrid languages? In *Proc. AiML-1*, pp. 41–62. CSLI Publ., 1998.
6. M. D'Agostino and M. Mondadori. The taming of the cut. Classical refutations with analytic cut. *J. Log. Comput.*, 4(3):285–319, 1994.
7. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic J. IGPL*, 8(3):265–292, 2000.
8. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel, 1983.
9. G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In *Proc. Gödel'86*, pp. 253–263. Plenum, 1987.
10. R. Goré. Tableau methods for modal and temporal logics. In *Handbook of Tableau Methods*, pp. 297–396. Springer, 1999.
11. R. Hähnle and S. Klingenbeck. A-ordered tableaux. *J. Logic Comput.*, 6(6):819–833, 1996.
12. I. L. Humberstone. The modal logic of 'all and only'. *Notre Dame J. Formal Logic*, 28(2):177–188, 1987.
13. U. Hustadt and R. A. Schmidt. Simplification and backjumping in modal tableau. In *Proc. TABLEAUX'98*, vol. 1397 of *LNAI*, pp. 187–201. Springer, 1998.
14. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. A refined tableau calculus with controlled blocking for the description logic $\mathcal{SHOI}$. In *Proc. TABLEAUX'13*, vol. 8123 of *LNCS*, pp. 188–202. Springer, 2013.
15. C. Lutz and U. Sattler. The complexity of reasoning with Boolean modal logics. In *Proc. AiML-3*, pp. 329–348. CSLI Publ., 2002.
16. B. Motik, R. Shearer, and I. Horrocks. Hypertableau reasoning for description logics. *J. Artificial Intelligence Res.*, 36:165–228, 2009.
17. A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In *Handbook of Automated Reasoning*, pp. 335–367. Elsevier, 2001.
18. R. A. Schmidt, J. G. Stell, and D. Rydeheard. Axiomatic and tableau-based reasoning for Kt(H,R). In *Proc. AiML-10*, pp. 478–497. College Publ., 2014.
19. R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Log. Methods Comput. Sci.*, 7(2:6):1–32, 2011.
20. R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *ACM Trans. Comput. Log.*, 15(1), 2014.
21. R. A. Schmidt and U. Waldmann. Modal tableau systems with blocking and congruence closure. In *Proc. TABLEAUX'15*, vol. 9323 of *LNCS*, pp. 38–53. Springer, 2015.
22. R. M. Smullyan. *First Order Logic.* Springer, 1971.
23. J. G. Stell, R. A. Schmidt, and D. Rydeheard. A bi-intuitionistic modal logic: Foundations and automation. *J. Logical Algebr. Methods Program.*, 85(4):500–519, 2016.

## Appendix: A Formal Definition of the Tableau Synthesis Framework

In the tableau synthesis framework a given logic $L$ is defined semantically. The framework is therefore equipped with

(i) a language $\mathcal{L}$ for the specification of syntax of the logic, and
(ii) a language $\mathsf{FO}(L)$ for the specification of the semantics of the logic.

The *syntax specification language* $\mathcal{L}$ is a propositional, possibly multi-sorted language. The set of sorts of $\mathcal{L}$ is denoted by $\mathsf{Sorts}$ and the set of the formulae of each sort $\mathsf{s}$ is denoted by $\mathcal{L}^s$.

The *semantic specification language* $\mathsf{FO}(L)$ is a multi-sorted first-order language with equality $\approx$. $\mathsf{FO}(L)$ is defined over the sorts in $\mathsf{Sorts}$ and contains an additional *domain* sort $\mathsf{D}$ equipped with function and predicate symbols necessary for the specification of the semantics of the logic. Formulae of the logic are expressed as terms of appropriate sorts. $\mathsf{FO}(L)$ contains additional *interpretation* symbols $\nu_{\mathsf{s}}$ for each sort $\mathsf{s}$ of the logic. Depending on the sort $\mathsf{s}$, $\nu_{\mathsf{s}}$ can either be a functional symbol, mapping formulae of sort $\mathsf{s}$ into the domain sort, or it can be a predicate symbol of sort $(\mathsf{s}, \mathsf{D}, \ldots, \mathsf{D})$. If $\nu_{\mathsf{s}}$ is a predicate symbol we can think of it as the 'holds' or 'satisfaction' predicate. For example, the definition of the semantics of the $[r]$ connective in basic modal logic can be specified as

$$\forall x\ (\nu_{\mathsf{f}}([r]p, x) \leftrightarrow \forall y\ (\nu_{\mathsf{r}}(r, x, y) \rightarrow \nu_{\mathsf{f}}(p, y))).$$

We can read this as saying: for any domain element $x$, $x \in ([r]p)^{\mathcal{I}}$ iff for any domain element $y$, $(x, y) \in r^{\mathcal{I}}$ implies $y \in p^{\mathcal{I}}$.

In general, let $E$ denote any formula of the language $\mathcal{L}$ of the logic $L$, and let $\phi^+$ and $\phi^-$ denote any formulae of the meta-language $\mathsf{FO}(\mathcal{L})$. A *normalised semantic specification* $S$ of a logic $L$ is a set of three types of formulae in $\mathsf{FO}(L)$:

$(S^+)$ $\quad \forall \overline{x}\ (\nu_{\mathsf{s}}(E(\overline{p}), \overline{x}) \rightarrow \phi^+(\overline{p}, \overline{x}))$

$(S^-)$ $\quad \forall \overline{x}\ (\phi^-(\overline{p}, \overline{x}) \rightarrow \nu_{\mathsf{s}}(E(\overline{p}), \overline{x}))$

$(S^b)$ $\quad$ Any $\mathsf{FO}(\mathcal{L})$-sentence without occurrences of non-atomic $\mathcal{L}$-formulae.

$\overline{x}$ denotes a sequence of $n$ variables ranging over the domain sort, where $E$ is a formula (with free variables $\overline{p}$) interpreted as an $n$-ary relation. $\overline{p}$ denotes a sequence of $m$ propositional variables of the object language.

An *$\mathcal{L}$-structure* $\mathcal{I}$ is a tuple $\mathcal{I} \overset{\mathsf{def}}{=} (\Delta^{\mathcal{I}}, f^{\mathcal{I}}, \ldots, P^{\mathcal{I}}, \ldots, \{\nu_{\mathsf{s}}^{\mathcal{I}}\}_{\mathsf{s} \in \mathsf{Sorts}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, $f^{\mathcal{I}}$ and $P^{\mathcal{I}}$ are interpretations of function and, respectively, predicate symbols of the domain sort and, for each $\mathsf{s} \in \mathsf{Sorts}$, $\nu_{\mathsf{s}}^{\mathcal{I}}$ is an interpretation of the symbol $\nu_{\mathsf{s}}$ in $\mathcal{I}$. An *$L$-model* is an $\mathcal{L}$-structure $\mathcal{I}$ such that all formulae of the semantic specification of the logic $L$ are true in $\mathcal{I}$ (for all possible interpretations of individual variables).

In the tableau synthesis framework, inference rules are specified in the $\mathsf{FO}(L)$ language. A *tableau calculus* is a set of inference rules which have the general form

$$\frac{X_0}{X_1 \mid \cdots \mid X_n},$$

where both the numerator $X_0$ and all denominators $X_i$ are finite sets of negated or unnegated atomic formulae in the language $\mathsf{FO}(L)$. The formulae in $X_0$ are called *premises*, while the formulae in $X_i$ $(1 \leq i \leq n)$ are called *conclusions*. The $X_i$ are non-empty, but $n$ may be zero, in which case the rule is a *closure rule* (also written $X_0/\bot$).

A *tableau derivation* or *tableau* in a tableau calculus $T$ is a finitely branching, ordered tree whose nodes are sets of formulae in $\mathsf{FO}(L)$. Assuming that $N$ is the input set of $\mathcal{L}$-formulae to be tested for satisfiability the root node of the tableau is the set $\{\nu_{\mathsf{s}}(E, \overline{a}) \mid E \in N \cap \mathcal{L}^{\mathsf{s}}, \mathsf{s} \in \mathsf{Sorts}\}$, where $\overline{a}$ denotes a sequence of fresh constant from the domain sort of an appropriate length.

Successor nodes are constructed in accordance with the inference rules in the calculus $T$. An inference rule $X_0/X_1 \mid \cdots \mid X_n$ is applicable to a selected formula $\phi$ in a node of the tableau, if $\phi$, together with other formulae in the node, are simultaneous instantiations of formulae in $X_0$. Then $n$ successor nodes are created which contain the formulae of the current node and the appropriate instances of $X_i$.

In a tableau derivation, a maximal path from the root node is called a *branch*. For a branch $\mathcal{B}$ of a tableau we write $\phi \in \mathcal{B}$ to indicate that the formula $\phi$ has been derived in $\mathcal{B}$, that is, $\phi$ belongs to a node of the branch $\mathcal{B}$. A branch of a tableau is *closed* if a closure rule has been applied in this branch, otherwise the branch is called *open*. The tableau derivation is *closed* if all its branches are closed and it is *open* otherwise.

The calculus $T$ is *sound* when for a satisfiable set of tableau formulae any fully expanded tableau derivation has an open branch. A tableau derivation is *fully expanded* if all branches are either closed, or open and fully expanded. A tableau calculus is *refutationally complete* if for any unsatisfiable set of tableau formulae there is a closed tableau derivation.

We say that a tableau calculus $T$ is *constructively complete* for a logic $L$ if for any open branch $\mathcal{B}$ in a derivation in $T$ there is an $L$-model $\mathcal{I}$ such that all the formulae in $\mathcal{B}$ are true in $\mathcal{I}$. If $T$ is constructively complete for a logic $L$ then $T$ is refutationally complete for $L$.

Following the tableau synthesis framework, we define a *canonical interpretation* $\mathcal{I}(\mathcal{B})$ in which the domain of $\mathcal{I}(\mathcal{B})$ is constructed from terms of the domain sort $\mathsf{D}$ modulo equalities derived in the branch $\mathcal{B}$. In particular, $\mathcal{I}(\mathcal{B})$ is defined as follows. Let the relation $\sim_{\mathcal{B}}$ is defined by

$$t \sim_{\mathcal{B}} t' \overset{\mathsf{def}}{\iff} t \approx t' \in \mathcal{B},$$

for any ground terms $t$ and $t'$ of the domain sort $\mathsf{D}$ in $\mathcal{B}$. Let $\|t\| \overset{\mathsf{def}}{=} \{t' \mid t \sim_{\mathcal{B}} t'\}$ be the equivalence class of an element $t$. The presence of special equality rules ensures that $\sim_{\mathcal{B}}$ is a congruence relation on all domain ground terms in $\mathcal{B}$ [19]. Then the domain of $\mathcal{I}(\mathcal{B})$ is defined as

$$\Delta^{\mathcal{I}(\mathcal{B})} \overset{\mathsf{def}}{=} \{\|t\| \mid t \text{ occurs in } \mathcal{B}\}.$$

The interpretation of predicate symbols in $\mathcal{I}(\mathcal{B})$ is defined by induction on length of formulae of $\mathcal{L}$ as follows:

– For every $n$-ary constant predicate symbol $P$,

$$P^{\mathcal{I}(\mathcal{B})} \overset{\mathsf{def}}{=} \{(\|t_1\|, \ldots, \|t_n\|) \mid P(t_1, \ldots, t_n) \in \mathcal{B}\}.$$

– For every $\mathsf{s} \in \mathsf{Sorts}$ and arity $n$ for the sort $\mathsf{s}$ in $FO(L)$,
  - if $n = 0$ then $\nu_{\mathsf{s}}^{\mathcal{I}(\mathcal{B})}(t) \overset{\mathsf{def}}{=} \|\nu_{\mathsf{s}}(t)\|$ for every term $t$.
  - if $n > 0$ then the interpretation $\nu_{\mathsf{s}}^{\mathcal{I}(\mathcal{B})}$ is defined as the smallest subset of $\mathcal{L}^{\mathsf{s}} \times (\Delta^{\mathcal{I}(\mathcal{B})})^n$ satisfying both the following, for every variable or constant $p$ of the sort $\mathsf{s}$, every connective $\sigma$, and any formulae $E_1, \ldots, E_m$:

$$(p, \|t_1\|, \ldots, \|t_n\|) \in \nu_{\mathsf{s}}^{\mathcal{I}(\mathcal{B})} \iff \nu_{\mathsf{s}}(p, t_1, \ldots, t_n) \in \mathcal{B},$$
$$(\sigma(\overline{E}), \|t_1\|, \ldots, \|t_n\|) \in \nu_{\mathsf{s}}^{\mathcal{I}(\mathcal{B})} \iff \mathcal{I}(\mathcal{B}) \models \phi^{\sigma}(\overline{E}, \|t_1\|, \ldots, \|t_n\|),$$

where $\phi^{\sigma}$ denotes an $\mathcal{L}$-open formula which defines the connective $\sigma$.

Showing constructive completeness amounts to showing (i) $\mathcal{I}(\mathcal{B})$ is an $L$-interpretation and (ii) it reflects $\mathcal{B}$, if it is open and fully expanded.

By definition, we say an interpretation $\mathcal{I}$ *reflects* a formula $E$ of the sort $\mathsf{s}$ occurring in a branch $\mathcal{B}$ iff for $n$ the arity of the sort $\mathsf{s}$ and for all ground terms $t_1, \ldots, t_n$ we have that

$$(E, \overline{\|t\|}) \in \nu_{\mathsf{s}}^{\mathcal{I}} \quad \text{whenever } \nu_{\mathsf{s}}(E, \bar{t}) \in \mathcal{B}, \text{ and}$$
$$(E, \overline{\|t\|}) \notin \nu_{\mathsf{s}}^{\mathcal{I}} \quad \text{whenever } \neg\nu_{\mathsf{s}}(E, \bar{t}) \in \mathcal{B}.$$

Similarly, $\mathcal{I}$ *reflects* a predicate constant $P$ from $\mathcal{B}$ iff for all ground terms $t_1, \ldots, t_n$ we have that

$$(\overline{\|t\|}) \in P^{\mathcal{I}} \quad \text{whenever } P(\bar{t}) \in \mathcal{B}, \text{ and}$$
$$(\overline{\|t\|}) \notin P^{\mathcal{I}} \quad \text{whenever } \neg P(\bar{t}) \in \mathcal{B}.$$

An interpretation $\mathcal{I}$ *reflects* a branch $\mathcal{B}$ if $\mathcal{I}$ reflects all predicate constants and formulae occurring in $\mathcal{B}$.

## Appendix: Proofs of Theorems and Statements

### Refinement of the Box Rule for Basic Modal Logic

**Theorem 2.** *The tableau calculus* $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ *is sound and constructively complete for the logic* $\mathrm{K}_m$*.*

*Proof.* We prove the general rule refinement condition of Theorem 1 holds for any open branch $\mathcal{B}$ of $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$. The result is then a consequence of Corollary 1. Let $\nu_{\mathsf{f}}([r_i]\phi, t)$ be in arbitrary open branch $\mathcal{B}$ of a derivation in the refined tableau calculus $\mathsf{Ref}((\mathsf{box}), T_{\mathrm{K}_m})$ and suppose $\mathcal{I}(\mathcal{B}) \not\models \neg\nu_{\mathsf{r}}(r_i, t, t')$. Therefore, $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(r_i, t, t')$. By the definition of $\mathcal{I}(\mathcal{B})$, this means that $\nu_{\mathsf{r}}(r_i, t, t') \in \mathcal{B}$. This implies that the refined rule $(\Box)$ has been applied to $\nu_{\mathsf{f}}([r_i]\phi, t)$ and $\nu_{\mathsf{r}}(r_i, t, t')$ in $\mathcal{B}$ and, consequently, $\nu_{\mathsf{f}}(\phi, t')$ is in $\mathcal{B}$.

**Refinement of the Box Rule in the Extended Calculus for the Modal Logic of 'Some', 'All' and 'Only'**

**Theorem 9.** *The tableau calculus* $\mathsf{Ref}((\mathsf{box}), T^{+}_{\mathrm{K}_m(\neg)})$ *using the* $(\Box)$ *rule instead of the* $(\mathsf{box})$ *rule is sound and constructively complete for the logic* $\mathrm{K}_m(\neg)$*.*

*Proof.* We prove the general rule refinement condition of Theorem 1 holds for any open branch $\mathcal{B}$ of $\mathsf{Ref}((\mathsf{box}), T^{+}_{\mathrm{K}_m(\neg)})$. The result is then a consequence of Corollary 1. Let $\nu_{\mathsf{f}}([\alpha]\phi, t)$ be in arbitrary open branch $\mathcal{B}$ of a derivation in the refined tableau calculus $\mathsf{Ref}((\mathsf{box}), T^{+}_{\mathrm{K}_m(\neg)})$ and suppose $\mathcal{I}(\mathcal{B}) \not\models \neg\nu_{\mathsf{r}}(\alpha, t, t')$. Therefore, $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(\alpha, t, t')$.

If $\alpha$ is an atomic relation then, because $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(\alpha, t, t')$, we have $\nu_{\mathsf{r}}(\alpha, t, t') \in \mathcal{B}$. Therefore, the refined rule $(\Box)$ has been applied to $\nu_{\mathsf{f}}([\alpha]\phi, t)$ and $\nu_{\mathsf{r}}(\alpha, t, t')$ in $\mathcal{B}$. As a consequence, $\nu_{\mathsf{f}}(\phi, t')$ is in $\mathcal{B}$.

If $\alpha$ is not atomic then $\alpha = \neg\alpha'$. By induction on the length of $\alpha$, we prove that $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(\alpha, t, t')$ implies $\nu_{\mathsf{r}}(\alpha', t, t') \notin \mathcal{B}$. If $\alpha'$ is atomic the case follows from the definition of $\mathcal{I}(\mathcal{B})$. If $\alpha'$ is not atomic then we have $\alpha' = \neg\alpha''$. Thus, $\nu_{\mathsf{r}}(\alpha', t, t') \in \mathcal{B}$ implies that $\neg\nu_{\mathsf{r}}(\alpha'', t, t') \in \mathcal{B}$. On the other hand, $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(\alpha, t, t')$ if and only if $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(\alpha'', t, t')$. If $\alpha''$ is atomic then $\nu_{\mathsf{r}}(\alpha'', t, t') \in \mathcal{B}$ by the definition of $\mathcal{I}(\mathcal{B})$. Thus, because $\mathcal{B}$ is open, $\neg\nu_{\mathsf{r}}(\alpha'', t, t') \notin \mathcal{B}$. This implies that $\nu_{\mathsf{r}}(\alpha', t, t') \notin \mathcal{B}$. If $\alpha'' = \neg\alpha'''$, then by the induction hypothesis we have $\nu_{\mathsf{r}}(\alpha''', t, t') \notin \mathcal{B}$. Thus, $\neg\nu_{\mathsf{r}}(\neg\alpha''', t, t') \notin \mathcal{B}$ and, consequently, $\nu_{\mathsf{r}}(\alpha', t, t') \notin \mathcal{B}$ by the rules of the calculus. Finally, the rule $([\neg])$ was applied to $\nu_{\mathsf{f}}([\neg\alpha']\phi, t)$ in $\mathcal{B}$ and, hence, $\mathcal{B}$ contains either $\nu_{\mathsf{r}}(\alpha', t, t')$ or $\nu_{\mathsf{f}}(\phi, t')$. As we proved, $\mathcal{I}(\mathcal{B}) \models \nu_{\mathsf{r}}(\alpha, t, t')$ implies that the first case is impossible. This leaves the only alternative: $\nu_{\mathsf{f}}(\phi, t') \in \mathcal{B}$.

Therefore the general rule refinement condition holds for $\mathsf{Ref}((\mathsf{box}), T^{+}_{\mathrm{K}_m(\neg)})$ and, by Corollary 1, $\mathsf{Ref}((\mathsf{box}), T^{+}_{\mathrm{K}_m(\neg)})$ is constructively complete.

**Atomic Rule Refinement**

**Theorem 3.** *Assume that for an open branch* $\mathcal{B}$ *of the refined tableau* $\mathsf{Ref}(\rho, T_L)$ *and for every set* $Y$ *of* $\mathcal{L}$*-formulae from* $\mathcal{B}$ *the following holds.*

**Atomic rule refinement condition:** *If all formulae in* $Y$ *are reflected in* $\mathcal{I}(\mathcal{B})$ *then for every* $E_1, \ldots, E_l \in Y$ *and domain terms* $t_1, \ldots, t_n$*,*

$$X_0(\overline{E}, t_1, \ldots, t_n) \subseteq \mathcal{B} \textit{ implies that}$$
$$X_1(\overline{E}, t_1, \ldots, t_n) = \{\neg\xi_1, \ldots, \neg\xi_k\} \textit{and all } \xi_1, \ldots, \xi_k \textit{ are } \mathcal{L}\text{-atomic.}$$

*Then,* $\mathcal{B}$ *is reflected in* $\mathcal{I}(\mathcal{B})$*.*

*Proof.* We show the general rule refinement condition holds in this case. Assume $X_0(\overline{E}, t_1, \ldots, t_n)$ is contained in $\mathcal{B}$ and $\mathcal{I}(\mathcal{B}) \not\models X_1(\overline{E}, \|t_1\|, \ldots, \|t_n\|)$. Therefore there is some $j = 1, \ldots, k$ such that $\mathcal{I}(\mathcal{B}) \models \xi_j(\|t_1\|, \ldots, \|t_n\|)$. Since $\xi_j(t_1, \ldots, t_n)$ is $\mathcal{L}$-atomic, by the definition of $\mathcal{I}(\mathcal{B})$ we have that $\xi_j(t_1, \ldots, t_n) \in$

$\mathcal{B}$. Consequently, the rule $\rho_j$ has been applied in $\mathcal{B}$ to the set of premises $X_0(\overline{E}, t_1, \ldots, t_n) \cup \{\xi_j(t_1, \ldots, t_n)\}$ and, hence, for some $i = 2, \ldots, m$, the set $X_i(\overline{E}, t_1, \ldots, t_n)$ is contained in the branch $\mathcal{B}$. Finally, by Theorem 1, we have that $\mathcal{B}$ is reflected in $\mathcal{I}(\mathcal{B})$.

### The $n$-Ary Splitting Rule Viewed as a Rule Refinable by Atom Rule Refinement

**Theorem 6.** *$T_L^{\mathsf{sp}+}$ is sound and constructively complete for the logic $L$.*

*Proof.* The rules $(\mathsf{split}^+_{mn})$ are particular cases of the $(\mathsf{split}_k)$ rules for $k = m+n$. That is, for any $m$ and $n$ such that $m + n > 1$ there is a substitution which converts the $(\mathsf{split}_k)$ rule with $k = m + n$ into the $(\mathsf{split}^+_{mn})$ rule.

Furthermore, for any $k > 1$ and a substitution $\sigma$ into the $(\mathsf{split}_k)$ rule there are $m$ and $n$ such that $m + n = k$ and, under the substitution $\sigma$, the premise and the conclusions of the rule $(\mathsf{split}^+_{mn})$ coincide respectively with the premise and the conclusions of the $(\mathsf{split}_k)$ rule (modulo associativity and commutativity of the disjunction of $L$).

Therefore, the rules $(\mathsf{split}_k)$ and the rules $(\mathsf{split}^+_{mn})$ are derivable from each other. The theorem statement follows immediately.

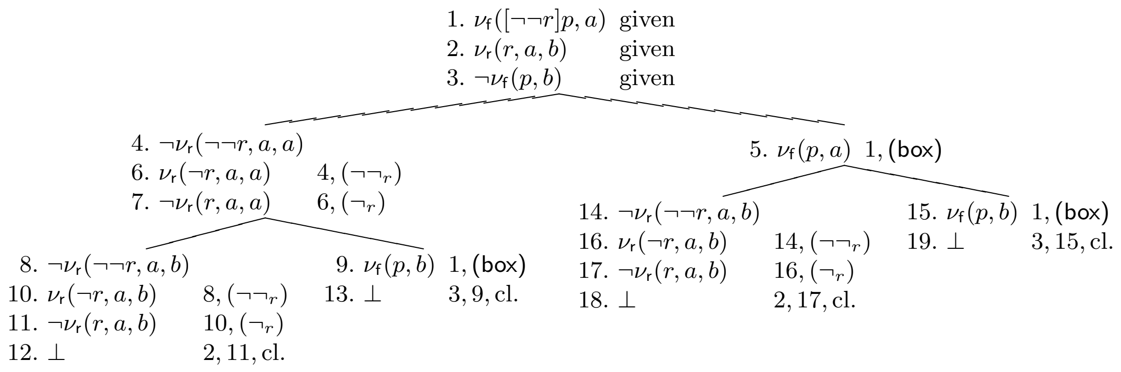
## Appendix: Sample Derivations for $\mathbf{K}_m(\neg)$

Let us consider the example following Theorem 8 to illustrate the benefit that rule refinement can have. The following set of formulae is $\mathrm{K}_m(\neg)$-unsatisfiable.

$$\{\nu_{\mathsf{f}}([\neg\neg r]p, a), \nu_{\mathsf{r}}(r, a, b), \neg\nu_{\mathsf{f}}(p, b)\}$$

It should thus be possible to construct a closed tableau derivation with both the unrefined and refined calculi.

*Derivation in the (unrefined) calculus $T_{\mathrm{K}_m(\neg)}$.*


1. $\nu_{\mathsf{f}}([\neg\neg r]p, a)$ given
2. $\nu_{\mathsf{r}}(r, a, b)$ given
3. $\neg\nu_{\mathsf{f}}(p, b)$ given
4. $\neg\nu_{\mathsf{r}}(\neg\neg r, a, a)$
6. $\nu_{\mathsf{r}}(\neg r, a, a)$ 4, $(\neg\neg_r)$
7. $\neg\nu_{\mathsf{r}}(r, a, a)$ 6, $(\neg_r)$
5. $\nu_{\mathsf{f}}(p, a)$ 1, (box)
8. $\neg\nu_{\mathsf{r}}(\neg\neg r, a, b)$
10. $\nu_{\mathsf{r}}(\neg r, a, b)$ 8, $(\neg\neg_r)$
11. $\neg\nu_{\mathsf{r}}(r, a, b)$ 10, $(\neg_r)$
12. $\bot$ 2, 11, cl.
9. $\nu_{\mathsf{f}}(p, b)$ 1, (box)
13. $\bot$ 3, 9, cl.
14. $\neg\nu_{\mathsf{r}}(\neg\neg r, a, b)$
16. $\nu_{\mathsf{r}}(\neg r, a, b)$ 14, $(\neg\neg_r)$
17. $\neg\nu_{\mathsf{r}}(r, a, b)$ 16, $(\neg_r)$
18. $\bot$ 2, 17, cl.
15. $\nu_{\mathsf{f}}(p, b)$ 1, (box)
19. $\bot$ 3, 15, cl.

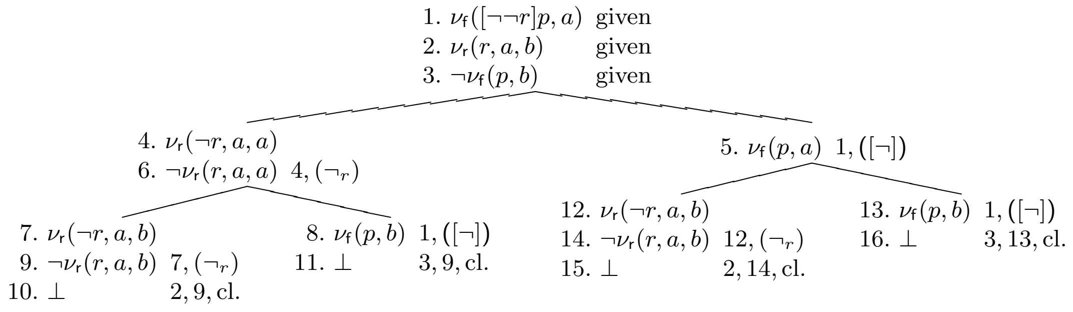
$$\frac{\nu_{\mathsf{f}}(\neg p, x)}{\neg\nu_{\mathsf{f}}(p, x)} \qquad \frac{\neg\nu_{\mathsf{f}}(\neg p, x)}{\nu_{\mathsf{f}}(p, x)} \qquad \frac{\nu_{\mathsf{f}}(p \vee q, x)}{\nu_{\mathsf{f}}(p, x) \mid \nu_{\mathsf{f}}(q, x)} \qquad \frac{\neg\nu_{\mathsf{f}}(p \vee q, x)}{\neg\nu_{\mathsf{f}}(p, x),\ \neg\nu_{\mathsf{f}}(q, x)}$$

$$(\Box)\ \frac{\nu_{\mathsf{f}}([r]p, x),\ \nu_{\mathsf{r}}(r, x, y)}{\nu_{\mathsf{f}}(p, y)} \qquad \frac{\neg\nu_{\mathsf{f}}([r]p, x)}{\nu_{\mathsf{r}}(r, x, f(r, p, x)),\ \neg\nu_{\mathsf{f}}(p, f(r, p, x))}$$

$$\frac{\nu_{\mathsf{f}}(p, x),\ \neg\nu_{\mathsf{f}}(p, x)}{\bot} \qquad \frac{\nu_{\mathsf{r}}(\neg r, x, y)}{\neg\nu_{\mathsf{r}}(r, x, y)} \qquad \frac{\neg\nu_{\mathsf{r}}(\neg r, x, y)}{\nu_{\mathsf{r}}(r, x, y)} \qquad ([\neg])\ \frac{\nu_{\mathsf{f}}([\neg r]p, x)}{\nu_{\mathsf{r}}(r, x, y) \mid \nu_{\mathsf{f}}(p, y)}$$

**Fig. 6.** Refinement of $T_{\mathrm{K}_m(\neg)}$

*Derivation in the refined, non-internalised calculus.* The refined, non-internalised calculus is given in Fig. 6. Because of the new ([¬])-rule the derivation in this calculus saves three steps.



1. $\nu_{\mathsf{f}}([\neg\neg r]p, a)$ given
2. $\nu_{\mathsf{r}}(r, a, b)$ given
3. $\neg\nu_{\mathsf{f}}(p, b)$ given

4. $\nu_{\mathsf{r}}(\neg r, a, a)$
6. $\neg\nu_{\mathsf{r}}(r, a, a)$ 4, $(\neg_r)$

7. $\nu_{\mathsf{r}}(\neg r, a, b)$
9. $\neg\nu_{\mathsf{r}}(r, a, b)$ 7, $(\neg_r)$
10. $\bot$ 2, 9, cl.

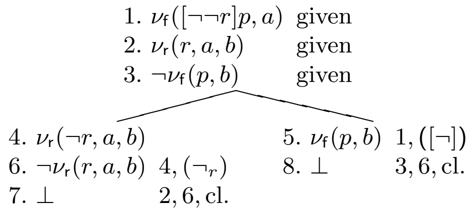
8. $\nu_{\mathsf{f}}(p, b)$ 1, ([¬])
11. $\bot$ 3, 9, cl.

5. $\nu_{\mathsf{f}}(p, a)$ 1, ([¬])

12. $\nu_{\mathsf{r}}(\neg r, a, b)$
14. $\neg\nu_{\mathsf{r}}(r, a, b)$ 12, $(\neg_r)$
15. $\bot$ 2, 14, cl.

13. $\nu_{\mathsf{f}}(p, b)$ 1, ([¬])
16. $\bot$ 3, 13, cl.

*Further optimisations.* Further optimisation is possible by careful instantiation of the free variable $y$ when applying the ([¬]) rule.



1. $\nu_{\mathsf{f}}([\neg\neg r]p, a)$ given
2. $\nu_{\mathsf{r}}(r, a, b)$ given
3. $\neg\nu_{\mathsf{f}}(p, b)$ given

4. $\nu_{\mathsf{r}}(\neg r, a, b)$
6. $\neg\nu_{\mathsf{r}}(r, a, b)$ 4, $(\neg_r)$
7. $\bot$ 2, 6, cl.

5. $\nu_{\mathsf{f}}(p, b)$ 1, ([¬])
8. $\bot$ 3, 6, cl.

The derivation is even shorter if we exploit logical equivalences $\neg\neg p \equiv p$ or $\neg\neg r \equiv r$.

| | | |
|---|---|---|
| 1. | $\nu_{\mathsf{f}}([\neg\neg r]p, a)$ | given |
| 2. | $\nu_{\mathsf{r}}(r, a, b)$ | given |
| 3. | $\neg\nu_{\mathsf{f}}(p, b)$ | given |
| 4. | $\nu_{\mathsf{f}}([r]p, a)$ | $1, \neg\neg r \equiv r$ |
| 5. | $\nu_{\mathsf{f}}(p, b)$ | $4, 2, (\Box)$ |
| 6. | $\bot$ | 3, 5, cl. |