Refinement in the Tableau Synthesis Framework

Dmitry Tishkovsky and Renate A. Schmidt

School of Computer Science, The University of Manchester, UK

Abstract. This paper is concerned with the possibilities of refining and improving calculi generated in the tableau synthesis framework [10]. A general method in the tableau synthesis framework allows to reduce the branching factor of tableau rules and preserves completeness if a general rule refinement condition holds. In this paper we consider two approaches to satisfy this general rule refinement condition.

1 Introduction

The tableau method is one of the most popular deduction approaches in automated reasoning. Tableau methods in various forms are successfully applied for many logics and are especially apt for dynamically developing areas requiring new logical formalisms. However, developing a tableau calculus for a new logic is still a challenging task which usually involves tedious proving of soundness and completeness results.

Based on the collective experience in the area, in recent work [10] we introduced a general framework for synthesising and studying semantic tableau calculi for propositional logics. The framework formalises a three step process for transforming the definition of the semantics of a logic into a sound and complete tableau calculus. The first two steps are to specify the semantics of the logic and to extract tableau inference rules from the semantic specification. Tableau rule extraction is automatic and produces a set of tableau rules operating on formulae in a generic tableau language. When certain natural conditions hold, the generated rules form a sound and constructively complete tableau calculus.

Initially, the generated calculi are in a basic form. Two deficiencies can be identified. One is that some rules of the generated calculus can have branches which are not necessary for guaranteeing completeness. The other problem is that the tableau language of the tableau synthesis framework can be excessively laden with extra-logical notation. Often, but not always, the generated tableau calculus can be encoded more compactly within the language of the logic. Both problems decrease the performance of tableau algorithms based on the calculus. That is why the tableau synthesis framework, addressing these problems, defines a third crucial step: the refinement of the generated tableau calculus.

In [10] we describe two refinements: rule refinement, and a refinement that internalises the language of the calculus within the language of the logic. While the internalisation of the tableau language can be done routinely by extending

* This research is supported by UK EPSRC research grant EP/H043748/1.
and massaging the language of the logic and retains constructive completeness of the calculus, rule refinement, and whether rule refinement preserves constructive completeness is generally more difficult to establish. Currently, rule refinement requires the verification of a general rule refinement condition which is inductive and needs to be checked manually. For the purposes of automating rule refinement it is therefore important to find other less generic conditions sufficient to preserve constructive completeness and, yet, can be automatically verified.

In this paper we describe two new approaches to satisfy the general rule refinement condition and illustrate these approaches on the examples of multi-modal logic $K_m$ satisfying any first-order frame conditions, and the logic $K_m(\neg)$, which is an extension of $K_m$ with the negation operator on accessibility relations. In the first 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. For the second approach we present a special atomic rule refinement condition and prove that it implies the general rule refinement condition. Consequently, this guarantees that the atomic rule refinement preserves constructive completeness of the tableau calculus.

We identify two important cases when the atomic rule refinement condition is satisfied automatically. The first case allows to refine rules which are generated from frame conditions for arbitrary combinations of modal logics. In the second case, using the atomic rule refinement, we show how to transform the generated tableau calculus into a hypertableau-like calculus and prove that the transformation preserves constructive completeness of the calculus.

The paper is structured as follows. The logics $K_m$ and $K_m(\neg)$ which serve as running examples are introduced in Section 2. We recall some notions from the tableau synthesis framework \cite{10} and generate tableau calculi for the considered logics in Section 3. Refinements introduced in the tableau synthesis framework are reviewed in Section 4. Examples of the application of rule refinements are given in Section 5. Atomic rule refinement, a special case of the rule refinement, is introduced and investigated in Section 6. In Section 7 we apply atomic rule refinement to the rules generated from frame conditions for the logic $K_m$. We use the atomic rule refinement to construct a hypertableau-like calculus for the logic $K_m(\neg)$ in Section 8. We conclude with a discussion of the presented results in Section 9. For the benefit of reviewers proofs and technical details are included in the Appendix.

2 The logics $K_m$ and $K_m(\neg)$

As examples to illustrate results of this paper we consider two logics: multi-modal logic $K_m$ with possible frame conditions, and the logic $K_m(\neg)$, the extension of $K_m$ with the operator of negation of relations.

Following the tableau synthesis framework \cite{10} the languages $\mathcal{L}(K_m)$ and $\mathcal{L}(K_m(\neg))$ of these logics have two sorts: a sort for formulae $f$ and a sort for relations $r$. The sorts of relations in these languages are formed over a set of relational constants $\{a_1, \ldots, a_m\}$. In $\mathcal{L}(K_m)$ every relation is a relational constant,
and in \( \mathcal{L}(K_m(\neg)) \) every relation \( \alpha \) is defined by the BNF \( \alpha \equiv a_1 \mid \cdots \mid a_m \mid \neg \alpha \), where \( \neg \) is the operator of negation on relations. The sorts of formulae in both languages are formed over a set of propositional variables \( \{p, q, \ldots\} \) and every formula \( \phi \) of each language is defined by the BNF \( \phi \equiv p \mid \neg \phi \mid \phi \lor \phi \mid [\alpha] \phi \), where \( \alpha \) ranges over all relations of the language.

According to the tableau synthesis framework, the semantic specification language \( \text{FO}(K_m) \) for \( K_m \) is a (multi-sorted) first-order language over sorts of \( \mathcal{L}(K_m) \) and an additional domain sort \( D \). Expressions of \( \mathcal{L}(K_m) \) are naturally embedded into \( \text{FO}(K_m) \) as terms of appropriate sorts. That is, every logical connective of \( \mathcal{L}(K_m) \) is a functional symbol of \( \text{FO}(K_m) \). Every propositional variable of \( \mathcal{L}(K_m) \) is an individual variable of the sort \( f \) in \( \text{FO}(K_m) \). Besides the individual constants \( a_1, \ldots, a_m \) for relations, the language \( \text{FO}(K_m) \) has a countable set of relation variables \( r, r', \ldots \). The additional sort \( D \) has a countable set of individual variables \( x, y, z, \ldots \). Furthermore, the semantic specification language has two predicate symbols \( \nu_f \) and \( \nu_r \) of sorts \( (f, D) \) and \( (r, D, D) \), respectively. The symbols \( \nu_f \) and \( \nu_r \) are required for the purpose of representing satisfiability with respect to domain elements. The meaning of these symbols can be understood from definitions below.

The semantic specification of \( K_m \) consists of the following three formulae.

\[
\forall x \ (\nu_r(\neg p, x) \leftrightarrow \neg \nu_r(p, x)) \\
\forall x \ (\nu_r(p \lor q, x) \leftrightarrow \nu_r(p, x) \lor \nu_r(q, x)) \\
\forall x \ (\nu_r([r]p, x) \leftrightarrow \forall y \ (\nu_r(r, x, y) \rightarrow \nu_r(p, y)))
\]

A model \( I \) of \( K_m \) is a tuple \( I = (\Delta^I, \nu^I_r, \nu^I_f) \) where \( \Delta^I \) is a non-empty set for interpretation of variables of the domain sort, \( \nu^I_r \) and \( \nu^I_f \) are interpretations of the predicates \( \nu_r \) and \( \nu_f \) respectively, and all the formulae of the semantic specification for \( K_m \) are true in \( I \) (for all \( K_m \)-formulae \( p, q \), and \( K_m \)-relation \( r \)). The purpose of symbols \( \nu_f \) and \( \nu_r \) is to define the semantics of the connectives of the logic by using conditions similar to satisfaction conditions in standard definitions. That is, given a \( K_m \)-model \( I \) and elements \( v \) and \( w \) from its domain, for a formula \( \phi \) and a relation \( \alpha \) of \( \mathcal{L}(K_m) \), \( I \models \nu_r(\phi, v) \) can be read as ‘\( \phi \) is true in \( v \)’ and \( I \models \nu_r(\alpha, v, w) \) is understood as ‘\( w \) is an \( \alpha \)-successor of \( v \) in \( I \)’.

For illustrative purposes, in Section 7 we consider two properties of the relations of \( K_m \). Following the terminology in modal logic we refer to such properties as frame conditions. The first condition expresses irreflexivity of relations and is specified by the formula \( \forall x \neg \nu_r(r, x, x) \). The second frame condition is intentionally even more exotic. It states existence of an immediate predecessor for each world in the model and is specified by the formula

\[
\forall x \exists y \forall z \left( \nu_r(r, y, x) \land x \neq y \land \left( \nu_r(r, y, z) \land \nu_r(r, z, x) \right) \rightarrow (z \approx x \lor z \approx y) \right).
\]

The semantic specification language \( \text{FO}(K_m(\neg)) \) of the logic \( K_m(\neg) \) differs from the language \( \text{FO}(K_m) \) only in that the sort of relations involves the negation operator. The semantic specification of \( K_m(\neg) \) extends the semantic specification
of $K_m$ by the following $\text{FO}(K_m(\neg))$-formula.

$$\forall x \forall y \ (\nu_4(\neg r, x, y) \iff \neg \nu_4(r, x, y))$$

Similarly as for the logic $K_m$, a model $I$ for $K_m(\neg)$ is a tuple $I = (\Delta^I, \nu^I_4, \nu^I)$, where $\Delta^I$ is not empty and all the formulae of the semantic specification for $K_m(\neg)$ are true in $I$.

The logic $K_m(\neg)$ is interesting because of the presence of three quantifier operators. These are necessity operator $[\alpha]$, the possibility operator $\neg[\alpha]\neg$ and a third operator, $[\neg\alpha]\neg$, called the sufficiency operator sometimes referred to as the window operator. $\nu_4([\alpha] \phi, v)$ can be read as saying $\phi$ is true in all $\alpha$-successors, $\nu_4([\neg\alpha]\neg \phi, v)$ as $\phi$ is true in some $\alpha$-successor, and $\nu_4([\neg\alpha]\neg \phi, v)$ as $\phi$ is true in only $\alpha$-successors of $v$. Following [5], we call $K_m(\neg)$ the modal logic of ‘some, all and only’. $K_m(\neg)$ is a sublogic of Boolean modal logic [4] and the description logics $\text{ALBO}$ and $\text{ALBO}^d$ [11]. $K_m(\neg)$ has the finite model property [4] but the tree model property fails for the logic (see, e.g., [7]). The results of [7] imply that the satisfiability problem in $K_m(\neg)$ is $\text{ExpTime}$-complete.

3 Tableau synthesis framework

In order to synthesise a tableau calculus for a given logic $L$, the tableau synthesis framework operates with two languages: $\mathcal{L}$, the language of specification of syntax of the logic, and $\text{FO}(L)$, the language of specification of semantics of the logic. Examples of these languages for the logics $K_m$ and $K_m(\neg)$ are defined in the previous section.

The syntax specification language $\mathcal{L}$ is a propositional, possibly multi-sorted language. The set of sorts of $\mathcal{L}$ is denoted by $\text{Sorts}$ and the set of the formulae of each sort $s$ is denoted by $\mathcal{L}^s$. The semantic specification language $\text{FO}(L)$ is a multi-sorted first-order language with equality (denoted by $\approx$). $\text{FO}(L)$ contains an additional domain sort $D$ equipped with function and predicate symbols necessary for the specification of the semantics of the logic. Expressions of $\mathcal{L}$ are embedded into $\text{FO}(L)$ as terms of appropriate sorts and $\text{FO}(L)$ contains additional interpretation symbols $\nu_4$ for each sort $s$ of the logic. Depending on the sort $s$, $\nu_4$ can either be a functional symbol, mapping formulae of sort $s$ into the domain sort, or it can be a predicate symbol of sort $(s, D, \ldots, D)$. If $\nu_4$ is a predicate symbol then we refer to $\nu_4$ as ‘holds’ or ‘satisfaction’ predicate.

A formula $\phi$ of $\text{FO}(L)$ is called $\mathcal{L}$-atomic if it is an atomic formula of $\text{FO}(L)$ and all occurrences of formulae of $\mathcal{L}$ are also atomic in $\phi$. Thus, $\nu_4(E, \neg)$ is $\mathcal{L}$-atomic only if $E$ is an atomic formula of $\mathcal{L}^s$. For example, the formulae $\nu_4(p, x)$ and $\nu_4(r, g(r, x), x)$ are $\mathcal{L}(K_m(\neg))$-atomic, but the formulae $\neg \nu_4(p, x)$, $\nu_4(p \lor q, x)$, and $\nu_4(\neg r, g(r, x), x)$ are not.

A semantic specification of a logic $L$ is a set $S$ of formulae of $\text{FO}(L)$ which satisfies additional properties. In particular, $S$ must define connectives of $L$ and contain only formulae of a special form (see normalised semantic specification in [10] for details). An $\mathcal{L}$-structure $I$ is a tuple $I = (\Delta^I, f^I, \ldots, P^I, \nu_4^I, \{\nu_s^I\}_{s \in \text{Sorts}})$, where $\Delta^I$ is a non-empty set, $f^I$ and $P^I$ are interpretations of function and,
respectively, predicate symbols of the domain sort and, for each \( s \in \text{Sorts} \), \( \nu^s_\mathcal{I} \) is an interpretation of the symbol \( \nu_s \) in \( \mathcal{I} \). An \( L \)-model is an \( L \)-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).

Within the tableau synthesis framework, the language \( \text{FO}(L) \) also plays the role of the tableau language. A tableau calculus is a set of inference rules which have the general form \( X_0/X_1 | \cdots | X_n \), where both the numerator \( X_0 \) and all denominators \( X_i \) are finite sets of negated or unnegated atomic formulae in the language \( \text{FO}(L) \). The formulae in the numerator are called premises, while the formulae in the denominators are called conclusions. The numerator and all the denominators 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 \( \text{FO}(L) \). Assuming that \( N \) is the input set of \( L \)-formulae to be tested for satisfiability the root node of the tableau is the set \( \{ \nu_s(E, \pi) \mid E \in N \cap L^s, s \in \text{Sorts} \} \), where \( \pi \) 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 \( X_0/X_1 | \cdots | X_n \) 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 \( X_i \).

We use the notation \( T(N) \) for a (in the limit) finished tableau built by applying the rules of the calculus \( T \) starting with the set \( N \) (of \( L \)-formulae) as input. That is, we assume that all branches in the tableau are fully expanded and all applicable rules have been applied in \( T(N) \).

In a tableau, a maximal path from the root node is called a branch. For a branch \( B \) of a tableau we write \( \phi \in B \) to indicate that the formula \( \phi \) belongs to a node of the branch \( B \). Considering any branch as a set of formulae it can be shown that the order of rule applications is not essential for a tableau derivation in the sense that all tableau derivations started from given input \( N \) contain same set of branches (as sets of formulae). Thus, without loss of generality, we assume that \( T(N) \) is unique.

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 \( T(N) \) is closed if all its branches are closed and \( T(N) \) is open otherwise. The calculus \( T \) is sound iff for any (possibly infinite) set of formulae \( N \), \( T(N) \) is open whenever \( N \) is satisfiable. \( T \) is complete iff \( T(N) \) is closed for any (possibly infinite) unsatisfiable set \( N \).

We say that a tableau calculus \( T \) is constructively complete for a logic \( L \) if for any open branch \( B \) in a derivation in \( T \) there is an \( L \)-model \( \mathcal{I}(B) \) such that all the formulae in \( B \) are true in \( \mathcal{I}(B) \). Clearly, if \( T \) is constructively complete for a logic \( L \) then \( T \) is complete for \( L \). Following the tableau synthesis framework, we assume that the domain of the model \( \mathcal{I}(B) \) is constructed from terms of the domain sort \( D \) modulo equalities derived in the branch \( B \). In particular, the domain
Tableau rules of $T_{K_m}$:

\[\nu(-p, x) \quad \nu(p, x)\]
\[\nu(p \lor q, x) \quad \nu(p, x) \quad \nu(q, x)\]
\[\nu([r]p, x) \quad \nu([r]p, x) \quad \nu(q, x)\]
\[\nu([r]p, x) \quad \nu([r]p, x) \quad \nu(q, x)\]

Additional rules of $T_{K_m}(-)$:

\[\nu(-r, x, y) \quad \nu([r]p, x) \quad \nu([r]p, x)\]
\[\nu(-r, x, y) \quad \nu([r]p, x) \quad \nu([r]p, x)\]

Figure 1. Generated tableau calculi for $K_m$ and $K_m(-)$

of the model is $\Delta^I(B) \equiv \{ ||t|| \mid t \text{ is a term of the domain sort and } t \text{ occurs in } B \}$, where $||t|| \equiv \{ t' \mid t \approx t' \in B \}$. We say that $B$ is reflected in $I(B)$ iff all the formulae in $B$ are true in $I(B)$ under the valuation $t \mapsto ||t||$ for each domain term $t$ (see [10] for details).

Given a semantic specification for a logic $L$, which satisfies additional conditions (see well-defined semantical specification in [10]), the tableau synthesis framework generates a tableau calculus sound and constructively complete for $L$.

The tableau calculi $T_{K_m}$ and $T_{K_m}(-)$ respectively generated in the tableau synthesis framework from the semantic specifications for $K_m$ and $K_m(-)$ are given in Figure 1. The calculus $T_{K_m}(-)$ extends $T_{K_m}$ by two additional rules for relational negation. Because the semantic specifications for $K_m$ and $K_m(-)$ are well-defined in the sense of [10], from Theorems 5.1 and 5.6 in [10] we immediately obtain the following result.

**Theorem 1 (Soundness and constructive completeness).** The calculi $T_{K_m}$ and $T_{K_m}(-)$ are sound and constructively complete for the logics $K_m$ and $K_m(-)$.

4 Existing refinements

It this section we briefly recall two refinement techniques in the tableau synthesis framework [10]. The first refinement addresses the problem that, in general, the degree of branching of the generated rules is not optimal and higher than is necessary. The refinement reduces the number of branches of a rule by constraining the rule with additional premises and having fewer conclusions. We refer to this refinement as rule refinement. Suppose $\rho$ is a tableau rule in a sound and constructively complete tableau calculus $T_L$ for a logic $L$. Suppose $\rho$ has the form $\rho \equiv X_0/X_1 \mid \cdots \mid X_m$. Let $X_i = \{ \psi_1, \ldots, \psi_k \}$ be one of the denominators of $\rho$
for some \( i \in \{1, \ldots, m \} \). For simplicity and without loss of generality, we assume that \( i = 1 \).

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

\[
\rho_j \overset{\mathtt{def}}{=} \frac{X_0 \cup \{\sim \psi_j \}}{X_2 | \cdots | X_m}.
\]

Each \( \rho_j \) is obtained from the rule \( \rho \) by removing the first denominator \( X_1 \) and adding the negation of one of the formulae in \( X_1 \) as a premise.

We denote by \( \text{ref}(\rho, T_L) \) the refined tableau calculus obtained from \( T_L \) by replacing the rule \( \rho \) with rules \( \rho_1, \ldots, \rho_k \). We say that \( \text{ref}(\rho, T_L) \) is a refinement of \( T_L \). One can show that each \( \rho_j \) is derivable \([5]\) in \( T_L \) and this implies that the calculus \( \text{ref}(\rho, T_L) \) is sound. In general, \( \text{ref}(\rho, T_L) \) is neither constructively complete nor complete. Nevertheless, the following theorem is proved in \([10]\).

**Theorem 2.** Let \( T_L \) be a tableau calculus which is sound and constructively complete for the logic \( L \). Let \( \rho \) be the rule \( X_0/X_1 | \cdots | X_m \) in \( T_L \) and suppose \( \text{ref}(\rho, T_L) \) is a refinement of \( T_L \). Further, suppose \( B \) is an open branch in a \( \text{ref}(\rho, T_L) \)-tableau derivation and for every set \( Y \) of \( L \)-formulae from \( B \) the following holds.

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

\[
\text{if } X_0(E, t_1, \ldots, t_n) \subseteq B \text{ and } I(B) \not\models X_i(E, \|t_1\|, \ldots, \|t_n\|) \text{ then } X_i(E, t_1, \ldots, t_n) \subseteq B, \text{ for some } i = 2, \ldots, m.
\]

Then, \( B \) is reflected in \( I(B) \).

Assuming that \( p_1, \ldots, p_l \) and \( x_1, \ldots, x_n \) are respectively all the \( L \)-variables and all the domain variables occurring in the rule \( \rho \), \( X_i(E, t_1, \ldots, t_n) \) denotes the set of all instances of the \( \text{FO}(L) \)-formulae from \( X_i \) under uniform substitution of \( E_1, \ldots, E_l \) and \( t_1, \ldots, t_n \) into \( p_1, \ldots, p_l \) and \( x_1, \ldots, x_n \) respectively.

The general rule refinement condition states that if there is not enough information in the branch \( B \) to derive the formulae of \( X_i(E, t_1, \ldots, t_n) \) in the model constructed from \( B \) then at least one of the other denominators of the rule is explicitly contained in the branch \( B \).

The general rule refinement condition corresponds to the condition (\( \dagger \)) in \([10]\) which is stronger than condition (\( \ddagger \)) in \([10, \text{Theorem 6.1}]\) but is enough for the purposes of this paper. A consequence of the theorem is the following.

**Corollary 1.** If the condition of Theorem 2 holds for every open branch \( B \) of any fully expanded \( \text{ref}(\rho, T_L) \)-tableau then the refined calculus \( \text{ref}(\rho, T_L) \) is constructively complete for the logic \( L \).

Generalising this refinement to turning more than one denominator into premises is not difficult.

The second refinement in the framework allows to internalise \( \nu_s \) and the domain sort symbols inside the language of the logic if there are appropriate constructs in \( L \) with the same semantics. In this case each atomic formula \( \nu_s(E, \bar{\pi}) \)
in the tableau calculus for $L$ is replaced by a suitable formula of the logic and, then, all syntactically redundant rules are removed from the transformed calculus. This refinement simplifies the tableau language and, in many cases, reduces the number of the rules in the tableau calculus. We refer to this refinement as the \textit{internalisation refinement}.

The intended way to apply these two refinement is in the order of their description here. Usually, this order is also the easiest way for applying the refinements and produces the best possible improvement of the generated calculus.

5 Using existing refinements

The box decomposition rule

\[
\frac{\nu_\eta[r]p, x}{\neg_\eta(r, x, y) \ | \ \nu_\eta(p, y)}
\]

in the calculus $T_{K_m}$ can be refined to the usual box rule

\[
\frac{\nu_\eta[r]p, x}{\nu_\Box(r, x, y) \ | \ \nu_\eta(p, y)}
\]

preserving constructive completeness of the calculus. It can be proved directly that the generic refinement condition is true for this rule in any branch of ref$(\box)$-derivation, and, thus, by Corollary 1 the calculus ref$(\box)T_{K_m}$ is still constructively complete. (We notice that constructive completeness of ref$(\box)T_{K_m}$ also follows from Corollary 2 in Section 6 because any instantiation of $\nu_\eta(r, x, y)$ in the language of $K_m$ is an $L(K_m)$-atomic formula.)

\textbf{Theorem 3.} The tableau calculus ref$(\box)T_{K_m}$ is sound and constructively complete for the logic $K_m$.

However, none of the rules of the tableau calculus for $K_m(\neg)$ from Figure 1 are refinable. In particular, the $\box$ rule cannot be refined to the $\Box$ rule without losing constructive completeness. Take for instance the set of formulae \{\$\nu_\eta[\neg r]p, a)$, $\nu_\eta(r, a, b)$, $\neg_\eta(p, b)$\}. It is $K_m(\neg)$-unsatisfiable but none of the rules of the refined calculus ref$(\box)T_{K_m(\neg)}$ are applicable to the set.

Nevertheless, using a small transformation trick with the semantic specificaton we can obtain a tableau calculus where this refinement is possible. We observe that the following statement is derivable from the semantic specification of $K_m(\neg)$.

\[\forall x \ (\nu_\eta[\neg r]p, x) \rightarrow \forall y \ (\neg_\eta(r, x, y) \rightarrow \nu_\eta(p, y))\]

This means that it can be added to the semantic specification of $K_m(\neg)$ without changing the class of models of the logic. We denote the tableau calculus generated from the semantic specification extended with this statement by $T_{K_m(\neg)}^+$. $T_{K_m(\neg)}^+$ consists of the rules listed in Figure 1 and the following additional rule.

\[
\frac{\nu_\eta[\neg r]p, x}{\nu_\eta(r, x, y) \ | \ \nu_\eta(p, y)}
\]
Refined tableau rules for $\text{K}_m$:

\[
\begin{array}{c}
\text{rule 1:} \\
\text{rule 2:} \\
\text{rule 3:} \\
\text{rule 4:}
\end{array}
\]

Additional refined rules for $\text{K}_m(\neg)$:

\[
\begin{array}{c}
\text{rule 1:} \\
\text{rule 2:} \\
\text{rule 3:} \\
\text{rule 4:}
\end{array}
\]

Figure 2. Refined tableau calculi $T^+_m(\neg)$ and $T^+_m(\neg)$.

It is possible to check that the well-definedness conditions from [10] are satisfied for the extended semantic specification of $\text{K}_m(\neg)$. Therefore, by results in the tableau synthesis framework, $T^+_m(\neg)$ is sound and constructively complete for $\text{K}_m(\neg)$.

The general rule refinement condition is now satisfied for the calculus obtained from $T^+_m(\neg)$ by refinement of the (box) rule and, thus, the following theorem is a consequence of Corollary 1.

**Theorem 4.** The tableau calculus $\text{ref}(\text{box})T^+_m(\neg)$ is sound and constructively complete for the logic $\text{K}_m(\neg)$.

The internalisation refinement is possible for the calculi in accordance with [10] if nominals and the ‘satisfaction’ operator @ of hybrid logic [2] are introduced to the tableau languages of $\text{K}_m$ and $\text{K}_m(\neg)$. More precisely, every formula $\nu_r(\phi,a)$ is replaced by the formula $\langle a \rangle \phi$, and every $\nu_r(\alpha,a,b)$ is replaced by the formula $\langle a \rangle \neg [\alpha] \neg b$. In this case the results of the refinement are labelled tableau calculi, which are also sound and constructively complete for $\text{K}_m$ and $\text{K}_m(\neg)$. Their rules are listed in Figure 2. We denote these calculi by $T^+_m$ and $T^+_m(\neg)$ respectively.

## 6 Atomic rule refinement

In this section we introduce the technique of atomic rule refinement. Under this refinement, all conclusions of a rule which are moved upward are negated $\mathcal{L}$-atomic formulae of the language $\text{FO}(L)$. More precisely, in the notation and with the assumptions of Theorem 2, the following result holds.

**Theorem 5.** Assume that for an open branch $B$ of the refined tableau $\text{ref}(p,T^+_L)$ and for every set $Y$ of $\mathcal{L}$-formulae from $B$ the following holds.
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 \( t_1, \ldots, t_n \),
\[
X_0(E, t_1, \ldots, t_n) \subseteq \mathcal{B} \implies \text{that}
X_1(E, t_1, \ldots, t_n) = \{ \neg \xi_1, \ldots, \neg \xi_k \} \text{ and all } \xi_1, \ldots, \xi_k \text{ are } \mathcal{L}-\text{atomic.}
\]
Then, \( \mathcal{B} \) is reflected in \( \mathcal{I}(\mathcal{B}) \).

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 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 5, in order to preserve constructive completeness of the calculus under atomic rule refinement 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 5 holds for every open branch \( \mathcal{B} \) of any fully expanded \( \text{ref}(\rho, \mathcal{T}_L) \)-tableau then the refined calculus \( \text{ref}(\rho, \mathcal{T}_L) \) is constructively complete for the logic \( L \).

7 Atomic rule refinement for frame conditions

In this and the following section we consider two important cases in which Corollary 2 holds.

The first case is important because it allows to automatically refine tableau rules generated from frame conditions of modal logics.

Consider the axiom of irreflexivity of \( K_m \)-relations introduced in Section 2:
\[
\forall x \neg \nu_r(r, x, x).
\]
The rule generated from this property is \( \nu_r(r, x, x) \). We claim that this rule can be refined to the following closure rule
\[
(\text{irr}) \quad \nu_r(r, x, x) \downarrow.
\]
Because the language of \( K_m \) contains only atomic relations \( a_1, \ldots, a_m \) and no relational operators, any instantiation of \( r \) (and variable \( x \)) in \( \nu_r(r, x, x) \) produces only \( \mathcal{L}(K_m) \)-atomic formulae of the form \( \nu_l(a_i, t, t) \) (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 \( \text{ref}(\text{box})T_{K_m} \) extended with the \( \text{[irr]} \) rule. Thus, by Corollary 2 the calculus \( \text{ref}(\text{box})T_{K_m} \) extended with the \( \text{[irr]} \) rule is sound and constructively complete for the logic \( K_m \) with irreflexive relations. Applying internalisation refinement we obtain the following theorem for the labelled tableau calculus.

Theorem 6. \( T_{K_m} \) extended with the rule \( @, \neg[r] \neg i / \downarrow \) is sound and constructively complete for \( K_m \) with irreflexive relations.
For another example consider the frame condition stating existence of an immediate predecessor of any element of a model in Section 2. We reduce it to a form which is acceptable in the tableau synthesis framework. Let \( g \) be a new Skolem function which depends on the two arguments of the sort of relations and the domain sort. We remove the existential quantifier from the frame condition and decompose the result into three formulae:

\[
\forall x \nu_r(r, g(r, x), x), \quad \forall x \left( x \not\approx g(r, x) \right), \\
\forall x \forall z \left( \left( \nu_r(r, g(r, x), z) \land \nu_r(r, z, x) \right) \to \left( g(r, x) \approx z \lor z \approx x \right) \right).
\]

From these formulae three rules are generated:

\[
\nu_r(r, g(r, x), x), \quad x \not\approx g(r, x), \\
\neg \nu_r(r, g(r, x), z) \mid \neg \nu_r(r, z, x) \mid g(r, x) \approx z \mid z \approx x.
\]

The 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 for the rule generated from the irreflexivity axiom we find that any instantiation of \( x \approx g(r, x) \), \( \nu_r(r, g(r, x), z) \), and \( \nu_r(r, z, x) \) within the language \( \text{FO}(K_m) \) cannot produce a formula which is not \( \mathcal{L}(K_m) \)-atomic. Hence, the atomic rule refinement condition holds for these rules in any branch of any tableau derivation constructed within the language \( \text{FO}(K_m) \). Therefore, refining the second rule once and the third rule twice the rules

\[
x \approx g(r, x) \quad \perp \\
\nu_r(r, g(r, x), z) \quad \nu_r(r, z, x) \\
g(r, x) \approx z \mid z \approx x
\]

are obtained. By Corollary 2, constructive completeness of any tableau calculus in the language \( \text{FO}(K_m) \) is preserved under these refinements. Internalising \( \text{FO}(K_m) \) in the hybrid logic extension of \( K_m \) and introducing a new function \( g \) which depends on two arguments of the relational sort and, respectively, the sort of nominals (see [10] for details) we, in particular, obtain the following theorem.

**Theorem 7.** \( T_{K_m}^K \) extended with the rules

\[
\frac{\Box \lnot [r] \neg i}{\Box [g(i, r)]}, \quad \frac{\Box g(r, i)}{\perp} \quad \text{and} \quad \frac{\Box \lnot [r] \neg j}{\Box [g(i, r)] j}, \quad \frac{\Box g(r, i)}{\Box j i}
\]

is sound and constructively complete for \( K_m \) over the class of models satisfying the frame condition of existence of an immediate predecessor.

### 8 Hypertableau

Let the given logic \( L \) have disjunction-like connectives \( \lor \) and negation-like connectives \( \neg \) for some sort \( s \) of the logic. Assume \( T_L \) is a tableau calculus sound
and constructively complete for $L$ and contains the rules
\[
\frac{\nu_s(p, \overline{F})}{\nu_s(p, \overline{p}, \overline{F})} \quad \text{and} \quad \frac{\nu_s(p, \overline{q}, \overline{F})}{\nu_s(p, \overline{p}, \overline{q}, \overline{F})}.
\]
which are the usual rules for disjunction and negation. We transform the calculus $T_L$ into a new calculus $T_L^{\text{np}}$ 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 derivable from the semantic specification of $L$:
\[
\nu_s(p \lor q, \overline{F}) \leftrightarrow \nu_s(q \lor p, \overline{F}),
\nu_s((p \lor q) \lor r, \overline{F}) \leftrightarrow \nu_s(p \lor (q \lor r), \overline{F}).
\]
This assumption is not essential for the transformation but allows to simplify disjunctions and avoid a combinatorial blow-up.

In the first step of the transformation, the usual disjunction rule $\nu_s(p \lor q, \overline{F})/\nu_s(p, \overline{F}) \vdash \nu_s(q, \overline{F})$ of $T_L$ is replaced by the set of the rules (for $k > 1$):
\[
\frac{(\text{split}_k)}{\nu_s(p_1 \lor \cdots \lor p_k, \overline{F})} \quad \frac{\nu_s(p_1, \overline{F}) \lor \cdots \lor \nu_s(p_k, \overline{F})}{}.
\]
We denote by $T_L^+$ a tableau calculus obtained from $T_L$ by replacing the usual disjunction rule by the rules $\{\text{split}_k\}$. The $\{\text{split}_k\}$ rules and the usual disjunction rule are derivable from each other. Therefore, the transformed $T_L^+$ calculus is sound and constructively complete.

For the second step consider the following rules (for $m + n > 1$).
\[
\frac{(\text{split}^+_{mn})}{\nu_s(p_1 \lor \cdots \lor p_m \lor q_1 \lor \cdots \lor q_n, \overline{F})} \quad \frac{\nu_s(p_1, \overline{F}) \lor \cdots \lor \nu_s(p_m, \overline{F}) \lor \nu_s(q_1, \overline{F}) \lor \cdots \lor \nu_s(q_n, \overline{F})}{}.
\]
(only atomic substitutions are allowed into $p_1, \ldots, p_m$)

That is, the rules are applicable only to formulae of the shape $\nu_s(\neg E_1 \lor \cdots \lor \neg E_m \lor q_1 \lor \cdots \lor q_n, \overline{F})$, where all $E_1, \ldots, E_m$ are atomic formulae of the logic $L$. We also implicitly assume that all $E_1, \ldots, E_m$ are not negated atomic formulae of $L$. Let $T_L^{++}$ be a tableau calculus obtained from $T_L^+$ by replacing the rules $\{\text{split}_k\}$ by the rules $\{\text{split}_m, \text{split}_n\}$. The rules $\{\text{split}_k\}$ and the rules $\{\text{split}^+_{mn}\}$ are derivable from each other and, thus, the following theorem holds.

**Theorem 8.** $T_L^{++}$ is sound and constructively complete for the logic $L$.

In the final step, we refine the rules obtained in the previous step to the set of rules (for $m + n > 1$) which are hypertableau-like rules.
\[
\frac{(\text{hyp}_{mn})}{\nu_s(\neg p_1 \lor \cdots \lor \neg p_m \lor q_1 \lor \cdots \lor q_n, \overline{F})} \quad \frac{\nu_s(p_1, \overline{F}) \lor \cdots \lor \nu_s(p_m, \overline{F}) \lor \nu_s(q_1, \overline{F}) \lor \cdots \lor \nu_s(q_n, \overline{F})}{}.
\]
(only atomic substitutions are allowed into $p_1, \ldots, p_m$)

Similarly to the rules in the previous step, an application of the rule $\{\text{hyp}_{mn}\}$ is allowed only to formulae of the shape $\nu_s(\neg E_1 \lor \cdots \lor \neg E_m \lor q_1 \lor \cdots \lor q_n)$, where
all $E_1, \ldots, E_m$ are atomic formulae and $F_1, \ldots, F_n$ are all not negated atomic formulae of the logic $L$. Notice that in the case of $n = 0$ the rules $\text{[hyp}_{mn}\text{]}$ are atomic closure rules.

Let $T_{L}^{hyp}$ be the calculus obtained from $T_{L}^{c}$ by adding the $\text{[hyp}_{mn}\text{]}$ rules. By Corollary 2 and Theorem 8 we obtain constructive completeness of $T_{L}^{hyp}$.

**Theorem 9.** $T_{L}^{hyp}$ 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 $T_{L}^{hyp}$ can be done more efficiently if the given logic $L$ has additional properties. We have already assumed associativity and commutativity of disjunction. Suppose now that satisfiability of formulae in a large subset of the language of $L$ is reducible to satisfiability of a set of clauses of formulae:

$$\nu_{s}(E, \overline{\tau}) \leftrightarrow \bigwedge_{i=1}^{I} \bigvee_{j=1}^{J_{i}} \nu_{s_{ij}}(E_{ij}, \overline{\tau}).$$

Thus, every formula $E$ has an equi-satisfiable clausal representation as set of clauses $C_{1}, \ldots, C_{I}$, where $C_{i} = E_{i1} \lor \cdots \lor E_{iJ_{i}}$ 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 $A$ be the reduction algorithm which transform any formula $E$ into such equi-satisfiable clausal normal form.

The case becomes interesting when for many formulae of the logic their clausal normal form has clauses with negated atomic formulae. This assumption implies that the $\text{[hyp}_{mn}\text{]}$ rules with $m > 0$ are applied on average often in derivations in $T_{L}^{hyp}$. Since the $\text{[hyp}_{mn}\text{]}$ rules with $m > 0$ create less branching points in derivations than the $\text{[hyp}_{mn}\text{]}$ rule with $m = 0$, derivations in $T_{L}^{hyp}$ contain less branches and, therefore, is more efficient.

We notice that the conclusions of the $\text{[hyp}_{mn}\text{]}$ rules are allowed to contain non-atomic $L$-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 clasification algorithm $A$ to every new conclusion of any rule which is different from the $\text{[hyp}_{mn}\text{]}$ rules. The first alternative uses the power of the original calculus $T_{L}$ and the second one uses the power of the clasification algorithm $A$. For efficiency of algorithms based on the tableau calculus $T_{L}^{hyp}$, these two alternatives have to be well-balanced depending on the complexity of the algorithm $A$ and how efficiently it is implemented.

The logic $K_{m}(\neg)$ supports a Boolean disjunction and negation on the sort of formulae. Therefore it is possible to devise a hypertableau calculus for $K_{m}(\neg)$. There is an efficient classification algorithm for Boolean part which runs in.

13
polynomial time on the length of the input \[9\]. Thus, we 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. The hypertableau calculus for $K_m(\neg)$ in a form of a labelled calculus is presented in Figure 3. By Theorem 9, this calculus is sound and constructively complete for $K_m(\neg)$.

9 Concluding remarks

The paper is an investigation of refinement techniques of tableau calculi developed within the tableau synthesis framework. Rule refinement reduces the number of branches of rules and, therefore, tableau algorithms based on refined tableau calculi run more efficiently comparing with the algorithms based on the original calculi. Furthermore, the refinement provides an incremental method of improving and optimising sound and constructive complete tableau calculi. The most generic condition which ensures that constructive completeness is preserved under rule refinement is second-order, and, thus, is difficult to check. In contrast, the condition for atomic rule refinement presented in this paper is purely syntactic and, thus, can be easily verified. It turns out that this kind of refinement can be applied in many cases of tableau calculi developed for various logics. The refinement works for rules reflecting frame conditions of modal logics and declarations of role properties in description logics, and allows to develop hypertableau-like calculi for logics with disjunction and negation.

As case studies we considered the logic $K_m$ with frame conditions and the logic $K_m(\neg)$ of ‘some’, ‘all’ and ‘only’. We showed that the tableau calculus for $K_m(\neg)$ generated by the tableau synthesis framework can be made refifiable by using a trick of extending semantic specification of $K_m(\neg)$ by a new statement derivable in the original specification. In this case, we proved that the general second-order rule refinement condition becomes true. On the basis of the refined tableau calculus we developed a hypertableau calculus for $K_m(\neg)$ applying the atomic rule refinement to the calculus.

The tableau calculus of $\text{ALBO}^{id}$ from \[11\] can be used for deciding $K_m(\neg)$. $\text{ALBO}^{id}$ is an extension of the description logic $\text{ALC}$ with individuals, the inverse role operator, Boolean operators on roles and the identity role. Although not
developed with the techniques described in this paper, we remark the tableau calculus of $\text{ALBO}^d$ can be obtained by altering the semantic specification similar as described for $K_m(\neg)$ in this paper.

We observe that the original rule is derivable from the rules obtained from it by the rule refinement method if the calculus contains the analytic cut rule $[3]$. Thus, this refinement preserves constructive completeness in the presence of the analytic cut rule. Therefore, KE tableau calculi can be systematically defined using refinement from calculi generated by the framework.

For simplicity of presentation we omitted explicit equality reasoning from the presented tableau calculi. However, it must be noted that if the calculus is able to derive an equality formula then some form of the equality reasoning must be performed within tableau derivations in order to keep completeness of the calculus. This can be done either by a special group of tableau equality rules $[10]$ or by means of ordered rewriting as it is implemented in MetTeL2 prover generator $[12]$.

For future work, it is of interest to implement the considered types of tableau calculi for different logics and compare their performance. With the MetTeL2 prover generator $[12]$, this task should be feasible but requires additional implementation efforts. Connections of the proposed hypertableau method and the hypertableau calculi of $[18]$ is also a promising direction of research.

The tableau refinement methods presented in this paper as an extension of $[10]$ gives a novel view on existing tableau calculi and makes development of new tableau calculi easy and accessible.

References

1. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper tableaux. vol. 1126 of LNLI, pp. 1–17. Springer, 1996.
2. P. Blackburn and J. Seligman. What are hybrid languages? pp. 41–62. CSLI Publ., 1998.
3. M. D’Agostino and M. Mondadori. The taming of the cut. Classical refutations with analytic cut. J. Log. Comput., 4(3):285–319, 1994.
4. G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In D. Skordev, ed., Proc. Gödel’86, pp. 253–263. Plenum, 1987.
5. R. Goré. Tableau methods for modal and temporal logics. In M. D’Agostino, D. M. Gabbay, R. Hähnle, and J. Posegga, eds, Handbook of Tableau Methods, pp. 297–396. Springer, 1999.
6. I. L. Humberstone. The modal logic of ‘all and only’. Notre Dame J. Formal Logic, 28(2):177–188, 1987.
7. C. Lutz and U. Sattler. The complexity of reasoning with Boolean modal logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, eds, Proc. AiML’02, pp. 329–348. CSLI Publ., 2002.
8. B. Motik, R. Shearer, and I. Horrocks. Hypertableau reasoning for description logics. J. Artif. Intell. Res. (JAIR), 36:165–228, 2009.
9. A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In A. Robinson and A. Voronkov, eds, Handbook of Automated Reasoning, pp. 335–367. Elsevier, 2001.
10. R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Log. Methods Comput. Sci.*, 7(2):1–32, 2011.
11. R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *CoRR*, abs/1208.1476, 2012.
12. D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeL2. In L. F. del Cerro, A. Herzig, and J. Mengin, eds, *Proc. JELIA ‘12*, vol. 7519 of *LNAI*, pp. 492–495. Springer, 2012.
Proofs of theorems and statements

For better understanding of the proofs in this section we give a detailed formal definition of the notion of constructive completeness of a tableau calculus [10].

Let \( \mathcal{B} \) denote an arbitrary open branch in a \( T \)-tableau derivation. We define an \( \mathcal{L} \)-structure \( \mathcal{I}(\mathcal{B}) \) as follows. Let the relation \( \sim_{\mathcal{B}} \) be defined by

\[
t \sim_{\mathcal{B}} t' \iff t \equiv t' \in \mathcal{B},
\]

for any ground terms \( t \) and \( t' \) of the domain sort \( \mathcal{D} \) in \( \mathcal{B} \). Let \( \|t\| \equiv \{t' \mid t \sim_{\mathcal{B}} t'\} \) 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 \( \mathcal{B} \) [10].

Then the domain of \( \mathcal{I}(\mathcal{B}) \) is defined as \( \Delta^2(\mathcal{B}) \equiv \{\|t\| \mid t \text{ occurs in } \mathcal{B}\} \). Interpretations of predicate symbols in \( \mathcal{I}(\mathcal{B}) \) are defined by induction on length of formulae of \( \mathcal{L} \) as follows:

- For every \( n \)-ary constant predicate symbol \( P \),
  \[
P^{\Delta^2(\mathcal{B})} \equiv \{\{(\|t_1\|, \ldots, \|t_n\|) \mid P(t_1, \ldots, t_n) \in \mathcal{B}\}.
\]

- For every \( s \in \text{Sorts} \) and \( n = \text{ar}(s) \)
  - if \( n = 0 \) then \( \nu_s^{\mathcal{I}(\mathcal{B})}(t) \equiv \|\nu_s(t)\| \) for every term \( t \).
  - if \( n > 0 \) then the interpretation \( \nu_s^{\mathcal{I}(\mathcal{B})} \) is defined as the smallest subset of \( \mathcal{L}^s \times (\Delta^2(\mathcal{B}))^n \) satisfying both the following, for every variable or constant \( p \) of the sort \( s \), every connective \( \sigma \), and any formulae \( E_1, \ldots, E_m \):
    \[
    (p, \|t_1\|, \ldots, \|t_n\|) \in \nu_s^{\mathcal{I}(\mathcal{B})} \iff \nu_s(p, t_1, \ldots, t_n) \in \mathcal{B},
    \]
    \[
    (\sigma(E), \|t_1\|, \ldots, \|t_n\|) \in \nu_s^{\mathcal{I}(\mathcal{B})} \iff \mathcal{I}(\mathcal{B}) \models \phi^\sigma(\|E\|, \|t_1\|, \ldots, \|t_n\|).
    \]

Recall that \( \phi^\sigma \) denotes an \( \mathcal{L} \)-open formula which defines the connective \( \sigma \).

We say a model \( \mathcal{I}(\mathcal{B}) \) reflects a formula \( E \) of the sort \( s \) occurring in a branch \( \mathcal{B} \) iff for \( n = \text{ar}(s) \) and for all ground terms \( t_1, \ldots, t_n \) we have that

\[
(E, \|t\|) \in \nu_s^{\mathcal{I}(\mathcal{B})} \quad \text{whenever} \quad \nu_s(E, \overline{t}) \in \mathcal{B}, \quad \text{and} \quad (E, \|t\|) \not\in \nu_s^{\mathcal{I}(\mathcal{B})} \quad \text{whenever} \quad \neg\nu_s(E, \overline{t}) \in \mathcal{B}.
\]

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

\[
(P, \|t\|) \in P^{\Delta^2(\mathcal{B})} \quad \text{whenever} \quad P(\overline{t}) \in \mathcal{B}, \quad \text{and} \quad (P, \|t\|) \not\in P^{\Delta^2(\mathcal{B})} \quad \text{whenever} \quad \neg P(\overline{t}) \in \mathcal{B}.
\]

A model \( \mathcal{I}(\mathcal{B}) \) reflects branch \( \mathcal{B} \) if \( \mathcal{I}(\mathcal{B}) \) reflects all predicate constants and formulae occurring in \( \mathcal{B} \).

A tableau calculus \( T \) is said to be constructively complete (for a logic \( \mathcal{L} \)) iff for any given set of formulae \( \mathcal{N} \), if \( \mathcal{B} \) is an open branch in a tableau derivation \( T(\mathcal{N}) \) then \( \mathcal{I}(\mathcal{B}) \) is an \( \mathcal{L} \)-model which reflects \( \mathcal{B} \). It is clear that if \( T \) is constructively complete for \( \mathcal{L} \) then \( T \) is complete for \( \mathcal{L} \).
Theorem 3. The tableau calculus $\text{ref}^I_{K_m}$ is sound and constructively complete for the logic $K_m$.

Proof. We prove the general rule refinement condition of Theorem 2 holds for any open branch $B$ of $\text{ref}^I_{K_m}$. The result is then a consequence of Corollary 1. Let $\nu((\alpha,\phi,t)\in B$ be in arbitrary open branch $B$ of a derivation in the refined tableau calculus $\text{ref}^I_{K_m}$. By the definition of $I(B)$, this means that $\nu((\alpha,\phi,t)\in B$. This implies that the refined rule has been applied to $\nu((\alpha,\phi,t)$ and $\nu((\alpha,\phi,t)\in B$ and, consequently, $\nu((\phi,t)\in B$.

Theorem 4. The tableau calculus $\text{ref}^I_{K_m(\neg)}$ is sound and constructively complete for the logic $K_m(\neg)$.

Proof. We prove the general rule refinement condition of Theorem 2 holds for any open branch $B$ of $\text{ref}^I_{K_m(\neg)}$. The result is then a consequence of Corollary 1. Let $\nu((\alpha,\phi,t)\in B$ be in arbitrary open branch $B$ of a derivation in the refined tableau calculus $\text{ref}^I_{K_m(\neg)}$ and $I(B)\not\models \nu((\alpha,\phi,t)\in B$. Therefore, the refined rule has been applied to $\nu((\phi,t)$ and $\nu((\alpha,\phi,t)\in B$. As a consequence, $\nu((\phi,t)\in B$.

If $\alpha$ is an atomic relation then, because $I(B)\models \nu((\alpha,t,t)$, we have that $\nu((\alpha,t,t)\in B$. Therefore, the refined rule has been applied to $\nu((\phi,t)$ and $\nu((\alpha,t,t)\in B$. As a consequence, $\nu((\phi,t)\in B$.

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

Therefore the general rule refinement condition holds for $\text{ref}^I_{K_m(\neg)}$ and, by Corollary 1, $\text{ref}^I_{K_m(\neg)}$ is constructively complete.

Theorem 5. Assume that for an open branch $B$ of the refined tableau $\text{ref}(\rho,T_L)$ and for every set $Y$ of $L$-formulae from $B$ the following holds.

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

$$X_0(\overline{t},t_1,\ldots,t_n) \subseteq B$$

implies that

$$X_1(\overline{E},t_1,\ldots,t_n) = \{\neg \xi_1,\ldots,\neg \xi_k\}$$

and all $\xi_1,\ldots,\xi_k$ are $L$-atomic.

Then, $B$ is reflected in $I(B)$.
Proof. We show the general rule refinement condition holds in this case. Assume \( X_0(E, t_1, \ldots, t_n) \) is contained in \( B \) and \( I(B) \models X_1(E, \|t_1\|, \ldots, \|t_n\|) \). Therefore there is some \( j = 1, \ldots, k \) such that \( I(B) \models \xi_j(\|t_1\|, \ldots, \|t_n\|) \). Since \( \xi_j(t_1, \ldots, t_n) \) is \( \mathcal{L} \)-atomic, by the definition of \( I(B) \) we have \( \xi_j(t_1, \ldots, t_n) \in B \). Consequently, the rule \( \rho_j \) has been applied in \( B \) to the set of premises \( X_0(E, t_1, \ldots, t_n) \cup \{ \xi_j(t_1, \ldots, t_n) \} \) and, hence, for some \( i = 2, \ldots, m \), the set \( X_i(E, t_1, \ldots, t_n) \) is contained in the branch \( B \). Finally, by Theorem 2, we have that \( B \) is reflected in \( I(B) \).

**Theorem 8.** \( T_{c^+}^L \) is sound and constructively complete for the logic \( L \).

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

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

Therefore, the rules \([\text{split}_k]\) and the rules \([\text{split}_{mn}]\) are derivable from each other. The theorem statement follows immediately.