Semantic Labelled Tableaux for Propositional $\mathbf{BI}^{\perp}$

Didier Galmiche and Daniel Méry

LORIA - Université Henri Poincaré
Campus Scientifique, BP 239
Vandœuvre-lès-Nancy, France
galmiche{dmery}@loria.fr

Abstract

In this paper, we study semantic labelled tableaux for the propositional Bunched Implications logic (**BI**) that freely combines intuitionistic logic (**IL**) and multiplicative intuitionistic linear logic (**MILL**). **BI** is a resource-aware logic that captures interferences between resources and it is well suited, because of its resource-based sharing interpretation, for reasoning about mutable data structures. We propose a labelled tableau calculus for $\mathbf{BI}^{\perp}$[1] based on particular labels and constraints. We prove the soundness and completeness of this calculus w.r.t. the Kripke resource semantics with an emphasis on countermodel construction. In addition, we prove the finite model property and as a consequence the decidability of $\mathbf{BI}^{\perp}$. Moreover, we analyze some algorithmic aspects of the tableau construction by providing a free variable variant of the calculus. We also develop the restrictions to **IL** and **MILL** that provide new tableau methods for both logics with generation of countermodels.

1 Introduction

In this paper, we consider the propositional Bunched Implications (**BI**) logic, in which multiplicative (or linear) and additive (or intuitionistic) connectives cohabit [29]. This resource-aware logic can be viewed as a merging of intuitionistic logic (**IL**) and multiplicative intuitionistic linear logic (**MILL**) that allows to capture interferences between resources. It can be well understood through its Kripke resource models [21] but also proof-theoretically through its bunched sequent calculus [34] and the corresponding $\alpha\lambda$-calculus [27]. With its so-called sharing interpretation, the **BI** logic appears as an appropriate logical foundation for logic programming [2], for reasoning about mutable data structures [20] and for problems dealing with interferences between resources. In this context, we aim to study proof-search methods that build proofs or countermodels in case of non-provability. There exist such methods, based on a sequent calculus [13] or a labelled tableau calculus [36], for **IL** and **MILL**. But they cannot be adapted or extended to **BI** and therefore specific notions and concepts seem necessary in order to capture the interferences between the different connectives which occur during the proof-search process.
We know that the tableau method directly provides countermodel generation for a wide range of logics, including **IL**, but not for substructural logics like linear logic [25, 26]. In this work, we focus on the Kripke resource semantics of **BI** [21] in which the possible worlds are justified in terms of pieces of information [37]. It is important to mention that the unit $\perp$ of the additive disjunction internalizes inconsistency in **BI** and that the Kripke resource semantics does not account for inconsistency ($\perp$ is nowhere forced). Accordingly, the completeness result w.r.t. this semantics only holds for $\mathbf{BI}^{\perp}$ [29, 35], fragment to which we initially restrict our study. A key point of our work consists in defining labels and labelled formulas, the role of which being to capture some semantical information into the syntax. Then, we propose a labelled tableau calculus for propositional $\mathbf{BI}^{\perp}$ with expansion rules that generate constraints on labels. Such constraints are able to capture the semantics of the connective interactions occuring in the proof-search. Thus, we define a tableau construction procedure that is characterized by the generation of two

[1] $\mathbf{BI}^{\perp}$denotes the BI logic without $\perp$, that is the unit of the additive disjunction $\vee$.

particular kinds of label constraints, called *assertions* and *requirements*. In this context, the provability is characterized through a tableau construction with specific closure conditions in which the set of requirements has to be verified from the set of assertions. Our labelled calculus is proved sound and complete (by a countermodel construction method). For a given $\mathbf{BI}^{\perp}$ formula A, we provide a procedure that yields either a tableau proof or a countermodel of A. Moreover, we prove the finite model property for propositional $\mathbf{BI}^{\perp}$ w.r.t. Kripke resource semantics, property which is important but not trivial for substructural logics [32]. As a consequence, we give a tableau construction procedure that terminates and generates finite countermodels in case of non-provability. We observe that the set of assertions can be represented as a graph, called *dependency graph*, that is built in parallel with the tableau for a formula A. Then, the requirement verification corresponds to checking the existence of some paths. Moreover, if A is not provable then a countermodel can be directly extracted from the related dependency graph. Such a graph appears as an essential semantical structure that captures the interactions between additives and multiplicatives inside **BI**. To have efficient constraint resolution is essential for an effective use of our labelled method and thus we define a variant of it based on free variables in labels and in which we can postpone some variable instantiations until enough information is available. It gives a way to control the introduction of new labels and therefore to limit the syntactic complexity of the labels. But, as a consequence, the construction of countermodels becomes more difficult.
As **BI** is conservative over **IL** and **MILL** [29], we have defined new labelled calculi for both logics, by some restrictions of the $\mathbf{TBI}^{\perp}$ calculus. Compared with other proof-search methods, these new methods for these logics provide efficient proof-search with an easy and direct generation of countermodels.
Our results could be related to the Labelled Deductive Systems (LDS) methodology [10] where the deductive process manipulates labelled formulas and inference rules simultaneously act on formulas and labels. This approach has been used in order to build uniform deductive systems for a wide range of substructural logics [5, 7]. From a set of labelled rules, that is the same for all the logics under consideration, the provability in each logic is analyzed through specific label constraints. The common idea with our approach is to capture the semantics through labels and to analyze provability through constraint satisfaction, but, in our work, we deal with a specialized, and not uniform, deductive system and with particular constraints (assertions and requirements) on labels, that are generated during the tableau construction. Moreover, the closure conditions involve specific constraints and are not instantiations of some more uniform constraints. Our way for defining and managing labels and constraints is strongly related to the resource semantics of **BI**. Further work will be devoted to the study of such relationships in order to determine if and how similar labelled calculi could be define for logics with well-defined resource semantics.

In section 2, we present **BI** with its basic proof-theory and Kripke resource semantics. The completeness w.r.t. this semantics only holds for $\mathbf{BI}^{\perp}$ (**BI** without $\perp$) [34] and consequently we first deal with this fragment. In section 3, we introduce the labelled tableau calculus $\mathbf{TBI}^{\perp}$ with appropriate definitions of *labels*, *labelled formulas* and *constraints*. We define expansion rules that include the generation of two kinds of constraints called *assertions* and *requirements*. The closure condition of a labelled tableau is defined through a particular notion of *complementarity* and the satisfaction of requirements from the assertions. Examples illustrate the construction of labelled tableaux with constraints for $\mathbf{BI}^{\perp}$ and its underlying problems. In section 4, we show the main properties of the $\mathbf{TBI}^{\perp}$ calculus that are soundness and completeness w.r.t. the Kripke resource semantics. In order to prove completeness we introduce the notion of *completed branch*, show how to construct a countermodel from such a branch [30] and give a tableau construction procedure that provides either a closed tableau or a tableau with a completed branch. Moreover, we prove the finite model property w.r.t. the Kripke semantics from the introduction of the notions of *liberalized expansion rules* and *branch redundancy*. As a consequence, we show the decidability of $\mathbf{BI}^{\perp}$. In section 5, we introduce the notion of *dependency graph* that represents the set of assertions introduced during the tableau construction. Then, the verification of requirements corresponds to the search of specific paths in such a graph. It is an essential semantical structure from which we can deduce the provability of a formula or its non-provability with the extraction of a Kripke resource countermodel following the steps of the completeness proof. In section 6, we give a free variable version of the $\mathbf{TBI}^{\perp}$ calculus and illustrate with examples its interest for proof-search in comparison with the initial calculus. It allows to strongly reduce the non-determinism in the choice of the labels satisfying the requirements and then to improve the efficiency of the proof-search but the construction of countermodels is more difficult. In sections 7 and 8, taking into account the conservativity theorem of **BI**, we restrict the $\mathbf{TBI}^{\perp}$ calculus respectively to **IL** and **MILL** in order to obtain, for each logic, a new tableau method based on labels and constraints with an easy generation of countermodels as a main characteristic. In section 9, we present some conclusions and perspectives.

2 The Bunched Implications Logic (BI)

In this section, we recall the essential features of the propositional logic of Bunched Implications (**BI**) [34], in which additive (intuitionistic) and multiplicative (linear) connectives cohabit. Such a logic requires that the antecedents of logical consequences are structured not as lists but rather as bunches.

2.1 Syntax and Proof-Theory of BI

Definition 2.1 (propositions). *The propositional language of* **BI** *consists of: a multiplicative unit* I*, the multiplicative connectives* $*$, $-\!\!*$*, the additive units* $\top$, $\bot$*, the additive connectives* $\wedge$, $\rightarrow$, $\vee$*, a countable set* $L = \mathrm{p}, \mathrm{q}, \ldots$ *of propositional letters.* $\mathcal{P}(L)$*, the collection of* **BI** *propositions over* L*, is given by the following inductive definition:* $\mathrm{A} ::= \mathrm{p} \mid \mathrm{I} \mid \mathrm{A} * \mathrm{A} \mid \mathrm{A} -\!\!* \mathrm{A} \mid \top \mid \bot \mid \mathrm{A} \wedge \mathrm{A} \mid \mathrm{A} \rightarrow \mathrm{A} \mid \mathrm{A} \vee \mathrm{A}$.

The additive connectives correspond to those of intuitionistic logic (**IL**) whereas the multiplicative connectives correspond to those of multiplicative intuitionistic linear logic (**MILL**). The antecedents of logical consequences are structured as bunches, in which there are two ways to combine operations that respectively display additive and multiplicative behavior.

Definition 2.2 (bunches). *Bunches are inductively defined as follows:* $\Gamma ::= \mathrm{A} \mid \emptyset_m \mid \Gamma, \Gamma \mid \emptyset_a \mid \Gamma; \Gamma$.

We write $\Gamma(\Delta)$ to represent a bunch in which Δ is a subtree. We consider "bunches" up to a structural equivalence ($\equiv$) defined as the least equivalence on "bunches" satisfying: a) commutative monoid equations for $\emptyset_a$ and ";", b) commutative monoid equations for $\emptyset_m$ and "," and c) congruence: $\Delta \equiv \Delta'$ implies $\Gamma(\Delta) \equiv \Gamma(\Delta')$. Let us note that ";" and "," do not distribute over one another.
Judgements are expressions of the form $\Gamma \vdash \mathrm{A}$, where Γ is a "bunch" and A is a proposition. The **LBI** sequent calculus is given in figure 1. It has the cut-elimination property and there exists an equivalent sequent calculus with the following additive rules that are derivable in the **LBI** calculus.

$$\frac{}{\Gamma; \mathrm{A} \vdash \mathrm{A}}\, ax \qquad \frac{\Gamma \vdash \mathrm{A} \quad \Gamma \vdash \mathrm{B}}{\Gamma \vdash \mathrm{A} \wedge \mathrm{B}}\, \wedge_R \qquad \frac{\Gamma \vdash \mathrm{A} \quad \Delta(\Gamma; \mathrm{B}) \vdash \mathrm{C}}{\Delta(\Gamma; \mathrm{A} \rightarrow \mathrm{B}) \vdash \mathrm{C}}\, \rightarrow_L \qquad \frac{\Gamma(\mathrm{A}) \vdash \mathrm{C} \quad \Gamma(\mathrm{B}) \vdash \mathrm{C}}{\Gamma(\mathrm{A} \vee \mathrm{B}) \vdash \mathrm{C}}\, \vee_L$$

As an example, we give a proof in **LBI** of the sequent $\mathrm{p} * (\mathrm{q} \wedge \mathrm{r}) \vdash (\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})$.

$$\dfrac{\dfrac{\dfrac{}{\mathrm{p} \vdash \mathrm{p}} \quad \dfrac{\dfrac{}{\mathrm{q}; \mathrm{r} \vdash \mathrm{q}}}{\mathrm{q} \wedge \mathrm{r} \vdash \mathrm{q}}\, \wedge_L}{\mathrm{p} * (\mathrm{q} \wedge \mathrm{r}) \vdash \mathrm{p} * \mathrm{q}}\, *_R \quad \dfrac{\dfrac{}{\mathrm{p} \vdash \mathrm{p}} \quad \dfrac{\dfrac{}{\mathrm{q}; \mathrm{r} \vdash \mathrm{r}}}{\mathrm{q} \wedge \mathrm{r} \vdash \mathrm{r}}\, \wedge_L}{\mathrm{p} * (\mathrm{q} \wedge \mathrm{r}) \vdash \mathrm{p} * \mathrm{r}}\, *_R}{\mathrm{p} * (\mathrm{q} \wedge \mathrm{r}) \vdash (\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})}\, \wedge_R$$

A proposition A is a theorem if $\emptyset_a \vdash \mathrm{A}$ or $\emptyset_m \vdash \mathrm{A}$ is provable in **LBI** [29]. But the following theorem provides a more simple definition.

Theorem 2.1. $\emptyset_a \vdash \mathrm{A}$ *(resp.* $\emptyset_m \vdash \mathrm{A}$*) is provable in* **LBI** *iff* $\top \vdash \mathrm{A}$ *(resp.* $\mathrm{I} \vdash \mathrm{A}$*) is provable in* **LBI**.

Corollary 2.1. *A proposition* A *is a theorem iff* $\emptyset_m \vdash \mathrm{A}$ *is provable in* **LBI**.

It is important to mention the following results about conservativity over **IL** and **MILL**.

– Identity and structures

$$\frac{}{\mathrm{A} \vdash \mathrm{A}}\, ax \qquad \frac{\Gamma \vdash \mathrm{A}}{\Delta \vdash \mathrm{A}}\, \Delta \equiv \Gamma \qquad \frac{\Gamma(\Delta) \vdash \mathrm{A}}{\Gamma(\Delta; \Delta') \vdash \mathrm{A}}\, w \qquad \frac{\Gamma(\Delta; \Delta) \vdash \mathrm{A}}{\Gamma(\Delta) \vdash \mathrm{A}}\, c$$

– Units

$$\frac{}{\bot \vdash \mathrm{A}}\, \bot_L \qquad \frac{\Gamma(\emptyset_m) \vdash \mathrm{A}}{\Gamma(\mathrm{I}) \vdash \mathrm{A}}\, \mathrm{I}_L \qquad \frac{}{\emptyset_m \vdash \mathrm{I}}\, \mathrm{I}_R \qquad \frac{\Gamma(\emptyset_a) \vdash \mathrm{A}}{\Gamma(\top) \vdash \mathrm{A}}\, \top_L \qquad \frac{}{\emptyset_a \vdash \top}\, \top_R$$

– Multiplicatives

$$\frac{\Gamma(\mathrm{A}, \mathrm{B}) \vdash \mathrm{C}}{\Gamma(\mathrm{A} * \mathrm{B}) \vdash \mathrm{C}}\, *_L \qquad \frac{\Gamma \vdash \mathrm{A} \quad \Delta \vdash \mathrm{B}}{\Gamma, \Delta \vdash \mathrm{A} * \mathrm{B}}\, *_R \qquad \frac{\Gamma \vdash \mathrm{A} \quad \Delta(\Delta', \mathrm{B}) \vdash \mathrm{C}}{\Delta(\Delta', \Gamma, \mathrm{A} -\!\!* \mathrm{B}) \vdash \mathrm{C}}\, -\!\!*_L \qquad \frac{\Gamma, \mathrm{A} \vdash \mathrm{B}}{\Gamma \vdash \mathrm{A} -\!\!* \mathrm{B}}\, -\!\!*_R$$

– Additives

$$\frac{\Gamma(\mathrm{A}; \mathrm{B}) \vdash \mathrm{C}}{\Gamma(\mathrm{A} \wedge \mathrm{B}) \vdash \mathrm{C}}\, \wedge_L \qquad \frac{\Gamma \vdash \mathrm{A} \quad \Delta \vdash \mathrm{B}}{\Gamma; \Delta \vdash \mathrm{A} \wedge \mathrm{B}}\, \wedge_R \qquad \frac{\Gamma \vdash \mathrm{A} \quad \Delta(\Delta'; \mathrm{B}) \vdash \mathrm{C}}{\Delta(\Delta'; \Gamma; \mathrm{A} \to \mathrm{B}) \vdash \mathrm{C}}\, \to_L$$

$$\frac{\Gamma; \mathrm{A} \vdash \mathrm{B}}{\Gamma \vdash \mathrm{A} \to \mathrm{B}}\, \to_R \qquad \frac{\Gamma(\mathrm{A}) \vdash \mathrm{C} \quad \Delta(\mathrm{B}) \vdash \mathrm{C}}{\Gamma(\mathrm{A} \vee \mathrm{B}); \Delta(\mathrm{A} \vee \mathrm{B}) \vdash \mathrm{C}}\, \vee_L \qquad \frac{\Gamma \vdash \mathrm{A}_{i\,(i=1,2)}}{\Gamma \vdash \mathrm{A}_1 \vee \mathrm{A}_2}\, \vee_{Ri}$$

Figure 1: The **LBI** sequent calculus

Theorem 2.2 (conservativity). **BI** *is conservative over* **IL**, *that is,* $\mathrm{A}_1; \ldots; \mathrm{A}_n \vdash \mathrm{B}$ *is provable in* **LBI** *iff* $\mathrm{A}_1, \ldots, \mathrm{A}_n \vdash \mathrm{B}$ *is provable in* **IL**, *where the* A_i *and* B *are propositions built up using additive connectives only.* **BI** *is conservative over* **MILL**, *that is,* $\mathrm{A}_1, \ldots, \mathrm{A}_n \vdash \mathrm{B}$ *is provable in* **LBI** *iff* $\mathrm{A}_1, \ldots, \mathrm{A}_n \vdash \mathrm{B}$ *is provable in* **MILL**, *where the* A_i *and* B *are propositions built up using multiplicative connectives only and these ones are replaced by the corresponding connectives in* **MILL**.

Let us remark that the conservativity does not extend to MAILL (MILL extended with additive connectives) that does not admit distribution, namely $\mathrm{A} \wedge (\mathrm{B} \vee \mathrm{C}) \vdash (\mathrm{A} \wedge \mathrm{B}) \vee (\mathrm{A} \wedge \mathrm{C})$ and its converse whereas **BI** admits it. All these results are developed and proved in [29, 35].

2.2 Kripke Resource Semantics

BI has two natural semantics: a categorical semantics of the proof theory (interpretation of proofs) using a doubly closed category (two monoidal closed structures) that has all the structure to model both **IL** and **MILL** and a Kripke-style semantics (interpretation of formulas) which combines the Kripke semantics of **IL** and Urquhart's semantics of **MILL** [29] that deals with possible worlds, arranged as a commutative monoid and justified in terms of "pieces of information" [37]. Here, we consider such a semantics that provides a way to read the formulas as propositions that are true or false relative to a given world. Let us summarize the main points about this semantics that is based on the notion of *resource*.

Definition 2.3. *A* Kripke resource monoid *(KRM)* $\mathcal{M} = \langle M, \sqsubseteq, \bullet, e \rangle$ *is a preordered commutative monoid in which* $\bullet$ *is order preserving, i.e., (P)* $m \sqsubseteq n$ *and* $m' \sqsubseteq n'$ *imply* $m \bullet m' \sqsubseteq n \bullet n'$.

Definition 2.4. *Let* $\mathcal{M}$ *be a KRM and* $\mathcal{P}(L)$ *be the collection of* **BI** *propositions over a language* L *of propositional letters, then an* elementary Kripke interpretation *(EKI) is a function* $[\![-]\!] : L \to \mathcal{P}(M)$ *that satisfies: (K) for any* $m, n \in M$ *such that* $m \sqsubseteq n$, $m \in [\![\mathrm{p}]\!]$ *implies* $n \in [\![\mathrm{p}]\!]$.

Definition 2.5. *An* elementary Kripke model *(EKM), is a triple* $\mathcal{K} = \langle \mathcal{M}, \models, [\![-]\!] \rangle$ *in which* $\mathcal{M}$ *is a KRM,* $[\![-]\!]$ *is an EKI and* $\models$ *is a forcing relation on* $M \times \mathcal{P}(L)$ *satisfying the following conditions:*

- $m \models \mathrm{p}$ *iff* $m \in [\![\mathrm{p}]\!]$
- $m \models \top$ *iff* *always*
- $m \models \bot$ *iff* *never*
- $m \models \mathrm{A} \wedge \mathrm{B}$ *iff* $m \models \mathrm{A}$ *and* $m \models \mathrm{B}$
- $m \models \mathrm{A} \vee \mathrm{B}$ *iff* $m \models \mathrm{A}$ *or* $m \models \mathrm{B}$
- $m \models \mathrm{A} \to \mathrm{B}$ *iff* $(\forall n \in M \ / \ m \sqsubseteq n)$ ($n \models \mathrm{A}$ *implies* $n \models \mathrm{B}$)
- $m \models \mathrm{I}$ *iff* $e \sqsubseteq m$
- $m \models \mathrm{A} * \mathrm{B}$ *iff* $(\exists n, n' \in M \ / \ n \bullet n' \sqsubseteq m)$ ($n \models \mathrm{A}$ *and* $n' \models \mathrm{B}$)
- $m \models \mathrm{A} \mathrel{-\!\!*} \mathrm{B}$ *iff* $(\forall n \in M)$ ($n \models \mathrm{A}$ *implies* $m \bullet n \models \mathrm{B}$)

The semantics of propositions given by the relation $\models$ is parametrized by the interpretation $[\![-]\!]$ for which the property (K) of definition 2.4 holds for atomic propositions. It is easy to prove, by structural induction on propositions, that if (K) holds for atomic propositions, then it also holds for any proposition.

These semantic clauses can be read in terms of resources. For instance, one says that a world forces $\mathrm{A} * \mathrm{B}$ if and only if there is a partition of that world into components which force A and B. On the other hand, one can understand the formula $\mathrm{A} \mathrel{-\!\!*} \mathrm{B}$ as follows: if m is the cost of the function and n is the cost of any argument then the cost of obtaining the application of the function to the argument is $m \bullet n$. Then, the multiplicatives depend on the internal structure of worlds and the additives should be understood in terms of conservation [29]. From this resource-based interpretation of **BI**, it is important to notice that **BI** is different from Linear Logic (LL) and does not admit its standard number-of-uses reading [19]. For instance, $\mathrm{A} \mathrel{-\!\!*} \mathrm{B} \vdash \mathrm{A} \to \mathrm{B}$ does not hold in **BI**, whereas $\mathrm{A} \multimap \mathrm{B} \vdash \mathrm{A} \to \mathrm{B}$ holds in LL, in which $\mathrm{A} \to \mathrm{B}$ is defined as $!\mathrm{A} \multimap \mathrm{B}$. On the other hand, $\vdash \mathrm{A} \mathrel{-\!\!*} (\mathrm{A} \to \mathrm{A} \to \mathrm{B}) \to \mathrm{B}$ holds in **BI** but not in LL because it does not respect the number-of-uses reading.

There is a recent interest in so-called resource-sensitive logics like linear logic [19] but in **BI** one does not regard a proposition as a resource and thus a proof as a way to manipulate resources. The reading is declarative: a proposition is a statement about the world the judgement of truth of which may involve consideration of resources. It may be seen most clearly through the term models required for the various completeness proofs and also in the *sharing interpretation* [35]. There are promising results about logic programming [2], interference in imperative programs [27] and reasoning about mutable data structures like pointers [20]. For instance, the problem that arises when dealing with pointers is that you can have several pointers referencing the same cell in the computer store. This situation, called *aliasing*, is typically a problem of resource sharing once the resources have been defined as being "heaps" or, in other words, "portions of the computer store". The computer store is made of cons cells having basic data (integers, for example), or pointers to other cons cells in their components. The model proposed in [20] deals with atomic propositions of the form $(x \mapsto E, F)$ which means that x is a location pointing to a cons cell with first and second components respectively containing E and F as values. An interesting example, which shows that $\mathrel{-\!\!*}$ is about describing fresh heaps, is given by the formula $(x \mapsto 3, 5) * ((x \mapsto 7, 5) \mathrel{-\!\!*} P)$. This formula can only be satisfied by heaps which contain a location x pointing to a cons cell whose content is $3, 5$ and such that, when updating the cons cell to $7, 5$, the resulting heap satisfies P. It corresponds to a description of the update operation in terms of deletion followed by extension. Similar ideas have been successfully applied to write pre and post-conditions, using **BI** as an assertion language, for operations such as assignment, allocation and deallocation of a cons cell [20]. In order to prove such formulas, assertions or properties expressed in **BI**, it seems important to design proof-search methods for this logic that build proofs but also generate countermodels.

Definition 2.6. *Let $\mathcal{K}$ be a EKM and Φ_Γ be the formula obtained from a bunch Γ by replacing each ";" by $\wedge$ and each "," by $*$ with association respecting the tree structure of Γ, the sequent $\Gamma \vdash \mathrm{A}$ is said to be* valid in $\mathcal{K}$ *(notation: $\Gamma \models_{\mathcal{K}} \mathrm{A}$), if and only if, for any world $m \in M$, $m \models \Phi_\Gamma$ implies $m \models \mathrm{A}$.*
The sequent $\Gamma \vdash \mathrm{A}$ is said to be valid *(notation: $\Gamma \models \mathrm{A}$), if and only if, for any EKM $\mathcal{K}$, $\Gamma \models_{\mathcal{K}} \mathrm{A}$.*

Theorem 2.3 (soundness). *If $\Gamma \vdash \mathrm{A}$ is provable in* **LBI**, *then $\Gamma \models \mathrm{A}$.*

The unit $\bot$ of the $\vee$ connective internalizes inconsistency in **BI** and the Kripke resource semantics does not account for inconsistency ($\bot$ is nowhere forced). Accordingly, $\bot$ has to be excluded for completeness [35] and then the completeness result w.r.t. this semantics only holds for $\mathbf{BI}^{\bot}$ (**BI** without $\bot$) [29].

Theorem 2.4 (completeness). *If* $\Gamma \models \mathrm{A}$ *in* $\mathbf{BI}^{\perp}$ *then* $\Gamma \vdash \mathrm{A}$ *is provable in* $\mathbf{LBI}^{\perp}$.

Corollary 2.2. *A proposition* A *is valid iff for any EKM* $\mathcal{K}$, $e \models_{\mathcal{K}} \mathrm{A}$.

Our goal is to define a labelled tableau method, the labels allowing to incorporate information about semantics into the syntax. In this context, the Kripke resource semantics seems to be a good starting point to define labels and labelled calculi dedicated to efficient and understandable proof-search methods. As we aim to design them from so-called completeness proofs by proof-search [30], we consider at first $\mathbf{BI}^{\perp}$ hoping that we could adapt our methodology and results to **BI** (with $\perp$). Topological Kripke models and also Grothendieck Sheaf-theoretic models have been proposed as semantics to have completeness for **BI** [35].

3 A Labelled Tableaux Calculus for $\mathbf{BI}^{\perp}$

Because of the cohabitation of multiplicative (linear) and additive (intuitionistic) connectives in **BI** and even if it includes **IL** and **MILL** as subsystems, we cannot directly extend proof-search (sequent, tableau, or connection) calculi defined for both sublogics to **BI** [3, 11, 13, 36]. We introduce labels and labelled formulas in a specific way compared to the general LDS approach of [10] in order to define a labelled tableau calculus $\mathbf{TBI}^{\perp}$, for propositional $\mathbf{BI}^{\perp}$, with generation of countermodels. Let us start defining this particular set of labels and constraints.

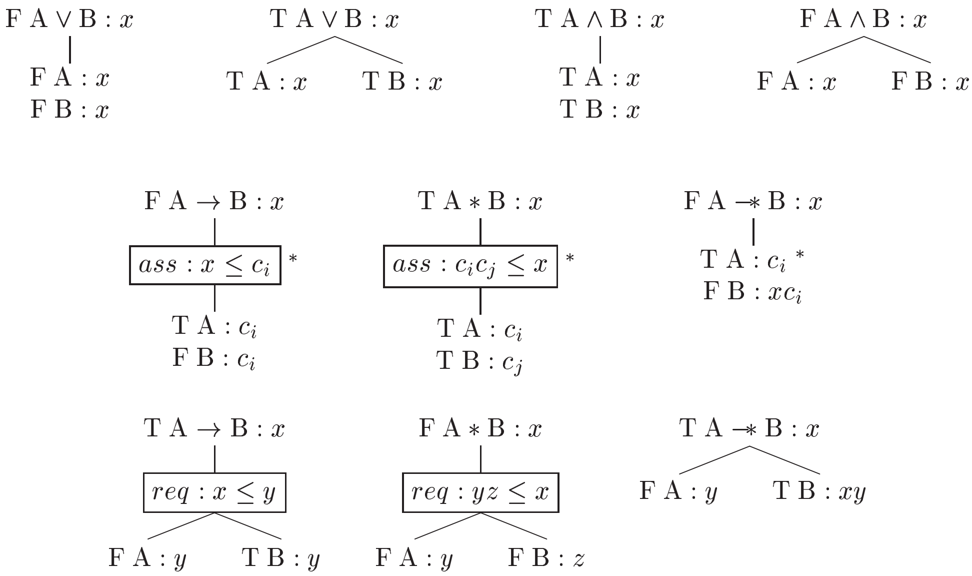
3.1 Labelled Tableaux

Definition 3.1. *A* labelling language *consists of the following symbols: a unit symbol* 1*, a binary function symbol* $\circ$*, a binary relation symbol* $\leq$*, a countable set of constants* $c_1, c_2, \ldots$
Labels *are inductively defined from the unit* 1 *and the constants as expressions of the form* $x \circ y$ *in which* x *and* y *are labels.* Atomic labels *are labels which do not contain any* $\circ$*, while* compound labels *contain at least one* $\circ$*.* Label constraints *are expressions of the form* $x \leq y$ *with* x *and* y *being labels.*

In order to simplify notations, we can omit the binary symbol $\circ$ when we write labels. Like in the LDS approach the basic units of the deductive process are *labelled formulas* with labels in a given labelling algebra [7]. The expansion rules act on the labels as well as on the formulas.
In this context, a *(labelled) signed formula* is a triple $\langle\, \mathrm{Sg}, \mathrm{A}, l\,\rangle$, denoted $\mathrm{Sg}\ \mathrm{A} : l$, Sg ($\in \{\, \mathrm{F}, \mathrm{T}\,\}$) being the sign of the formula A ($\in \mathcal{P}(L)$) and l ($\in \mathcal{L}$) being its label.
A $\mathbf{TBI}^{\perp}$ tableau is a rooted tree, with nodes labelled with a signed formula, that is built according to the expansion rules of figure 2. The rules of the first line are the classical α and β rules, like in standard tableau methods [9]. The rules of the second line (called $\pi\alpha$ rules) are such that the labels c_i and c_j are new constants and the label constraints are called *assertions*. Let us note that we assume the implicit assertion $c_i \leq c_i$ for the F $-\!\!*$ rule (not depicted on figure 2 for conciseness). The rules of the third line (called $\pi\beta$ rules) are such that the labels y and z only contain constants that exist at the moment of the expansion and the constraints are called *requirements*.
For example, in the case of $\mathrm{A} * \mathrm{B}$, we have the two following rules:

$$\begin{array}{c} \mathrm{T}\ \mathrm{A} * \mathrm{B} : x \\ | \\ \boxed{ass : c_i c_j \leq x} \\ | \\ \mathrm{T}\ \mathrm{A} : c_i \\ \mathrm{T}\ \mathrm{B} : c_j \end{array} \qquad\qquad \begin{array}{c} \mathrm{F}\ \mathrm{A} * \mathrm{B} : x \\ | \\ \boxed{req : yz \leq x} \\ \diagup \quad \diagdown \\ \mathrm{F}\ \mathrm{A} : y \qquad \mathrm{F}\ \mathrm{B} : z \end{array}$$

Both rules introduce constraints on the labels. The first one introduces the new constants c_i and c_j and an assertion $c_i c_j \leq x$. The second one introduces the labels y and z and a requirement $yz \leq x$.
We have distinguished two kinds of constraints: the assertions and the requirements. In the proof-search process, the former will behave as known facts (or hypotheses) and the latter will express goals that must



* c_i, c_j are new constants

Figure 2: Expansion Rules for $\mathbf{BI}^{\perp}$

be satisfied from the assertions. Since there may be several labels satisfying a given requirement, the $\pi\beta$ rules are different from the $\pi\alpha$ rules because they can be expanded several times during a tableau expansion, much like the γ rules in the first order classical tableau method [9].

A key point is that the labels and constraints introduced in this labelled tableau calculus are interpreted in a labelling algebra.

Definition 3.2. *Labels and constraints are interpreted onto a* labelling algebra $\mathcal{L} = \langle L, \leq, \circ, 1 \rangle$, *which is an order-preserving preordered commutative monoid of labels, more precisely:*

1. *L is a set of labels*
2. *$\leq$ is a preordering*
3. *equality on labels is defined by : $x = y$ iff $x \leq y$ and $y \leq x$*
4. *$\circ$ is a binary operation on L that satisfies:*
   - *associativity: $(x \circ y) \circ z = x \circ (y \circ z)$*
   - *commutativity: $x \circ y = y \circ x$*
   - *identity: $x \circ 1 = 1 \circ x = x$*
   - *compatibility: $x \leq y$ implies $x \circ z \leq y \circ z$*

We say that x is a sublabel *of y (notation: $x \preceq y$), if there exists a label z such that $y = x \circ z$. We note $x \prec y$ if $x \preceq y$ and $x \neq y$. $\mathcal{P}(x)$ denotes the set of the sublabels of x.*

Let us remark that our labelling algebra is a free monoid of labels and, consequently, it generates an infinity of labels. Indeed, once you have a constant c_1, you also have c_1c_1, $c_1c_1c_1$ and so on. Nevertheless, the whole free monoid is often useless for verifying whether some constraints hold with respect to a given labelling algebra. This point can be justified if we consider labels as resources and formulas as statements about resources. Then, in many cases, formulas only express something about a (finite) subset of the resources. Therefore, we deal with *partially defined* labelling algebras which are obtained from sets of constraints by means of a specific closure operator.

Definition 3.3. *The* domain *of a set K of label constraints is the set of all sublabels occurring in some constraint of K, i.e., $\mathcal{D}(K) = \bigcup_{x \leq y \in K}(\mathcal{P}(x) \cup \mathcal{P}(y))$. The* closure $\overline{K}$ *of K is defined as follows:*

1. $K \subseteq \overline{K}$
2. if $x \in \mathcal{D}(\overline{K})$ *then* $x \leq x \in \overline{K}$ *(reflexivity)*
3. if $x \leq y \in \overline{K}$ *and* $y \leq z \in \overline{K}$ *then* $x \leq z \in \overline{K}$ *(transitivity)*
4. if $y \circ z \in \mathcal{D}(\overline{K})$ *then* $x \leq y \in \overline{K}$ *implies* $x \circ z \leq y \circ z \in \overline{K}$ *(compatibility)*

Any set K of constraints gives rise to a particular preordering relation and thus to a particular labelling algebra in which the constraints of K are taken as axioms. Since the two notions are closely related, we do not distinguish between the closure of a set of constraints and the (partially defined) labelling algebra it generates.

Let us remark that each $\pi\beta$ rule admits a trivially satisfied requirement w.r.t. the underlying labelling algebra, namely, $x \leq x$ for $\mathrm{T} \rightarrow$ and $1 \circ x \leq x$ for $\mathrm{F}\ *$ and $\mathrm{T} \multimap$. Therefore, a $\pi\beta$ formula can always be expanded at least once. Moreover, since $\circ$ is assumed to be commutative, if y and z are labels such that $yz \leq x$ then the two following expansions are allowed for the formula $\mathrm{F\ A} * \mathrm{B} : x$:

$$
\begin{array}{c}
\mathrm{F\ A} * \mathrm{B} : x \\
| \\
\boxed{req : yz \leq x} \\
\mathrm{F\ A} : y \qquad \mathrm{F\ B} : z
\end{array}
\qquad\qquad
\begin{array}{c}
\mathrm{F\ A} * \mathrm{B} : x \\
| \\
\boxed{req : zy \leq x} \\
\mathrm{F\ A} : z \qquad \mathrm{F\ B} : y
\end{array}
$$

Moreover, each $\pi\alpha$ rule admits a special case when the multiplicative unit I is concerned:

$$
\begin{array}{c}
\mathrm{T\ A} * \mathrm{I} : x \\
| \\
\mathrm{T\ I} : 1 \\
\mathrm{T\ A} : x
\end{array}
\qquad\qquad
\begin{array}{c}
\mathrm{T\ I} * \mathrm{A} : x \\
| \\
\mathrm{T\ I} : 1 \\
\mathrm{T\ A} : x
\end{array}
\qquad\qquad
\begin{array}{c}
\mathrm{F\ I} \multimap \mathrm{A} : x \\
| \\
\mathrm{T\ I} : 1 \\
\mathrm{F\ A} : x
\end{array}
\qquad\qquad
\begin{array}{c}
\mathrm{F\ I} \rightarrow \mathrm{A} : x \\
| \\
\boxed{ass : x, 1 \leq c_i} \\
| \\
\mathrm{T\ I} : c_i \\
\mathrm{F\ A} : c_i
\end{array}
$$

The rules for $\mathrm{T}\ *$ and $\mathrm{F} \multimap$ introduce neither new constants nor new assertions and correspond to the equivalences $\mathrm{I} * \mathrm{A} \dashv\vdash \mathrm{A} * \mathrm{I} \dashv\vdash \mathrm{A}$ and $\mathrm{I} \multimap \mathrm{A} \dashv\vdash \mathrm{A}$ in **LBI**. In the case of $\mathrm{F\ I} \rightarrow \mathrm{A} : x$, a new assertion $1 \leq c_i$, reflecting on c_i the semantical satisfaction condition for I, is required in addition to the usual assertion $x \leq c_i$.

We now introduce useful notations before we define what a closed labelled tableau is. Given a branch $\mathcal{B}$, we write $\mathcal{F}(\mathcal{B})$ to denote the set of its signed-formulas, $Ass(\mathcal{B})$ to denote the set of its assertions and $Req(\mathcal{B})$ to denote the set of its requirements. We define the *domain* $\mathcal{D}(\mathcal{B})$ of $\mathcal{B}$ as the set $\mathcal{D}(Ass(\mathcal{B}))$. Finally, $C(\mathcal{B})$ is the set $\mathcal{D}(\mathcal{B})$ restricted to the constants. We can now proceed with the notion of closed tableau.

Definition 3.4. *Let* A *be a proposition of* $\mathbf{BI}^{\perp}$*, a* tableau sequence for A *is a sequence of tableaux* $t_1, t_2, \ldots$ *for which* t_1 *is the one-node tree defined by* $\mathcal{F}(t_1) = \{\, \mathrm{F\ A} : 1 \,\}$, $Ass(t_1) = \{\, 1 \leq 1 \,\}$, $Req(t_1) = \emptyset$ *and* t_{i+1} *is obtained from* t_i *by the application of an expansion rule of figure 2 (on a branch of* t_i*).*

Definition 3.5. *Let* $\mathcal{B}$ *be a tableau branch, the two signed formulas* $\mathrm{T\ A} : x$ *and* $\mathrm{F\ A} : y$ *of* $\mathcal{F}(\mathcal{B})$ *are* complementary in $\mathcal{B}$ *if and only if* $x \leq y \in \overline{Ass}(\mathcal{B})$*, i.e., if the constraint* $x \leq y$ *belongs to the reflexive, transitive and compatible closure of* $Ass(\mathcal{B})$*.*

Definition 3.6. *A tableau* t *is* closed *if, for all its branches* $\mathcal{B}$*,*
i) there are two formulas $\mathrm{T\ A} : x$ *and* $\mathrm{F\ A} : y$ *that are complementary in* $\mathcal{B}$ *or there is* $\mathrm{F} \top : x$ *in* $\mathcal{B}$ *or* $\mathrm{F\ I} : x$ *in* $\mathcal{B}$ *with* $1 \leq x \in \overline{Ass}(\mathcal{B})$ *or* $\mathrm{T\ I} : x$ *in* $\mathcal{B}$ *with* $1 \leq x \notin \overline{Ass}(\mathcal{B})$*;*
ii) for all $x \leq y \in Req(\mathcal{B})$, $x \leq y \in \overline{Ass}(\mathcal{B})$*.*
A tableau sequence $\mathcal{T} = t_1, t_2, \ldots, t_n$ *is* closed *if it contains at least one closed tableau.*

Finally, let us mention that a branch can be represented as a flat structure $\langle\, \mathcal{F}(\mathcal{B}), Ass(\mathcal{B}), Req(\mathcal{B}) \,\rangle$ for which the expansion rules are easily adapted. For example, the rule for $\mathrm{T\ A} * \mathrm{B} : x$ becomes the following:

$$
\frac{\langle\, \mathcal{F}(\mathcal{B}) = \{\, ..., \mathrm{T\ A} * \mathrm{B} : x, ... \,\}, Ass(\mathcal{B}), Req(\mathcal{B}) \,\rangle \qquad c_i, c_j \notin \mathcal{D}(\mathcal{B})}{\langle\, \mathcal{F}(\mathcal{B}) \cup \{\, \mathrm{T\ A} : c_i, \mathrm{T\ B} : c_j \,\}, Ass(\mathcal{B}) \cup \{\, c_i c_j \leq x \,\}, Req(\mathcal{B}) \,\rangle}\ \mathrm{T}\ *
$$

Such a representation will prove useful later, when we discuss dependency graphs.

3.2 Examples

Let us illustrate the tableau method with some examples. For a better understanding of how the expansion process works, we have annotated the signed formulas with a check-symbol ($\surd_i$) that indicates the step i at which it has been expanded (or a list of steps if the formula has been expanded more than once). Please note that, in order to build a closed tableau sequence, we expand the formulas which introduce new constants before expanding those which reuse existing labels. We also indicate closed branches by ending them with a cross-symbol ($\times$).

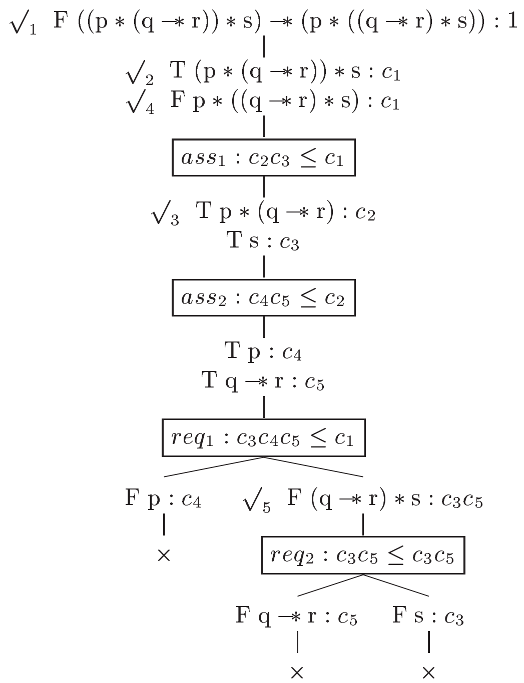


Figure 3: Labelled Tableau for $((\mathrm{p} * (\mathrm{q} \mathbin{-\!\!*} \mathrm{r})) * \mathrm{s}) \mathbin{-\!\!*} (\mathrm{p} * ((\mathrm{q} \mathbin{-\!\!*} \mathrm{r}) * \mathrm{s}))$

Let us consider the tableau of figure 3, we observe that we have complementary formulas in each branch: p with c_4, $\mathrm{q} \mathbin{-\!\!*} \mathrm{r}$ with c_5 and s with c_3. Moreover we have two assertions, namely, $ass_1 : c_2c_3 \leq c_1$ and $ass_2 : c_4c_5 \leq c_2$, and two requirements, namely, $req_1 : c_3c_4c_5 \leq c_1$ and $req_2 : c_3c_5 \leq c_3c_5$ that have to be satisfied from the assertions. These requirements are satisfied and thus all the branches are closed. In fact, the requirement $req_2 : c_3c_5 \leq c_3c_5$ is directly satisfied. From $ass_2 : c_4c_5 \leq c_2$ we can deduce by the compatibility property that we have the assertion $c_3c_4c_5 \leq c_3c_2$ and with $ass_1 : c_2c_3 \leq c_1$ we deduce by transitivity that the requirement $req_1 : c_3c_4c_5 \leq c_1$ is also satisfied.

Let us now consider the non-closed tableau of figure 4. In order to build it, we begin the expansion with the initial multiplicative $\mathbin{-\!\!*}$, producing two new formulas $\mathrm{T}\ (\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r}) : c_1$ and $\mathrm{F}\ \mathrm{p} * (\mathrm{q} \wedge \mathrm{r}) : c_1$. Step 2 deals with the additive $\wedge$, yielding two multiplicatives: $\mathrm{T}\ \mathrm{p} * \mathrm{q} : c_1$ and $\mathrm{T}\ \mathrm{p} * \mathrm{r} : c_1$. The expansion of the first $T\,*$ rule in step 3 introduces two new constants c_2, c_3 and the assertion $ass_1 : c_2c_3 \leq c_1$. The second $*$ is handled in step 4 and introduces two new constants c_4, c_5 together with the assertion $ass_2 : c_4c_5 \leq c_1$.
We are now left with the remaining $\pi\beta$ formula. To expand it, we must match a requirement and thus find a compound label yz such that $yz \leq c_1$. From the assertions in the branch, we deduce that two instantiations are possible: either $y = c_2$ and $z = c_3$, or $y = c_4$ and $z = c_5$. Both instantiations must be tried to achieve a completed tableau. Step 5 therefore goes on with the first one, namely, $req_1 : c_2c_3 \leq c_1$,

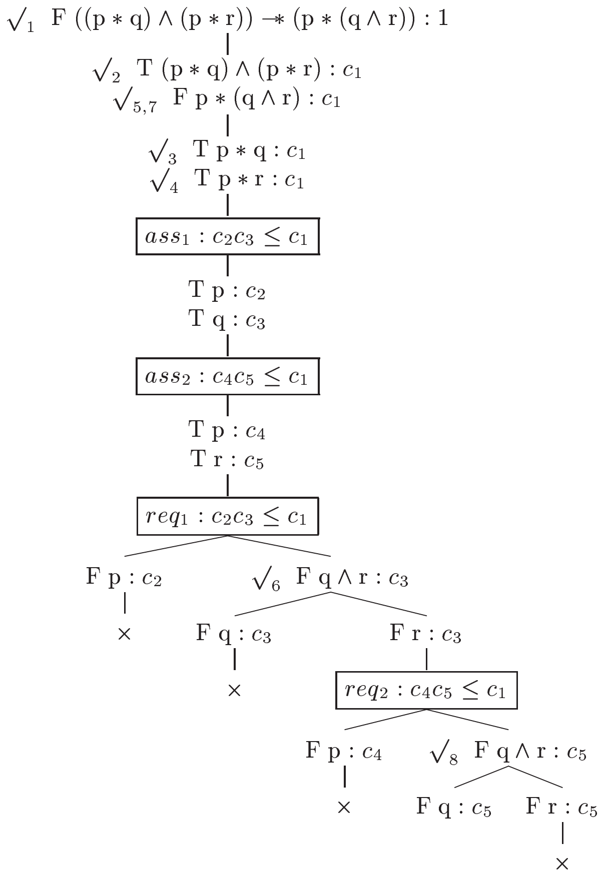


Figure 4: Labelled Tableau for ((p ∗ q) ∧ (p ∗ r)) −∗ (p ∗ (q ∧ r))

splitting the tableau into two new branches. The first branch is immediately closed since it contains both T p : c_2 and F p : c_2 and $c_2 \leq c_2$ is trivially satisfied. The second branch yields an additive conjunction which is immediately expanded in step 6 and we have two new branches. Once again, the first branch is closed since it contains $\langle$ T q : c_3, F q : c_3 $\rangle$. The second branch contains a potential connection $\langle$ T r : c_5, F r : c_3 $\rangle$ but there is no way to show $c_5 \leq c_3$ from the closure of the assertions in the branch.

After step 6, there are no more α or $\pi\alpha$ formulas to expand. However, the tableau is not yet completed (see definition 4.3) because we have not tested the second possibility for the $\pi\beta$ formula, that is, $y = c_4$ and $z = c_5$. Step 7 therefore expands the remaining unclosed branch with this second instantiation and yields a subtree which is similar to the one developed at step 5, except that c_2 is replaced by c_4 and c_3 is replaced by c_5. We obtain two more closed branches, one with $\langle$ T p : c_4, F p : c_4 $\rangle$ and another one with $\langle$ T r : c_5, F r : c_5 $\rangle$. Thus, all formulas have been fulfilled and we notice that the fourth branch of the tree is not closed. Indeed, we have two potential connections $\langle$ T q : c_3, F q : c_5 $\rangle$ and $\langle$ T r : c_5, F r : c_3 $\rangle$, but neither $c_3 \leq c_5$ nor $c_5 \leq c_3$ can be deduced from the closure of the assertions available in the branch.

4 Properties of the TBI$^\perp$ calculus

In this section, we show the soundness and completeness of the **TBI**$^\perp$ calculus w.r.t. the Kripke resource semantics. This deductive system allows not only to build a proof but also a countermodel in case of non-provability.

4.1 Soundness

Definition 4.1. *Let $\mathcal{K} = \langle \mathcal{M}, \models, [\![-]\!] \rangle$ be a EKM and $\mathcal{B}$ be a tableau branch, a* realization of $\mathcal{B}$ in $\mathcal{K}$ *is a mapping $\|-\| : \mathcal{D}(\mathcal{B}) \to M$, from the domain of $\mathcal{B}$ to the worlds of M, that satisfies*

1. $\|1\| = e$,
2. $\|x \circ y\| = \|x\| \bullet \|y\|$,
3. *for any signed formula* $\mathrm{T}\ \mathrm{A} : x$ *in* $\mathcal{B}$, $\|x\| \models \mathrm{A}$,
4. *for any signed formula* $\mathrm{F}\ \mathrm{A} : x$ *in* $\mathcal{B}$, $\|x\| \not\models \mathrm{A}$,
5. *for any assertion* $x \leq y$ *in* $Ass(\mathcal{B})$, $\|x\| \sqsubseteq \|y\|$ *holds in* $\mathcal{K}$.

Let us note that $\|-\|$ is completely defined by its values on the constants of $\mathcal{B}$. Moreover, we have the following lemma, which states that the last condition extends to the closure of the assertions.

Lemma 4.1. *Let t be a tableau, $\mathcal{B}$ be a branch of t and $\|-\|$ be a realization of $\mathcal{B}$ in some EKM $\mathcal{K}$, for any $x \leq y \in \overline{Ass}(\mathcal{B})$, $\|x\| \sqsubseteq \|y\|$ holds in $\mathcal{K}$.*

Proof. Straightforward since $\bullet$ is compatible w.r.t. $\sqsubseteq$ in $\mathcal{K}$ and any assertion $x \leq y$ in $\overline{Ass}(\mathcal{B})$ derives, by compatibility of $\circ$ w.r.t. $\leq$, from assertions of $Ass(\mathcal{B})$ the realization of which holds in $\mathcal{K}$ by definition.

Definition 4.2. *A tableau branch $\mathcal{B}$ is* realizable *if there exists a realization of $\mathcal{B}$ in some EKM. A tableau t is* realizable *if it contains a realizable branch.*

Lemma 4.2. *A closed tableau cannot be realizable.*

Proof. Let t be a closed tableau that is also realizable. It contains a branch $\mathcal{B}$ which is realizable in some *EKM* for some realization $\|-\|$. We now proceed by case analysis on the closure of the branch.

- if the branch is closed due to complementary formulas $\langle \mathrm{T}\ \mathrm{A} : x, \mathrm{F}\ \mathrm{A} : y \rangle$, then, by definition, we have $x \leq y \in \overline{Ass}(\mathcal{B})$ which, by lemma 4.1, implies $\|x\| \sqsubseteq \|y\|$. But, since $\|-\|$ realizes $\mathcal{B}$, we have $\|x\| \models \mathrm{A}$ and $\|y\| \not\models \mathrm{A}$. Then, we have a contradiction because, by property (K), we should have $\|y\| \models \mathrm{A}$.
- other cases are similar.

Lemma 4.3. *If t' is a tableau obtained from a tableau t by application of an expansion rule of figure 2, then, if t is realizable, t' is also realizable.*

Proof. Since t is realizable, it contains a branch $\mathcal{B}$ which is realizable in some *EKM* for some realization $\|-\|$. If the signed formula $\mathrm{Sg}\ \mathrm{X} : x$ that has been expanded to obtain t' does not belong to $\mathcal{B}$, then t' is realizable since it still contains $\mathcal{B}$. Otherwise, we show by case analysis on $\mathrm{Sg}\ \mathrm{X} : x$ that the corresponding expansion rule preserves realizability.

- case $\mathrm{T}\ \mathrm{A} * \mathrm{B} : x$
  $\mathcal{B}$ is expanded into $\mathcal{B}'$ with $\mathcal{F}(\mathcal{B}') = \mathcal{F}(\mathcal{B}) \cup \{ \mathrm{T}\ \mathrm{A} : c_i, \mathrm{T}\ \mathrm{B} : c_j \}$ and $Ass(\mathcal{B}') = Ass(\mathcal{B}) \cup \{ c_i c_j \leq x \}$, c_i and c_j being new constants. Since $\|-\|$ realizes $\mathcal{B}$, we have $\|x\| \models \mathrm{A} * \mathrm{B}$. Therefore, there exist m, $n \in M$ such that $m \bullet n \sqsubseteq \|x\|$, $m \models \mathrm{A}$ and $n \models \mathrm{B}$. We simply extend $\|-\|$ to c_i, c_j by setting $\|c_i\| = m$ and $\|c_j\| = n$. Thus, we obtain $\|c_i\| \models \mathrm{A}$ and $\|c_j\| \models \mathrm{B}$. Since $\|c_i\| \bullet \|c_j\| \sqsubseteq \|x\|$, $\mathcal{B}'$ is realizable and, consequently, t' is realizable.
- case $\mathrm{F}\ \mathrm{A} * \mathrm{B} : x$
  $\mathcal{B}$ splits into $\mathcal{B}'$ and $\mathcal{B}''$ such that $\mathcal{F}(\mathcal{B}') = \mathcal{F}(\mathcal{B}) \cup \{ \mathrm{F}\ \mathrm{A} : y \}$, $\mathcal{F}(\mathcal{B}'') = \mathcal{F}(\mathcal{B}) \cup \{ \mathrm{F}\ \mathrm{B} : z \}$ and $Req(\mathcal{B}')$ $= Req(\mathcal{B}'') = Req(\mathcal{B}) \cup \{ yz \leq x \}$. A valid application of the $\mathrm{F}*$ rule requires that $yz \leq x$ should be in $\overline{Ass}(\mathcal{B})$. Thus, lemma 4.1 implies $\|y\| \bullet \|z\| \sqsubseteq \|x\|$. Since $\|-\|$ realizes $\mathcal{B}$, we have $\|x\| \not\models \mathrm{A} * \mathrm{B}$. Therefore, for any m, $n \in M$ such that $m \bullet n \sqsubseteq \|x\|$, either $m \not\models \mathrm{A}$, or $n \not\models \mathrm{B}$, which implies that either $\|y\| \not\models \mathrm{A}$, or $\|z\| \not\models \mathrm{B}$. Then, either $\mathcal{B}'$, or $\mathcal{B}''$ is realizable and, consequently, t' is realizable.
- other cases are similar.

Corollary 4.1. *Let $t_1, t_2, \ldots$ be a tableau sequence, if t_i is realizable, then, for $j > i$, t_j is realizable.*

Theorem 4.1 (soundness). *Let* A *be a proposition of* $\mathbf{BI}^{\perp}$*, if there exists a closed tableau sequence for* A*, then,* A *is valid in Kripke resource semantics.*

Proof. Let $\mathcal{T} = t_1, \ldots, t_n$ be a closed tableau sequence and suppose that A does not hold in Kripke resource semantics, then, by corollary 2.2, there exists an *EKM* such that $e \not\models \mathrm{A}$. Thus, the initial tableau t_1 is trivially realizable and lemma 4.3 entails that all t_i such that $i > 1$ are also realizable. It follows from lemma 4.2 that none of the t_i can be closed and, consequently, $\mathcal{T}$ cannot be closed, which is a contradiction.

4.2 Completeness and Countermodel Construction

In this subsection we prove the completeness of the tableau system w.r.t. the Kripke resource semantics. For that, we first introduce the notion of completed branch [7] and then show the model existence lemma, *i.e.*, how to construct a Kripke resource model from such a branch. We conclude the section by giving a tableau construction procedure which, for any given formula, produces either a closed tableau for it, or a tableau that contains a completed branch, thereby proving completeness.

Definition 4.3. *Let* $\mathcal{B}$ *be a tableau branch, a signed formula* $\mathrm{Sg}\ \mathrm{X} : x$ *is* fulfilled *(or* completely analyzed*) in* $\mathcal{B}$*, noted* $\mathcal{B} \Vdash \mathrm{Sg}\ \mathrm{X} : x$*, if it satisfies one of the following conditions*

1. $\mathcal{B} \Vdash \mathrm{F}\ \top : x$
2. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{I} : x$ *iff* $1 \leq x \notin \overline{Ass}(\mathcal{B})$
3. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{p} : x$ *iff there exists* $\mathrm{F}\ \mathrm{p} : y \in \mathcal{B}$ *such that* $x \leq y \in \overline{Ass}(\mathcal{B})$
4. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} \wedge \mathrm{B} : x$ *iff either* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : x$ *or* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{B} : x$
5. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} \vee \mathrm{B} : x$ *iff both* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : x$ *and* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{B} : x$
6. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} \to \mathrm{B} : x$ *iff there exists* $y \in \mathcal{D}(\mathcal{B})$ *such that* $x \leq y \in \overline{Ass}(\mathcal{B})$ *and both* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : y$ *and* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{B} : y$
7. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} * \mathrm{B} : x$ *iff for any* $y,\ z \in \mathcal{D}(\mathcal{B})$ *such that* $yz \leq x \in \overline{Ass}(\mathcal{B})$ *either* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : y$*, or* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{B} : z$
8. $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} \mathrel{-\!\!*} \mathrm{B} : x$ *iff there exist* $y,\ xy \in \mathcal{D}(\mathcal{B})$ *such that both* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : y$ *and* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{B} : xy$

9. $\mathcal{B} \Vdash \mathrm{T}\ \top : x$
10. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{I} : x$ *iff* $1 \leq x \in \overline{Ass}(\mathcal{B})$
11. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{p} : x$ *iff there exists* $\mathrm{T}\ \mathrm{p} : y \in \mathcal{B}$ *such that* $y \leq x \in \overline{Ass}(\mathcal{B})$
12. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} \wedge \mathrm{B} : x$ *iff both* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : x$ *and* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{B} : x$
13. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} \vee \mathrm{B} : x$ *iff either* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : x$*, or* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{B} : x$
14. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} \to \mathrm{B} : x$ *iff for any* $y \in \mathcal{D}(\mathcal{B})$ *such that* $x \leq y \in \overline{Ass}(\mathcal{B})$ *either* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : y$*, or* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{B} : y$
15. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} * \mathrm{B} : x$ *iff there exist* $y,\ z \in \mathcal{D}(\mathcal{B})$ *such that* $yz \leq x \in \overline{Ass}(\mathcal{B})$ *and both* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : y$ *and* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{B} : z$
16. $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} \mathrel{-\!\!*} \mathrm{B} : x$ *iff for any* $y,\ xy \in \mathcal{D}(\mathcal{B})$ *either* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : y$*, or* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{B} : xy$

Definition 4.4. *A tableau branch* $\mathcal{B}$ *is* completed *if it is* open *and every signed formula* $\mathrm{Sg}\ \mathrm{A} : x$ *in it is* fulfilled*. A tableau is* completed *if it has a completed branch.*

Lemma 4.4 (monotonicity). *A completed branch* $\mathcal{B}$ *satisfies Kripke monotonicity, more precisely, 1)* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : x$ *and* $y \leq x \in \overline{Ass}(\mathcal{B})$ *imply* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : y$*, and 2)* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : x$ *and* $x \leq y \in \overline{Ass}(\mathcal{B})$ *imply* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : y$*.*

Proof. By structural induction on $\mathrm{Sg}\ \mathrm{X} : x$.

Lemma 4.5 (consistency). *If* $\mathcal{B}$ *is a completed branch, then, for any* **BI** *proposition* A*, we cannot have both* $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{A} : x$ *and* $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{A} : x$*.*

Proof. Let us suppose that both $\mathcal{B} \Vdash \mathrm{T}\ \mathrm{p} : x$ and $\mathcal{B} \Vdash \mathrm{F}\ \mathrm{p} : x$. Then, $\mathrm{T}\ \mathrm{p} : y \in \mathcal{B}$ for some y such that $y \leq x \in \overline{Ass}(\mathcal{B})$ and $\mathrm{F}\ \mathrm{p} : z \in \mathcal{B}$ for some z such that $x \leq z \in \overline{Ass}(\mathcal{B})$. Therefore, by transitivity, we have $y \leq z \in \overline{Ass}(\mathcal{B})$, which implies that $\mathcal{B}$ is closed. It is a contradiction with the definition of a completed branch. The property can be easily extended to non-atomic propositions by structural induction.

4.2.1 Countermodel Construction

Countermodel generation is important in automated reasoning because a countermodel gives information about the non-provability of a formula. In this subsection, we explain how to build a countermodel of A from a completed branch in a labelled tableau of A.

Definition 4.5. *Let $\mathcal{B}$ be a completed branch, the M-structure $\mathcal{M}(\mathcal{B}) = \langle M, \sqsubseteq, \bullet, 1 \rangle$ is defined as follows*
i) M is the set $\mathcal{D}(\mathcal{B})$, extended with a particular element denoted by π
ii) $\bullet$ is a composition law defined by

$$\begin{cases} x \bullet 1 = 1 \bullet x = x & \\ x \bullet y = y \bullet x = xy & \textit{if } xy \in M \\ x \bullet y = y \bullet x = \pi & \textit{otherwise} \end{cases}$$

iii) the relation $\sqsubseteq$ between elements of M is defined by

$$x \sqsubseteq y \textit{ if and only if } y = \pi \textit{ or } x \leq y \in \overline{Ass}(\mathcal{B})$$

Lemma 4.6. *Let $\mathcal{B}$ be a completed branch, $\mathcal{M}(\mathcal{B}) = \langle M, \sqsubseteq, \bullet, 1 \rangle$ is a Kripke resource monoid.*

Proof. By definition, we have identity w.r.t. 1 and commutativity of $\bullet$. It is routine to show that $\bullet$ is also associative. Moreover, we can deduce from the closure operations performed on the assertions that $\bullet$ is compatible w.r.t. $\sqsubseteq$.

Definition 4.6. *Let $\mathcal{M}(\mathcal{B}) = \langle M, \sqsubseteq, \bullet, 1 \rangle$ be the M-structure of a completed branch $\mathcal{B}$ and $\mathcal{P}(L)$ denote the collection of* **BI** *propositions over a language L of propositional letters. We define the interpretation $[\![-]\!]_{\mathcal{B}} : L \to \mathcal{P}(M)$ for any atomic proposition p by $[\![\mathrm{p}]\!]_{\mathcal{B}} = \{ \pi \} \cup \{ x \mid \mathcal{B} \Vdash \mathrm{T\,p} : x \}$.*

Lemma 4.7 (model existence). *If $\mathcal{B}$ is a completed branch, then, the triple $\langle \mathcal{M}(\mathcal{B}), \models, [\![-]\!]_{\mathcal{B}} \rangle$ is a Kripke resource model of $\mathcal{B}$, i.e.,*
1) $[\![-]\!]_{\mathcal{B}}$ satisfies condition (K) of definition 2.4.
2) For any proposition A, we have
  a) $\pi \models \mathrm{A}$,
  b) $\mathcal{B} \Vdash \mathrm{T\,A} : x$ implies $x \models \mathrm{A}$,
  c) $\mathcal{B} \Vdash \mathrm{F\,A} : x$ implies $x \not\models \mathrm{A}$.

Proof. 1) Let us suppose $x \in [\![\mathrm{p}]\!]_{\mathcal{B}}$ and $x \sqsubseteq y$. If $y = \pi$, then, $\pi \in [\![\mathrm{p}]\!]_{\mathcal{B}}$ by definition. If $y \neq \pi$, then, $x \sqsubseteq y$ implies $x \leq y \in \overline{Ass}(\mathcal{B})$ and it follows from lemma 4.4 that $\mathcal{B} \Vdash \mathrm{T\,p} : y$, i.e., $y \in [\![\mathrm{p}]\!]_{\mathcal{B}}$.

2.a) We proceed by structural induction on A.

- if $\mathrm{A} \equiv \mathrm{p}$ then, $\pi \models \mathrm{p}$ because $\pi \in [\![\mathrm{p}]\!]_{\mathcal{B}}$.
- if $\mathrm{A} \equiv \mathrm{B} \wedge \mathrm{C}$ then, by the induction hypothesis, $\pi \models \mathrm{B}$ and $\pi \models \mathrm{C}$. Thus, $\pi \models \mathrm{B} \wedge \mathrm{C}$.
- if $\mathrm{A} \equiv \mathrm{B} \vee \mathrm{C}$ then, by the induction hypothesis, $\pi \models \mathrm{B}$. Thus, $\pi \models \mathrm{B} \vee \mathrm{C}$.
- if $\mathrm{A} \equiv \mathrm{B} \to \mathrm{C}$ then, by the induction hypothesis, $\pi \models \mathrm{C}$. Since π is the only $x \in M$ such that $\pi \sqsubseteq x$, we have $\pi \models \mathrm{B} \to \mathrm{C}$.
- if $\mathrm{A} \equiv \mathrm{B} \multimap\!\!* \, \mathrm{C}$ then, by the induction hypothesis, $\pi \models \mathrm{C}$. Since, for any $x \in M$, $\pi \bullet x = \pi$, we have $\pi \models \mathrm{B} \multimap\!\!* \, \mathrm{C}$.
- if $\mathrm{A} \equiv \mathrm{B} * \mathrm{C}$ then, by the induction hypothesis, $\pi \models \mathrm{B}$ and $\pi \models \mathrm{C}$. Since $\pi \bullet \pi = \pi$, we have $\pi \models \mathrm{B} * \mathrm{C}$.

2.b), 2.c) We proceed by structural induction on $\mathrm{Sg\,A} : x$.

- if $\mathcal{B} \Vdash \mathrm{T\,p} : x$ then, by definition, $x \in [\![\mathrm{p}]\!]_{\mathcal{B}}$, i.e., $x \models \mathrm{p}$.
- if $\mathcal{B} \Vdash \mathrm{F\,p} : x$ then, by lemma 4.5, $\mathcal{B} \not\Vdash \mathrm{T\,p} : x$. Thus, $x \notin [\![\mathrm{p}]\!]_{\mathcal{B}}$, i.e., $x \not\models \mathrm{p}$.
- if $\mathcal{B} \Vdash \mathrm{T\,B} \wedge \mathrm{C} : x$ then, both $\mathcal{B} \Vdash \mathrm{T\,B} : x$ and $\mathcal{B} \Vdash \mathrm{T\,C} : x$. By the induction hypothesis, we have $x \models \mathrm{B}$ and $x \models \mathrm{C}$, i.e., $x \models \mathrm{B} \wedge \mathrm{C}$.

- if $\mathcal{B} \Vdash \mathrm{F\,B} \wedge \mathrm{C} : x$ then, either $\mathcal{B} \Vdash \mathrm{F\,B} : x$ or $\mathcal{B} \Vdash \mathrm{F\,C} : x$. By the induction hypothesis, we have $x \not\models \mathrm{B}$ or $x \not\models \mathrm{C}$, i.e., $x \not\models \mathrm{B} \wedge \mathrm{C}$.
- case $\mathcal{B} \Vdash \mathrm{T\,B} \vee \mathrm{C} : x$ is similar to case $\mathcal{B} \Vdash \mathrm{F\,B} \wedge \mathrm{C} : x$
- case $\mathcal{B} \Vdash \mathrm{F\,B} \vee \mathrm{C} : x$ is similar to case $\mathcal{B} \Vdash \mathrm{T\,B} \wedge \mathrm{C} : x$
- if $\mathcal{B} \Vdash \mathrm{T\,B} \to \mathrm{C} : x$ then, for any y in $\mathcal{D}(\mathcal{B})$ such that $x \leq y \in \overline{Ass}(\mathcal{B})$, either $\mathcal{B} \Vdash \mathrm{F\,B} : y$ or $\mathcal{B} \Vdash \mathrm{T\,C} : y$. By the induction hypothesis, we have either $y \not\models \mathrm{B}$ or $y \models \mathrm{C}$ for any $y \neq \pi$ in M such that $x \sqsubseteq y$. Since, by 2.a), $\pi \models \mathrm{C}$, we have $x \models \mathrm{B} \to \mathrm{C}$.
- if $\mathcal{B} \Vdash \mathrm{F\,B} \to \mathrm{C} : x$ then, there exists y in $\mathcal{D}(\mathcal{B})$ such that $x \leq y \in \overline{Ass}(\mathcal{B})$, $\mathcal{B} \Vdash \mathrm{T\,B} : y$ and $\mathcal{B} \Vdash \mathrm{F\,C} : y$. By the induction hypothesis, there exists $y \in M$ such that $x \sqsubseteq y$, $y \models \mathrm{B}$ and $y \not\models \mathrm{C}$, i.e., $x \not\models \mathrm{B} \to \mathrm{C}$.
- if $\mathcal{B} \Vdash \mathrm{T\,B} \mathrel{-\!\!*} \mathrm{C} : x$ then, let $y \in M$. If $x \bullet y = xy$ then, both y and xy are in $\mathcal{D}(\mathcal{B})$. Therefore, either $\mathcal{B} \Vdash \mathrm{F\,B} : y$, or $\mathcal{B} \Vdash \mathrm{T\,C} : xy$. By the induction hypothesis, we have either $y \not\models \mathrm{B}$, or $x \bullet y \models \mathrm{C}$. If $x \bullet y = \pi$ then, by 2.a), we have $\pi \models \mathrm{C}$. Thus, for any $y \in M$, either $y \not\models \mathrm{B}$ or $x \bullet y \models \mathrm{C}$, i.e., $x \models \mathrm{B} \mathrel{-\!\!*} \mathrm{C}$.
- if $\mathcal{B} \Vdash \mathrm{F\,B} \mathrel{-\!\!*} \mathrm{C} : x$ then, there exist y and xy in $\mathcal{D}(\mathcal{B})$ such that both $\mathcal{B} \Vdash \mathrm{T\,B} : y$ and $\mathcal{B} \Vdash \mathrm{F\,C} : xy$. Since, $xy \in \mathcal{D}(\mathcal{B})$ implies $x \bullet y = xy$, it follows from the induction hypothesis that there exists $y \in M$ such that $y \models \mathrm{B}$ and $xy \not\models \mathrm{C}$, i.e., $x \not\models \mathrm{B} \mathrel{-\!\!*} \mathrm{C}$.
- case $\mathcal{B} \Vdash \mathrm{T\,B} * \mathrm{C} : x$ is similar to case $\mathcal{B} \Vdash \mathrm{F\,B} \mathrel{-\!\!*} \mathrm{C} : x$.
- case $\mathcal{B} \Vdash \mathrm{F\,B} * \mathrm{C} : x$ is similar to case $\mathcal{B} \Vdash \mathrm{T\,B} \mathrel{-\!\!*} \mathrm{C} : x$.

We could argue that an interesting model is a model that is minimal in some sense. The study of model generation for substructural logics is recent and the notion of minimality itself is rather difficult to state for models of such logics.

4.2.2 A Tableau Construction Procedure

In the previous section, we have explained how to build a model from a completed branch. To show the completeness theorem, we now need a tableau construction procedure which, given a formula A, builds a tableau sequence $t_1, t_2, \ldots$ until there exists a tableau t_i that is either closed or that contains a (possibly infinite) completed branch. Giving such a procedure is routine and various strategies can be devised concerning the choice of which formula to expand at each step. The only requirement is that the strategy should be fair, that is, it should not run into a loop by always visiting the same subset of formulas while leaving some others unvisited. Let us give such a procedure.

The initial signed formula is $\mathrm{F\,A} : 1$ and it is marked unvisited.
Repeat until $\mathcal{T}$ is closed or completed
  if there is no unvisited formula $\mathrm{Sg\,X} : x$ in any open branch $\mathcal{B}$
  then mark all $\pi\beta$-formulas in all open branches as unvisited
  else mark $\mathrm{Sg\,X} : x$ as visited
    if $\mathrm{Sg\,X} : x$ is not fulfilled in $\mathcal{B}$
    then if $\mathrm{Sg\,X} : x$ is of type $\pi\beta$
      then expand $\mathrm{Sg\,X} : x$ with a label y for which the formula is not fulfilled
      else expand $\mathrm{Sg\,X} : x$ according to its type

Theorem 4.2 (completeness). *If* A *is a proposition of* $\mathbf{BI}^{\perp}$ *which is valid in Kripke resource semantics, then there exists a closed tableau sequence for* A.

Proof. Suppose there is no closed tableau sequence for A. Then, the above tableau construction procedure yields a tableau in which there is a completed branch $\mathcal{B}$. Since $\mathcal{B}$ contains the initial formula $\mathrm{F\,A} : 1$, lemma 4.7 implies that we can build a model $\langle\, \mathcal{M}(\mathcal{B}), \models, [\![-]\!]_{\mathcal{B}}\,\rangle$ of $\mathcal{B}$ such that $1 \not\models \mathrm{A}$, which means that A is not valid in the Kripke resource semantics.

4.3 Finite Model Property and Decidability

In this subsection, we discuss the finite model property w.r.t. Kripke resource models and, as a consequence, the decidability of $\mathbf{BI}^{\perp}$. For that, we investigate the situations in which a tableau may have infinite branches. We introduce two central notions which are *liberalized expansion rules* and *branch redundancy.* The former provides a way to control the syntactic complexity of the labels by restricting the introduction of new atomic labels during the tableau expansion. The latter characterizes the potential need of an infinite expansion process to achieve a completed branch and can roughly be viewed as a kind of loop-checking.

4.3.1 Liberalized Rules

The rules $\mathrm{F} \rightarrow$, $\mathrm{F} \multimap\!\!*$ and $\mathrm{T} *$ introduce new constants, and then, can lead to infinite branches. Let us consider, for example, the situation depicted in figure 5 for the formula $((\mathrm{q} \rightarrow \mathrm{p}) \rightarrow \mathrm{p}) \rightarrow \mathrm{p}$.

The left-most branch of the tableau infinitely grows because the $\mathrm{F\ q} \rightarrow \mathrm{p}$ formulas repeatedly create a new constant for which the parent formula $\mathrm{T}\ (\mathrm{q} \rightarrow \mathrm{p}) \rightarrow \mathrm{p}$ must be re-expanded in order to achieve a completed branch. But let us suppose that we could reuse at step 5 the constant c_2 introduced at step 3 (instead of creating the new constant c_3), then, step 5 would simply reintroduce $\mathrm{T\ q} : c_2$, $\mathrm{F\ p} : c_2$, which were already introduced at step 3, and also the trivial assertion $c_2 \leq c_2$. Thus, the branch would become completed (and finite) after step 5.

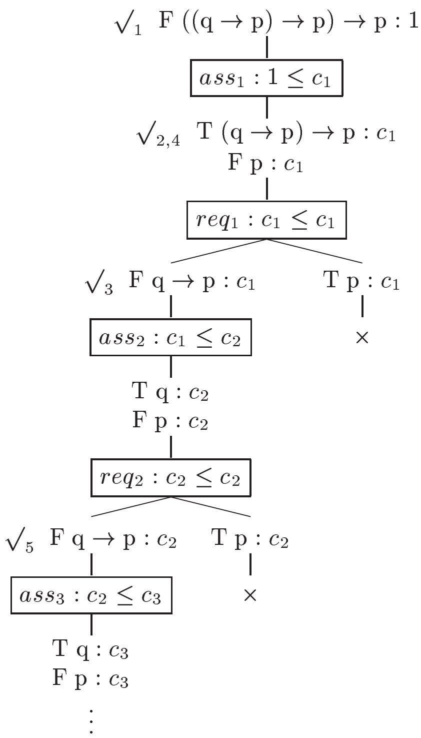


Figure 5: Infinite Tableau

The liberalized rules can be justified by the following regularity property (see [7] and also the sections on IL and MILL for more details): *whenever there exists a world satisfying a proposition* A*, there also exists a world, called* A-characteristic*, which is the least to satisfy* A *w.r.t. the underlying preordering.*
Therefore, we can assume that any new constant introduced by an expansion rule refers to a characteristic world. For example, the first time $\mathrm{F\ A} \multimap\!\!* \mathrm{B} : x$ is expanded into $\mathrm{T\ A} : a$, $\mathrm{F\ B} : xa$, the new constant a is assumed to refer to an A-characteristic world. Consequently, if we later come across the same formula with a distinct label, *e.g.*, $\mathrm{F\ A} \multimap\!\!* \mathrm{B} : y$, we can reuse a and produce $\mathrm{F\ B} : ya$ instead of creating a new constant, for this new constant would also refer to an A-characteristic world, and so, be equivalent to a.
Let us note that $\mathrm{T\ A} : a$ does not need to be reintroduced since it was already introduced by the expansion of $\mathrm{F\ A} \multimap\!\!* \mathrm{B} : x$. Moreover, in the case of $\mathrm{T\ A} * \mathrm{B} : x$, both constants a and b can be reused, and so, the expansion of a subsequent $\mathrm{T\ A} * \mathrm{B} : y$ only produces a new assertion $ab \leq y$ since $\mathrm{T\ A} : a$ and $\mathrm{T\ B} : b$ do not need to be reintroduced.

With the liberalized rules, only *finitely many* distinct constants can be introduced in a tableau branch $\mathcal{B}$, which solves the problem of figure 5 and the corresponding (more general) case of a branch containing formulas of the form $\mathrm{F\ C}_k \rightarrow \mathrm{D}_k : y_k$ $(1 \leq k \leq p)$ in the scope of formulas of the form $\mathrm{T\ A}_i \rightarrow \mathrm{B}_i : x_i$ $(1 \leq i \leq n)$ with, for any i, $x_i \leq y_k \in \overline{Ass}(\mathcal{B})$ for some k.
However, this is not sufficient to prevent infinite branches because, even with a finite number of atomic labels, one can still generate an infinite number of non-atomic labels through composition. In order to ensure finiteness, we need the additional notion of *branch redundancy.*

4.3.2 Branch Redundancy

Since only *finitely many* atomic labels can be introduced during the expansion process, the only way for a tableau to grow infinitely is by endlessly producing new compound labels of increasing length.

The tableau of figure 6 illustrates such a situation. In this tableau, we can see that step 2 expands the $\mathrm{T} \multimap\!\!*$ formula with the only compound label available at the moment in which a occurs, namely, $1a$ $(= a)$. Step 3 then expands the $\mathrm{T} \rightarrow$ formula with the same label a since it is the only label which is greater than a, the requirement $a \leq a$ being trivially satisfied.

But, after step 3, a formula F $-\!\!*$ labelled with a is introduced in the scope of the T $-\!\!*$ formula which is also labelled with a. Therefore, the expansion of the F $-\!\!*$ formula at step 4 now necessarily creates a new compound label ab in which a, the label of the dominating T $-\!\!*$ formula, occurs as a sublabel. Thus, we are obliged to try a new expansion of the T $-\!\!*$ formula with the label ab. This is done in step 5, leading to steps 6 and 7 which are similar to step 3 and 4, except for the substitution of the label a by the compound label ab.

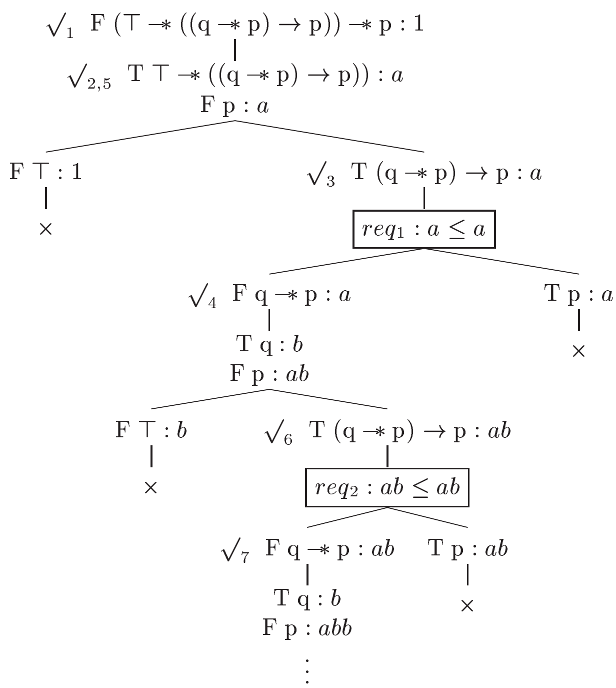


Figure 6: Redundant Tableau

As expected, step 7 creates the compound label abb in which a also occurs as a sublabel and, once again, we are obliged to try an expansion of the T $-\!\!*$ formula with the label abb. Thus, we run into a loop F p : ab, F p : abb, F p : $abbb$, ... which always reproduces the same signed formula F p, but with labels that differ from each other by a single occurrence of the same constant b.

The general situation that corresponds to figure 6 is the case of a branch in which some formulas of the form F C $-\!\!*$ D : y occur in the scope of some formula of the form T A $-\!\!*$ B : x, with x being a sublabel of y. In the example of figure 6, the addition of the extra assertion $bb = b$ solves the problem since all labels ab^i ($i \in \mathbb{N}$) involving b more than once collapse (and become equivalent) to ab, making the branch finite and completed right after step 7. In the general case, the loop may reproduce the same subset of formulas, but with labels differing in more than one, though still a fixed number of occurrences of the same constant. Let us suppose for example that p is replaced in figure 6 by the non-atomic formula $\mathrm{r} \wedge \mathrm{s}$, then, when we reach the first occurence of F $\mathrm{r} \wedge \mathrm{s}$, labelled with ab, we can choose to follow the branch that contains r and, when we later encounter the second occurrence of F $\mathrm{r} \wedge \mathrm{s}$, labelled with abb, we can choose to follow the branch that contains s. Repeating such an alternating process generates the sequences F r : $ab^{(2i+1)}$ and F s : $ab^{(2i+2)}$ ($i \geq 0$), making the repetitions of F r (resp. F s) differ in two occurrences of b from one another.

Anyway, since there are only *finitely many* subformulas of the initial formula to prove, an open branch cannot infinitely grow without reaching a point where any newly introduced signed formula (and its related assertion) has already been introduced up to a fixed number of occurrences of the same constant. Such a branch will be said *redundant*, which is formalized in the following definition.

Definition 4.7 (redundancy). *A completed branch $\mathcal{B}$ is said to be* redundant for *the constant c if there exists $i \geq 1$ and $k > i$ such that for any $j \geq k$*

1 T A : $xc^j \in \mathcal{B}$ *implies* T A : $xc^{j-i} \in \mathcal{B}$ *and*
2 F A : $xc^j \in \mathcal{B}$ *implies* F A : $xc^{j-i} \in \mathcal{B}$ *and*
3 $x \leq yc^j \in Ass(\mathcal{B})$ *implies* $x \leq yc^{j-i} \in Ass(\mathcal{B})$ *and*
4 $xc^j \leq y \in Ass(\mathcal{B})$ *implies* $xc^{j-i} \leq y \in Ass(\mathcal{B})$.

In order to prove the finite model property, we observe that we have the following result for a completed branch $\mathcal{B}$.

Theorem 4.3 (finite model property). *A completed branch $\mathcal{B}$ has a finite Kripke resource model.*

Proof. The proof is by induction on the number of constants for which $\mathcal{B}$ is redundant.
Base case: if there is no constant for which $\mathcal{B}$ is redundant then $\mathcal{B}$ is finite and by lemma 4.7 a Kripke

resource model of $\mathcal{B}$ which is, by construction, obviously finite.
Induction case: we assume that the proposition holds for any completed branch which is redundant for less than n constants. Let us suppose that $\mathcal{B}$ is redundant for n constants. We select one of those constants and denote it c. Then, we add the extra assertions $c^j = c^{j-i}$ for any $j \geq k$, where i, j and k refer to definition 4.7. It is routine to show that the composition modulo rewriting $c^j = c^{j-i}$ preserves the properties of associativity, commutativity, and identity w.r.t. 1. Moreover, the properties 3 and 4 of definition 4.7 ensure that the branch $\mathcal{B}$ remains non contradictory. Since now, we cannot have more than k occurrences of c in the labels, we have treated the redundancy of $\mathcal{B}$ w.r.t. the constant c and we apply the induction hypothesis to obtain the result.

The finite model property w.r.t. Kripke resource models is a new result for $\mathbf{BI}^{\perp}$ which has significant consequences for its further applications, a major one being the decidability of $\mathbf{BI}^{\perp}$. Indeed, the tableau construction procedure given in section 4.2.2, which is a semi-decision procedure, can be improved into a decision procedure by taking both the liberalized versions of the expansion rules and the notion of redundancy into account. Since, under the liberalized rules, an open branch cannot infinitely grow without becoming redundant, the termination of the procedure can be enforced by stopping the (potentially infinite) construction of a completed branch as soon as it has been recognized redundant.

5 Dependency Graphs and Provability

As explained in the previous sections, we have to deal, during the tableau construction, with two kinds of rules, namely $\pi\alpha$ and $\pi\beta$. The first introduces new constants and forces, via so-called assertions, the relationships between these new constants. The second forces the introduction of labels from already existing constants. Moreover, these labels cannot be freely chosen: they must satisfy constraints called requirements, which have to be derivable from the existing assertions. In this section, we explain how the assertions can be represented through dependency graphs and how such structures can be used to determine provability.

5.1 Dependency Graphs

We give here a procedure that builds, in parallel with tableau expansions, a dependency graph which reflects the information that can be derived from a given set of assertions.

Definition 5.1. *Given a branch $\mathcal{B}$, the associated dependency graph $DG(\mathcal{B}) = [N(\mathcal{B}), A(\mathcal{B})]$ is defined as the following directed graph: the set of nodes $N(\mathcal{B})$ is the set of labels $\mathcal{D}(\mathcal{B})$ and the set of arrows $A(\mathcal{B})$ is built from the set of assertions $\overline{Ass}(\mathcal{B})$ as follows: there is an arrow $x \twoheadrightarrow y$ in $A(\mathcal{B})$ iff there is an assertion $x \leq y$ in $\overline{Ass}(\mathcal{B})$.*

In order to simplify notations and because the graphs we consider are assumed to be reflexive and transitive, we do not explicitly mention, in $A(\mathcal{B})$, arrows of the form $x \twoheadrightarrow x$, or of the form $x \twoheadrightarrow y$ when there already exist $x \twoheadrightarrow z$ and $z \twoheadrightarrow y$ in $A(\mathcal{B})$.
Moreover, $Pred(\mathcal{B}, x)$ will denote the set of *immediate* predecessors of a node x in the graph $DG(\mathcal{B})$ while $Pred^{+}(\mathcal{B}, x)$ will denote the set of *all* (not only immediate) predecessors. Let us remind that x is a sublabel of y (notation: $x \preceq y$) if there exists a label z such that $y = xz$. $\mathcal{P}(x)$ denotes the set of all sublabels of a label x, $y(x)$ means a label y having x as a sublabel and $y[x/z]$ (or $y[z]$ when clear from the context) means the substitution of x by z in y.
We now define a procedure that builds the dependency graph $DG(\mathcal{B})$ of a branch $\mathcal{B}$. From that we can derive a procedure to compute the closure $\overline{Ass}(\mathcal{B})$. Therefore, we connect to each tableau expansion rule of figure 2 a corresponding graph expansion rule. Thus, we can define expansion rules that deal with pairs $\langle \mathcal{B}, DG(\mathcal{B}) \rangle$ where $\mathcal{B}$ is a branch and $DG(\mathcal{B})$ the corresponding dependency graph.
The initial configuration is $\langle \mathcal{B}_0, DG(\mathcal{B}_0) \rangle$ defined as follows:

$$\langle \mathcal{F}(\mathcal{B}_0) = \{ \mathrm{F}\ \mathrm{A} : 1 \}, Ass(\mathcal{B}_0) = \{ 1 \leq 1 \}, Req(\mathcal{B}_0) = \emptyset \rangle\ ,\ [N(\mathcal{B}_0) = \{ 1 \}, A(\mathcal{B}_0) = \emptyset]$$

The expansion rules of a dependency graph are such that the given graph is only expanded by the $\pi\alpha$ rules, all the other rules, introducing neither new constants, nor new assertions, simply leave the dependency graph unchanged. Let us give now these three expansion rules.

For the rule T $*$, we have

$$\frac{\langle\, \mathcal{F}(\mathcal{B})(\mathrm{T\ A} * \mathrm{B} : x), Ass(\mathcal{B}), Req(\mathcal{B})\,\rangle \;\;,\;\; [N(\mathcal{B}), A(\mathcal{B})]}{\langle\, \mathcal{F}(\mathcal{B}'), Ass(\mathcal{B}'), Req(\mathcal{B}')\,\rangle \;\;,\;\; [N(\mathcal{B}'), A(\mathcal{B}')]}\ \mathrm{T}\ *$$

$\mathcal{F}(\mathcal{B}') := \mathcal{F}(\mathcal{B}) \cup \{\, \mathrm{T\ A} : c_i, \mathrm{T\ B} : c_j \,\}$
$Ass(\mathcal{B}') := Ass(\mathcal{B}) \cup \{\, c_i \leq c_i, c_j \leq c_j, c_ic_j \leq x \,\}$
$Req(\mathcal{B}') := Req(\mathcal{B})$

$N(\mathcal{B}') := N(\mathcal{B})$; $A(\mathcal{B}') := A(\mathcal{B})$
for all $y(x)$ in $N(\mathcal{B})$ do
$N(\mathcal{B}') := N(\mathcal{B}') \cup \{\, y[c_i], y[c_j], y[c_ic_j] \,\}$
$A(\mathcal{B}') := A(\mathcal{B}') \cup \{\, y[c_ic_j] \twoheadrightarrow y[x] \,\}$
for all $z(x)$ in $Pred(\mathcal{B}, y(x))$ do
$A(\mathcal{B}') := A(\mathcal{B}') \cup \{\, z[c_i] \twoheadrightarrow y[c_i], z[c_j] \twoheadrightarrow y[c_j] \,\}$
$A(\mathcal{B}') := A(\mathcal{B}') \cup \{\, z[c_ic_j] \twoheadrightarrow y[c_ic_j] \,\}$ done
done

For the rule F $\to$, we have

$$\frac{\langle\, \mathcal{F}(\mathcal{B})(\mathrm{F\ A} \to \mathrm{B} : x), Ass(\mathcal{B}), Req(\mathcal{B})\,\rangle \;\;,\;\; [N(\mathcal{B}), A(\mathcal{B})]}{\langle\, \mathcal{F}(\mathcal{B}'), Ass(\mathcal{B}'), Req(\mathcal{B}')\,\rangle \;\;,\;\; [N(\mathcal{B}'), A(\mathcal{B}')]}\ \mathrm{F} \to$$

$\mathcal{F}(\mathcal{B}') := \mathcal{F}(\mathcal{B}) \cup \{\, \mathrm{T\ A} : c_i, \mathrm{F\ B} : c_i \,\}$
$Ass(\mathcal{B}') := Ass(\mathcal{B}) \cup \{\, c_i \leq c_i, x \leq c_i \,\}$
$Req(\mathcal{B}') := Req(\mathcal{B})$

$N(\mathcal{B}') := N(\mathcal{B}) \cup \{\, c_i \,\}$
$A(\mathcal{B}') := A(\mathcal{B}) \cup \{\, x \twoheadrightarrow c_i \,\}$

For the rule F $-\!\!*$, we have

$$\frac{\langle\, \mathcal{F}(\mathcal{B})(\mathrm{F\ A} -\!\!* \mathrm{B} : x), Ass(\mathcal{B}), Req(\mathcal{B})\,\rangle \;\;,\;\; [N(\mathcal{B}), A(\mathcal{B})]}{\langle\, \mathcal{F}(\mathcal{B}'), Ass(\mathcal{B}'), Req(\mathcal{B}')\,\rangle \;\;,\;\; [N(\mathcal{B}'), A(\mathcal{B}')]}\ \mathrm{F} -\!\!*$$

$\mathcal{F}(\mathcal{B}') := \mathcal{F}(\mathcal{B}) \cup \{\, \mathrm{T\ A} : c_i, \mathrm{F\ B} : xc_i \,\}$
$Ass(\mathcal{B}') := Ass(\mathcal{B}) \cup \{\, c_i \leq c_i \,\}$
$Req(\mathcal{B}') := Req(\mathcal{B})$

$x^l := \{\, x \,\} \cup Pred^+(\mathcal{B}, x)$
$N(\mathcal{B}') := N(\mathcal{B}) \cup \mathcal{P}(x^l)c_i$
$A(\mathcal{B}') := A(\mathcal{B})$
for all y in $\mathcal{P}(x^l)$ do
for all z in $Pred(\mathcal{B}, y)$ do
$A(\mathcal{B}') := A(\mathcal{B}') \cup \{\, zc_i \twoheadrightarrow yc_i \,\}$
done
done

Theorem 5.1. *Let $\mathcal{B}$ be a branch and $DG(\mathcal{B})$ be its associated dependency graph, the application of an expansion rule to $\langle\, \mathcal{B}, DG(\mathcal{B})\,\rangle$ provides a new branch $\mathcal{B}'$ and a new graph DG' that is the dependency graph $DG(\mathcal{B}')$ of $\mathcal{B}'$.*

Proof. In each case, we verify that the construction of $DG(\mathcal{B}')$ from $DG(\mathcal{B})$ by an expansion rule, leads to a graph that is a dependency graph associated to the branch $\mathcal{B}'$.

5.2 Constraints and Provability

After having defined the dependency graph $DG(\mathcal{B})$ associated to a given branch $\mathcal{B}$, we observe that the problem of checking whether a requirement $x \leq y$ of $Req(\mathcal{B})$ holds with respect to the set $\overline{Ass}(\mathcal{B})$, can be directly solved on the corresponding dependency graph. It corresponds to checking if there exists a path in the graph $DG(\mathcal{B})$ from the node x to the node y.

We illustrate the use of the dependency graph as a central semantical structure for the analysis of provability in $\mathbf{BI}^{\perp}$. Thus, we consider the formula $((\mathrm{p} * (\mathrm{q} -\!\!* \mathrm{r})) * \mathrm{s}) -\!\!* (\mathrm{p} * ((\mathrm{q} -\!\!* \mathrm{r}) * \mathrm{s}))$ the tableau expansion of which is given on figure 3 and focus on the parallel construction of the associated dependency graph. Figure 7 presents the final dependency graph, with nodes $i : c_j$ where i indicates which expansion step has introduced the node. Initially, we begin with the dependency graph $DG(\mathcal{B}_0)$ with $N(\mathcal{B}_0) = \{\, 1 \,\}$ and $A(\mathcal{B}_0) = \emptyset$. Step 1 decomposes the outermost $-\!\!*$ and, since $c_1 1 = c_1$, the resulting graph is $DG(\mathcal{B}_1)$

where $N(\mathcal{B}_1) = \{\, 1, c_1 \,\}$ and $A(\mathcal{B}_1) = \emptyset$. The expansion of $\mathrm{T}\ (\mathrm{p} * (\mathrm{q} \mathrel{-\!\!*} \mathrm{r})) * \mathrm{s} : c_1$ in step 2 introduces two new constants c_2, c_3 and the new assertion $ass_1 : c_2c_3 \leq c_1$. The rule for $\mathrm{T}\,*$ yields the graph $DG(\mathcal{B}_2)$ with $N(\mathcal{B}_2) = N(\mathcal{B}_1) \cup \{\, c_2, c_3, c_2c_3 \,\}$ and $A(\mathcal{B}_2) = A(\mathcal{B}_1) \cup \{\, c_2c_3 \twoheadrightarrow c_1 \,\}$. Step 3 expands the signed formula $\mathrm{T}\ \mathrm{p} * (\mathrm{q} \mathrel{-\!\!*} \mathrm{r}) : c_2$ which introduces two new constants c_4, c_5 and the new assertion $ass_2 : c_4c_5 \leq c_2$. The dependency graph is $DG(\mathcal{B}_3)$ with $N(\mathcal{B}_3) = N(\mathcal{B}_2) \cup \{\, c_4, c_5, c_4c_5 \,\} \cup \{\, c_3c_4c_5, c_3c_4, c_3c_5 \,\}$ and $A(\mathcal{B}_3) = A(\mathcal{B}_2) \cup \{\, c_4c_5 \twoheadrightarrow c_2 \,\} \cup \{\, c_3c_4c_5 \twoheadrightarrow c_2c_3 \,\}$ [2]. We add the arrow $c_4c_5 \twoheadrightarrow c_2$, but since we already have a node c_2c_3, we must create a node $c_2c_3[c_2/c_4c_5] = c_3c_4c_5$ and a new arrow $c_3c_4c_5 \twoheadrightarrow c_2c_3$. Moreover, as we add the node $c_3c_4c_5$, we must take care to add all nodes corresponding to the sublabels of $c_3c_4c_5$ that does not already appear in the graph. It provides two more nodes, namely, c_3c_4 and c_3c_5.

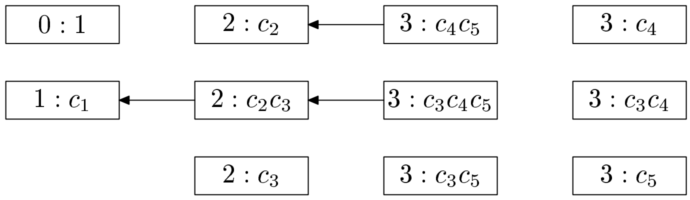


Figure 7: Dependency Graph for $((\mathrm{p} * (\mathrm{q} \mathrel{-\!\!*} \mathrm{r})) * \mathrm{s}) \mathrel{-\!\!*} (\mathrm{p} * ((\mathrm{q} \mathrel{-\!\!*} \mathrm{r}) * \mathrm{s}))$

With step 4, we reach the point where we need to expand the $\pi\beta$ signed formula $\mathrm{F}\ \mathrm{p} * ((\mathrm{q} \mathrel{-\!\!*} \mathrm{r}) * \mathrm{s}) : c_1$. We must therefore find a compound label xy such that $xy \leq c_1$. But, using the dependency graph, this simply means that xy should be taken among the nodes that are ancestors of the node c_1. Thus, we only have two possibilities, namely, c_2c_3 and $c_3c_4c_5$. Only the latter contains c_4, which is needed to close the first branch on p, so that $x = c_4$ and $y = c_3c_5$. The same kind of reasoning is applied at step 5 where we must find a compound label xy such that $xy \leq c_3c_5$. But this time, as the node c_3c_5 has no ancestors except itself (remind that the graph is implicitly reflexive), the only possible value for xy is c_3c_5. However, the choice of the branch on which to spread c_3 still remains. Here, looking upward in the branch suggests that spreading c_3 in the right branch allows us to close it with s and, at the same time, to close the left branch with $\mathrm{q} \mathrel{-\!\!*} \mathrm{r}$.

5.3 Countermodel Extraction

We explain here how to extract a countermodel from a dependency graph when the formula is not provable. In fact, we observe that the dependency graph could be directly seen as (a representation of) a countermodel. But let us illustrate this point with the formula $((\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})) \mathrel{-\!\!*} (\mathrm{p} * (\mathrm{q} \wedge \mathrm{r}))$.

Figure 8 shows its dependency graph. Compared with our previous example, steps 1 to 4 are quite similar and lead to the dependency graph $DG(\mathcal{B}_4)$ whose nodes are $N(\mathcal{B}_4) = \{\, 1, c_1, c_2, c_3, c_4, c_5, c_2c_3, c_4c_5 \,\}$ and arrows are $A(\mathcal{B}_4) = \{\, c_2c_3 \twoheadrightarrow c_1, c_4c_5 \twoheadrightarrow c_1 \,\}$. Once again, when we must expand the $\pi\beta$ formula, we have to find a compound label yz such that $yz \leq c_1$. From the dependency graph, we see that there are two possibilities, namely, c_2c_3 and c_4c_5. We first try an expansion with the former one. Since the right branch is not closed, we go through step 6. After step 6, all $\pi\alpha$ formulas have been treated and the right-most branch is still open. Then, we must try the second possibility, c_4c_5, for the $\pi\beta$ formula (step 7). Since the right-most branch is still open, we continue with step 8, by developing the additive conjunct. After that, the right-most remains open, but this time, we can say from the dependency graph that all possibilities have been tried for the $\pi\beta$ formula. Therefore, the tableau expansion is completed and contains an open branch, which means that the formula is unprovable.

Since all formulas in the open branch are fulfilled, we have a finite completed branch. We can therefore extract the corresponding countermodel following the different steps of the completeness proof.
For that, we take the nodes of the dependency graph, to which we add the particular element π, as the set of worlds M. The preordering relation $\sqsubseteq$ is simply induced by the arrows of the dependency graph, which gives $c_2c_3 \sqsubseteq c_1$, $c_4c_5 \sqsubseteq c_1$ and $m \sqsubseteq \pi$ for any $m \in M$.

[2] We have written the resulting graph using two unions in order to distinguish what directly comes from the tableau rule and what comes from the closure.

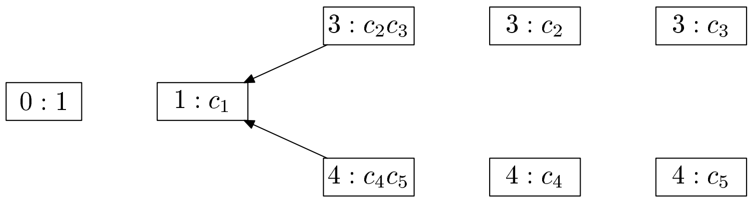


Figure 8: Dependency Graph for $((\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})) \mathrel{-\!\!*} (\mathrm{p} * (\mathrm{q} \wedge \mathrm{r}))$

The $\bullet$ operation is defined by the following table:

$\bullet$	1	c_1	c_2	c_3	c_4	c_5	c_2c_3	c_4c_5	π
1	1	c_1	c_2	c_3	c_4	c_5	c_2c_3	c_4c_5	π
c_1	c_1	π	π	π	π	π	π	π	π
c_2	c_2	π	π	c_2c_3	π	π	π	π	π
c_3	c_3	π	c_2c_3	π	π	π	π	π	π
c_4	c_4	π	π	π	π	c_4c_5	π	π	π
c_5	c_5	π	π	π	c_4c_5	π	π	π	π
c_2c_3	c_2c_3	π	π	π	π	π	π	π	π
c_4c_5	c_4c_5	π	π	π	π	π	π	π	π
π	π	π	π	π	π	π	π	π	π

The forcing relation $\models$ is parametrized by the following atomic interpretation:

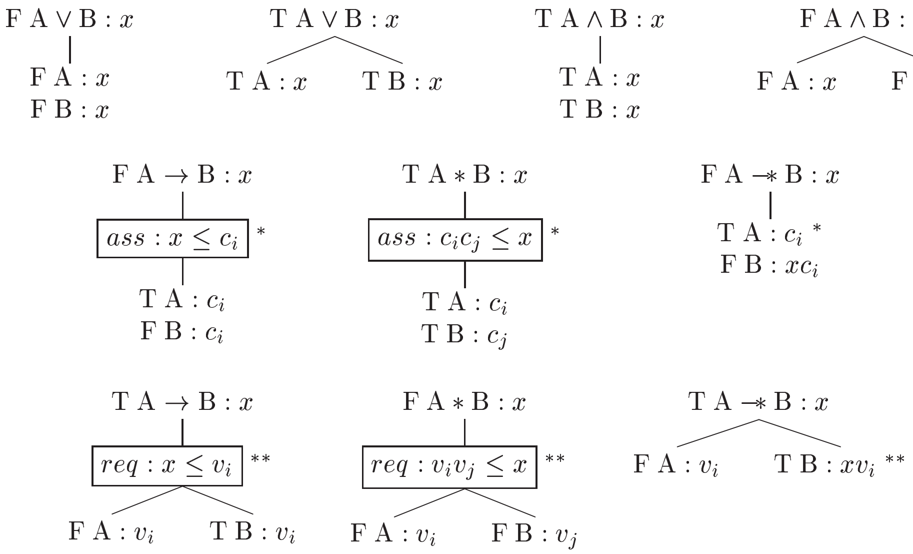
x	p	q	r
$[\![\mathrm{x}]\!]$	$\{\pi, c_2, c_4\}$	$\{\pi, c_3\}$	$\{\pi, c_5\}$

Now, we verify that we actually built a countermodel by checking that, on one hand, $c_1 \models (\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})$ and, on the other hand, $c_1 \not\models \mathrm{p} * (\mathrm{q} \wedge \mathrm{r})$. First, we have $c_2 \bullet c_3 \models \mathrm{p} * \mathrm{q}$ because, by definition, $c_2 \models \mathrm{p}$ and $c_3 \models \mathrm{q}$. Since $c_2 \bullet c_3 \sqsubseteq c_1$, it follows from (K) that $c_1 \models (\mathrm{p} * \mathrm{q})$. Similarly, we can prove $c_1 \models (\mathrm{p} * \mathrm{r})$. Thus, $c_1 \models (\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})$. Second, we observe that no world other than π satisfies q and r at the same time. Thus, π is the only world satisfying $\mathrm{q} \wedge \mathrm{r}$. Since c_1 cannot be obtained by any composition with π, $c_1 \not\models \mathrm{p} * (\mathrm{q} \wedge \mathrm{r})$.

The dependency graph, that we could also call Kripke resource graph (because of the direct relationship with the Kripke resource semantics), is a central concept in this labelled tableau method. It captures the semantics and the interactions between additive and multiplicative connectives and allows to check if the initial formula is provable or not. It is important to notice that we aim to generate countermodels that could be helpful for the analysis of the proof failure. In addition to an efficient generation, we need also to have a good representation of countermodels and, in this context, a dependency graph can be directly seen as the essential graphical representation of a countermodel, from which we can generate a countermodel in the given semantics. Moreover, the question of the minimality of a countermodel could be studied by from refinements of this semantic graph structure.

6 A Free Variable Labelled Tableau for BI$^\perp$

In the **TBI**$^\perp$ calculus we have two kinds of constrained rules. The F $\mathrel{-\!\!*}$, F $\rightarrow$ and T $*$ rules create new (atomic) labels and new assertions which are used as hypotheses in the proof-search process. On the contrary, the T $\mathrel{-\!\!*}$, T $\rightarrow$ and F $*$ rules introduce labels which are supposed to satisfy a specific requirement w.r.t. the assertions. Therefore, making clever choices for these labels is a critical issue for the efficiency of the tableau method. This difficulty can be partially overcome by stating free variable versions of these rules which allow to postpone the choice of suitable labels for a given requirement until enough information is available [7].



* c_i, c_j are new constants
** v_i, v_j are new variables

Figure 9: Expansion Rules with Free Variables

6.1 A Free Variable Calculus

We enrich the labelling language with a denumerable set of variables v_1,v_2,...,v_n. All the previous notions (labelling algebra, labelled formulas, expansion set, configurations) are extended to labels with variables.

Definition 6.1. *A* label schema α *is a label that contains variables. A* ground label *is a label that only contains constants. We note* $\mathcal{L}^\dagger$ *the restriction of* $\mathcal{L}$ *to ground labels.*

We extend the expansion rules to label schemas, then, in addition to the set $C(\mathcal{B})$, we consider the set $V(\mathcal{B})$ of the variables occurring in a branch $\mathcal{B}$.

Definition 6.2. *Let* A *be a proposition of* $\mathbf{BI}^\perp$*, a* tableau sequence for A *is a sequence of tableaux* $t_1, t_2, \ldots, t_n$ *for which the initial one is* $t_1 = \{\langle \{\mathrm{F\,A} : 1\}, \{1 \leq 1\}, \emptyset \rangle\}$ *and* t_{i+1} *is obtained from* t_i *by the application of an expansion rule of figure 9 (on a branch of* t_i*).*

Definition 6.3. *Let* V *be a set of variables, a* ground substitution σ *is an application from* $V \to \mathcal{L}^\dagger$*, where* $\mathcal{L}^\dagger$ *is the set of ground labels; a* ground substitution σ for a branch $\mathcal{B}$ *is a ground substitution the domain of which is* $V(\mathcal{B})$*.*

Definition 6.4. *Let* $\mathcal{B}$ *be a tableau branch, two signed formulas* T A : x, F A : y *of* $\mathcal{F}(\mathcal{B})$ *are said* σ-complementary in $\mathcal{B}$ *if and only if* $\sigma(x) \leq \sigma(y) \in \overline{\sigma(Ass(\mathcal{B}))}$*, i.e., if the ground constraint* $\sigma(x) \leq \sigma(y)$ *belongs to the reflexive, transitive and compatible closure of the ground assertions* $\sigma(Ass(\mathcal{B}))$*.*

Definition 6.5. *A tableau* t *is* σ-closed *if there exists a ground substitution* σ *whose domain is* $\bigcup_{\mathcal{B}\in t} V(\mathcal{B})$ *and such that for any* $\mathcal{B}$ *in* t*,*
i) there are two formulas T A : x *and* F A : y *that are* σ*-complementary in* $\mathcal{B}$ *or there is* F $\top$: x *in* $\mathcal{B}$ *or there is* F I : x *in* $\mathcal{B}$ *with* $1 \leq \sigma(x) \in \overline{\sigma(Ass(\mathcal{B}))}$*, or there is* T I : x *in* $\mathcal{B}$ *with* $1 \leq \sigma(x) \notin \overline{\sigma(Ass(\mathcal{B}))}$*;*
ii) for all $x \leq y \in Req(\mathcal{B})$ $\sigma(x) \leq \sigma(y) \in \overline{\sigma(Ass(\mathcal{B}))}$*.*

Definition 6.6. *A tableau sequence* $\mathcal{T} = t_1, t_2, \ldots$ *is* σ-closed *if it contains at least one* σ*-closed tableau* t_i *and* σ *is such that any variable* v *is mapped to a ground label which is built using exclusively constants that were introduced before* v *in the sequence* $t_1, \ldots, t_i$*.*

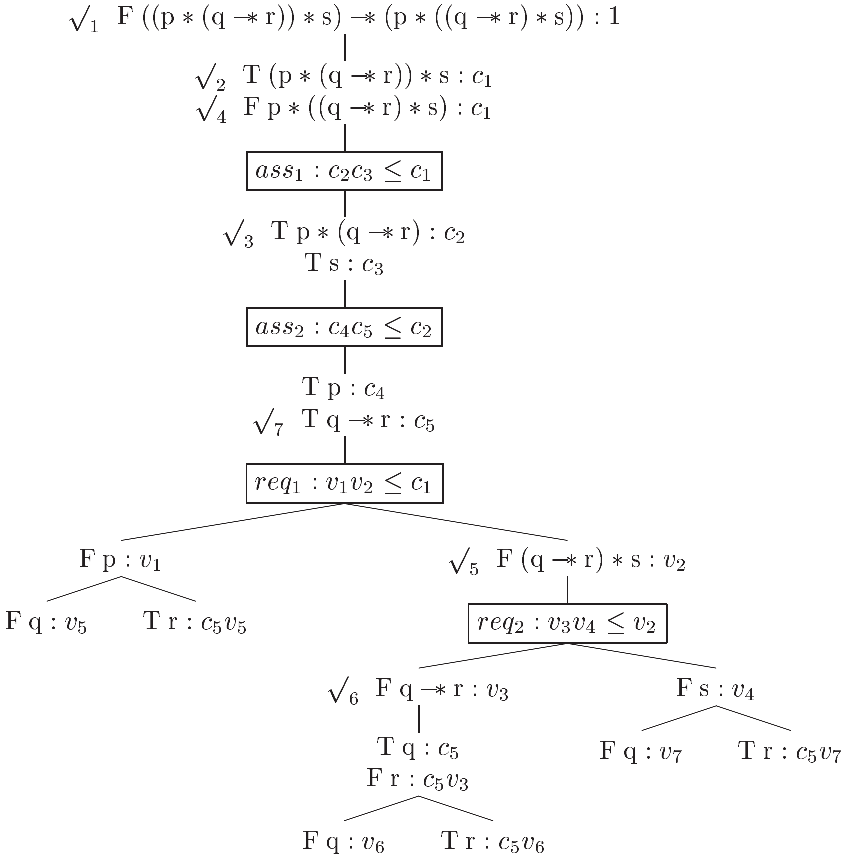


Figure 10: Free Variable Tableau for $(\mathrm{p} * (\mathrm{q} -\!\!* \mathrm{r})) * \mathrm{s}) -\!\!* (\mathrm{p} * ((\mathrm{q} -\!\!* \mathrm{r}) * \mathrm{s})$

From these definitions, we can directly prove, like in section 4, the properties of *soundness*, *completeness* and *finite model property* of a free variable version of $\mathbf{TBI}^{\perp}$.

6.2 Examples of free variable tableaux

Let us consider the formula $((\mathrm{p}*(\mathrm{q}-\!\!*\mathrm{r}))*\mathrm{s})-\!\!*(\mathrm{p}*((\mathrm{q}-\!\!*\mathrm{r})*\mathrm{s}))$, on figure 10, using our free variable labelled tableau system. Compared to the initial system, tableau expansions go through without change until step 3. The key difference appears at step 4, when decomposing the $\pi\beta$ formula $\mathrm{F}\ \mathrm{p} * ((\mathrm{q} -\!\!* \mathrm{r}) * \mathrm{s}) : c_1$. Previously, we had to find a compound label yz such that $yz \leq c_1$. Now, we simply introduce two new variables v_1, v_2, together with a new requirement $req_1 : v_1 v_2 \leq c_1$ expressing a constraint which, once ground, they are supposed to match if we aim a sound expansion step. We proceed with step 5, which is just similar, introducing new variables v_3, v_4 and the corresponding requirement $req_2 : v_3 v_4 \leq v_2$. Since the formula $\mathrm{F}\ \mathrm{q} -\!\!* \mathrm{r} : v_3$ introduced in the last step is a $\pi\alpha$ formula, it is immediately expanded in step 6. Finally, step 7 develops the $\pi\beta$ formula $\mathrm{T}\ \mathrm{q} -\!\!* \mathrm{r} : c_5$, introducing by the way three new variables (one for each extended branch), namely, v_5, v_6 and v_7. After step 7, the tableau is completed.

The problem is now to find a ground substitution that allows us to close each branch of the tableau. We only have one potential connection, namely, $\langle \mathrm{T}\ \mathrm{p} : c_4, \mathrm{F}\ \mathrm{p} : v_1 \rangle$, to close the first and the second branch at the same time. Setting $v_1 \mapsto c_4$ does the job, making v_5 useless by the way. For the third branch, the only possible connection is $\langle \mathrm{T}\ \mathrm{q} : c_5, \mathrm{F}\ \mathrm{q} : v_6 \rangle$, giving $v_6 \mapsto c_5$. Considering $\langle \mathrm{T}\ \mathrm{r} : c_5 v_6, \mathrm{F}\ \mathrm{r} : c_5 v_3 \rangle$ on the fourth branch suggests $v_3 \mapsto v_6$. The last two branches, once again, contain only one potential connection, namely, $\langle \mathrm{T}\ \mathrm{s} : c_3, \mathrm{F}\ \mathrm{s} : v_4 \rangle$, which yields $v_4 \mapsto c_3$ and makes v_7 useless. We therefore have found a partial (v_2 is yet undefined) ground substitution that closes each branch of the tableau. In order to verify that this substitution is an admissible solution, we must also check, branch by branch, whether the requirements can be satisfied from the assertions. Beginning with $req_2 : v_3v_4 \leq v_2$ will allow us to fix a value for v_2. Indeed, simply choosing $v_2 \mapsto c_3c_5$ makes the requirement trivially satisfied. It remains to ensure that $req_1 : v_1v_2 \leq c_1$ also holds. More precisely, what we need to show is that $req_1 : c_3c_4c_5 \leq c_1$ can be derived from the reflexive, transitive and compatible closure of the assertions $ass_1 : c_2c_3 \leq c_1$ and $ass_2 : c_4c_5 \leq c_2$. But this requirement is simply obtained by first applying compatibility on ass_2 and $c_3 \leq c_3$, which yields $c_3c_4c_5 \leq c_2c_3$, and then using transitivity on ass_1. Thus, our ground substitution closes each branch of the tableau and, moreover, satisfies all requirements. The conclusion is therefore that the formula is provable.

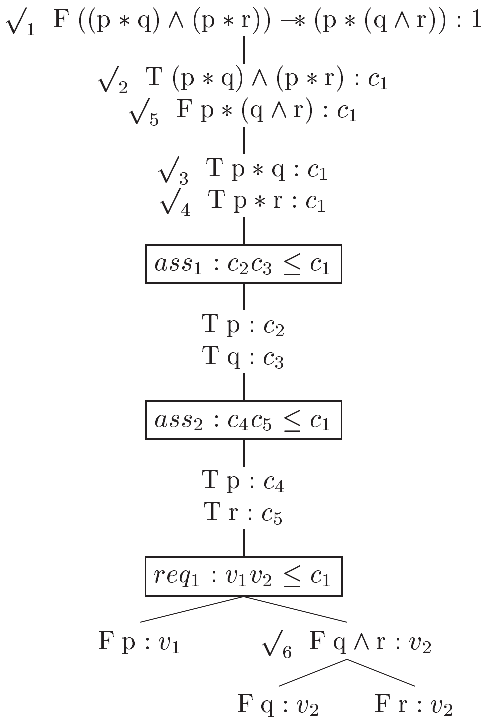
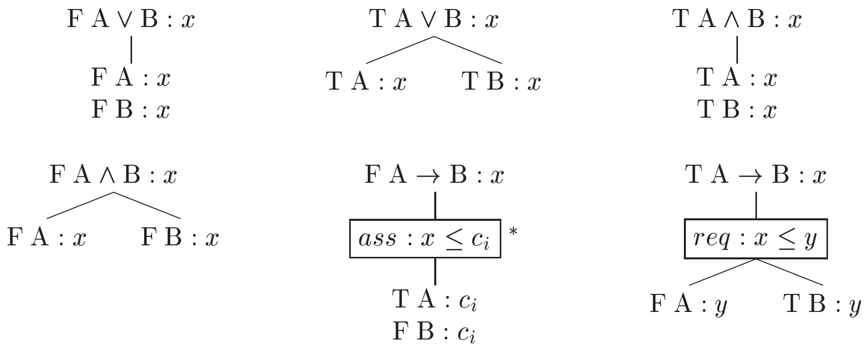


Figure 11: Free variable tableau for $(\mathrm{p} * \mathrm{q}) \wedge (\mathrm{p} * \mathrm{r})) \mathrel{-\!\!*} (\mathrm{p} * (\mathrm{q} \wedge \mathrm{r})$

We now reconsider our second example in figure 11. There is no change until step 4. Step 5, on the contrary, introduces two new variables v_1, v_2 and the new requirement $req_1 : v_1v_2 \leq c_1$. Step 6 simply spreads variable v_2 in both branches of the additive conjunct. After step 6, the tableau expansion is completed. At this step, it is important to mention that, in the free variable version, one first tries to close branches and checks, afterwards, if the requirements are satisfied from the assertions, while in the initial version, one tries to solve requirements as they are introduced by each $\pi\beta$ expansion step in order to perform a valid expansion.

Let us come back to our example. Once again, we are faced with the problem of finding a ground substitution that closes each branch of the tableau. We begin with the first branch and notice that there are two potential connections $\langle\, \mathrm{T}\, \mathrm{p} : c_2, \mathrm{F}\, \mathrm{p} : v_1 \,\rangle$ and $\langle\, \mathrm{T}\, \mathrm{p} : c_4, \mathrm{F}\, \mathrm{p} : v_1 \,\rangle$. The first connection yields $v_1 \mapsto c_2$, but then, we must match the requirement $req_1 : v_1v_2 \leq c_1$. From the assertion $ass_1 : c_2c_3 \leq c_1$, it is clear that the only possibility for v_2 is $v_2 \mapsto c_3$, which closes the second branch with $\langle\, \mathrm{T}\, \mathrm{q} : c_3, \mathrm{F}\, \mathrm{q} : c_3 \,\rangle$ but leaves the third one open since $c_5 \leq c_3$ cannot be proved for $\langle\, \mathrm{T}\, \mathrm{r} : c_5, \mathrm{F}\, \mathrm{r} : c_3 \,\rangle$. We must therefore try the other possibility, that is, $v_1 \mapsto c_4$. It leads, by assertion $ass_2 : c_4c_5 \leq c_1$, to set $v_2 \mapsto c_5$, which allows us to close the first and the third branches of the tableau. But then, the second branch remains open since $c_3 \leq c_5$ cannot be proved for $\langle\, \mathrm{T}\, \mathrm{q} : c_3, \mathrm{F}\, \mathrm{q} : c_5 \,\rangle$. Thus, whatever admissible ground substitution



* c_i is a new constant

Figure 12: Expansion Rules for IL

we may choose, we are left with an open branch and the resulting conclusion is that the formula is not provable.

These examples illustrate the actual interest to introduce free variables into labels. In general, the use of free variables rules is crucial for efficient implementation [9]. They reduce at each step the number of possible next steps in the construction of the tableau proof and thus the size of the search space. Free variables can later be instantiated when a tableau is closed. These characteristics can be used for our labelled tableau calculus in order to optimize proof search in $\mathbf{BI}^{\perp}$.

Free variables help us to improve efficiency of the proof-search since they remove many irrelevant $\pi\beta$ expansion steps by introducing variables whose evaluation is postponed until enough information is available. However, the countermodels become more difficult to generate because multiple $\pi\beta$ extensions in the initial system correspond, in the free variable version, to distinct variable instantiations. In fact, the introduction of variables involves a kind of branches "factorization". Therefore, countermodels must be built by recomposition from distinct open branches in several steps, in fact, one step for each admissible valuation of the variables, i.e., valuations which make the requirements hold w.r.t. (ground) assertions. Another difficulty that arises with free variables is how to limit the application of the $\pi\beta$ rules, i.e., the ones that introduce variables, while still retaining completeness. In [7], it is claimed that such rules need only be expanded once, but this point has to be thoroughly and carefully studied. Our feeling is that some formulas(like $((((\mathrm{p} \to \mathrm{q}) \to \mathrm{p}) \to \mathrm{p}) \to \mathrm{q}) \to \mathrm{q}$, for example) may require several expansions of the same $\pi\beta$ formula, which could be handled by assigning multiplicities to $\pi\beta$ much like in matrix-based methods. Further work will focus on this point.

7 A New Tableau Method for IL

From the conservativity result of theorem 2.2, we can see intuitionistic logic (**IL**) as a subsystem of **BI**. Consequently, the restriction of the initial $\mathbf{TBI}^{\perp}$ calculus to the additive connectives, i.e., the expansion rules of figure 12, directly provides a new tableau method for **IL**. Compared to LDS works [5] and also to standard tableaux methods [9] for **IL** this new method is based on an original and powerful treatment of the Kripke semantics through labels and constraints. It naturally leads to a direct construction of countermodels. The introduction of assertions and requirements improves proof-search by capturing the necessary and sufficient semantical information into a dependency graph. This structure allows to analyze provability through the verification of requirements on a finite portion of the underlying infinite labelling algebra. In case of non-provability, the countermodel can be immediately generated from an open branch without any further recomposition (by instantation) from other open branches like in standard methods [9]. In order to emphasize the characteristics of this new tableau method for **IL**, we develop and analyze it in details in this section.

All definitions (complementary formulas, closed tableau, fulfilled formula, etc...) given for $\mathbf{BI}^{\perp}$ are restricted to **IL** by simply discarding statements involving either multiplicative connectives, or the multiplicative unit I, or the $\circ$ connective. As an immediate consequence of this restriction, the resulting labelling algebra is only composed of atomic labels and the constraints are preordering relations between such atomic labels.
As we consider $\mathbf{BI}^{\perp}$, the direct restriction to **IL** does not include the $\perp$ unit. Nevertheless, since Kripke models are complete for **IL** (with $\perp$), the restriction can be straightforwardly extended to $\perp$ by adding $\mathrm{T} \perp : x$ in the definition of closure and $\mathcal{B} \Vdash \mathrm{F} \perp : x$ in the definition of fulfillment.

Definition 7.1. *Let $\mathcal{B}$ be an tableau branch, two signed formulas $\mathrm{T\,A} : x$, $\mathrm{F\,A} : y$ of $\mathcal{F}(\mathcal{B})$ are said* complementary in $\mathcal{B}$ *if and only if $x \le y \in \overline{Ass}(\mathcal{B})$.*

Definition 7.2. *A tableau t is* closed *if, for all its branches $\mathcal{B}$,*
i) there are two formulas $\mathrm{T\,A} : x$ and $\mathrm{F\,A} : y$ that are complementary in $\mathcal{B}$ or there is $\mathrm{F} \top : x$ in $\mathcal{B}$ or $\mathrm{T} \perp : x$ in $\mathcal{B}$; ii) for all $x \le y \in Req(\mathcal{B})$, we have $x \le y \in \overline{Ass}(\mathcal{B})$.
A tableau sequence $\mathcal{T} = t_1, t_2, \ldots, t_n$ is closed *if it contains at least one closed tableau.*

7.1 A Tableau Construction Procedure for IL

Let us recall the main definitions in order to define a procedure for the tableau construction in **IL**.

Definition 7.3. *Let $\mathcal{B}$ be a tableau branch, a signed formula $\mathrm{Sg\,X} : x$ is* fulfilled *(or* completely analyzed*) in $\mathcal{B}$, noted $\mathcal{B} \Vdash \mathrm{Sg\,X} : x$, if it satisfies one of the following conditions*

1. $\mathcal{B} \Vdash \mathrm{F} \perp : x$
2. $\mathcal{B} \Vdash \mathrm{F} \top : x$
3. $\mathcal{B} \Vdash \mathrm{F\,p} : x$ *iff there exists* $\mathrm{F\,p} : y \in \mathcal{B}$ *such that* $x \le y \in \overline{Ass}(\mathcal{B})$
4. $\mathcal{B} \Vdash \mathrm{F\,A} \wedge \mathrm{B} : x$ *iff either* $\mathcal{B} \Vdash \mathrm{F\,A} : x$ *or* $\mathcal{B} \Vdash \mathrm{F\,B} : x$
5. $\mathcal{B} \Vdash \mathrm{F\,A} \vee \mathrm{B} : x$ *iff both* $\mathcal{B} \Vdash \mathrm{F\,A} : x$ *and* $\mathcal{B} \Vdash \mathrm{F\,B} : x$
6. $\mathcal{B} \Vdash \mathrm{F\,A} \to \mathrm{B} : x$ *iff there exists* $y \in \mathcal{D}(\mathcal{B})$ *such that* $x \le y \in \overline{Ass}(\mathcal{B})$ *and both* $\mathcal{B} \Vdash \mathrm{T\,A} : y$ *and* $\mathcal{B} \Vdash \mathrm{F\,B} : y$
7. $\mathcal{B} \Vdash \mathrm{T} \top : x$
8. $\mathcal{B} \Vdash \mathrm{T\,p} : x$ *iff there exists* $\mathrm{T\,p} : y \in \mathcal{B}$ *such that* $y \le x \in \overline{Ass}(\mathcal{B})$
9. $\mathcal{B} \Vdash \mathrm{T\,A} \wedge \mathrm{B} : x$ *iff both* $\mathcal{B} \Vdash \mathrm{T\,A} : x$ *and* $\mathcal{B} \Vdash \mathrm{T\,B} : x$
10. $\mathcal{B} \Vdash \mathrm{T\,A} \vee \mathrm{B} : x$ *iff either* $\mathcal{B} \Vdash \mathrm{T\,A} : x$, *or* $\mathcal{B} \Vdash \mathrm{T\,B} : x$
11. $\mathcal{B} \Vdash \mathrm{T\,A} \to \mathrm{B} : x$ *iff for any* $y \in \mathcal{D}(\mathcal{B})$ *such that* $x \le y \in \overline{Ass}(\mathcal{B})$ *either* $\mathcal{B} \Vdash \mathrm{F\,A} : y$, *or* $\mathcal{B} \Vdash \mathrm{T\,B} : y$

Definition 7.4. *A tableau branch $\mathcal{B}$ is* completed *if it is* open *and every signed formula $\mathrm{Sg\,A} : x$ of $\mathcal{B}$ is* fulfilled. *A tableau is* completed *if it has a completed branch.*

Lemma 7.1 (monotonicity). *A completed branch $\mathcal{B}$ satisfies Kripke monotonicity, more precisely,*
1) $\mathcal{B} \Vdash \mathrm{F\,A} : x$ and $y \le x \in \overline{Ass}(\mathcal{B})$ imply $\mathcal{B} \Vdash \mathrm{F\,A} : y$, and
2) $\mathcal{B} \Vdash \mathrm{T\,A} : x$ and $x \le y \in \overline{Ass}(\mathcal{B})$ imply $\mathcal{B} \Vdash \mathrm{T\,A} : y$.

Proof. By structural induction on $\mathrm{Sg\,X} : x$.

Lemma 7.2 (consistency). *If $\mathcal{B}$ is a completed branch, then, for any* **IL** *proposition* A, *we cannot have both $\mathcal{B} \Vdash \mathrm{T\,A} : x$ and $\mathcal{B} \Vdash \mathrm{F\,A} : x$.*

Proof. See the proof of lemma 4.5.

We give a tableau construction procedure that is an algorithm that, for a given formula A, builds a tableau sequence $t_1, t_2, \ldots, t_n$ until there exists a tableau t_i that is either closed or includes a completed branch. This procedure for **IL**, that is called **TBI/IL**, is the following:

The initial signed formula is F A : 1 and it is marked unvisited.
Repeat until $\mathcal{T}$ is closed or completed
  if there is no unvisited formula Sg X : x in any open branch $\mathcal{B}$
  then mark all $\pi\beta$-formulas in all open branches as unvisited
  else mark Sg X : x as visited
    if Sg X : x is not fulfilled in $\mathcal{B}$
    then if Sg X : x is of type $\pi\beta$
      then expand Sg X : x with a label y for which the formula is not fulfilled
      else expand Sg X : x according to its type

An important result is the termination of this procedure. First, let us remark that the F $\rightarrow$ rule is the only one that introduces new atomic labels and its application can potentially lead to infinite branches. In order to have a terminating procedure, we give a liberalized version of the F $\rightarrow$ rule that allows, under given conditions, to reuse existing constants.

Lemma 7.3. *Let $\mathcal{B}$ be a branch and* F A $\rightarrow$ B : x *be a signed formula in $\mathcal{B}$, the following rule*

$$\begin{array}{c}\mathrm{F\ A} \rightarrow \mathrm{B} : x \\ | \\ \mathrm{T\ A} : x \\ \mathrm{F\ B} : x\end{array}$$

is admissible provided there exists T A : y *in $\mathcal{B}$ such that* $y \leq x \in \overline{Ass}(\mathcal{B})$.

Proof. Let $\mathcal{K} = \langle \mathcal{M}, \models, [\![-]\!] \rangle$ be a Kripke model and $\|-\|$ be a realization of $\mathcal{B}$ in $\mathcal{K}$. Let us suppose that $\|x\| = m$ and $\|y\| = n$, then, by definition of a realization, F A $\rightarrow$ B : x, T A : $y \in \mathcal{B}$ and $y \leq x \in \overline{Ass}(\mathcal{B})$ imply $m \not\models$ A $\rightarrow$ B, $n \models$ A and $n \sqsubseteq m$. Since $n \models$ A and $n \sqsubseteq m$, it follows from property (K) of definition 2.4 that $m \models$ A. On the other hand, $m \not\models$ A $\rightarrow$ B implies that there exists n' such that $m \sqsubseteq n'$, $n' \models$ A and $n' \not\models$ B, which, by contraposition of condition (K), yields $m \not\models$ B.

Theorem 7.1 (termination). *The tableau construction procedure, including the liberalized version of the* F $\rightarrow$ *rule, terminates.*

Proof. We only give the sketch of the proof. For that, we slightly modify our notation for constants. When we expand a labelled signed formula F A $\rightarrow$ B : x, we suppose, instead of introducing a new constant c_i and a new assertion $x \leq c_i$, that we simply introduce a term $c_i(x)$, with the meaning that c_i depends on x in such a way that the assertion $x \leq c_i$ is implicitly verified.
Then, starting from the the initial formula to prove, we associate to each subformula of the form F A $\rightarrow$ B a unique functor symbol which will be used, parametrized with the corresponding label, by each expansion of F A $\rightarrow$ B. For example, if c_i is the functor symbol associated to F A $\rightarrow$ B, then, the expansion of F A $\rightarrow$ B : x introduces the term $c_i(x)$ and, if we later expand F A $\rightarrow$ B : y, we simply reuse the same functor symbol c_i, but parametrized with the label y, i.e., $c_i(y)$. This roughly corresponds to a kind of skolemization of the initial formula, and thus, we only have one "real" atomic label left, namely, the unit 1. The other labels are terms of the form $c_{i1}(c_{i2}(\cdots c_{ik}(1)\cdots))$, or $c_{i1}c_{i2}\cdots c_{ik}(1)$ for brevity. Moreover, if s is a subterm of t, we implicitly have $s \leq t$. Now, we suppose that we need to expand a formula F A $\rightarrow$ B : $c_{i1}\cdots c_{ik}(1)$, with some $c_{ij(j\in[1..k])}$ being the functor symbol corresponding to F A $\rightarrow$ B. Then, F A $\rightarrow$ B previously occured in the branch with the label $c_{i(j+1)}\cdots c_{ik}(1)$ and has been expanded to produce T A : $c_{ij}\cdots c_{ik}(1)$. Besides, since $c_{ij}\cdots c_{ik}(1)$ is a subterm of $c_{i1}\cdots c_{ik}(1)$, we also have $c_{ij}\cdots c_{ik}(1) \leq c_{i1}\cdots c_{ik}(1)$. Therefore, we are in the situation required by the liberalized F $\rightarrow$ rule to be applicable and we can perform our expansion step by simply reusing the term $c_{i1}\cdots c_{ik}(1)$ instead of introducing a new composition $c_{ij}c_{i1}\cdots c_{ik}(1)$.
This shows that, if we use the liberalized version of F $\rightarrow$, the only terms that can occur in a tableau branch are terms which do not contain the same functor symbol twice. Now, let us remark that since there are obviously finitely many distinct subformulas of the initial formula to prove, there are also finitely many associated functor symbols. Consequently, there are finitely many possibilities to combine those functors so that each of them occurs at most once in the combination, which means that the F $\rightarrow$ expansion steps

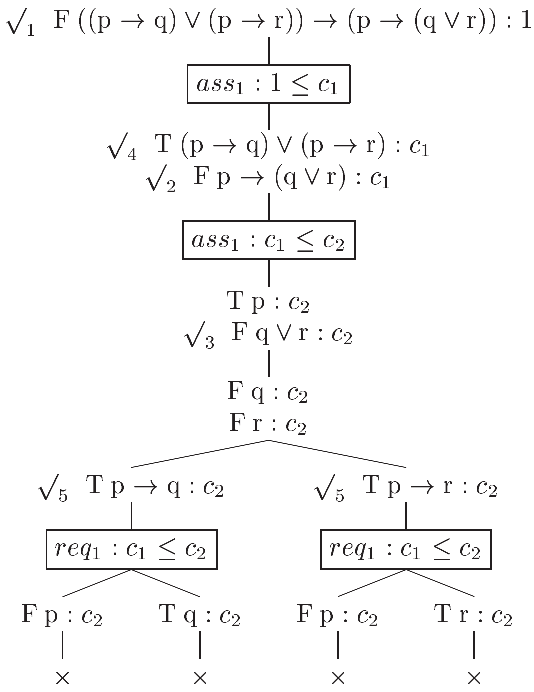


Figure 13: Labelled Tableau for $((\mathrm{p} \to \mathrm{q}) \vee (\mathrm{p} \to \mathrm{r})) \to (\mathrm{p} \to (\mathrm{q} \vee \mathrm{r}))$

cannot infinitely produce new compositions during a branch expansion. The termination property then directly follows.

7.2 Proof and Countermodel Construction

In this subsection, we illustrate how to build proofs and countermodels in our new calculus for **IL**. There exist several deduction systems for (propositional) **IL** that build proofs but also countermodels in case of non-provability [8, 13, 24, 33]. The specificity of our method is the use of labels in order to improve proof-search. The example, given in figure 13, with the formula $((\mathrm{p} \to \mathrm{q}) \vee (\mathrm{p} \to \mathrm{r})) \to (\mathrm{p} \to (\mathrm{q} \vee \mathrm{r}))$ well illustrates how the **TBI/IL** calculus works. In this example, the tableau is closed and consequently the given formula is provable in **IL**.

In a second example, given in figure 14, we consider the formula $(\mathrm{p} \to (\mathrm{q} \vee \mathrm{r})) \to ((\mathrm{p} \to \mathrm{q}) \vee (\mathrm{p} \to \mathrm{r}))$ that leads to an unclosed tableau. As one can notice, the $\pi\beta$ signed formula $\mathrm{T}\ \mathrm{p} \to (\mathrm{q} \vee \mathrm{r}) : c_1$ has been expanded twice: a first time in step 5 with the constant c_2 and a second time in step 7 with the constant c_3 because both $c_1 \leq c_2$ and $c_1 \leq c_3$ belong to the closure of the assertions. As in $\mathbf{TBI}^{\perp}$ such multiple expansions are necessary in order to have a completed tableau.

With our tableau system, we can directly generate countermodels from open branches. Since we do not deal with the $\circ$ connective, the expansion of a tableau branch provides, via its assertions, a preordering relation from which, in the case of the existence of a completed branch, we can directly define a Kripke model in the following way.

Definition 7.5. *Let $\mathcal{B}$ be a completed branch, the structure $\mathcal{M}(\mathcal{B})$ is defined by $\mathcal{M}(\mathcal{B}) = \langle M, \sqsubseteq \rangle$ where*
i) M is the set $\mathcal{D}(\mathcal{B})$,
ii) the relation $\sqsubseteq$ is defined by: $x \sqsubseteq y$ if and only if $x \leq y \in \overline{Ass}(\mathcal{B})$.

Definition 7.6. *Let $\mathcal{M}(\mathcal{B}) = \langle M, \sqsubseteq \rangle$ be the M-structure of a completed branch $\mathcal{B}$ and $\mathcal{P}(L)$ denote the collection of* **IL** *propositions over a language L of propositional letters. We define the interpretation*

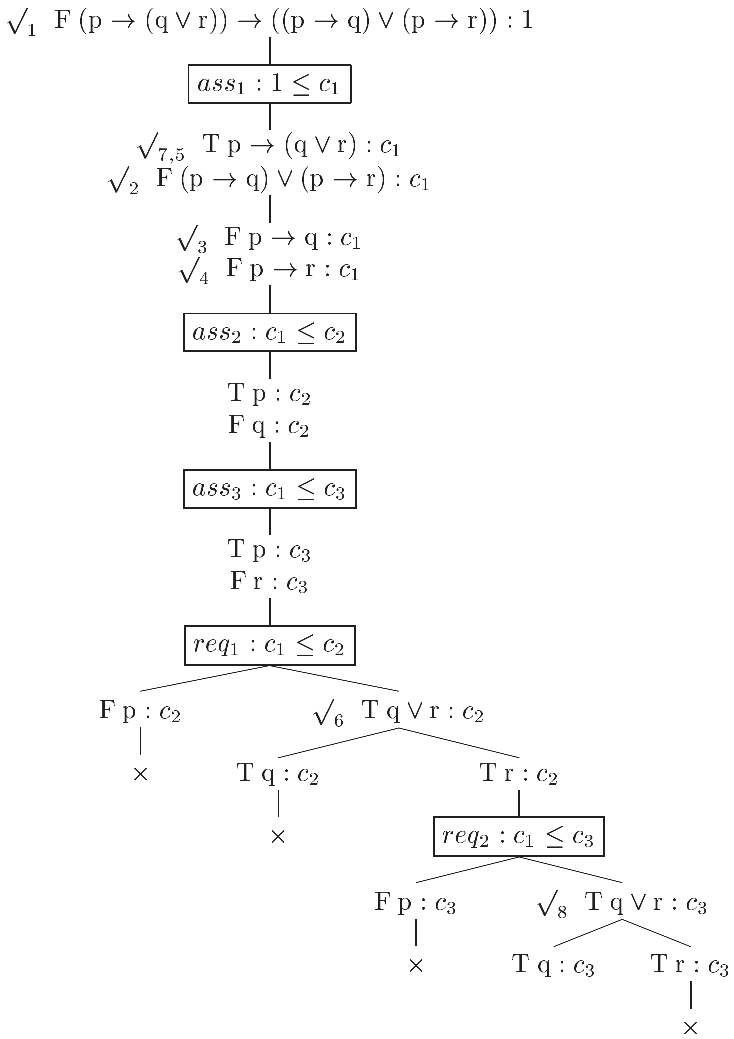


Figure 14: Labelled Tableau for $(\mathrm{p} \to (\mathrm{q} \vee \mathrm{r})) \to ((\mathrm{p} \to \mathrm{q}) \vee (\mathrm{p} \to \mathrm{r}))$

$[\![-]\!]_{\mathcal{B}} : L \to \mathcal{P}(M)$, for any atomic proposition p, *by* $[\![\mathrm{p}]\!]_{\mathcal{B}} = \{\, x \mid \mathcal{B} \Vdash \mathrm{T\,p} : x \,\}$.

Theorem 7.2. *Let $\mathcal{B}$ be a completed branch, the triple $\langle\, \mathcal{M}(\mathcal{B}), \models, [\![-]\!]_{\mathcal{B}} \,\rangle$ is a Kripke model of $\mathcal{B}$,* i.e.,
1) $[\![-]\!]_{\mathcal{B}}$ satisfies the (K) condition of definition 2.4;
2) for any proposition A, *we have*
a) $\mathcal{B} \Vdash \mathrm{T\,A} : x$ implies $x \models \mathrm{A}$,
b) $\mathcal{B} \Vdash \mathrm{F\,A} : x$ implies $x \not\models \mathrm{A}$.

Proof.
1) We suppose $x \in [\![\mathrm{p}]\!]_{\mathcal{B}}$ and $x \sqsubseteq y$. Then, $\mathcal{B} \Vdash \mathrm{T\,p} : x$ and $x \leq y \in \overline{Ass}(\mathcal{B})$. Therefore, by lemma 7.1, $\mathcal{B} \Vdash \mathrm{T\,p} : y$, i.e., $y \in [\![\mathrm{p}]\!]_{\mathcal{B}}$.

2.a), 2.b) We proceed by structural induction on Sg A : x.
- if $\mathcal{B} \Vdash \mathrm{T\,p} : x$ then, by definition, $x \in [\![\mathrm{p}]\!]_{\mathcal{B}}$, i.e., $x \models \mathrm{p}$.
- if $\mathcal{B} \Vdash \mathrm{F\,p} : x$ then, by lemma 7.2, $\mathcal{B} \not\Vdash \mathrm{T\,p} : x$. Thus, $x \notin [\![\mathrm{p}]\!]_{\mathcal{B}}$, i.e., $x \not\models \mathrm{p}$.
- if $\mathcal{B} \Vdash \mathrm{T\,B} \wedge \mathrm{C} : x$ then, both $\mathcal{B} \Vdash \mathrm{T\,B} : x$ and $\mathcal{B} \Vdash \mathrm{T\,C} : x$. By the induction hypothesis, it follows that $x \models \mathrm{B}$ and $x \models \mathrm{C}$, i.e., $x \models \mathrm{B} \wedge \mathrm{C}$.

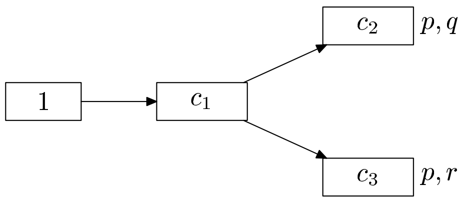


Figure 15: Kripke Countermodel for $(\mathrm{p} \to (\mathrm{q} \vee \mathrm{r})) \to ((\mathrm{p} \to \mathrm{q}) \vee (\mathrm{p} \to \mathrm{r}))$

- if $\mathcal{B} \Vdash \mathrm{F\,B} \wedge \mathrm{C} : x$ then, either $\mathcal{B} \Vdash \mathrm{F\,B} : x$ or $\mathcal{B} \Vdash \mathrm{F\,C} : x$. By the induction hypothesis, it follows that $x \not\models \mathrm{B}$ or $x \not\models \mathrm{C}$, i.e., $x \not\models \mathrm{B} \wedge \mathrm{C}$.
- case $\mathcal{B} \Vdash \mathrm{T\,B} \vee \mathrm{C} : x$ is similar to case $\mathcal{B} \Vdash \mathrm{F\,B} \wedge \mathrm{C} : x$
- case $\mathcal{B} \Vdash \mathrm{F\,B} \vee \mathrm{C} : x$ is similar to case $\mathcal{B} \Vdash \mathrm{T\,B} \wedge \mathrm{C} : x$
- if $\mathcal{B} \Vdash \mathrm{T\,B} \to \mathrm{C} : x$ then, for any y in $\mathcal{D}(\mathcal{B})$ such that $x \leq y \in \overline{Ass}(\mathcal{B})$, either $\mathcal{B} \Vdash \mathrm{F\,B} : y$ or $\mathcal{B} \Vdash \mathrm{T\,C} : y$. By the induction hypothesis, it follows that either $y \not\models \mathrm{B}$ or $y \models \mathrm{C}$ for any y in M such that $x \sqsubseteq y$, i.e., $x \models \mathrm{B} \to \mathrm{C}$.
- if $\mathcal{B} \Vdash \mathrm{F\,B} \to \mathrm{C} : x$ then, there exists y in $\mathcal{D}(\mathcal{B})$ such that $x \leq y \in \overline{Ass}(\mathcal{B})$, $\mathcal{B} \Vdash \mathrm{F\,B} : y$ and $\mathcal{B} \Vdash \mathrm{T\,C} : y$. By the induction hypothesis, it follows that there exists $y \in M$ such that $x \sqsubseteq y$, $y \models \mathrm{B}$ and $y \not\models \mathrm{C}$, i.e., $x \not\models \mathrm{B} \to \mathrm{C}$.

For the example of figure 14, in which the fourth branch is open, theorem 7.2 yields the countermodel $\langle \mathcal{M}(\mathcal{B}), \models, [\![-]\!] \rangle$ where $M = \{ 1, c_1, c_2, c_3 \}$, $\sqsubseteq$ is defined by $1 \sqsubseteq c_1 \sqsubseteq c_2, c_3$ and $\models$ is given by the table

x	p	q	r
$[\![\mathrm{x}]\!]$	$\{ c_2, c_3 \}$	$\{ c_2 \}$	$\{ c_3 \}$

It corresponds to the Kripke tree given in figure 7.2, on which we can verify that $c_1 \models \mathrm{p} \to (\mathrm{q} \vee \mathrm{r})$, since $c_2 \models \mathrm{q} \vee \mathrm{r}$ and $c_3 \models \mathrm{q} \vee \mathrm{r}$. Moreover, $c_1 \not\models (\mathrm{p} \to \mathrm{q}) \vee (\mathrm{p} \to \mathrm{r})$ because $c_2 \not\models \mathrm{p} \to \mathrm{r}$ and $c_3 \not\models \mathrm{p} \to \mathrm{q}$.

Let us make some remarks on the countermodel generation. Firstly, we can easily notice that the countermodels we obtain are in fact Kripke trees with 1 as the root. Furthermore, such Kripke trees are so-called strong countermodels since the root is precisely the node at which the initial tableau formula is not forced. Secondly, the countermodels are directly obtained from a completed branch without performing any backtracking or further computations like in standard methods dedicated to **IL**. Indeed, backtracking has been internalized in the multiple $\mathrm{T} \to$ expansions. In practice, these rules can be more efficiently handled by using free variables in order to delay the choice of labels used to close branches and then avoid to test all possibilities.

7.3 Properties of the IL Labelled Calculus

We can refine the properties of the $\mathbf{TBI}^{\perp}$ calculus and then obtain the following:

Theorem 7.3 (soundness). *Let* A *be a proposition of IL, if there exists a closed tableau sequence for* A *in* **TBI/IL**, *then* A *is valid in Kripke semantics.*

Proof. See the proof of theorem 4.1 with restriction to **IL**.

Theorem 7.4 (completeness). *Let* A *be a* **IL** *formula, if* A *is valid in Kripke semantics, then there exists a closed tableau sequence for* A *in* **TBI/IL**.

Proof. Let us suppose that there is no closed tableau sequence for A. Then, by theorem 7.1, the tableau construction procedure given in subsection 7.1 terminates with a tableau which contains a completed branch. From this branch we can build, following theorem 7.2, a Kripke model in which A is not satisfied. Therefore, A is not valid in the Kripke semantics.

Theorem 7.5 (finite model property). *Let* A *be a proposition of* **IL**, *if* A *is not valid in Kripke semantics, then, there exists a finite Kripke model in which* A *is not satisfied.*

Proof. Let us assume that A is not valid in the Kripke semantics. From theorem 7.1, we deduce that the tableau construction procedure terminates by providing a tableau with a completed branch that is necessarily finite. The construction given in definition 7.5 results in a finite Kripke model.

Further work will be devoted to efficient constraint satisfaction in **IL**. This new method emphasizes that, in a logic with a resource semantics, labels and constraints can be used in order to capture its consequence relation. It appears as a more general approach than trying to capture the right order of rules applications in the sequent calculus, with prefixes for instance. This point will be more deeply analyzed in comparison with other constraint-based proof calculi for **IL** [38]. In case of the definition of a free variable variant, we have the same problems that we already mentioned in the case of **BI**. Compared to other proof-search methods based on specific sequent calculi with countermodel generation facility [24, 33], our labelled tableau method is efficient and avoids many backtrackings. As said before, the notion of minimality is difficult to state for the models of such substructural logics and further work will be also devoted to it from this study.

8 A New Tableau Method for MILL

From the conservativity result we previously mentioned in theorem 2.2, we can see multiplicative intuitionistic linear logic (**MILL**) as a subsystem of **BI**. Consequently, the restriction of the initial **TBI** system to the multiplicative connectives, i.e., the expansion rules of figure 16, directly provides a tableau method for **MILL**.
There exist proof-search methods for (propositional) multiplicative linear logic (**MLL**) based on different notions like connections [11, 22], tableaux [25, 26] with prefixes or canonical proofs [1, 18]. They cannot be easily extended to **MILL** and the design of a tableau method with labels from the Urquhart's semantics appears as an appropriate solution. The restriction of **TBI** to the multiplicatives leads to a suitable tableau system for **MILL**. It can be related to a tableau formulation of D'Agostino and Gabbay's LKE tableau for the (I, $\otimes$, $\multimap$) fragment[7]. As it corresponds to a first tableau method for **MILL**, we present the main points into details.

All definitions (complementary formulas, closed tableau, fulfilled formula, etc...) given for **BI** are restricted to **MILL** by discarding statements involving either the additive connectives, or the additive units. Since we keep the $\circ$ connective, the **TBI**/**MILL** calculus is mostly based on a restriction on propositions, i.e., propositions containing only multiplicatives. Therefore, as opposed to **TBI**/**IL**, the labelling algebra and the corresponding closure operator remain unchanged w.r.t. **BI**. The main difference is that we now deal with assertions whose left-hand side is necessarily a composition of two atomic labels. This comes from that the T $*$ rule is now the only rule that introduces assertions.

8.1 A tableau construction procedure for MILL

Let us recall the main definitions in the case of this restriction to **MILL**.

Definition 8.1. *Let* $\mathcal{B}$ *be a tableau branch, a signed formula* $\mathrm{Sg}\,\mathrm{X} : x$ *is* fulfilled *(or* completely analyzed*) in* $\mathcal{B}$*, noted* $\mathcal{B} \Vdash \mathrm{Sg}\,\mathrm{X} : x$*, if it satisfies one of the following conditions*

1. $\mathcal{B} \Vdash \mathrm{F\,I} : x$ *iff* $1 \leq x \notin \overline{Ass}(\mathcal{B})$
2. $\mathcal{B} \Vdash \mathrm{F\,A} * \mathrm{B} : x$ *iff for any* $y,\, z \in \mathcal{D}(\mathcal{B})$ *such that* $yz \leq x \in \overline{Ass}(\mathcal{B})$ *either* $\mathcal{B} \Vdash \mathrm{F\,A} : y$*, or* $\mathcal{B} \Vdash \mathrm{F\,B} : z$
3. $\mathcal{B} \Vdash \mathrm{F\,A} \mathrel{-\!\!*} \mathrm{B} : x$ *iff there exist* $y,\, xy \in \mathcal{D}(\mathcal{B})$ *such that both* $\mathcal{B} \Vdash \mathrm{T\,A} : y$ *and* $\mathcal{B} \Vdash \mathrm{F\,B} : xy$
4. $\mathcal{B} \Vdash \mathrm{T\,I} : x$ *iff* $1 \leq x \in \overline{Ass}(\mathcal{B})$
5. $\mathcal{B} \Vdash \mathrm{T\,A} * \mathrm{B} : x$ *iff there exist* $y,\, z \in \mathcal{D}(\mathcal{B})$ *such that* $yz \leq x \in \overline{Ass}(\mathcal{B})$ *and both* $\mathcal{B} \Vdash \mathrm{T\,A} : y$ *and* $\mathcal{B} \Vdash \mathrm{T\,B} : z$
6. $\mathcal{B} \Vdash \mathrm{T\,A} \mathrel{-\!\!*} \mathrm{B} : x$ *iff for any* $y,\, xy \in \mathcal{D}(\mathcal{B})$ *either* $\mathcal{B} \Vdash \mathrm{F\,A} : y$*, or* $\mathcal{B} \Vdash \mathrm{T\,B} : xy$

$$\begin{array}{c} \mathrm{T}\,\mathrm{A} * \mathrm{B} : x \\ | \\ \boxed{ass : c_i c_j \leq x}\ ^{*} \\ | \\ \mathrm{T}\,\mathrm{A} : c_i \\ \mathrm{T}\,\mathrm{B} : c_j \end{array} \qquad \begin{array}{c} \mathrm{F}\,\mathrm{A} -\!\!* \mathrm{B} : x \\ | \\ \mathrm{T}\,\mathrm{A} : c_i \\ \mathrm{F}\,\mathrm{B} : x c_i \end{array}$$

$$\begin{array}{c} \mathrm{F}\,\mathrm{A} * \mathrm{B} : x \\ | \\ \boxed{req : yz \leq x} \\ \mathrm{F}\,\mathrm{A} : y \qquad \mathrm{F}\,\mathrm{B} : z \end{array} \qquad \begin{array}{c} \mathrm{T}\,\mathrm{A} -\!\!* \mathrm{B} : x \\ \mathrm{F}\,\mathrm{A} : y \qquad \mathrm{T}\,\mathrm{B} : xy \end{array}$$

* c_i, c_j are new constants

Figure 16: Expansion Rules for MILL

Definition 8.2. *A tableau branch* $\mathcal{B}$ *is* completed *if it is* open *and every signed formula* Sg A : x *of* $\mathcal{B}$ *is* fulfilled. *A tableau is* completed *if it has a completed branch.*

Lemma 8.1 (monotonicity). *A completed branch* $\mathcal{B}$ *satisfies Kripke monotonicity, more precisely,*
1) $\mathcal{B} \Vdash \mathrm{F}\,\mathrm{A} : x$ *and* $y \leq x \in \overline{Ass}(\mathcal{B})$ *imply* $\mathcal{B} \Vdash \mathrm{F}\,\mathrm{A} : y$*, and*
2) $\mathcal{B} \Vdash \mathrm{T}\,\mathrm{A} : x$ *and* $x \leq y \in \overline{Ass}(\mathcal{B})$ *imply* $\mathcal{B} \Vdash \mathrm{T}\,\mathrm{A} : y$*.*

Proof. By structural induction on Sg A : x.

Lemma 8.2 (consistency). *If* $\mathcal{B}$ *is a completed branch, then, for any* **BI** *proposition* A*, we cannot have both* $\mathcal{B} \Vdash \mathrm{T}\,\mathrm{A} : x$ *and* $\mathcal{B} \Vdash \mathrm{F}\,\mathrm{A} : x$*.*

Proof. See the proof of lemma 4.5.

We give a tableau construction procedure that is an algorithm that, for a given formula A, builds a tableau sequence $t_1, t_2, \ldots, t_n$ until there exists a tableau t_i that is either closed or includes a completed branch. This procedure for **MILL**, that is called **TBI/MILL**, is the following:
The initial signed formula is F A : 1 and it is marked unvisited.
Repeat until $\mathcal{T}$ is closed or completed
  if there is no unvisited formula Sg X : x in any open branch $\mathcal{B}$
  then mark all $\pi\beta$-formulas in all open branches as unvisited
  else mark Sg X : x as visited
    if Sg X : x is not fulfilled in $\mathcal{B}$
    then if Sg X : x is of type $\pi\beta$
      then expand Sg X : x with a label y for which the formula is not fulfilled
      else expand Sg X : x according to its type

As in **TBI/IL**, the $\pi\alpha$ rules, namely the F $-\!\!*$ and T $*$ rules, require the introduction of new constants that can lead to infinite branches. Again, a key point is to propose liberalized versions of those rules which are justified by the completeness of **MILL** with respect to *regular* Kripke resource models, i.e., models for which the forcing relation satisfies the following regularity property [7]: *whenever there exists a world satisfying a proposition* A*, there also exists a world, called* A*-characteristic, which is the least, w.r.t. the underlying preordering, to satisfy* A.
In this context, for the realization of a tableau branch in some Kripke resource model, we can assume that every constant introduced in a tableau branch refers to a characteristic world. For example, the

$$
\begin{array}{c}
\surd_1 \;\; \mathrm{F}\,((\mathrm{p} * \mathrm{q}) \mathrel{-\!\!*} \mathrm{r})) \mathrel{-\!\!*} (\mathrm{q} \mathrel{-\!\!*} (\mathrm{p} \mathrel{-\!\!*} \mathrm{r})) : 1 \\
| \\
\surd_4 \;\; \mathrm{T}\,(\mathrm{p} \mathrel{-\!\!*} \mathrm{q}) \mathrel{-\!\!*} \mathrm{r} : c_1 \\
\surd_2 \;\; \mathrm{F}\,\mathrm{q} \mathrel{-\!\!*} (\mathrm{p} \mathrel{-\!\!*} \mathrm{r}) : c_1 \\
| \\
\mathrm{T}\,\mathrm{q} : c_2 \\
\surd_3 \;\; \mathrm{F}\,\mathrm{p} \mathrel{-\!\!*} \mathrm{r} : c_1 c_2 \\
| \\
\mathrm{T}\,\mathrm{p} : c_3 \\
\mathrm{F}\,\mathrm{r} : c_1 c_2 c_3 \\
\begin{array}{cc}
\begin{array}{c}
\surd_5 \;\; \mathrm{F}\,\mathrm{p} * \mathrm{q} : c_2 c_3 \\
| \\
\boxed{req_1 : c_3 c_2 \leq c_2 c_3} \\
\begin{array}{cc}
\mathrm{F}\,\mathrm{p} : c_3 & \mathrm{F}\,\mathrm{q} : c_2 \\
| & | \\
\times & \times
\end{array}
\end{array}
&
\begin{array}{c}
\mathrm{T}\,\mathrm{r} : c_1 c_2 c_3 \\
| \\
\times
\end{array}
\end{array}
\end{array}
$$

Figure 17: Labelled Tableau for $((\mathrm{p} * \mathrm{q}) \mathrel{-\!\!*} \mathrm{r})) \mathrel{-\!\!*} (\mathrm{q} \mathrel{-\!\!*} (\mathrm{p} \mathrel{-\!\!*} \mathrm{r}))$

constant a introduced in $\mathrm{T\,A} : a$, $\mathrm{F\,B} : xa$ by the expansion of $\mathrm{F\,A} \mathrel{-\!\!*} \mathrm{B} : x$ is assumed to refer to an A-characteristic world.

Lemma 8.3. *Let $\mathcal{B}$ be a tableau branch. The following liberalized rules,*

$$
\begin{array}{c}
\mathrm{F\,A} \mathrel{-\!\!*} \mathrm{B} : x \\
| \\
\mathrm{T\,A} : a \\
\mathrm{F\,B} : xa
\end{array}
\qquad\qquad
\begin{array}{c}
\mathrm{T\,A} * \mathrm{B} : x \\
| \\
\boxed{ass : ab \leq x} \\
| \\
\mathrm{T\,A} : a \\
\mathrm{T\,B} : b
\end{array}
$$

in which a (resp. b) is not new if $\mathrm{T\,A} : a$ *(resp.* $\mathrm{T\,B} : b$*) has been already introduced in $\mathcal{B}$ by a previous $\pi\alpha$ tableau expansion, are admissible in* **TBI/MILL***.*

Proof. Let $\mathcal{K}$ be a regular Kripke resource model $\langle \mathcal{M}, \models, [\![-]\!] \rangle$, we only prove the result for $\mathrm{F} \mathrel{-\!\!*}$, the other case being similar. The soundness of the liberalized $\mathrm{F} \mathrel{-\!\!*}$ is justified by the following semantical equivalence : $x \not\models \mathrm{A} \mathrel{-\!\!*} \mathrm{B}$ *iff there exists an* A*-characteristic world a such that* $x \bullet a \not\models \mathrm{B}$.
The if case is obvious. For the only-if case, let us suppose that $x \not\models \mathrm{A} \mathrel{-\!\!*} \mathrm{B}$. Then, there exists y such that $y \models \mathrm{A}$ and $x \bullet y \not\models \mathrm{B}$. Since $y \models \mathrm{A}$, the regularity of $\models$ implies that there exists an A-characteristic world a, which implies $a \sqsubseteq y$ and, by order preservation, $x \bullet a \sqsubseteq x \bullet y$. By Kripke monotonicity we finally have $x \bullet a \not\models \mathrm{B}$.
Now, suppose that we have a realization $\|-\|$ of $\mathcal{B}$ in $\mathcal{K}$ with $\mathrm{T\,A} : a$ and $\mathrm{F\,A} \mathrel{-\!\!*} \mathrm{B} : x$ in $\mathcal{B}$. Then, $\|a\| \models \mathrm{A}$ and $\|x\| \not\models \mathrm{A} \mathrel{-\!\!*} \mathrm{B}$. Since $\|a\|$ is assumed to be A-characteristic, it follows from $\|x\| \not\models \mathrm{A} \mathrel{-\!\!*} \mathrm{B}$ and our previous remarks that $\|x\| \bullet \|a\| (= \|xa\|) \not\models \mathrm{B}$, and then the expansion of a liberalized $\mathrm{F} \mathrel{-\!\!*}$ preserves realizability.

Let us remark that, in the case of $\mathrm{T\,A} * \mathrm{B} : x$, either the constant for A, or the one for B, or both can be reused. The assertion associated to the rule is not discarded. On the contrary, it adds more constraints on the reused constants. Let us also remark that lemma 8.3 implies that, once a $\pi\alpha$ formula has been expanded with some constant a, this constant can be reused for every subsequent occurrence of the same $\pi\alpha$ formula. For example, if we have already expanded $\mathrm{F\,A} \mathrel{-\!\!*} \mathrm{B} : x$ with $\mathrm{T\,A} : a$, $\mathrm{F\,B} : xa$, and we, later on, reach a point where we need to expand $\mathrm{F}\ \mathrm{A} -\!\!* \mathrm{B} : y$, we can simply reuse a to produce $\mathrm{F}\ \mathrm{B} : ya$ instead of introducing a new constant b. Then, by using the same constant for distinct occurrences of the same $\pi\alpha$ formula and distinct constants for distinct $\pi\alpha$ formulas, we can keep track of the expansion process inside the labels since then, the constants composing the label of a signed formula clearly identify the $\pi\alpha$ formulas from which it derives.

As for $\mathbf{BI}^{\perp}$, the semi-decision procedure given in subsection 8.1 can be improved into a decision procedure by taking both the liberalized rules and the notion of redundancy given in definition 4.7 into account.

8.2 Examples

In this section, we illustrate how to build proofs and countermodels in **TBI/MILL**. It is a new tableau method for **MILL** that allows to directly build countermodels in case of non-provability.

Let us illustrate its characteristics with the following examples. Figure 17 gives a tableau for the formula $((\mathrm{p} * \mathrm{q}) -\!\!* \mathrm{r})) -\!\!* (\mathrm{q} -\!\!* (\mathrm{p} -\!\!* \mathrm{r}))$. We obtain a tableau in which the three branches are closed and the formula is therefore provable in **MILL**. The first branch is closed due to the connection $\langle\, \mathrm{T}\ \mathrm{p} : c_3, \mathrm{F}\ \mathrm{p} : c_3 \,\rangle$, the second is closed due to $\langle\, \mathrm{T}\ \mathrm{q} : c_2, \mathrm{F}\ \mathrm{q} : c_2 \,\rangle$ and the third is closed due to $\langle\, \mathrm{T}\ \mathrm{r} : c_1c_2c_3, \mathrm{F}\ \mathrm{r} : c_1c_2c_3 \,\rangle$.

The second example, given in figure 18, gives an open tableau for the formula $((\mathrm{p} * \mathrm{q}) -\!\!* \mathrm{r})) -\!\!* (\mathrm{q} -\!\!* (\mathrm{p} -\!\!* \mathrm{r}))$, which means that it is not provable in **MILL**. Since we have no $\pi\beta$ formulas in the tableau, every formula is immediately fulfilled after it has been decomposed, and we finally obtain an open branch which is therefore also a finite completed branch.

$$
\begin{array}{c}
\surd_1\ \ \mathrm{F}\ \mathrm{p} -\!\!* ((\mathrm{p} * \mathrm{q}) -\!\!* \mathrm{p}) : 1 \\
| \\
\mathrm{T}\ \mathrm{p} : c_1 \\
\surd_2\ \ \mathrm{F}\ (\mathrm{p} * \mathrm{q}) -\!\!* \mathrm{p} : c_1 \\
| \\
\surd_3\ \ \mathrm{T}\ \mathrm{p} * \mathrm{q} : c_2 \\
\mathrm{F}\ \mathrm{p} : c_1c_2 \\
| \\
\boxed{ass_1 : c_3c_4 \leq c_2} \\
| \\
\mathrm{T}\ \mathrm{p} : c_3 \\
\mathrm{T}\ \mathrm{q} : c_4
\end{array}
$$

Figure 18: Labelled Tableau for $\mathrm{p} -\!\!* ((\mathrm{p} * \mathrm{q}) -\!\!* \mathrm{p})$

Let us see how to construct a countermodel from such a completed branch. Such a construction, in the case of **MILL**, exactly follows the steps given for $\mathbf{BI}^{\perp}$ in definition 4.5.

For the example of figure 18, the construction results in the set of worlds.

$$M = \{\, 1, c_1, c_2, c_3, c_4, c_1c_2, c_1c_3, c_1c_4, c_3c_4, c_1c_3c_4, \pi \,\}$$

The preordering $\sqsubseteq$ is defined by $c_3c_4 \sqsubseteq c_2$, $c_1c_3c_4 \sqsubseteq c_1c_2$ and $(\forall m \in M)(m \sqsubseteq \pi)$.

Moreover, the $\bullet$ operation is defined by the following table

$\bullet$	1	c_1	c_2	c_3	c_4	c_1c_2	c_1c_3	c_1c_4	c_3c_4	$c_1c_3c_4$	π
1	1	c_1	c_2	c_3	c_4	c_1c_2	c_1c_3	c_1c_4	c_3c_4	$c_1c_3c_4$	π
c_1	c_1	π	c_1c_2	c_1c_3	c_1c_4	π	π	π	$c_1c_3c_4$	π	π
c_2	c_2	c_1c_2	π	π	π	π	π	π	π	π	π
c_3	c_3	c_1c_3	π	π	c_3c_4	π	π	$c_1c_3c_4$	π	π	π
c_4	c_4	c_1c_4	π	c_3c_4	π	π	$c_1c_3c_4$	π	π	π	π
c_1c_2	c_1c_2	π	π	π	π	π	π	π	π	π	π
c_1c_3	c_1c_3	π	π	π	$c_1c_3c_4$	π	π	π	π	π	π
c_1c_4	c_1c_4	π	π	$c_1c_3c_4$	π	π	π	π	π	π	π
c_3c_4	c_3c_4	$c_1c_3c_4$	π	π	π	π	π	π	π	π	π
$c_1c_3c_4$	$c_1c_3c_4$	π	π	π	π	π	π	π	π	π	π
π	π	π	π	π	π	π	π	π	π	π	π

The forcing relation $\models$ is parametrized by the following atomic interpretation

x	p	q
$[\![\mathrm{x}]\!]$	$\{c_1, c_3, \pi\}$	$\{c_4, \pi\}$

It is routine to check that $c_1 \models \mathrm{p}$ (1) and $c_1 \not\models (\mathrm{p} * \mathrm{q}) \mathrel{-\!*} \mathrm{p}$ (2).
(1) is immediate by definition of the forcing relation. For (2), on one hand, we have that $c_2 \models \mathrm{p} * \mathrm{q}$ because $c_3 \models \mathrm{p}$, $c_4 \models \mathrm{q}$ and $c_3c_4 \sqsubseteq c_2$. On the other hand, we have $c_1c_2 \not\models \mathrm{p}$ because $c_1c_2 \notin [\![\mathrm{p}]\!]$. Therefore, $c_1 \not\models (\mathrm{p} * \mathrm{q}) \mathrel{-\!*} \mathrm{p}$.

In further work, we aim to compare our semantic tableau method with other recent proof-search methods that build countermodels in **ILL** but also to deal with different semantics of **ILL** that are based on monoids, Petri nets or resources [12, 14]. Proof-search in **MILL** has been recently studied with labelled proof nets [3] and also related to connection methods in linear logic [11]. These new (resource) semantics could lead to new definitions of labels and then improve such labelled proof-search methods. Moreover, we will analyze the relationships, mainly from a countermodel construction perspective with the labelled tableau method for **MILL** based on the so-called simple semantics [36]. Both methods emphasize the power of labels to capture semantical interactions between connectives. We could propose some refinements of the semantics from the study of this labelled method that emphasizes the role of requirements and assertions on labels. The requirements, respectively the assertions could be seen as questions, respectively answers that have to interact and could be related to game semantics proposals for linear logic [4]. We will study the actual impact of the choice of different semantics for the countermodel generation process and thus propose new appropriate resource semantics for proof-search with countermodel generation facility in such substructural logics.

8.3 Properties of the MILL labelled calculus

Theorem 8.1 (soundness). *Let* A *be a proposition of* **MILL**, *if there exists a closed tableau sequence for* A *in* **TBI/MILL**, *then* A *is valid in Urquhart semantics.*

Proof. See the proof of theorem 4.1 with restrictions to **MILL**.

Theorem 8.2 (completeness). *Let* A *be a proposition of* **MILL**, *if* A *is valid in Urquhart semantics then, there exists a closed tableau sequence for* A *in* **TBI/MILL**.

Proof. See the proof of theorem 4.2 with restrictions to **MILL**.

Theorem 8.3 (finite model property). *Let* A *be a proposition of* **MILL**, *if* A *is not valid in Urquhart semantics then there exists a finite model in which* A *is not satisfied.*

Proof. See the proof of theorem 4.3 with restrictions to **MILL**.

There have been several attempts to show the finite model property for fragments of some substructural logics, for example, for fragments of intuitionistic linear logic [6, 31] like **MILL**. For instance, the method of logical congruence [23] has been adapted to intuitionistic phase semantics [31]. Here we have deduced a new result that is the finite model property of **MILL** w.r.t. Urquhart semantics, the labels allowing to directly generate countermodels in terms of this "resource semantics". Let us mention that a similar approach has been recently developed with respect to a monoid-based semantics [36].

Both adaptations of $\mathbf{TBI}^{\perp}$ to **IL** and **MILL** illustrate that the use of labelled deductive systems and then the study of appropriate resource semantics are good starting points to study the countermodel generation for substructural and mixed logics like **BI**. Further works will be devoted to the definition of new resource semantics for such logics with completeness proofs by proof-search and generation of usable countermodels [14].

9 Conclusion

In this paper we have designed a labelled tableau calculus for the propositional **BI** logic, more precisely $\mathbf{BI}^{\perp}$. This new logic is a resource-aware logic with an emphasis on interferences between resources that has been recently studied as an assertion language for reasoning about mutable data structures [20] and also for logic programming [2]. It seems to be a well suited logical foundation, in order to specify and prove some particular properties of programs or systems. Here we focus on proof-search in **BI** logic and define a labelled calculus for $\mathbf{BI}^{\perp}$ that yields not only proofs for theorems but also countermodels for non-theorems. The resulting countermodels are finite and thus, in addition to the completeness of the calculus, we have proved the finite model property for $\mathbf{BI}^{\perp}$ w.r.t. the Kripke resource semantics. The proof-search method is mainly based on the **BI** resource semantics and the introduction of specific labels and constraints that capture the essence of this semantics into the proof-search process. Moreover, these concepts allow to generate countermodels in a simple way. Similar work has been recently done for **MILL** with another simple semantics [36].
Moreover, we have adapted, by restriction, our labelled tableau method to **IL** and **MILL** logics, and thus obtained new semantic tableaux methods for both logics with easy countermodel construction. Comparisons with other proof-search methods confirm the power and the usefulness of such labelled calculi tailored from a given resource semantics. In fact, labels and constraints are appropriate for proof-search in such a mixed logic, taking into account the interaction and interference between different kinds of connectives. We have studied the representation and resolution of label constraints during the tableau construction by defining and using dependency graphs from which countermodels can be extracted. We have also recently defined labelled calculi for Mixed Linear Logic in which commutative and non-commutative connectives cohabit [17] and also used labels and dependency graphs as key structures for semantic proof-search. In this context, such structures are also appropriate to capture the interactions between different kinds of connectives. In order to analyze some algorithmic aspects of our semantics method, we have proposed another labelled calculus with free variables in labels. It provides a way to control the introduction of new constants, and thus, to limit the syntactic complexity of the labels. It leads to a better efficiency of the proof-search but to a more difficult generation of countermodels.
Recently, we have studied how to extend these labelled calculi to propositional **BI** (with $\perp$) including the countermodel generation facility. A potential difficulty consists in relating labels with the topological resource models or the Grothendieck sheaf-theoretic models for which completeness of **BI** has been proved [35]. In fact, we have defined a labelled tableau method for **BI**, that extends the presented one and thus to prove the decidability of the propositional **BI** logic and the finite model property w.r.t. the above mentioned semantics [16]. Moreover, we have also recently shown that our approach based on labels and constraints can also be applied to design a connection method for **BI** logic [15]. The study of labels, constraints and dependency graphs will lead to further works in order to define new resource semantics for **BI** and its subsystems. Moreover, we will study such new semantic proof-search methods for a classical version of **BI** that has been proposed as an assertion language for mutable data structures [20] and the possible connections with recent verification formalisms that capture the informal local reasoning in pointer programs [28]. The idea of dealing with labelled formulas is related to the LDS methodology

that has been devised to provide uniform deductive systems [10] for a wide range of substructural logics like linear, relevance and intuitionistic logics [5, 7]. But, so far, the problem of generating countermodels in such frameworks has not been studied. In contrast with this approach, we have designed, starting from a given resource semantics, a specialized labelled tableau framework, for **BI** and its subsystems **IL** and **MILL**, based on particular labels and constraints for which the countermodel construction is an integrated part of the proof-search process. An interesting point will be to study, in other logics, the relationships resource semantics and corresponding labelled proof-search methods and, more precisely, if our approach could be applied or generalized to other logics for which well-defined resource semantics are defined.

References

[1] J.M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[2] P.A. Armelin and D. Pym. Bunched Logic Programming (extended abstract). In *First International Joint Conference on Automated Reasoning, IJCAR 2001, LNCS 2083*, pages 289–304, Siena, Italy, 2001.

[3] V. Balat and D. Galmiche. *Labelled Deduction*, volume 17 of *Applied Logic Series*, chapter Labelled Proof Systems for Intuitionistic Provability. Kluwer Academic Publishers, 2000.

[4] A. Blass. A Game Semantics for Linear Logic. *Annals of Pure and Applied Logic*, 56:183–220, 1992.

[5] K. Broda, M. Finger, and A. Russo. Labelled Natural Deduction for Substructural Logics. *Logic Journal of the IGPL*, 7(3):283–318, 1999.

[6] W. Buszkowski. The finite model property for BCI and related systems. *Studia Logica*, 57:303–323, 1996.

[7] M. D'Agostino and D.M. Gabbay. A Generalization of Analytic Deduction via Labelled Deductive Systems. Part I: Basic substructural logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

[8] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57:795–807, 1992.

[9] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer Verlag, 1990.

[10] D.M. Gabbay. *Labelled Deductive Systems, Volume I - Foundations.* Oxford University Press, 1996.

[11] D. Galmiche. Connection Methods in Linear Logic and Proof nets Construction. *Theoretical Computer Science*, 232(1-2):231–272, 2000.

[12] D. Galmiche and D. Larchey-Wendling. Provability in Intuitionistic Linear Logic from a New Interpretation on Petri Nets - extended abstract. *Electronic Notes in Theoretical Computer Science*, 17, 1998.

[13] D. Galmiche and D. Larchey-Wendling. Structural Sharing and Efficient Proof-search in Propositional Intuitionistic Logic. In *Asian Computing Science Conference, ASIAN'99, LNCS 1742*, pages 101–112, Phuket, Thailand, December 1999.

[14] D. Galmiche and D. Larchey-Wendling. Quantales as completions of ordered monoids: revised semantics for Intuitionistic Linear Logic. *Electronic Notes in Theoretical Computer Science*, 35, 2000.

[15] D. Galmiche and D. Méry. Connection-based proof search in propositional BI logic. In *18th Int. Conference on Automated Deduction, CADE-18, LNAI 2392*, pages 111–128, 2002. Copenhagen, Danemark.

[16] D. Galmiche, D. Méry, and D. Pym. Resource Tableaux (extended abstract). In *16th Int. Workshop on Computer Science Logic, CSL 2002, LNCS 2471*, pages 183–199, September 2002. Edinburgh, Scotland.

[17] D. Galmiche and J.M. Notin. Calculi with Dependency Relations for Mixed Linear Logic. In *International Workshop on Logic and Complexity in Computer Science, LCCS'2001*, pages 81–102, Créteil, France, 2001.

[18] D. Galmiche and G. Perrier. On Proof Normalization in Linear Logic. *Theoretical Computer Science*, 135(1):67–110, 1994.

[19] J.Y. Girard. Linear Logic: its Syntax and Semantics. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995.

[20] S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *28th ACM Symposium on Principles of Programming Languages, POPL 2001*, pages 14–26, London, UK, 2001.

[21] S. Ishtiaq and D. Pym. Kripke resources models of a dependently-typed bunched λ-calculus. In *13th Int. Workshop on Computer Science Logic, CSL'99, LNCS 1683*, pages 235–249, Madrid, Spain, 1999. September.

[22] C. Kreitz, H. Mantel, J. Otten, and S. Schmitt. Connection-based proof construction in Linear Logic. In *14th Int. Conference on Automated Deduction, CADE-14, LNCS 1249*, pages 207–221, Townsville, North Queensland, Australia, 1997.

[23] Y. Lafont. The finite model property for various fragments of linear logic. *Journal of Symbolic Logic*, 62(4):1202–1208, 1997.

[24] D. Larchey-Wendling, D. Méry, and D. Galmiche. STRIP: Structural sharing for efficient proof-search. In *First International Joint Conference on Automated Reasoning, IJCAR 2001, LNCS 2083*, pages 696–700, Siena, Italy, 2001.

[25] H. Mantel and J. Otten. linTAP : A Tableau Prover for Linear Logic. In *Int. Conference on Analytic Tableaux and Related Methods, TABLEAUX'99, LNCS 1617*, pages 216–231, Saratoga Springs, NY, USA, 1999.

[26] R.K. Meyer, M.A. McRobbie, and N. Belnap. Linear Analytic Tableaux. In *4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, LNCS 918*, pages 278–293, St Goar am Rhein, Germany, 1995.

[27] P. O'Hearn. Resource interpretations, bunched implications and the $\alpha\lambda$-calculus. In *4th Int. Conference on Typed Lambda Calculi and Applications, TLCA'99, LNCS 1581*, pages 258–279, L'Aquila, Italy, 1999.

[28] P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *15th Int. Workshop on Computer Science Logic, CSL 2001, LNCS 2142*, pages 1–19, Paris, France, 2001.

[29] P.W. O'Hearn and D. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.

[30] M. Okada and K. Terui. Completeness proofs for linear logic based on proof search method (preliminary report). In *Type theory and its applications to computer systems*, pages 57–75, RIMS, Kyoto University, 1998.

[31] M. Okada and K. Terui. The finite model property for various fragments of intuitionistic linear logic. *Journal of Symbolic Logic*, 64(2):790–802, 1999.

[32] H. Ono. Decidability and Finite Model Property of Substructural Logics. In *The Tbilisi Symposium on Logic, Language and Computation*, pages 262–274. CSLI Publications, 1997.

[33] L. Pinto and R. Dyckhoff. Loop-free construction of counter-models for intuitionistic propositional logic. In Behara and al., editors, *Symposia Gaussiana*, pages 225–232, 1995.

[34] D. Pym. On Bunched Predicate Logic. In *14h Symposium on Logic in Computer Science*, pages 183–192, Trento, Italy, July 1999. IEEE Computer Society Press.

[35] D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002.

[36] K. Terui. Labelled tableau calculi generating simple models for substructural logics. Technical report, ILLC publications, University of Amsterdam, 1999.

[37] A. Urquhart. Semantics for Relevant Logic. *Journal of Symbolic Logic*, 37:159–169, 1972.

[38] A. Voronkov. Proof-search in intuitionistic logic based on constraint satisfaction. In *5th Int. Workshop on Theorem Proving with Analytic Tableaux and Related Methods, LNAI 1071*, pages 312–327, Terrasini, Italy, May 1996.