Automated Reasoning Techniques as Proof-search in Sequent Calculus

Mahfuza Farooque

École Polytechnique

Thèse de Doctorat

Spécialité **Informatique**

Automated reasoning techniques as proof-search in sequent calculus

Thèse présentée par

Mahfuza Farooque

au jury composé de

Sylvain Conchon	Rapporteur
Stéphane Graham-Lengrand	Directeur de thèse
Chuck C. Liang	Examinateur
Assia Mahboubi	Examinateur
Dale Miller	Examinateur
Aaron Stump	Rapporteur
Benjamin Werner	Président

Soutenue le 19 Décembre 2013

To my parents, siblings and my friends
(who always inspire me in my hard time)

Abstract

This thesis designs a theoretical and general framework where proof-search can modularly interact with domain-specific procedure(s). This framework is a focussed sequent calculus for polarised classical logic, with quantifiers, and it is designed in the view of capturing various computer-aided reasoning techniques that exist in logic programing, goal-directed systems, proof-assistants, and automated provers.

This thesis starts with a survey of focused sequent calculi for polarised classical logic, as a journey from Gentzen's original sequent calculus to Liang-Miller's LKF.

It then introduces a new system $\mathsf{LK}^p(\mathcal{T})$ which extends, with on-the-fly polarisation and calls to decision procedures, Liang-Miller's LKF. Key results of the meta-theory of the system are proved: the cut-elimination property, the property that changing the polarity of connectives does not change the provability of formulae, and finally, the logical completeness of $\mathsf{LK}^p(\mathcal{T})$. While Gentzen's original rules offer a lot of non-determinism in proof-search, focusing provides a tight control on the breadth of the search space. Together with on-the-fly polarisation of literals, this equips the sequent calculus $\mathsf{LK}^p(\mathcal{T})$ with features that are particularly appropriate for computer-aided reasoning:

For instance, a widely-used technique for solving propositional satisfiability (SAT) problems (whether or not a formula over Boolean variables can be made true by choosing truth values for its variables) is the DPLL procedure. Satisfiability-modulo-Theories (SMT) problems generalise SAT problems by the presence of a background theory for which a decision procedure is known, and can be solved by correspondingly generalising DPLL into $\mathsf{DPLL}(\mathcal{T})$, which most SMT-solvers implement.

This thesis investigates how each of the steps of $\mathsf{DPLL}(\mathcal{T})$ can be emulated as the standard steps of proof-search in $\mathsf{LK}^p(\mathcal{T})$: the gradual and goal-directed construction of a proof-tree. This allows the $\mathsf{DPLL}(\mathcal{T})$ algorithm to be applied up-to-a-point, where a switch to another technique can be made (depending on the newly generated goals). This differs from previous work where an SMT-technique is called until it finishes.

The proof-search control that is provided by focusing and on-the-fly polarisation allows us to derive a stronger result than the mere simulation of $\mathsf{DPLL}(\mathcal{T})$: the proofs in $\mathsf{LK}^p(\mathcal{T})$ that are the images of those $\mathsf{DPLL}(\mathcal{T})$ runs concluding that a formula is unsatisfiable, can be characterised by a simple criterion only involving the way formulae are placed into the focus of sequents (the device implementing focusing). From this criterion we directly get a simple proof-search strategy that is bi-similar to $\mathsf{DPLL}(\mathcal{T})$ runs: that which performs the depth-first completion of incomplete proof-trees (starting with the leftmost open leaf), using any inference steps satisfying the given criterion on polarities and focusing. That way, we ensure that bottom-up proof-search in $\mathsf{LK}^p(\mathcal{T})$ can be as efficient as the $\mathsf{DPLL}(\mathcal{T})$ procedure.

Finally, clause and connection tableaux are other widely used techniques of automated reasoning, of a rather different nature from that of DPLL. This thesis also described how such tableaux techniques can be described as bottom-up proof-search in $\mathsf{LK}^p(\mathcal{T})$. The simulation is given for both propositional and first-order logic, opening up new perspectives of generalisation and collaboration between tableaux techniques and DPLL, even in presence of a background theory.

Contents

Chapter 1

Introduction

Proofs play a central role in computer science; they are used to certify the correctness of both software and hardware behaviour. Structural proof theory is a branch of mathematical logic that considers mathematical proofs as objects of study and investigates the structure of such objects in the prospect, for instance, of building semantics for them. We seek to derive logical properties from the analysis of the structure of proofs.

In this thesis we investigate the notions of proofs that are given by systems of inference rules, which are used to prove a new proposition using previously proved ones. Proofs are then presented as trees, where each node, together with its children, forms a valid instance of some inference rule. A natural process of proof-search is then specified by the gradual construction of proof-trees, starting from the formula to be proved, and trying to apply inference rules bottom-up until a full tree is constructed.

A central question is what kind of inference systems are best appropriate to specify interesting or efficient proof-search procedures. Sequent calculi are such systems, with an ambivalent role: they both define the notion of provability for a logic as well as specify reasonable proof-search procedures.

However, Gentzen's original sequent calculus has major defects for proof-search: the freedom with which rules can be applied leads to a very broad and redundant search space. It needs to be refined, or controlled, to produce reasonable proof-search procedures.

Structural proof theory and its semantics have evolved the concepts of *polarities* and *focusing*, which greatly impact the way we understand the proof-search mechanisms: Miller et al. introduce a notion of *uniform proofs* [MNPS91] which is used to extend the concepts of *logic programing* beyond the logical fragment of Horn clauses, and uniform proofs themselves can be generalised as the concept of *focusing*, which gives sequent calculus proof enough structure to specify reasonable proof-search procedures in linear [And92], intuitionistic, and classical logic [LM09].

More generally, *focused sequent calculi* can be used to describe *goal-directed proof-search* -the foundational paradigm of a broad range of tools from logic programming to higher-order proof-assistants. Indeed, goal-directed proof-search is used in type theories [Bar92] for *type inhabitation* and *higher-order* unification, and in [LDM11] it is shown that these mechanisms can be specified by the focusing mechanism of sequent calculus. This is used as the foundation for the new proof contraction engine in Coq 8.4 [Coq], a proof assistant for *Interactive Theorem Proving.*

Focusing in sequent calculus thus seem to be a versatile concept for specifying proof-search. Our aim is to propose focused sequent calculi as a theoretical and general framework to capture various computer-aided reasoning techniques from logic programming, goal-directed systems, proof-assistants, and automated provers. The hope is to open up the possibility of generalising such techniques to more expressive fragment, making them collaborate, and provide trusted way to implement them.

In this thesis, we investigate how a focused sequent calculus can be used to capture techniques from automated reasoning.

The main area tackled by this thesis is SAT and SMT-solving:

SAT stands for propositional satisfiability, where we seek to determine whether a (quantifier-free) formula over Boolean variables, usually given in Conjunctive Normal Form, is true by choosing true/false values for its variables. SAT-solvers are often based on the Davis, Putnam, Logemann and Loveland (DPLL) procedure [DP60, DLL62].

SMT-problems generalise propositional SAT problems, as they are concerned with the satisfiability of (quantifier-free) formulae over atomic propositions from a theory such as linear arithmetic or bit vectors. Given a procedure deciding the consistency with respect to such a theory of a conjunction of atoms or negated atoms, SMT-solving organises a cooperation between this procedure and SAT-solving techniques, thus providing a decision procedure for SMT-problems. This smart extension of successful SAT-solving techniques opened a prolific area of research and led to the implementation of ever-improving tools, namely SMT-solvers, now crucial to a number of applications in software verification.

The architecture of SMT-solvers is based on an extension of the DPLL procedure, called DPLL($\mathcal{T}$) [NOT06], that supports the integration of the theory-specific decision procedure, therefore addressing SMT-problems.

The second area of automated reasoning tackled in this thesis is that of *Tableaux calculi*. The popularity of tableaux calculi is in large part due to its successful practical and theoretical implementation in computer science [RV01]. These calculi are used in automated and interactive theorem provers to be used in software verification fields.

Tableaux calculi are usually designed as a refutational method, i.e. considering how the negation of what we want to prove could have a model, and the main and common idea of tableaux calculi is to construct a finitely branching tree of possibilities (some kind of case analysis) for such a model to exist. Beyond this, there are many variants of tableaux calculus: clause tableaux, connection tableaux, hyper tableaux, matrices, mating, model elimination, model generation and so on. Among them, we have considered clause tableaux to implement in our focused sequent calculus, as well as restrictions of clause tableaux that significantly reduce the search space: weak connection tableaux and strong connection tableaux. Connection tableaux also form a notion of goal-directed proof-search, like other computer-aided reasoning techniques that have been previosuly described as proof-search in a focused sequent calculus. They are therefore a natural candidate for our general methodology. In propositional logic, it is easy enough to build a correspondence between clause or connection tableaux and some proof-search procedure for a focused sequent calculus; but it is more difficult to do it in first-order logic. We also consider a simulation of *clause tableaux modulo theories* [Tin07] in sequent calculus. This simulation is an opportunity to relate DPLL($\mathcal{T}$) and clause tableaux modulo theories.

The structure of this thesis is outlined below:

- Chapter 2 presents a survey of the history of focused sequent calculus for polarised classical logic. We present Gentzen's sequent calculus LK and a popular variant $\mathbf{G3}_c$. We discuss the cut-elimination procedure and its non-deterministic nature in Gentzen sequent calculus, mostly due to the structural rules of *contraction* and *weakening*. We then give an overview of two contributions by Girard: linear logic [Gir87] on the one hand, and a refinement of Gentzen's sequent calculus for classical logic on the other hand [Gir91]. Indeed, as it is difficult to see constructive aspects in LK, Girard introduced the concept of *polarity* in classical sequent calculus and refined LK into a new system LC. Linear logic, on the other hand, is the framework in which Andreoli introduced the key concept of *focusing* and its impact on proof-search and logic programming [And92]. We review this, and finally how Liang and Miller [LM09] eventually imported those concept in a sequent calculus LKF for polarised classical logic, with the view of proof-search in mind.

  In brief, this chapter discusses the above systems, their proof-search mechanisms and their cut-elimination procedures.

- Chapter 3 introduces a new system $\mathsf{LK}^p(\mathcal{T})$ which extends system LKF [LM09] to a background theory. The novelty of this system is that it can *call a decision procedure* for that theory, and integrate those calls to the natural proof-search procedure of sequent calculus. This system also allows proof-search to set the polarities of literals *on-the-fly*. The calls to the decision procedure jeopardise the elimination of cuts, but we identify a condition on sequents, called *safety*, that guarantees that cuts deriving safe sequents are indeed admissible in $\mathsf{LK}^p(\mathcal{T})$. Besides the cut-elimination procedure of $\mathsf{LK}^p(\mathcal{T})$, we also show that the system is complete. In presence of a background theory, changing the polarities of literals affects the provability of formulae (in contrast to what happens with the empty theory). Yet, changing the polarities of connectives does not affect the provability of formulae. To prove this we present a system $\mathsf{LK}^+(\mathcal{T})$ which is a slightly relaxed version of $\mathsf{LK}^p(\mathcal{T})$.
- Chapter 4 then shows how the main technique for SMT-solving, namely $\mathsf{DPLL}(\mathcal{T})$, can be simulated in the focused sequent calculus $\mathsf{LK}^p(\mathcal{T})$ for polarised classical logic modulo theories.

  Our work does not try to improve the $\mathsf{DPLL}(\mathcal{T})$ technique itself, or current SMT-solvers based on it, but makes a step towards the integration of the technique into a sequent calculus framework.

  A now wide literature achieves the integration of SMT-tools in various frameworks using the blackbox approach. For instance, several proof assistants propose an infrastructure allowing the user to call an external SMT-solver as a blackbox and re-interpret its output to reconstruct a proof within the system [Web11, AFG$^+$11, BCP11, BBP11].

  Here, we aim at a new and deeper integration where $\mathsf{DPLL}(\mathcal{T})$ is performed within the system. Recently, an internal implementation of some SMT-techniques was made available in the Coq proof assistant [LC09], but this implement remains specific to Coq's *reflection* feature [Bou97] (and therefore can hardly be adapted to a framework without reflection).

  We rather investigate a broader and more basic context where we can perform each of the *steps* of $\mathsf{DPLL}(\mathcal{T})$ as the standard steps of proof-search in sequent calculus: the gradual and goal-directed construction of a proof-tree. This allows the $\mathsf{DPLL}(\mathcal{T})$ algorithm to be applied *up-to-a-point*, where a switch to another technique can be made (depending on the newly generated goals), whereas the use of *reflection* or of a blackbox call only works when the entire goal can be treated by a run of $\mathsf{DPLL}(\mathcal{T})$.

  More precisely, we identify an elementary version of $\mathsf{DPLL}(\mathcal{T})$ that is the direct extension of the *Classical DPLL procedure* to a background theory $\mathcal{T}$, as well as being a restriction of the *Abstract DPLL Modulo Theories* system, that allows more advanced features such as *backjumping* both of which can be found in [NOT06].

  We present the simulation of $\mathsf{DPLL}(\mathcal{T})$ into $\mathsf{LK}^p(\mathcal{T})$ in two ways: (i) an indirect simulation and (ii) a direct simulation.

  The indirect simulation splits the simulation of $\mathsf{DPLL}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$ into two parts: First we show a simulation of $\mathsf{DPLL}(\mathcal{T})$ into an intermediate inference system introduced by Tinelli [Tin02]; and then we make a simulation of that system into $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

  The direct simulation of $\mathsf{DPLL}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$ allows us to derive a stronger result than the mere simulation of $\mathsf{DPLL}(\mathcal{T})$: The proofs in $\mathsf{LK}^p(\mathcal{T})$ that are the images of *Elementary* $\mathsf{DPLL}(\mathcal{T})$ runs finishing on UNSAT, can be characterised by a simple criterion only involving the way polarities are assigned to literals, and the way formulae are placed into the *focus* of sequents (the device implementing focusing). From this criterion we directly obtain a simple proof-search strategy that is bisimilar to Elementary $\mathsf{DPLL}(\mathcal{T})$ runs: That which performs the depth-first completion of incomplete proof-trees (starting with the leftmost open leaf), using any inference steps satisfying the given criterion on polarities and focusing. In this way, we ensure that bottom-up proof-search in sequent calculus can be as efficient as the (Elementary) $\mathsf{DPLL}(\mathcal{T})$ procedure. Beyond Elementary $\mathsf{DPLL}(\mathcal{T})$, we also show the simulation of the standard $\mathsf{DPLL}(\mathcal{T})$ procedure into $\mathsf{LK}^p(\mathcal{T})$, but with weaker results than with Elementary $\mathsf{DPLL}(\mathcal{T})$.
- Lastly, Chapter 5 presents a simulation of clause and connection tableaux into $\mathsf{LK}^p(\mathcal{T})$ for both propositional and first-order logic. Moreover, we also present a simulation of clause tableaux

modulo theories [Tin07] into $\mathsf{LK}^p(\mathcal{T})$. In the propositional case, we present an isomorphic image of clause tableaux, clause tableaux modulo theories and connection tableaux into sequent calculus.

Chapter 2

Focusing Gentzen's sequent calculus for classical logic: a survey

Classical sequent calculus LK was first introduced by Gentzen in 1934. The main purpose of this chapter is to investigate how classical sequent calculus can be modified and constructed with the features of *focus* and *polarity*. We therefore make a survey from Gentzen's classical sequent calculus LK to the focused sequent calculus LKF which is introduced by Liang and Miller [LM09]. To make a focused sequent calculus for classical logic, linear logic plays an important role. Therefore we also present linear logic and linear sequent calculus[Gir87].

This chapter is presented with the following sections: Section 2.1 presents Gentzen's LK system and one of its variants $\mathbf{G3}_c$. In Section 2.2 we introduce linear logic and its sequent calculus LL; then we present Girard's system LC for classical logic, where the concept of polarity is first introduced. Section 2.3 presents the focused sequent calculus LLF for linear logic. Section 2.4 presents system LKF, a version of LLF for classical logic. For each system we discuss cut-elimination and its properties.

2.1 Gentzen's sequent calculus

Gentzen [Gen35] introduced the sequent calculus as a formalism of mathematical logic and reasoning, and sequent calculus is now a class of deduction systems. Perhaps the most well-known one is for classical first-order logic, but the notation and general principles are useful for many different logics and type theories, etc. We have already mentioned that proofs are labelled finite trees with a single root, with axioms at the top nodes, and each node-label connected with the labels of the successor nodes (if any) according to one of the rules. In (a bi-sided) sequent calculus, rules are divided into *left*-(L) and *right*-(R) rules. For any logical operator $*$, $L*$ (resp. $R*$) indicates the rules where a formula with $*$ as main operator as introduced on the left (resp. right). Gentzen's original system for classical logic is called the LK system which is presented in Figure 2.1.

In this figure, a denotes an atomic formula and the notation $\{{}^{t}\!/\!_{x}\}A$ represents the (capture-avoiding) substitution of a (first-order) term t for x in the (first-order) formula A. Moreover, the left-hand side and the right-hand side of a sequent are here lists of formulae (hence the presence of the exchange rules $(LExc)$ and $(RExc)$).

Remark 1 It is useful to see that we could have used alternative rules for $(L\wedge)$ and $(R\vee)$, namely the four following rules (where i is 1 or 2):

$$\frac{\Gamma, A_i \vdash \Delta}{\Gamma, A_1 \wedge A_2 \vdash \Delta}\,(L_i\wedge) \qquad \frac{\Gamma \vdash A_i, \Delta}{\Gamma \vdash A_1 \vee A_2, \Delta}\,(R_i\vee)$$

Identity axiom

$$\frac{}{a \vdash a}\,(Ax)$$

Negation Rules

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\,(L\neg) \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\,(R\neg)$$

Logical Rules

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,(L\wedge) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\,(R\wedge)$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\,(L\vee) \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\,(R\vee)$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\,(L \Rightarrow) \qquad \frac{A, \Gamma \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\,(R \Rightarrow)$$

$$\frac{\Gamma, \{{}^{t}\!/\!_{x}\} A \vdash \Delta}{\Gamma, \forall x A \vdash \Delta}\,(L\forall) \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x A, \Delta}\,(R\forall)^{*}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x A \vdash \Delta}\,(L\exists)^{*} \qquad \frac{\Gamma \vdash \{{}^{t}\!/\!_{x}\} A, \Delta}{\Gamma \vdash \exists x A, \Delta}\,(R\exists)$$

Structural Rules

$$\frac{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}\,(LExc) \qquad \frac{\Gamma \vdash \Delta_1, B, A, \Delta_2}{\Gamma \vdash \Delta_1, A, B, \Delta_2}\,(RExc)$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,(LW) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\,(RW)$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\,(LC) \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\,(RW)$$

$(^{*})$ x is not free in Γ, Δ

Figure 2.1: The LK system for classical logic

The connectives defined with the above rules are provably equivalent to those defined by the rules of Fig. 2.1, and the proofs of those equivalences critically rely on the structural rules.

Variants of Gentzen's sequent calculus (system LK) for classical logic are presented in [TS00]. We will present here a popular variant: $\mathbf{G3}_c$, where structural rules are absorbed into the logical rules.

G3$_c$ system

We now look at the **G3$_c$** system.

Definition 1 (G3$_c$ system) Sequents of **G3$_c$** system are of the form $\Gamma \vdash \Delta$. Here, Γ and Δ are finite multisets of formulae, and are called the *context*: more precisely, Γ is called the *antecedent* and Δ is called the *succedent* of a sequent. The formula that appears explicitly in the conclusion of each rule is called the *principal formula* or *main formula*. The formulae in the premises from which the principal formula is derived are the *active formulae*. The other formulae are called *side formulae*. The **G3$_c$** system is defined in Figure 2.2, where a denotes an atomic formula.[1] ※

$$\frac{}{a, \Gamma \vdash a, \Delta}\,(Ax) \qquad \frac{}{\Gamma, \bot \vdash \Delta}\,(L\bot)$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,(L\wedge) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\,(R\wedge)$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\,(L\wedge) \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\,(R\wedge)$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\,(L \Rightarrow) \qquad \frac{A, \Gamma \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\,(R \Rightarrow)$$

$$\frac{\forall x A, \Gamma, \{{}^{t}\!/\!_{x}\} A \vdash \Delta}{\Gamma, \forall x A \vdash \Delta}\,(L\forall) \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x A, \Delta}\,(R\forall)^{*}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x A \vdash \Delta}\,(L\exists)^{*} \qquad \frac{\Gamma \vdash \{{}^{t}\!/\!_{x}\} A, \Delta, \exists x A}{\Gamma \vdash \exists x A, \Delta}\,(R\exists)$$

$(^{*})$ x is not free in Γ, Δ

Figure 2.2: The **G3$_c$** system for classical logic

Gentzen's original system for classical logic has structural rules, but in the **G3$_c$** system, the structural rules of weakening and contraction are "absorbed" with the rules and axioms. Indeed, these rules are admissible in this system:

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

In order to prove the completeness of a sequent calculus, it is often convenient to extend it with one or several *cut* rule(s) such as:

$$\frac{\Gamma \vdash A, \Delta \quad A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,(cut) \text{ or } \frac{\Gamma \vdash A, \Delta \quad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$

and then prove that every sequent that has a proof using that rule also has a proof not using it (see the next subsection).

Cut-rules express a form of transitivity of $\vdash$, and can be seen as general forms of a lemma: it says that when a formula A is in the succedent of a proved sequent and in the antecedent of another proved sequent, then by "cutting out" the formula A (which is called the *cut-formula*) we build a proof of the remaining context; in other words, to prove $\Gamma \vdash \Delta$, we first prove A as a lemma, and then continue the proof with A as an extra assumption.

[1]In rule Ax, both occurrences of a are principal; in $L\bot$ the occurrence of $\bot$ is principal.

But an important property of the rules of Fig. 2.2 is the sub-formula property: every formula in the premises of a rule is the sub-formula of some formula in the conclusion of the rule. This is not the case of cuts, as it is not necessary that the cut-formula A in the premises is a sub-formula of a formula in Γ, Δ. This is problematic for bottom-up proof-search, as the formula A has to be guessed (and there are infinitely many possibilities).

Cut-elimination

However, a major theorem of sequent calculus is the *Hauptsatz* [Gen35] or cut-elimination theorem. This theorem says that cuts are *admissible*; i.e. every formula that is provable using cuts, is also provable without cuts.

In [TS00], the cut-elimination proof is based on certain local transformation steps, defining an algorithm that permutes for instance cut-rules upwards, over the other rules, or replaces a cut on a compound formula A by some cuts on its immediate sub-formulae.

Notice that the permutation of cuts over other cuts may lead to termination problems, as illustrated below:

$$\dfrac{\dfrac{\Gamma \vdash \Delta, A \qquad \Gamma' \vdash \Delta', B}{\Gamma, \Gamma' \vdash \Delta, \Delta', B} \qquad B, \Gamma''' \vdash \Delta''}{\Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}$$

reduces to

$$\dfrac{\Gamma \vdash \Delta, A \qquad \dfrac{\Gamma' \vdash \Delta', B \qquad B, \Gamma'' \vdash \Delta'}{A, \Gamma, \Gamma' \vdash \Delta', \Delta''}}{\Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}$$

However, by either forbidding a cut to permute over another cut, or by carefully controlling when that happens, it is possible to make the cut-elimination procedure strongly normalising.

Another question is confluence: The design of a cut-elimination process by local rewrite steps raises the question of whether, when permuting a cut upwards, we should start pushing it into its left-hand-side branch first or into its right-hand-side branch first. The following example is due to Lafont:

Consider two distinct (cut-free) proofs, π_1 and π_2, of the sequent $\Gamma \vdash \Delta$. Since weakenings are admissible, we can easily get two proofs π_1' and π_2' of $\Gamma \vdash \Delta, A$ and $\Gamma, A \vdash \Delta$, respectively. In essence, these are the same as π_1 and π_2, but with an extra formula that is added to every sequent of the proof-tree, not participating to any of the inference rules. Cutting them gives a new proof of $\Gamma \vdash \Delta$:

$$\dfrac{\dfrac{\pi_1'}{\Gamma \vdash \Delta, A} \qquad \dfrac{\pi_2'}{\Gamma, A \vdash \Delta}}{\Gamma \vdash \Delta}$$

Eliminating that cut can be done in two ways: pushing the cut to the left eventually yields π_1, erasing π_2, while pushing the cut to the right eventually yields π_2, erasing π_1.

This shows that, in general, this kind of cut-elimination in classical logic is not confluent, as the complete duality illustrated by De Morgan's laws leads to a symmetry that gives us no particular reason to give priority to the left or to the right.

A similar example (actually more problematic than the one above) can be build with contractions: Consider two distinct (cut-free) proofs, π_1 and π_2, of the sequents $\Gamma \vdash \Delta, A, A$ and $\Gamma, A, A \vdash \Delta$, respectively. Since contractions are admissible, we can easily get two proofs π_1' and π_2' of $\Gamma \vdash \Delta, A$ and $\Gamma, A \vdash \Delta$, respectively. Cutting them gives a proof of $\Gamma \vdash \Delta$:

$$\dfrac{\dfrac{\pi_1'}{\Gamma \vdash \Delta, A} \qquad \dfrac{\pi_2'}{\Gamma, A \vdash \Delta}}{\Gamma \vdash \Delta}$$

Eliminating that cut can again be done in two ways: pushing the cut to the left duplicates π_2, while pushing the cut to the right duplicates π_1.

2.2 Linear logic and the sequent calculus **LC** for classical logic

In this section, we discuss *linear logic* and a sequent calculus for it [Gir87]; then we present a *polarised* sequent calculus *LC* for classical logic, inspired by linear logic.

2.2.1 Linear logic and its sequent calculus

Linear logic is a refinement of e.g. classical and intuitionistic logic, with new connectives that can be used to decompose the usual connectives. This logic plays an important role in modern computer science and engineering. Now, we give a brief idea of linear logic and then we present a sequent calculus LL.

An important feature of linear logic is a *duality* between connectives and formulae, similar to the De Morgan duality of classical logic. As in classical logic, a great role is played by the involutive negation function, here called *linear negation* and denoted $(\cdot)^{\perp}$; we use it to internalise *De Morgan laws* for all connectives and quantifiers. For instance, $\exists x A$ is the dual of $\forall x A^{\perp}$ and $A = A^{\perp\perp}$.

The main feature of linear logic is the controlled use of the structural rules of weakening and contraction. Girard criticised these structural rules from Gentzen's sequent calculus in [Gir95], because weakening generates "fake dependencies", for instance when proving $A \Rightarrow (B \vee \neg B)$. Hence the introduction of a *linear implication* $\multimap$, where the premise must be "used" exactly once.

Restricting the structural rules of weakening and contraction leads to breaking the property described in Remark 1: the choice of inference rules for the conjunction and disjunction becomes critical, and the two versions then lead to two different connectives that are not equivalent.

Therefore, linear logic presents two disjunctions: $\oplus$ (plus) and $\parr$ (par). Intuitively, a $\oplus$ disjunction is proved by proving one (and only one) of its two sides (as in intuitionistic logic), whereas a $\parr$ disjunction can be proved by "keeping" both sub-formulae and possibly relate them, just like $A \vee \neg A$ can be proved in classical logic. Moreover, similarly to classical logic, an implication $A \multimap B$ can be seen as an abbreviation for $(A)^{\perp} \parr B$.

Dually, there are also two conjunctions in linear logic: $\otimes$ (times) and & (with). The former is the dual of $\parr$ and the latter is the dual of $\oplus$.

This duality plays an important role in the proof of cut-elimination, which also hold in linear logic (see next section) and whose details give an intuition about the connectives: Both conjunctions express the "availability of two possibilities", but in the case of &, only one of the possibilities will be used during cut-elimination (as it will face a $\oplus$ disjunction), and in the case of $\otimes$, both possibilities will be used.

Linear quantifier $\forall$ is close to & and $\exists$ is close to $\oplus$.

Now, weakening and contractions are not completely ruled out of the system, they are *controlled* by two new connectives: ! and ?, that are called *exponentials*.

Girard also presents a sequent calculus LL for linear logic [Gir87]. The preliminaries for LL and the system are given below:

Definition 2 (Formulae, negation)

The notion of *atoms* from Gentzen's sequent calculus is here enriched as the notion of *literals*: atoms equipped with an involutive *negation* function mapping a literal a to a literal $a^\perp$.
Formulae of LL are given by the following grammar:

$$A, B, \ldots := \ a \mid 1 \mid \perp \mid \top \mid 0 \mid A \otimes B \mid A \parr B \mid A \& B \mid A \oplus B \mid \forall x A \mid \exists x A \mid !A \mid ?A$$

where a ranges over literals.
Negation is then extended by De Morgan Laws, and linear implication is also defined as connective:

$$\begin{array}{lcl@{\qquad\qquad}lcl}
1^\perp & := & \perp & \perp^\perp & := & 1 \\
\top^\perp & := & 0 & 0^\perp & := & \top \\
(A \otimes B)^\perp & := & A^\perp \parr B^\perp & (A \parr B)^\perp & := & A^\perp \otimes B^\perp \\
(A \& B)^\perp & := & A^\perp \oplus B^\perp & (A \oplus B)^\perp & := & A^\perp \& B^\perp \\
(!A)^\perp & := & ?A^\perp & (?A)^\perp & := & !A^\perp \\
(\forall x A)^\perp & := & \exists x A^\perp & (\exists x A)^\perp & := & \forall x A^\perp
\end{array}$$

$$A \multimap B := A^\perp \parr B$$

※

Definition 3 (LL system)

We use the De Morgan duality to avoid the redundancy of having both the left-introduction rule(s) for a connective and the right-introduction rule(s) for the dual connective:[2]
The sequent calculus *LL* is *mono-sided*, with sequents of the form $\vdash \Delta$ where Δ is a multiset of formulae. Its rules are given in Figure 2.3. ※

Linear connectives are presented in three categories:

- Multiplicative connectives: The connectives $\otimes$, $\parr$, $\multimap$, together with the neutral elements 1 (w.r.t. $\otimes$) and $\perp$ (w.r.t. $\parr$) are called *multiplicative connectives*.
- Additives connectives: The connectives & and $\oplus$, together with the neutral elements $\top$ (w.r.t &) and 0 (w.r.t $\oplus$) are called *additive connectives*.
- Exponential connectives: The connectives ! and ? are called *exponential connectives*, and are also called *linear modalities*.

It is interesting to note that:

- $\otimes$ is multiplicative and conjunctive with neutral 1,
- $\oplus$ is additive and disjunctive with neutral 0,
- $\parr$ is disjunctive with neutral $\perp$ and
- & is conjunctive with neutral $\top$.

The cut rule of LL is:

$$\frac{\vdash \Gamma, A \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}$$

Cut-elimination

Once again, designing a cut-elimination process by local rewrite steps raises the question of whether, when permuting a cut upwards, we should start pushing it into its left-hand-side branch first or into its right-hand-side branch.

And in linear logic as in classical logic, the complete duality illustrated by De Morgan's laws leads to a symmetry that gives us no particular reason to give priority to the left or to the right.

However, the two examples of non-confluence in the previous section are prevented in linear logic by its tight control on the structural rules: weakenings and contractions can only be done on formulae of the form $?A$, which will only be cut against the formula $!A^\perp$ that cannot be weakened or contracted; hence, there cannot be a weakening on both sides of the cut, nor a contraction.

[2]the rules are the exact duals of each other

Identity

$$\frac{}{\vdash a, a^{\perp}}\ (\text{Identity})$$

Logical inference rules

$$\frac{}{\vdash 1}\ (\text{one}) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\ (\text{false})$$

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, A \otimes B, \Delta}\ (\text{times}) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ (\text{par})$$

$$\frac{}{\vdash \Gamma, \top}\ (\text{true}) \qquad (\text{no rule for zero})$$

$$\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\ (\text{with}) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\ (\text{left plus}) \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ (\text{right plus})$$

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ (\text{of course}) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ (\text{weakening})$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ (\text{dereliction}) \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ (\text{contraction})$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall x A}\ (\forall)^{*} \qquad \frac{\vdash \Gamma, \{{}^{t}\!/_{x}\}A}{\vdash \Gamma, \exists x A}\ (\exists)$$

(*) x is not free in Γ

Figure 2.3: The LL system for linear logic

This gives hope for a confluent cut-elimination procedure; however, the choice of pushing a cut to the left or to the right still allows the following example:

$$\frac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma', A}\,(r) \qquad \dfrac{\vdash A^{\perp}, \Delta}{\vdash A^{\perp}, \Delta'}\,(s)}{\vdash \Gamma', \Delta'}\ \mathit{cut}$$

where r (resp. s) abstractly represents the application of some inference rules not involving A (resp. $A^{\perp}$).

There is no natural way to eliminate this cut, since the unspecified rules (r) and (s) do not act on A or $A^{\perp}$; we have the choice of pushing the cut to the right first:

$$\frac{\dfrac{\dfrac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\,\mathit{cut}}{\vdash \Gamma', \Delta}\,(r)}{\vdash \Gamma', \Delta'}\ (s)$$

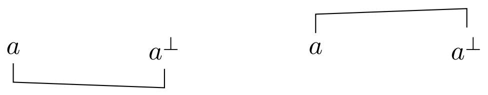
or to the left first:

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, A \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\,\mathit{cut}}{\vdash \Gamma, \Delta'}\,(s)}{\vdash \Gamma', \Delta'}\,(r)$$

and the resulting proofs will remain different even when a cut-free proof is eventually produced.

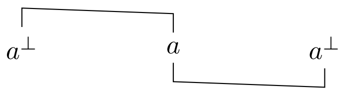
To solve this problem, Girard presents the cut-elimination procedure of LL by means of proof-nets. A *proof net* is constructed on the base of a proof-structure. A *proof-structure* is a graph whose vertices are formulae and whose edges are links; each formula is the conclusion of exactly one link and the premise of at most one link [Gir87]. The formulae which are not premises are the *conclusions* of the structure. Among a proof-structures, proof-nets are those which are obtained as the interpretation of sequent calculus proofs. To obtain a sequent calculus proof from a proof-net, we need the proof-structure to come with a *sequentialisation* that indicates in which order the vertices and edges of the proof-structure have been constructed. The existence of a sequentialisation for a given proof-structure can be characterised by a geometrical criterion on the proof-structure itself [Gir87].

In proof-nets, the cut and axiom links are represented as:



The cut-elimination of LL is presented with proof-nets in [Gir87]. It is defined by case analysis on the vertices above a and $a^\perp$ in the left-hand side picture above.

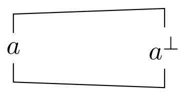
- The following configuration

  is for instance replaced by:

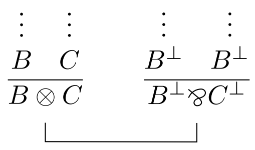
  $$\begin{array}{c}\vdots \\ a^\perp \\ \vdots\end{array}$$

  ...bearing in mind that a proof-structure such as

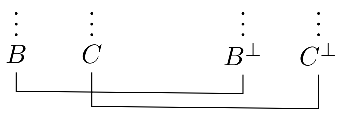
  

  does not satisfy the correctness criterion and is therefore not a proof-net.

- As another example, if the cut-formulae are conclusions of logical links for $\otimes$ and $⅋$,

  is replaced by:

This cut-elimination procedure is shown to be strongly normalising [Gir87, Dan90], and therefore cuts are admissible in LL.

This cut-elimination procedure has some nice features. For instance, it is confluent. It also allows a high level of parallelism since (at least in the multiplicative fragment) the cuts are local and all cut-links can be simultaneously reduced. Finally, the linear features of linear logic provide a tight control on the complexity of the cut-elimination process, which has made linear logic play a key role in the field of *Implicit Computational Complexity*.

2.2.2 The sequent calculus LC

Girard introduced a *polarised* sequent calculus LC for classical logic [Gir91] which is an improvement of LK. The main purpose of this system is to make classical logic more *constructive*[3] and deterministic. Gentzen's sequent calculus LK is non-deterministic, as illustrated by the non-confluence examples. Girard first introduced the notion of *polarity* of a formula of classical logic. The main idea of polarities is to get rid of the non-determinism of cut-elimination that pertains to Gentzen's sequent calculus. To do this, Girard extends the LK system to system LC with the notion of polarity.

Definition 4 (Formulae) The formulae of LC are defined in the following grammar:[4]

$$A, B, \ldots := \; a \mid \neg a \mid A \wedge B \mid A \vee B \mid \forall x A \mid \exists x A \mid V \mid F \mid \neg V \mid \neg F$$

As in linear logic, the constants for "true" and "false" come in two versions: V, F, $\neg V$, $\neg F$. *Negation* is defined via De Morgan's laws:

$$\begin{array}{lll lll}
(a)^\perp & := & \neg a & (\neg a)^\perp & := & a \\
(V)^\perp & := & \neg V & (\neg V)^\perp & := & V \\
(F)^\perp & := & \neg F & (\neg F)^\perp & := & F \\
(A \wedge B)^\perp & := & A^\perp \vee B^\perp & (A \vee B)^\perp & := & A^\perp \wedge B^\perp \\
(\forall x A)^\perp & := & \exists x A^\perp & (\exists x A)^\perp & := & \forall x A^\perp
\end{array}$$

Negation is an involutive function; therefore $A^{\perp^\perp}$ is identical to A and $A \Rightarrow B$ abbreviates $A^\perp \vee B$. ※

Definition 5 (Polarity) The notion of *polarity* of a formula in classical logic (either positive (+) or negative (-)) is defined as follows:

- Atomic formulae a, as well as V and F, are positive.
- Negations of atomic formulae $\neg a$, as well as $\neg V$ and $\neg F$, are negative.
- For compound formulae, polarities depend on the polarities of sub-formulae, according to the following rules:

A	B	$A \wedge B$	$A \vee B$	$\forall x A$	$\exists x A$
$+$	$+$	$+$	$+$	$-$	$+$
$-$	$+$	$+$	$-$	$-$	$+$
$+$	$-$	$+$	$-$		
$-$	$-$	$-$	$-$		

With such a definition, the polarities for $\Rightarrow$ and the negation operation satisfy the following table:

$A \Rightarrow B$	$A^\perp$
$-$	$-$
$+$	$+$
$-$	
$-$	

※

[3] A system is called *constructive* when it is able to define a "reasonable" semantics of proofs, compatible with cut-elimination [Gir91].

[4] Here, we come back to Gentzen's notion of atoms, denoted a, b, etc, rather than the notion of literal, which we could define as atoms a, b and negated atoms $\neg a$, $\neg b$, etc.

Some remarks:

- Atomic formula V is not identified with the negation of F, in order to get *positive* constants for the two truth values.
- Interestingly enough, polarities of the connectives $\wedge$, $\vee$, $\Rightarrow$, $(\cdot)^\perp$ follow the usual truth table where "+" stands for false and "-" stands for "true". This polarity table is actually isomorphic to the truth table.
- Quantifiers have a forced polarity.
- Note that we can always turn a formula A into a positive formula A^+ that is (classically) equivalent to A: $A^+ := \ A \wedge V$. Similarly, we can always turn a formula A into a negative formula A^- that is (classically) equivalent to A: $A^- := \ A \vee \neg V$.

  Now, we discuss the LC system.

Definition 6 (The LC system) The LC system is a mono-sided sequent calculus, with sequents of the form of $\vdash \Gamma; \Pi$ where Γ and Π are multisets of formulae,[5] and Π is either empty or consists of exactly one positive formula. The space after ";" is called the *stoup*, Γ is called the *body* and what is in the stoup is called the *head*. The rules of the LC sequent calculus are presented in Figure 2.4. In this system, P, Q, R refer to positive formulae, L, M, N refer to negative formulae and A, B, C refer to "undefined" formulae whose polarity is not specified. ※

Identity

$$\frac{}{\vdash \neg P;P}\ \text{Identity}$$

Logic

$$\frac{}{\vdash\ ;V} \qquad \frac{}{\vdash \Gamma, \neg F;\Pi}$$

$$\frac{\vdash \Gamma;P \quad \vdash \Delta;Q}{\vdash \Gamma, \Delta;P \wedge Q} \qquad \frac{\vdash \Gamma, A, B;\Pi}{\vdash \Gamma, A \vee B;\Pi}\ \text{when } A \vee B \text{ is negative}$$

$$\frac{\vdash \Gamma;P \quad \vdash \Delta, N;}{\vdash \Gamma, \Delta;P \wedge N} \qquad \frac{\vdash \Gamma, M; \quad \vdash \Delta;Q}{\vdash \Gamma, \Delta;M \wedge Q}$$

$$\frac{\vdash \Gamma, M;\Pi \quad \vdash \Gamma, N;\Pi}{\vdash \Gamma, M \wedge N;\Pi} \qquad \frac{\vdash \Gamma;P}{\vdash \Gamma;P \vee Q} \qquad \frac{\vdash \Gamma;Q}{\vdash \Gamma;P \vee Q}$$

$$\frac{\vdash \Gamma, A;\Pi}{\vdash \Gamma, \forall x A;\Pi}\ * \qquad \frac{\vdash \Gamma, \{{}^{t}\!/\!_{x}\}N;}{\vdash \Gamma;\exists x N} \qquad \frac{\vdash \Gamma;\{{}^{t}\!/\!_{x}\}P}{\vdash \Gamma;\exists x P}$$

Structure

$$\frac{\vdash \Gamma;P}{\vdash \Gamma, P;}\ \text{dereliction} \qquad \frac{\vdash \Gamma;\Pi}{\vdash \Gamma, A;\Pi}\ \text{weakening} \qquad \frac{\vdash \Gamma, A, A;\Pi}{\vdash \Gamma, A;\Pi}\ \text{contraction}$$

(*) x is not free in Γ

Figure 2.4: The LC system for classical logic

There are two cuts in the LC system:

[5] Girard introduced the system with sequences of formulae and an exchange rule similar to Gentzen's.

$$\frac{\vdash \Gamma;P \qquad \vdash P^{\perp},\Delta;\Pi}{\vdash \Gamma,\Delta;\Pi}\text{p-cut} \qquad \frac{\vdash \Gamma,N; \qquad \vdash N^{\perp},\Delta;\Pi}{\vdash \Gamma,\Delta;\Pi}\text{n-cut}$$

and these can be used to prove the following theorem.

First, notice that the sequents of the LK system straightforwardly translate as sequents of system LC with empty stoups. If a sequent of LK is provable (in LK), then its corresponding sequent in LC (with an empty stoup) is provable in LC [Gir91].

This translation introduces cuts, i.e. certain cut-free proofs in LK will translate in LC to proofs with cuts.

Cut-elimination

The sequent calculus LC admits the above cuts: in [Gir91], a syntactic cut-elimination procedure is sketched that has the interesting property of preserving a certain denotational semantics of proofs (with or without cuts). This would clearly not be the case of the general cut-elimination procedure of LK or $\mathbf{G3}_c$ (unless it is restricted in some ways) as illustrated by the non-confluence examples.

The preservation of semantics by cut-elimination is a key feature in the interpretation of proofs as programs (Curry-Howard correspondence), which can be seen for instance between natural deduction for minimal logic and the simply-typed λ-calculus, and therefore LC gives an interesting light on the computational interpretation of classical proofs and their "constructive" contents. This is strengthened by some variants of the *disjunction property* and the *existential property* that hold in LC for the formula in the stoup:

- if $\vdash ;P \vee Q$ is provable then either $\vdash ;P$ or $\vdash ;Q$ is provable;
- if $\vdash ;\exists x P$ is provable then for some term t, $\vdash ;P[t/x]$ is provable.

Finally, an important aspect of LC is that the polarity of a formula does not change the provability of the formula; it only affects the shape of its proofs.

Girard identifies several directions for further work, of which we will mention two:

1. materialise the computational contents of LC proofs with a syntax that would be to LC what typed λ-calculus is to LJ, and an hopefully confluent normalisation that represents cut-elimination;
2. put classical, intuitionistic and linear logic inside the same system in such a way that these three systems appears as fragments, since several notions seem to be common features (stoup, polarities, etc).

2.3 Introducing focusing in linear logic

The concept of *focusing* has been introduced in [AP89, AP91, And92] in the context of linear logic. It actually influenced Girard's work on LC for classical logic, but also relates to the work by Miller et al. [MNPS91] on the concept of *uniform proofs* for Hereditary Harrop formulae (in intuitionistic logic).

Both [And92] and [MNPS91] explicitly take the view of proof-search, as in logic programming. At the basis of logic programming, proof-search on *Horn clauses* can be understood as a meaningful computational paradigm because this class of formulae makes a simple goal-directed proof-search strategy logically complete (with well-identified backtrack points and a reasonably efficient covering of the proof-search space). In [MNPS91], this is shown to still hold when the class is extended to Hereditary Harrop formulae.

Andreoli's aim is to design a notion of Logic Programming in linear logic, and use for this the "natural" proof-search process specified by the rules of an inference system:

Namely, this proof-search incrementally builds a proof-tree for the item to be proved (typically, a sequent); it starts with an incomplete proof-tree made of only one node labelled by the item, and then tries to applies the inference rules bottom-up so as to gradually construct a bigger and bigger incomplete proof-tree, and eventually "close" its branches with inference rules that have no premises; if and when all branches become closed, then the resulting object is a complete proof-tree of the original item.

Of course, at each node, the process must choose a rule instance to apply, and if after that choice the incomplete proof-tree cannot be completed, then the process needs to backtrack and make another choice.

Andreoli's point is that the inference rules of LL are too permissive for this natural proof-search process to be efficient.

The main advantage of *focusing* regarding this natural proof-search process is that, instead of trying to search for all the possible proofs of a sequent, we only search for proofs of a particular shape. Such proofs should form a subset of all proofs that is logically complete: the proofs from this subset can be viewed as "normal" representatives of equivalence classes of proofs, in this case *P-equivalence*[6] classes.

2.3.1 Preliminaries

Logical inference rules lead to multiple cases of permutation of inferences. Therefore, to apply a logical inference bottom-up, at a given node of the proof-search, we encounter problems in making a selection. These are namely to:

- Select, in the sequent of the node, a principal formula to be decomposed.
- Select an instance of the logical inference rule associated with the topmost connective of the selected principal formula.

To make the proof-search more deterministic, the connectives of linear logic are partitioned into two groups:

- The *positive connectives*:
  - Multiplicative: 1, $\otimes$, $!$
  - Additive: 0, $\oplus$, $\exists$
- The *negative connectives* :
  - Multiplicative: $\bot$, $⅋$, $?$
  - Additive: $\top$, $\&$, $\forall$

It is important to note that the dual of an negative connective is positive and vice versa. A non-atomic formula whose top-most connective is positive (resp. negative) is called a positive (resp. negative) formula. Andreoli's *focusing* results [And92] are informally described below:

- Negative connectives can be decomposed, in sequent calculus style, with *invertible* inference rules that are called *asynchronous*: a proof-search strategy can perform the bottom-up application of those rules as basic proof-search steps without loss of generality (if the goal was provable, it remains provable after applying the step); in other words, no backtracking is necessary on the application of such steps, even though other steps were possible.
- Positive connectives are the (De Morgan's) duals of negative connectives, and their decomposition rules, which are called *synchronous*, are not necessarily invertible.

Clearly, asynchronous rules can be applied eagerly, i.e. can be chained, without creating backtrack points and losing completeness.

Quite surprisingly, it turns out that synchronous rules (although possibly creating backtrack points) **can also be chained** without losing completeness [And92].

This result can be expressed as the completeness of a sequent calculus with a *focus* device, which syntactically highlights a formula in the sequent and forces the next proof-search step to decompose it with a synchronous rule, keeping the focus on its newly-revealed sub-formulae. Focusing considerably reduces the proof-search space, otherwise heavily redundant when Gentzen-style inference rules are used.

[6]Two proofs are said to be P-equivalent if each of them is obtained from the other by simple permutations of inference rules and elimination or introduction of useless "loops".

This relates to the *stoup* in Girard's LC: if a formula is in the stoup, then the next inference rule decomposes it. The chaining of synchronous rules then relates to the behaviour of the LC rules

$$\frac{\vdash \Gamma;P \quad \vdash \Delta;Q}{\vdash \Gamma,\Delta;P\wedge Q} \qquad \frac{\vdash \Gamma;P}{\vdash \Gamma;P\vee Q} \qquad \frac{\vdash \Gamma;Q}{\vdash \Gamma;P\vee Q} \qquad \frac{\vdash \Gamma;\left\{{}^{t}\!/\!{}_{x}\right\}P}{\vdash \Gamma;\exists x P}$$

where the direct sub-formulae of the formula being decomposed in the stoup are themselves kept in the stoup as long as they remain positive.

2.3.2 The triadic system LLF

Now, we present the triadic system *LLF* that is used in [And92] to materialise those ideas in linear logic.

DEFINITION 7 (LLF)

First, we split the set of literals into a set of positive literals and a set of negative literals, such that a is positive if and only if $a^{\perp}$ is negative.
A triadic sequent is of one of the following forms:

Focused sequent:	$\vdash \Theta : \Delta \Downarrow F$	where F is a formula said to be in the *stoup* or in *focus*
Unfocused sequent:	$\vdash \Theta : \Delta \Uparrow L$	where L is an ordered list of formulae;

where Θ and Δ are multisets of formulae, and Δ contains only literals and positive formulae. The LLF sequent calculus is given by the rules of Figure 2.5. Here, F, G stand for formulae, p stands for a positive literal. ※

- A sequent $\vdash \Theta : \Delta \Uparrow L$ corresponds to the case where the sequent possibly contains a *negative formula* (in L).
- A sequent $\vdash \Theta : \Delta \Downarrow F$ corresponds to the case where all the negative formulae have been decomposed and a formula F has been selected as *principal formula*.

Negative formulae are decomposed immediately as soon as they appear in the sequent. Positive formulae are delayed until all the negative formulae have been decomposed, and must be non-deterministically selected to be processed: in other words, positive connectives "synchronise" the selection process and the decomposition process (hence the name *synchronous*). When a positive formula starts being decomposed, it keeps on being decomposed until an atomic or a negative formula is reached. The grouping of synchronous rules is called a synchronous phase, while the grouping of asynchronous rules is called an asynchronous phase.

The kind of non-determinism involved in the synchronous phases (including the selection of the positive formula to be placed in the stoup) is a *don't know non-determinism* (if the wrong choice is made, backtracking will be needed to make another choice); that which appears in the asynchronous phase is a *don't care non-determinism* (no backtracking is necessary on the application of asynchronous rules, even though other rule applications were possible).

More precisely:

- When all the negative formulae have been decomposed in an $\Uparrow$-sequent (i.e. L is empty), a principal formula must be non-deterministically selected by the *Decision rule* and a new synchronous phase is started (with $\Downarrow$ sequents). The principal formula may be picked either inside the second field of the sequent, containing the non-negative formulae which have been delayed by the *Reaction rule* $[R\Uparrow]$ above (Decision $[D_1]$), or in the first field, i.e. the 'reserve tank' of unrestricted formulae (Decision $[D_2]$) where a copy of the selected formula is kept.
- The Reaction rule $[R\Downarrow]$ is triggered when an negative formula is reached at the end of a synchronous phase. The arrow is just turned upside down, which means that the synchronous phase is finished and the negative formula must be decomposed.
- When a positive literal p is reached at the end of a synchronous phase, an *Identity rule* must be used, so that $p^{\perp}$ must be found in the rest of the sequent, either as a restricted resource in the second field (Identity $[I_1]$), or as an unrestricted resource in the first field (Identity $[I_2]$).

Identities

$$\frac{}{\vdash \Theta : p^\perp \Downarrow p}\,(I_1) \qquad \frac{}{\vdash \Theta, p^\perp : \Downarrow p}\,(I_2)$$

Synchronous rules

$$\frac{}{\vdash \Theta : \Downarrow 1}\,(1) \qquad \frac{\vdash \Theta : \Delta_1 \Downarrow F \quad \vdash \Theta : \Delta_2 \Downarrow G}{\vdash \Theta : \Delta_1, \Delta_2 \Downarrow F \otimes G}\,(\otimes) \qquad \frac{\vdash \Theta : \Uparrow F}{\vdash \Theta : \Downarrow !F}\,(!)$$

$$\frac{\vdash \Theta : \Delta \Downarrow F}{\vdash \Theta : \Delta \Downarrow F \oplus G}\,(\oplus_l) \qquad \frac{\vdash \Theta : \Delta \Downarrow G}{\vdash \Theta : \Delta \Downarrow F \oplus G}\,(\oplus_r) \qquad \frac{\vdash \Theta : \Delta \Downarrow \{{}^{t}\!/\!_{x}\} F}{\vdash \Theta : \Delta \Downarrow \exists x F}\,(\exists)$$

Asynchronous rules

$$\frac{\vdash \Theta : \Delta \Uparrow L}{\vdash \Theta : \Delta \Uparrow L, \perp}\,(\perp) \qquad \frac{\vdash \Theta : \Delta \Uparrow L, F, G}{\vdash \Theta : \Delta \Uparrow L, F \parr G}\,(\parr) \qquad \frac{\vdash \Theta, F : \Delta \Uparrow L}{\vdash \Theta : \Delta \Uparrow L, ?F}\,(?)$$

$$\frac{}{\vdash \Theta : \Delta \Uparrow L, \top}\,(\top) \qquad \frac{\vdash \Theta : \Delta \Uparrow L, F \quad \vdash \Theta : \Delta \Uparrow L, G}{\vdash \Theta : \Delta \Uparrow L, F \,\&\, G}\,(\&) \qquad \frac{\vdash \Theta : \Delta \Uparrow L, F}{\vdash \Theta : \Delta \Uparrow L, \forall x F}\,(\forall)^*$$

Reaction $\Uparrow$: if F is a (negative) literal or a positive formula

$$\frac{\vdash \Theta : \Delta, F \Uparrow L}{\vdash \Theta : \Delta \Uparrow L, F}\,(R\Uparrow)$$

Reaction $\Downarrow$: if F is negative

$$\frac{\vdash \Theta : \Delta \Uparrow F}{\vdash \Theta : \Delta \Downarrow F}\,(R\Downarrow)$$

Decisions: If F is not a negative literal

$$\frac{\vdash \Theta : \Delta \Downarrow F}{\vdash \Theta : \Delta, F \Uparrow}\,(D_1) \qquad \frac{\vdash \Theta, F : \Delta \Downarrow F}{\vdash \Theta, F : \Delta \Uparrow}\,(D_2)$$

(*) x is not free in Θ, Δ, L

Figure 2.5: The LLF system for linear logic

- When a formula prefixed with the negative modality ? is encountered, it is immediately stored in the first field of the sequent for possible future use, instead of being put back in the third field of the sequent for further decomposition of negative connectives, as in the standard negative case.
- The modality !, unlike standard positive connectives, terminates a synchronous phase, but it requires that, at the moment of the interruption, the second field of the sequent is empty.

The focusing discipline considerably reduces the amount of non-determinism involved in the proof-search, while still being logically complete. Proving completeness can be done in different ways, one of which goes through the elimination of the following cut-rule:

$$\frac{\vdash \Theta : \Delta \Downarrow A \quad \vdash \Theta : \Delta \Downarrow A^\perp}{\vdash \Theta : \Delta, \Delta' \Uparrow}\, cut$$

Notice that polarities are such that, in the above cut, only one formula out of A and $A^\perp$ is positive and only one is negative. Therefore the focusing structure of the proof proving the left premise must me radically different from that proving the right premise, and therefore the sophisticated structure of the sequent calculus breaks the left-right symmetry. This reduces the non-determinism of the cut-elimination process (which [And92] does not explicitly give for the triadic system), and focusing can thus be seen as an alternative to proof-nets to define a meaningful computational interpretation of cut-elimination in linear logic.

This idea will be imported back into classical logic, for which the right notion of proof-nets may be more difficult to identify.

Indeed, following Girard's and Andreoli's work on linear logic, a substantial literature investigated the impact of proof-nets, polarities, and focusing on classical logic: Among such literature, *polarised classical logic* emerged [Lau02], acknowledging the fact that the choice of inference rules for $\wedge$ and $\vee$ defines two versions of the connectives, namely $\wedge^+$ (resp. $\vee^+$) and $\wedge^-$ and (resp. $\vee^-$), which, although provably equivalent, differ in their proofs, in the semantics of their proofs, and in their proof-search mechanisms. This work also relates to a rich literature aiming at giving an computational interpretation of cut-elimination in classical logic, inspired by its various translations in linear and intuitionistic logics (e.g. [DJS95, DJS97, CH00, LQdF05]). Polarised classical logic develops and enriches Girard's work on LC, in particular by explaining the proof theory of classical formulae as given by LC as a combination of

- an encoding from classical formulae to *polarised* classical formulae
- a proof theory for polarised classical logic.

The next section describes a proof-theoretical approach to polarised classical logic, with the specific view of proof-search in mind and therefore along the lines of Andreoli's work for linear logic.

2.4 The focused sequent calculus LKF for classical logic

Liang and Miller introduced in [LM09] a focused sequent calculus LJF for intuitionistic logic and a focused sequent calculus *LKF* for polarised classical logic which builds on, extends, and clarifies LC with the polarities and focusing concepts of [And92, Lau02] applied to classical logic.

As already said, proof-search on *Hereditary Harrop formulae* can be understood as a meaningful computational paradigm [MNPS91], because this class of formulae makes a simple goal-directed proof-search strategy logically complete (with well-identified backtrack points and a reasonably efficient covering of the proof-search space).

LLF and LKF show that this still holds for linear logic and polarised classical logic, respectively, where logical connectives and literals have *polarities*: positive or negative.

A sequent with a positive literal in focus must be proved immediately by an axiom (a.k.a identity) on that literal; hence, the polarity of literals greatly affects the shape of proofs. As illustrated in e.g. [LM09], the following sequent expresses the *Fibonacci* logic program together with a goal $\mathsf{fib}(n,p)$ (where n and p are closed terms):[7]

$$\begin{array}{l}\mathsf{fib}(0,0),\\ \mathsf{fib}(1,1),\\ \forall i p_1 p_2 (\mathsf{fib}(i,p_1) \Rightarrow \mathsf{fib}(i+1,p_2) \Rightarrow \mathsf{fib}(i+2,p_1+p_2))\\ \vdash \mathsf{fib}(n,p)\end{array}$$

The goal will be proved with backward-reasoning if the fib literals are negative (yielding a proof of exponential size in n), and forward-reasoning if they are positive (yielding many proofs, one of which being linear).

In *classical* logic, polarities of connectives and literals do not affect the provability of formulae, but still greatly affect the shape of proofs, and hence the basic proof-construction steps.

[7]Of course, *addition* is used in the example and needs to makes sense in the logic programming system; one way of doing this is to define it in a logic programming style, another way is to have arithmetic primitively in the system, as we shall develop in the next chapter.

Definition 8 (Formulae, negation) As in LLF, *literals* are classified as either *positive* or *negative*, and $a^\perp$ has the opposite polarity of the literal a.

$$\begin{array}{lcl} \text{Positive formulae} & := & p \mid \top^+ \mid \bot^+ \mid A{\wedge^+}B \mid A{\vee^+}B \mid \exists x A \\ \text{Negative formulae} & := & p^\perp \mid \top^- \mid \bot^- \mid A{\wedge^-}B \mid A{\vee^-}B \mid \forall x A \end{array}$$

where p ranges over positive literals.
Negation is extended as usual using De Morgan's laws, such that the negation of a positive formula is negative and vice versa. ※

Definition 9 (LKF system) There are two kinds of sequents:

$$\begin{array}{ll} \text{Focused sequent:} & \vdash \Theta \Downarrow A \text{ where } A \text{ is in focus} \\ \text{Unfocused sequent:} & \vdash \Theta \Uparrow \Delta \end{array}$$

where Θ is a multiset of positive formulae and negative literals and Δ is a multiset of formulae.
The rules of LKF are defined in Figure 2.6, where P is positive, N is negative, C is a positive formula or negative literal, and p is a positive literal. ※

Synchronous rules

$$\frac{\vdash \Theta \Downarrow A \qquad \vdash \Theta \Downarrow B}{\vdash \Theta \Downarrow A{\wedge^+}B} \qquad \frac{\vdash \Theta \Downarrow A_i}{\vdash \Theta \Downarrow A_1{\vee^+}A_2}$$

$$\frac{\vdash \Theta \Downarrow \{{}^t\!/_x\}A}{\vdash \Theta \Downarrow \exists x A} \qquad \frac{}{\vdash \Theta \Downarrow \top^+}$$

Asynchronous rules

$$\frac{\vdash \Theta \Uparrow \Delta, A \qquad \vdash \Theta \Uparrow \Delta, B}{\vdash \Theta \Uparrow \Delta, A{\wedge^-}B} \qquad \frac{\vdash \Theta \Uparrow \Delta, A, B}{\vdash \Theta \Uparrow \Delta, A{\vee^-}B} \qquad \frac{\vdash \Theta \Uparrow \Delta, A}{\vdash \Theta \Uparrow \Delta, (\forall x A)}\ x \text{ not free in } \Theta, \Delta$$

$$\frac{}{\vdash \Theta \Uparrow \Delta, \top^-} \qquad \frac{\vdash \Theta \Uparrow \Delta}{\vdash \Theta \Uparrow \Delta, \bot^-}$$

Structure rules

$$\frac{\vdash \Theta, C \Uparrow \Delta}{\vdash \Theta \Uparrow \Delta, C}\,\mathit{Store} \qquad \frac{\vdash P, \Theta \Uparrow P}{\vdash P, \Theta \Uparrow}\,\mathit{Select} \qquad \frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}\,\mathit{Release} \qquad \frac{}{\vdash p^\perp, \Theta \Downarrow p}\,\mathit{Id}$$

Figure 2.6: The LKF system for classical logic

Again, as in LLF, the gradual proof-tree construction defined by the bottom-up application of the inference rules of LKF, is a goal-directed mechanism whose intuition can be given as follows:

Asynchronous rules are invertible: they are applied eagerly when trying to construct the proof-tree of a given sequent; *Store* is applied when hitting a positive formula or a negative literal on the right-hand side of a sequent, storing it.

When the right-hand side of a sequent becomes empty, a choice must be made to place a positive formula in focus, using rule (*Select*), before applying synchronous rules.

Each such rule decomposes the formula in focus, keeping the revealed sub-formulae in the focus of the corresponding premises, until a positive literal or a non-positive formula is obtained: the former case must be closed immediately with *Id*, and the latter case uses the *Release* rule to drop the focus and start applying asynchronous rules again. The synchronous and the structural rules are in general not invertible,[8] so each application of those yields in general a backtrack point in the proof-search.

[8](but they may be so, e.g. for $\wedge^+$)

Remark 2 Defining negative implication $A{\Rightarrow^-}B$ as $(A^\perp){\vee^-}B$ and positive implication $A{\Rightarrow^+}B$ as $(A^\perp){\vee^+}B$, we derive the following rules for those connectives:

$$\frac{\vdash \Theta \Downarrow A^\perp}{\vdash \Theta \Downarrow A{\Rightarrow^+}B} \qquad \frac{\vdash \Theta \Downarrow B^\perp}{\vdash \Theta \Downarrow A{\Rightarrow^+}B} \qquad \frac{\vdash \Theta \Uparrow \Delta, B, A^\perp}{\vdash \Theta \Uparrow \Delta, A{\Rightarrow^-}B}$$

The LKF system can be translated into the LLF system. And it is sound and complete for classical logic, regardless of which polarities are placed on literals and connectives [LM09]. The following cut rules are admissible

$$\frac{\vdash \Theta \Uparrow \Delta, C \quad \vdash \Theta' \Uparrow \Delta', C^\perp}{\vdash \Theta, \Theta' \Uparrow \Delta, \Delta'}\,Cut_p \qquad \frac{\vdash \Theta \Downarrow B \quad \vdash \Theta' \Uparrow \Delta', B^\perp}{\vdash \Theta, \Theta' \Uparrow \Delta'}\,Cut_k \qquad \frac{\vdash \Theta, P \Downarrow B \quad \vdash \Theta' \Uparrow P^\perp}{\vdash \Theta\Theta' \Downarrow B}\,Cut_f$$

A syntactic argument for the admissibility of these cuts is sketched in [LM09].

System LKF assumes that all literals come with a pre-determined polarity. In the sequent calculi that we develop in the rest of this thesis, we will generalise this to allow the polarity of literals to be determined *on-the-fly* during proof-search: the root of a proof-tree might have none of its literals polarised, but literals may become positive or negative as progress is made in the proof-search.

Chapter 3

A sequent calculus for classical logic modulo theories

In this chapter we briefly review the proof-search motivation for focused proof systems, and present the focused sequent calculus $\mathit{LK}^p(\mathcal{T})$ for (polarised) classical logic *modulo a theory.*

3.1 $\mathsf{LK}^p(\mathcal{T})$: Definitions

The sequent calculus $\mathsf{LK}^p(\mathcal{T})$ manipulates the formulae of first-order logic, with the specificity that connectives are of one of two kinds: positive ones and negative ones, and each *boolean* connective comes in two versions, one of each kind. This section develops the preliminaries and the definition of the $\mathsf{LK}^p(\mathcal{T})$ system.

Definition 10 (Terms and literals) Consider an infinite set of elements called *variables.*
The set of *terms* over a first-order (function) signature F_Σ is defined by:

$$t, t_1, t_2, \ldots := \; x \mid f(t_1, \ldots, t_n)$$

with f/n (f of arity n) ranging over F_Σ and x ranging over variables.
Let P_Σ be a first-order predicate signature equipped with an involutive and arity-preserving function called *negation.* The negation of a predicate symbol P is denoted $P^\perp$.
Let $\mathcal{L}^\top$ be the set $\{P(t_1, \ldots, t_n) \mid P/n \in P_\Sigma, t_1, \ldots t_n \text{ terms}\}$, to which we extend the involutive function of negation with:

$$(P(t_1, \ldots, t_n))^\perp := \; P^\perp(t_1, \ldots, t_n)$$

The substitution, in a term t', of a term t for a variable x, denoted $\{{}^t\!/\!_x\}t'$, is defined as usual, and straightforwardly extended to elements of $\mathcal{L}^\top$.
In the rest of this chapter, we consider a subset $\mathcal{L} \subseteq \mathcal{L}^\top$, of elements called *literals* and denoted $l, l_1, l_2 \ldots$, that is closed under negation and under substitution.[1]
For a set $\mathcal{A}$ of literals, we write $\{{}^t\!/\!_x\}\mathcal{A}$ for the set $\{\{{}^t\!/\!_x\}l \mid l \in \mathcal{A}\}$. The closure of $\mathcal{A}$ under all possible substitutions is denoted $\mathcal{A}^\downarrow$. ※

Notation 11 We often write $\mathcal{V}, \mathcal{V}'$ for the set or multiset union of $\mathcal{V}$ and $\mathcal{V}'$. ※

Remark 3 Negation obviously commutes with substitution.

[1]Very often we will take $\mathcal{L} = \mathcal{L}^\top$, but it is not a necessity.

DEFINITION 12 (Inconsistency predicates)

An *inconsistency predicate* is a predicate over sets of literals

1. satisfied by the set $\{l, l^{\perp}\}$ for every literal l;
2. that is upward closed (if a subset of a set satisfies the predicate, so does the set);
3. such that if the sets $\mathcal{A}, l$ and $\mathcal{A}, l^{\perp}$ satisfy it, then so does $\mathcal{A}$.
4. such that if a set $\mathcal{A}$ satisfies it, then so does $\{{}^{t}\!/_{x}\}\mathcal{A}$.

The smallest inconsistency predicate is called the *syntactical inconsistency* predicate.[2] If a set $\mathcal{A}$ of literals satisfies the syntactical inconsistency predicate, we say that $\mathcal{A}$ is *syntactically inconsistent*, denoted $\mathcal{A} \models$. Otherwise $\mathcal{A}$ is *syntactically consistent.*

In the rest of this chapter, we specify a "theory" $\mathcal{T}$ by considering another inconsistency predicate called the *semantical inconsistency* predicate. If a set $\mathcal{A}$ of literals satisfies it, we say that $\mathcal{A}$ is *semantically inconsistent*, denoted by $\mathcal{A} \models_{\mathcal{T}}$. Otherwise $\mathcal{A}$ is *semantically consistent.* ※

REMARK 4

- In the conditions above, (1) corresponds to *basic inconsistency*, (2) corresponds to *weakening*, (3) corresponds to *cut-admissibility* and (4) corresponds to *stability under instantiation. Contraction* is built-in because inconsistency predicates are predicates over sets of literals (not multisets).
- If $\mathcal{A}$ is syntactically consistent, $\{{}^{t}\!/_{x}\}\mathcal{A}$ might not be syntactically consistent.

DEFINITION 13 (Formulae)

The *formulae* of polarised classical logic are given by the following grammar:

$$\begin{array}{llll} \text{Formulae} & A, B, \ldots & ::= & l \\ & & | & A\wedge^{+}B \mid A\vee^{+}B \mid \exists xA \mid \top^{+} \mid \bot^{+} \\ & & | & A\wedge^{-}B \mid A\vee^{-}B \mid \forall xA \mid \top^{-} \mid \bot^{-} \end{array}$$

where l ranges over $\mathcal{L}$.

The set of *free variables* of a formula A, denoted $\mathsf{FV}(A)$, and α-conversion, are defined as usual so that both $\exists xA$ and $\forall xA$ bind x in A.

The size of a formula A, denoted $\sharp(A)$, is its size as a tree (number of nodes).

Negation is extended from literals to all formulae:

$(A\wedge^{+}B)^{\perp}$	$:=$	$A^{\perp}\vee^{-}B^{\perp}$	$(A\wedge^{-}B)^{\perp}$	$:=$	$A^{\perp}\vee^{+}B^{\perp}$
$(A\vee^{+}B)^{\perp}$	$:=$	$A^{\perp}\wedge^{-}B^{\perp}$	$(A\vee^{-}B)^{\perp}$	$:=$	$A^{\perp}\wedge^{+}B^{\perp}$
$(\exists xA)^{\perp}$	$:=$	$\forall xA^{\perp}$	$(\forall xA)^{\perp}$	$:=$	$\exists xA^{\perp}$
$(\top^{+})^{\perp}$	$:=$	$\bot^{-}$	$(\top^{-})^{\perp}$	$:=$	$\bot^{+}$
$(\bot^{+})^{\perp}$	$:=$	$\top^{-}$	$(\bot^{-})^{\perp}$	$:=$	$\top^{+}$

The *substitution* in a formula A of a term t for a variable x, denoted $\{{}^{t}\!/_{x}\}A$, is defined in the usual capture-avoiding way. ※

NOTATION 14 For a set (resp. multiset) $\mathcal{V}$ of literals / formulae, $\mathcal{V}^{\perp}$ denotes $\{A^{\perp} \mid A \in \mathcal{V}\}$ (resp. $\{\!\{A^{\perp} \mid A \in \mathcal{V}\}\!\}$). Similarly, we write $\{{}^{t}\!/_{x}\}\mathcal{V}$ for $\{\{{}^{t}\!/_{x}\}A \mid A \in \mathcal{V}\}$ (resp. $\{\!\{\{{}^{t}\!/_{x}\}A \mid A \in \mathcal{V}\}\!\}$), and $\mathsf{FV}(\mathcal{V})$ for the set $\bigcup_{A\in\mathcal{V}} \mathsf{FV}(A)$. ※

DEFINITION 15 (Polarities)

A *polarisation set* $\mathcal{P}$ is a set of literals ($\mathcal{P} \subseteq \mathcal{L}$) that is syntactically consistent, and such that $\mathsf{FV}(\mathcal{P})$ is finite.

Given such a set, we define $\mathcal{P}$*-positive formulae* and $\mathcal{P}$*-negative formulae* as the formulae generated by the following grammars:

$$\begin{array}{lll} \mathcal{P}\text{-positive formulae} & P, \ldots & ::= p \mid A\wedge^{+}B \mid A\vee^{+}B \mid \exists xA \mid \top^{+} \mid \bot^{+} \\ \mathcal{P}\text{-negative formulae} & N, \ldots & ::= p^{\perp} \mid A\wedge^{-}B \mid A\vee^{-}B \mid \forall xA \mid \top^{-} \mid \bot^{-} \end{array}$$

where p ranges over $\mathcal{P}$.

[2] It is the predicate that is true of a set $\mathcal{A}$ of literals iff $\mathcal{A}$ contains both l and $l^{\perp}$ for some $l \in \mathcal{L}$.

In the rest of the chapter, p, p',... will denote a literal that is $\mathcal{P}$-positive, when the polarisation set $\mathcal{P}$ is clear from context.
Let $\mathsf{U}_{\mathcal{P}}$ be the set of all *$\mathcal{P}$-unpolarised literals*, i.e. literals that are neither $\mathcal{P}$-positive nor $\mathcal{P}$-negative. ※

Remark 5 Notice that the negation of a $\mathcal{P}$-positive formula is $\mathcal{P}$-negative and vice versa. On the contrary, nothing can be said of the polarity of the result of substitution on a literal w.r.t. the polarity of the literal: e.g. l could be in $\mathcal{P}$-positive, while $\{{}^{t}\!/\!_{x}\}l$ could be $\mathcal{P}$-negative or $\mathcal{P}$-unpolarised.

Definition 16 ($\mathsf{LK}^p(\mathcal{T})$)
The sequent calculus $\mathsf{LK}^p(\mathcal{T})$ manipulates two kinds of sequents:

Focused sequents	$\Gamma \vdash^{\mathcal{P}} [A]$
Unfocused sequents	$\Gamma \vdash^{\mathcal{P}} \Delta$

where $\mathcal{P}$ is a polarisation set, Γ is a (finite) multiset of literals and $\mathcal{P}$-negative formulae, Δ is a (finite) multiset of formulae, and A is said to be in the *focus* of the (focused) sequent.
By $\mathsf{lit}_{\mathcal{P}}(\Gamma)$ we denote the sub-multiset of Γ consisting of its $\mathcal{P}$-positive literals (i.e. $\mathcal{P} \cap \Gamma$ as a set).
The rules of $\mathsf{LK}^p(\mathcal{T})$, given in Figure 3.1, are of three kinds: *synchronous rules*, *asynchronous rules*, and *structural rules*. These correspond to three alternating phases in the proof-search process that is described by the rules. ※

Synchronous rules

$$(\wedge^+)\frac{\Gamma \vdash^{\mathcal{P}} [A] \qquad \Gamma \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]} \qquad (\vee^+)\frac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]} \qquad (\exists)\frac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}A]}{\Gamma \vdash^{\mathcal{P}} [\exists x A]}$$

$$(\top^+)\frac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]} \qquad (\mathsf{Init}_1)\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^\perp \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [l]}\ l \text{ is } \mathcal{P}\text{-positive} \qquad (\mathsf{Release})\frac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]}\ N \text{ is not } \mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \qquad \Gamma \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash^{\mathcal{P}} A \wedge^- B, \Delta} \qquad (\vee^-)\frac{\Gamma \vdash^{\mathcal{P}} A_1, A_2, \Delta}{\Gamma \vdash^{\mathcal{P}} A_1 \vee^- A_2, \Delta} \qquad (\forall)\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta}{\Gamma \vdash^{\mathcal{P}} (\forall x A), \Delta}\ x \notin \mathsf{FV}(\Gamma, \Delta, \mathcal{P})$$

$$(\perp^-)\frac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta, \perp^-} \qquad (\top^-)\frac{}{\Gamma \vdash^{\mathcal{P}} \Delta, \top^-} \qquad (\mathsf{Store})\frac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}\ \begin{array}{l} A \text{ is a literal} \\ \text{or is } \mathcal{P}\text{-positive}\end{array}$$

Structural rules

$$(\mathsf{Select})\frac{\Gamma, P^\perp \vdash^{\mathcal{P}} [P]}{\Gamma, P^\perp \vdash^{\mathcal{P}}}\ P \text{ is not } \mathcal{P}\text{-negative} \qquad (\mathsf{Init}_2)\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

$$\begin{array}{lrll} \text{where} & \mathcal{P}; A := & \mathcal{P}, A & \text{if } A \in \mathsf{U}_{\mathcal{P}} \\ & \mathcal{P}; A := & \mathcal{P} & \text{if not} \end{array}$$

Figure 3.1: System $\mathsf{LK}^p(\mathcal{T})$

The gradual proof-tree construction defined by the inference rules of $\mathsf{LK}^p(\mathcal{T})$ is a goal-directed mechanism whose intuition can be given as follows:

Asynchronous rules are invertible: $(\wedge^-)$ and $(\vee^-)$ are applied eagerly when trying to construct the proof-tree of a given sequent; (*Store*) is applied when hitting a literal or a positive formula on the right-hand side of a sequent, storing its negation on the left.

When the right-hand side of a sequent becomes empty, a sanity check can be made with (*Init*$_2$) to check the semantical consistency of the stored (positive) literals (w.r.t. the theory), otherwise a choice must be made to place a formula in focus which is not $\mathcal{P}$-negative, before applying synchronous rules like $(\wedge^+)$ and $(\vee^+)$. Each such rule decomposes the formula in focus, keeping the revealed sub-formulae in the focus of the corresponding premises, until a positive literal or a non-positive formula is obtained: the former case must be closed immediately with (Init_1) calling the decision procedure, and the latter case uses the $(\mathit{Release})$ rule to drop the focus and start applying asynchronous rules again. The synchronous and the structural rules are in general not invertible, and each application of those yields a potential backtrack point in the proof-search.

Remark 6 The polarisation of literals (if not already polarised) happens in the (Store) rule, where the construction $\mathcal{P};A$ plays a crucial role. It will be useful to notice the commutation $\mathcal{P};A;B = \mathcal{P};B;A$ unless $A = B^{\perp} \in \mathsf{U}_{\mathcal{P}}$.

Example 1 Consider $\mathcal{T}$ to be the theory of integer (or boolean) arithmetic.

Let $\Gamma := \{(x = 0 \wedge^- \top^-), (x = 1 \wedge^- \top^-)\}$, and assume that each of the literals mentioned in Γ is polarised by $\mathcal{P} := \{x = 0, x = 1\}$.

We can build the following proof-tree:

$$\begin{array}{c}
\hline
\Gamma, x = 0 \vdash^{\mathcal{P}} [x \neq 1]\\
\hline
\Gamma, x = 0 \vdash^{\mathcal{P}} [x \neq 1 \vee^+ \bot^+]\\
\hline
\Gamma, x = 0 \vdash^{\mathcal{P}}\\
\hline
\Gamma \vdash^{\mathcal{P}} x \neq 0\\
\hline
\Gamma \vdash^{\mathcal{P}} [x \neq 0]\\
\hline
\Gamma \vdash^{\mathcal{P}} [x \neq 0 \vee^+ \bot^+]\\
\hline
\Gamma \vdash^{\mathcal{P}}
\end{array}$$

as $x = 0, x = 1 \models_{\mathcal{T}}$.

Now change the polarisation set to $\mathcal{P} := \{x \neq 0, x \neq 1\}$.

It may seem counter-intuitive but it is rather straightforward to realise that the sequent $\Gamma \vdash^{\mathcal{P}}$ is no longer provable:

Since there are no literals in Γ, we cannot immediately close the branch with (Init_2), so the only way to construct a proof would be by placing the focus on either $(x \neq 0 \vee^+ \bot^+)$ or $(x \neq 1 \vee^+ \bot^+)$; then we would need to choose the left-hand side and, because of $\mathcal{P}$, we would then be forced to apply (Init_1), which would require either $x = 0$ or $x = 1$ to be semantically inconsistent just on its own.

From this we conclude that the polarity of literals affects the provability of (even unfocused) sequents, which is different from the case of the empty theory: in Liang-Miller's LKF [LM09], the polarities of connectives and literals do not affect provability, they only affect the shape of proofs.

We shall see in Section 3.7 what meaningful notion of completeness we can state for $\mathsf{LK}^p(\mathcal{T})$, given that phenomenon.

At this point, it is also interesting to understand why, in rules (Init_1) and (Init_2), we choose to only close the branch by finding a semantical inconsistency that involves the $\mathcal{P}$-positive literals of Γ and that ignores its $\mathcal{P}$-negative or $\mathcal{P}$-unpolarised literals.

Example 2 As in the previous example, let $\Gamma := \{(x = 0 \wedge^- \top^-), (x = 1 \wedge^- \top^-)\}$, and $\mathcal{P} := \{x \neq 0, x \neq 1\}$.

While $\Gamma \vdash^{\mathcal{P}}$ is not provable in the (*cut-free*) system $\mathsf{LK}^p(\mathcal{T})$, the situation may be different if we can use a *cut*-rule so that, in order to prove $\Gamma \vdash^{\mathcal{P}}$, it suffices to prove $\Gamma, x \neq 0 \vdash^{\mathcal{P}}$ and $\Gamma, x = 0 \vdash^{\mathcal{P}}$: as before, the former can be proved with

$$\begin{array}{c}
\hline
\Gamma, x \neq 0 \vdash^{\mathcal{P}} [x \neq 0]\\
\hline
\Gamma, x \neq 0 \vdash^{\mathcal{P}} [x \neq 0 \vee^+ \bot^+]\\
\hline
\Gamma, x \neq 0 \vdash^{\mathcal{P}}
\end{array}$$

while we can construct for the latter the following proof-tree:

$$\dfrac{\dfrac{\Gamma, x=0 \vdash^{\mathcal{P}} [x \neq 1]}{\Gamma, x=0 \vdash^{\mathcal{P}} [x \neq 1 \vee^{+} \perp^{+}]}}{\Gamma, x=0 \vdash^{\mathcal{P}}}$$

Whether or not this proof-tree can be closed by (Init_1) depends on whether or not we may involve the ($\mathcal{P}$-negative) literal $x = 0$ to find a semantical inconsistency. If we may, then the system with cuts proves a sequent that the system without cuts cannot prove.

We prefer to guarantee the cut-elimination property and therefore disallow $x = 0$ to be involved in the semantical inconsistency, on the grounds that it is $\mathcal{P}$-negative.

In Section 3.5 we shall see that, owing to our choice of ignoring in (Init_1) and (Init_2) those literals of Γ that are $\mathcal{P}$-negative or $\mathcal{P}$-unpolarised, cuts *are* admissible (at least when all literals are polarised, as it is the case here).

3.2 Admissibility of basic rules

In this section, we show the admissibility and invertibility of some rules, in order to prove the meta-theory of $\mathsf{LK}^p(\mathcal{T})$.

Lemma 7 (Weakening and contraction) The following rules are height-preserving admissible in $\mathsf{LK}^p(\mathcal{T})$:

$$(\mathsf{W}_l)\dfrac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma, A \vdash^{\mathcal{P}} \Delta} \qquad (\mathsf{W}_f)\dfrac{\Gamma \vdash^{\mathcal{P}} [B]}{\Gamma, A \vdash^{\mathcal{P}} [B]}$$

$$(\mathsf{C}_l)\dfrac{\Gamma, A, A \vdash^{\mathcal{P}} \Delta}{\Gamma, A \vdash^{\mathcal{P}} \Delta} \qquad (\mathsf{C}_f)\dfrac{\Gamma, A, A \vdash^{\mathcal{P}} [B]}{\Gamma, A \vdash^{\mathcal{P}} [B]} \qquad (\mathsf{C}_r)\dfrac{\Gamma \vdash^{\mathcal{P}} \Delta, A, A}{\Gamma \vdash^{\mathcal{P}} \Delta, A}$$

※

Proof: By induction on the derivation of the premiss. □

Lemma 8 (Identities) The identity rules are admissible in $\mathsf{LK}^p(\mathcal{T})$:

$$(\mathsf{Id}_1)\dfrac{}{\Gamma, l \vdash^{\mathcal{P}} [l]}\ l \text{ is } \mathcal{P}\text{-positive} \qquad (\mathsf{Id}_2)\dfrac{}{\Gamma, l, l^{\perp} \vdash^{\mathcal{P}}}$$

※

Proof: It is trivial to prove Id_1.

If l or $l^{\perp}$ is $\mathcal{P}$-positive, the Id_2 rule can be obtained by a derivation of the following form:

$$\dfrac{(\mathsf{Id}_1)\dfrac{}{\Gamma, l, l^{\perp} \vdash^{\mathcal{P}} [l]}}{\Gamma, l, l^{\perp} \vdash^{\mathcal{P}}}$$

where l is assumed to be the $\mathcal{P}$-positive literal.

If $l \in \mathsf{U}_{\mathcal{P}}$, we polarise it positively with

$$\dfrac{(\mathsf{Release})\dfrac{(\mathsf{Store})\dfrac{\Gamma, l, l^{\perp}, l \vdash^{\mathcal{P}, l}}{\Gamma, l, l^{\perp} \vdash^{\mathcal{P}} l^{\perp}}}{\Gamma, l, l^{\perp} \vdash^{\mathcal{P}} [l^{\perp}]}}{\Gamma, l, l^{\perp} \vdash^{\mathcal{P}}}$$

□

3.3 Invertibility of the asynchronous phase

We have mentioned that the asynchronous rules are invertible; now in this section, we prove it.

LEMMA 9 (Invertibility of asynchronous rules) All asynchronous rules are invertible in $\mathsf{LK}^p(\mathcal{T})$. ※

Proof: By induction on the derivation proving the conclusion of the asynchronous rule considered.

- Inversion of $A\wedge^- B$: by case analysis on the last rule actually used
  - ($\wedge^-$)

$$\frac{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} A\wedge^- B, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, C\wedge^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, \Delta' \qquad \Gamma \vdash^{\mathcal{P}} A, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C\wedge^- D, \Delta'}$$

and

$$\frac{\Gamma \vdash^{\mathcal{P}} B, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} B, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} B, C\wedge^- D, \Delta'}$$

  - ($\vee^-$)

$$\frac{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, C\vee^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C\vee^- D, \Delta'}$$

and

$$\frac{\Gamma \vdash^{\mathcal{P}} B, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} B, C\vee^- D, \Delta'}$$

  - ($\forall$)

$$\frac{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, C, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, (\forall x C), \Delta'}\, x \notin \mathsf{FV}(\Gamma, \Delta', A\wedge^- B)$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, (\forall x C), \Delta'}\, x \notin \mathsf{FV}(\Gamma, \Delta', A)$$

and

$$\frac{\Gamma \vdash^{\mathcal{P}} B, C, \Delta'}{\Gamma \vdash^{\mathcal{P}} B, (\forall x C), \Delta'}\, x \notin \mathsf{FV}(\Gamma, \Delta', B)$$

  - (Store)

$$\frac{\Gamma, C^\perp \vdash^{\mathcal{P};C^\perp} A\wedge^- B, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, C, \Delta'}\, C \text{ literal or } \mathcal{P}\text{-positive formula}$$

By induction hypothesis we get

$$\frac{\Gamma, C^\perp \vdash^{\mathcal{P};C^\perp} A, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C, \Delta'}\, C \text{ literal or } \mathcal{P}\text{-positive formula}$$

and

$$\frac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} B, \Delta'}{\Gamma \vdash^{\mathcal{P}} B, C, \Delta'}\ C \text{ literal or } \mathcal{P}\text{-positive formula}$$

– $(\perp^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, \perp^-, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, \perp^-, \Delta'}$$

and

$$\frac{\Gamma \vdash^{\mathcal{P}} B, \Delta'}{\Gamma \vdash^{\mathcal{P}} B, \perp^-, \Delta'}$$

– $(\top^-)$

$$\frac{}{\Gamma \vdash^{\mathcal{P}} A\wedge^- B, \top^-, \Delta'}$$

We get

$\frac{}{\Gamma \vdash^{\mathcal{P}} A, \top^-, \Delta'}$ and $\frac{}{\Gamma \vdash^{\mathcal{P}} B, \top^-, \Delta'}$

- Inversion of $A\vee^- B$: by case analysis on the last rule

– $(\wedge^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} A\vee^- B, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} A\vee^- B, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\vee^- B, C\wedge^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, B, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} A, B, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, B, C\wedge^- D, \Delta'}$$

– $(\vee^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} A\vee^- B, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\vee^- B, C\vee^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, B, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, B, C\vee^- D, \Delta'}$$

– $(\forall\ )$

$$\frac{\Gamma \vdash^{\mathcal{P}} A\vee^- B, C, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\vee^- B, (\forall x C), \Delta'}\ x \notin \mathsf{FV}(\Gamma, \Delta')$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, B, C, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, B, (\forall x C), \Delta'}\ x \notin \mathsf{FV}(\Gamma, \Delta')$$

– (Store)

$$\frac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} A\vee^- B, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\vee^- B, C, \Delta'}\ C \text{ literal or } \mathcal{P}\text{-postive formula}$$

By induction hypothesis we get

$$\frac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} A, B, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, B, C, \Delta'} \ C \text{ literal or } \mathcal{P}\text{-positive formula}$$

– $(\perp^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} A\vee^- B, \Delta'}{\Gamma \vdash^{\mathcal{P}} A\vee^- B, \perp^-, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, B, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, B, \perp^-, \Delta'}$$

– $(\top^-)$

$$\frac{}{\Gamma \vdash^{\mathcal{P}} A\vee^- B, \top^-, \Delta'}$$

We get

$$\frac{}{\Gamma \vdash^{\mathcal{P}} A, B, \top^-, \Delta'}$$

- Inversion of $\forall x A$: by case analysis on the last rule

– $(\wedge^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} (\forall x A), C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} (\forall x A), D, \Delta'}{\Gamma \vdash^{\mathcal{P}} (\forall x A), C\wedge^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} A, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C\wedge^- D, \Delta'} \ x \notin \mathsf{FV}(\Gamma, \Delta')$$

– $(\vee^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} (\forall x A), C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} (\forall x A), C\vee^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C\vee^- D, \Delta'}$$

– $(\forall\ )$

$$\frac{\Gamma \vdash^{\mathcal{P}} (\forall x A), D, \Delta'}{\Gamma \vdash^{\mathcal{P}} (\forall x A), (\forall x D), \Delta'} \ x \notin \mathsf{FV}(\Gamma, \Delta')$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, (\forall x D), \Delta'} \ x \notin \mathsf{FV}(\Gamma, \Delta')$$

– (Store)

$$\frac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} (\forall x A), \Delta'}{\Gamma \vdash^{\mathcal{P}} (\forall x A), C, \Delta'} \ C \text{ literal or } \mathcal{P}\text{-positive formula}$$

By induction hypothesis we get

$$\frac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} A, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C, \Delta'} \ C \text{ literal or } \mathcal{P}\text{-positive formula}$$

– ($\bot^-$)

$$\frac{\Gamma \vdash^{\mathcal{P}} (\forall x A), \Delta'}{\Gamma \vdash^{\mathcal{P}} (\forall x A), \bot^-, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, \bot^-, \Delta'}$$

– ($\top^-$)

$$\frac{}{\Gamma \vdash^{\mathcal{P}} (\forall x A), \top^-, \Delta'}$$

We get

$$\frac{}{\Gamma \vdash^{\mathcal{P}} A, \top^-, \Delta'}$$

- Inversion of (Store): where A is a literal or $\mathcal{P}$-positive formula.
By case analysis on the last rule

– ($\wedge^-$)

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} A, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C \wedge^- D, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} C, \Delta' \quad \Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} D, \Delta'}{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} C \wedge^- D, \Delta'}$$

– ($\vee^-$)

$$\frac{\Gamma \vdash^{\mathcal{P}} A, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, C \vee^- D, \Delta'}$$

By induction hypothesis

$$\frac{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} C, D, \Delta'}{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} C \vee^- D, \Delta'}$$

– ($\forall$)

$$\frac{\Gamma \vdash^{\mathcal{P}} A, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, (\forall x D), \Delta'}\ x \notin \mathsf{FV}(\Gamma, \Delta')$$

By induction hypothesis we get

$$\frac{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} D, \Delta'}{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} (\forall x D), \Delta'}\ x \notin \mathsf{FV}(\Gamma, \Delta')$$

– (Store)

$$\frac{\Gamma, B^\bot \vdash^{\mathcal{P};B^\bot} A, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, B, \Delta'}\ B \text{ literal or } \mathcal{P}\text{-positive formula}$$

By induction hypothesis we can construct:

$$\frac{\Gamma, A^\bot, B^\bot \vdash^{\mathcal{P};B^\bot;A^\bot} \Delta'}{\Gamma, A^\bot \vdash^{\mathcal{P};A^\bot} B, \Delta'}$$

provided $\mathcal{P}; B^\bot; A^\bot = \mathcal{P}; A^\bot; B^\bot$, which is always the case unless $A = B^\bot$ and $A \in \mathsf{U}_{\mathcal{P}}$, in which case we build:

$$\dfrac{(\mathsf{Id}_2)\dfrac{}{\Gamma, A^\perp, B^\perp \vdash^{\mathcal{P};A^\perp;B^\perp} \Delta'}}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} B, \Delta}$$

– $(\perp^-)$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} A, \Delta'}{\Gamma \vdash^{\mathcal{P}} A, \perp^-, \Delta'}$$

By induction hypothesis we get

$$\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \Delta'}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \perp^-, \Delta'}$$

– $(\top^-)$

$$\dfrac{}{\Gamma \vdash^{\mathcal{P}} A, \top^-, \Delta'}$$

We get

$$\dfrac{}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \top^-, \Delta'}$$

- Inversion of $(\perp^-)$: by case analysis on the last rule

– $(\wedge^-)$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} \perp^-, C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} \perp^-, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} \perp^-, C \wedge^- D, \Delta'}$$

By induction hypothesis we get

$$\dfrac{\Gamma \vdash^{\mathcal{P}} C, \Delta' \quad \Gamma \vdash^{\mathcal{P}} D, \Delta'}{\Gamma \vdash^{\mathcal{P}} C \wedge^- D, \Delta'}$$

– $(\vee^-)$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} \perp^-, C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} \perp^-, C \vee^- D, \Delta'}$$

By induction hypothesis

$$\dfrac{\Gamma \vdash^{\mathcal{P}} C, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} C \vee^- D, \Delta'}$$

– $(\forall\ )$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} \perp^-, D, \Delta'}{\Gamma \vdash^{\mathcal{P}} \perp^-, (\forall x D), \Delta'}\ x \notin \mathsf{FV}(\Gamma, \Delta')$$

By induction hypothesis we get

$$\dfrac{\Gamma \vdash^{\mathcal{P}} D, \Delta'}{\Gamma \vdash^{\mathcal{P}} (\forall x D), \Delta'}\ x \notin \mathsf{FV}(\Gamma, \Delta')$$

– (Store)

$$\dfrac{\Gamma, B^\perp \vdash^{\mathcal{P};B^\perp} \perp^-, \Delta'}{\Gamma \vdash^{\mathcal{P}} \perp^-, B, \Delta'}\ B \text{ literal or } \mathcal{P}\text{-positive formula}$$

By induction hypothesis we get

$$\dfrac{\Gamma, B^\perp \vdash^{\mathcal{P};B^\perp} \Delta'}{\Gamma \vdash^{\mathcal{P}} B, \Delta'}\ B \text{ literal or } \mathcal{P}\text{-positive formula}$$

– $(\bot^-)$

$$\frac{\Gamma \vdash^{\mathcal{P}} \bot^-, \Delta'}{\Gamma \vdash^{\mathcal{P}} \bot^-, \bot^-, \Delta'}$$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} \Delta'}{\Gamma \vdash^{\mathcal{P}} \bot^-, \Delta'}$$

– $(\top^-)$

$$\frac{}{\Gamma \vdash^{\mathcal{P}} \top^-, \bot^-, \Delta'}$$

We get

$$\frac{}{\Gamma \vdash^{\mathcal{P}} \top^-, \Delta'}$$

- Inversion of $(\top^-)$: Nothing to do.

□

3.4 On-the-fly polarisation

The side-conditions of the $\mathsf{LK}^p(\mathcal{T})$ rules make it quite clear that the polarisation of literals plays a crucial role in the shape of proofs. The less flexible the polarisation of literals is, the more structure is imposed on proofs. We therefore concentrated the polarisation of literals in just one rule: (Store). In this section, we describe more flexible ways of changing the polarity of literals without modifying the provability of sequents. We do this by showing the admissibility and invertibility of some "on-the-fly" polarisation rules.

LEMMA 10 (Invertibility) The following rules are invertible in $\mathsf{LK}^p(\mathcal{T})$:

$$(\mathsf{Pol})\frac{\Gamma \vdash^{\mathcal{P},l} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\ \mathsf{lit}_{\mathcal{P},l}(\Gamma, \Delta^{\bot}), l^{\bot} \models_{\mathcal{T}} \qquad (\mathsf{Pol}_i)\frac{\Gamma \vdash^{\mathcal{P},l} [A]}{\Gamma \vdash^{\mathcal{P}} [A]}\ \mathsf{lit}_{\mathcal{P},l}(\Gamma), l^{\bot} \models_{\mathcal{T}}$$

where $l \in \mathsf{U}_{\mathcal{P}}$. ※

Proof: By simultaneous induction on the derivation of the conclusion (by case analysis on the last rule used in that derivation):

- $(\wedge^-)$,$(\vee^-)$,$(\forall\ )$,$(\bot^-)$,$(\top^-)$
  For these rules, whatever is done with the polarisation set $\mathcal{P}$ can be done with the polarisation set $\mathcal{P}, l$:

$$\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \quad \Gamma \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash^{\mathcal{P}} A \wedge^- B, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P},l} A, \Delta \quad \Gamma \vdash^{\mathcal{P},l} B, \Delta}{\Gamma \vdash^{\mathcal{P},l} A \wedge^- B, \Delta}$$

$$\frac{\Gamma \vdash^{\mathcal{P}} A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} A \wedge^- B, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P},l} A, B, \Delta}{\Gamma \vdash^{\mathcal{P},l} A \vee^- B, \Delta}$$

$$\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta}{\Gamma \vdash^{\mathcal{P}} \forall x A, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P},l} A, \Delta}{\Gamma \vdash^{\mathcal{P},l} \forall x A, \Delta}$$

$$\frac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \bot^-, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P},l} \Delta}{\Gamma \vdash^{\mathcal{P},l} \bot^-, \Delta}$$

$$\frac{}{\Gamma \vdash^{\mathcal{P}} \top^-, \Delta} \qquad \frac{}{\Gamma \vdash^{\mathcal{P},l} \top^-, \Delta}$$

- (Store): We assume

$$\frac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}\ A \text{ is a literal or is } \mathcal{P}\text{-positive}$$

  Notice that A is either a literal or a $\mathcal{P}, l$-positive formula, so we can prove

$$\frac{\Gamma, A^\perp \vdash^{\mathcal{P},l;A^\perp} \Delta}{\Gamma \vdash^{\mathcal{P},l} A, \Delta}$$

  provided we can prove the premiss.

  - If $A \neq l$, then $\mathcal{P}, l; A^\perp = \mathcal{P}; A^\perp, l$ and applying the induction hypothesis finishes the proof (unless $A = l^\perp$ in which case the derivable sequent $\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \Delta$ is the same as the premiss to be proved);
  - If $A = l$, we build

$$(\mathsf{Store})\dfrac{(\mathsf{Store})\dfrac{\dfrac{(\mathsf{Init}_1)\dfrac{\mathsf{lit}_{\mathcal{P}',l}(\Gamma, l^\perp, \Gamma'), l^\perp \models_{\mathcal{T}}}{\Gamma, l^\perp, \Gamma' \vdash^{\mathcal{P}',l} [l]}}{\Gamma, l^\perp, \Gamma' \vdash^{\mathcal{P}',l}}}{\Gamma, \Gamma' \vdash^{\mathcal{P}',l} l}}{\Gamma \vdash^{\mathcal{P},l} l, \Delta}$$

  for some $\mathcal{P}' \supseteq \mathcal{P}$ and some $\Gamma' \supseteq \mathsf{lit}_{\mathcal{L}}(\Delta^\perp)$. The closing condition $\mathsf{lit}_{\mathcal{P}',l}(\Gamma, l^\perp, \Gamma'), l^\perp \models_{\mathcal{T}}$ holds, since $\mathsf{lit}_{\mathcal{P},l}(\Gamma, l^\perp, \Delta^\perp), l^\perp \subseteq \mathsf{lit}_{\mathcal{P}',l}(\Gamma, l^\perp, \mathsf{lit}_{\mathcal{L}}(\Delta^\perp)), l^\perp$ is assumed inconsistent.

- (Select): We assume

$$\frac{\Gamma \vdash^{\mathcal{P}} [A]}{\Gamma \vdash^{\mathcal{P}}}\ \begin{array}{l} A \text{ is not } \mathcal{P}\text{-negative} \\ A^\perp \in \Gamma \end{array}$$

  - If $A \neq l^\perp$, then A is not $\mathcal{P}, l$-negative and we can use the induction hypothesis (invertibility of Pol_i) to construct:

$$\frac{\Gamma \vdash^{\mathcal{P},l} [A]}{\Gamma \vdash^{\mathcal{P},l}}$$

  - If $A = l^\perp$, then $l \in \Gamma$ and the hypothesis can only be derived by

$$\dfrac{\dfrac{\dfrac{\Gamma, l \vdash^{\mathcal{P},l}}{\Gamma \vdash^{\mathcal{P}} l^\perp}}{\Gamma \vdash^{\mathcal{P}} [l^\perp]}}{\Gamma \vdash^{\mathcal{P}}}$$

  as $\mathcal{P}; l = \mathcal{P}, l$; then we can construct:

$$(\mathsf{C}_l)\frac{\Gamma, l \vdash^{\mathcal{P},l}}{\Gamma \vdash^{\mathcal{P},l}}$$

- (Init_2): We assume

$$\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

  We build

$$\frac{\mathsf{lit}_{\mathcal{P},l}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P},l}}$$

- $(\wedge^+)$,$(\vee^+)$,$(\exists\ )$,$(\top^+)$
  Again, for these rules, whatever is done with the polarisation set $\mathcal{P}$ can be done with the polarisation set $\mathcal{P}, l$:

$$\frac{\Gamma\vdash^{\mathcal{P}}[A]\quad\Gamma\vdash^{\mathcal{P}}[B]}{\Gamma\vdash^{\mathcal{P}}[A\wedge^+B]}\qquad\frac{\Gamma\vdash^{\mathcal{P},l}[A]\quad\Gamma\vdash^{\mathcal{P},l}[B]}{\Gamma\vdash^{\mathcal{P},l}[A\wedge^+B]}$$

$$\frac{\Gamma\vdash^{\mathcal{P}}[A_i]}{\Gamma\vdash^{\mathcal{P}}[A_1\vee^+A_2]}\qquad\frac{\Gamma\vdash^{\mathcal{P},l}[A_i]}{\Gamma\vdash^{\mathcal{P},l}[A_1\vee^+A_2]}$$

$$\frac{\Gamma\vdash^{\mathcal{P}}[\{{}^t\!/_x\}A]}{\Gamma\vdash^{\mathcal{P}}[\exists xA]}\qquad\frac{\Gamma\vdash^{\mathcal{P},l}[\{{}^t\!/_x\}A]}{\Gamma\vdash^{\mathcal{P},l}[\exists xA]}$$

$$\frac{}{\Gamma\vdash^{\mathcal{P}}[\top^+]}\qquad\frac{}{\Gamma\vdash^{\mathcal{P},l}[\top^+]}$$

- (Release): We assume

$$\frac{\Gamma\vdash^{\mathcal{P}}A}{\Gamma\vdash^{\mathcal{P}}[A]}\ A\text{ is not }\mathcal{P}\text{-positive}$$

  - If $A\neq l$, then we build:

$$\frac{\Gamma\vdash^{\mathcal{P},l}A}{\Gamma\vdash^{\mathcal{P},l}[A]}$$

    since A is not $\mathcal{P},l$-positive, and we close the branch by applying the induction hypothesis (invertibility of Pol), whose side-condition $\mathsf{lit}_{\mathcal{P},l}(\Gamma,A^\perp),l^\perp\models_{\mathcal{T}}$ is implied by $\mathsf{lit}_{\mathcal{P},l}(\Gamma),l^\perp\models_{\mathcal{T}}$.

  - if $A=l$ then we build

$$\frac{\mathsf{lit}_{\mathcal{P},l}(\Gamma),l^\perp\models_{\mathcal{T}}}{\Gamma\vdash^{\mathcal{P},l}[l]}$$

    where $\mathsf{lit}_{\mathcal{P},l}(\Gamma),l^\perp\models_{\mathcal{T}}$ is the side-condition of (Pol_i) that we have assumed.

- (Init_1) We assume

$$\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma),l'^\perp\models_{\mathcal{T}}}{\Gamma\vdash^{\mathcal{P}}[l']}\ l'\text{ is }\mathcal{P}\text{-positive}$$

  We build:

$$\frac{\mathsf{lit}_{\mathcal{P},l}(\Gamma),l'^\perp\models_{\mathcal{T}}}{\Gamma\vdash^{\mathcal{P},l}[l']}$$

  since l' is $\mathcal{P},l$-positive.

□

Corollary 11 The following rules are admissible in $\mathsf{LK}^p(\mathcal{T})$:

$$(\mathsf{Store}^=)\frac{\Gamma,A^\perp\vdash^{\mathcal{P}}\Delta}{\Gamma\vdash^{\mathcal{P}}A,\Delta}\ \begin{array}{l}A\text{ is a literal or}\\ \text{a }\mathcal{P}\text{-positive formula}\end{array}\qquad(\mathsf{W}_r)\frac{\Gamma\vdash^{\mathcal{P}}\Delta}{\Gamma\vdash^{\mathcal{P}}\Delta,\Delta'}$$

※

Proof: For the first rule: if A is polarised, we use (Store) and it does not change $\mathcal{P}$; otherwise A is an unpolarised literal l and we build

$$(\mathsf{Store})\frac{\dfrac{\Gamma,l^\perp\vdash^{\mathcal{P}}\Delta}{\Gamma,l^\perp\vdash^{\mathcal{P},l^\perp}\Delta}}{\Gamma\vdash^{\mathcal{P}}l,\Delta}$$

The topmost inference is the invertibility of (Pol), given that $\mathsf{lit}_{\mathcal{P},l^\perp}(\Gamma,l^\perp),l\models_{\mathcal{T}}$.

For the second case, we simply do a multiset induction on Δ', using rule $(\mathsf{Store}^=)$ for the base case, followed by a left weakening. □

Now we can show that removing polarities is admissible:

LEMMA 12 (Admissibility) The following rules are admissible in $\mathsf{LK}^p(\mathcal{T})$:

$$(\mathsf{Pol})\frac{\Gamma \vdash^{\mathcal{P},l} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta} \qquad (\mathsf{Pol}_a)\frac{\Gamma \vdash^{\mathcal{P},l} [A]}{\Gamma \vdash^{\mathcal{P}} [A]}\ l \notin \Gamma \text{ or } \mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}}$$

where $l \in \mathsf{U}_{\mathcal{P}}$. ※

Proof: By a simultaneous induction on the derivation of the premiss, again by case analysis on the last rule used in the assumed derivation.

- $(\wedge^-)$,$(\vee^-)$,$(\forall\ )$,$(\perp^-)$,$(\top^-)$
  For these rules, whatever is done with the polarisation set $\mathcal{P}, l$ can be done with the polarisation set $\mathcal{P}$:

$$\frac{\Gamma \vdash^{\mathcal{P},l} A, \Delta \quad \Gamma \vdash^{\mathcal{P},l} B, \Delta}{\Gamma \vdash^{\mathcal{P},l} A \wedge^- B, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \quad \Gamma \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash^{\mathcal{P}} A \wedge^- B, \Delta}$$

$$\frac{\Gamma \vdash^{\mathcal{P},l} A, B, \Delta}{\Gamma \vdash^{\mathcal{P},l} A \vee^- B, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P}} A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} A \wedge^- B, \Delta}$$

$$\frac{\Gamma \vdash^{\mathcal{P},l} A, \Delta}{\Gamma \vdash^{\mathcal{P},l} \forall x A, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P}} A, \Delta}{\Gamma \vdash^{\mathcal{P}} \forall x A, \Delta}$$

$$\frac{\Gamma \vdash^{\mathcal{P},l} \Delta}{\Gamma \vdash^{\mathcal{P},l} \perp^-, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \perp^-, \Delta}$$

$$\frac{}{\Gamma \vdash^{\mathcal{P},l} \top^-, \Delta} \qquad \frac{}{\Gamma \vdash^{\mathcal{P}} \top^-, \Delta}$$

- (Store): We assume

$$\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P},l;A^{\perp}} \Delta}{\Gamma \vdash^{\mathcal{P},l} A, \Delta}\ A \text{ is a literal or } \mathcal{P},l\text{-positive}$$

  Notice that A is either a literal or a $\mathcal{P}$-positive formula.
  - If $A = l^{\perp}$, we build

$$(\mathsf{Store})\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P},A^{\perp}} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}$$

    whose premiss is the derivable sequent $\Gamma, A^{\perp} \vdash^{\mathcal{P},l;A^{\perp}} \Delta$.
  - If $A = l$, we build

$$(\mathsf{Store}^{=})\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}$$

    using the admissibility of $\mathsf{Store}^{=}$, and we can prove the premiss from the induction hypothesis, as we have $\mathcal{P}, l; A^{\perp} = \mathcal{P}, l$.
  - In all other cases, we build

$$(\mathsf{Store})\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P};A^{\perp}} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}$$

    whose premiss is provable from the induction hypothesis, as we have $\mathcal{P}, l; A^{\perp} = \mathcal{P}; A^{\perp}, l$.
- (Select): We assume

$$\frac{\Gamma \vdash^{\mathcal{P},l} [A]}{\Gamma \vdash^{\mathcal{P},l}}\ \begin{array}{l} A^{\perp} \in \Gamma \\ \text{and } A \text{ is not } \mathcal{P},l\text{-negative}\end{array}$$

  - If $l \in \Gamma$ then we can build:

$$\dfrac{\dfrac{\dfrac{(\mathsf{W}_l)\dfrac{\Gamma \vdash^{\mathcal{P};l}}{\Gamma, l \vdash^{\mathcal{P};l}}}{\Gamma \vdash^{\mathcal{P}} l^{\perp}}}{\Gamma \vdash^{\mathcal{P}} [l^{\perp}]}}{\Gamma \vdash^{\mathcal{P}}}$$

and we close with the assumption since $\mathcal{P}; l = \mathcal{P}, l$.

– If $l \notin \Gamma$ then using the induction hypothesis (admissibility of Pol_a) we construct :

$$\dfrac{\Gamma \vdash^{\mathcal{P}} [A]}{\Gamma \vdash^{\mathcal{P}}}$$

since A is not $\mathcal{P}$-negative.

- (Init_2): We assume

$$\dfrac{\mathsf{lit}_{\mathcal{P},l}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P},l}}$$

– If $l \in \Gamma$ then again we can build:

$$\dfrac{\dfrac{\dfrac{(\mathsf{W}_l)\dfrac{\Gamma \vdash^{\mathcal{P};l}}{\Gamma, l \vdash^{\mathcal{P};l}}}{\Gamma \vdash^{\mathcal{P}} l^{\perp}}}{\Gamma \vdash^{\mathcal{P}} [l^{\perp}]}}{\Gamma \vdash^{\mathcal{P}}}$$

and we close with the assumption since $\mathcal{P}; l = \mathcal{P}, l$.

– If $l \notin \Gamma$, $\mathsf{lit}_{\mathcal{P},l}(\Gamma) = \mathsf{lit}_{\mathcal{P}}(\Gamma)$, then we can build:

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

- $(\wedge^+)$,$(\vee^+)$,$(\exists\ )$,$(\top^+)$
Again, for these rules, whatever is done with the polarisation set $\mathcal{P}, l$ can be done with the polarisation set $\mathcal{P}$:

$$\dfrac{\Gamma \vdash^{\mathcal{P},l} [A] \quad \Gamma \vdash^{\mathcal{P},l} [B]}{\Gamma \vdash^{\mathcal{P},l} [A \wedge^+ B]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} [A] \quad \Gamma \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]}$$

$$\dfrac{\Gamma \vdash^{\mathcal{P},l} [A_i]}{\Gamma \vdash^{\mathcal{P},l} [A_1 \vee^+ A_2]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]}$$

$$\dfrac{\Gamma \vdash^{\mathcal{P},l} [\{{}^{t}\!/\!_{x}\} A]}{\Gamma \vdash^{\mathcal{P},l} [\exists x A]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\} A]}{\Gamma \vdash^{\mathcal{P}} [\exists x A]}$$

$$\dfrac{}{\Gamma \vdash^{\mathcal{P},l} [\top^+]} \qquad \dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]}$$

- $(\mathsf{Release})$: We assume

$$\dfrac{\Gamma \vdash^{\mathcal{P},l} A}{\Gamma \vdash^{\mathcal{P},l} [A]}\ A \text{ is not } \mathcal{P}, l\text{-positive}$$

By induction hypothesis (admissibility of Pol) we can build:

$$\dfrac{\Gamma \vdash^{\mathcal{P}} A}{\Gamma \vdash^{\mathcal{P}} [A]}$$

- (Init_1): We assume
$$\frac{\mathsf{lit}_{\mathcal{P},l}(\Gamma), {l'}^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P},l} [l']}\ l' \text{ is } \mathcal{P}, l\text{-positive}$$
  – If $l' \neq l$, then l' is $\mathcal{P}$-positive and we can build
$$\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), {l'}^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [l']}$$
    The condition $\mathsf{lit}_{\mathcal{P}}(\Gamma), {l'}^{\perp} \models_{\mathcal{T}}$ holds for the following reasons:
    If $l \notin \Gamma$, then $\mathsf{lit}_{\mathcal{P}}(\Gamma) = \mathsf{lit}_{\mathcal{P},l}(\Gamma)$ and the condition is that of the hypothesis.
    If $l \in \Gamma$, then the side-condition of (Pol_a) implies $\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}}$; moreover, the condition of the hypothesis can be rewritten as $\mathsf{lit}_{\mathcal{P}}(\Gamma), l, {l'}^{\perp} \models_{\mathcal{T}}$; the fact that semantical inconsistency admits cuts then proves the desired condition.
  – If $l' = l$ then we build
$$\cfrac{\cfrac{\Gamma, l^{\perp} \vdash^{\mathcal{P},l^{\perp}}}{\Gamma \vdash^{\mathcal{P}} l}}{\Gamma \vdash^{\mathcal{P}} [l]}$$
    which we close as follows: If $l \in \Gamma$ then we can apply Id_2, otherwise we apply Init_2: the condition $\mathsf{lit}_{\mathcal{P},l^{\perp}}(\Gamma, l^{\perp}) \models_{\mathcal{T}}$ holds because $\mathsf{lit}_{\mathcal{P},l^{\perp}}(\Gamma, l^{\perp}) = \mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} = \mathsf{lit}_{\mathcal{P},l}(\Gamma), l^{\perp}$ and the condition of the hypothesis is $\mathsf{lit}_{\mathcal{P},l}(\Gamma), l^{\perp} \models_{\mathcal{T}}$.

□

COROLLARY 13 The $(\mathsf{Store}^{=})$ rule is invertible, and the (Select^{-}) rule is admissible:

$$(\mathsf{Store}^{=})\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}\ \begin{array}{l}A \text{ is literal}\\ \text{or is } \mathcal{P}\text{-positive}\end{array} \qquad (\mathsf{Select}^{-})\frac{\Gamma, l^{\perp} \vdash^{\mathcal{P},l^{\perp}} [l]}{\Gamma, l^{\perp} \vdash^{\mathcal{P},l^{\perp}}}$$

※

Proof:

$(\mathsf{Store}^{=})$ Using the invertibility of (Store), we get a proof of $\Gamma, A^{\perp} \vdash^{\mathcal{P};A^{\perp}} \Delta$. If A is $\mathcal{P}$-polarised, then $\mathcal{P}; A^{\perp} = \mathcal{P}$ and we are done. Otherwise we have a proof of $\Gamma, A^{\perp} \vdash^{\mathcal{P},A^{\perp}} \Delta$ and we apply the admissibility of (Pol) to conclude.

(Select^{-}) The only proof of $\Gamma, l^{\perp} \vdash^{\mathcal{P},l^{\perp}} [l]$ comes from $\Gamma, l^{\perp}, l^{\perp} \vdash^{\mathcal{P},l^{\perp}}$, then we use the admissibility of contraction.

□

3.5 Cut-elimination

In Chapter 2, we discussed the cut-elimination property in sequent calculus. The cut rule usually allows the encoding of other inference systems into sequent calculus. Therefore the cut-elimination property is often the key property to prove the completeness of a sequent calculus. In particular, this is how we prove completeness of $\mathsf{LK}^p(\mathcal{T})$: using the admissibility of following cut rule:

$$\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \quad \Gamma \vdash^{\mathcal{P}} A^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\ \mathsf{cut}_7$$

To prove that cut_7 is admissible, we will use inductions, relying on the admissibility of six other form of cuts: cut_1, cut_2, cut_3, cut_4, cut_5, cut_6.

Moreover, another form of cut, cut_8, deriving from cut_7, we also be proved admissible and will be used to simulate the $\mathsf{DPLL}(\mathcal{T})$ procedure as proof-search in $\mathsf{LK}^p(\mathcal{T})$ (see Chapter 4)

3.5.1 Cuts with the theory

Theorem 14 (cut_1 and cut_2)

The following rules are admissible in $\mathsf{LK}^p(\mathcal{T})$, assuming $l \notin \mathsf{U}_{\mathcal{P}}$:

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_1 \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [B]}\,\mathsf{cut}_2$$

※

Proof:

By simultaneous induction on the derivation of the right premiss.

We reduce cut_1 by case analysis on the last rule used to prove the right premiss.

- $(\wedge^-)$

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \dfrac{\Gamma, l \vdash^{\mathcal{P}} B, \Delta \quad \Gamma, l \vdash^{\mathcal{P}} C, \Delta}{\Gamma, l \vdash^{\mathcal{P}} B \wedge^- C, \Delta}}{\Gamma \vdash^{\mathcal{P}} B \wedge^- C, \Delta}\,\mathsf{cut}_1$$

reduces to

$$\dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash^{\mathcal{P}} B, \Delta}\,\mathsf{cut}_1 \quad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} C, \Delta}{\Gamma \vdash^{\mathcal{P}} C, \Delta}\,\mathsf{cut}_1}{\Gamma \vdash^{\mathcal{P}} B \wedge^- C, \Delta}$$

- $(\vee^-)$

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \dfrac{\Gamma, l \vdash^{\mathcal{P}} B_1, B_2, \Delta}{\Gamma, l \vdash^{\mathcal{P}} B_1 \vee^- B_2, \Delta}}{\Gamma \vdash^{\mathcal{P}} B_1 \vee^- B_2, \Delta}\,\mathsf{cut}_1 \quad \text{reduces to} \quad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} B_1, B_2, \Delta}{\Gamma \vdash^{\mathcal{P}} B_1, B_2, \Delta}\,\mathsf{cut}_1}{\Gamma \vdash^{\mathcal{P}} B_1 \vee^- B_2, \Delta}$$

- $(\forall\ )$

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \dfrac{\Gamma, l \vdash^{\mathcal{P}} B, \Delta}{\Gamma, l \vdash^{\mathcal{P}} \forall x B, \Delta}}{\Gamma \vdash^{\mathcal{P}} \forall x B, \Delta}\,\mathsf{cut}_1 \quad \text{reduces to} \quad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash^{\mathcal{P}} B, \Delta}\,\mathsf{cut}_1}{\Gamma \vdash^{\mathcal{P}} \forall x B, \Delta}$$

- (Store) where B is a literal or $\mathcal{P}$-positive formula.

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \dfrac{\Gamma, l, B^{\bot} \vdash^{\mathcal{P};B^{\bot}} \Delta}{\Gamma, l \vdash^{\mathcal{P}} B, \Delta}}{\Gamma \vdash^{\mathcal{P}} B, \Delta}\,\mathsf{cut}_1 \quad \text{reduces to} \quad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P};B^{\bot}}(\Gamma, B^{\bot}), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l, B^{\bot} \vdash^{\mathcal{P};B^{\bot}} \Delta}{\Gamma, B^{\bot} \vdash^{\mathcal{P};B^{\bot}} \Delta}\,\mathsf{cut}_1}{\Gamma \vdash^{\mathcal{P}} B, \Delta}$$

We have $\mathsf{lit}_{\mathcal{P};B^{\bot}}(\Gamma, B^{\bot}), l^{\bot} \models_{\mathcal{T}}$ since $\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \subseteq \mathsf{lit}_{\mathcal{P};B^{\bot}}(\Gamma, B^{\bot}), l^{\bot}$ and we assume semantical inconsistency to satisfy weakening.

- $(\bot^-)$

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \dfrac{\Gamma, l \vdash^{\mathcal{P}} \Delta}{\Gamma, l \vdash^{\mathcal{P}} \bot^-, \Delta}}{\Gamma \vdash^{\mathcal{P}} \bot^-, \Delta}\,\mathsf{cut}_1 \quad \text{reduces to} \quad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_1}{\Gamma \vdash^{\mathcal{P}} \bot^-, \Delta}$$

- $(\top^-)$

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\bot} \models_{\mathcal{T}} \quad \dfrac{}{\Gamma, l \vdash^{\mathcal{P}} \top^-, \Delta}}{\Gamma \vdash^{\mathcal{P}} \top^-, \Delta}\,\mathsf{cut}_1 \quad \text{reduces to} \quad \dfrac{}{\Gamma \vdash^{\mathcal{P}} \top^-, \Delta}$$

- (Select) where $P^{\bot} \in \Gamma, l$ and P is not $\mathcal{P}$-negative.

If $P^{\bot} \in \Gamma$,

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P}} [P]}{\Gamma, l \vdash^{\mathcal{P}}}}{\Gamma \vdash^{\mathcal{P}}}\,\mathsf{cut}_1 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} [P]}{\Gamma \vdash^{\mathcal{P}} [P]}\,\mathsf{cut}_2}{\Gamma \vdash^{\mathcal{P}}}$$

If $P^{\perp} = l$, then as P is not $\mathcal{P}$-negative and $l \notin \mathsf{U}_{\mathcal{P}}$ we get that $l^{\perp}$ is $\mathcal{P}$-positive, so

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l \models_{\mathcal{T}}}{\Gamma, l \vdash^{\mathcal{P}} [l^{\perp}]}}{\Gamma, l \vdash^{\mathcal{P}}}}{\Gamma \vdash^{\mathcal{P}}}\,\mathsf{cut}_1 \qquad \text{reduces to} \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}\,\mathsf{Init}_2$$

since semantical inconsistency admits cuts.

- (Init_2)

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l \models_{\mathcal{T}}}{\Gamma, l \vdash^{\mathcal{P}}}}{\Gamma \vdash^{\mathcal{P}}}\,\mathsf{cut}_1 \qquad \text{reduces to} \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

since semantical inconsistency admits cuts.
We reduce cut_2 again by case analysis on the last rule used to prove the right premiss.

- ($\wedge^+$)

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P}} [B] \quad \Gamma, l \vdash^{\mathcal{P}} [C]}{\Gamma, l \vdash^{\mathcal{P}} [B \wedge^+ C]}}{\Gamma \vdash^{\mathcal{P}} [B \wedge^+ C]}\,\mathsf{cut}_2$$

reduces to

$$\dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [B]}\,\mathsf{cut}_2 \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} [C]}{\Gamma \vdash^{\mathcal{P}} [C]}\,\mathsf{cut}_2}{\Gamma \vdash^{\mathcal{P}} [B \wedge^+ C]}$$

- ($\vee^+$)

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P}} [B_i]}{\Gamma, l \vdash^{\mathcal{P}} [B_1 \vee^+ B_2]}}{\Gamma \vdash^{\mathcal{P}} [B_1 \vee^+ B_2]}\,\mathsf{cut}_2 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} [B_i]}{\Gamma \vdash^{\mathcal{P}} [B_i]}\,\mathsf{cut}_2}{\Gamma \vdash^{\mathcal{P}} [B_1 \vee^+ B_2]}$$

- ($\exists$)

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P}} [\{{}^{t}\!/_{x}\}B]}{\Gamma, l \vdash^{\mathcal{P}} [\exists x B]}}{\Gamma \vdash^{\mathcal{P}} [\exists x B]}\,\mathsf{cut}_2 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} [\{{}^{t}\!/_{x}\}B]}{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/_{x}\}B]}\,\mathsf{cut}_2}{\Gamma \vdash^{\mathcal{P}} [\exists x B]}$$

- ($\top^+$)

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{}{\Gamma, l \vdash^{\mathcal{P}} [\top^+]}}{\Gamma \vdash^{\mathcal{P}} [\top^+]}\,\mathsf{cut}_2 \qquad \text{reduces to} \qquad \dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]}$$

- ($\mathsf{Release}$)

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P}} N}{\Gamma, l \vdash^{\mathcal{P}} [N]}}{\Gamma \vdash^{\mathcal{P}} [N]}\,\mathsf{cut}_2 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \Gamma, l \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} N}\,\mathsf{cut}_1}{\Gamma \vdash^{\mathcal{P}} [N]}$$

- (Init_1)

$$\cfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp} \models_{\mathcal{T}} \quad \cfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l, p^{\perp} \models_{\mathcal{T}}}{\Gamma, l \vdash^{\mathcal{P}} [p]}}{\Gamma \vdash^{\mathcal{P}} [p]}\,\mathsf{cut}_2 \qquad \text{reduces to} \qquad \cfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]}$$

since weakening gives $\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp}, p^{\perp} \models_{\mathcal{T}}$ and semantical inconsistency admits cuts.

□

3.5.2 Safety and instantiation

Now we would like to prove the admissibility of other cuts, where both premisses are derived as a judgement of $\mathsf{LK}^p(\mathcal{T})$. Unfortunately, the expected cut-rules are not necessarily admissible unless we consider a notion of *safety*:

Example 3 Consider $\mathcal{T}$ to be the theory of Peano's arithmetic.

Let $\Gamma := \{(x < 1\wedge^{-}\top^{-}), (x = 1\wedge^{-}\top^{-})\}$, and $\mathcal{P} := \{x \neq 1, x \geq 1\}$.

We can build the following proof-trees

$$\cfrac{\cfrac{\cfrac{}{\Gamma, x \neq 0 \vdash^{\mathcal{P},x\neq 0} [x \geq 1]}}{\Gamma, x \neq 0 \vdash^{\mathcal{P},x\neq 0} [x \geq 1\vee^{+}\perp^{+}]}}{\cfrac{\Gamma, x \neq 0 \vdash^{\mathcal{P},x\neq 0}}{\Gamma \vdash^{\mathcal{P}} x = 0}} \qquad \cfrac{\cfrac{\cfrac{}{\Gamma, x = 0 \vdash^{\mathcal{P},x=0} [x \neq 1]}}{\Gamma, x = 0 \vdash^{\mathcal{P},x=0} [x \neq 1\vee^{+}\perp^{+}]}}{\cfrac{\Gamma, x = 0 \vdash^{\mathcal{P},x=0}}{\Gamma \vdash^{\mathcal{P}} x \neq 0}}$$

since, for the former, we have $x \neq 0, x < 1 \models_{\mathcal{T}}$, and for the latter we have $x = 0, x = 1 \models_{\mathcal{T}}$.

A simple cut would then prove $\Gamma \vdash^{\mathcal{P}}$. However, there is no cut-free proof of $\Gamma \vdash^{\mathcal{P}}$: Since there are no literals in Γ, we cannot immediately close the branch with (Init_2), so the only way to construct a proof would be by placing the focus on either $(x \geq 1\vee^{+}\perp^{+})$ or $(x \neq 1\vee^{+}\perp^{+})$; then we would need to choose the left-hand side and, because of $\mathcal{P}$, we would then be forced to apply (Init_1), which would require either $x > 1$ or $x = 1$ to be semantically inconsistent just on its own.

Here, we see that having left $x = 0$ unpolarised in $\mathcal{P}$ allows the construction of a proof with a cut that cannot be eliminated: indeed, we can quickly check that having $x = 0$ in $\mathcal{P}$ would make $\Gamma \vdash^{\mathcal{P}} x = 0$ unprovable, and having $x \neq 0$ in $\mathcal{P}$ would make $\Gamma \vdash^{\mathcal{P}} x \neq 0$ unprovable.

We will see that if no literals are $\mathcal{P}$-polarised, then cuts are admissible, and if all literals are $\mathcal{P}$-polarised (as in Liang-Miller's LKF [LM09]), cuts are also admissible. We therefore introduce a criterion called *safety*, to capture those two situations, and other in-between situations where cuts are admissible. The above example will of course not satisfy safety.

Definition 17 (Safety)

- A pair $(\Gamma, \mathcal{P})$ (of a context and a polarisation set) is said to be *safe* if: for all $\Gamma' \supseteq \Gamma$, for all semantically consistent sets of literals $\mathcal{R}$ with $\mathsf{lit}_{\mathcal{P}}(\Gamma') \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P}}(\Gamma') \cup \mathsf{U}^{\downarrow}_{\mathcal{P}}$, and for all $\mathcal{P}$-positive literal l, if $\mathcal{R}, l^{\perp} \models_{\mathcal{T}}$ then $\mathsf{lit}_{\mathcal{P}}(\Gamma'), l^{\perp} \models_{\mathcal{T}}$.
- A sequent $\Gamma \vdash^{\mathcal{P}} [A]$ (resp. $\Gamma \vdash^{\mathcal{P}} \Delta$) is said to be *safe* if the pair $(\Gamma, \mathcal{P})$ (resp. $((\Gamma, \Delta^{\perp}), \mathcal{P})$) is safe.

※

Remark 15 Safety is a property that is monotonic in its first argument: if $(\Gamma, \mathcal{P})$ is safe and $\Gamma \subseteq \Gamma'$ then $(\Gamma', \mathcal{P})$ is safe (this property is built into the definition by the quantification over Γ').

When restricted to safe sequents, the expected cuts are indeed admissible. In order to show that the safety condition is not very restrictive, we show the following lemma:

Lemma 16 (Cases of safety)

1. Empty theory:
   When the theory is empty (semantical inconsistency coincides with syntactical inconsistency), the safety of $(\Gamma, \mathcal{P})$ means that either $\mathsf{lit}_{\mathcal{P}}(\Gamma)$ is syntactically inconsistent, or every $\mathcal{P}$-positive literal that is an instance of a $\mathcal{P}$-unpolarised literal must be in Γ (i.e. $\mathcal{P} \cap \mathsf{U}^{\downarrow}_{\mathcal{P}} \subseteq \Gamma$).
   In the particular case of propositional logic ($\{{}^{t}\!/\!_{x}\}l = l$ for every $l \in \mathcal{L}$), every sequent is safe.
2. Full polarisation:
   When every literal is polarised ($\mathsf{U}_{\mathcal{P}} = \emptyset$), every sequent (with polarisation set $\mathcal{P}$) is safe.
3. No polarisation:
   When every literal is unpolarised ($\mathsf{U}_{\mathcal{P}} = \mathcal{L}$), every sequent (with polarisation set $\mathcal{P}$) is safe.
4. Safety is an invariant of proof-search:
   for every rule of $\mathsf{LK}^p(\mathcal{T})$, if its conclusion is safe then each of its premisses is safe.

※

Proof:

1. In the case of the empty theory, if $\mathcal{R}$ is consistent then $\mathcal{R}, l^{\perp} \models_{\mathcal{T}}$ means that $l \in \mathcal{R}$, so either $l \in \mathsf{lit}_{\mathcal{P}}(\Gamma')$ or $l \in \mathsf{U}^{\downarrow}_{\mathcal{P}}$; that this should imply $\mathsf{lit}_{\mathcal{P}}(\Gamma'), l^{\perp} \models_{\mathcal{T}}$ means that $l \in \mathsf{lit}_{\mathcal{P}}(\Gamma')$ anyway, unless $\mathsf{lit}_{\mathcal{P}}(\Gamma')$ is syntactically inconsistent. In particular for $\Gamma' = \Gamma$.
   In the case of propositional logic, there are no $\mathcal{P}$-positive literals that are in $\mathsf{U}^{\downarrow}_{\mathcal{P}} = \mathsf{U}_{\mathcal{P}}$, so every sequent is safe.

2. When every literal is polarised ($\mathsf{U}_{\mathcal{P}} = \emptyset$), then $\mathcal{R} = \mathsf{lit}_{\mathcal{P}}(\Gamma')$ and the result is trivial.

3. When every literal is unpolarised ($\mathsf{U}_{\mathcal{P}} = \mathcal{L}$), the property holds trivially.

4. For every rule of $\mathsf{LK}^p(\mathcal{T})$, if its conclusion is safe then each of its premisses is safe.
   Every rule is trivial (considering monotonicity) except (Store), for which it suffices to show:
   Assume $(\Gamma, \mathcal{P})$ is safe and $A \in \Gamma$; then $(\Gamma, (\mathcal{P}; A))$ is safe.
   Consider $\Gamma' \supseteq \Gamma$ and $\mathcal{R}$ such that $\mathsf{lit}_{\mathcal{P};A}(\Gamma') \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P};A}(\Gamma') \cup \mathsf{U}^{\downarrow}_{\mathcal{P};A}$.

   - If $A \in \mathsf{U}_{\mathcal{P}}$, then $\mathcal{P}; A = \mathcal{P}, A$ and the inclusions can be rewritten as
     $$\mathsf{lit}_{\mathcal{P}}(\Gamma'), A \ \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P}}(\Gamma'), A \ \cup \mathsf{U}^{\downarrow}_{\mathcal{P},A}$$
     Since $\mathsf{U}_{\mathcal{P},A} \subseteq \mathsf{U}_{\mathcal{P}}$ we have $\mathsf{U}^{\downarrow}_{\mathcal{P},A} \subseteq \mathsf{U}^{\downarrow}_{\mathcal{P}}$ and therefore
     $$\mathsf{lit}_{\mathcal{P}}(\Gamma') \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P}}(\Gamma') \cup \mathsf{U}^{\downarrow}_{\mathcal{P}}$$
     Hence, $\mathcal{R}$ is a set for which safety of $(\Gamma, \mathcal{P})$ implies $\mathsf{lit}_{\mathcal{P}}(\Gamma'), l^{\perp} \models_{\mathcal{T}}$ for every $l \in \mathcal{P}$ such that $\mathcal{R}, l^{\perp} \models_{\mathcal{T}}$.
     For $l = A$, then trivially $\mathsf{lit}_{\mathcal{P},A}(\Gamma'), l^{\perp} \models_{\mathcal{T}}$ as $A \in \Gamma'$.
   - If $A \notin \mathsf{U}_{\mathcal{P}}$, then $\mathcal{P}; A = \mathcal{P}$ and the result is trivial.

□

Now cut-elimination in presence of quantifiers relies heavily on the fact that, if a proof can be constructed with a free variables x, then it can be replayed when x is instantiated by a particular term throughout the proof. In a polarised world, this is made difficult by the fact that a polarisation set $\mathcal{P}$ (i.e. a set that is syntactically consistent) might not remain a polarisation set after instantiation (i.e. $\{{}^{t}\!/\!_{x}\}\mathcal{P}$ might not be syntactically consistent: imagine $p(x, 3)$ is $\mathcal{P}$-positive and $p(3, x)$ is $\mathcal{P}$-negative, then after substituting 3 for x, what is the polarity of $p(3, 3)$?). Hence, polarities will have to be changed and therefore the exact same proof may not be replayed, but under the hypothesis that the substituted sequent is safe, we manage to reconstruct *some* proof. The first step to prove this is the following lemma:

Lemma 17 (Admissibility of instantiation with the theory) Let x be a variable.
Let t be a term with $x \notin \mathsf{FV}(t)$ and $\mathcal{P}$ be a polarisation set with $x \notin \mathsf{FV}(\mathcal{P})$.
Let $l_1, \ldots, l_n$ be n literals, and $\mathcal{A}$ be a set of literals.
Let $\mathcal{P}_i := \mathcal{P}; l_1; \ldots; l_i$ with $\mathcal{P}_0 := \mathcal{P}$, and similarly let $\mathcal{P}'_i := \mathcal{P}; \{^{t}\!/\!_{x}\}l_1; \ldots; \{^{t}\!/\!_{x}\}l_i$ with $\mathcal{P}'_0 := \mathcal{P}$.
Assume

- for all i such that $1 \leq i \leq n$, we have $l_i \in \Gamma$;
- $(\{^{t}\!/\!_{x}\}\Gamma, \mathcal{P}'_n)$ is safe;
- $\mathsf{lit}_{\mathcal{P}_n}(\Gamma), \mathcal{A} \models_{\mathcal{T}}$.

Then either $\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}\mathcal{A} \models_{\mathcal{T}}$ or $\{^{t}\!/\!_{x}\}\Gamma \vdash^{\mathcal{P}'_n}$ is derivable in $\mathsf{LK}^p(\mathcal{T})$. ※

Proof: Let $\{l'_1, \ldots, l'_m\}$ be the set of literals $\{l \in \mathsf{lit}_{\mathcal{P}_n}(\Gamma) \mid \{^{t}\!/\!_{x}\}l \text{ is not } \mathcal{P}'_n\text{-positive}\}$. We have

$$\{^{t}\!/\!_{x}\}\mathsf{lit}_{\mathcal{P}_n}(\Gamma) \subseteq \mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}l'_1, \ldots, \{^{t}\!/\!_{x}\}l'_m$$

Since $\mathsf{lit}_{\mathcal{P}_n}(\Gamma), \mathcal{A} \models_{\mathcal{T}}$ and semantical inconsistency is stable under instantiation and weakening, we have $\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}l'_1, \ldots, \{^{t}\!/\!_{x}\}l'_m, \{^{t}\!/\!_{x}\}\mathcal{A} \models_{\mathcal{T}}$.

- If all of the sets $(\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}{l'_j}^{\perp})_{1 \leq j \leq n}$ are semantically inconsistent, then from

  $$\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}l'_1, \ldots, \{^{t}\!/\!_{x}\}l'_m, \{^{t}\!/\!_{x}\}\mathcal{A} \models_{\mathcal{T}}$$

  we get $\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}\mathcal{A} \models_{\mathcal{T}}$, since semantically inconsistency admits cuts.

- Otherwise, there is some $l'_j \in \mathsf{lit}_{\mathcal{P}_n}(\Gamma)$ such that $\{^{t}\!/\!_{x}\}l'_j$ is not $\mathcal{P}'_n$-positive and such that $\mathcal{R} := \mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}{l'_j}^{\perp}$ is semantically consistent.

  Notice that l'_j is not $\mathcal{P}$-positive, otherwise $\{^{t}\!/\!_{x}\}l'_j$ would also be $\mathcal{P}$-positive (since $x \notin \mathsf{FV}(\mathcal{P})$), so $l'_j = l_i$ for some i such that $1 \leq i \leq n$, with $l_i \in \mathsf{U}_{\mathcal{P}_{i-1}}$.

  Now, if $\{^{t}\!/\!_{x}\}\Gamma$ is syntactically inconsistent, we build

  $$\mathsf{Id}_2\,\dfrac{}{\{^{t}\!/\!_{x}\}\Gamma \vdash^{\mathcal{P}'_n}}$$

  If on the contrary $\{^{t}\!/\!_{x}\}\Gamma$ is syntactically consistent, then $\{\{^{t}\!/\!_{x}\}l_1, \ldots, \{^{t}\!/\!_{x}\}l_n\}$ is also syntactically consistent (as every element is assumed to be in $\{^{t}\!/\!_{x}\}\Gamma$).

  Therefore, $\{^{t}\!/\!_{x}\}l_i$ must be $\mathcal{P}$-negative, otherwise it would ultimately be $\mathcal{P}'_n$-positive.
  So $\{^{t}\!/\!_{x}\}l_i^{\perp}$ is $\mathcal{P}$-positive, and ultimately $\mathcal{P}'_n$-positive.

  Now $(\{^{t}\!/\!_{x}\}\Gamma, \mathcal{P}'_n)$ is assumed to be safe, so we want to apply this property to $\Gamma' := \Gamma$, to the semantically consistent set $\mathcal{R}$, and to the $\mathcal{P}'_n$-positive literal $\{^{t}\!/\!_{x}\}l_i^{\perp}$, so as to conclude

  $$\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}l_i \models_{\mathcal{T}}$$

  To apply the safety property, we note that that $\mathcal{R}, \{^{t}\!/\!_{x}\}l_i \models_{\mathcal{T}}$ and that

  $$\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma) \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma) \cup \mathsf{U}^{\perp}_{\mathcal{P}'_n}$$

  provided we have $l_i \in \mathsf{U}_{\mathcal{P}'_n}$.

  In order to prove that proviso, first notice that $l_i \in \mathsf{U}_{\mathcal{P}}$, since $l_i \in \mathsf{U}_{\mathcal{P}_{i-1}}$. Now we must have $x \in \mathsf{FV}(l_i)$, otherwise $l_i = \{^{t}\!/\!_{x}\}l_i$ and we know that $\{^{t}\!/\!_{x}\}l_i$ is $\mathcal{P}$-negative. Since none of the literals $(\{^{t}\!/\!_{x}\}l_k)_{1 \leq k \leq n}$ have x as a free variable, we conclude the proviso $l_i \in \mathsf{U}_{\mathcal{P}'_n}$.

  Therefore safety ensures $\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}l_i \models_{\mathcal{T}}$ and we can finally build

  $$\mathsf{Select}\,\dfrac{\mathsf{Init}_1\,\dfrac{\mathsf{lit}_{\mathcal{P}'_n}(\{^{t}\!/\!_{x}\}\Gamma), \{^{t}\!/\!_{x}\}l_i \models_{\mathcal{T}}}{\{^{t}\!/\!_{x}\}\Gamma \vdash^{\mathcal{P}'_n} [\{^{t}\!/\!_{x}\}l_i^{\perp}]}}{\{^{t}\!/\!_{x}\}\Gamma \vdash^{\mathcal{P}'_n}}$$

  as $\{^{t}\!/\!_{x}\}l_i^{\perp}$ is $\mathcal{P}'_n$-positive.

□

We can finally state and prove the admissibility of instantiation:

LEMMA 18 (Admissibility of instantiation) Let x be a variable.
Let t be a term with $x \notin \mathsf{FV}(t)$ and $\mathcal{P}$ be a polarisation set with $x \notin \mathsf{FV}(\mathcal{P})$.
Let $l_1, \dots, l_n$ be n literals.
Let $\mathcal{P}_i := \mathcal{P}; l_1; \dots; l_i$ with $\mathcal{P}_0 := \mathcal{P}$, and similarly let $\mathcal{P}'_i := \mathcal{P}; \{{}^t\!/_x\} l_1; \dots; \{{}^t\!/_x\} l_i$ with $\mathcal{P}'_0 := \mathcal{P}$.
The following rules are admissible in $\mathsf{LK}^p(\mathcal{T})$:[3]

$$(\mathsf{Inst})\frac{\Gamma \vdash^{\mathcal{P}_n} \Delta}{\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n} \{{}^t\!/_x\}\Delta} \qquad (\mathsf{Inst}_f)\frac{\Gamma \vdash^{\mathcal{P}_n} [B]}{\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n} [\{{}^t\!/_x\}B] \text{ or } \{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n}}$$

where we assume

- for all i such that $1 \leq i \leq n$, we have $l_i \in \Gamma$;
- $\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n} \{{}^t\!/_x\}\Delta$ is safe in (Inst);
- $(\{{}^t\!/_x\}\Gamma, \mathcal{P}'_n)$ is safe in (Inst_f).

※

Proof: By induction on the derivation of the premiss.

- $(\wedge^-)$,$(\vee^-)$,$(\forall)$,$(\bot^-)$,$(\top^-)$,$(\wedge^+)$,$(\vee^+)$,$(\exists)$,$(\top^+)$
  These rules are straightforward as the polarisation set is not involved.
- (Store) We assume

$$\frac{\Gamma, A^\bot \vdash^{\mathcal{P}_n; A^\bot} \Delta}{\Gamma \vdash^{\mathcal{P}_n} A, \Delta} A \text{ is a literal or is } \mathcal{P}_n\text{-positive}$$

  Using the induction hypothesis on the premiss we can build

$$\frac{\{{}^t\!/_x\}\Gamma, \{{}^t\!/_x\}A^\bot \vdash^{\mathcal{P}'_n; \{{}^t\!/_x\}A^\bot} \{{}^t\!/_x\}\Delta}{\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n} \{{}^t\!/_x\}A, \{{}^t\!/_x\}\Delta}$$

  since $\{{}^t\!/_x\}A$ is a literal or is $\mathcal{P}'_n$-positive.
- (Select) We assume

$$\frac{\Gamma, P^\bot \vdash^{\mathcal{P}_n} [P]}{\Gamma, P^\bot \vdash^{\mathcal{P}_n}} P \text{ is not } \mathcal{P}_n\text{-negative}$$

  If $\{{}^t\!/_x\}P$ is not $\mathcal{P}'_n$-negative, then we can apply the induction hypothesis and build

$$\frac{\{{}^t\!/_x\}\Gamma, \{{}^t\!/_x\}P^\bot \vdash^{\mathcal{P}'_n} [\{{}^t\!/_x\}P]}{\{{}^t\!/_x\}\Gamma, \{{}^t\!/_x\}P^\bot \vdash^{\mathcal{P}'_n}}$$

  Otherwise, $\{{}^t\!/_x\}P$ is a $\mathcal{P}'_n$-negative literal and we can do the same as above with the (Select^-) rule instead of (Select).
- (Init_2) We assume

$$\frac{\mathsf{lit}_{\mathcal{P}_n}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}_n}}$$

  We use Lemma 17 with $\mathcal{A} := \emptyset$, since we know $\mathsf{lit}_{\mathcal{P}_n}(\Gamma) \models_{\mathcal{T}}$.
  If we get $\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/_x\}\Gamma) \models_{\mathcal{T}}$, we build a proof with the same rule (Init_2):

$$\frac{\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/_x\}\Gamma) \models_{\mathcal{T}}}{\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n}}$$

  If not, we directly get a proof of $\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n}$.
- (Init_1) We assume

[3] The admissibility of (Inst_f) means that if $\Gamma \vdash^{\mathcal{P}_n} [B]$ is derivable in $\mathsf{LK}^p(\mathcal{T})$ then either $\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n} [\{{}^t\!/_x\}B]$ or $\{{}^t\!/_x\}\Gamma \vdash^{\mathcal{P}'_n}$ is derivable in $\mathsf{LK}^p(\mathcal{T})$.

$$\frac{\mathsf{lit}_{\mathcal{P}_n}(\Gamma), p^\perp \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}_n} [p]}$$

where p is $\mathcal{P}_n$-positive.

We use Lemma 17 with $\mathcal{A} := \{p\}$, since we know $\mathsf{lit}_{\mathcal{P}_n}(\Gamma), p^\perp \models_{\mathcal{T}}$.

If we get $\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}p^\perp \models_{\mathcal{T}}$, we build a proof with the same rule (Init_1):

$$\frac{\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}p^\perp \models_{\mathcal{T}}}{\{{}^t\!/\!{}_x\}\Gamma \vdash^{\mathcal{P}'_n} [\{{}^t\!/\!{}_x\}p]}$$

If not, we directly get a proof of $\{{}^t\!/\!{}_x\}\Gamma \vdash^{\mathcal{P}'_n}$.

- ($\mathsf{Release}$) We assume

$$\frac{\Gamma \vdash^{\mathcal{P}_n} N}{\Gamma \vdash^{\mathcal{P}_n} [N]}\ N \text{ is not } \mathcal{P}_n\text{-positive}$$

If $\{{}^t\!/\!{}_x\}N$ is not $\mathcal{P}'_n$-positive, then we can apply the induction hypothesis and build

$$\frac{\{{}^t\!/\!{}_x\}\Gamma \vdash^{\mathcal{P}'_n} \{{}^t\!/\!{}_x\}N}{\{{}^t\!/\!{}_x\}\Gamma \vdash^{\mathcal{P}'_n} [\{{}^t\!/\!{}_x\}N]}$$

Otherwise, N is a literal l that is not $\mathcal{P}_n$-positive, but such that $\{{}^t\!/\!{}_x\}l$ is $\mathcal{P}'_n$-positive.

– If $\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}l \models_{\mathcal{T}}$, then we build

$$\mathsf{cut}_1\frac{\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}l \models_{\mathcal{T}} \qquad \{{}^t\!/\!{}_x\}\Gamma, \{{}^t\!/\!{}_x\}l^\perp \vdash^{\mathcal{P}'_n}}{\{{}^t\!/\!{}_x\}\Gamma \vdash^{\mathcal{P}'_n}}$$

where the right premiss is proved as follows:

Notice that the assumed derivation of $\Gamma \vdash^{\mathcal{P}_n} l$ necessarily contains a sub-derivation concluding $\Gamma, l^\perp \vdash^{\mathcal{P}_n; l^\perp}$, and applying the induction hypothesis on this yields a derivation of $\{{}^t\!/\!{}_x\}\Gamma, \{{}^t\!/\!{}_x\}l^\perp \vdash^{\mathcal{P}'_n}$.

– Assume now that $\mathcal{R} := \mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}l$ is semantically consistent. We build

$$\mathsf{Init}_1\frac{\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}l^\perp \models_{\mathcal{T}}}{\{{}^t\!/\!{}_x\}\Gamma \vdash^{\mathcal{P}'_n} [\{{}^t\!/\!{}_x\}l]}$$

and we have to prove the side-condition $\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma), \{{}^t\!/\!{}_x\}l^\perp \models_{\mathcal{T}}$.

This is trivial if $\{{}^t\!/\!{}_x\}l \in \{{}^t\!/\!{}_x\}\Gamma$ (as $\{{}^t\!/\!{}_x\}l$ is $\mathcal{P}'_n$-positive).

If on the contrary $\{{}^t\!/\!{}_x\}l \notin \{{}^t\!/\!{}_x\}\Gamma$, then we get it from the assumed safety of $(\{{}^t\!/\!{}_x\}\Gamma, \mathcal{P}'_n)$, applied to $\Gamma' := \Gamma$, to the semantically consistent set $\mathcal{R}$, and to the $\mathcal{P}'_n$-positive literal $\{{}^t\!/\!{}_x\}l$. To apply the safety property, we note that $\mathcal{R}, \{{}^t\!/\!{}_x\}l^\perp \models_{\mathcal{T}}$ and that

$$\mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma) \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P}'_n}(\{{}^t\!/\!{}_x\}\Gamma) \cup \mathsf{U}^{\downarrow}_{\mathcal{P}'_n}$$

provided we have $\{{}^t\!/\!{}_x\}l \in \mathsf{U}^{\downarrow}_{\mathcal{P}'_n}$.

We prove that $l \in \mathsf{U}_{\mathcal{P}'_n}$ as follows:
First notice that $l \in \mathsf{U}_{\mathcal{P}}$, otherwise l would be $\mathcal{P}$-negative and so would be $\{{}^t\!/\!{}_x\}l$ (since $x \notin \mathsf{FV}(\mathcal{P})$). Then notice that $\{{}^t\!/\!{}_x\}l$ must be $\mathcal{P}$-positive, since it is $\mathcal{P}'_n$-positive but $\{{}^t\!/\!{}_x\}l \notin \{{}^t\!/\!{}_x\}\Gamma$. Therefore $l \neq \{{}^t\!/\!{}_x\}l$, so $x \in \mathsf{FV}(l)$, and finally we get $l \in \mathsf{U}_{\mathcal{P}'_n}$, since none of the literals $(\{{}^t\!/\!{}_x\}l_k)_{1 \leq k \leq n}$ have x as a free variable.

□

3.5.3 More general cuts

THEOREM 19 (cut_3, cut_4 and cut_5) The following rules are admissible in $\mathsf{LK}^p(\mathcal{T})$:[4]

$$(\mathsf{cut}_3)\frac{\Gamma \vdash^{\mathcal{P}} [A] \quad \Gamma \vdash^{\mathcal{P}} A^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}$$

$$(\mathsf{cut}_4)\frac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta} \qquad (\mathsf{cut}_5)\frac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [B]}{\Gamma \vdash^{\mathcal{P}} [B] \text{ or } \Gamma \vdash^{\mathcal{P}}}$$

where

- N is assumed to not be $\mathcal{P}$-positive in cut_4 and cut_5;
- the sequent $\Gamma \vdash^{\mathcal{P}} \Delta$ in cut_3 and cut_4, and the pair $(\Gamma, \mathcal{P})$ in cut_5, are all assumed to be safe.

※

Proof: By simultaneous induction on the following lexicographical measure:

- the size of the cut-formula (A or N)
- the fact that the cut-formula (A or N) is positive or negative
  (if of equal size, a positive formula is considered smaller than a negative formula)
- the height of the derivation of the right premiss

Weakenings and contractions (as they are admissible in the system) are implicitly used throughout this proof.

In order to eliminate cut_3, we analyse which rule is used to prove the left premiss. We then use invertibility of the negative phase so that the last rule used in the right premiss is its dual one.

- $(\wedge^+)$

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} [A] \quad \Gamma \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} A^{\perp}, B^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} A \vee^- B, \Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3$$

reduces to

$$\dfrac{\Gamma \vdash^{\mathcal{P}} [B] \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} [A] \quad \Gamma \vdash^{\mathcal{P}} A^{\perp}, B^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} B^{\perp}, \Delta}\,\mathsf{cut}_3}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3$$

- $(\vee^+)$

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} A_1^{\perp}, \Delta \quad \Gamma \vdash^{\mathcal{P}} A_2^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} A_1 \wedge^- A_2, \Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3$$

reduces to

$$\dfrac{\Gamma \vdash^{\mathcal{P}} [A_i] \qquad \Gamma \vdash^{\mathcal{P}} A_i^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3$$

- $(\exists\ )$

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\} A]}{\Gamma \vdash^{\mathcal{P}} [\exists x A]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} A^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} (\forall x A^{\perp}), \Delta}\, x \notin \mathsf{FV}(\Gamma, \Delta, \mathcal{P})}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3$$

reduces to

[4]The admissibility of cut_5 means that if $\Gamma \vdash^{\mathcal{P}} N$ and $\Gamma, N \vdash^{\mathcal{P};N} [B]$ are derivable in $\mathsf{LK}^p(\mathcal{T})$ then either $\Gamma \vdash^{\mathcal{P}} [B]$ or $\Gamma \vdash^{\mathcal{P}}$ is derivable in $\mathsf{LK}^p(\mathcal{T})$.

$$\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}A] \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} A^{\perp},\Delta}{\Gamma \vdash^{\mathcal{P}} (\{{}^{t}\!/\!_{x}\}A^{\perp}),\Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3$$

using Lemma 18 (admissibility of instantiation) with $n = 0$, noticing that $x \notin \mathsf{FV}(\mathcal{P})$ and that $\Gamma \vdash^{\mathcal{P}} (\{{}^{t}\!/\!_{x}\}A^{\perp}),\Delta$ is safe (since $\Gamma \vdash^{\mathcal{P}} \Delta$ is safe).[5]

- $(\top^{+})$

$$\dfrac{\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \bot^{-},\Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3 \qquad \text{reduces to} \qquad \Gamma \vdash^{\mathcal{P}} \Delta$$

- (Init_1)

$$\dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma),p^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]} \qquad \dfrac{\Gamma,p \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} (p^{\perp}),\Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3 \qquad \text{reduces to} \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma),p^{\perp} \models_{\mathcal{T}} \qquad \Gamma,p \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_1$$

with $p \in \mathcal{P}$.

- $(\mathsf{Release})$

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]} \qquad \dfrac{\Gamma,N \vdash^{\mathcal{P};N} \Delta}{\Gamma \vdash^{\mathcal{P}} (N^{\perp}),\Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_3 \qquad \text{reduces to} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} N \qquad \Gamma,N \vdash^{\mathcal{P};N} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_4$$

where N is not $\mathcal{P}$-positive. We will describe below how cut_4 is reduced.
In order to reduce cut_4, we analyse which rule is used to prove the right premiss.

- $(\wedge^{-})$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,N \vdash^{\mathcal{P};N} B,\Delta \quad \Gamma,N \vdash^{\mathcal{P};N} C,\Delta}{\Gamma,N \vdash^{\mathcal{P};N} B\wedge^{-}C,\Delta}}{\Gamma \vdash^{\mathcal{P}} B\wedge^{-}C,\Delta}\,\mathsf{cut}_4$$

reduces to

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma,N \vdash^{\mathcal{P};N} B,\Delta}{\Gamma \vdash^{\mathcal{P}} B,\Delta}\,\mathsf{cut}_4 \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma,N \vdash^{\mathcal{P};N} C,\Delta}{\Gamma \vdash^{\mathcal{P}} C,\Delta}\,\mathsf{cut}_4}{\Gamma \vdash^{\mathcal{P}} B\wedge^{-}C,\Delta}$$

- $(\vee^{-})$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,N \vdash^{\mathcal{P};N} B,C,\Delta}{\Gamma,N \vdash^{\mathcal{P};N} B\vee^{-}C,\Delta}}{\Gamma \vdash^{\mathcal{P}} B\vee^{-}C,\Delta}\,\mathsf{cut}_4 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma,N \vdash^{\mathcal{P};N} B,C,\Delta}{\Gamma \vdash^{\mathcal{P}} B,C,\Delta}\,\mathsf{cut}_4}{\Gamma \vdash^{\mathcal{P}} B\vee^{-}C,\Delta}$$

- $(\forall\ )$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,N \vdash^{\mathcal{P};N} B,\Delta}{\Gamma,N \vdash^{\mathcal{P};N} \forall xB,\Delta}}{\Gamma \vdash^{\mathcal{P}} \forall xB,\Delta}\,\mathsf{cut}_4 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma,N \vdash^{\mathcal{P};N} B,\Delta}{\Gamma \vdash^{\mathcal{P}} B,\Delta}\,\mathsf{cut}_4}{\Gamma \vdash^{\mathcal{P}} \forall xB,\Delta}$$

- $(\bot^{-})$

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,N \vdash^{\mathcal{P};N} \Delta}{\Gamma,N \vdash^{\mathcal{P};N} \bot^{-},\Delta}}{\Gamma \vdash^{\mathcal{P}} \bot^{-},\Delta}\,\mathsf{cut}_4 \qquad \text{reduces to} \qquad \dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma,N \vdash^{\mathcal{P};N} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_4}{\Gamma \vdash^{\mathcal{P}} \bot^{-},\Delta}$$

[5] Using α-conversion, we can also pick x such that $x \notin \mathsf{FV}(t)$.

- (Store)

$$\dfrac{\Gamma\vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,N,B^{\perp}\vdash^{\mathcal{P};N;B^{\perp}}\Delta}{\Gamma,N\vdash^{\mathcal{P};N} B,\Delta}}{\Gamma\vdash^{\mathcal{P}} B,\Delta}\,\mathsf{cut}_4 \qquad\text{reduces to}\qquad \dfrac{\dfrac{\Gamma,B^{\perp}\vdash^{\mathcal{P};B^{\perp}} N \quad \Gamma,N,B^{\perp}\vdash^{\mathcal{P};B^{\perp};N}\Delta}{\Gamma,B^{\perp}\vdash^{\mathcal{P};B^{\perp}}\Delta}\,\mathsf{cut}_4}{\Gamma\vdash^{\mathcal{P}} B,\Delta}$$

whose left branch is closed by using
 – possibly the admissibility of (Pol) (if $B\in\mathsf{U}_{\mathcal{P}}$), so as to get $\Gamma,B^{\perp}\vdash^{\mathcal{P}} N$,
 – then the admissibility of (W_l) (on $B^{\perp}$), to get to the provable premiss $\Gamma\vdash^{\mathcal{P}} N$;

whose right branch is the same as the provable $\Gamma,N,B^{\perp}\vdash^{\mathcal{P};N;B^{\perp}}\Delta$ unless $B=N\in\mathsf{U}_{\mathcal{P}}$, in which case the commutation $\mathcal{P};B^{\perp};N=\mathcal{P};N;B^{\perp}$ does not hold. In this last case, we build

$$(\mathsf{W}_r)\dfrac{\Gamma\vdash^{\mathcal{P}} B}{\Gamma\vdash^{\mathcal{P}} B,\Delta}$$

- (Init_2) when $N\notin\mathsf{U}_{\mathcal{P}}$, in which case $\mathcal{P};N=\mathcal{P}$ and $\mathsf{lit}_{\mathcal{P}}(\Gamma,N)=\mathsf{lit}_{\mathcal{P}}(\Gamma)$ (since $N\notin\mathcal{P}$ either):

$$\dfrac{\Gamma\vdash^{\mathcal{P}} N \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma)\models_{\mathcal{T}}}{\Gamma,N\vdash^{\mathcal{P};N}}}{\Gamma\vdash^{\mathcal{P}}}\,\mathsf{cut}_4 \qquad\text{reduces to}\qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma)\models_{\mathcal{T}}}{\Gamma\vdash^{\mathcal{P}}}$$

- (Init_2) when $N\in\mathsf{U}_{\mathcal{P}}$, in which case $\mathsf{lit}_{\mathcal{P};N}(\Gamma,N)=\mathsf{lit}_{\mathcal{P}}(\Gamma),N$:

$$\dfrac{\dfrac{\Gamma,N^{\perp}\vdash^{\mathcal{P},N^{\perp}}}{\Gamma\vdash^{\mathcal{P}} N} \qquad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma),N\models_{\mathcal{T}}}{\Gamma,N\vdash^{\mathcal{P};N}}}{\Gamma\vdash^{\mathcal{P}}}\,\mathsf{cut}_4 \qquad\text{reduces to}\qquad \dfrac{\mathsf{lit}_{\mathcal{P},N^{\perp}}(\Gamma),N\models_{\mathcal{T}} \qquad \Gamma,N^{\perp}\vdash^{\mathcal{P},N^{\perp}}}{\Gamma\vdash^{\mathcal{P}}}\,\mathsf{cut}_1$$

since $\mathsf{lit}_{\mathcal{P}}(\Gamma),N\models_{\mathcal{T}}$ implies $\mathsf{lit}_{\mathcal{P},N^{\perp}}(\Gamma),N\models_{\mathcal{T}}$.

- (Select) on formula $N^{\perp}$

$$\dfrac{\Gamma\vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,N\vdash^{\mathcal{P};N}[N^{\perp}]}{\Gamma,N\vdash^{\mathcal{P};N}}}{\Gamma\vdash^{\mathcal{P}}}\,\mathsf{cut}_4 \qquad\text{reduces to}\qquad \dfrac{\dfrac{\Gamma\vdash^{\mathcal{P}} N \quad \Gamma,N\vdash^{\mathcal{P};N}[N^{\perp}]}{\Gamma\vdash^{\mathcal{P}}[N^{\perp}]}\,\mathsf{cut}_5 \qquad \Gamma\vdash^{\mathcal{P}} N}{\Gamma\vdash^{\mathcal{P}}}\,\mathsf{cut}_3$$

$$\text{or to}\qquad \dfrac{\Gamma\vdash^{\mathcal{P}} N \quad \Gamma,N\vdash^{\mathcal{P};N}[N^{\perp}]}{\Gamma\vdash^{\mathcal{P}}}\,\mathsf{cut}_5$$

depending on the outcome of cut_5

- (Select) on a formula P that is not $\mathcal{P};N$-negative

$$\dfrac{\Gamma,P^{\perp}\vdash^{\mathcal{P}} N \qquad \dfrac{\Gamma,P^{\perp},N\vdash^{\mathcal{P};N}[P]}{\Gamma,P^{\perp},N\vdash^{\mathcal{P};N}}}{\Gamma,P^{\perp}\vdash^{\mathcal{P}}}\,\mathsf{cut}_4 \qquad\text{reduces to}\qquad \dfrac{\dfrac{\Gamma,P^{\perp}\vdash^{\mathcal{P}} N \quad \Gamma,P^{\perp},N\vdash^{\mathcal{P};N}[P]}{\Gamma,P^{\perp}\vdash^{\mathcal{P}}[P]}\,\mathsf{cut}_5}{\Gamma,P^{\perp}\vdash^{\mathcal{P}}}$$

$$\text{or to}\qquad \dfrac{\Gamma,P^{\perp}\vdash^{\mathcal{P}} N \quad \Gamma,P^{\perp},N\vdash^{\mathcal{P};N}[N^{\perp}]}{\Gamma,P^{\perp}\vdash^{\mathcal{P}}}\,\mathsf{cut}_5$$

depending on the outcome of cut_5

We have reduced all cases of cut_4; we now reduce the cases for cut_5 (again, by case analysis on the last rule used to prove the right premiss).

- $(\wedge^{+})$ We are given

$$\Gamma\vdash^{\mathcal{P}} N \qquad\text{and}\qquad \dfrac{\Gamma,N\vdash^{\mathcal{P};N}[B_1] \quad \Gamma,N\vdash^{\mathcal{P};N}[B_2]}{\Gamma,N\vdash^{\mathcal{P};N}[B_1\wedge^{+}B_2]}$$

and by cut_5 we want to derive either $\Gamma \vdash^{\mathcal{P}} [B_1 \wedge^+ B_2]$ or $\Gamma \vdash^{\mathcal{P}}$.
If we can, we build

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [B_1]}{\Gamma \vdash^{\mathcal{P}} [B_1]}\mathsf{cut}_5 \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [B_2]}{\Gamma \vdash^{\mathcal{P}} [B_2]}\mathsf{cut}_5}{\Gamma \vdash^{\mathcal{P}} [B_1 \wedge^+ B_2]}$$

Otherwise we build

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [B_i]}{\Gamma \vdash^{\mathcal{P}}}\mathsf{cut}_5$$

where i is (one of) the premiss(es) for which cut_5 produces a proof of $\Gamma \vdash^{\mathcal{P}}$.

- $(\vee^+)$ We are given

$$\Gamma \vdash^{\mathcal{P}} N \quad \text{and} \quad \dfrac{\Gamma, N \vdash^{\mathcal{P};N} [B_i]}{\Gamma, N \vdash^{\mathcal{P};N} [B_1 \vee^+ B_2]}$$

and by cut_5 we want to derive either $\Gamma \vdash^{\mathcal{P}} [B_1 \vee^+ B_2]$ or $\Gamma \vdash^{\mathcal{P}}$.
If we can, we build

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [B_i]}{\Gamma \vdash^{\mathcal{P}} [B_i]}\mathsf{cut}_5}{\Gamma \vdash^{\mathcal{P}} [B_1 \vee^+ B_2]}$$

Otherwise we build

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [B_i]}{\Gamma \vdash^{\mathcal{P}}}$$

- $(\exists\ )$ We are given

$$\Gamma \vdash^{\mathcal{P}} N \quad \text{and} \quad \dfrac{\Gamma, N \vdash^{\mathcal{P};N} [\{{}^{t}\!/\!_{x}\}B]}{\Gamma, N \vdash^{\mathcal{P};N} [\exists x B]}$$

and by cut_5 we want to derive either $\Gamma \vdash^{\mathcal{P}} [\exists x B]$ or $\Gamma \vdash^{\mathcal{P}}$.
If we can, we build

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [\{{}^{t}\!/\!_{x}\}B]}{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}B]}\mathsf{cut}_5}{\Gamma \vdash^{\mathcal{P}} [\exists x B]}$$

Otherwise we build

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} [\{{}^{t}\!/\!_{x}\}B]}{\Gamma \vdash^{\mathcal{P}}}$$

- $(\top^+)$ We are given

$$\Gamma \vdash^{\mathcal{P}} N \quad \text{and} \quad \dfrac{}{\Gamma, N \vdash^{\mathcal{P};N} [\top^+]}$$

and by cut_5 we want to derive either $\Gamma \vdash^{\mathcal{P}} [\top^+]$ or $\Gamma \vdash^{\mathcal{P}}$.
We build

$$\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]}$$

- $(\mathsf{Release})$ We are given:

$$\Gamma \vdash^{\mathcal{P}} N \qquad \text{and} \qquad \dfrac{\Gamma, N \vdash^{\mathcal{P};N} N'}{\Gamma, N \vdash^{\mathcal{P};N} [N']}$$

where N' is not $\mathcal{P}; N$-positive;
and by cut_5 we want to derive either $\Gamma \vdash^{\mathcal{P}} [N']$ or $\Gamma \vdash^{\mathcal{P}}$.

We build

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} N \quad \Gamma, N \vdash^{\mathcal{P};N} N'}{\Gamma \vdash^{\mathcal{P}} N'}\,\mathsf{cut}_4}{\Gamma \vdash^{\mathcal{P}} [N']}$$

since N' is not $\mathcal{P}$-positive.

- (Init_1) We are given:

$$\Gamma \vdash^{\mathcal{P}} N \qquad \text{and} \qquad \dfrac{\mathsf{lit}_{\mathcal{P};N}(\Gamma, N), p^{\perp} \models_{\mathcal{T}}}{\Gamma, N \vdash^{\mathcal{P};N} [p]}$$

with $p \in \mathcal{P}; N$,
and by cut_5 we want to derive either $\Gamma \vdash^{\mathcal{P}} [p]$ or $\Gamma \vdash^{\mathcal{P}}$.
If N is $\mathcal{P}$-negative
then $\mathcal{P}; N = \mathcal{P}$ and p is $\mathcal{P}$-positive. So $\mathsf{lit}_{\mathcal{P};N}(\Gamma, N), p^{\perp} = \mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}$ and we build

$$(\mathsf{Init}_1)\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}}{\Gamma \vdash^{\mathcal{P}} [p]}$$

If $N \in \mathsf{U}_{\mathcal{P}}$ $(\mathsf{lit}_{\mathcal{P};N}(\Gamma, N), p^{\perp} = \mathsf{lit}_{\mathcal{P}}(\Gamma), N, p^{\perp})$
– if $p = N$ then we build

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]}$$

as N is not $\mathcal{P}$-positive;
– if $p \neq N$ then p is $\mathcal{P}$-positive

1. if $\mathsf{lit}_{\mathcal{P}}(\Gamma), N \models_{\mathcal{T}}$
then applying invertibility of $(\mathsf{Store}^{=})$ on $\Gamma \vdash^{\mathcal{P}} N$ gives $\Gamma, N^{\perp} \vdash^{\mathcal{P}}$ and we build:

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), N \models_{\mathcal{T}} \quad \Gamma, N^{\perp} \vdash^{\mathcal{P}}}{\Gamma \vdash^{\mathcal{P}}}\,\mathsf{cut}_1$$

2. if $\mathsf{lit}_{\mathcal{P}}(\Gamma), N \not\models_{\mathcal{T}}$
then $\mathcal{R} :=\ \mathsf{lit}_{\mathcal{P}}(\Gamma), N$ is a set of literals satisfying $\mathsf{lit}_{\mathcal{P}}(\Gamma) \subseteq \mathcal{R} \subseteq \mathsf{lit}_{\mathcal{P}}(\Gamma) \cup \mathsf{U}_{\mathcal{P}}$ (since $N \in \mathsf{U}_{\mathcal{P}}$) and $\mathcal{R}, p^{\perp} \models_{\mathcal{T}}$.
Hence we get $\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp} \models_{\mathcal{T}}$ as well, since $(\Gamma, \mathcal{P})$ is assumed to be safe.
We can finally build

$$(\mathsf{Init}_1)\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]}$$

□

THEOREM 20 (cut_6, cut_7, and cut_8) The following rules are admissible in $\mathsf{LK}^p(\mathcal{T})$:[6]

$$\dfrac{\Gamma \vdash^{\mathcal{P}} N, \Delta \quad \Gamma, N \vdash^{\mathcal{P};N} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_6 \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} A, \Delta \quad \Gamma \vdash^{\mathcal{P}} A^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_7 \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P};l} \Delta \quad \Gamma, l^{\perp} \vdash^{\mathcal{P};l^{\perp}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_8$$

※

Proof: cut_6 is proved admissible by induction on the multiset Δ: the base case is the admissibility of cut_4, and the other cases just require the inversion of the connectives in Δ (using $(\mathsf{Store}^{=})$ instead of (Store), to avoid modifying the polarisation set).

For cut_7, we can assume without loss of generality (swapping A and $A^{\perp}$) that A is not $\mathcal{P}$-positive. Applying inversion on $\Gamma \vdash^{\mathcal{P}} A^{\perp}, \Delta$ gives a proof of $\Gamma, A \vdash^{\mathcal{P};A} \Delta$, and cut_7 is then obtained by cut_6:

[6](again, assuming $\Gamma \vdash^{\mathcal{P}} \Delta$ is safe)

$$\dfrac{\Gamma \vdash^{\mathcal{P}} A, \Delta \quad \Gamma, A \vdash^{\mathcal{P};A} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_6$$

cut_8 is obtained as follows:

$$\dfrac{\dfrac{\Gamma, l^{\perp} \vdash^{\mathcal{P};l^{\perp}} \Delta}{\Gamma \vdash^{\mathcal{P}} l, \Delta} \qquad \dfrac{\Gamma, l \vdash^{\mathcal{P};l} \Delta}{\Gamma \vdash^{\mathcal{P}} l^{\perp}, \Delta}}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_7$$

□

3.6 Changing the polarity of connectives

In this section, we show that changing the polarity of connectives does not change provability in $\mathsf{LK}^p(\mathcal{T})$. To prove this property of the $\mathsf{LK}^p(\mathcal{T})$ system, we generalise it into a new system $\mathsf{LK}^+(\mathcal{T})$.

Definition 18 ($\mathsf{LK}^+(\mathcal{T})$) The sequent calculus $\mathsf{LK}^+(\mathcal{T})$ manipulates one kind of sequent:

$$\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\Delta \qquad \text{where } \mathcal{X} ::= \bullet \mid A$$

Here, $\mathcal{P}$ is a polarisation set, Γ is a multiset of literals and $\mathcal{P}$-negative formulae, Δ is a multiset of formulae, and $\mathcal{X}$ is said to be in the *focus* of the sequent.
The rules of $\mathsf{LK}^+(\mathcal{T})$, given in Figure 3.2, are again of three kinds: synchronous rules, asynchronous rules, and structural rules. ※

Synchronous rules

$$(\wedge^+)\dfrac{\Gamma \vdash^{\mathcal{P}} [A]\Delta \qquad \Gamma \vdash^{\mathcal{P}} [B]\Delta}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]\Delta} \qquad (\vee^+)\dfrac{\Gamma \vdash^{\mathcal{P}} [A_i]\Delta}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]\Delta} \qquad (\exists)\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}A]\Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x A]\Delta}$$

$$(\top^+)\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]\Delta} \qquad (\mathsf{Init}_1)\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [l]\Delta}\; l \text{ is } \mathcal{P}\text{-positive} \qquad (\mathsf{Release})\dfrac{\Gamma \vdash^{\mathcal{P}} [\bullet]N}{\Gamma \vdash^{\mathcal{P}} [N]}\; N \text{ not } \mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta \qquad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A \wedge^- B, \Delta} \qquad (\vee^-)\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A_1, A_2, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A_1 \vee^- A_2, \Delta} \qquad (\forall)\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), \Delta}\; x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta, \mathcal{P})$$

$$(\perp^-)\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, \Delta} \qquad (\top^-)\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\top^-} \qquad (\mathsf{Store})\dfrac{\Gamma, A^{\perp} \vdash^{\mathcal{P};A^{\perp}} [\mathcal{X}]\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta}\; A \text{ literal or } \mathcal{P}\text{-positive}$$

Structural rules

$$(\mathsf{Select})\dfrac{\Gamma, P^{\perp} \vdash^{\mathcal{P}} [P]\Delta}{\Gamma, P^{\perp} \vdash^{\mathcal{P}} [\bullet]\Delta}\; P \text{ not } \mathcal{P}\text{-negative} \qquad (\mathsf{Init}_2)\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta}$$

Figure 3.2: System $\mathsf{LK}^+(\mathcal{T})$

Remark 21 The $\mathsf{LK}^+(\mathcal{T})$ system is an extension system of $\mathsf{LK}^p(\mathcal{T})$: the $\mathsf{LK}^p(\mathcal{T})$ system is the fragment of $\mathsf{LK}^+(\mathcal{T})$ where every sequent $\Gamma, P^{\perp} \vdash^{\mathcal{P}} [\bullet]\Delta$ is requested to have either $\mathcal{X} = \bullet$ or Δ is empty. In terms of bottom-up proof-search, this only restricts the structural rules to the case where Δ is empty.

As in $\mathsf{LK}^p(\mathcal{T})$, (left-)weakening and (left-)contraction are height-preserving admissible in $\mathsf{LK}^+(\mathcal{T})$.

We can now prove a new version of identity:

Lemma 22 (Identities) For all $\mathcal{P}$, A, Δ, the sequent $\vdash^{\mathcal{P}} [A^\perp]A, \Delta$ is provable in $\mathsf{LK}^+(\mathcal{T})$. ※

Proof: By induction on A using an extended but well-founded order on formulae:
a formula is smaller than another one when

- either it contains fewer connectives
- or the number of connectives is equal, neither formulae are literals, and the former formula is negative and the latter is positive.

We now treat all possible shapes for the formula A:

- $A = A_1 \wedge^- A_2$

$$\dfrac{\dfrac{\vdash^{\mathcal{P}} [{A_1}^\perp]A_1, \Delta}{\vdash^{\mathcal{P}} [{A_1}^\perp \vee^+ {A_2}^\perp]A_1, \Delta} \qquad \dfrac{\vdash^{\mathcal{P}} [{A_2}^\perp]A_2, \Delta}{\vdash^{\mathcal{P}} [{A_1}^\perp \vee^+ {A_2}^\perp]A_2, \Delta}}{\vdash^{\mathcal{P}} [{A_1}^\perp \vee^+ {A_2}^\perp]A_1 \wedge^- A_2, \Delta}$$

We can complete the proof on the left-hand side by applying the induction hypothesis on A_1 and on the right-hand side by applying the induction hypothesis on A_2.

- $A = A_1 \vee^- A_2$

$$\dfrac{\dfrac{\vdash^{\mathcal{P}} [{A_1}^\perp]A_1, A_2, \Delta \quad \vdash^{\mathcal{P}} [{A_2}^\perp]A_1, A_2, \Delta}{\vdash^{\mathcal{P}} [{A_1}^\perp \wedge^+ {A_2}^\perp]A_1, A_2, \Delta}}{\vdash^{\mathcal{P}} [{A_1}^\perp \wedge^+ {A_2}^\perp]A_1 \vee^- A_2, \Delta}$$

We can complete the proof on the left-hand side by applying the induction hypothesis on A_1 and on the right-hand side by applying the induction hypothesis on A_2.

- $A = \forall x A$

$$\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}} [A^\perp]A, \Delta}{\vdash^{\mathcal{P}} [\{t/x\}A^\perp]A, \Delta}\text{ choosing t=x}}{\vdash^{\mathcal{P}} [\exists x A^\perp]A, \Delta}}{\vdash^{\mathcal{P}} [\exists x A^\perp]\forall x A, \Delta}\ x \notin \mathsf{FV}(\exists x A^\perp, \Delta)$$

We can complete the proof by applying the induction hypothesis on A.

- $A = \bot^-$

$$\dfrac{}{\vdash^{\mathcal{P}} [\top^+]\bot -, \Delta}\ \top^+$$

- $A = p^\perp$, with p not being $\mathcal{P}$-negative:

$$\dfrac{\dfrac{}{p \vdash^{\mathcal{P};p} [p]\Delta}}{\vdash^{\mathcal{P}} [p]p^\perp, \Delta}$$

as p is then $\mathcal{P};p$-positive.

- $A = P$ where P is $\mathcal{P}$-positive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}} [P]P^\perp}{P^\perp \vdash^{\mathcal{P}'} [P]P^\perp}}{P^\perp \vdash^{\mathcal{P}'} [\bullet]P^\perp}}{P^\perp \vdash^{\mathcal{P}'} [P^\perp]}}{P^\perp, \Delta^\perp \vdash^{\mathcal{P}'} [P^\perp]}}{P^\perp \vdash^{\mathcal{P}} [P^\perp]\Delta}}{\vdash^{\mathcal{P}} [P^\perp]P, \Delta}$$

If P is a literal, we complete the proof with the case just above. If it is not a literal, then P is smaller than $P^\perp$ and we complete the proof by applying the induction hypothesis on P.

□

We now want to show that all asynchronous rules are invertible in $\mathsf{LK}^+(\mathcal{T})$. We first start with the following lemma:

Lemma 23 (Generalised (Init) and negative Select)

The following rules are height-preserving admissible in $\mathsf{LK}^+(\mathcal{T})$:

$$(\mathsf{Init})\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\Delta} \qquad (\mathsf{Select}^-)\frac{\Gamma \vdash^{\mathcal{P};l^\perp} [l]\Delta}{\Gamma \vdash^{\mathcal{P};l^\perp} [\bullet]\Delta}$$

where $l^\perp \in \Gamma$ and it is not $\mathcal{P}$-negative in (Select^-). ※

Proof: For each rule, by induction on the proof of the premiss.

For (Init):

- if it is obtained by $(\wedge^-), (\vee^-), (\forall), (\bot^-)$, we can straightforwardly use the induction hypothesis on the premiss(es), and if it is $(\top^-)$ it is trivial;
- if it is obtained by

$$\frac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}]\Delta'}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta'}$$

then we can use the induction hypothesis on the premiss as $\mathsf{lit}_{\mathcal{P};A^\perp}(\Gamma, A^\perp), \mathsf{lit}_{\mathcal{L}}(\Delta'^\perp) = \mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(A^\perp, \Delta'^\perp)$;
- the last possible way to obtain it is with $\Delta = \emptyset$ and

$$\frac{\Gamma \vdash^{\mathcal{P}} [\bullet]N}{\Gamma \vdash^{\mathcal{P}} [N]}$$

for some N that is not $\mathcal{P}$-positive, and we conclude with (Init_2).

For (Select^-), first notice that l is $\mathcal{P}; l^\perp$-negative, and then:

- if again it is obtained by $(\wedge^-), (\vee^-), (\forall), (\bot^-)$, we can straightforwardly use the induction hypothesis on the premiss(es), and if it is $(\top^-)$ it is trivial;
- if it is obtained by

$$\frac{\Gamma, A^\perp \vdash^{\mathcal{P};l^\perp;A^\perp} [l]\Delta'}{\Gamma \vdash^{\mathcal{P};l^\perp} [l]A, \Delta'}$$

then we can use the induction hypothesis on the premiss, if A is not $l^\perp$ (so that $\mathcal{P}; l^\perp; A^\perp = \mathcal{P}; A^\perp; l^\perp$ and $l^\perp$ is not $\mathcal{P}; A^\perp$-negative); if $A = l^\perp$, then we build

$$(\mathsf{Init}_2)\frac{\mathsf{lit}_{\mathcal{P};l^\perp}(\Gamma), \mathsf{lit}_{\mathcal{L}}(A^\perp, \Delta'^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P};l^\perp} [\bullet]A, \Delta'}$$

as $A \in \mathsf{lit}_{\mathcal{P};l^\perp}(\Gamma)$.
- the last possible way to obtain it is with $\Delta = \emptyset$ and

$$\frac{\dfrac{\Gamma, l^\perp \vdash^{\mathcal{P};l^\perp} [\bullet]}{\Gamma \vdash^{\mathcal{P};l^\perp} [\bullet]l}}{\Gamma \vdash^{\mathcal{P};l^\perp} [l]}$$

and we conclude with the height-preserving admissibility of contraction.

□

We can now state and prove the invertibility of asynchronous rules:

Lemma 24 (Invertibility of asynchronous rules)

All asynchronous rules are height-preserving invertible in $\mathsf{LK}^+(\mathcal{T})$. ※

Proof: By induction on the derivation proving the conclusion of the asynchronous rule considered.

- Inversion of $A\wedge^- B$: by case analysis on the last rule actually used
  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, C\wedge^- D, \Delta}$

    By induction hypothesis we get

    $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C, \Delta \qquad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C\wedge^- D, \Delta}$ and $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, C\wedge^- D, \Delta}$
  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, C\vee^- D, \Delta}$

    By induction hypothesis we get

    $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C\vee^- D, \Delta}$ and $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, C\vee^- D, \Delta}$
  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, C, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, (\forall x C), \Delta}\ x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta, A\wedge^- B)$

    By induction hypothesis we get

    $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, (\forall x C), \Delta}\ x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta, A)$ and $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, C, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, (\forall x C), \Delta}\ x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta, B)$
  - $\dfrac{\Gamma, C^\perp \vdash^{\mathcal{P};C^\perp} [\mathcal{X}]A\wedge^- B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, C, \Delta}$ C literal or $\mathcal{P}$-positive

    By induction hypothesis we get

    $\dfrac{\Gamma, C^\perp \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta}{\Gamma \vdash^{\mathcal{P};C^\perp} [\mathcal{X}]A, C, \Delta}$ C literal or $\mathcal{P}$-positive and $\dfrac{\Gamma, C^\perp \vdash^{\mathcal{P};C^\perp} [\mathcal{X}]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, C, \Delta}$ C literal or $\mathcal{P}$-positive
  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, \bot^-, \Delta}$

    By induction hypothesis we get

    $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \bot^-, \Delta}$ and $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, \bot^-, \Delta}$
  - $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B, \top^-, \Delta}$

    We get

    $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \top^-, \Delta}$ and $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, \top^-, \Delta}$
  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]A\wedge^- B, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D]A\wedge^- B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^+ D,]A\wedge^- B, \Delta}$

    By induction hypothesis we get

    $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]A, \Delta \qquad \Gamma \vdash^{\mathcal{P}} [D]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^+ D]A, \Delta}$ and $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]B, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^+ D]B, \Delta}$
  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]A\wedge^- B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^+ C_2]A\wedge^- B, \Delta}$

    By induction hypothesis we get

    $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^+ C_2]A, \Delta}$ and $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^+ C_2]B, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}C]A\wedge^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x C]A\wedge^{-}B, \Delta}$

By induction hypothesis we get

$$\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}C]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x C]A, \Delta} \quad \text{and} \quad \dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}C]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x C]B, \Delta}$$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]A\wedge^{-}B, \Delta}$

We get

$$\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]A, \Delta} \quad \text{and} \quad \dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]B, \Delta}$$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]A\wedge^{-}B, \Delta}$ p is $\mathcal{P}$-positive

We get

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]A, \Delta} \quad \text{and} \quad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]B, \Delta}$$

–

$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]A\wedge^{-}B, \Delta}$

We get

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]A, \Delta} \quad \text{and} \quad \dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]B, \Delta}$$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [P]A\wedge^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]A\wedge^{-}B, \Delta}$ where $P^{\perp} \in \Gamma$ is not $\mathcal{P}$-positive

By induction hypothesis we get

$$\dfrac{\Gamma \vdash^{\mathcal{P}} [P]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]A, \Delta} \quad \text{and} \quad \dfrac{\Gamma \vdash^{\mathcal{P}} [P]B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]B, \Delta}$$

- Inversion of $A\vee^{-}B$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, C\wedge^{-}D, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, C\wedge^{-}D, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, C\vee^{-}D, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, C\vee^{-}D, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, C, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, (\forall x C), \Delta}$ $x \notin \mathsf{FV}(\Gamma, \mathcal{X}, A\vee^{-}B, \Delta)$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, C, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}], A, B, (\forall x C), \Delta}$ $x \notin \mathsf{FV}(\Gamma, \mathcal{X}, A, B, \Delta)$

– $\dfrac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} [\mathcal{X}]A\vee^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, C, \Delta}$ C literal or $\mathcal{P}$-positive

By induction hypothesis we get

$$\frac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} [\mathcal{X}]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, C, \Delta}\ C \text{ literal or } \mathcal{P}\text{-positive}$$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, \bot^{-}, \Delta}$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, \bot^{-}, \Delta}$$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\vee^{-}B, \top^{-}, \Delta}$

We get

$$\frac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, B, \top^{-}, \Delta}$$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]A\vee^{-}B, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D]A\vee^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^{+}D]A\vee^{-}B, \Delta}$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} [C]A, B, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^{+}D]A, B, C\wedge^{-}D, \Delta}$$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]A\vee^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^{+}C_2]A\vee^{-}B, \Delta}$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} [C_i]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^{+}C_2]A, B, \Delta}$$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}C]A\vee^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists xC]A\vee^{-}B, \Delta}$

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!_{x}\}C]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists xC]A, B, \Delta}$$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]A\vee^{-}B, \Delta}$

We get

$$\frac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]A, B, \Delta}$$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]A\vee^{-}B, \Delta}\ p \text{ is } \mathcal{P}\text{-positive}$

We get

$$\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]A, B, \Delta}$$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]A\vee^{-}B, \Delta}$

We get

$$\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]A, B, \Delta}$$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [P]A\vee^{-}B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]A\vee^{-}B, \Delta}$ where $P^{\perp} \in \Gamma$ is not $\mathcal{P}$-positive

By induction hypothesis we get

$$\frac{\Gamma \vdash^{\mathcal{P}} [P]A, B, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]A, B, \Delta}$$

- Inversion of $\forall xA$
  – $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall xA), C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall xA), D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall xA), C\wedge^{-}D, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C\wedge^{-}D, \Delta}\, x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta)$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), C\vee^{-}D, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C\vee^{-}D, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), (\forall y D), \Delta}\, y \notin \mathsf{FV}(\Gamma, \mathcal{X}, (\forall x A), \Delta)$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, (\forall y D), \Delta}\, y \notin \mathsf{FV}(\Gamma, \mathcal{X}, A, \Delta)$

– $\dfrac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} [\mathcal{X}](\forall x A), \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), C, \Delta}$ C literal or $\mathcal{P}$-positive

By induction hypothesis we get $\dfrac{\Gamma, C^{\perp} \vdash^{\mathcal{P};C^{\perp}} [\mathcal{X}]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, C, \Delta}$ C literal or $\mathcal{P}$-positive

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), \bot^{-}, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \bot^{-}, \Delta}$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x A), \top^{-}, \Delta}$

We get $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A, \top^{-}, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C](\forall x A), \Delta \quad \Gamma \vdash^{\mathcal{P}} [D](\forall x A), \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^{+}D](\forall x A), \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]A, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^{+}D]A, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i](\forall x A), \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^{+}C_2](\forall x A), \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^{+}C_2]A, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!{}_{x}\}D](\forall x A), \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x D](\forall x A), \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^{t}\!/\!{}_{x}\}D]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x D]A, \Delta}$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}](\forall x A), C, \Delta}$

We get $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^{+}]A, \Delta}$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p](\forall x A), \Delta}\, p$ is $\mathcal{P}$-positive

We get $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp}, \mathsf{lit}_{\mathcal{L}}(\Delta^{\perp}) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]A, \Delta}$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet](\forall x A), \Delta}$

We get $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet] A, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [P](\forall x A), \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet](\forall x A), \Delta}$ where $P^\perp \in \Gamma$ is not $\mathcal{P}$-positive

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [P] A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet] A, \Delta}$

- Inversion of storing a literal or $\mathcal{P}$-positive formulae A

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, C \wedge^- D, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] C, \Delta \quad \Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] D, \Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] C \wedge^- D, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, C \vee^- D, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] C, D, \Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] C \vee^- D, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, (\forall x D), \Delta}\ x \notin \mathsf{FV}(\Gamma, \mathcal{X}, A, \Delta)$

By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] D, \Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] (\forall x D), \Delta}\ x \notin \mathsf{FV}(\Gamma, A^\perp, \mathcal{X}, \Delta)$

– $\dfrac{\Gamma, B^\perp \vdash^{\mathcal{P};B^\perp} [\mathcal{X}] A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, B, \Delta}$ B literal or $\mathcal{P}$-positive

We build $\dfrac{\Gamma, A^\perp, B^\perp \vdash^{\mathcal{P};A^\perp;B^\perp} [\mathcal{X}] \Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] B, \Delta}$ B literal or $\mathcal{P}$-positive

proving the premiss using the induction hypothesis in case $\mathcal{P}; B^\perp; A^\perp = \mathcal{P}; A^\perp; B^\perp$, which holds unless $A = B^\perp$ and $A \in \mathsf{U}_{\mathcal{P}}$.

In that case we have $\mathcal{P}; A^\perp = \mathcal{P}, A^\perp$, and we prove $\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] B, \Delta$ with (Init) (Lemma 23), as $\mathsf{lit}_{\mathcal{P};A^\perp}(\Gamma, A^\perp), \mathsf{lit}_{\mathcal{L}}(B^\perp, \Delta^\perp) \models_{\mathcal{T}}$.

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, \bot^-, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] \Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] \bot^-, \Delta}$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}] A, \top^-, \Delta}$

We get $\dfrac{}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\mathcal{X}] \top^-, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C] A, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D] A, \Delta}{\Gamma \vdash^{\mathcal{P}} [C \wedge^+ D] A, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P}} [C] \Delta \quad \Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [D] \Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [C \wedge^+ D] \Delta}$

- $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [C_1 \vee^+ C_2]A, \Delta}$

  By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [C_i]\Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [C_1 \vee^+ C_2]\Delta}$

- $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^t\!/\!_x\}D]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x D]A, \Delta}$

  By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\{{}^t\!/\!_x\}D]\Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\exists x D]\Delta}$

- $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]A, \Delta}$

  We get $\dfrac{}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\top^+]\Delta}$

- $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^\perp, \mathsf{lit}_{\mathcal{L}}(A^\perp, \Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]A, \Delta}$ p is $\mathcal{P}$-negative

  We get $\dfrac{\mathsf{lit}_{\mathcal{P};A^\perp}(\Gamma, A^\perp), p^\perp, \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [p]\Delta}$

  as p is also $\mathcal{P}; A^\perp$-positive.

- $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(A^\perp, \Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]A, \Delta}$

  We get $\dfrac{\mathsf{lit}_{\mathcal{P};A^\perp}(\Gamma, A^\perp), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\bullet]\Delta}$

- $\dfrac{\Gamma \vdash^{\mathcal{P}} [P]A, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]A, \Delta}$ where $P^\perp \in \Gamma$ is not $\mathcal{P}$-positive

  By induction hypothesis we get $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [P]\Delta}{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} [\bullet]\Delta}$

  using either (Select) or (Select^-) depending on whether P is $\mathcal{P}; A^\perp$-negative.

- Inversion of $(\perp^-)$

  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, C \wedge^- D, \Delta}$

    By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]C, \Delta \quad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]C \wedge^- D, \Delta}$

  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, C \vee^- D, \Delta}$

    By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]C, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]C \vee^- D, \Delta}$

  - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, (\forall x D), \Delta}$ $x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta)$

    By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]D, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}](\forall x D), \Delta}$ $x \notin \mathsf{FV}(\Gamma, \mathcal{X}, \Delta)$

  - $\dfrac{\Gamma, B^\perp \vdash^{\mathcal{P};B^\perp} [\mathcal{X}]\perp^-, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-, B, \Delta}$ B literal or $\mathcal{P}$-positive

By induction hypothesis we get $\dfrac{\Gamma, B^\perp \vdash^{\mathcal{P};B^\perp} [\mathcal{X}]\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B, \Delta}$ B literal or $\mathcal{P}$-positive

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\bot^-, \Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\bot^-, \bot^-, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\bot^-, \Delta}$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\bot^-, \top^-, \Delta}$

We get $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\top^-, \Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]\bot^-, \Delta \quad \Gamma \vdash^{\mathcal{P}} [D]\bot^-, \Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^+ D]\bot^-, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [C]\Delta \quad \Gamma \vdash^{\mathcal{P}} [D]\Delta}{\Gamma \vdash^{\mathcal{P}} [C\wedge^+ D]\Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]\Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^+ C_2]\bot^-, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [C_i]\Delta}{\Gamma \vdash^{\mathcal{P}} [C_1\vee^+ C_2]\Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^t\!/\!{}_x\}D]\bot^-, \Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x D]\bot^-, \Delta}$

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{{}^t\!/\!{}_x\}D]\Delta}{\Gamma \vdash^{\mathcal{P}} [\exists x D]\Delta}$

– $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]\bot^-, \Delta}$

We get $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]\Delta}$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^\perp, \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]\bot^-, \Delta}$ p is $\mathcal{P}$-positive

By induction hypothesis we get $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^\perp, \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma, A^\perp \vdash^{\mathcal{P}} [p]\Delta}$

– $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet]\bot^-, \Delta}$

By induction hypothesis we get $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma, A^\perp \vdash^{\mathcal{P}} [\bullet]\Delta}$

– $\dfrac{\Gamma \vdash^{\mathcal{P}} [P]\bot^-, \Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]\bot^-, \Delta}$ where $P^\perp \in \Gamma$ is not $\mathcal{P}$-positive

By induction hypothesis we get $\dfrac{\Gamma \vdash^{\mathcal{P}} [P]\Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta}$

- Inversion of $\top^-$: nothing to do.

□

Now that we have proved the invertibility of asynchronous rules, we can use it to transform any proof of $\mathsf{LK}^+(\mathcal{T})$ into a proof of $\mathsf{LK}^p(\mathcal{T})$.

Lemma 25 (Encoding $\mathsf{LK}^+(\mathcal{T})$ in $\mathsf{LK}^p(\mathcal{T})$)

1. If $\Gamma \vdash^{\mathcal{P}} [A]$ is provable in $\mathsf{LK}^+(\mathcal{T})$, then $\Gamma \vdash^{\mathcal{P}} [A]$ is provable in $\mathsf{LK}^p(\mathcal{T})$.
2. If $\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta$ is provable in $\mathsf{LK}^+(\mathcal{T})$, then $\Gamma \vdash^{\mathcal{P}} \Delta$ is provable in $\mathsf{LK}^p(\mathcal{T})$.

※

Proof: By simultaneous induction on the assumed derivation.

1. For the first item we get, by case analysis on the last rule of the derivation:
   - $\dfrac{\Gamma \vdash^{\mathcal{P}} [A_1] \quad \Gamma \vdash^{\mathcal{P}} [A_2]}{\Gamma \vdash^{\mathcal{P}} [A_1 \wedge^+ A_2]}$ with $A = A_1 \wedge^+ A_2$.

     The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [A_1]$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} [A_1]$ and the induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [A_2]$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} [A_2]$. We get:

     $$\dfrac{\Gamma \vdash^{\mathcal{P}} [A_1] \quad \Gamma \vdash^{\mathcal{P}} [A_2]}{\Gamma \vdash^{\mathcal{P}} [A_1 \wedge^+ A_2]}$$
   - $\dfrac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]}$ with $A = A_1 \vee^+ A_2$.

     The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [A_i]$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} [A_i]$. We get:

     $$\dfrac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]}$$
   - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\{t/x\}A]}{\Gamma \vdash^{\mathcal{P}} [\exists x A]}$ with $A = \exists x A$.

     The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\{t/x\}A]$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} [\{t/x\}A]$. We get:

     $$\dfrac{\Gamma \vdash^{\mathcal{P}} [\{t/x\}A]}{\Gamma \vdash^{\mathcal{P}} [\exists x A]}$$
   - $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]}$ with $A = p$ where p is a $\mathcal{P}$-positive literal.

     We can perform the same step in $\mathsf{LK}^p(\mathcal{T})$:

     $$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), p^{\perp} \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [p]}$$
   - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\bullet]N}{\Gamma \vdash^{\mathcal{P}} [N]}$ with $A = N$ and N is not $\mathcal{P}$-positive.

     The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\bullet]N$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} N$. We get:

     $$\dfrac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]}$$
2. For the second item, we use the height-preserving invertibility of the asynchronous rules, so that we can assume without loss of generality that if Δ is not empty then the last rule of the derivation decomposes one of its formulae.
   - $\dfrac{\Gamma \vdash^{\mathcal{P}} [\bullet]A_1, \Delta_1 \quad \Gamma \vdash^{\mathcal{P}} [\bullet]A_2, \Delta_1}{\Gamma \vdash^{\mathcal{P}} [\bullet]A_1 \wedge^- A_2, \Delta_1}$ with $\Delta = A_1 \wedge^- A_2, \Delta_1$.

     The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\bullet]A_1, \Delta_1$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A_1, \Delta_1$ and the induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\bullet]A_2, \Delta_2$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A_2, \Delta_2$. We get:

     $$\dfrac{\Gamma \vdash^{\mathcal{P}} A_1, \Delta_1 \quad \Gamma \vdash^{\mathcal{P}} A_2, \Delta_1}{\Gamma \vdash^{\mathcal{P}} A_1 \wedge^- A_2, \Delta_1}$$

- $\dfrac{\Gamma \vdash^{\mathcal{P}} [\bullet] A_1, A_2, \Delta_1}{\Gamma \vdash^{\mathcal{P}} [\bullet] A_1 \vee^- A_2, \Delta_1}$ with $\Delta = A_1 \vee^- A_2, \Delta_1$.

  The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\bullet] A_1, A_2, \Delta_1$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A_1, A_2, \Delta_1$ and we get:

$$\dfrac{\Gamma \vdash^{\mathcal{P}} A_1, A_2, \Delta_1}{\Gamma \vdash^{\mathcal{P}} A_1 \vee^- A_2, \Delta_1}$$

- $\dfrac{\Gamma \vdash^{\mathcal{P}} [\bullet] A, \Delta_1}{\Gamma \vdash^{\mathcal{P}} [\bullet] \forall x A, \Delta_1}\, x \notin \mathsf{FV}(\Gamma, \Delta_1)$ with $\Delta = \forall x A, \Delta_1$.

  The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\bullet] A, \Delta_1$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A, \Delta_1$. We get:

$$\dfrac{\Gamma \vdash^{\mathcal{P}} A, \Delta_1}{\Gamma \vdash^{\mathcal{P}} \forall x A, \Delta_1}\, x \notin \mathsf{FV}(\Gamma, \Delta_1)$$

- $\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P}; A^\perp} [\bullet] \Delta_1}{\Gamma \vdash^{\mathcal{P}} [\bullet] A, \Delta_1}$ with $\Delta = A, \Delta_1$ and A is a literal or is $\mathcal{P}$-positive.

  The induction hypothesis on $\Gamma, A^\perp \vdash^{\mathcal{P}; A^\perp}_{\mathsf{LK}^+(\mathcal{T})} [\bullet] \Delta_1$ gives $\Gamma, A^\perp \vdash^{\mathcal{P}; A^\perp}_{\mathsf{LK}^p(\mathcal{T})} \Delta_1$. We get:

$$\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P}; A^\perp} \Delta_1}{\Gamma \vdash^{\mathcal{P}} A, \Delta_1}$$

- $\dfrac{\Gamma \vdash^{\mathcal{P}} [\bullet] \Delta_1}{\Gamma \vdash^{\mathcal{P}} [\bullet] \bot^-, \Delta_1}$ with $\Delta = \bot^-, \Delta_1$.

  The induction hypothesis on $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [\bullet] \Delta_1$ gives $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} \Delta_1$. We get:

$$\dfrac{\Gamma \vdash^{\mathcal{P}} \Delta_1}{\Gamma \vdash^{\mathcal{P}} \bot^-, \Delta_1}$$

- $\dfrac{}{\Gamma \vdash^{\mathcal{P}} [\bullet] \top^-, \Delta_1}$ with $\Delta = \top^-, \Delta_1$.

  We get:

$$\dfrac{}{\Gamma \vdash^{\mathcal{P}} \top^-, \Delta_1}$$

- $\dfrac{\Gamma, P^\perp \vdash^{\mathcal{P}} [P] \Delta}{\Gamma, P^\perp \vdash^{\mathcal{P}} [\bullet] \Delta}$ where P is not $\mathcal{P}$-negative.

  As already mentioned, we can assume without loss of generality that Δ is empty. The induction hypothesis on $\Gamma, P^\perp \vdash^{\mathcal{P}}_{\mathsf{LK}^+(\mathcal{T})} [P]$ gives $\Gamma, P^\perp \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} [P]$. We get:

$$\dfrac{\Gamma, P^\perp \vdash^{\mathcal{P}} [P]}{\Gamma, P^\perp \vdash^{\mathcal{P}}}$$

- $\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma), \mathsf{lit}_{\mathcal{L}}(\Delta^\perp) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [\bullet] \Delta}$

  As already mentioned, we can assume without loss of generality that Δ is empty. We get:

$$\dfrac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

□

Lemma 26 We have:

1. $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} \top^{+\perp}, \top^-$, and
2. $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} \top^{-\perp}, \top^+$, and
3. $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} (A\wedge^+ B)^\perp, (A\wedge^- B)$, and
4. $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} (A\wedge^- B)^\perp, (A\wedge^+ B)$, provided that sequent is safe.

※

Proof:

1. For the first item we get:

$$\dfrac{}{\vdash^{\mathcal{P}} \top^{+\perp}, \top^-}$$

2. For the second item we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\top^-, \top^{+\perp} \vdash^{\mathcal{P}} [\top^+]}}{\top^-, \top^{+\perp} \vdash^{\mathcal{P}}}}{\top^- \vdash^{\mathcal{P}} \top^+}}{\vdash^{\mathcal{P}} \top^{-\perp}, \top^+}$$

3. For the third item we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P};A} [A^\perp]B^\perp, A}{A \vdash^{\mathcal{P};A} [A^\perp]B^\perp, A}}{A \vdash^{\mathcal{P};A} [\bullet]B^\perp, A}}{\vdash^{\mathcal{P}} [\bullet]A^\perp, B^\perp, A} \qquad \dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P};B} [B^\perp]A^\perp, B}{B \vdash^{\mathcal{P};B} [B^\perp]A^\perp, B}}{B \vdash^{\mathcal{P};B} [\bullet]A^\perp, B}}{\vdash^{\mathcal{P}} [\bullet]A^\perp, B^\perp, B}}{\vdash^{\mathcal{P}} [\bullet](A^\perp \vee^- B^\perp), (A\wedge^- B)}}{\vdash^{\mathcal{P}} [\bullet](A\wedge^+ B)^\perp, (A\wedge^- B)}}{\vdash^{\mathcal{P}} (A\wedge^+ B)^\perp, (A\wedge^- B)}\;\text{Lemma 25(2)}$$

Both left hand side and right hand side can be closed by Lemma 22.

4. For the fourth item, we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}} [A^\perp]A}{\vdash^{\mathcal{P}} [A^\perp \vee^+ B^\perp]A}}{A\wedge^- B \vdash^{\mathcal{P}} [A^\perp \vee^+ B^\perp]A}}{A\wedge^- B \vdash^{\mathcal{P}} [\bullet]A}}{A\wedge^- B \vdash^{\mathcal{P}} A}\;\text{Lemma 25(2)}}{(A\wedge^- B), (A^\perp \vee^- B^\perp) \vdash^{\mathcal{P}} A} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}} [B^\perp]B}{\vdash^{\mathcal{P}} [A^\perp \vee^+ B^\perp]B}}{A\wedge^- B \vdash^{\mathcal{P}} [A^\perp \vee^+ B^\perp]B}}{A\wedge^- B \vdash^{\mathcal{P}} [\bullet]B}}{A\wedge^- B \vdash^{\mathcal{P}} B}\;\text{Lemma 25(2)}}{A\wedge^- B \vdash^{\mathcal{P}} A^\perp, B}}{(A\wedge^- B), (A^\perp \vee^- B^\perp) \vdash^{\mathcal{P}} A^\perp, B} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}} [A]A^\perp, B^\perp \qquad \vdash^{\mathcal{P}} [B]A^\perp, B^\perp}{\vdash^{\mathcal{P}} [A\wedge^+ B]A^\perp, B^\perp}}{A^\perp \vee^- B^\perp \vdash^{\mathcal{P}} [A\wedge^+ B]A^\perp, B^\perp}}{A^\perp \vee^- B^\perp \vdash^{\mathcal{P}} [\bullet]A^\perp, B^\perp}}{A^\perp \vee^- B^\perp \vdash^{\mathcal{P}} A^\perp, B^\perp}\;\text{Lemma 25(2)}}{(A\wedge^- B), (A^\perp \vee^- B^\perp) \vdash^{\mathcal{P}} A^\perp, B^\perp}}{(A\wedge^- B), (A^\perp \vee^- B^\perp) \vdash^{\mathcal{P}} A^\perp}\;\mathsf{cut}_7}{\dfrac{(A\wedge^- B), (A\wedge^+ B)^\perp \vdash^{\mathcal{P}}}{\vdash^{\mathcal{P}} (A\wedge^- B)^\perp, (A\wedge^+ B)}}\;\mathsf{cut}_7$$

All branches are closed by Lemma 22. □

Lemma 27

If $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} \Delta, C$ and $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} D, C^\perp$ then $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} \Delta, D$, provided that sequent is safe. ※

Proof:

$$\dfrac{\dfrac{\Gamma \vdash^{\mathcal{P}} \Delta, C}{\Gamma \vdash^{\mathcal{P}} D, \Delta, C} \qquad \dfrac{\Gamma \vdash^{\mathcal{P}} D, C^{\perp}}{\Gamma \vdash^{\mathcal{P}} \Delta, D, C^{\perp}}}{\Gamma \vdash^{\mathcal{P}} \Delta, D}\,\mathsf{cut}_7$$

□

Corollary 28 (Changing the polarity of connectives) Provided those sequents are safe,

1. If $\Gamma \vdash^{\mathcal{P}} \top^{+}, \Delta$ then $\Gamma \vdash^{\mathcal{P}} \top^{-}, \Delta$;
2. If $\Gamma \vdash^{\mathcal{P}} \top^{-}, \Delta$ then $\Gamma \vdash^{\mathcal{P}} \top^{+}, \Delta$;
3. If $\Gamma \vdash^{\mathcal{P}} \perp^{+}, \Delta$ then $\Gamma \vdash^{\mathcal{P}} \perp^{-}, \Delta$;
4. If $\Gamma \vdash^{\mathcal{P}} \perp^{-}, \Delta$ then $\Gamma \vdash^{\mathcal{P}} \perp^{+}, \Delta$;
5. If $\Gamma \vdash^{\mathcal{P}} A \wedge^{+} B, \Delta$ then $\Gamma \vdash^{\mathcal{P}} A \wedge^{-} B, \Delta$;
6. If $\Gamma \vdash^{\mathcal{P}} A \wedge^{-} B, \Delta$ then $\Gamma \vdash^{\mathcal{P}} A \wedge^{+} B, \Delta$;
7. If $\Gamma \vdash^{\mathcal{P}} A \vee^{+} B, \Delta$ then $\Gamma \vdash^{\mathcal{P}} A \vee^{-} B, \Delta$;
8. If $\Gamma \vdash^{\mathcal{P}} A \vee^{-} B, \Delta$ then $\Gamma \vdash^{\mathcal{P}} A \vee^{+} B, \Delta$.

Furthermore, notice that in each implication, the safety of one sequent implies the safety of the other. ※

Proof:

1. By Lemma 27 and Lemma 26(1).
2. By Lemma 27 and Lemma 26(2).
3. By Lemma 27 and Lemma 26(1).
4. By Lemma 27 and Lemma 26(2).
5. By Lemma 27 and Lemma 26(3).
6. By Lemma 27 and Lemma 26(4).
7. By Lemma 27 and Lemma 26(3).
8. By Lemma 27 and Lemma 26(4).

□

We have proven that changing the polarities of the connectives that are present in a sequent, does not change the provability of that sequent in $\mathsf{LK}^p(\mathcal{T})$.

3.7 Completeness

$\mathsf{LK}^p(\mathcal{T})$ is a complete system for first-order logic modulo a theory. To show this, we review the grammar of first-order formulae and map those formulae to polarised formulae.

Definition 19 (Plain formulae) Let P^a_Σ be a sub-signature of the first-order predicate signature P_Σ such that for every predicate symbol P/n of P_Σ, P/n is in P^a_Σ if and only if $P^\perp/n$ is not in P^a_Σ.
Let $\mathcal{A}$ be the subset of $\mathcal{L}$ consisting of those literals whose predicate symbols are in P^a_Σ. Literals in $\mathcal{A}$, denoted a, a', etc, are called *atoms*.
The formulae of first-order logic, here called *plain formulae*, are given by the following grammar:

$$\text{Plain formulae} \quad A, B, \ldots \quad ::= \quad a \mid A \vee B \mid A \wedge B \mid \forall x A \mid \exists x A \mid \neg A$$

where a ranges over atoms. ※

DEFINITION 20 (ψ) Let ψ be the function that maps every plain formula to a set of formulae (in the sense of Definition 13) defined as follows:

$$\begin{array}{lll}
\psi(a) & := & \{a\}\\
\psi(A\wedge B) & := & \{A'\wedge^- B', A'\wedge^+ B' \mid A'\in\psi(A), B'\in\psi(B)\}\\
\psi(A\vee B) & := & \{A'\vee^- B', A'\vee^+ B' \mid A'\in\psi(A), B'\in\psi(B)\}\\
\psi(\exists xA) & := & \{\exists xA' \mid A'\in\psi(A)\}\\
\psi(\forall xA) & := & \{\forall xA' \mid A'\in\psi(A)\}\\
\psi(\neg A) & := & \{A'^{\perp} \mid A'\in\psi(A)\}\\
\psi(\Delta, A) & := & \{\Delta', A' \mid \Delta'\in\psi(\Delta), A'\in\psi(A)\}\\
\psi(\emptyset) & := & \emptyset
\end{array}$$

※

REMARK 29 1. $\psi(A)\neq\emptyset$

2. If $A'\in\psi(A)$, then $\{{}^t\!/\!_x\}A'\in\psi(\{{}^t\!/\!_x\}A')$.
3. If $C'\in\psi(\{{}^t\!/\!_x\}A)$, then $C'=\{{}^t\!/\!_x\}A'$ for some $A'\in\psi(A)$.

NOTATION 21 When F is a plain formula and Ψ is a set of plain formulae, $\Psi\models F$ means that Ψ entails F in first-order classical logic.
Given a theory $\mathcal{T}$ (given by a semantical inconsistency predicate), we define the set of all *theory lemmas* as

$$\Psi_{\mathcal{T}} := \{l_1\vee\cdots\vee l_n \mid \psi(l_1)^{\perp},\cdots,\psi(l_n)^{\perp}\models_{\mathcal{T}}\}$$

We generalise the notation $\models_{\mathcal{T}}$ to write $\Psi\models_{\mathcal{T}} F$ when $\Psi_{\mathcal{T}},\Psi\models F$, in which case we say that F is a *semantical consequence* of Ψ. ※

NOTATION 22 In the rest of this section we will use the notation $A\wedge^? B$ (resp. $A\vee^? B$) to ambiguously represent either $A\wedge^+ B$ or $A\wedge^- B$ (resp. $A\vee^+ B$ or $A\vee^- B$). This will make the proofs more compact, noticing that Corollary 28(2) and 28(4) respectively imply the admissibility in $\mathsf{LK}^p(\mathcal{T})$ of

$$\frac{\Gamma\vdash^{\mathcal{P}}\Delta, A\wedge^- B}{\Gamma\vdash^{\mathcal{P}}\Delta, A\wedge^? B}\qquad\frac{\Gamma\vdash^{\mathcal{P}}\Delta, A\vee^- B}{\Gamma\vdash^{\mathcal{P}}\Delta, A\vee^? B}$$

provided the sequents are safe (and note that safety of the conclusion entails safety of the premiss). ※

LEMMA 30 (Equivalence between different polarisations)
For all $A', A''\in\psi(A)$, we have $\Gamma\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A', A''^{\perp},\Delta$, provided the sequent is safe. ※

Proof: In the proof below, for any formula A, the notations A' and A'' will systematically designate elements of $\psi(A)$.

The proof is by induction on A:

1. $A=a$
   Let $A', A''\in\psi(a)=\{a\}$. Therefore $A'=A''=A=a$.

$$\cfrac{\cfrac{}{\Gamma,\psi^{\perp}(a),\psi(a),\Gamma'\vdash^{\mathcal{P}'}}(\mathsf{Id}_2)}{\Gamma\vdash^{\mathcal{P}}\psi(a),\psi^{\perp}(a),\Delta}$$

2. $A=A_1\wedge A_2$
   Let $A'_1, A''_1\in\psi(A_1)$, $A'_2, A''_2\in\psi(A_2)$ and $A'=A'_1\wedge^? A'_2$, $A''=A''_1\wedge^? A''_2$.

$$\cfrac{\cfrac{\cfrac{\cfrac{\Gamma\vdash^{\mathcal{P}} A'_1, A''^{\perp}_1,\Delta}{\Gamma\vdash^{\mathcal{P}} A'_1, A''^{\perp}_1, A''^{\perp}_2,\Delta}\qquad\cfrac{\Gamma\vdash^{\mathcal{P}} A'_2, A''^{\perp}_2,\Delta}{\Gamma\vdash^{\mathcal{P}} A'_2, A''^{\perp}_1, A''^{\perp}_2,\Delta}}{\Gamma\vdash^{\mathcal{P}} A'_1\wedge^- A'_2, A''^{\perp}_1\vee^- A''^{\perp}_2,\Delta}}{\Gamma\vdash^{\mathcal{P}} A', A''^{\perp}_1\vee^- A''^{\perp}_2,\Delta}}{\Gamma\vdash^{\mathcal{P}} A', A''^{\perp},\Delta}$$

We can complete the proof on the left-hand side by applying the induction hypothesis on A_1 and on the right-hand side by applying the induction hypothesis on A_2.

3. $A = A_1 \vee A_2$

   By symmetry, using the previous case.

4. $A = \forall x A_1$

   Let $A' = \forall x A'_1$ and $A'' = \forall x A''_1$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}'} [{A''_1}^{\perp}]A''_1}{\vdash^{\mathcal{P}'} [\exists x A''_1]A''_1}}{\Gamma, \forall x A''_1 \vdash^{\mathcal{P}} [\bullet]A''_1, \Delta}}{\Gamma \vdash^{\mathcal{P}} A''_1, \exists x {A''_1}^{\perp}, \Delta}\text{Lemma 25(2)} \qquad \Gamma \vdash^{\mathcal{P}} A'_1, {A''_1}^{\perp}, \Delta}{\dfrac{\Gamma \vdash^{\mathcal{P}} A'_1, \exists x {A''_1}^{\perp}, \Delta}{\Gamma \vdash^{\mathcal{P}} \forall x A'_1, \exists x {A''_1}^{\perp}, \Delta}}\text{Lemma 27}$$

   We can complete the proof on the left-hand side by Lemma 22 and the right-hand side by applying the induction hypothesis on A_1.

5. $A = \exists x A_1$

   By symmetry, using the previous case.

6. $A = \neg A_1$

   Let $A', A'' \in \psi(\neg A_1)$.

   Let $A' = {A'_1}^{\perp}$ with $A'_1 \in \psi(A_1)$ and $A'' = {A''_1}^{\perp}$ with $A''_1 \in \psi(A_1)$.

   The induction hypothesis on A_1 we get: $\Gamma \vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A', {A''}^{\perp}, \Delta$ and we are done.

$\square$

We have seen in Section 3.6 that the polarity of connectives does not change provability, but in Example 1 we have seen that the polarity of literals *does* affect provability. While Example 2 hinted at how to align the provability *with* cuts with the provability *without* cuts, the two examples showed that those two coinciding provabilities may well fail to be complete (w.r.t. an intended theory) if the polarisation set $\mathcal{P}$ is ill-chosen. Therefore, we identify a criterion on polarisation sets, which will ensure completeness:

DEFINITION 23 (Theory restricting) A polarisation set $\mathcal{P}$ *does not restrict* the theory $\mathcal{T}$ if for all sets $\mathcal{B}$ of literals that are semantically inconsistent (i.e. $\mathcal{B} \models_{\mathcal{T}}$), there is a subset $\mathcal{B}' \subseteq \mathcal{B}$ that is already semantically inconsistent and such that at most one literal of $\mathcal{B}'$ is $\mathcal{P}$-negative. ※

REMARK 31 The empty polarisation set restricts no theories. The empty theory is restricted by no polarisation sets (if $\mathcal{B}$ is syntactically inconsistent, then is contains both l and $l^{\perp}$, so taking the inconsistent $\mathcal{B}' := \{l, l^{\perp}\}$, at most one of its two elements is $\mathcal{P}$-negative).

THEOREM 32 (Completeness of $\mathsf{LK}^p(\mathcal{T})$) Assume $\mathcal{P}$ does not restrict $\mathcal{T}$ and $\Delta \models_{\mathcal{T}} A$. Then for all $A' \in \psi(A)$ and $\Delta' \in \psi(\Delta)$, we have $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A', {\Delta'}^{\perp}$, provided that sequent is safe. ※

Proof: We prove a slightly more general statement:
for all $A' \in \psi(A)$ and all multisets Δ' of formulae that contain an element of $\psi(\Delta)$ as a sub-multiset, we have $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A', {\Delta'}^{\perp}$, provided that sequent is safe.

We characterise $\Delta \models_{\mathcal{T}} A$ by the derivability of the sequent $\Psi_{\mathcal{T}}, \Delta \vdash A$ in a standard natural deduction system for first-order classical logic. We write $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A$ for this derivability property.

For any formula A, the notation A' will systematically designate an element of $\psi(A)$.

The proof is by induction of $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A$, and case analysis on the last rule:

- Axiom:
$$\frac{}{\Psi_{\mathcal{T}}, \Delta \vdash A}\ A \in \Psi_{\mathcal{T}}, \Delta$$
By case analysis:
– If $A \in \Delta$ then we prove $\vdash^{\mathcal{P}} A', {\Delta'}^{\bot}$ with $A', A'' \in \psi(A)$ and $A'' \in \Delta'$, using Lemma 30.
– If $A \in \Psi_{\mathcal{T}}$ then A is of the form $l_1 \vee \cdots \vee l_n$ with $\psi(l_1)^{\bot}, \dots, \psi(l_n)^{\bot} \models_{\mathcal{T}}$.
Let $\{\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot}\}$ be a subset of $\{\psi(l_1)^{\bot}, \dots, \psi(l_n)^{\bot}\}$ that is already semantically inconsistent and such that at most one literal is $\mathcal{P}$-negative, say possibly $\psi(l'_m)^{\bot}$.
Let $C' \in \psi(A)$. C' is of the form $\psi(l_1) \vee^{?} \cdots \vee^{?} \psi(l_n)$.
We build
$$\dfrac{\dfrac{\dfrac{\dfrac{\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot} \vdash^{\mathcal{P}'}}{\vdash^{\mathcal{P}} \psi(l'_1), \dots, \psi(l'_m)}}{\vdash^{\mathcal{P}} {\Delta'}^{\bot}, \psi(l_1), \dots, \psi(l_n)}}{\vdash^{\mathcal{P}} {\Delta'}^{\bot}, \psi(l_1) \vee^{-} \cdots \vee^{-} \psi(l_n)}}{\vdash^{\mathcal{P}} {\Delta'}^{\bot}, C'}$$
where $\mathcal{P}' := \mathcal{P}; \psi(l'_1)^{\bot}; \dots; \psi(l'_m)^{\bot}$.
If $\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot}$ is syntactically inconsistent, we close with Id_2.
Otherwise
$$\mathcal{P}; \psi(l'_1)^{\bot}; \dots; \psi(l'_{m-1})^{\bot} = \mathcal{P}, \psi(l'_1)^{\bot}, \dots, \psi(l'_{m-1})^{\bot}$$
as none of the $\psi(l'_i)^{\bot}$, for $1 \le i \le m-1$, is $\mathcal{P}$-negative. And for all i such that $1 \le i \le m-1$, the literal $\psi(l'_i)^{\bot}$ is $\mathcal{P}'$-positive.
Now if $\psi(l'_m)^{\bot}$ is $\mathcal{P}'$-positive as well, we have
$$\mathsf{lit}_{\mathcal{P}'}(\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot}) = \psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot}$$
and we can close with (Init_2).
If $\psi(l'_m)^{\bot}$ is not $\mathcal{P}'$-positive, we simply have
$$\mathsf{lit}_{\mathcal{P}'}(\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot}) = \psi(l'_1)^{\bot}, \dots, \psi(l'_{m-1})^{\bot}$$
but we can still build
$$\dfrac{(\mathsf{Init}_1)\dfrac{\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot} \models_{\mathcal{T}}}{\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot} \vdash^{\mathcal{P}'} [\psi(l'_m)]}}{\psi(l'_1)^{\bot}, \dots, \psi(l'_m)^{\bot} \vdash^{\mathcal{P}'}}$$

- And Intro:
$$\frac{\Psi_{\mathcal{T}}, \Delta \vdash A_1 \quad \Psi_{\mathcal{T}}, \Delta \vdash A_2}{\Psi_{\mathcal{T}}, \Delta \vdash A_1 \wedge A_2}$$
$A' \in \psi(A_1 \wedge A_2)$ is of the form $A'_1 \wedge^{?} A'_2$ with $A'_1 \in \psi(A_1)$ and $A'_2 \in \psi(A_2)$.
Since $\vdash^{\mathcal{P}} A'_1 \wedge^{?} A'_2, {\Delta'}^{\bot}$ is assumed to be safe, $\vdash^{\mathcal{P}} A'_1, {\Delta'}^{\bot}$ and $\vdash^{\mathcal{P}} A'_2, {\Delta'}^{\bot}$ are also safe, and we can apply the induction hypothesis
– on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A_1$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A'_1, {\Delta'}^{\bot}$
– and on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A_2$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A'_2, {\Delta'}^{\bot}$.
We build:
$$\dfrac{\dfrac{\vdash^{\mathcal{P}} A'_1, {\Delta'}^{\bot} \quad \vdash^{\mathcal{P}} A'_2, {\Delta'}^{\bot}}{\vdash^{\mathcal{P}} A'_1 \wedge^{-} A'_2, {\Delta'}^{\bot}}}{\vdash^{\mathcal{P}} A'_1 \wedge^{?} A'_2, {\Delta'}^{\bot}}$$

- And Elim:

$$\frac{\Psi_{\mathcal{T}}, \Delta \vdash A_1 \wedge A_{-1}}{\Psi_{\mathcal{T}}, \Delta \vdash A_i}$$

with $i \in \{1, -1\}$.
Since $\psi(A_{-i}) \neq \emptyset$, let $A'_{-i} \in \psi(A_{-i})$ and $C' = A'_1 \wedge^- A'_{-1}$ ($C' \in \psi(A_1 \wedge A_{-1})$).
Since $\vdash^{\mathcal{P}} A'_i, {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} C', A'_i, {\Delta'}^{\perp}$ is also safe, and we can apply the induction hypothesis on $\Psi_{\mathcal{T}}, \Delta \vdash A_1 \wedge A_{-1}$ (with ${A'_i}^{\perp}, \Delta'$ and C') to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} C', A'_i, {\Delta'}^{\perp}$.
We finally get:

$$\dfrac{\dfrac{\vdash^{\mathcal{P}} C', A'_i, {\Delta'}^{\perp}}{\vdash^{\mathcal{P}} A'_i, A'_i, {\Delta'}^{\perp}}\text{Lemma 9}}{\vdash^{\mathcal{P}} A'_i, {\Delta'}^{\perp}}\mathsf{C}_r$$

- Or Intro:

$$\frac{\Psi_{\mathcal{T}}, \Delta \vdash A_i}{\Psi_{\mathcal{T}}, \Delta \vdash A_1 \vee A_{-1}}$$

$A' \in \psi(A_1 \vee A_{-1})$ is of the form $A'_1 \vee^? A'_{-1}$ with $A'_1 \in \psi(A_1)$ and $A'_{-1} \in \psi(A_{-1})$.
Since $\vdash^{\mathcal{P}} A'_1 \vee^? A'_{-1}, {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} A'_1, A'_{-1}, {\Delta'}^{\perp}$ is also safe, and we can apply the induction hypothesis on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A_i$ (with ${A'_{-i}}^{\perp}, \Delta'$ and A'_i) to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A'_1, A'_{-1}, {\Delta'}^{\perp}$ and we build:

$$\dfrac{\dfrac{\vdash^{\mathcal{P}} A'_1, A'_{-1}, {\Delta'}^{\perp}}{\vdash^{\mathcal{P}} A'_1 \vee^- A'_{-1}, {\Delta'}^{\perp}}}{\vdash^{\mathcal{P}} A'_1 \vee^? A'_{-1}, {\Delta'}^{\perp}}$$

- Or Elim:

$$\frac{\Psi_{\mathcal{T}}, \Delta \vdash A_1 \vee A_2 \quad \Psi_{\mathcal{T}}, \Delta, A_1 \vdash C \quad \Psi_{\mathcal{T}}, \Delta, A_2 \vdash C}{\Psi_{\mathcal{T}}, \Delta \vdash C}$$

Let $D' = A'_1 \vee^- A'_2$ with $A'_1 \in \psi(A_1)$ and $A'_2 \in \psi(A_2)$.
Since $\vdash^{\mathcal{P}} C', {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} C', {A'_1}^{\perp}, {\Delta'}^{\perp}$ and $\vdash^{\mathcal{P}} C', {A'_2}^{\perp}, {\Delta'}^{\perp}$ and $\vdash^{\mathcal{P}} C', D', {\Delta'}^{\perp}$ are also safe, and we can apply the induction hypothesis
– on $\Psi_{\mathcal{T}}, \Delta, A_1 \vdash_{\mathsf{FOL}} C$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} C', {A'_1}^{\perp}, {\Delta'}^{\perp}$
– on $\Psi_{\mathcal{T}}, \Delta, A_2 \vdash_{\mathsf{FOL}} C$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} C', {A'_2}^{\perp}, {\Delta'}^{\perp}$.
– and on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A_1 \vee A_2$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} C', D', {\Delta'}^{\perp}$.
We build:

$$\dfrac{\vdash^{\mathcal{P}} D', C', {\Delta'}^{\perp} \qquad \dfrac{\dfrac{\dfrac{\vdash^{\mathcal{P}} {A'_1}^{\perp}, C', {\Delta'}^{\perp} \quad \vdash^{\mathcal{P}} {A'_2}^{\perp}, C', {\Delta'}^{\perp}}{\vdash^{\mathcal{P}} {A'_1}^{\perp} \wedge^- {A'_2}^{\perp}, C', {\Delta'}^{\perp}}}{\vdash^{\mathcal{P}} {A'_1}^{\perp} \wedge^+ {A'_2}^{\perp}, C', {\Delta'}^{\perp}}}{\vdash^{\mathcal{P}} (A'_1 \vee^- A'_2)^{\perp}, C', {\Delta'}^{\perp}}}{\vdash^{\mathcal{P}} C', {\Delta'}^{\perp}}\mathsf{cut}_7$$

- Universal quantifier Intro:

$$\frac{\Psi_{\mathcal{T}}, \Delta \vdash A}{\Psi_{\mathcal{T}}, \Delta \vdash \forall x A}\, x \notin \Gamma$$

$C' \in \psi(\forall x A)$ is of the form $\forall x A'$ with $A' \in \psi(A)$.
Since $\vdash^{\mathcal{P}} C', {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} A', {\Delta'}^{\perp}$ is also safe, and we can apply the induction hypothesis on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} A$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A', {\Delta'}^{\perp}$ to get:

$$\dfrac{\vdash^{\mathcal{P}} A', {\Delta'}^{\perp}}{\vdash^{\mathcal{P}} \forall x A', {\Delta'}^{\perp}}$$

- Universal quantifier Elim:

$$\dfrac{\Psi_{\mathcal{T}}, \Delta \vdash \forall x A}{\Psi_{\mathcal{T}}, \Delta \vdash \{{}^{t}\!/_{x}\}A}$$

$C' \in \psi(\{{}^{t}\!/_{x}\}A)$ is of the form $\{{}^{t}\!/_{x}\}A'$ with $A' \in \psi(A)$ (by Remark 29).

Since $\vdash^{\mathcal{P}} C', {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} (\forall x A'), C', {\Delta'}^{\perp}$ is also safe, and we can apply the induction hypothesis on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} \forall x A$ (with ${C'}^{\perp}, \Delta'$ and $(\forall x A')$) to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} (\forall x A'), C', {\Delta'}^{\perp}$.

We build

$$\begin{array}{cl}
\vdash^{\mathcal{P}} (\forall x A'), \{{}^{t}\!/_{x}\}A', {\Delta'}^{\perp} & \\
\hdashline
\vdash^{\mathcal{P}} A', \{{}^{t}\!/_{x}\}A', {\Delta'}^{\perp} & \text{Lemma 9} \\
\hline
\vdash^{\mathcal{P}} \{{}^{t}\!/_{x}\}A', \{{}^{t}\!/_{x}\}A', {\Delta'}^{\perp} & \text{Lemma 18} \\
\hline
\vdash^{\mathcal{P}} \{{}^{t}\!/_{x}\}A', {\Delta'}^{\perp} & \mathsf{C}_r
\end{array}$$

- Existential quantifier Intro:

$$\dfrac{\Psi_{\mathcal{T}}, \Delta \vdash \{{}^{t}\!/_{x}\}A}{\Psi_{\mathcal{T}}, \Delta \vdash \exists x A}$$

$C' \in \psi(\exists x A)$ is of the form $\exists x A'$ with $A' \in \psi(A)$.

Let $A'_t = \{{}^{t}\!/_{x}\}A'$ ($A'_t \in \psi(\{{}^{t}\!/_{x}\}A)$ by Remark 29).

Since $\vdash^{\mathcal{P}} C', {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} A'_t, {\Delta'}^{\perp}$ is also safe, and we can apply the induction hypothesis on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} \{{}^{t}\!/_{x}\}A$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} A'_t, {\Delta'}^{\perp}$.

By Lemma 27 it suffices to prove $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} \exists x A', {A'_t}^{\perp}$ in order to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} C', {\Delta'}^{\perp}$:

$$\begin{array}{cl}
\vdash^{\mathcal{P}} [A'_t]{A'_t}^{\perp} & \\
\hline
\vdash^{\mathcal{P}} [\exists x A']{A'_t}^{\perp} & \\
\hline
\forall x {A'}^{\perp} \vdash^{\mathcal{P}} [\bullet]{A'_t}^{\perp} & \\
\hline
\vdash^{\mathcal{P}} \exists x A', {A'_t}^{\perp} & \text{Lemma 25(2)}
\end{array}$$

We can complete the proof by applying Lemma 22.

- Existential quantifier Elim:

$$\dfrac{\Psi_{\mathcal{T}}, \Delta \vdash \exists x A \qquad \Gamma, \Delta, A \vdash B}{\Psi_{\mathcal{T}}, \Delta \vdash B}\ x \notin \Gamma, B$$

Let $C' = \exists x A'$ with $A' \in \psi(A)$.

Since $\vdash^{\mathcal{P}} B', {\Delta'}^{\perp}$ is assumed to be safe, $\vdash^{\mathcal{P}} B', C', {\Delta'}^{\perp}$ and $\vdash^{\mathcal{P}} B', {A'}^{\perp}, {\Delta'}^{\perp}$ are also safe, and we can apply the induction hypothesis

– on $\Psi_{\mathcal{T}}, \Delta \vdash_{\mathsf{FOL}} \exists x A$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} B', C', {\Delta'}^{\perp}$;

– on $\Gamma, \Delta, A \vdash_{\mathsf{FOL}} B$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} B', {A'}^{\perp}, {\Delta'}^{\perp}$.

We build

$$\dfrac{\vdash^{\mathcal{P}} C', B', {\Delta'}^{\perp} \qquad \begin{array}{c}
\vdash^{\mathcal{P}} {A'}^{\perp}, B', {\Delta'}^{\perp} \\
\hline
\vdash^{\mathcal{P}} \forall x({A'}^{\perp}), B', {\Delta'}^{\perp} \\
\hdashline
\vdash^{\mathcal{P}} {C'}^{\perp}, B', {\Delta'}^{\perp}
\end{array}}{\vdash^{\mathcal{P}} B', {\Delta'}^{\perp}}\ \mathsf{cut}_7$$

- Negation Intro:

$$\dfrac{\Psi_{\mathcal{T}}, \Delta, A \vdash B \wedge \neg B}{\Psi_{\mathcal{T}}, \Delta \vdash \neg A}$$

If $C' \in \psi(\neg A)$ then ${C'}^{\perp} \in \psi(A)$. Let $D' = D'_1 \wedge^- D'_2$ with $D'_1 \in \psi(B)$ and $D'_2 \in \psi(\neg B)$. Therefore ${D'_2}^{\perp} \in \psi(B)$, $D' \in \psi(B \wedge \neg B)$ and $\Delta', {C'}^{\perp} \in \psi(\Delta, A)$.

Since $\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C'$ is assumed to be safe, $\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C', D'$ is also safe, and we can apply the induction hypothesis on $\Psi_{\mathcal{T}}, \Delta, A \vdash_{\mathsf{FOL}} B \wedge \neg B$ to get $\vdash^{\mathcal{P}}_{\mathsf{LK}^p(\mathcal{T})} {\Delta'}^{\perp}, C', D'$. We build

$$\dfrac{\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C', D' \qquad \dfrac{\dfrac{\dfrac{}{\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C', {D'_1}^{\perp}, {D'_2}^{\perp}}\;\text{Lemma 30}}{\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C', {D'_1}^{\perp} \vee^- {D'_2}^{\perp}}}{\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C', {D'}^{\perp}}\;\text{Corollary 28(4)}}{\vdash^{\mathcal{P}} {\Delta'}^{\perp}, C'}\;\mathsf{cut}_7$$

- Negation Elimination:

$$\dfrac{\Psi_{\mathcal{T}}, \Delta \vdash \neg\neg A}{\Psi_{\mathcal{T}}, \Delta \vdash A}$$

$A' \in \psi(A)$ is such that $A' \in \psi(\neg\neg A)$.

The induction hypothesis on $\Psi_{\mathcal{T}}, \Delta \vdash \neg\neg A$ gives $\vdash^{\mathcal{P}} {\Delta'}^{\perp}, A'$ and we are done.

□

Chapter 4

Simulating SMT-solving in the sequent calculus

An important area of automated reasoning is about satisfiability problems and how to solve them.

The most basic satisfiability problem is propositional/Boolean satisfiability *(SAT)*, where the goal is to decide whether a condition about Boolean variables (e.g. a formula over Boolean variables formed with logical connectives), can be made true by choosing true/false values for its variables. The main techniques for solving propositional SAT-problems are based on the *DPLL procedure* (for Davis-Putnam-Logemann-Loveland) [DP60, DLL62], cutting the exploration of the exponential number of possible truth assignments in a complete way.

Satisfiability Modulo Theories (*SMT*) generalises Boolean SAT, replacing Boolean variables by the (ground) literals of a given theory for which we have a procedure deciding whether a conjunction of literals is consistent. Examples of such theories are Linear Rational Arithmetic, Linear Integer Arithmetic, the theory of arrays, etc.

Solving SMT problems can be done by combining a SAT-solver, or the techniques that it implements, with the decision procedure. Correspondingly, the DPLL procedure can be generalised into a procedure called *DPLL($\mathcal{T}$)* that integrates the decision procedure for such a theory $\mathcal{T}$ and solves SMT problems for $\mathcal{T}$. SMT is a very active area of research, and a particularly interesting aspect is the combination of many theories, so as to treat hybrid SMT-problems.

In this chapter we describe how DPLL($\mathcal{T}$) can be seen as a proof-search procedure in the sequent calculus $\mathsf{LK}^p(\mathcal{T})$. This chapter is organised as follows: Section 4.1 presents variants of the DPLL and DPLL($\mathcal{T}$) procedures. Section 4.2 gives the preliminaries for the simulation of DPLL($\mathcal{T}$) into sequent calculus, including the exact version of $\mathsf{LK}^p(\mathcal{T})$ that we use for the simulation. Section 4.3 shows an indirect simulation of DPLL($\mathcal{T}$) in $\mathsf{LK}^p(\mathcal{T})$ and Section 4.4 presents a direct simulation of DPLL($\mathcal{T}$) in $\mathsf{LK}^p(\mathcal{T})$.

4.1 Variations on Davis-Putnam-Logemann-Loveland

In this chapter, we consider SMT-problems expressed as sets of clauses, as it is traditionally the case in most presentations of SMT-solving techniques. In particular, we do not consider a more general framework where problems are not in clausal form, or that features definition mechanisms for literals.

Definition 24 (Literals) Let $\mathcal{L}$ be a set of elements called *literals*, equipped with an involutive function called *negation* from $\mathcal{L}$ to $\mathcal{L}$. In the rest of this section, a possibly primed or indexed lowercase l always denotes a literal, and $l^\perp$ denotes its negation. ※

DEFINITION 25 (Clause) A *clause* is a finite set of literals, which can be seen as their disjunction.
In the rest of the paper, a possibly indexed upper cased C always denotes a clause. The empty clause is denoted by $\bot$ *empty clause*. The number of literals in a clause C is denoted $\sharp(C)$. The possibly indexed symbol ϕ always denotes a finite sets of clauses $\{C_1, \dots, C_n\}$. We use $\sharp(\phi)$ to denote the maximum of the sizes of the clauses in ϕ. Finally $\mathsf{lit}(\phi)$ denotes the *set of literals* that appear in ϕ or whose negations appear in ϕ. ※

NOTATION 26 Viewing clauses as disjunctions of literals and ϕ as sets of such disjunctions, we use the notations $\phi \models C$ and $\phi \models \neg C$, using the standard notion of entailment of propositional logic. ※

DEFINITION 27 (Decision literals and model sequences)
We consider a copy $\mathcal{L}^d$ of the set $\mathcal{L}$ of literals, whose elements are called *decision literals*, i.e. a tagged version of the literals in $\mathcal{L}$. Decision literals are denoted by l^d.
We use the possibly indexed symbol Δ to denote *model sequences*, i.e. finite sequences of possibly tagged literals, as formally defined by the following grammar:

$$\Delta ::= \ \emptyset \mid \Delta, l^d \mid \Delta, l$$

where l ranges over literals, and l^d ranges over decision literals.
For such a sequence Δ, we write $\overline{\Delta}$ for the subset of $\mathcal{L}$ containing all the literals in Δ with their potential tags removed.
The sequences that $\mathsf{DPLL}(\mathcal{T})$ will construct will always be duplicate-free, so the difference between Δ and $\overline{\Delta}$ is just a matter of tags and ordering. When the context is unambiguous, we will sometimes use Δ when we mean $\overline{\Delta}$.
The sequences that $\mathsf{DPLL}(\mathcal{T})$ will construct will never contain a literal l and its negation $l^\perp$, so the set $\overline{\Delta}$ can also be seen as a (partial) truth assignment that maps literals to truth values (a literal is mapped to true if it belong to $\overline{\Delta}$, or to false if its negation does). ※

4.1.1 DPLL

The DPLL procedure can be presented in different ways. A popular way is to present it in terms of a state transition system, as is done for instance in [NOT05]. We follow this style.

DEFINITION 28 (DPLL states) A *DPLL state* is either the state UNSAT or a pair of the form $\Delta \| \phi$, where ϕ is a set of clauses and Δ is model sequence. ※

Now, we present and discuss the Classical DPLL procedure.

DEFINITION 29 (Classical DPLL)
Classical DPLL is the following transition system between DPLL states, where a *transition* from state S to state S' is denoted by $S \Rightarrow S'$:

- Pure literal:
  $\Delta \| \phi \Rightarrow \Delta, l \| \phi$ where $l \notin \Delta$, $l^\perp \notin \Delta$, $l \in \phi$ and $l^\perp \notin \phi$.
- Unit propagate:
  $\Delta \| \phi, C \vee l \Rightarrow \Delta, l \| \phi, C \vee l$ where $\Delta \models \neg C$, $l \notin \Delta$, $l^\perp \notin \Delta$.
- Decide:
  $\Delta \| \phi \Rightarrow \Delta, l^d \| \phi$ where $l \notin \Delta$, $l^\perp \notin \Delta$, and $l \in \mathsf{lit}(\phi)$.
- Fail:
  $\Delta \| \phi, C \Rightarrow$ UNSAT, with $\Delta \models \neg C$ and there is no decision literal in Δ.
- Backtrack:
  $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l^\perp \| \phi, C$ if $\Delta_1, l, \Delta_2 \models \neg C$ and no decision literal is in Δ_2.

※

Each rule of the Classical DPLL procedure is described below:

- *Pure literal*: If a literal l is pure in ϕ, i.e. there is no occurrence of $l^\perp$ in ϕ, and if l is undefined in Δ then we extend Δ with l.

- *Unit propagate*: If a clause of ϕ contains a literal l, whose truth value is not defined by the model sequence Δ and all the remaining literals of the clause are false, then we extend Δ with l. Indeed, l being true is the only way to make the clause true.
- *Decide*: This rule represents a case analysis on literal l. An undefined literal l is chosen from ϕ, and added to Δ. If later we realise that Δ, l cannot extend to a model sequence of ϕ then the alternative extension $\Delta, l^\perp$ should be considered. Therefore the literal is annotated as a *decision* literal, denoted by l^d.
- *Fail*: This rule detects a *conflicting clause* C. A clause is conflicting in a state $\Delta \| \phi, C$ if $\Delta \models \neg C$. If there is a conflicting clause C and Δ contains no decision literals, then we know that there is no model of ϕ, C and we produce the UNSAT state.
- *Backtrack*: If a conflicting clause C is detected and *Fail* does not apply, then there remains at least one possibility to be explored: backtracking by one decision level to the latest decision literal l^d, which we replace by $l^\perp$, removing any subsequent literals in the current model sequence.

Using the transition system of the DPLL procedure, one can decide the satisfiability of an input set of clauses ϕ by generating a derivation:

$$\emptyset \| \phi \Rightarrow \ldots \Rightarrow F_n$$

F_n is a *final state* when no more transition rule applies to it. If F_n is the form of UNSAT, ϕ is unsatisfiable; if F_n is the form of $\Delta \| \phi$ then ϕ is satisfiable and Δ is a model for it.

The following examples illustrate Classical DPLL procedure:

Example 4

	C_1	C_2	C_3	
$\emptyset$	$1^\perp \vee 2^\perp$	$1^\perp \vee 2$	$1 \vee 2^\perp$	Decide (C_2)
1^d	$1^\perp \vee 2^\perp$	$1^\perp \vee 2$	$1 \vee 2^\perp$	Unit propagate (C_2)
$1^d\ 2$	$1^\perp \vee 2^\perp$	$1^\perp \vee 2$	$1 \vee 2^\perp$	Backtrack
$1^\perp$	$1^\perp \vee 2^\perp$	$1^\perp \vee 2$	$1 \vee 2^\perp$	Unit propagate (C_3)
$1^\perp\ 2^\perp$	$1^\perp \vee 2^\perp$	$1^\perp \vee 2$	$1 \vee 2^\perp$	

In Example 4, ϕ is a set of clauses C_1, C_2 and C_3. When Classical DPLL runs on ϕ, the Decide rule applies on clause C_2, selecting an undefined literal 1^d to place in Δ. The Unit propagate rule applies on clause C_2. Since literal $1^\perp$ in C_2 is false in the model sequence Δ, Δ is extended with literal 2 to make the clause (C_2) true. Now, clause C_1 is in conflict with Δ. Since Δ contains a decision literal and there is also a conflict clause, the Backtrack rule is applied, changing 1^d to $1^\perp$ and removing all the consequences of 1^d from Δ. The Unit propagate rule is applied on C_3 and Δ is extended with literal $2^\perp$. Since no more transition rules apply and the final state is the form of $\Delta \| \phi$, Δ is a model of ϕ.

Example 5 gives a slightly more sophisticated illustration of the transition rules of DPLL:

Example 5

	C_1	C_2	C_3	C_4	C_5	
$\emptyset$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Decide (C_2)
9^d	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Unit propagate (C_2)
$9^d\ 7^\perp$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Unit propagate (C_3)
$9^d\ 7^\perp\ 3$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Unit propagate (C_1)
$9^d\ 7^\perp\ 3\ 5$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Backtrack
$9^\perp$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Unit propagate (C_4)
$9^\perp\ 5$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Decide (C_3)
$9^\perp\ 5\ 3^{d\perp}$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	Unit propagate (C_4)
$9^\perp\ 5\ 3^{d\perp}\ 7$	$9^\perp \vee 3^\perp \vee 5$	$9^\perp \vee 7^\perp$	$7 \vee 3$	$9 \vee 5$	$7 \vee 3^\perp \vee 5^\perp$	

Classical DPLL can be extended with extra rules to improve efficiency: *Backjump*, *Forget*, *Learn* and *Restart*.

Definition 30 (Extra rules for DPLL)

- Backjump:
  $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l_{bj} \| \phi, C$ if, for some clause C_0 such that $C_0 \subseteq \mathsf{lit}(\phi, C)$,
  1. $\Delta_1, l^d, \Delta_2 \models \neg C$
  2. $\Delta_1 \models \neg C_0$
  3. $\phi, C \models C_0 \vee l_{bj}$
  4. $l_{bj} \notin \Delta_1$, $l_{bj}^{\perp} \notin \Delta_1$ and $l_{bj} \in \mathsf{lit}(\phi, \Delta_1, l^d, \Delta_2)$.
- Learn:
  $\Delta \| \phi \Rightarrow \Delta \| \phi, C$ if $C \subseteq \mathsf{lit}(\phi)$ and $\phi \models C$.
- Forget:
  $\Delta \| \phi, C \Rightarrow \Delta \| \phi$ if $\phi \models C$.
- Restart:
  $\Delta \| \phi \Rightarrow \emptyset \| \phi$

※

The Backjump rule allows to backtrack by "more than one level", i.e. to an earlier decision literal that the latest. In this rule, $C_0 \vee l_{bj}$ is called the *backjump clause*. Backjump clauses can be added to the clause set as *learned clauses* (a.k.a *lemmas*), using rule Learn.

The Forget rule is used to remove any redundant clause from ϕ, if we realise that the size of the clause set starts incurring a computational cost that outweighs the benefits of explicitly having redundant clauses in the set.

The Restart rule can be used for instance if we think that the number of decision literals in Δ is so high that we would reach a quicker conclusion by restarting from the empty model but drawing benefits from the clauses we have learned so far. It is mentioned in [NOT06] that the combination of Learn and Restart improves efficiency, both in theory and in practise, of the DPLL procedure.

Finally, the way backjump clauses are computed and then learned from a detected conflict gives rise to an area of SAT-solving called *conflict-driven clause learning* (CDCL) [SS99, JS97]. This traditional presentation of DPLL keeps the production of backjump clauses and learned clauses implicit, i.e. given by an oracle, and we will stick to this style so that any specific oracle can fit seamlessly in our simulations.

4.1.2 DPLL-modulo-theories

DPLL-modulo-theories DPLL($\mathcal{T}$) extends DPLL procedure with a background theory $\mathcal{T}$ to solve SMT-problems: DPLL($\mathcal{T}$) aims at proving the satisfiability or inconsistency of a set of clauses with respect to the theory $\mathcal{T}$. For this we instantiate the abstract notion of literal from Definition 24 with the concrete notion from first-order logic, i.e. that of Definition 10:

Definition 31 (Literals) The definition of literals is as in Definition 10 in Chapter 3. ※

Notation 32 Viewing clauses as disjunctions of literals and ϕ as sets of such disjunctions, we use the notations $\phi \models_{\mathcal{T}} C$ and $\phi \models_{\mathcal{T}} \neg C$ as a generalisation/variant of Notation 21 in Chapter 3. ※

Definition 33 (Closure) We define $\mathsf{Clo}(\Delta) := \{l \mid \Delta, l^{\perp} \models_{\mathcal{T}}\}$, the *closure* of a model sequence Δ by "semantical entailment". For any set of clauses ϕ, the set of literals occurring in ϕ that are semantically entailed by Δ is denoted by $\mathsf{Clo}_{\phi}(\Delta) := \mathsf{Clo}(\Delta) \cap \mathsf{lit}(\phi)$. ※

Remark 33 Obviously, if $l \in \Delta$, then $l \in \mathsf{Clo}(\Delta)$.
If $\phi_1 \subseteq \phi_2$, then for any Δ, $\mathsf{Clo}_{\phi_1}(\Delta) \subseteq \mathsf{Clo}_{\phi_2}(\Delta)$.

Definition 34 (Abstract DPLL modulo theories: DPLL($\mathcal{T}$))
The *Abstract DPLL($\mathcal{T}$)* system is given by the following transitions rules:

- Unit propagate:
  $\Delta \| \phi, C \vee l \Rightarrow \Delta, l \| \phi, C \vee l$ where $\Delta \models \neg C$, $l \notin \Delta$, $l^{\perp} \notin \Delta$.
- Decide:
  $\Delta \| \phi \Rightarrow \Delta, l^d \| \phi$ where $l \notin \Delta$, $l^{\perp} \notin \Delta$, and $l \in \mathsf{lit}(\phi)$.
- Fail:
  $\Delta \| \phi, C \Rightarrow$ UNSAT, with $\Delta \models \neg C$ and there is no decision literal in Δ.
- Theory propagate:
  $\Delta \| \phi \Rightarrow \Delta, l \| \phi$ where $l \in \mathsf{Clo}_\phi(\Delta)$ and $l \notin \Delta, l^{\perp} \notin \Delta$.
- $\mathcal{T}$-Backjump: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l_{bj} \| \phi, C$ if, for some clause C_0 such that $C_0 \subseteq \mathsf{lit}(\phi, C)$,
  1. $\Delta_1, l^d, \Delta_2 \models \neg C$
  2. $\Delta_1 \models \neg C_0$
  3. $\phi, C \models_{\mathcal{T}} C_0 \vee l_{bj}$
  4. $l_{bj} \notin \Delta_1$, $l_{bj}^{\perp} \notin \Delta_1$ and $l_{bj} \in \mathsf{lit}(\phi, \Delta_1, l^d, \Delta_2)$.
- $\mathcal{T}$-Learn:
  $\Delta \| \phi \Rightarrow \Delta \| \phi, C$ if $C \subseteq \mathsf{lit}(\phi)$ and $\phi \models_{\mathcal{T}} C$.
- $\mathcal{T}$-Forget:
  $\Delta \| \phi, C \Rightarrow \Delta \| \phi$ if $\phi \models_{\mathcal{T}} C$.
- Restart:
  $\Delta \| \phi \Rightarrow \emptyset \| \phi$

Sometimes, to distinguish this Abstract DPLL($\mathcal{T}$) system from other versions of DPLL($\mathcal{T}$) (e.g. that of the next section), we will denote it DPLL$_{bj}$($\mathcal{T}$). ※

The Theory propagate rule assigns a truth value to literal which is undefined in Δ but entailed by the theory $\mathcal{T}$. Compared to DPLL, Backjump, Learn and Forget rules are modified to accommodate the background theory $\mathcal{T}$, and are now called *$\mathcal{T}$-Backjump*, *$\mathcal{T}$-Learn* and *$\mathcal{T}$-Forget*.

4.1.3 The Elementary DPLL($\mathcal{T}$) system

In this section we introduce a system called *Elementary DPLL($\mathcal{T}$)*, which is both a (logically complete) restriction of Abstract DPLL($\mathcal{T}$) as well as being a straightforward generalisation of Classical DPLL to the presence of a background theory $\mathcal{T}$.

We will use this Elementary DPLL($\mathcal{T}$) system for the simulation of DPLL($\mathcal{T}$) in sequent calculus in the rest of this chapter. We will also simulate the whole Abstract DPLL($\mathcal{T}$) system, but the simulation of Elementary DPLL will be tighter, with nicer properties (e.g. a bisimulation result) than the simulation of the most general system.

We now describe the Elementary DPLL($\mathcal{T}$) procedure as a transition system between states.

Definition 35 (Elementary DPLL($\mathcal{T}$)) The transition rules of the Elementary DPLL($\mathcal{T}$) procedure are defined in Figure 4.1. ※

This transition system is an extension of the Classical DPLL procedure, as presented in [NOT06], to the background theory $\mathcal{T}$. We removed the Pure literal rule, in general unsound in presence of a theory $\mathcal{T}$. The first four rules are explicitly taken from the abstract DPLL($\mathcal{T}$) system of [NOT06]. Unit propagate and *Theory propagate* are renamed as Propagate and Propagate$_{\mathcal{T}}$ for consistency with the other rule names. The other rules of that system (namely $\mathcal{T}$-Backjump, $\mathcal{T}$-Learn, $\mathcal{T}$-Forget, etc), are not considered here in their full generality, but specific cases and combinations are covered by the rest of our elementary DPLL($\mathcal{T}$) system, so that it is logically complete. Backtrack is a restricted version of $\mathcal{T}$-Backjump (this holds on the basis that the full system satisfies some basic invariant -Lemma 3.6 of [NOT06]). The last two new rules *Fail*$_{\mathcal{T}}$ and Backtrack$_{\mathcal{T}}$ are newly added in this system; checking the semantical consistency of the model sequence Δ. Fail$_{\mathcal{T}}$ (resp. Backtrack$_{\mathcal{T}}$) is a combination of $\mathcal{T}$-Learn, Fail (resp. Backtrack), and $\mathcal{T}$-Forget steps.

- Decide:
  $\Delta\|\phi \Rightarrow \Delta, l^d\|\phi$ where $l \notin \Delta$, $l^\perp \notin \Delta$, $l \in \mathsf{lit}(\phi)$.
- Propagate:
  $\Delta\|\phi, C \vee l \Rightarrow \Delta, l\|\phi, C \vee l$ where $\Delta \models \neg C$, $l \notin \Delta$, $l^\perp \notin \Delta$.
- Fail:
  $\Delta\|\phi, C \Rightarrow$ UNSAT, with $\Delta \models \neg C$ and there is no decision literal in Δ.
- Backtrack:
  $\Delta_1, l^d, \Delta_2\|\phi, C \Rightarrow \Delta_1, l^\perp\|\phi, C$ if $\Delta_1, l, \Delta_2 \models \neg C$ and there is no decision literal in Δ_2.
- Propagate$_\mathcal{T}$:
  $\Delta\|\phi \Rightarrow \Delta, l\|\phi$ where $l \in \mathsf{Clo}_\phi(\Delta)$ and $l \notin \Delta, l^\perp \notin \Delta$.
- Fail$_\mathcal{T}$:
  $\Delta\|\phi \Rightarrow$ UNSAT, with $\Delta \models_\mathcal{T}$ and there is no decision literal in Δ.
- Backtrack$_\mathcal{T}$:
  $\Delta_1, l^d, \Delta_2\|\phi \Rightarrow \Delta_1, l^\perp\|\phi$ if $\Delta_1, l, \Delta_2 \models_\mathcal{T}$ and there is no decision literal in Δ_2.

Figure 4.1: Elementary DPLL($\mathcal{T}$)

By Example 6 and Example 7, we illustrate the reason for the introduction of Fail$_\mathcal{T}$ and Backtrack$_\mathcal{T}$ in Elementary DPLL($\mathcal{T}$). For these examples we use the theory of *Linear Rational Arithmetic.*

EXAMPLE 6

	C_1	C_2	C_3	
$\emptyset$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate(C_1)
$x > 0$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate(C_2)
$x > 0, (x + y > 0)^\perp$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate$_\mathcal{T}$
$x > 0, (x + y > 0)^\perp, (y > 0)^\perp$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate$_\mathcal{T}$
$x > 0, (x + y > 0)^\perp, (y > 0)^\perp, (x = -1)^\perp$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Fail
UNSAT				

In this example, while DPLL($\mathcal{T}$) run on ϕ, the Propagate rule first applies on clause C_1 and on second run again this rule is applied on clause C_2; Δ is extended with $x > 0$ and $(x + y > 0)^\perp$, respectively. Propagate$_\mathcal{T}$ applies next two run and Δ is extended with $(y > 0)^\perp \in \mathsf{Clo}_\phi(\Delta)$ and $(x = -1)^\perp \in \mathsf{Clo}_\phi(\Delta)$; these two literals are theory $\mathcal{T}$-entailed by Δ. Now, clause C_3 is in conflict with Δ i.e. $\Delta \models \neg C_3$. Since there is no decision literal in Δ and Δ is conflicted with C_3, the Fail rule is applied. The final state of the DPLL($\mathcal{T}$) run is UNSATand therefore ϕ is $\mathcal{T}$-unsatisfiable.

Now, if we change our choice of selecting rules on the last Propagate$_\mathcal{T}$ rule with Propagate; this has created a $\mathcal{T}$-inconsistency from which we could not derive UNSAT without a Fail$_\mathcal{T}$ step (or, alternatively, a $\mathcal{T}$-Learn step in [NOT06]). Therefore a reason to introduce rule Fail$_\mathcal{T}$ is to allow the second run to finish with the same output as the first. This is illustrated in Example 7:

EXAMPLE 7

	C_1	C_2	C_3	
$\emptyset$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate(C_1)
$x > 0$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate(C_2)
$x > 0, (x + y > 0)^\perp$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate$_\mathcal{T}$
$x > 0, (x + y > 0)^\perp, (y > 0)^\perp$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Propagate(C_3)
$x > 0, (x + y > 0)^\perp, (y > 0)^\perp, (x = -1)$	$x > 0$	$(x + y > 0)^\perp$	$(y > 0 \vee x = -1)$	Fail$_\mathcal{T}$

The two runs above should illustrate the fact that none of the variations on DPLL that we presented are deterministic transition systems: for instance the Decide rule can be applied from any state if there is still a literal eligible for picking, and it furthermore does not enforce a strategy for picking that literal. At the level of implementation, this (non deterministic) transition system is turned into a deterministic algorithm, whose efficiency crucially relies on the strategies adopted to perform the choices left unspecified by $\mathsf{DPLL}(\mathcal{T})$.

4.2 Preliminaries

In this section, we give the preliminaries of our simulation of the $\mathsf{DPLL}(\mathcal{T})$ procedure into focused sequent calculus. In Chapter 3, we presented a new system, $\mathsf{LK}^p(\mathcal{T})$, and the motivation for it is to perform proof-search modulo theories and in particular simulate the $\mathsf{DPLL}(\mathcal{T})$ techniques.

Definition 36 (System used to simulate $\mathsf{DPLL}(\mathcal{T})$)
The system that we use for our simulations is presented in Fig. 4.2, and we assume additionally that its sequents are all safe. ※

Synchronous rules

$$(\wedge^+)\frac{\Gamma \vdash^{\mathcal{P}} [A] \qquad \Gamma \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]} \qquad (\vee^+)\frac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]}$$

$$(\top^+)\frac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]} \qquad (\mathsf{Init}_1)\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^\perp \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [l]}\ l \text{ is } \mathcal{P}\text{-positive} \qquad (\mathsf{Release})\frac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]}\ N \text{ is not } \mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \qquad \Gamma \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash^{\mathcal{P}} A \wedge^- B, \Delta} \qquad (\vee^-)\frac{\Gamma \vdash^{\mathcal{P}} A_1, A_2, \Delta}{\Gamma \vdash^{\mathcal{P}} A_1 \vee^- A_2, \Delta}$$

$$(\bot^-)\frac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta, \bot^-} \qquad (\top^-)\frac{}{\Gamma \vdash^{\mathcal{P}} \Delta, \top^-} \qquad (\mathsf{Store})\frac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}\ \begin{array}{l} A \text{ is a literal} \\ \text{or is } \mathcal{P}\text{-positive}\end{array}$$

Structural rules

$$(\mathsf{Select})\frac{\Gamma, P^\perp \vdash^{\mathcal{P}} [P]}{\Gamma, P^\perp \vdash^{\mathcal{P}}}\ P \text{ is not } \mathcal{P}\text{-negative} \qquad (\mathsf{Init}_2)\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

Admissible/Invertible rules

$$(\mathsf{Pol})\frac{\Gamma \vdash^{\mathcal{P},l}}{\Gamma \vdash^{\mathcal{P}}}\ l \in \Gamma \text{ and } \mathsf{lit}_{\mathcal{P}}(\Gamma), l^\perp \models_{\mathcal{T}} \qquad (\mathsf{cut}_7)\frac{\Gamma \vdash^{\mathcal{P}} l \quad \Gamma \vdash^{\mathcal{P}} l^\perp}{\Gamma \vdash^{\mathcal{P}}}\ l \in \Gamma$$

where $\mathcal{P}; A := \mathcal{P}, A$ if $A \in \mathsf{U}_{\mathcal{P}}$
$\mathcal{P}; A := \mathcal{P}$ if not

Figure 4.2: System for the simulation of $\mathsf{DPLL}(\mathcal{T})$

This system is the propositional fragment of the $\mathsf{LK}^p(\mathcal{T})$ system from Chapter 3, extended with the rules (Pol) and (cut_7) (or more precisely restricted versions of them) that we have shown to be admissible (and invertible) in $\mathsf{LK}^p(\mathcal{T})$. It is therefore strictly equivalent to propositional $\mathsf{LK}^p(\mathcal{T})$ as far as complete proofs are concerned. But as we will formalise our simulation results in terms of incomplete proofs, having those admissible rules explicitly in the system yields tighter results.

We now give a general idea of the simulation of $\mathsf{DPLL}(\mathcal{T})$ into sequent calculus.

As we know, a complete and successful run of the $\mathsf{DPLL}(\mathcal{T})$ procedure is a sequence of transitions $\emptyset \| \phi \Rightarrow^* \mathsf{UNSAT}$, which ensure that the set of clauses ϕ is inconsistent modulo the theory.

Hence, we are devising a proof-search process aiming at building a proof-tree for sequents of the form $\phi' \vdash$, where ϕ' represents the set of clauses ϕ as a sequent calculus structure.

The assumption, in the sequent calculus that we use for the simulation, that sequents are safe will not be restrictive for our purpose, as we will always start our simulations of $\mathsf{DPLL}(\mathcal{T})$ with the empty polarisation set, making the initial sequent $\phi' \vdash$ safe (Lemma 16.3), and then safety is preserved by the bottom-up application of every rule (Lemma 16.4), i.e. by the process by which we simulate $\mathsf{DPLL}(\mathcal{T})$.

Now, in order to construct a proof of $\phi' \vdash$ in sequent calculus from a run $\emptyset \| \phi \Rightarrow^* \mathsf{UNSAT}$ of $\mathsf{DPLL}(\mathcal{T})$, we proceed incrementally by considering the intermediate steps of the $\mathsf{DPLL}(\mathcal{T})$ run:

$$\emptyset \| \phi \Rightarrow^* \Delta \| \phi \Rightarrow^* \mathsf{UNSAT}$$

In the intermediate $\mathsf{DPLL}(\mathcal{T})$ state $\Delta \| \phi$, the model sequence Δ is a log of both the search space explored so far (in $\emptyset \| \phi \Rightarrow^* \Delta \| \phi$) and the search space that remains to be explored (in $\Delta \| \phi \Rightarrow^* \mathsf{UNSAT}$). In this log, a tagged decision literal l^d indicates a point where the procedure has made an exploratory choice (the case where l is true has been/is being explored, the case where $l^\perp$ is true remains to be explored), while untagged literals in Δ are predictable consequences of the decisions made so far and of the set of clauses ϕ to be falsified.

If we are to express the $\mathsf{DPLL}(\mathcal{T})$ procedure as the *incremental construction* of a proof-tree, we should get from $\emptyset \| \phi \Rightarrow^* \Delta \| \phi$ a proof-tree that is not yet complete and get from $\Delta \| \phi \Rightarrow^* \mathsf{UNSAT}$ some (complete) proof-tree(s) that can be “plugged into the holes” of the incomplete tree. We should read in Δ the "interface" between the incomplete tree that has been constructed and the complete sub-trees to be constructed.

We use the plural here since there can be more than one sub-tree left to construct: $\Delta \| \phi \Rightarrow^* \mathsf{UNSAT}$ contains the information to build not only a proof of $\overline{\Delta}, \phi' \vdash$, but also proofs of the sequents corresponding to the other parts of the search space to be explored, characterised by the tagged literals in Δ. Since the decision tags play such a crucial role in the narrowing of the search space, we need to extract from a $\mathsf{DPLL}(\mathcal{T})$ state $\Delta \| \phi$ all the information that a proof search strategy for sequents has to know about. Intuitively, we need to be more precise on the status of the pending proofs required at the open leaves of a partial proof tree. For instance, a run from $l_1, l_2^d, l_3, l_4^d \| \phi \Rightarrow^* \mathsf{UNSAT}$ contains the information to build a proof of $l_1, l_2, l_3, l_4, \phi' \vdash$ but also the proofs of $l_1, l_2, l_3, l_4^\perp, \phi' \vdash$ and $l_1, l_2^\perp, \phi' \vdash$. Those extra sequents are obtained by collecting from a model sequence Δ its "backtrack points" as follows :

DEFINITION 37 (Backtrack points) The *backtrack points* $[\![\Delta]\!]$ of a model sequence Δ of possibly tagged literals is the list of sets of untagged literals recursively defined by the rules of Fig 4.3, where $[\,]$ and $::$ are the standard list constructors.
We will sometimes use $[\![\Delta]\!]$ as a set, forgetting the order, and will also use the notation $[\Delta]$ for the set $\{\overline{\Delta}\} \cup [\![\Delta]\!]$. ※

$$\boxed{\begin{array}{lcl} [\![()]\!] & := & [\,] \\ [\![\Delta, l]\!] & := & [\![\Delta]\!] \\ [\![\Delta, l^d]\!] & := & \overline{\Delta, l^\perp} :: [\![\Delta]\!] \end{array}}$$

Figure 4.3: Collecting backtrack points

REMARK 34 The length of $[\![\Delta]\!]$ is the number of decision literals in Δ.

REMARK 35 We have $\overline{\Delta} \in [\Delta]$ and $[\![\Delta]\!] \subseteq [\Delta]$.

In the rest of this chapter, we will consider two ways of simulating $\mathsf{DPLL}(\mathcal{T})$ in sequent calculus. There are two main gaps between $\mathsf{DPLL}(\mathcal{T})$ and sequent calculus:

- Firstly, a (successful) $\mathsf{DPLL}(\mathcal{T})$ run is a rewrite sequence finishing with the state UNSAT, while a (successful) proof-search run is (/ produces) a proof-tree; the former uses decision literals and backtracking, the latter uses branching.

- Secondly, the structures handled by automated reasoning techniques such as DPLL($\mathcal{T}$) are very flexible (e.g. clauses are sets or multisets of literals), while sequent calculus implements a root-first decomposition of formulae trees.

Therefore, the first kind of simulation we consider is an indirect one, based on an intermediary inference system introduced in [Tin02] which we will call here $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. This system is intermediate, since it manipulates proof-trees as in the sequent calculus but with the flexible structures of DPLL($\mathcal{T}$). It would be difficult to claim that $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ is a sequent calculus because it does not implement the root-first decomposition of formulae trees, and is quite ad hoc: it is clearly designed to express the concepts of DPLL($\mathcal{T}$) and not much else.

The simulation is thus split into two simulations: one from DPLL($\mathcal{T}$) in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ and one from $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$.

- The former simulation relates two presentations of the DPLL($\mathcal{T}$) concepts taken from the literature, where we found however no formalization of how these two presentations relate (an informal description is given in [Tin02] in English).
- The latter simulation is built on previous work by Ivan Gazeau [Gaz10] for Classical DPLL (no theories involved), which was encoded in LKF; we therefore lifted his results to the presence of a theory (hence the introduction of $\mathsf{LK}^p(\mathcal{T})$) and to the extra rules of $\mathcal{T}$-Backjump, $\mathcal{T}$-Learn, $\mathcal{T}$-Forget and Restart.

The second kind of simulation is direct from DPLL($\mathcal{T}$) to $\mathsf{LK}^p(\mathcal{T})$, and is in fact simpler than composing the two parts of the indirect simulation. Moreover, we get some interesting results using this direct simulation: we identify in $\mathsf{LK}^p(\mathcal{T})$ the image, by the simulation, of the Elementary DPLL($\mathcal{T}$) runs, using a simple criterion on polarities and a bisimulation theorem.

Therefore, the rest of the chapter is structured in the following way:

- in Section 4.3, we discuss the indirect simulation of both Elementary and Abstract DPLL($\mathcal{T}$).
- in Section 4.4, we discuss the direct simulation of both Elementary and Abstract DPLL($\mathcal{T}$) into $\mathsf{LK}^p(\mathcal{T})$ system. Moreover, we also present a reverse simulation of $\mathsf{LK}^p(\mathcal{T})$ system into Elementary DPLL($\mathcal{T}$) procedure by an appropriate management of polarities.

4.3 An indirect simulation of DPLL($\mathcal{T}$)

In this section, we present the indirect simulation of DPLL($\mathcal{T}$) into the sequent calculus $\mathsf{LK}^p(\mathcal{T})$.

First, we simulate the DPLL($\mathcal{T}$) procedure in the intermediate system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. This formalises what is informally described in [Tin02]. We first simulate Elementary DPLL($\mathcal{T}$), and then the full Abstract DPLL($\mathcal{T}$).

Before discussing the simulation of DPLL($\mathcal{T}$), we briefly present the $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ system.

4.3.1 The $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ system

In this section, we present all preliminaries and properties of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

Definition 38 (The system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$) Given a theory $\mathcal{T}$, the system *$LK_{DPLL}(\mathcal{T})$*, given in Figure 4.4, is an inference system on sequents of the form $\Delta;\phi \vdash$, where Δ is a set of literals and ϕ is a set of clauses. ※

The *Assert* rule models the fact that every literal occurring as a unit clause in the current clause set must be satisfied for the whole clause set to be satisfied. The *Split* rule is mainly used to branch the proof tree from the DPLL($\mathcal{T}$) rewrite sequence system. This rule corresponds to the decomposition in smaller subproblems of the DPLL($\mathcal{T}$) method. This rule is the only *don't know non-deterministic* rule of the calculus. The *Resolve* rule removes from a clause all literals whose complement has been asserted (which corresponds to generating the simplified clause by unit resolution and the discarding the clause by backward subsumption). The *Subsume* rule removes from the clauses that contain an asserted literal (because all of these clause will be satisfied in any model in which the asserted literal is true); it is part of the system for convenience as it could be

$$\frac{\Delta, l^{\perp};\phi \vdash \qquad \Delta, l;\phi \vdash}{\Delta;\phi \vdash}\ \textit{Split} \text{ where } l \in \mathsf{lit}(\phi),\ \Delta, l^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

$$\frac{}{\Delta;\phi, \perp \vdash}\ \textit{Empty} \qquad \frac{\Delta, l;\phi, l \vdash}{\Delta;\phi, l \vdash}\ \textit{Assert} \text{ where } \Delta, l^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

$$\frac{\Delta;\phi \vdash}{\Delta;\phi, l \vee C \vdash}\ \textit{Subsume} \text{ where } \Delta, l^{\perp} \models_{\mathcal{T}} \qquad \frac{\Delta;\phi, C \vdash}{\Delta;\phi, l \vee C \vdash}\ \textit{Resolve} \text{ where } \Delta, l \models_{\mathcal{T}}$$

Figure 4.4: System $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

removed with no loss of completeness. To close the branch of a proof tree we use the *Empty* rule. It models the fact that a derivation can be terminated as soon as the empty clause is derived.

Now, we discuss some meta-theories of the $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ system, and present some rules that are size-preserving admissible:

Lemma 36 (Weakening1) The following rule is size-preserving admissible in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

$$\frac{\Delta;\phi \vdash}{\Delta;\phi, C \vdash}$$

※

Proof: By induction on $\Delta;\phi \vdash$. □

Lemma 37 (Weakening2) The following rule is size-preserving admissible in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

$$\frac{\Delta;\phi \vdash}{\Delta';\phi \vdash}\ \mathsf{Clo}_{\phi}(\Delta) \subseteq \mathsf{Clo}_{\phi}(\Delta')$$

※

Proof: By induction on the derivation of $\Delta;\phi \vdash$:

- *Resolve*:

$$\frac{\Delta;\phi, C \vdash}{\Delta;\phi, l \vee C \vdash}\ \Delta, l \models_{\mathcal{T}}$$

We assume $\mathsf{Clo}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$
from which we get $\mathsf{Clo}_{\phi,C}(\Delta) \subseteq \mathsf{Clo}_{\phi,C}(\Delta')$, so we can apply the induction hypothesis to construct

$$\frac{\Delta';\phi, C \vdash}{\Delta';\phi, l \vee C \vdash}\ \Delta', l \models_{\mathcal{T}}$$

The side-condition is a consequence of the assumption $\mathsf{Clo}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$.

- *Subsume*:

$$\frac{\Delta;\phi \vdash}{\Delta;\phi, l \vee C \vdash}\ \Delta, l^{\perp} \models_{\mathcal{T}}$$

We assume $\mathsf{Clo}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$
from which we get $\mathsf{Clo}_{\phi}(\Delta) \subseteq \mathsf{Clo}_{\phi}(\Delta')$, so we can apply the induction hypothesis to construct

$$\frac{\Delta';\phi \vdash}{\Delta';\phi, l \vee C \vdash}\ \Delta', l^{\perp} \models_{\mathcal{T}}$$

The side-condition is a consequence of the assumption $\mathsf{Clo}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$.

- *Assert*:

$$\frac{\Delta, l;\phi, l \vdash}{\Delta;\phi, l \vdash}\ \Delta, l^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

  We assume $\mathsf{Clo}_{\phi,l}(\Delta) \subseteq \mathsf{Clo}_{\phi,l}(\Delta')$
  from which we get $\mathsf{Clo}_{\phi,l}(\Delta, l) \subseteq \mathsf{Clo}_{\phi,l}(\Delta', l)$.
  - If $\Delta' \models_{\mathcal{T}} l$, then $\mathsf{Clo}(\Delta', l) = \mathsf{Clo}(\Delta')$, so we have $\mathsf{Clo}_{\phi,l}(\Delta, l) \subseteq \mathsf{Clo}_{\phi,l}(\Delta')$. The induction hypothesis then gives $\Delta';\phi, l \vdash$.
  - If $\Delta' \models_{\mathcal{T}} l^{\perp}$, then we construct

$$\cfrac{\cfrac{}{\Delta';\phi, \perp \vdash}\ \textit{Empty}}{\Delta';\phi, l \vdash}\ \textit{Resolve}$$

  - If $\Delta' \not\models_{\mathcal{T}} l$ and $\Delta' \not\models_{\mathcal{T}} l^{\perp}$: we first apply the induction hypothesis to get $\Delta', l;\phi, l \vdash$ and we conclude by constructing

$$\frac{\Delta', l;\phi, l \vdash}{\Delta';\phi, l \vdash}\ \Delta', l^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta', l \not\models_{\mathcal{T}}$$

- *Split*:

$$\frac{\Delta, l^{\perp};\phi, l \vee C \vdash \qquad \Delta, l;\phi, l \vee C \vdash}{\Delta;\phi, l \vee C \vdash}\ \Delta, l^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

  We assume $\mathsf{Clo}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$ from which we get both
  $\mathsf{Clo}_{\phi,l\vee C}(\Delta, l) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta', l)$ and $\mathsf{Clo}_{\phi,l\vee C}(\Delta, l^{\perp}) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta', l^{\perp})$.
  - If $\Delta' \models_{\mathcal{T}} l$, then $\mathsf{Clo}(\Delta') = \mathsf{Clo}(\Delta', l)$, so we have $\mathsf{Clo}_{\phi,l\vee C}(\Delta, l) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$. The induction hypothesis then gives $\Delta';\phi, l \vee C \vdash$.
  - If $\Delta' \models_{\mathcal{T}} l^{\perp}$, then $\mathsf{Clo}(\Delta') = \mathsf{Clo}(\Delta', l^{\perp})$, so we have $\mathsf{Clo}_{\phi,l\vee C}(\Delta, l^{\perp}) \subseteq \mathsf{Clo}_{\phi,l\vee C}(\Delta')$. The induction hypothesis then gives $\Delta';\phi, l \vee C \vdash$.
  - If $\Delta' \not\models_{\mathcal{T}} l$ and $\Delta' \not\models_{\mathcal{T}} l^{\perp}$: the induction hypothesis on both premises gives $\Delta', l;\phi, l \vee C \vdash$ and $\Delta', l^{\perp};\phi, l \vee C \vdash$, and we can conclude

$$\frac{\Delta', l^{\perp};\phi, l \vee C \vdash \qquad \Delta', l;\phi, l \vee C \vdash}{\Delta';\phi, l \vee C \vdash}\ \Delta' \not\models_{\mathcal{T}} l \text{ and } \Delta' \not\models_{\mathcal{T}} l^{\perp}$$

- *Empty*: Straightforward.

□

Lemma 38 (Invertibility of Resolve) *Resolve* is size-preserving invertible in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. ※

Proof: By induction on the derivation of $\Delta;\phi, C \vee l \vdash$ we prove $\Delta;\phi, C \vdash$ (with the assumption $\Delta, l \models_{\mathcal{T}}$):

- *Resolve*:
  easily permutes with other instances of *Resolve* and with instances of *Subsume*.
- *Assert*:
  The side-condition of the rule guarantees that the literal added to the model, say l', is different from l:

$$\frac{\Delta, l';\phi', l', C \vee l \vdash}{\Delta;\phi', l', C \vee l \vdash}\ \Delta, l'^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$

  We can construct:

$$\frac{\Delta, l';\phi', l', C \vdash}{\Delta;\phi', l', C \vdash}\ \Delta, l'^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$

  whose premiss is proved by the induction hypothesis.

- *Split*:
$$\frac{\Delta, {l'}^{\perp};\phi, C \vee l \vdash \quad \Delta, l';\phi, C \vee l \vdash}{\Delta;\phi, C \vee l \vdash}\ l' \in \mathsf{lit}(\phi, C \vee l) \text{ and } \Delta, {l'}^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$
We can construct:
$$\frac{\Delta, l';\phi, C \vdash \quad \Delta, {l'}^{\perp};\phi, C \vdash}{\Delta;\phi, C \vdash}\ l' \in \mathsf{lit}(\phi, C) \text{ and } \Delta, {l'}^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$
whose branches are closed by using the induction hypothesis. The side-condition $l' \in \mathsf{lit}(\phi, C)$ is satisfied because $l \neq l'$.
- *Empty*: Straightforward.

□

Definition 39 (Extension of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$: $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$) We introduce a new system $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ which is the extention of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ with *Weakening*, *Weakening2* and the *Inverted Resolve*, as shown in Fig. 4.5. ※

$$\frac{\Delta;\phi \vdash}{\Delta;\phi, C \vdash} \qquad \frac{\Delta;\phi \vdash}{\Delta';\phi \vdash}\ \mathsf{Clo}_{\phi}(\Delta) \subseteq \mathsf{Clo}_{\phi}(\Delta') \qquad \frac{\Delta;\phi, l \vee C \vdash}{\Delta;\phi, C \vdash}\ \Delta, l \models_{\mathcal{T}}$$

Figure 4.5: Extension of system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

By the previous lemmas, system $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ is strictly equivalent to $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ as far as complete proofs are concerned. But again (like we extended $\mathsf{LK}^p(\mathcal{T})$), as we will formalise our simulation results in terms of incomplete proofs, having those admissible rules explicitly in the system yields tighter results.

Definition 40 (Size of proof-trees in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$) The size of proof-trees in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ is defined as the size of trees in the usual sense, but not counting the occurrences of *Weakening* or the *Inverted Resolve* rules. For that reason, dashed lines will be used for the occurrences of those inference rules. ※

Remark 39 The size-preserving admissibility results of those three rules in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ entails that a proof-tree in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ of size n, can be transformed into a proof-tree in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ of size at most n.

Lemma 40 If $\Delta \models_{\mathcal{T}} \neg C$ then there is a proof-tree concluding $\Delta;C, \phi \vdash$ of size at most $\sharp(\phi) + 1$. ※

Proof: Here $\Delta \models_{\mathcal{T}} \neg C$ means $C = l_1 \vee \ldots \vee l_n$ and for all l_i, $\forall l_i\ \Delta \models_{\mathcal{T}} l_i^{\perp}$ where i=1,. . . ,n. We can therefore construct
$$\cfrac{\cfrac{}{\Delta;\bot, \phi \vdash}\ \text{Empty}}{\Delta;C, \phi \vdash}\ \text{Resolve}$$

□

We have discussed the preliminaries of the indirect simulation of $\mathsf{DPLL}(\mathcal{T})$. Now we present and show the simulation of Elementary and Abstract $\mathsf{DPLL}(\mathcal{T})$. This simulation is decomposed into two simulations: (i) Section 4.3.2 shows the simulation of $\mathsf{DPLL}(\mathcal{T})$ into $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ (or more precisely, $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$) and (ii) Section 4.3.3 shows the simulation of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ into $\mathsf{LK}^p(\mathcal{T})$.

Therefore, starting from a full run of $\mathsf{DPLL}(\mathcal{T})$ finishing on UNSAT, the first simulation incrementally builds a complete proof-tree in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$, which we can then transform (using our size-preserving admissibility results) into a complete tree in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ of smaller or equal size. The second simulation will then describe the incremental construction of that (complete) proof-tree as the incremental construction of a (complete) proof-tree of $\mathsf{LK}^p(\mathcal{T})$.

4.3.2 Simulation of DPLL($\mathcal{T}$) in LK$_{\mathsf{DPLL}}$($\mathcal{T}$)

We proceed with the encoding of the Elementary $\mathsf{DPLL}(\mathcal{T})$ procedure as the construction of a derivation tree in system $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$. We have already given the basic idea of the simulation of $\mathsf{DPLL}(\mathcal{T})$ into seqeunt calculus in Section 4.2: if $\Delta\|\phi \Rightarrow^* \mathsf{UNSAT}$ then there is a $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ proof of $\overline{\Delta};\phi \vdash$ (i.e. there is no $\mathcal{T}$-model of ϕ extending Δ).

Now, coming back to the $\mathsf{DPLL}(\mathcal{T})$ transition sequence $\emptyset \Rightarrow^* \Delta\|\phi$ and its intuitive counterpart in an inference system, we have to formalise the notion of *incomplete* proof-tree together with the notion of "filing holes".

DEFINITION 41 (Incomplete proof tree, extension of incomplete proof-tree) An incomplete proof-tree in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ is a tree labelled with sequents, whose leaves are tagged as either *open* or *closed*, and such that every node that is not an open leaf is an instance of the $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ rules.
An incomplete proof-tree that has no open leaf is isomorphic to a derivation in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$.
A complete proof-tree is (isomorphic to) an incomplete proof-tree whose leaves are all closed.
An incomplete proof-tree π' is an *n-extension* of π if π' is π or if π' is obtained from π by replacing one of its open leaves by an incomplete proof-tree of size at most n and whose conclusion has the same label as that leaf. ※

DEFINITION 42 (Correspondence between DPLL($\mathcal{T}$) states and incomplete proof-trees) An incomplete proof-tree π *corresponds to* a $\mathsf{DPLL}(\mathcal{T})$ state $\Delta\|\phi$ if the sequents labelling its open leaves are all distinct and form a sub-set of $\{\Delta';\phi \vdash \ \mid \Delta' \in [\Delta] \wedge \Delta' \not\models_{\mathcal{T}}\}$.
An incomplete proof-tree π corresponds to UNSAT if it has no open leaf. ※

The $\mathsf{DPLL}(\mathcal{T})$ procedure starts from an initial state i.e. $\emptyset\|\phi$, to which corresponds the incomplete proof-tree consisting of one node (both a root and a leaf) labelled with the sequent $;\phi \vdash$.

Note that, different incomplete proof-trees might correspond to the same $\mathsf{DPLL}(\mathcal{T})$ state, as different $\mathsf{DPLL}(\mathcal{T})$ runs can lead to that state from various initial $\mathsf{DPLL}(\mathcal{T})$ states. The simulation theorem below expresses the fact that, when $\mathsf{DPLL}(\mathcal{T})$ rewrites one state to another state, any incomplete proof-tree corresponding to the formal state can be extended into a incomplete proof-tree corresponding to the latter state.

THEOREM 41 (Simulation of DPLL($\mathcal{T}$) in LK$_{\mathsf{DPLL}}$($\mathcal{T}$)) If $\Delta\|\phi \Rightarrow \mathcal{S}_2$ is a rewrite step of $\mathsf{DPLL}(\mathcal{T})$ and if π_1 corresponds to $\Delta\|\phi$ then there is, in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$, a $\sharp(\phi)+1$-extension π_2 of π_1 corresponding to $\mathcal{S}_2$. ※

Proof: By case analysis on the $\mathsf{DPLL}(\mathcal{T})$ rule. Notice that the bottom-up application of the $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ rules preserves the invariant that, in a sequent $\Delta;\phi \vdash$, Δ is semantically consistent ($\Delta \not\models_{\mathcal{T}}$), and therefore in our case analysis, we do not have to worry about the semantical consistency condition of the correspondence between $\mathsf{DPLL}(\mathcal{T})$ states and incomplete proof-trees.

- Decide: $\Delta\|\phi \Rightarrow \Delta, l^d\|\phi$ where $l \notin \Delta$, $l^\perp \notin \Delta$, $l \in \mathsf{lit}(\phi)$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$. We 1-extend it into π_2 by replacing the open leaf labelled with $\overline{\Delta};\phi \vdash$ (if there is such a leaf) by one of three proof-trees:
  - If $\overline{\Delta}, l \models_{\mathcal{T}}$, we have $\mathsf{Clo}(\overline{\Delta}) = \mathsf{Clo}(\overline{\Delta}, l^\perp)$ and we take:

  $$\frac{\overline{\Delta}, l^\perp;\phi \vdash}{\overline{\Delta};\phi \vdash}\ \text{Weakening2}$$

    The new open leaves form a sub-set of $\{\overline{\Delta}, l^\perp;\phi \vdash\ \} \cup \{\Delta';\phi \vdash \ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash \ \mid \Delta' \in [\Delta, l^d]\}$ (since $\overline{\Delta}, l^\perp = \overline{\Delta, l^\perp} \in [\Delta, l^\perp] = [\![\Delta, l^d]\!] \subseteq [\Delta, l^d]$) and therefore π_2 corresponds to $\Delta, l^d\|\phi$.
  - If $\overline{\Delta}, l^\perp \models_{\mathcal{T}}$, we have $\mathsf{Clo}(\overline{\Delta}) = \mathsf{Clo}(\overline{\Delta}, l)$ and we take

  $$\frac{\overline{\Delta}, l;\phi \vdash}{\overline{\Delta};\phi \vdash}\ \text{Weakening2}$$

The new open leaves form a sub-set of $\{\overline{\Delta}, l;\phi \vdash\ \} \cup \{\Delta';\phi \vdash\ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash\ \mid \Delta' \in [\Delta, l^d]\}$ (since $\overline{\Delta}, l = \overline{\Delta, l} \in [\Delta, l^d]$) and therefore π_2 corresponds to $\Delta, l^d \| \phi$.

– If $\overline{\Delta}, l \not\models_{\mathcal{T}}$ and $\overline{\Delta}, l^\perp \not\models_{\mathcal{T}}$, we take

$$\frac{\overline{\Delta}, l;\phi \vdash \qquad \overline{\Delta}, l^\perp;\phi \vdash}{\overline{\Delta};\phi \vdash}\ \text{Split}$$

The new open leaves form a sub-set of $\{\overline{\Delta}, l;\phi \vdash\ \} \cup \{\overline{\Delta}, l^\perp;\phi \vdash\ \} \cup \{\Delta';\phi \vdash\ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash\ \mid \Delta' \in [\Delta, l^d]\}$ and therefore π_2 corresponds to $\Delta, l^d \| \phi$. (since $\overline{\Delta}, l^\perp = \overline{\Delta, l^\perp} \in [\Delta, l^\perp] = [\![\Delta, l^d]\!] \subseteq [\Delta, l^d]$)

- Propagate: $\Delta \| \phi, C \vee l \Rightarrow \Delta, l \| \phi, C \vee l$ where $\overline{\Delta} \models \neg C$, $l \notin \Delta$, $l^\perp \notin \Delta$.

Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi, C \vee l$. We $\sharp(\phi, C \vee l)$+1-extend it into π_2 by replacing the open leaf labelled with $\overline{\Delta};\phi, C \vee l \vdash$ (if there is such a leaf) by one of three proof-trees:

– If $\overline{\Delta}, l^\perp \models_{\mathcal{T}}$, we have $\mathsf{Clo}(\overline{\Delta}) = \mathsf{Clo}(\overline{\Delta}, l)$ and we take:

$$\frac{\overline{\Delta}, l;\phi \vdash}{\overline{\Delta};\phi \vdash}\ \text{Weakening2}$$

The new open leaves form a sub-set of $\{\overline{\Delta}, l;\phi, C \vee l \vdash\ \} \cup \{\Delta';\phi, C \vee l \vdash\ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi, C \vee l \vdash\ \mid \Delta' \in [\Delta, l]\}$ (since $\overline{\Delta}, l = \overline{\Delta, l} \in [\Delta, l]$) and therefore π_2 corresponds to $\Delta, l \| \phi, C \vee l$.

– If $\overline{\Delta}, l \models_{\mathcal{T}}$ then Lemma 40 directly provides an incomplete proof-tree of $\overline{\Delta};\phi, C \vee l \vdash$.

– If $\overline{\Delta}, l \not\models_{\mathcal{T}}$ and $\overline{\Delta}, l^\perp \not\models_{\mathcal{T}}$, we can construct the following tree:

$$\dfrac{\dfrac{\dfrac{\overline{\Delta}, l;\phi, C \vee l \vdash}{\overline{\Delta}, l;\phi, l \vdash}\ \text{Inverted Resolve}}{\dfrac{\overline{\Delta};\phi, l \vdash}{\overline{\Delta};\phi, C \vee l \vdash}\ \text{Resolve}}\ \text{Assert}$$

where the side-conditions of *Resolve* are provided by the hypothesis $\Delta'' \models \neg C$.

The new open leaves form a sub-set of $\{\overline{\Delta}, l;\phi, C \vee l \vdash\ \} \cup \{\Delta';\phi \vdash\ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash\ \mid \Delta' \in [\Delta, l]\}$ and therefore π_2 corresponds to $\Delta, l \| \phi, C \vee l$. (since $\overline{\Delta}, l = \overline{\Delta, l} \in [\Delta, l]$)

- Fail: $\Delta \| \phi, C \Rightarrow^* \mathsf{UNSAT}$ with $\overline{\Delta} \models \neg C$ and there is no decision literal in Δ.

Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi, C$. Since there are no decision literals in Δ, π_1 can have at most one open leaf, labelled by $\overline{\Delta};\phi, C \vdash$.

We $\sharp(\phi, C)$+1-extend π_1 into π_2 by replacing that leaf by a complete tree deriving $\overline{\Delta};\phi, C \vdash$. We obtain that tree by applying Lemma 40 on the hypothesis $\overline{\Delta} \models \neg C$. The new tree π_2 is complete and therefore corresponds to the UNSAT state of the $\mathsf{DPLL}(\mathcal{T})$ run.

- Propagate$_{\mathcal{T}}$: $\Delta \| \phi \Rightarrow \Delta, l \| \phi$ where $\overline{\Delta} \models_{\mathcal{T}} l$, $l \in \mathsf{lit}(\phi)$ and $l \notin \Delta, l^\perp \notin \Delta$.

Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi$. We 1-extend it into π_2 by replacing the open leaf labelled with $\overline{\Delta};\phi \vdash$ (if it exists) by the following proof-tree:

$$\frac{\overline{\Delta}, l;\phi \vdash}{\overline{\Delta};\phi \vdash}\ \text{Weakening2}$$

as we have $\mathsf{Clo}(\overline{\Delta}) = \mathsf{Clo}(\overline{\Delta}, l)$.

The new open leaves form a sub-set of $\{\overline{\Delta}, l;\phi \vdash\ \} \cup \{\Delta';\phi \vdash\ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash\ \mid \Delta' \in [\Delta, l]\}$ (since $\overline{\Delta}, l = \overline{\Delta, l} \in [\Delta, l]$) and therefore π_2 corresponds to $\Delta, l \| \phi$.

- Fail$_{\mathcal{T}}$: $\Delta \| \phi \Rightarrow \mathsf{UNSAT}$ with $\overline{\Delta} \models_{\mathcal{T}}$ and there is no decision literal in Δ.

Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$. Since there are no decision literals in Δ, and $\overline{\Delta}$ is semantically inconsistent, π_1 has no open leaf, so it is a complete proof-tree that corresponds to the state UNSAT.

- Backtrack: $\Delta_1, l^d, \Delta_2\|\phi, C \Rightarrow \Delta_1, l^\perp\|\phi, C$
  if $\overline{\Delta_1, l, \Delta_2} \models \neg C$ and no decision literal is in Δ_2.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta_1, l^d, \Delta_2\|\phi, C$. We $\sharp(\phi, C)$+1-extend π_1 into π_2 by replacing the leaf labelled with $\overline{\Delta_1, l^d, \Delta_2};\phi, C \vdash$ (if it exists) by a complete tree deriving $\overline{\Delta_1, l^d, \Delta_2};\phi, C \vdash$. We obtain that incomplete proof-tree by applying Lemma 40 on the assumption $\overline{\Delta_1, l^d, \Delta_2} \models \neg C$.

  The new open leaves form a sub-set of $\{\Delta';\phi, C \vdash \ | \ \Delta' \in [\![\Delta_1, l^d, \Delta_2]\!]\}$, which is equal to $\{\Delta';\phi, C \vdash \ | \ \Delta' \in [\![\Delta_1, l^d]\!]\}$ as there are no decision literal in Δ_2, and that set is a sub-set of $\{\Delta';\phi, C \vdash \ | \ \Delta' \in [\Delta_1, l^\perp]\}$ since $\overline{\Delta_1}, l^\perp = \overline{\Delta_1, l^\perp} \in [\Delta_1, l^\perp]$. Therefore π_2 corresponds to $\Delta_1, l^\perp\|\phi, C$ state of the DPLL($\mathcal{T}$) run.

- Backtrack$_\mathcal{T}$: $\Delta_1, l^d, \Delta_2\|\phi \Rightarrow \Delta_1, l^\perp\|\phi$ if $\overline{\Delta_1, l, \Delta_2} \models_\mathcal{T}$ and no decision literal is in Δ_2.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta_1, l^d, \Delta_2\|\phi$. Since Δ_1, l^d, Δ_2 is semantically inconsistent, the open leaves of π_1 form a subset of $\{\Delta';\phi \vdash \ | \ \Delta' \in [\![\Delta_1, l^d, \Delta_2]\!]\}$, and we conclude as in the previous case.

□

We now simulate the full system called Abstract DPLL($\mathcal{T}$) (Definition 34), which involves $\mathcal{T}$-Backjump and $\mathcal{T}$-Learn etc. features.

In order to simulate those extra rules in LK$_{\mathsf{DPLL}}$($\mathcal{T}$), we need to extend LK$_{\mathsf{DPLL}}$($\mathcal{T}$) with a cut rule as follows:

Definition 43 (LK$_{\mathsf{DPLL}}$($\mathcal{T}$) with cut) System *LK$^c_{DPLL}$($\mathcal{T}$)* is obtained by extending system LK$_{\mathsf{DPLL^+}}$($\mathcal{T}$) with the following cut-rule:

$$\frac{\Delta;\phi, l_1, \ldots, l_n \vdash \qquad \Delta;\phi, C \vdash}{\Delta;\phi \vdash}\ \mathit{Cut} \text{ where } C = l_1^\perp \vee \ldots \vee l_n^\perp$$

We define the size of proof-trees in LK$^c_{\mathsf{DPLL}}$($\mathcal{T}$) as we did for LK$_{\mathsf{DPLL^+}}$($\mathcal{T}$) (still ignoring *Weakening1*, *Weakening2* or the *Inverted Resolve*), but also ignoring the left-branch of the cut-rules. As we shall see in the simulation theorem, this definition mimics the fact that the length of DPLL($\mathcal{T}$) sequences is a complexity measure that ignores the cost of checking the side-conditions. ※

Furthermore, the simulation will be such that several open branches of the incomplete proof-trees can be labelled by the same sequent. Therefore we have to slightly relax the notion of corespondance as follows:

Definition 44 (Correspondence between DPLL($\mathcal{T}$) states and incomplete proof-trees) An incomplete proof-tree π *corresponds to* a DPLL($\mathcal{T}$) state $\Delta\|\phi$ if the sequents labelling its open leaves form a sub-set of $\{\Delta';\phi \vdash \ | \ \Delta' \in [\Delta] \wedge \Delta' \not\models_\mathcal{T}\}$. ※

And we now have to consider the *simultaneous* extension of several branches of incomplete proof-trees, which we formalise by the following concepts:

Definition 45 ($n, \phi, \mathcal{S}$-sync action, parallel n-extension of incomplete proof-trees)

- π_ϕ is a $n, \phi, \mathcal{S}$-sync action if it is a function that maps every model sequences $\Delta \in \mathcal{S}$ to an incomplete proof-tree of size at most n and concluding $\Delta;\phi \vdash$.
- π_2 is a *parallel n-extension* of π_1 according to π_ϕ if π_ϕ is a $n, \phi, \mathcal{S}$-sync action and if π_2 is obtained from π_1 by replacing all the open leaves of π_1 labelled by sequents of the form $\Delta;\phi \vdash$ (where $\Delta \in \mathcal{S}$) by $\pi_\phi(\Delta)$.

※

THEOREM 42 (Simulation of Abstract $\mathsf{DPLL}(\mathcal{T})$ into $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$)

If $\Delta\|\phi \Rightarrow_{\mathsf{DPLL}_{bj}(\mathcal{T})} \mathcal{S}_2$ and π_1 corresponds to $\Delta\|\phi$, there is parallel $\sharp(\phi)+3$-extension π_2 of π_1 (according to some π_ϕ) such that π_2 corresponds to $\mathcal{S}_2$. ※

Proof: Since $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ is a sub-system of $\mathsf{LK}^{\mathsf{c}}_{\mathsf{DPLL}}(\mathcal{T})$, we only need to simulate (in $\mathsf{LK}^{\mathsf{c}}_{\mathsf{DPLL}}(\mathcal{T})$) the new rules.

- $\mathcal{T}$-Backjump: $\Delta_1, l^d, \Delta_2\|\phi, C \Rightarrow \Delta_1, l_{bj}\|\phi, C$ if, for some clause C_0 such that $C_0 \subseteq \mathsf{lit}(\phi, C)$,
  1. $\Delta_1, l^d, \Delta_2 \models \neg C$
  2. $\Delta_1 \models \neg C_0$
  3. $\phi, C \models_{\mathcal{T}} C_0 \vee l_{bj}$
  4. $l_{bj} \notin \Delta_1$, $l_{bj}^{\perp} \notin \Delta_1$ and $l_{bj} \in \mathsf{lit}(\phi, \Delta_1, l^d, \Delta_2)$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta_1, l^d, \Delta_2\|\phi, C$. We have to build a π_2 that corresponds to $\Delta_1, l_{bj}\|\phi, C$ in the $\mathsf{DPLL}_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $\Delta';\phi, C \vdash$ where $\Delta' \in [\Delta_1, l_{bj}]$.

  Let $\mathcal{S} = [\Delta_1, l^d, \Delta_2]\backslash[\![\Delta_1]\!]$ and π_ϕ be the $\sharp(\phi, C)+3$, $\phi, C, \mathcal{S}$-sync action that maps every $\Delta \in \mathcal{S}$ to

$$\dfrac{\dfrac{\dfrac{;\phi, C, \neg C', l_{bj}^{\perp} \vdash}{\overline{\Delta_1};\phi, C, \neg C', l_{bj}^{\perp} \vdash}\,Weakening2 \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{\Delta_1}, l_{bj};\phi, C \vdash}{\overline{\Delta_1}, l_{bj};\phi, C, l_{bj} \vdash}\,Subsume}{\overline{\Delta_1};\phi, C, l_{bj} \vdash}\,Assert}{\overline{\Delta_1};\phi, C, C' \vee l_{bj} \vdash}\,Resolve}{\overline{\Delta_1};\phi, C \vdash}\,cut}{\Delta;\phi, C \vdash}\,Weakening2$$

  It is a valid incomplete proof-tree because $\Delta \in \mathcal{S}$ entails $\overline{\Delta_1} \subseteq \Delta$ and therefore $\mathsf{Clo}_\phi(\overline{\Delta_1}) \subseteq \mathsf{Clo}_\phi(\Delta)$. The left branch is closed by assumption (3) and the completeness of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ [Tin02] on $\phi, C, \neg C', l_{bj}^{\perp} \models_{\mathcal{T}}$. We cannot anticipate the size of the proof-tree closing that branch, and we therefore ignore that proof-tree to compute the size of the whole tree, just as the length of the $\mathsf{DPLL}(\mathcal{T})$ run ignores the cost of checking $\phi, C \models_{\mathcal{T}} C' \vee l_{bj}$.

  Let π_2 be the *parallel* $\sharp(\phi, C)+3$-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{\overline{\Delta_1}, l_{bj};\phi, C \vdash\ \} \cup \{\Delta';\phi \vdash\ \mid \Delta' \in [\![\Delta_1]\!]\} \subseteq \{\Delta';\phi \vdash\ \mid \Delta' \in [\Delta_1, l_{bj}]\}$ (since $\overline{\Delta_1}, l_{bj} = \overline{\Delta_1, l_{bj}} \in [\Delta_1, l_{bj}]$ and $[\![\Delta_1, l_{bj}]\!] = [\![\Delta_1]\!]$) and therefore π_2 corresponds to $\Delta_1, l_{bj}\|\phi, C$.

- $\mathcal{T}$-Learn: $\Delta\|\phi \Rightarrow \Delta\|\phi, C$ if each atom of C occurs in ϕ or in Δ and $\phi \models_{\mathcal{T}} C$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$. We have to build a π_2 that corresponds to $\Delta\|\phi, C$ in the $\mathsf{DPLL}_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $\Delta';\phi, C \vdash$ where $\Delta' \in [\Delta]$.

  Let $\mathcal{S} = [\Delta]$ and π_ϕ be the $\sharp(\phi)$,ϕ,$\mathcal{S}$-sync action that maps every $\Delta \in \mathcal{S}$ to:

$$\dfrac{\dfrac{;\phi, \neg C \vdash}{\Delta;\phi, \neg C \vdash}\,Weakening2 \qquad \overline{\Delta};\phi, C \vdash}{\Delta;\phi \vdash}\,cut$$

  The left branch of the cut is closed by assumption and completeness of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ [Tin02] on $\phi, \neg C \models_{\mathcal{T}}$. We cannot anticipate the size of the proof-tree closing that branch, and we therefore ignore that proof-tree to compute the size of the whole tree, just as the length of the $\mathsf{DPLL}(\mathcal{T})$ run ignores the cost of checking $\phi \models_{\mathcal{T}} C$.

  Let π_2 be the *parallel* $\sharp(\phi)$-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{\Delta';\phi, C \vdash\ \mid \Delta' \in [\Delta]\}$ and therefore π_2 corresponds to $\Delta\|\phi, C$.

- $\mathcal{T}$-Forget: $\Delta\|\phi, C \Rightarrow \Delta\|\phi$ if $\phi \models_{\mathcal{T}} C$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi, C$. We have to build a π_2 that corresponds to $\Delta\|\phi$ in the DPLL$_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $\Delta';\phi \vdash$ where $\Delta' \in [\Delta]$.

  Let $\mathcal{S} = [\Delta]$ and π_ϕ be the $1, \phi, C, \mathcal{S}$-sync action that maps every $\Delta' \in \mathcal{S}$ to

$$\frac{\Delta';\phi \vdash}{\Delta';\phi, C \vdash}\ Weakening1$$

  Let π_2 be the *parallel* 1-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{\Delta';\phi \vdash \ \mid \Delta' \in [\Delta]\}$ and therefore π_2 corresponds to $\Delta\|\phi$.

- Restart: $\Delta\|\phi \Rightarrow \emptyset\|\phi$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$. We have to build a π_2 that corresponds to $\emptyset\|\phi$ in the DPLL$_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $;\phi \vdash$.

  Let $\mathcal{S} = [\Delta]$ and π_ϕ be the $1, \phi, \mathcal{S}$-sync action that maps every $\Delta' \in \mathcal{S}$ to:

$$\frac{;\phi \vdash}{\Delta';\phi \vdash}\ Weakening2$$

  Let π_2 be the *parallel* 1-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{;\phi \vdash \ \}$ and therefore π_2 corresponds to $\emptyset\|\phi$.

□

4.3.3 Simulation of LK$_{\text{DPLL}}(\mathcal{T})$ in LK$^p(\mathcal{T})$

In this section, we present the simulation of the intermediate system LK$_{\text{DPLL}}(\mathcal{T})$ in the focused sequent calculus LK$^p(\mathcal{T})$. The main gap between LK$_{\text{DPLL}}(\mathcal{T})$ (or even DPLL($\mathcal{T}$)) and a sequent calculus such as LK$^p(\mathcal{T})$ is the fact that the structures handled by the former are very flexible (e.g. clauses are sets of literals), while sequent calculus implements a root-first decomposition of formulae trees.

The way we encode clauses as formulae of sequent calculus is as follows: a clause C will be represented by a formula C' which is a disjunctive tree whose leaves contain at least all the literals of C but also other literals that we can consider as garbage. Of course, one could fear that the presence of garbage parts within C' degrades the efficiency of proof-search when simulating DPLL($\mathcal{T}$). This garbage comes from the original clauses at the start of the DPLL($\mathcal{T}$) rewriting sequence, which might have been simplified in later steps of DPLL($\mathcal{T}$) but which remain unchanged in sequent calculus. The size of the garbage is therefore smaller than the size of the original problem. We ensure that the inspection, by the proof-search process, of the garbage in $C'^{\perp}$, takes no more inference steps than the size of the garbage itself (the waste of time is linear in the size of the garbage). In order to ensure this, we use polarities and the focusing properties of LK$^p(\mathcal{T})$: the garbage literals in C' must be negative atoms that are negated in the model/context.

The representation of clauses as formulae, informally described above, and the representation of sequents, lead to the formal notion of correspondence below:

DEFINITION 46 ($\mathcal{P}$-correspondence) Let $\mathcal{P}$ be a multiset of literals.

- A formula C' $\mathcal{P}$-corresponds to a clause C (in system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$), where $C = l_1 \vee \ldots \vee l_p$, if $C' = l'_1 \vee^- \ldots \vee^- l'_{p'}$ with $\{l_j\}_{j=1\ldots p} \subseteq \{l'_j\}_{j=1\ldots p'}$ and for any $l \in \{l'_j\}_{j=1\ldots p'} \backslash \{l_j\}_{j=1\ldots p}$, $l^\perp \in \mathcal{P}$.
- An $\mathsf{LK}^p(\mathcal{T})$ sequent $\Delta, C'_1, \ldots, C'_{m'} \vdash^{\mathcal{P}}$ corresponds to an $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ sequent $\Delta; C_1, \ldots, C_m \vdash$, if
  - $m' \geq m$
  - C'_i $\mathcal{P}$-corresponds to C_i for $1 \leq i \leq m$
  - $\Delta \vdash^{\mathcal{P}} C'_i$ for $m < i \leq m'$
  - for all $l \in \Delta$, $l \in \mathcal{P}$
  - for all $l \in \mathcal{P}$, $\Delta \models_{\mathcal{T}} l$.

※

LEMMA 43 If C' $\mathcal{P}$-corresponds to C, then C' also $(\mathcal{P}, l)$-corresponds to C. ※

Proof: Straightforward. □

We can now prove the simulation theorem:

THEOREM 44

Assume $\dfrac{(\mathcal{S}_i)_i}{\mathcal{S}}$ is a rule of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. For every $\mathsf{LK}^p(\mathcal{T})$ sequent $\mathcal{S}'$ that corresponds to $\mathcal{S}$, there exists an incomplete proof-tree in $\mathsf{LK}^p(\mathcal{T})$ whose open leaves $(\mathcal{S}'_i)$ are such that $\forall i$, $\mathcal{S}'_i$ corresponds to $\mathcal{S}_i$. ※

Proof: By case analysis:

- Split:

$$\frac{\Delta, l^\perp; \phi \vdash \qquad \Delta, l; \phi \vdash}{\Delta; \phi \vdash} \text{ where } l \in \mathsf{lit}(\phi), \Delta, l^\perp \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

  Assume that $\Delta, \phi' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi \vdash$.

  In particular, $\phi' = C'_1, \ldots, C'_m$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$.

  We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation that uses an analytic cut:

$$\frac{\dfrac{\Delta, l^\perp, \phi' \vdash^{\mathcal{P}, l^\perp}}{\Delta, \phi' \vdash^{\mathcal{P}} l} \qquad \dfrac{\Delta, l, \phi' \vdash^{\mathcal{P}, l}}{\Delta, \phi' \vdash^{\mathcal{P}} l^\perp}}{\Delta, \phi' \vdash^{\mathcal{P}}}$$

  as neither l nor $l^\perp$ can be in $\mathcal{P}$.

  $\Delta, l^\perp, \phi' \vdash^{\mathcal{P}, l^\perp}$ (resp. $\Delta, l, \phi' \vdash^{\mathcal{P}, l}$) $\mathcal{P}$-corresponds to $\Delta, l^\perp; \phi \vdash$ (resp. $\Delta, l; \phi \vdash$).

- Assert:

$$\frac{\Delta, l; \phi, l \vdash}{\Delta; \phi, l \vdash} \Delta, l^\perp \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

  Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, l \vdash$.

  In particular, $\phi' = C'_1, \ldots, C'_m$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$, and C' $\mathcal{P}$-corresponds to l, that is to say $C' = \vee_{i=1}^p l_i$ where $l = l_{i_0}$ for some $i_0 \in 1 \ldots n$.

  We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation:

$$\dfrac{\begin{array}{cc}(i \neq i_0)\dfrac{\mathsf{lit}_{\mathcal{P}}(\Delta, \phi', C'), l_i \models_{\mathcal{T}}}{\Delta, \phi', C' \vdash^{\mathcal{P}} [l_i^{\perp}]} & \dfrac{\dfrac{l_{i_0}, \Delta, \phi', C' \vdash^{\mathcal{P}, l_{i_0}}}{\Delta, \phi', C' \vdash^{\mathcal{P}} {l_{i_0}}^{\perp}}}{\Delta, \phi', C' \vdash^{\mathcal{P}} [{l_{i_0}}^{\perp}]} \\ \multicolumn{2}{c}{\vdots\, \wedge^{+}} \\ \multicolumn{2}{c}{\Delta, \phi', C' \vdash^{\mathcal{P}} [{C'}^{\perp}]}\end{array}}{\Delta, \phi', C' \vdash^{\mathcal{P}}}$$

For $i \neq i_0$, $l_i^{\perp} \in \mathcal{P}$, so it is positive and we can use an axiom (remember that $\Delta \models l_i^{\perp}$).
Because of the side-condition of the Assert rule, neither l nor $l^{\perp}$ can be in $\mathcal{P}$. Therefore, we release focus and store l_{i_0}, so that $l_{i_0}, \Delta, \phi', C' \vdash^{\mathcal{P}, l_{i_0}}$ $\mathcal{P}$-corresponds to $\Delta, l; \phi, l \vdash$.

- Empty:

$$\dfrac{}{\Delta; \phi, \perp \vdash}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, \perp \vdash$. In particular, $\phi' = C'_1, \ldots, C'_m$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$, and finally C' $\mathcal{P}$-corresponds to $\perp$.
We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation:

$$\dfrac{\begin{array}{c}\dfrac{\mathsf{lit}_{\mathcal{P}}(\Delta, \phi', C'), l_i \models_{\mathcal{T}}}{\Delta, \phi', C' \vdash^{\mathcal{P}} [l_i^{\perp}]} \\ \vdots\, \wedge^{+}. \\ \Delta, \phi', C' \vdash^{\mathcal{P}} [{C'}^{\perp}]\end{array}}{\Delta, \phi', C' \vdash^{\mathcal{P}}}$$

Again, $l_i^{\perp} \in \mathcal{P}$, so it is positive and we can use an axiom (remember that $\Delta \models l_i^{\perp}$).

- Resolve:

$$\dfrac{\Delta; \phi, C \vdash}{\Delta; \phi, l \vee C \vdash}\ \Delta, l \models_{\mathcal{T}}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, l \vee C \vdash$.
In particular, $\phi' = C'_1, \ldots, C'_m$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$, and finally C' $\mathcal{P}$-corresponds to $l \vee C$.
We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation:

$$(\mathsf{Pol})\dfrac{\Delta, \phi', C' \vdash^{\mathcal{P}, l^{\perp}}}{\Delta, \phi', C' \vdash^{\mathcal{P}}}$$

The side-condition of the rule, namely $\mathsf{lit}_{\mathcal{P}, l^{\perp}}(\Delta, \phi', C'), l \models_{\mathcal{T}}$ is a consequence of the side-condition $\Delta, l \models_{\mathcal{T}}$, since $\mathsf{lit}_{\mathcal{P}}(\Delta) = \Delta$ by definition of the correspondence.
Finally, it suffices to notice that $\Delta, \phi', C' \vdash^{\mathcal{P}, l^{\perp}}$ corresponds to $\Delta; \phi, C \vdash$.

- Subsume:

$$\dfrac{\Delta; \phi \vdash}{\Delta; \phi, l \vee C \vdash}\ \Delta, l^{\perp} \models_{\mathcal{T}}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, l \vee C \vdash$.
In particular, $\phi' = C'_1, \ldots, C'_m$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$, and finally C' $\mathcal{P}$-corresponds to $l \vee C$, that is to say $C' = \vee_{i=1}^{p} l_i$ where $l = l_{i_0}$ for some $i_0 \in 1 \ldots n$.
We show that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ also corresponds to $\Delta; \phi \vdash$. The only thing we need to prove is that $\Delta \vdash^{\mathcal{P}} C'$ is derivable in $\mathsf{LK}^p(\mathcal{T})$.

We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation:

$$\dfrac{\dfrac{\dfrac{\Delta, l_1^\perp, \dots, l_p^\perp \vdash^{\mathcal{P}'}}{\Delta \vdash^{\mathcal{P}} l_1, \dots, l_p}}{\Delta \vdash^{\mathcal{P}} l_1 \vee^- \cdots \vee^- l_p}}{\Delta \vdash^{\mathcal{P}} C'}$$

To close the branch, there are two cases:

– if $l \notin \mathcal{P}$, then $l^\perp \in \mathcal{P}'$ and we can conclude with Init_2, since $\mathsf{lit}_{\mathcal{P}'}(\Delta, {l_1}^\perp, \dots, {l_n}^\perp)$ contains $\Delta, l^\perp$.

– if $l \in \mathcal{P}$, then we conclude with the following proof-tree:

$$\dfrac{\dfrac{\mathsf{lit}_{\mathcal{P}'}(\Delta, {l_1}^\perp, \dots, {l_n}^\perp), l^\perp \models_{\mathcal{T}}}{\Delta, l_1^\perp, \dots, l_p^\perp \vdash^{\mathcal{P}'} [l]}}{\Delta, l_1^\perp, \dots, l_p^\perp \vdash^{\mathcal{P}'}}$$

as again $\mathsf{lit}_{\mathcal{P}'}(\Delta, {l_1}^\perp, \dots, {l_n}^\perp)$ contains Δ.

□

Finally, if we want to simulate $\mathsf{DPLL}_{bj}(\mathcal{T})$ (i.e. $\mathsf{DPLL}(\mathcal{T})$ with backjump), then we would need to simulate system $\mathsf{LK}^c_{\mathsf{DPLL}}(\mathcal{T})$ rather than $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. This can be done if, one the sequent calculus side, we do not only consider the analytic cut shown in Fig. 4.2, but also the more general form of cut_7, as well as the $(\mathsf{Store}^=)$ rule:

$$\dfrac{\Gamma \vdash^{\mathcal{P}} A, \Delta \quad \Gamma \vdash^{\mathcal{P}} A^\perp, \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta}\,\mathsf{cut}_7 \qquad (\mathsf{Store}^=)\dfrac{\Gamma, A^\perp \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}$$

With these rules, which in Chapter 3 we have proved admissible and invertible in $\mathsf{LK}^p(\mathcal{T})$, the cut-rule of $\mathsf{LK}^c_{\mathsf{DPLL}}(\mathcal{T})$ can be simulated as follows:

LEMMA 45 (Simulation of cut) Consider the cut rule of $\mathsf{LK}^c_{\mathsf{DPLL}}(\mathcal{T})$:

$$\dfrac{\Delta;\phi, l_1, \dots, l_n \vdash \quad \Delta;\phi, C \vdash}{\Delta;\phi \vdash}\; C = l_1^\perp \vee \dots \vee l_n^\perp$$

For every $\mathsf{LK}^p(\mathcal{T})$ sequent $\mathcal{S}$ that corresponds to $\Delta;\phi \vdash$, there exists an incomplete proof-tree in $\mathsf{LK}^p(\mathcal{T})$ (with $(\mathsf{Store}^=)$ and the general version of cut_7) whose open leaves $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively correspond to $\Delta;\phi, l_1, \dots, l_n \vdash$ and $\Delta;\phi, C \vdash$.

※

Proof: Assume that $\Delta, \phi' \vdash^{\mathcal{P}}$ corresponds to $\Delta;\phi \vdash$.

In particular, $\phi' = C_1', \dots, C_m'$ and $\phi = C_1, \dots, C_n$ with C_i' $\mathcal{P}$-corresponding to C_i for $i = 1 \dots n$.

We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation that uses the general form of cut_7:

$$\dfrac{\dfrac{\dfrac{\Delta, \phi', l_1, \dots, l_n \vdash^{\mathcal{P}}}{\Delta, \phi' \vdash^{\mathcal{P}} l_1^\perp, \dots, l_n^\perp}}{\Delta, \phi' \vdash^{\mathcal{P}} {l_1}^\perp \vee^- \cdots \vee^- {l_n}^\perp} \quad \dfrac{\Delta, \phi', {l_1}^\perp \vee^- \cdots \vee^- {l_n}^\perp \vdash^{\mathcal{P}}}{\Delta, \phi' \vdash^{\mathcal{P}} l_1 \wedge^+ \cdots \wedge^+ l_n}}{\Delta, \phi' \vdash^{\mathcal{P}}}$$

Clearly, $\Delta, \phi', l_1, \dots, l_n \vdash^{\mathcal{P}}$ corresponds to $\Delta;\phi, l_1, \dots, l_n \vdash$ and $\Delta, \phi', ({l_1}^\perp \vee^- \dots \vee^- {l_n}^\perp) \vdash^{\mathcal{P}}$ corresponds to $\Delta;\phi, C \vdash$. □

4.4 Direct simulation of $\mathsf{DPLL}(\mathcal{T})$

In Section 4.3, we have discussed an indirect simulation of Elementary and Abstract $\mathsf{DPLL}(\mathcal{T})$ procedure into $\mathsf{LK}^p(\mathcal{T})$, via an intermediate inference system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

This was the occasion of clarifying and formalising the relation between the $\mathsf{DPLL}(\mathcal{T})$ presentation of [NOT06] and the $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ system of [Tin02].

The latter could then be related to our sequent calculus $\mathsf{LK}^p(\mathcal{T})$, which was also the occasion to strengthen the preliminary work done by Gazeau [Gaz10].

However, the composition of the two simulations has a few shortcomings:

- It does work perfectly well on complete proofs:
  The target of the incremental simulation of (Elementary) $\mathsf{DPLL}(\mathcal{T})$ is $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$, an extension of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ with rules that we have shown admissible. Therefore, any complete proof-tree obtained by the simulation can then be transformed into a complete proof-tree in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ itself, which is the source calculus of our second incremental simulation from $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$.
  But composing in this way the two simulations, only provides an encoding of complete and successful $\mathsf{DPLL}(\mathcal{T})$ runs as complete $\mathsf{LK}^p(\mathcal{T})$ proof-trees; it does not provide an incremental simulation from $\mathsf{DPLL}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$. For this we would need to incrementally simulate $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ in $\mathsf{LK}^p(\mathcal{T})$, which would probably require the extension of $\mathsf{LK}^p(\mathcal{T})$ with new admissible rules (such as Weakenings, etc) or perhaps the relaxation of the notion of correspondence.
- And it turns out that the notion of correspondence that we currently have, is already quite relaxed: for instance the $\mathsf{LK}^p(\mathcal{T})$ sequents that we use contain some “garbage” parts.
- All of this together makes it difficult to strengthen our simulations in the view of, for instance, identifying the target fragment of $\mathsf{LK}^p(\mathcal{T})$ reached by the simulations, characterising it with a simple criterion, and possibly define backward simulations. It also makes it difficult to understand the complexity aspects of the simulations.

Therefore, we now switch to another approach, making a direct simulation of Elementary and Abstract $\mathsf{DPLL}(\mathcal{T})$ into $\mathsf{LK}^p(\mathcal{T})$ system. The polarities of the $\mathsf{LK}^p(\mathcal{T})$ system will play a great role to understand the target fragment of $\mathsf{LK}^p(\mathcal{T})$ reached by the simulation.

4.4.1 Simulating Elementary $\mathsf{DPLL}(\mathcal{T})$

The aim of this section is to describe how the Elementary $\mathsf{DPLL}(\mathcal{T})$ procedure can be transposed into a proof-search process for sequents of the $\mathsf{LK}^p(\mathcal{T})$ calculus.

Again, a complete and successful run of $\mathsf{DPLL}(\mathcal{T})$ is a sequence of transitions $\emptyset\|\phi \Rightarrow^* \mathsf{UNSAT}$, which ensures that the set of clauses ϕ is inconsistent modulo the theory. Hence, we are devising a proof-search process aiming at building an $\mathsf{LK}^p(\mathcal{T})$ proof-tree for sequents of the form $\phi' \vdash$, where ϕ' represents the set of clauses ϕ as a sequent calculus structure, in the following sense:

Definition 47 (Representation of clauses as formulae)
An $\mathsf{LK}^p(\mathcal{T})$ formula C' *represents* a $\mathsf{DPLL}(\mathcal{T})$ clause $\{l_j\}_{j=1\dots p}$ if $C' = l_1 \vee^- \dots \vee^- l_p$.
A set of formulae ϕ' *represents* a set of clauses ϕ if there is a bijection f from ϕ to ϕ' such that for all clauses C in ϕ, $f(\phi)$ represents C. ※

Remark 46 If C' represents C, then $\sharp(C') \leq 2\sharp(C)$ (there are fewer symbols $\vee^-$ than there are literals in C).

Note, here that we carefully use the negative disjunction connective to translate $\mathsf{DPLL}(\mathcal{T})$ clauses. This is crucial not only to mimic $\mathsf{DPLL}(\mathcal{T})$ without duplicating formulae but more generally to control the search space. Again coming back to the $\mathsf{DPLL}(\mathcal{T})$ transition sequence $\emptyset\|\phi \Rightarrow^* \Delta\|\phi$ and its intuitive counterpart in sequent calculus, we have to formalise the notion of *incomplete* proof-tree together with the notion of “filling its holes”:

DEFINITION 48 (Incomplete proof-tree, extension of an incomplete proof-tree)
An *incomplete proof-tree* in $\mathsf{LK}^p(\mathcal{T})$ is a tree labelled with sequents,

- whose leaves are tagged as either *open* or *closed*;
- whose open leaves are labelled with sequents without focus or right-hand side;
- and such that every node that is not an open leaf, together with its children, forms an instance of the $\mathsf{LK}^p(\mathcal{T})$ rules.

The *size* of a incomplete proof-tree is its number of nodes.
An incomplete proof-tree π' is an *extension* of π, if there is a tree (edge and nodes preserving) homomorphism from π to π'. It is an *n-extension* of π, if moreover the difference of size between π' and π is less than or equal to n. ※

REMARK 47 An incomplete proof-tree that has no open leaf is (isomorphic to) a well-formed complete $\mathsf{LK}^p(\mathcal{T})$ proof of the sequent labelling its root. In that case, we say the proof-tree is *complete*.

The intuition that an intermediate $\mathsf{DPLL}(\mathcal{T})$ state describes an "interface" between an incomplete proof-tree and the complete proof-trees that should be plugged into its holes, is formalised as follows:

DEFINITION 49 (Correspondence)
An incomplete proof-tree π *corresponds to* a $\mathsf{DPLL}(\mathcal{T})$ state $\Delta\|\phi$ if:

- the length of $\overline{\Delta} :: [\![\Delta]\!]$ is the number of open leaves of π;
- if Δ_i is the i^{th} element of $\overline{\Delta} :: [\![\Delta]\!]$, then the sequent labelling the i^{th} open leaf of π (taken left-to-right) is of the form $\Delta', \phi' \vdash^{\Delta_i}$, where:
  - ϕ' represents ϕ (in the sense of Definition 47);
  - for all $l \in \Delta'$, $l \in \Delta_i$
  - for all $l \in \Delta_i$, $\Delta' \models_{\mathcal{T}} l$.[1]

An incomplete proof-tree π *corresponds to* the state UNSAT if it has no open leaf. ※

REMARK 48 In the general case, different incomplete proof-trees might correspond to a same $\mathsf{DPLL}(\mathcal{T})$ state (just like different $\mathsf{DPLL}(\mathcal{T})$ runs may reach that state from the initial one).

Note that we do not require anything from the conclusion of an incomplete proof-tree corresponding to $\Delta\|\phi$: just as our correspondence says nothing about the $\mathsf{DPLL}(\mathcal{T})$ transitions taking place after $\Delta\|\phi$ (nor about the trees to be plugged into the open leaves), it says nothing about the transitions taking place before $\Delta\|\phi$ (nor about the incomplete proof-tree, except for its open leaves).

If an incomplete proof-tree π corresponds to a $\mathsf{DPLL}(\mathcal{T})$ state $\Delta\|\phi$ where there is no decision literals in Δ, then there is exactly one open leaf in π, and it is labelled by a sequent of the form $\Delta', \phi' \vdash^{\overline{\Delta}}$, where ϕ' represents ϕ and $\mathsf{Clo}_\phi(\Delta) = \mathsf{Clo}_\phi(\Delta')$.

To the initial state $\emptyset\|\phi$ of a run of the $\mathsf{DPLL}(\mathcal{T})$ procedure corresponds the incomplete proof-tree consisting of one node (both root and open leaf) labelled with the sequent $\phi' \vdash$, where ϕ' represents ϕ.

The simulation theorem below provides a systematic way of interpreting any $\mathsf{DPLL}(\mathcal{T})$ transition as a completion of incomplete proof-trees that preserves the correspondence given in Definition 49 and controls the growth of the proof trees.

THEOREM 49 (Simulation of $\mathsf{DPLL}(\mathcal{T})$ in $\mathsf{LK}^p(\mathcal{T})$)
If $\Delta\|\phi \Rightarrow \mathcal{S}_2$ is a valid $\mathsf{DPLL}(\mathcal{T})$ transition, and π_1 is an incomplete proof tree in $\mathsf{LK}^p(\mathcal{T})$ corresponding to $\Delta\|\phi$, then there exists a $(2\sharp(\phi) + 3)$-extension π_2 of π_1 that corresponds to $\mathcal{S}_2$. ※

Proof: By case analysis on the nature of the transition, completing the leftmost open leaf of π_1:

[1]The last two conditions entail in particular that $\mathsf{Clo}_\phi(\Delta') = \mathsf{Clo}_\phi(\Delta_i)$.

- Decide: $\Delta \| \phi \Rightarrow \Delta, l^d \| \phi$ where $l \notin \Delta$, $l^\perp \notin \Delta$, $l \in \mathsf{lit}(\phi)$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$. The leftmost leaf (corresponding to $\overline{\Delta}$) is of the form $\Delta', \phi' \vdash^{\overline{\Delta}}$ where ϕ' represents ϕ and $\mathsf{Clo}_\phi(\overline{\Delta}) = \mathsf{Clo}_\phi(\Delta')$.

  We extend π_1 into π_2 by replacing the leftmost leaf by the following (incomplete) proof-tree:

$$\dfrac{\dfrac{\Delta', l, \phi' \vdash^{\overline{\Delta};l}}{\Delta', \phi' \vdash^{\overline{\Delta}} l^\perp} \qquad \dfrac{\Delta', l^\perp, \phi' \vdash^{\overline{\Delta};l^\perp}}{\Delta', \phi' \vdash^{\overline{\Delta}} l}}{\Delta', \phi' \vdash^{\overline{\Delta}}}$$

  Note that we use here the analytic cut rule of $\mathsf{LK}^p(\mathcal{T})$. π_2 is a 3-extension of π_1 that corresponds to $\Delta, l^d\|\phi$. Indeed, we have $\overline{\Delta, l^d} :: [\![\Delta, l^d]\!] = (\overline{\Delta}, l) :: (\overline{\Delta}, l^\perp) :: [\![\Delta]\!]$ and $\mathsf{Clo}_\phi(\overline{\Delta}, l) = \mathsf{Clo}_\phi(\Delta', l)$ and $\mathsf{Clo}_\phi(\overline{\Delta}, l^\perp) = \mathsf{Clo}_\phi(\Delta', l^\perp)$. The two new leaves are tagged as open.

- Propagate: $\Delta\|\phi, C \vee l \Rightarrow \Delta, l\|\phi, C \vee l$ where $\Delta \models \neg C$, $l \notin \Delta$, $l^\perp \notin \Delta$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi, C \vee l$. The open leaf corresponding to $\overline{\Delta}$ is of the form $\Delta', \phi', C' \vdash^{\overline{\Delta}}$ where ϕ' represents ϕ, C' represents $C \vee l$ and $\mathsf{Clo}_{\phi, C\vee l}(\overline{\Delta}) = \mathsf{Clo}_{\phi, C\vee l}(\Delta')$. Let $C = l_1 \vee \ldots \vee l_n$. From $\Delta \models \neg C$ we get $\forall i$, $l_i^\perp \in \overline{\Delta} \subseteq \mathsf{Clo}_{\phi, C\vee l}(\overline{\Delta}) = \mathsf{Clo}_{\phi, C\vee l}(\Delta')$.

  We extend π_1 into π_2 by replacing this leaf by the following (incomplete) proof-tree:

$$\dfrac{\dfrac{\dfrac{\Delta', l, \phi', C' \vdash^{\overline{\Delta};l}}{\Delta', \phi', C' \vdash^{\overline{\Delta}} l^\perp}}{\Delta', \phi', C' \vdash^{\overline{\Delta}} [l^\perp]} \qquad \left(\dfrac{}{\Delta', \phi', C' \vdash^{\overline{\Delta}} [l_i^\perp]}\right)_{l_i \in C}}{\begin{array}{c} \vdots\, (\wedge^+). \\ \dfrac{\Delta', \phi', C' \vdash^{\overline{\Delta}} [C'^\perp]}{\Delta', \phi', C' \vdash^{\overline{\Delta}}} \end{array}}$$

  The top-right rules can be applied since $l_i^\perp \in \overline{\Delta}$ and $\Delta', l_i \models_{\mathcal{T}}$ and the new leaves are closed. The top-left leaf is tagged as open.

  Noticing that $\overline{\Delta, l} :: [\![\Delta, l]\!] = (\overline{\Delta}, l) :: [\![\Delta]\!]$, we get that π_2 is a $\sharp(C') + 3$-extension of π_1 that corresponds to $\Delta, l\|\phi, C \vee l$ (and note that $\sharp(C') \leq 2\sharp(\phi)$).

- Fail: $\Delta\|\phi, C \Rightarrow \mathsf{UNSAT}$ with $\Delta \models \neg C$ and there is no decision literal in Δ.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi, C$. Since there are no decision literals in Δ, π_1 has exactly one open leaf, and it is labelled by $\Delta', \phi', C' \vdash^{\overline{\Delta}}$ where ϕ' represents ϕ, C' represents C and $\mathsf{Clo}_{\phi, C}(\Delta) = \mathsf{Clo}_{\phi, C}(\Delta')$. Let $C = l_1 \vee \ldots \vee l_n$.

  From $\Delta \models \neg C$ we get $\forall i$, $l_i^\perp \in \overline{\Delta} \subseteq \mathsf{Clo}_{\phi, C}(\Delta) = \mathsf{Clo}_{\phi, C}(\Delta')$.

  We extend π_1 into π_2 by replacing the open leaf by the following (complete) proof-tree:

$$\dfrac{\left(\dfrac{}{\Delta', \phi', C' \vdash^{\overline{\Delta}} [l_i^\perp]}\right)_{l_i \in C}}{\dfrac{\vdots\, \wedge^+.}{\dfrac{\Delta', \phi', C' \vdash^{\overline{\Delta}} [C'^\perp]}{\Delta', \phi', C' \vdash^{\overline{\Delta}}}}}$$

The top rules can be applied since $l_i^\perp \in \overline{\Delta}$ and $\Delta', l_i \models_{\mathcal{T}}$. All the leaves are closed. π_2 is a $\sharp(C') + 1$-extension of π_1 that is complete, and therefore corresponds to the UNSAT state of the DPLL($\mathcal{T}$) run (and note that $\sharp(C') \leq 2\sharp(\phi)$).

- Backtrack: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l^\perp \| \phi, C$ if $\Delta_1, l, \Delta_2 \models \neg C$ and no decision literal is in Δ_2.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta_1, l^d, \Delta_2 \| \phi, C$. The open leaf corresponding to $\overline{\Delta_1, l^d, \Delta_2}$ is of the form $\Delta', \phi', C' \vdash^{\overline{\Delta_1}, l, \overline{\Delta_2}}$ where ϕ' represents ϕ, C' represents C and $\mathsf{Clo}_\phi(\Delta_1, l, \Delta_2) = \mathsf{Clo}_\phi(\Delta')$. Let $C = l_1 \vee \ldots \vee l_n$.

  From $\overline{\Delta_1}, l, \overline{\Delta_2} \models \neg C$ we get $\forall i,\ l_i^\perp \in \overline{\Delta_1}, l, \overline{\Delta_2} \subseteq \mathsf{Clo}_{\phi,C}(\overline{\Delta_1}, l, \overline{\Delta_2}) = \mathsf{Clo}_{\phi,C}(\Delta')$.

  We extend π_1 into π_2 by replacing this leaf by the following (complete) proof-tree:

$$\dfrac{\left(\dfrac{}{\Delta', \phi', C' \vdash^{\overline{\Delta_1}, l, \overline{\Delta_2}} [l_i^\perp]}\right)_{l_i \in C}}{\dfrac{\vdots\, \wedge^+.}{\dfrac{\Delta', \phi', C' \vdash^{\overline{\Delta_1}, l, \overline{\Delta_2}} [C'^\perp]}{\Delta', \phi', C' \vdash^{\overline{\Delta_1}, l, \overline{\Delta_2}}}}}$$

  The top rules can be applied since $l_i^\perp \in \overline{\Delta_1}, l, \overline{\Delta_2}$ and $\Delta', l_i \models_{\mathcal{T}}$.

  Noticing that $[\![\Delta_1, l^d, \Delta_2]\!] = (\overline{\Delta_1}, l^\perp) :: [\![\Delta_1]\!]$, we get that π_2 is a $\sharp(C') + 1$-extension of π_1 that corresponds to $\Delta_1, l^\perp \| \phi, C$ (and note that $\sharp(C') \leq 2\sharp(\phi)$).

- Propagate$_{\mathcal{T}}$: $\Delta \| \phi \Rightarrow \Delta, l \| \phi$ where $l \in \mathsf{Clo}_\phi(\Delta)$ and $l \notin \Delta, l^\perp \notin \Delta$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi$. The open leaf corresponding to $\overline{\Delta}$ is of the form $\Delta', \phi' \vdash^{\overline{\Delta}}$ where ϕ' represents ϕ and $\mathsf{Clo}_\phi(\Delta) = \mathsf{Clo}_\phi(\Delta')$. We extend π_1 into π_2 by replacing this leaf by the following (incomplete) proof-tree:

$$\dfrac{\Delta', \phi' \vdash^{\overline{\Delta}, l}}{\Delta', \phi' \vdash^{\overline{\Delta}}}$$

  Noticing that $\mathsf{Clo}_\phi(\Delta) = \mathsf{Clo}_\phi(\Delta, l)$, π_2 is a 1-extension of π_1 that corresponds to $\Delta, l \| \phi$.

- Fail$_{\mathcal{T}}$: $\Delta \| \phi \Rightarrow$ UNSAT with $\Delta \models_{\mathcal{T}}$ and there is no decision literal in Δ.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi$. Since there are no decision literals in Δ, π_1 has exactly one open leaf, and it is labelled by $\Delta', \phi' \vdash^{\overline{\Delta}}$ where ϕ' represents ϕ and $\mathsf{Clo}_\phi(\Delta) = \mathsf{Clo}_\phi(\Delta')$.

  We extend π_1 into π_2 by replacing the open leaf by the following (complete) proof-tree:

$$\dfrac{}{\Delta', \phi' \vdash^{\overline{\Delta}}}\ \Delta', \phi' \models_{\mathcal{T}}$$

Here, π_2 is a 1-extension of π_1 that is complete and corresponds to UNSAT state of the DPLL($\mathcal{T}$) run.

- Backtrack$_{\mathcal{T}}$: $\Delta_1, l^d, \Delta_2 \| \phi \Rightarrow \Delta_1, l^\perp \| \phi$ if $\Delta_1, l, \Delta_2 \models_{\mathcal{T}}$ and no decision literal is in Δ_2.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta_1, l^d, \Delta_2 \| \phi$. The open leaf corresponding to $\overline{\Delta_1, l^d, \Delta_2}$ is of the form $\Delta', \phi' \vdash^{\overline{\Delta_1}, l, \overline{\Delta_2}}$ where ϕ' represents ϕ and $\mathsf{Clo}_\phi(\Delta_1, l, \Delta_2) = \mathsf{Clo}_\phi(\Delta')$.

  We extend π_1 into π_2 by replacing this leaf by the following (complete) proof-tree:

$$\dfrac{}{\Delta', \phi' \vdash^{\overline{\Delta_1}, l, \overline{\Delta_2}}}\ \Delta', \phi' \models_{\mathcal{T}}$$

  Noticing that $[\![\Delta_1, l^d, \Delta_2]\!] = (\overline{\Delta_1}, l^\perp) :: [\![\Delta_1]\!]$, we get that π_2 is a 1-extension of π_1 that corresponds to $\Delta_1, l^\perp \| \phi$ state of the DPLL($\mathcal{T}$) run.

□

Corollary 50 If $\emptyset \| \phi \Rightarrow^n$ UNSAT and ϕ' represents ϕ then there is an complete proof in LK$^p(\mathcal{T})$ of $\phi' \vdash$, of size smaller than $(2\sharp(\phi) + 3)n$. ※

4.4.2 Turning the simulation into a bisimulation

Now the point of having mentioned quantitative information in Theorem 49, via the notion of n-extension, is to motivate the idea that performing proof-search directly in LK$^p(\mathcal{T})$ is in essence not less efficient than running DPLL($\mathcal{T}$): we have a linear bound in the length of the DPLL($\mathcal{T}$) run (and the proportionality ratio is itself an affine function of the size of the original problem).

We also need to make sure that this final proof-tree is indeed found as efficiently as running DPLL($\mathcal{T}$), which can be done by identifying, in LK$^p(\mathcal{T})$, a (complete) search space that is isomorphic to (and hence no wider than) that of DPLL($\mathcal{T}$). We analyse for this a proof-search strategy, in LK$^p(\mathcal{T})$, that captures all the proof-extensions that we have used in the simulation of DPLL($\mathcal{T}$), i.e. the proof of Theorem 49:

Definition 50 (DPLL($\mathcal{T}$)-extensions)

An incomplete proof tree π_2 is a *DPLL($\mathcal{T}$)-extension* of an incomplete proof tree π_1 if

1. it extends π_1 by replacing its leftmost open leaf with an incomplete proof-tree whose only occurrences of (Select) are such that P is a (positive) conjunction of literals that are all in $\mathcal{P}$ except maybe one that is $\mathcal{P}$-unpolarised;
2. and any incomplete proof-tree satisfying point 1. and extended by π_2 is π_2 itself.

※

Not only are all the extensions that we have used in the simulation DPLL($\mathcal{T}$)-extensions, but all DPLL($\mathcal{T}$)-extensions correspond to one of the extensions used in the simulation, yielding a simulation back into DPLL($\mathcal{T}$):

Theorem 51 (Simulation of the strategy back into DPLL($\mathcal{T}$))

If π_2 is a DPLL($\mathcal{T}$)-extension of π_1, and π_1 corresponds to $\Delta \| \phi$, then there is a valid DPLL($\mathcal{T}$) transition $\Delta \| \phi \Rightarrow \mathcal{S}_2$ such that π_2 corresponds to $\mathcal{S}_2$. ※

Proof: By case analysis on the shape of the incomplete proof-tree replacing the leftmost open leaf of π_1. Definition 50 leads to five cases:

-
$$\dfrac{\dfrac{\cdots}{\Gamma, P^\perp \vdash^{\mathcal{P}} [P]}}{\Gamma, P^\perp \vdash^{\mathcal{P}}}$$

  where P is a (positive) conjunction of literals that are all in $\mathcal{P}$;
  this is simulated by a Fail or Backtrack (depending on whether π_2 is complete) on the clause represented by $P^\perp$;

-
$$\dfrac{\dfrac{\dots}{\Gamma, P^{\perp} \vdash^{\mathcal{P}} [P]}}{\Gamma, P^{\perp} \vdash^{\mathcal{P}}}$$
where P is a (positive) conjunction of literals that are all in $\mathcal{P}$ except one that is $\mathcal{P}$-unpolarised; this is simulated by Propagate on the clause represented by $P^{\perp}$;
-
$$\dfrac{\Gamma \vdash^{\mathcal{P}} l \quad \Gamma \vdash^{\mathcal{P}} l^{\perp}}{\Gamma \vdash^{\mathcal{P}}}$$
is simulated by Decide on l;
-
$$\dfrac{\Gamma \vdash^{\mathcal{P},l}}{\Gamma \vdash^{\mathcal{P}}}$$
is simulated by $\mathsf{Propagate}_{\mathcal{T}}$ on l;
-
$$\dfrac{}{\Gamma \vdash^{\mathcal{P}}}\ \mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}$$
is simulated by $\mathsf{Fail}_{\mathcal{T}}$ or $\mathsf{Backtrack}_{\mathcal{T}}$ (depending on whether π_2 is complete).

The details are the same as in the proof of Theorem 41. □

If an complete proof-tree of $\mathsf{LK}^p(\mathcal{T})$, whose conclusion is an SMT-problem i.e. it corresponds to an initial state of $\mathsf{DPLL}(\mathcal{T})$, systematically uses the rules in the way described by the above shapes, then it is the image of a $\mathsf{DPLL}(\mathcal{T})$ run.

Corollary 52 (Bisimulation)

The Elementary $\mathsf{DPLL}(\mathcal{T})$ procedure is bisimilar to the gradual completion of (incomplete) proof-trees of $\mathsf{LK}^p(\mathcal{T})$ as defined by the strategy of $\mathsf{DPLL}(\mathcal{T})$-extensions. ※

4.4.3 Extending the simulation with backjump and lemma learning

We have already mentioned the Abstract $\mathsf{DPLL}(\mathcal{T})$ system in Definition 34, an advanced version of our Elementary $\mathsf{DPLL}(\mathcal{T})$ that involves backjumping and lemma learning features.

In this section, we extend our simulation result from Section 4.4 in order to simulate $\mathsf{DPLL}_{bj}(\mathcal{T})$ in our sequent calculus. In the rules $\mathcal{T}$-Backjump and $\mathcal{T}$-Learn (see Section 4.1), we see that a new clause is used (e.g. in the side-conditions) that we had not seen before (respectively: $C_0 \vee l_{bj}$ and C). In order to simulate those extra rules in $\mathsf{LK}^p(\mathcal{T})$, we need to extend the calculus with the general form of cut_7, as we did at the end of Section 4.3.3, so that the production of the new clause corresponds to the choice of the cut-formula.

Definition 51 ($\mathsf{LK}^p(\mathcal{T})$ with cut) System $\mathsf{LK}^p_{\mathsf{c}}(\mathcal{T})$ is obtained by extending system $\mathsf{LK}^p(\mathcal{T})$ with the following cut-rule, which is admissible in the cut-free system (Section 3.5)
$$\dfrac{\Gamma \vdash^{\mathcal{P}} A^{\perp} \quad \Gamma \vdash^{\mathcal{P}} A}{\Gamma \vdash^{\mathcal{P}}}\,\mathsf{cut}$$
※

As opposed to what happens in an elementary $\mathsf{DPLL}(\mathcal{T})$ run, the extra rules of $\mathit{DPLL}_{bj}(\mathcal{T})$ can add or remove objects from a state (clauses to falsify, literals). On the contrary, once such an object is introduced in a $\mathsf{LK}^p(\mathcal{T})$ sequent by the proof-search process, this data persists in the entire subtree proving the sequent. This phenomenon is described in [NOT05], which concludes that an abstract presentation of $\mathsf{DPLL}(\mathcal{T})$ based on sequent calculus is necessarily too rigid to model the rules that practical implementations of $\mathsf{DPLL}(\mathcal{T})$ rely on. The simulation theorem we propose in this section shows that a combination of tags and polarisation can actually overcome this discrepancy. Such a simulation theorem for $\mathsf{DPLL}_{bj}(\mathcal{T})$ however requires to slightly relax the notion of correspondence between states and incomplete proof-trees in $\mathsf{LK}^p(\mathcal{T})$: we should allow the sequent label of an open leaf to contain some objects that have disappeared from the corresponding state.

Definition 52 ($\mathsf{LK}^p_c(\mathcal{T})$ incomplete proof-trees corresponding to DPLL($\mathcal{T}$) states) An incomplete proof-tree π *corresponds* a DPLL($\mathcal{T}$) state $\Delta\|\phi$ if:

- there is a mapping from the open leaves of π to the elements of the sequence $\overline{\Delta}::[\![\Delta]\!]$
- the sequent labelling an open leaf mapped to a set Δ_0 in the sequence $\overline{\Delta}::[\![\Delta]\!]$ is of the form $\Delta',\phi',\Gamma\vdash^{\mathcal{P}}$, where:
  - $\Delta_0\subseteq\mathcal{P}\subseteq\mathsf{Clo}_\phi(\Delta')$;
  - ϕ' represents ϕ;
  - for each $A\in\Gamma$, $\phi\models_{\mathcal{T}} A$.

An incomplete proof-tree π *corresponds to* the state UNSAT if it has no open leaf. ※

The difference with the previous notion of *correspondence* is that the open leaves are no longer in 1-to-1 correspondence with the elements of $\overline{\Delta}::[\![\Delta]\!]$: for any $\Delta_0\in\overline{\Delta}::[\![\Delta]\!]$, 0, 1, or several open leaves may be mapped to it. Furthermore, the sequent $\Delta',\phi',\Gamma\vdash^{\mathcal{P}}$ labelling such a leaf corresponding to Δ_0 may

- declare more positive literals than Δ_0
- have Δ' entail more literals than Δ_0
- contain extra formulae Γ on its left-hand side, that are not representing clauses in the DPLL($\mathcal{T}$) state $\Delta\|\phi$, but that are consequences of them.

We can now formulate and prove the equivalent of Theorem 41 for $\mathsf{DPLL}_{bj}(\mathcal{T})$.

Theorem 53 (Simulation of $\mathsf{DPLL}_{bj}(\mathcal{T})$ in $\mathsf{LK}^p_c(\mathcal{T})$)
If $\Delta\|\phi\Rightarrow\mathcal{S}_2$ is a valid $\mathsf{DPLL}_{bj}(\mathcal{T})$ transition and π_1 corresponds to $\Delta\|\phi$, then there exists an extension π_2 of π_1 that corresponds to $\mathcal{S}_2$. ※

Proof: The proof goes by case analysis on the nature of the transition, extending the open leaves of π_1 that are mapped to $\overline{\Delta}$ in the correspondence between π_1 and $\Delta\|\phi$.

Since $\mathsf{DPLL}_{bj}(\mathcal{T})$ extends DPLL($\mathcal{T}$), we still have to simulate Decide, Propagate, Propagate$_{\mathcal{T}}$, Fail, Fail$_{\mathcal{T}}$, Fail and Backtrack$_{\mathcal{T}}$. Rules Propagate$_{\mathcal{T}}$, Fail$_{\mathcal{T}}$, and Backtrack$_{\mathcal{T}}$ can be decomposed into a $\mathcal{T}$-Learn step, a Propagate / Fail / Backtrack step, and a $\mathcal{T}$-Forget step, so they will be covered by the simulation of the other rules. For Decide, Propagate, Fail and Backtrack, we could argue that we have already treated them in the proof of Theorem 41. Yet that theorem involves a different kind of correspondence between states and incomplete proof-trees and, while this notion is stronger, Theorem 41 does not strictly speaking entail what we need here. However, the way the leaves are extended is exactly the same as in the proof of Theorem 41;

- Decide: $\Delta\|\phi\Rightarrow\Delta,l^d\|\phi$ if $l\in\mathsf{lit}(\phi)$, $l\notin\Delta$, $l^\perp\notin\Delta$.

  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$.

  We extend π_1 into π_2 as follows: consider an open leaf mapped to $\overline{\Delta}$ (necessarily labelled by a sequent of the form $\Delta',\phi',\Gamma\vdash^{\mathcal{P}}$); then

  - If neither $l\in\mathcal{P}$ nor $l^\perp\in\mathcal{P}$, the leaf is replaced by the following proof-tree:

$$\dfrac{\dfrac{\Delta',l,\phi',\Gamma\vdash^{\mathcal{P};l}}{\Delta',\phi',\Gamma\vdash^{\mathcal{P}} l^\perp}\qquad\dfrac{\Delta',l^\perp,\phi',\Gamma\vdash^{\mathcal{P};l^\perp}}{\Delta',\phi',\Gamma\vdash^{\mathcal{P}} l}}{\Delta',\phi',\Gamma\vdash^{\mathcal{P}}}$$

    The new left (resp. right) leaf is mapped to the element $\overline{\Delta,l}$ (resp. the element $\overline{\Delta,l^\perp}$) of $\overline{\Delta,l^d}::[\![\Delta,l^d]\!]=(\overline{\Delta},l)::(\overline{\Delta},l^\perp)::[\![\Delta]\!]$.

– If $l \in \mathcal{P}$ (resp. $l^\perp \in \mathcal{P}$) is untouched, by is now mapped (in the new correspondence) to the element $\overline{\Delta, l}$ (resp. the element $\overline{\Delta, l^\perp}$) of $\overline{\Delta, l^d} :: [\![\Delta, l^d]\!] = (\overline{\Delta}, l) :: (\overline{\Delta}, l^\perp) :: [\![\Delta]\!]$.

π_2 is an extension of π_1 that corresponds to $\Delta, l^d \| \phi$.

- Propagate: $\Delta \| \phi, C \vee l \Rightarrow \Delta, l \| \phi, C \vee l$ if $\Delta \models \neg C$, $l \notin \Delta$, $l^\perp \notin \Delta$.

Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi, C \vee l$.

We extend π_1 into π_2 as follows: consider an open leaf mapped to $\overline{\Delta}$ (necessarily labelled by a sequent of the form $\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}$, with C' now representing $C \vee l$); then

– If neither $l \in \mathcal{P}$ nor $l^\perp \in \mathcal{P}$, the leaf is replaced by the following proof-tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta', l, \phi', C', \Gamma \vdash^{\mathcal{P};l}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} l^\perp}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [l^\perp]} \qquad \left(\dfrac{}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [l_i^\perp]} \right)_{l_i \in C}}{\begin{array}{c} \vdots\, (\wedge^+). \\ \Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [C'^\perp] \end{array}}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}}$$

For the top-right rules, hypothesis $\Delta \models \neg C$ entails that for all $l_i \in C$, $l_i^\perp \in \overline{\Delta} \subseteq \mathcal{P} \subseteq \mathsf{Clo}_{\phi,C}(\Delta')$, so $l_i^\perp$ is $\mathcal{P}$-positive and $\Delta', l_i \models_{\mathcal{T}}$. The top-left leaf is tagged as open, and mapped to $\overline{\Delta, l}$.

– If $l \in \mathcal{P}$ then the leaf is untouched, but is now mapped to $\overline{\Delta, l}$.

– If $l^\perp \in \mathcal{P}$ then we close the branch altogether by replacing the leaf with:

$$\dfrac{\dfrac{\dfrac{}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [l^\perp]} \qquad \left(\dfrac{}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [l_i^\perp]} \right)_{l_i \in C}}{\begin{array}{c} \vdots\, (\wedge^+). \\ \Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [C'^\perp] \end{array}}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}}$$

π_2 is an extension of π_1 that corresponds to $\Delta, l \| \phi, C \vee l$.

- Fail: $\Delta \| \phi, C \Rightarrow \mathsf{UNSAT}$ if $\Delta \models \neg C$ and there is no decision literal in Δ.
- Backtrack: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l^\perp \| \phi, C$ if $\Delta_1, l, \Delta_2 \models \neg C$ and there is no decision literal in Δ_2.

We treat Fail and Backtrack at the same time, taking $\Delta := \Delta_1, l, \Delta_2$ in the case of Backtrack.

Let π_1 be an incomplete proof-tree corresponding to $\Delta \| \phi, C$.

We extend π_1 into π_2 by replacing every open leaf mapped to $\overline{\Delta}$ (necessarily labelled by a sequent of the form $\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}$) by the following (complete) proof-tree:

$$\dfrac{\dfrac{\left(\dfrac{}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [l_i^\perp]} \right)_{l_i \in C}}{\begin{array}{c} \vdots\, (\wedge^+). \\ \Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} [C'^\perp] \end{array}}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}}$$

For the top rules, hypothesis $\Delta \models \neg C$ entails that for all $l_i \in C$, $l_i^\perp \in \overline{\Delta} \subseteq \mathcal{P} \subseteq \mathsf{Clo}_{\phi,C}(\Delta')$, so $l_i^\perp$ is $\mathcal{P}$-positive and $\Delta', l_i \models_{\mathcal{T}}$.

Moreover in the case of Fail, since there are no decision literals in Δ ($[\![\Delta]\!] = \emptyset$), the open leaves of π_1 are all mapped to $\overline{\Delta}$ and therefore π_2 is complete and corresponds to the UNSAT state of the DPLL($\mathcal{T}$) run.

In the case of $\mathsf{Backtrack}$, the open leaves remaining in π_2 are those open leaves of π_1 mapped to $[\![\Delta_1, l^d, \Delta_2]\!] = \overline{\Delta_1, l^\perp} :: [\![\Delta_1]\!]$, and therefore π_2 corresponds to the state $\Delta_1, l^\perp \| \phi, C$.

- $\mathcal{T}$-$\mathsf{Backjump}$: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l_{bj} \| \phi, C$ if, for some clause $C_0 = \{l_1, \ldots, l_n\} \subseteq \mathsf{lit}(\phi, C)$,

1. $\Delta_1, l^d, \Delta_2 \models \neg C$.
2. $\Delta_1 \models \neg C_0$
3. $\phi, C \models_{\mathcal{T}} C_0 \vee l_{bj}$
4. $l_{bj} \notin \Delta_1$, $l_{bj}^\perp \notin \Delta_1$ and $l_{bj} \in \mathsf{lit}(\phi, \Delta_1, l^d, \Delta_2)$.

Let π_1 be an incomplete proof-tree corresponding to $\Delta_1, l^d, \Delta_2 \| \phi, C$. We have to build a π_2 that corresponds to $\Delta_1, l_{bj} \| \phi, C$.

Notice that if Δ_0 is in $\overline{\Delta_1, l^d, \Delta_2} :: [\![\Delta_1, l^d, \Delta_2]\!]$, then either it is in $[\![\Delta_1]\!]$ or $\overline{\Delta_1} \subseteq \Delta_0$.

We extend π_1 into π_2 as follows: consider an open leaf mapped to some Δ_0 in $\overline{\Delta_1, l^d, \Delta_2} :: [\![\Delta_1, l^d, \Delta_2]\!]$, such that $\overline{\Delta_1} \subseteq \Delta_0$ (that leaf is necessarily labelled by a sequent of the form $\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}$); then

– If neither $l_{bj} \notin \mathcal{P}$ and $l_{bj}^\perp \notin \mathcal{P}$, the leaf is replaced by the following proof-tree:

$$\dfrac{\dfrac{\dfrac{\pi_3}{\Delta', \phi', l_{bj}^\perp, C', \Gamma \vdash^{\mathcal{P}; l_{bj}^\perp}}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} l_{bj}} \qquad \dfrac{\Delta', \phi', l_{bj}, C', \Gamma \vdash^{\mathcal{P}; l_{bj}}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}} l_{bj}^\perp}}{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}}$$

with a complete proof π_3 that we construct as follows:

First, notice that hypothesis $\overline{\Delta_1} \models \neg C_0$ entails that for all $l_i \in C_0$, $l_i^\perp \in \overline{\Delta_1} \subseteq \Delta_0 \subseteq \mathcal{P} \subseteq \mathsf{Clo}_{\phi}(\Delta')$ and therefore $\Delta' \models_{\mathcal{T}} l_i^\perp$. Together with hypothesis $\phi, C \models_{\mathcal{T}} C_0 \vee l_{bj}$, we then have $\phi, C, \Delta', l_{bj}^\perp \models_{\mathcal{T}}$. Applying Theorem 32 on this provides a complete proof-tree of $\Delta', \phi', l_{bj}^\perp, C', \Gamma \vdash^{\mathcal{P}}$.

– If $l_{bj} \in \mathcal{P}$, the leaf is untouched, but is now mapped to $\overline{\Delta_1, l_{bj}}$.

– If $l_{bj}^\perp \in \mathcal{P}$, we close the whole branch altogether as follows: again, we have $\phi, C, \Delta', l_{bj}^\perp \models_{\mathcal{T}}$ but we now also have $l_{bj}^\perp \in \mathcal{P} \subseteq \mathsf{Clo}_{\phi}(\Delta')$ and therefore $\Delta' \models_{\mathcal{T}} l_{bj}^\perp$. Combining this together we have $\phi, C, \Delta' \models_{\mathcal{T}}$. Applying Theorem 32 on this provides a complete proof-tree of $\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}$.

- $\mathcal{T}$-Learn: $\Delta \| \phi \Rightarrow \Delta \| \phi, C$ if $C = \{l_1, \ldots, l_n\} \subseteq \mathsf{lit}(\phi, \Delta)$ and $\phi \models_{\mathcal{T}} C$.

Let π_1 be a incomplete proof-tree corresponding to $\Delta \| \phi$. We have to build π_2 that corresponds to $\Delta \| \phi, C$.

We extend π_1 into π_2 as follows: consider *every* open leaf; it is labelled by a sequent of the form $\Delta', \phi', \Gamma \vdash^{\mathcal{P}}$; let C' represent C; then replace the leaf by

$$\dfrac{\dfrac{\dfrac{\pi_3}{\Delta', l_1^\perp, \ldots, l_n^\perp, \phi', \Gamma \vdash^{\mathcal{P}; l_1^\perp, \ldots, l_n^\perp}}}{\Delta', \phi', \Gamma \vdash^{\mathcal{P}} C'} \qquad \dfrac{\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}; C'}}{\Delta', \phi', \Gamma \vdash^{\mathcal{P}} {C'}^\perp}}{\Delta', \phi', \Gamma \vdash^{\mathcal{P}}}\ \mathsf{cut\ on}\ C'$$

Again, π_3 is provided by applying Theorem 32 on the hypothesis $\phi \models_{\mathcal{T}} C$, i.e. $\phi, C^\perp \models_{\mathcal{T}}$: we get a complete proof-tree of $\Delta', \phi', {C'}^\perp, \Gamma \vdash^{\mathcal{P}, l_1^\perp, \ldots, l_n^\perp}$.

- $\mathcal{T}$-Forget: $\Delta\|\phi, C \Rightarrow \Delta\|\phi$ if $\phi \models_{\mathcal{T}} C$.
  Let π_1 be an incomplete proof-tree corresponding to $\Delta\|\phi$. We take $\pi_2 := \pi_1$. It corresponds to $\Delta\|\phi$ since, in a sequent $\Delta', \phi', C', \Gamma \vdash^{\mathcal{P}}$ labelling an open leaf, we can consider C', Γ as the new Γ (knowing that $\phi \models_{\mathcal{T}} C$).
- Restart: $\Delta\|\phi \Rightarrow \emptyset\|\phi$
  Similarly, take $\pi_2 := \pi_1$.

□

At this point, we should look into defining a proof-search strategy identifying those complete proof-trees that are the images of $\mathsf{DPLL}_{bj}(\mathcal{T})$ runs, just as we did with Elementary $\mathsf{DPLL}(\mathcal{T})$. This is left for future work. This seems however quite tricky since we have relaxed our notion of correspondence.

Also notice that, in Theorem 53, we have not mentioned quantitative information about the simulation. This is because, while simulating just one step of $\mathsf{DPLL}_{bj}(\mathcal{T})$, we now need to extend (possibly) several open branches. To be fair, the steps are identical for every open leaf that we extend. So this simultaneous extension prevents us from refining the theorem with quantitative information (unless we found a sophisticated way of counting the size of proof-trees, using sharing), but it is clear that an implementation of the simulation would clearly not perform those steps several times.

In other words, we introduced the Elementary $\mathsf{DPLL}(\mathcal{T})$ system in order to get the tight results from Sections 4.4 and 4.4.2, which we do not easily get for the full Abstract $\mathsf{DPLL}(\mathcal{T})$ system of [NOT06].

As mentioned in the introduction, the sequent calculus can be seen as a system that both *defines* a logic and *specifies* a proof-search procedure for it. That view, however, is somewhat weakened when cuts are allowed in the proof-search (and in the simulation of $\mathsf{DPLL}(\mathcal{T})$, they are), since they arguably do not participate to the definition of the logic.

Nevertheless, it would be impossible to have a polynomial simulation of $\mathsf{DPLL}(\mathcal{T})$ without using cuts: some tautologies classes have polynomially-sized proofs when cuts are allowed but only exponentially-sized proofs when they are not (e.g. the "Statman tautologies" [Sta78]); as $\mathsf{DPLL}(\mathcal{T})$ uses the Decide rule, it has access to those short proofs with cuts, and it would be hopeless to try and simulate it in a cut-free calculus without risking an exponential growth of the proof-search run compared to the $\mathsf{DPLL}(\mathcal{T})$ run.

Now the use of cuts in proof-search does not entirely defeat the point of using sequent calculus to specify a goal-directed proof-search procedure.

Firstly, allowing cuts in proof-search can be controlled by a single flag to be toggled on and off without jeopardising completeness; and the bottom-up application of the other rules (especially with focusing) is more syntax-directed than in other systems (e.g. natural deduction) with which soundness of automated reasoning techniques could also be proved.

In the case of $\mathsf{DPLL}(\mathcal{T})$, even the use of cut can be tightly controlled, as the only instances that are needed are analytic: the only cut-formulae are the literals present in the original sequent, in finite numbers. This actually means that, as an alternative to using cuts, we could equivalently add, to the sequent to prove, as many instances of the Law of Excluded Middle $l \vee l^{\perp}$ as there are literals l in the problem, and run the proof-search process specified by the cut-free calculus.

Finally, sequent calculus is also appealing for the shape of its incomplete proof-trees, which offers promising ways of making different techniques collaborate: after running $\mathsf{DPLL}(\mathcal{T})$ for a certain number of steps, the open leaves of the partial proof-tree that has been built can be exploited and taken over by other proof-search techniques that may or may not use cuts. Given an inference system where the soundness of different techniques can be proved, it is not necessarily the case that the incomplete proof-trees of that system offer such an interface for collaboration.

Chapter 5

Simulating clause and connection tableaux in the sequent calculus

Another area of automated reasoning is that of *tableaux calculi* (see e.g. [RV01]). Different styles of tableaux calculi exist; one of the most fundamental ones is that of *analytic tableaux*. These can be seen as a reformulation of the sequent calculus for the specific purpose of proof-searching (rather than defining the proofs for a logic); in that respect, analytic tableaux and sequent calculus proofs mostly differ in the way trees are written and in whether or not existential variables (a.k.a meta-variables) are explicitly handled.

This chapter therefore tackles another kind of tableaux method, whose connection to the sequent calculus and its proof-search methods is less obvious: *clause tableaux*, and some interesting restrictions of it known as *weak connection tableaux* and *strong connection tableaux*. Connection tableaux impose a connectivity condition on clause tableaux, ensuring a real goal-directed nature of the proof-search mechanisms.

In this chapter, we show that these tableaux can be seen as proof-constructs in our focused sequent calculus $\mathsf{LK}^p(\mathcal{T})$, making again an interesting use of polarities. As the name "clause tableaux" suggests, the tableaux calculi tackled in this chapter use the data-structures of clauses and literals, just like the SMT-techniques tackled in Chapter 4 did. This allows us to re-use some of the material from Chapter 4 and draw interesting parallels between the two kinds of automated reasoning techniques treated in this thesis (similar connections are developed in [Tin07]).

Therefore, this chapter is organised as follows: Section 5.1 recalls the standard presentation of clause tableaux and connection tableaux for propositional logic, as well as an interesting generalisation [Tin07]: *clause tableaux modulo theories*; Section 5.2 shows the simulation of clause tableaux (modulo theories) in our sequent calculus, Section 5.3 presents the simulation of weak connection tableaux; Section 5.4 presents the simulation of strong connection tableaux; and finally in Section 5.5, we lift the simulation of the weak and strong connection tableaux to pure first-order logic.

5.1 Clause and connection tableaux

In this section, we recall the definitions for clause tableaux and connection tableaux.

Definition 53 (Literals, Clauses) Literals and Clauses are respectively defined as in Definitions 24 and 25 in Chapter 4. ※

DEFINITION 54 (Clause tableaux) A *clause tableau* for a given set S of clauses is a finitely branching tree:

- whose root is labelled by S.
- whose other nodes are labelled by literals and
- that is obtained by an inductive construction defined by the following two rules:

  Initial rule: a single root node (labelled by S) is a clause tableau for (S).

  Expansion rule: let T be a tableau for S and $S\|\Gamma$ denote one of its branches where Γ is the sequence of literals on the branch. Let $C = l_1 \vee \cdots \vee l_n$ be a clause of S. The tree consisting of expanding that branch by adding new leaves labelled with $l_1 \cdots l_n$ as shown in Figure 5.1a, is a clause tableau for (S). That expansion can be represented by the rule below:

$$\frac{S\|\Gamma}{l_1 \ldots l_n}$$

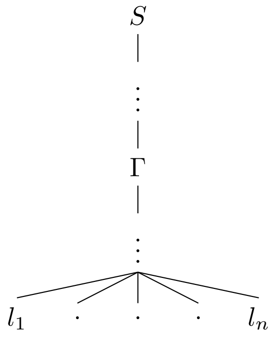
  The clause C used for the expansion is called the *expansion clause*.

※

DEFINITION 55 (Closed branch, closed tableau) A *literal occurs in a tableau* if it labels any of its non-root nodes. A branch is *closed* if l and $l^{\perp}$ occur in the branch. The fact that a branch is closed can be represented by a tag $\star$ below the leaf of that branch. The tagging operation can be represented by the following *closing rule*:

$$\frac{S\|\Gamma, l, \Gamma', l^{\perp}, \Gamma''}{\star}$$

and by the tree presented in Figure 5.1b.
A tableau is *closed*, whenever every branch in it is closed. ※


(a) Expansion Branch

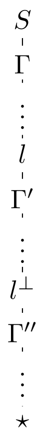

(b) Closed Branch

Figure 5.1: Clause Tableau

THEOREM 54 (Soundness and Completeness) [RV01]
A set of clauses is unsatisfiable if and only if there exists a closed clause tableau for it. ※

Connection Tableau

Selecting a clause for the expansion rule can be tricky in a tableau construction, since there is a lot of freedom in the choice, making the search space broad. The search space can be narrowed in a complete way by restricting the choice of clause for an expansion rule.

The first way to enforce such a restriction is a *weak connection*:

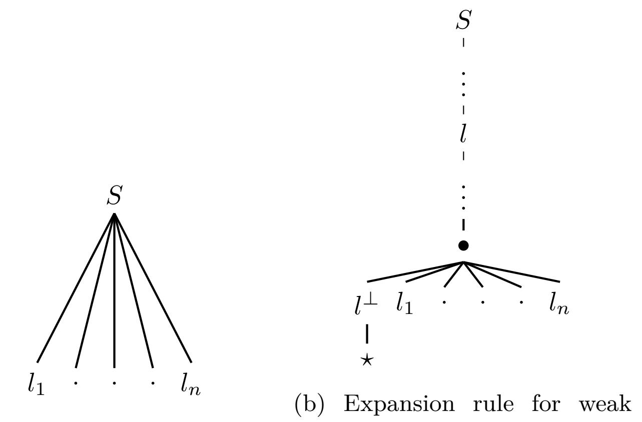
Definition 56 (Weak Connection Tableau)

Initial rule: For any clause $l_1 \vee \cdots \vee l_n = C \in S$, the tree constructed by expanding the root node with n new subtrees with nodes l_i, as shown in Fig. 5.2a, is a weak connection tableau for S.

Expansion rule: Let T be a weak connection tableau, Γ be one of the branch of T ending with l not necessarily as leaf, and $C \in S$ where $C = l^\perp \vee l_1 \vee \cdots \vee l_n$. The tableau T' obtained from T by expanding the branch $S \| \Gamma$ using an expansion clause C such that $l^\perp \in C$, is a weak connection tableau. This weakly connected expansion can be represented by the following rule:

$$\frac{S, C \| \Gamma, l, \Gamma'}{l^\perp, l_1, \cdots, l_n}$$

※



(a) Initial expansion (b) Expansion rule for weak connected tableaux

Figure 5.2: Weak connection tableau

Theorem 55 (Soundness and Completeness) [RV01]

A set of clauses is unsatisfiable if and only if there exists a closed weak connection tableau for it.[1] ※

The weak connection restriction can even be strengthened to define a complete notion of tableaux: *strong connection tableaux.*

Definition 57 (Strong Connection Tableau)

Initial rule: This rule is the same as that for strong connection tableau (Definition 56).

Expansion rule: Let T be a strong connection tableau, Γ be one of the branches of T ending with l and $C \in S$ where $C = l^\perp \vee l_1 \vee \cdots \vee l_n$. The tableau T' obtained from T by expanding the branch $S \| \Gamma$ using an expansion clause C such that $l^\perp \in C$, as shown in Fig. 5.3b, is a strong connection tableau. This strongly connected expansion can be represented by the following rule:

$$\frac{S, C \| \Gamma, l}{l^\perp, l_1, \cdots, l_n}$$

※

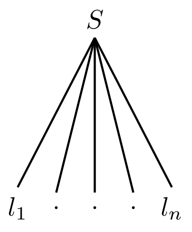
Theorem 56 (Soundness and Completeness) [RV01]

A set of clauses is unsatisfiable if and only if there exists a closed strong connection tableau for it.[2] ※

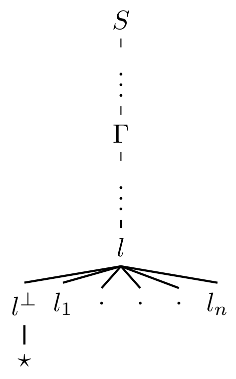
In both strong and weak connection tableaux, the literal of the expansion clause that is used for immediate closure is called the *active literal.*

In Example 8, we illustrate the notions of weak and strong connection tableaux:

[1]Being *closed* is the notion defined for clause tableaux in general (Definition 55).
[2]Again, being *closed* is the notion defined for clause tableaux in general (Definition 55).



(a) Initial expansion



(b) Expansion rule for strongly connected tableaux

Figure 5.3: Strong connection tableau

Example 8 Let $S = 1^\perp \vee 2, 2^\perp \vee 1, 4^\perp \vee 1, 4^\perp \vee 3, 3^\perp, 4$ and $S' = 1 \vee 2^\perp, 3^\perp \vee 2^\perp, 1^\perp \vee 3, 2$. Connection tableaux for S and S' are presented in Figure 5.4, where the closing tags for weak connections are denoted $\star$, those for strong connections are denoted $\star$ and every other closing tag is denoted $\star$.

Notice in Fig. 5.4a that, after having made the expansion with clause $4^\perp \vee 1$, we would never be able to complete the tableau as a strong connection tableau: strong connections would impose the right-most branch to enter a loop of 1 and 2. So we would need to detect the loop and backtrack, so as to produce Fig. 5.4b. Instead, we could just try to build a weak connection tableau, and then as we show in Fig. 5.4a we can exit the loop by expanding on clause $4^\perp \vee 3$, with a weak connection on 4 (denoted by the blue star), and finish. With weak connections we never have to backtrack (except maybe on the initial clause).

Fig. 5.4c shows how a strong connection tableau can still be build with one of the branches being closed by a regular closing tag (the black star).

Clause tableaux modulo theories

In [Tin07], Tinelli presented a notion of *Clause tableaux modulo theories* as an extension of Clause tableaux that can call a decision procedure as in SMT-solving. The construction of Clause tableaux, Definition 54 is the same as for Clause tableaux modulo theories, but the definition of a closed branch (and therefore that of a closed tableaux) is changed, so as to call the decision procedure:

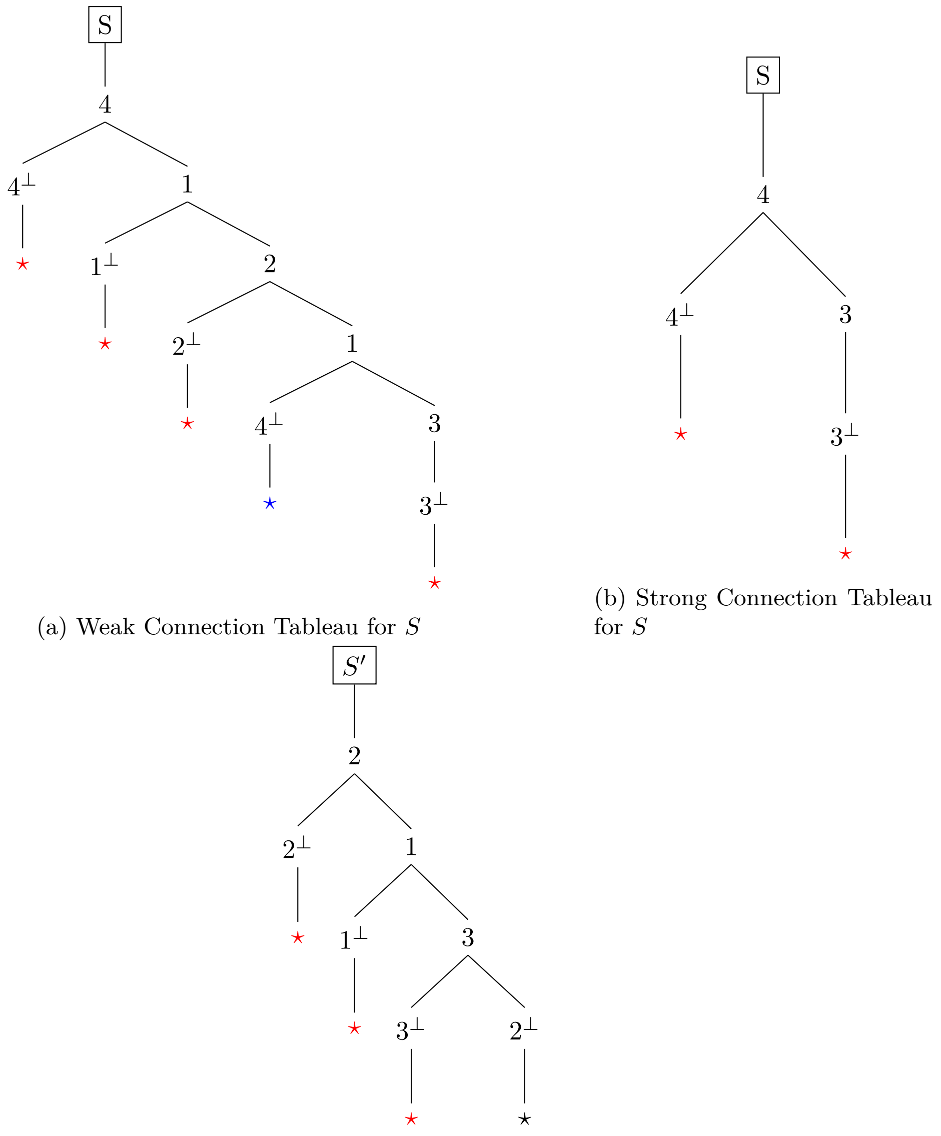
Definition 58 (Closed branch, closed tableau) A branch $S\|\Gamma$ is *closed* if Γ is $\mathcal{T}$-inconsistent ($\Gamma \models_{\mathcal{T}}$), which can be represented by the following closing rule:

$$\frac{S\|\Gamma}{\star}\ \Gamma \text{ is } \mathcal{T}\text{-inconsistent}$$

A tableau is *closed*, whenever every branch in it is $\mathcal{T}$-inconsistent. ※

Clause tableaux modulo theories are sound and complete for propositional logic modulo theories [Tin07].

It is tempting to try and restrict clause tableaux modulo theories using the concepts of weak and strong connections. Obvious candidates for such restrictions would be a notion of *weak connection tableaux modulo theories*, obtained by combining Definition 56 (for the construction of tableaux) and Definition 58 (for the closedness condition), and a notion of *strong connection tableaux modulo theories*, obtained by combining Definition 57 (for the construction of tableaux) and Definition 58. Since we do not know of any completeness properties for these candidates, we keep these as an



(a) Weak Connection Tableau for S

(b) Strong Connection Tableau for S

(c) Strong Connection Tableau for S'

Figure 5.4: Connection Tableaux

open research topics, and now turn to the simulation of the above tableaux calculi in our focused sequent calculus.

5.2 Simulation of clause tableaux (modulo theories)

In this section we simulate clause tableaux modulo theories in our sequent calculus. This simulation will hold in particular for the empty theory, therefore providing a simulation of clause tableaux in sequent calculus.

The exact system that we use for our simulations is the propositional fragment of the $\mathsf{LK}^p(\mathcal{T})$ system from Chapter 3, extended with the rule (DPol) that we have shown to be admissible (and invertible) in $\mathsf{LK}^p(\mathcal{T})$ ((DPol) is a restricted case of inverted (Pol)). The resulting system, which is explicitly given in Fig 5.5, is therefore strictly equivalent to (propositional) $\mathsf{LK}^p(\mathcal{T})$ as far as complete proofs are concerned. But as we will formalise our simulation results in terms of incomplete proofs, having those admissible rules explicitly in the system yields tighter results.

DEFINITION 59 (System $\mathsf{LK}^{pd}(\mathcal{T})$)

The sequent calculus $\mathit{LK}^{pd}(\mathcal{T})$ is an extension of propositional $\mathsf{LK}^{p}(\mathcal{T})$, given by the rules of Figure 5.5. The admissible/invertible rule allows the removal of polarity on literals. ※

Synchronous rules

$$(\wedge^+)\frac{\Gamma \vdash^{\mathcal{P}} [A] \qquad \Gamma \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]} \qquad (\vee^+)\frac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]}$$

$$(\mathsf{Init}_1)\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma), l^\perp \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}} [l]}\ l \text{ is } \mathcal{P}\text{-positive} \qquad (\top^+)\frac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]}$$

$$(\mathsf{Release})\frac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]}\ N \text{ is not } \mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \qquad \Gamma \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash A \wedge^- B, \Delta} \qquad (\vee^-)\frac{\Gamma \vdash^{\mathcal{P}} A_1, A_2, \Delta}{\Gamma \vdash^{\mathcal{P}} A_1 \vee^- A_2, \Delta} \qquad (\top^-)\frac{}{\Gamma \vdash^{\mathcal{P}} \Delta, \top^-}$$

$$(\mathsf{Store})\frac{\Gamma, A^\perp \vdash^{\mathcal{P};A^\perp} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}\ A \text{ is } \mathcal{P}\text{-positive or literal} \qquad (\perp^-)\frac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta, \perp^-}$$

Structural rules

$$(\mathsf{Select})\frac{\Gamma, P^\perp \vdash^{\mathcal{P}} [P]}{\Gamma, P^\perp \vdash^{\mathcal{P}}}\ P \text{ is not } \mathcal{P}\text{-negative} \qquad (\mathsf{Init}_2)\frac{\mathsf{lit}_{\mathcal{P}}(\Gamma) \models_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}}$$

Admissible/Invertible rule

$$(\mathsf{DPol})\frac{\Gamma \vdash^{\mathcal{P}}}{\Gamma \vdash^{\mathcal{P},l}}\ l \in \Gamma$$

Figure 5.5: System $\mathsf{LK}^{pd}(\mathcal{T})$

We also prove this small lemma that will be useful for our simulations.

LEMMA 57 (Init_3)

The following rule is admissible and invertible in $\mathsf{LK}^{p}(\mathcal{T})$:

$$(\mathsf{Init}_3)\frac{}{\Gamma \vdash^{\emptyset}}\ \mathsf{lit}_{\mathcal{L}}(\Gamma) \models_{\mathcal{T}}$$

※

Proof: Invertibility is trivial as there is no premise.
Admissibility can be proved trivially (by Id_2) if there are both l and $l^\perp$ in Γ for some literal l.
Otherwise, let $\mathsf{lit}_{\mathcal{L}}(\Gamma) = \{l_1, \ldots, l_n\}$, and for every i in $0 \ldots n-1$, we construct :

$$\text{(Select)}\dfrac{\text{(Release)}\dfrac{\text{(Store)}\dfrac{\Gamma, l_1, \ldots, l_{i+1} \vdash^{\mathcal{P}, l_1, \ldots, l_{i+1}}}{\Gamma, l_1, \ldots, l_i \vdash^{\mathcal{P}, l_1, \ldots, l_i} {l_{i+1}}^{\perp}}}{\Gamma, l_1, \ldots, l_i \vdash^{\mathcal{P}, l_1, \ldots, l_i} [{l_{i+1}}^{\perp}]}}{\Gamma, l_1, \ldots, l_i \vdash^{\mathcal{P}, l_1, \ldots, l_i}}$$

Finally, we close $\Gamma, \mathsf{lit}_{\mathcal{L}}(\Gamma) \vdash^{\mathsf{lit}_{\mathcal{L}}(\Gamma)}$ with Init_2. □

We now turn to the simulation itself. In that simulation, a closed clause tableaux modulo theories, for a set of clauses S, corresponds to a complete proof-tree of $S' \vdash$ (where S' represents S in the sense of Definition 47 of Chapter 4). If it is not closed, it corresponds to an incomplete proof-tree.

Definition 60 (Encoding clause tableaux modulo theories in $\mathsf{LK}^{pd}(\mathcal{T})$)

For every set S of clauses, and for every set S' of formulae representing S (in the sense of Definition 47), we define a mapping Φ from the set of clause tableaux modulo theories for S to the set of the incomplete proof-trees concluding $S' \vdash$ such that the following property holds: There is a bijection f from the (open) branches of a tableau T to the (open) leaves of $\Phi(T)$ such that, for all open branches $S\|\Gamma$ of T, $f(S\|\Gamma)$ is labelled with the sequent $S', \Gamma \vdash^{\emptyset}$.

Initial rule: If T is just the root node labelled by S, then we define $\Phi(T)$ as the one-node proof-tree $S' \vdash^{\emptyset}$.

Expansion rule: If T' is a clause tableau modulo theories obtained by expanding the branch $S\|\Gamma$ of a tableau T, using an expansion clause $C \in S$, we define $\Phi(T')$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{(DPol)}\dfrac{S', \Gamma, l_1 \vdash^{\emptyset}}{S', \Gamma, l_1 \vdash^{l_1}}}{S', \Gamma \vdash^{\emptyset} {l_1}^{\perp}}}{S', \Gamma \vdash^{\emptyset} [{l_1}^{\perp}]} \qquad \cdots \qquad \dfrac{\dfrac{\text{(DPol)}\dfrac{S', l_n \vdash^{\emptyset}}{S', l_n \vdash^{l_n}}}{S' \vdash^{\emptyset} {l_n}^{\perp}}}{S' \vdash^{\emptyset} [{l_n}^{\perp}]}}{S', \Gamma \vdash^{\emptyset} [{C'}^{\perp}]}}{S', \Gamma \vdash^{\emptyset}}$$

Closing rule: If a branch $S\|\Gamma$ is $\mathcal{T}$-inconsistent, then we define $\Phi(T)$ as follows :

$$(\mathsf{Init}_3)\dfrac{}{S', \Gamma \vdash^{\emptyset}}\ \Gamma \models_{\mathcal{T}}$$

※

Definition 61 (Tableau-like incomplete proof-tree)

An incomplete proof-tree is S'-*tableau-like* if

- its conclusion is $S' \vdash^{\emptyset}$,
- its leaves are all labelled with empty polarisation sets,
- every occurrence of rule (Store) is below an occurrence of rule (DPol), and
- every occurrence of rule (Select) is placing the focus on the negation of an element of S'.

※

Theorem 58 (Characterisation of clause tableaux modulo theories)

Given a set S of clauses and a set S' of formulae representing S,

1. an incomplete proof-tree is S'-tableau-like if and only if it is the image by Φ of a clause tableaux modulo theories for S;
2. if this is the case, then the clause tableaux is closed if and only if the proof-tree can be completed by closing every leaf with (Init_3).

※

Proof:

- Notice that the incomplete proof-tree constructed in Definition 60 are all tableau-like.

- Given a tableau-like incomplete proof-tree, every occurrence of rule (Select) corresponds to an expansion on the clause represented by the negation of the formula placed in focus.

□

5.3 Simulation of weak connection tableaux

In this section we refine our simulation of clause tableaux modulo theories specifically for the case

- of the empty theory
- and of weak connection tableaux.

In particular, we show how weak connectivity can be expressed in terms of polarities, provided we equip the sequent calculus with the polarisation rule (Pol).

We now explicitly give the system used for the simulation of weak connection tableaux (with no theories).

DEFINITION 62 (System LK^{pdp}) The sequent calculus LK^{pdp} is given by the rules of Figure 5.6. The admissible/invertible rules allow both the addition and removal of polarities on literals. ※

Synchronous rules

$$(\wedge^+)\frac{\Gamma \vdash^{\mathcal{P}} [A] \qquad \Gamma \vdash^{\mathcal{P}} [B]}{\Gamma \vdash^{\mathcal{P}} [A \wedge^+ B]} \qquad (\vee^+)\frac{\Gamma \vdash^{\mathcal{P}} [A_i]}{\Gamma \vdash^{\mathcal{P}} [A_1 \vee^+ A_2]}$$

$$(\mathsf{Init}_1)\frac{l \in \Gamma}{\Gamma \vdash^{\mathcal{P}} [l]}\ l \text{ is } \mathcal{P}\text{-positive} \qquad (\top^+)\frac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]}$$

$$(\mathsf{Release})\frac{\Gamma \vdash^{\mathcal{P}} N}{\Gamma \vdash^{\mathcal{P}} [N]}\ N \text{ is not } \mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\frac{\Gamma \vdash^{\mathcal{P}} A, \Delta \qquad \Gamma \vdash^{\mathcal{P}} B, \Delta}{\Gamma \vdash A \wedge^- B, \Delta} \qquad (\vee^-)\frac{\Gamma \vdash^{\mathcal{P}} A_1, A_2, \Delta}{\Gamma \vdash^{\mathcal{P}} A_1 \vee^- A_2, \Delta} \qquad (\top^-)\frac{}{\Gamma \vdash^{\mathcal{P}} \Delta, \top^-}$$

$$(\mathsf{Store})\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P};A^{\perp}} \Delta}{\Gamma \vdash^{\mathcal{P}} A, \Delta}\ A \text{ is } \mathcal{P}\text{-positive or literal} \qquad (\perp^-)\frac{\Gamma \vdash^{\mathcal{P}} \Delta}{\Gamma \vdash^{\mathcal{P}} \Delta, \perp^-}$$

Structural rules

$$(\mathsf{Select})\frac{\Gamma, P^{\perp} \vdash^{\mathcal{P}} [P]}{\Gamma, P^{\perp} \vdash^{\mathcal{P}}}\ P \text{ is not } \mathcal{P}\text{-negative}$$

Admissible/Invertible rules

$$(\mathsf{Pol})\frac{\Gamma \vdash^{\mathcal{P},l}}{\Gamma \vdash^{\mathcal{P}}}\ l \in \Gamma \qquad (\mathsf{DPol})\frac{\Gamma \vdash^{\mathcal{P}}}{\Gamma \vdash^{\mathcal{P},l}}\ l \in \Gamma$$

Figure 5.6: System LK^{pdp}

The (Pol) rule is the inverse of the (DPol) rule, both admissible and invertible in $\mathsf{LK}^{p}(\emptyset)$. Notice that the rules of the system are simplified for the empty theory: in particular, the condition for (Init_1) is simplified and rule (Init_2) never applies (so we removed it from the system).

We now turn to the simulation itself.

Definition 63 (Encoding weak connection tableaux in LK^{pdp})

For every set S of clauses, and for every set S' of formulae representing S, we define a mapping ϕ from the set of weak connection tableaux for S to the set of the incomplete proof-trees concluding $S' \vdash$ such that the following property holds:

There is a bijection f from the (open) branches of a tableau T to the (open) leaves of $\phi(T)$ such that, for all open branches $S\|\Gamma$ of T, $f(S\|\Gamma)$ is labelled with the sequent $S', \Gamma \vdash^{\emptyset}$.

We define $\phi(T)$ by induction on the inductive construction of T, checking that the above property is indeed an invariant of the induction.

Initial rule: If $l_1 \vee \ldots \vee l_n = C \in S$ and T is the form of Fig. 5.2a, then we define $\phi(T)$ as :

$$\dfrac{\dfrac{\dfrac{\dfrac{S', l_1 \vdash}{S' \vdash l_1{}^{\perp}}}{S' \vdash [l_1{}^{\perp}]} \quad \cdots \quad \dfrac{\dfrac{S', l_n \vdash}{S' \vdash l_n{}^{\perp}}}{S' \vdash [l_n{}^{\perp}]}}{S' \vdash [C'^{\perp}]}}{S' \vdash}$$

Expansion rule: If T' is a weak connection tableau obtained by expanding the branch $S\|\Gamma$ of a tableau T using an expansion clause $C \in S$ (as illustrated in Figure 5.2b), we define $\phi(T')$ as follows:

we replace in $\phi(T)$ the open leaf corresponding to $S\|\Gamma$ by the following incomplete proof tree:

$$(\mathsf{Pol})\dfrac{\dfrac{\dfrac{\dfrac{}{S', \Gamma \vdash^{l^{\perp}} [l^{\perp}]} \quad \dfrac{\dfrac{(\mathsf{DPol})\dfrac{S', \Gamma, l_1 \vdash}{S', \Gamma, l_1 \vdash^{l^{\perp}, l_1^{\perp}}}}{S', \Gamma \vdash^{l^{\perp}} l_1{}^{\perp}}}{S', \Gamma \vdash^{l^{\perp}} [l_1{}^{\perp}]} \quad \cdots \quad \dfrac{\dfrac{(\mathsf{DPol})\dfrac{S', \Gamma, l_n \vdash}{S', \Gamma, l_n \vdash^{l^{\perp}, l_n^{\perp}}}}{S', \Gamma \vdash^{l^{\perp}} l_n{}^{\perp}}}{S', \Gamma \vdash^{l^{\perp}} [l_n{}^{\perp}]}}{S', \Gamma \vdash^{l^{\perp}} [C'^{\perp}]}}{S', \Gamma \vdash^{l^{\perp}}}}{S', \Gamma \vdash}$$

Closing rule: To simulate the closing of a branch (as illustrated in Fig. 5.1b), we can complete the branch of the proof-tree with the following rule:[3]

$$(\mathsf{Id}_2)\dfrac{}{S', \Gamma, l, \Gamma', l^{\perp}, \Gamma'' \vdash}$$

※

Definition 64 (Weak-tableau-like incomplete proof-tree)

An incomplete proof-tree is S'-*weak-tableau-like* if

- its conclusion is $S' \vdash^{\emptyset}$,
- its leaves are all labelled with empty polarisation sets,
- every occurrence of rule (Pol) has $\mathcal{P} = \emptyset$ and is below an occurrence of rule (Select), and
- in every occurrence of rule (Select), the focus is the negation of an element of S' and it is a conjunction of literals such that, unless the rule is the last rule of the tree (deriving its conclusion), one of these literals is positive.

※

[3]As the notion of closed branch for weak connection tableaux is the same as that for clause tableaux (for the empty theory), this completion of the proof-tree to simulate the closing operation is exactly the same as that which we used for clause tableaux: the proof-tree abbreviated by (Init_3) is that abbreviated by (Id_2) in the particular case of the empty theory.

THEOREM 59 (Characterisation of weak connection tableaux)
Given a set S of clauses and a set S' of formulae representing S,

1. an incomplete proof-tree is S'-weak-tableau-like if and only if it is the image by ϕ of a weak connection tableaux for S;
2. if this is the case, then the weak connection tableaux is closed if and only if the proof-tree can be completed by closing every leaf with (Id_2).

※

Proof:

- Notice that the incomplete proof-tree constructed in Definition 63 are all weak-tableau-like.
- Given a weak-tableau-like incomplete proof-tree, every occurrence of rule (Select) occurs in the compound proof-tree that simulates an expansion on the clause represented by the negation of the formula placed in focus.

□

5.4 Simulation of strong connection tableaux

We now investigate whether and how strong connections can be expressed in terms of polarities. Again, in the empty theory.

The fundamental remark is that the definition of strong connection tableaux relies on the fact that a branch of a tableau is an ordered data-structure: the latest literal of the branch is easily identified. In our sequent calculus, however, the antecedent of a sequent is a multiset, and therefore the order in which elements have been added in it, is not an information that the data-structure provides.

Coincidently, the identification of the latest introduced literal is exactly what is provided by the sequent structure of the $\mathsf{LK}^+(\mathcal{T})$ calculus from Section 3.6 of Chapter 3.

In this section we therefore simulate strong connection tableaux in (the propositional fragment of) $\mathsf{LK}^+(\emptyset)$ (the theory being still empty in our case), or more precisely an small extension of it, just like we extended (propositional) $\mathsf{LK}^p(\mathcal{T})$ into LK^{pdp} (with the admissible/invertible rules (Pol) and (DPol)) to simulate weak connection tableaux.

We therefore need to adapt rules (Pol) and (DPol) to the framework of $\mathsf{LK}^+(\mathcal{T})$ (Definition 18), proving their admissibility and invertibility.

LEMMA 60 (Adding and removing polarities)
The following rules are admissible and invertible in $\mathsf{LK}^+(\mathcal{T})$:

$$(\mathsf{Pol})\frac{\Gamma \vdash^{\mathcal{P},l} [\bullet]\Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta}\ l \in \Gamma, \Delta^{\perp} \qquad (\mathsf{DPol})\frac{\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta}{\Gamma \vdash^{\mathcal{P},l} [\bullet]\Delta}\ l \in \Gamma, \Delta^{\perp}$$

※

Proof:
Lemma 25.2 on $\Gamma \vdash^{\mathcal{P},l} [\bullet]\Delta$ gives $\Gamma \vdash^{\mathcal{P},l} \Delta$ in $\mathsf{LK}^p(\mathcal{T})$ and by admissibility of (Pol) in $\mathsf{LK}^p(\mathcal{T})$ we get $\Gamma \vdash^{\mathcal{P}} \Delta$ and finally $\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta$.

Lemma 25.2 on $\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta$ gives $\Gamma \vdash^{\mathcal{P}} \Delta$ in $\mathsf{LK}^p(\mathcal{T})$ and by invertibility of (Pol) in $\mathsf{LK}^p(\mathcal{T})$ we get $\Gamma \vdash^{\mathcal{P},l} \Delta$ and finally $\Gamma \vdash^{\mathcal{P},l} [\bullet]\Delta$. □

DEFINITION 65 (System LK^{pdp+}) The rules of LK^{pdp+}, given explicitly in Figure 5.7, extends $\mathsf{LK}^+(\emptyset)$ with the admissible/invertible rules (Pol) and (DPol) that allow both the addition and removal of polarities on literals. ※

We now turn to the simulation itself.

Synchronous rules

$$(\wedge^+)\frac{\Gamma \vdash^{\mathcal{P}} [A]\Delta \qquad \Gamma \vdash^{\mathcal{P}} [B]\Delta}{\Gamma \vdash^{\mathcal{P}} [A\wedge^+ B]\Delta} \qquad (\vee^+)\frac{\Gamma \vdash^{\mathcal{P}} [A_i]\Delta}{\Gamma \vdash^{\mathcal{P}} [A_1\vee^+ A_2]\Delta}$$

$$(\mathsf{Init}_1)\frac{l \in \Gamma \text{ or } l^{\perp} \in \Delta}{\Gamma \vdash^{\mathcal{P},l} [l]\Delta}\ l \text{ is } \mathcal{P}\text{-positive} \qquad (\top^+)\frac{}{\Gamma \vdash^{\mathcal{P}} [\top^+]}$$

$$(\mathsf{Release})\frac{\Gamma \vdash^{\mathcal{P}} [\bullet]N}{\Gamma \vdash^{\mathcal{P}} [N]}\ N \text{ is not } \mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\frac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A,\Delta \qquad \Gamma \vdash^{\mathcal{P}} [\mathcal{X}]B,\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A\wedge^- B,\Delta} \qquad (\vee^-)\frac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A_1, A_2,\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A_1\vee^- A_2,\Delta}$$

$$(\perp^-)\frac{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-,\Delta} \qquad (\top^-)\frac{}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]\perp^-,\Delta} \qquad (\mathsf{Store})\frac{\Gamma, A^{\perp} \vdash^{\mathcal{P};A^{\perp}} [\mathcal{X}]\Delta}{\Gamma \vdash^{\mathcal{P}} [\mathcal{X}]A,\Delta}\ A \text{ is } \mathcal{P}\text{-positive or literal}$$

Structural rules

$$(\mathsf{Select})\frac{\Gamma, P^{\perp} \vdash^{\mathcal{P}} [P]\Delta}{\Gamma, P^{\perp} \vdash^{\mathcal{P}} [\bullet]\Delta}\ P \text{ is not } \mathcal{P}\text{-negative}$$

Admissible/Invertible rules

$$(\mathsf{Pol})\frac{\Gamma \vdash^{\mathcal{P},l} [\bullet]\Delta}{\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta}\ l \in \Gamma, \Delta^{\perp} \qquad (\mathsf{DPol})\frac{\Gamma \vdash^{\mathcal{P}} [\bullet]\Delta}{\Gamma \vdash^{\mathcal{P},l} [\bullet]\Delta}\ l \in \Gamma, \Delta^{\perp}$$

Figure 5.7: System LK^{pdp+}

Definition 66 (Encoding strong connection tableaux in LK^{pdp+})

For every set S of clauses, and for every set S' of formulae representing S, we define a mapping ψ from the set of strong connection tableaux for S to the set of the incomplete proof-trees concluding $S' \vdash [\bullet]$ such that the following property holds:
There is a bijection f from the (open) branches of a tableau T to the (open) leaves of $\psi(T)$ such that, for all open branches $S\|\Gamma$ of T, $f(S\|\Gamma)$ is labelled with the sequent $S', \Gamma \vdash^{\emptyset} [\bullet]l$.
We define $\psi(T)$ by induction on the inductive construction of T, checking that the above property is indeed an invariant of the induction.
Initial rule: If $l_1 \vee \ldots \vee l_n = C \in S$ and T is the form of Figure 5.3a then we define $\psi(T)$ as :

$$\cfrac{\cfrac{\cfrac{S' \vdash^{\emptyset} [\bullet]{l_1}^{\perp}}{S' \vdash^{\emptyset} [{l_1}^{\perp}]} \qquad \cdots \qquad \cfrac{S' \vdash^{\emptyset} [\bullet]{l_n}^{\perp}}{S' \vdash^{\emptyset} [{l_n}^{\perp}]}}{S' \vdash^{\emptyset} [{C'}^{\perp}]}}{S' \vdash^{\emptyset} [\bullet]}$$

Expansion rule: If T' is a strong connection tableau obtained by expanding the branch $S\|\Gamma, l$ of a tableau T using an expansion clause $C \in S$ (as illustrated in Figure 5.3b), we define $\psi(T')$ as follows:
we replace in $\psi(T)$ the open leaf corresponding to $S\|\Gamma, l$ by the following incomplete tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{S',\Gamma\vdash^{l}[l]l^{\perp}}\qquad \dfrac{\dfrac{\dfrac{S',\Gamma,l\vdash[\bullet]{l_1}^{\perp}}{S',\Gamma,l\vdash^{l}[\bullet]{l_1}^{\perp}}}{S',\Gamma,l\vdash^{l}[{l_1}^{\perp}]}}{S',\Gamma\vdash^{l}[{l_1}^{\perp}]l^{\perp}}\quad\cdots\quad \dfrac{\dfrac{\dfrac{S',\Gamma,l\vdash[\bullet]{l_n}^{\perp}}{S',\Gamma,l\vdash^{l}[\bullet]{l_n}^{\perp}}}{S',\Gamma,l\vdash^{l}[{l_1}^{\perp}]}}{S',\Gamma\vdash^{l}[{l_n}^{\perp}]l^{\perp}}}{S',\Gamma\vdash^{l}[{C'}^{\perp}]l^{\perp}}}{S',\Gamma\vdash^{l}[\bullet]l^{\perp}}}{S',\Gamma\vdash[\bullet]l^{\perp}}$$

Closing rule: To simulate the closing of a branch (as illustrated in Fig. 5.1b), we can complete the branch of the proof-tree as follows:

$$(\mathsf{Store})\dfrac{(\mathsf{Id}_2)\dfrac{}{S',\Gamma,l,\Gamma_0,l_0\vdash^{l_0}[\bullet]}}{S',\Gamma,l,\Gamma_0\vdash[\bullet]l_0^{\perp}}$$

where $\Gamma_0, l_0 = \Gamma', l^{\perp}, \Gamma''$.

※

Definition 67 (Strong-tableau-like incomplete proof-tree)

An incomplete proof-tree is S'-*strong-tableau-like* if

- its conclusion is $S' \vdash^{\emptyset}$,
- its leaves are all labelled with empty polarisation sets and with non-empty right-hand sides,
- every occurrence of rule (Init_1) is of the form

$$\dfrac{}{\Gamma\vdash^{\mathcal{P},l}[l]l^{\perp}}$$

- every occurrence of rule (Pol) is of the form

$$\dfrac{\Gamma\vdash^{l}[\bullet]l^{\perp}}{\Gamma\vdash[\bullet]l^{\perp}}$$

- in every occurrence of rule (Select), the focus is the negation of an element of S' and it is a conjunction of literals such that, unless the rule is the last rule of the tree (deriving its conclusion), one of these literals is positive.

※

Theorem 61 (Characterisation of strong connection tableaux)

Given a set S of clauses and a set S' of formulae representing S,

1. an incomplete proof-tree is S'-strong-tableau-like if and only if it is the image by ψ of a strong connection tableaux for S;
2. if this is the case, then the strong connection tableaux is closed if and only if the proof-tree can be completed by closing every leaf with the combination of (Store) and (Id_2).

※

Proof:

- Notice that the incomplete proof-tree constructed in Definition 66 are all strong-tableau-like.
- Given a strong-tableau-like incomplete proof-tree, every occurrence of rule (Select) occurs in the compound proof-tree that simulates an expansion on the clause represented by the negation of the formula placed in focus.

□

5.5 Extending the simulations to pure first-order logic

So far we have simulated in our sequent calculus two general methodologies of automated reasoning: SMT-solving ($\mathsf{DPLL}(\mathcal{T})$) and clause/connection tableaux. However, as long as quantifiers are not considered, it is likely that $\mathsf{DPLL}(\mathcal{T})$-based techniques are more efficient than clause/connection tableaux, which become interesting mostly when quantifiers and unification are involved.

Our next step is therefore to lift our simulations of clause and connection tableaux to the framework of pure first-order logic, in the hope that the combination of this with our simulation of $\mathsf{DPLL}(\mathcal{T})$ provides new interesting ways to deal with a background theory in presence of quantifiers (or, equivalently, to deal with first-order logic in presence of a background theory).

For this, our sequent calculus $\mathsf{LK}^p(\mathcal{T})$ for first-order logic modulo theories, as presented in Chapter 3, is not exactly fit for our current purpose: We hit the main difference between tableaux and sequent calculus: whether or not existential variables (a.k.a meta-variables) are explicitly handled.

The meta-theory of $\mathsf{LK}^p(\mathcal{T})$ has been done in Chapter 3 without explicit meta-variables, and now for the purpose of proof-search, we need them. Therefore in this section we will propose a version of $\mathsf{LK}^p(\mathcal{T})$ with explicit meta-variables, but with an empty theory!, to lift our previous simulations of clause and connection tableaux to the framework of pure first-order logic.

The nature of this last section of the present dissertation is therefore more prospective and less formal than the previous sections and chapters, as the formal meta-theory of $\mathsf{LK}^p(\mathcal{T})$ with explicit meta-variables is still in its infancy.

Still, we start with some formal preliminaries that will allow us to present clause and connection tableaux for pure first-order logic and our version of $\mathsf{LK}^p(\mathcal{T})$ with explicit meta-variables.

5.5.1 Preliminaries

Definition 68 (Meta-terms) Consider a set of elements called variables and a set MV of elements called *meta-variables*. Let F_Σ be a first-order term signature. The set of *meta-terms* over signature F_Σ is defined by:

$$t, t_1, t_2, \ldots := x \mid X \mid f(t_1, \ldots, t_n)$$

with f ranging over F_Σ, X ranging over meta-variables and x ranging over variables. ※

Definition 69 (Literals) Let P_Σ be a first-order predicate signature equipped with an involutive and arity-preserving function called negation.
The set $\mathcal{L}$ of *literals* is given by the following grammar:

$$l, l_1, l_2 := P(t_1, \ldots, t_n)$$

with P ranging over P_Σ.
Negation on predicate symbol is extended to an involutive function of negation on literals:

$$(P(t_1, \ldots, t_n))^\perp := P^\perp(t_1, \ldots, t_n)$$

※

Definition 70 (Substitution)
A *substitution* is a partial function from meta-variables to terms (we can write $(\sigma(X) = t)$) satisfying the following condition:

$$\mathsf{Dom}(\sigma) \cap \mathsf{MV}(\sigma) = \emptyset$$

where we define $\mathsf{MV}(\sigma)$ as $\bigcup_{X \in \mathsf{Dom}(\sigma)} \mathsf{MV}(\sigma(X))$.
The empty substitution (i.e. the substitution σ such that $\mathsf{Dom}(\sigma) = \emptyset$) is denoted $\emptyset$.
Substitutions are extended to all terms (we can write $(\sigma(u) = t)$) as follows:

$$\begin{array}{lcl} \sigma(f(t_1, \ldots, t_n)) & = & f(\sigma(t_1, \ldots, t_n)) \\ \sigma(x) & = & x \\ \sigma(X) & = & X \text{ where } X \notin \mathsf{Dom}(\sigma) \end{array}$$

Substitutions are then extended to literals (we can write $(\sigma(l) = l')$) as follows:

$$\sigma(P(t_1, \dots, t_n)) = P(\sigma(t_1, \dots, t_n))$$

A *ground term* (resp. literal) is a term (resp. literal) that contains no meta-variables.
Let V be a set of meta-variables. A substitution σ such that, for all $X \in V$, $\sigma(X)$ is a ground term is called a *grounding substitution* for V.
A substitution σ that happens to be a (finite) permutation of MV is called a *renaming substitution.*
If σ is a substitution and Ω is a set or a list of meta-variables, $\sigma_{|\Omega}$ is the restriction of σ to Ω. ※

DEFINITION 71 (First-order unification) A substitution σ is called an *unifier* of the *unification problem* $t_1 = u_1, \dots, t_n = u_n$ if $\sigma(t_1) = \sigma(u_1), \dots, \sigma(t_n) = \sigma(u_n)$.
When a unification problem has a unifier, then it has a *most general unifier* σ: for any other unifier σ', there exists a substitution σ'' such that $\sigma' = \sigma'' \circ \sigma$ and $\mathsf{MV}(\sigma)$ only contains meta-variables from the unification problem.
We write $\sigma = \mathsf{mgu}(l_1, \dots, l_n)$ when σ is a most general unifier of the unification problem $l_1 = l_2 = \dots = l_n$.
Two literals l and l' are *renamings* of one another if $\mathsf{mgu}(l, l')$ exists and is a renaming substitution.
We write $\sigma'' = \mathsf{mgu}(\sigma, \sigma')$ if σ'' is the most general unifier of $t_1 = u_1, \dots, t_n = u_n$, where $(t_i, u_i)_i = (\sigma(X), \sigma'(X))_{X \in \mathsf{Dom}(\sigma) \cup \mathsf{Dom}(\sigma')}$. ※

5.5.2 Clause and connection tableaux in first-order logic

We now present the clause tableaux, and strong and weak connection tableaux for first-order logic.

DEFINITION 72 (Clause) A clause is a finite set of literals with no free meta-variables. We can see it as a disjunction of literals with every variable that is free in one of the literals as universally quantified (outside the clause). In that view, we will represent the clause $\{l_1, \dots, l_n\}$ as $\forall x_1 \dots \forall x_p (l_1 \vee \dots \vee l_n)$ where $\{x_1, \dots, x_p\} = \bigcup_{i=1}^{i=n} \mathsf{FV}(l_i)$. ※

Given the definition and notation above, we will avoid using the phrase "free variables" for a clause, which would be an ambiguous notion; nor will we discuss, for the same reason, any notion of α-equivalence between clauses (and therefore, any notion of equality between clauses).

DEFINITION 73 (New instance) Let $C = \forall x_1 \dots \forall x_p (l_1 \vee \dots \vee l_n)$ be a clause. Let $\mathcal{M}$ be a set of meta-variables. C' is a *new instance of C, fresh for* $\mathcal{M}$, if C' is the set of literals $\{\{X_1/x_1 \dots X_p/x_p\} l_i \mid l_i \in C\}$, for some pairwise distinct meta variables $X_1 \dots X_p$ such that $\{X_1, \dots, X_p\} \cap \mathcal{M} = \emptyset$. ※

REMARK 62 A clause has no meta-variables and a clause instance has no variables.

DEFINITION 74 (Clause tableau for first-order logic) A *clause tableau* for a given set S of clauses is a substitution and a finitely branching tree:

- whose root is labelled by S.
- whose other nodes are labelled by literals and whose branches are tagged as open or closed,
- such that the pair is obtained by an inductive construction defined by the following rules:

  Initial Rule: the empty substitution and a single open root node labelled by S is a clause tableau for S.

Expansion Rule: Let

- (σ, T) be a tableau for S.
- $S\|\Gamma$ be an open branch of T.
- $\{l_1, \ldots, l_n\}$ be a new instance of a clause of S that is fresh for $\mathsf{FMV}(T)$.
- Let T' be the tree obtained by expanding $S\|\Gamma$ with n new open leaves $l_1, \ldots, l_n$.

Then (σ, T') is a tableau for S.

Closing Rule: Let (σ, T) be a tableau for S and $S\|\Gamma$ be an open branch of T and $\sigma' = \mathsf{mgu}(l, l')$ for some literals l and $l'^{\perp}$ in Γ. If $\sigma'' = \mathsf{mgu}(\sigma, \sigma')$, then the pair (σ'', T') is a tableau for S where T' is T whose branch $S\|\Gamma$ is tagged as closed.

※

Notice that none of the literals labelled the (non-root) nodes of a clause tableau have any free variables.

Clearly, the third construction rule can be postponed, and we could have defined clause tableaux with only 1. and 2., not involving substitutions, and then said that a tableau is closed (or can be closed) if there is a substitution σ such that, on every branch $S\|\Gamma$ of the tableau, there are two literals l and l' such that σ is a unifier of $l = l'^{\perp}$.

DEFINITION 75 (Weak Connection Tableau)

Initial Rule: For any new instance $\{l_1 \ldots l_n\}$ of some clause in S, the tree constructed by expanding the root node with n new leaves, together with the empty substitution, is a weak connection tableau for S.

Expansion Rule: Let (σ, T) be a weak connection tableau for S, let $S\|\Gamma$ be one of the open branches of T and let $\{l_0, l_1 \ldots l_n\}$ be a new instance of some clause in S.
If there is a substitution $\sigma' = \mathsf{mgu}(l, l_i^{\perp})$ for some $i \in 1 \ldots n$ and some l in Γ, and if $\sigma'' = \mathsf{mgu}(\sigma, \sigma')$, then the pair (σ'', T') is a weak connection tableau for S, where T' is obtained from T by expanding its branch $S\|\Gamma$ with $\{l_0, l_1 \ldots l_n\}$, tagging the branch $S\|\Gamma, l_i$ as closed.

Closing Rule: As for clause tableaux (in Definition 74).

※

DEFINITION 76 (Strong Connection Tableau)

Initial Rule: For any new instance $\{l_1, \ldots, l_n\}$ of some clause in S, the tree constructed by expanding the root node with n new leaves, together with the empty substitution, is a strong connection tableau for S.

Expansion Rule: Let (σ, T) be a strong connection tableau for S, let $S\|\Gamma$ be one of the open branches of T ending with l and let $\{l_0, l_1 \ldots l_n\}$ be a new instance of some clause in S.
If there is a substitution $\sigma' = \mathsf{mgu}(l, l_i^{\perp})$ for some $i \in 1 \ldots n$ and if $\sigma'' = \mathsf{mgu}(\sigma, \sigma')$, then the pair (σ'', T') is a strong connection tableau for S, where T' is obtained from T by expanding its branch $S\|\Gamma$ with $\{l_0, l_1 \ldots l_n\}$, tagging the branch $S\|\Gamma, l_i$ as closed.

Closing Rule: As for clause tableaux (in Definition 74).

※

Clause tableaux, weak connection tableaux, and strong connection tableaux, are all sound and complete for first-order logic [RV01].

5.5.3 The $\mathsf{LK}_{\mathsf{F}}^{pd}$ system and the simulation of clause tableaux

We now introduce a variant of the first-order $\mathsf{LK}^p(\mathcal{T})$ system, for the empty theory, that is called $\mathsf{LK}_{\mathsf{F}}^{pd}$, and that is designed for the simulation of clause tableaux for first-order logic.

Definition 77 (Formulae, polarities)

The formulae of $\mathsf{LK}_{\mathsf{F}}^{pd}$ are given by the same grammar as in Definition 13, keeping in mind that there can now be meta-variables in terms, literals and formulae:

$$\text{Formulae} \quad A, B, \ldots \quad ::= \; l \mid A \wedge^+ B \mid A \vee^+ B \mid A \wedge^- B \mid A \vee^- B \mid \top^+ \mid \top^- \mid \bot^+ \mid \bot^- \mid \forall x A \mid \exists x A$$

where l ranges over literals. The construct of $\forall x A$ and $\exists x A$ bind x in A. The set of free variables of a formula A is denoted by $\mathsf{FV}(A)$. The set of (free) meta-variables that appear in a formula A is denoted by $\mathsf{MV}(A)$, although notice that **there is no binder for meta-variables**.
A formula A is *closed* if both $\mathsf{FV}(A) = \emptyset$ and $\mathsf{MV}(A) = \emptyset$.
The negation of a formula and the capture-avoiding substitution of variables in a formula are defined as usual. Polarisation sets, $\mathcal{P}$-positive and $\mathcal{P}$-negative formulae, and the set $\mathsf{U}_{\mathcal{P}}$ of $\mathcal{P}$-unpolarised literals, are all defined as in Definition 15. ※

Definition 78 (System $\mathsf{LK}_{\mathsf{F}}^{pd}$)

The sequent calculus $\mathsf{LK}^p(\mathcal{T})$ manipulates two kinds of sequents:

$$\begin{array}{ll} \text{Focused sequents} & \Gamma \vdash^{\mathcal{P}}_{\Sigma,\Omega} [A] \Rightarrow \sigma \\ \text{Unfocused sequents} & \Gamma \vdash^{\mathcal{P}}_{\Sigma,\Omega} \Delta \Rightarrow \sigma \end{array}$$

where $\mathcal{P}$ is a polarisation set, Γ is a (finite) multiset of literals and $\mathcal{P}$-negative formulae, Δ is a (finite) multiset of formulae, Σ is the signature for function symbols[4], Ω is a list of distinct meta-variables, and σ is a substitution.

All the meta-variables used in a sequent should be declared in Ω (this will be an invariant of our proof-search system), and $\sharp(\Omega)$ denotes the length of list Ω (i.e. the number of declared meta-variables).
We should see Σ, Ω, Γ, A or Δ, and $\mathcal{P}$ as the *inputs* of proof-search, and σ as the output: if in the sequent we instantiate all the meta-variables of Ω according to the output substitution σ, we should get a sequent that is provable in $\mathsf{LK}^p(\emptyset)$.
The sequent calculus $\mathsf{LK}_{\mathsf{F}}^{pd}$ is given by the rules of Figure 5.8. ※

Let us give an intuition about those rules. Their names should relate them to the rules in systems $\mathsf{LK}^p(\emptyset)$ and $\mathsf{LK}^{pd}(\emptyset)$.

- In rules $(\top^+)$ and $(\top^-)$, we output the empty substitution.
- In every rule that has two premisses ($(\wedge^+)$ and $(\wedge^-)$), we collect the output substitutions from both branches and produce their most general unifier, assuming that this exists.
- In rule $(\exists)$, we delay the instantiation of the existentially quantified variable, by picking a new meta-variable X, which we declare by adding it to Ω; the value to which X is mapped by the output substitution σ is the witness that we would need to reconstitute a proof-tree in $\mathsf{LK}^p(\emptyset)$.
- In rule $(\forall)$, we perform on-the-fly skolemisation: instead of just keeping the freed variable x as in $\mathsf{LK}^p(\emptyset)$, we enrich x by "applying it" to the list Ω of meta-variables that have previously been introduced. We do this by turning x into a new function symbol added to the signature Σ, with arity $\sharp(\Omega)$. Doing this will record the fact that the meta-variables in Ω cannot depend on x, as any instantiation of these meta-variables with terms mentioning x would now be ruled out by the occurs-check of every unification call.

  However, this will never be used in the simulation of clause or connection tableaux, since these techniques are designed for clausal forms, so all the necessary skolemisation steps have been done in pre-processing. Correspondingly, we will never encounter, in the simulation, a $\forall$ quantifier on the right-hand side of a sequent.
- The (Init_1) rule is where the computation of unifiers start.
- Rule (Store) is basically the same as in $\mathsf{LK}^p(\emptyset)$, but for the fact that the criterion used for adding or not adding a stored literal to the polarisation set involves the resulting substitution. This is contradicting the view that the resulting substitution is an output of the proof-search, but this criterion seems necessary to then be able to encode our system back into $\mathsf{LK}^p(\emptyset)$. Understanding how to deal with this in an actual proof-search procedure is left for further

[4]Skolemisation will extend it on-the-fly during proof-search.

Synchronous rules

$$(\wedge^+)\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[A]\Rightarrow\sigma_1\qquad\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[B]\Rightarrow\sigma_2}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[A\wedge^+B]\Rightarrow\mathsf{mgu}(\sigma_1,\sigma_2)}\qquad(\vee^+)\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[A_i]\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[A_1\vee^+A_2]\Rightarrow\sigma}$$

$$(\exists)\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,X::\Omega}[\{{}^{X}\!/_{x}\}A]\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[\exists xA]\Rightarrow\sigma}\qquad(\mathsf{Init}_1)\dfrac{}{\Gamma,l\vdash^{\mathcal{P},l''}_{\Sigma,\Omega}[l']\Rightarrow\mathsf{mgu}(l,l',l'')}\qquad(\top^+)\dfrac{}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[\top^+]\Rightarrow\emptyset}$$

$$(\mathsf{Release})\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}N\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}[N]\Rightarrow\sigma}\ N\text{ is not }\mathcal{P}\text{-positive}$$

Asynchronous rules

$$(\wedge^-)\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}A,\Delta\Rightarrow\sigma_1\qquad\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}B,\Delta\Rightarrow\sigma_2}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}A\wedge^-B,\Delta\Rightarrow\mathsf{mgu}(\sigma_1,\sigma_2)}\qquad(\vee^-)\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}A_1,A_2,\Delta\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}A_1\vee^-A_2,\Delta\Rightarrow\sigma}$$

$$(\forall)\dfrac{\Gamma\vdash^{\mathcal{P}}_{(x/\sharp(\Omega))::\Sigma,\Omega}\left\{{}^{x(\Omega)}\!/_{x}\right\}A,\Delta\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}(\forall xA),\Delta\Rightarrow\sigma}\qquad(\top^-)\dfrac{}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}\top^-,\Delta\Rightarrow\emptyset}$$

$$(\bot^-)\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}\Delta\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}\bot^-,\Delta\Rightarrow\sigma}\qquad(\mathsf{Store})\dfrac{\Gamma,A^\bot\vdash^{\mathcal{P};A^\bot}_{\Sigma,\Omega}\Delta\Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}A,\Delta\Rightarrow\sigma}\ A\text{ is }\mathcal{P}\text{-positive or literal}$$

Structural rules

$$(\mathsf{Select})\dfrac{\Gamma,P^\bot\vdash^{\mathcal{P}}_{\Sigma,\Omega}[P]\Rightarrow\sigma}{\Gamma,P^\bot\vdash^{\mathcal{P}}_{\Sigma,\Omega}\ \Rightarrow\sigma}\ P\text{ is not }\mathcal{P}\text{-negative}$$

Extra rule

$$(\mathsf{DPol})\dfrac{\Gamma\vdash^{\mathcal{P}}_{\Sigma,\Omega}\ \Rightarrow\sigma}{\Gamma\vdash^{\mathcal{P},l}_{\Sigma,\Omega}\ \Rightarrow\sigma}\ l\in\Gamma$$

where
$$\begin{array}{lcll}\mathcal{P};A & := & \mathcal{P},A & \text{if }\sigma(A)\in\mathsf{U}_{\sigma(\mathcal{P})}\\ \mathcal{P};A & := & \mathcal{P} & \text{if not}\end{array}$$

Figure 5.8: System $\mathsf{LK}^{pd}_{\mathsf{F}}$

work; however, for the particular proof-search process that simulates clause or connection tableaux, the issue never comes up, as the next step after (Store) is to empty the polarisation set anyway with rule (DPol).

Definition 79 (Representation of clauses as formulae)

An formula C' *represents* a clause $C=\{l_j\}_{j=1\dots n}$ with $\{x_1\dots x_p\}=\mathsf{FV}(l_1,\dots,l_n)$ if $C'=\forall x_1\dots\forall x_p(l_1\vee^-\dots\vee^-l_n)$.
A set of formulae S' *represents* a set of clauses S if there is a bijection f from S to S' such that for all clauses C in S, $f(S)$ represents C. ※

DEFINITION 80 (Encoding clause tableaux in $\mathsf{LK}^{pd}_{\mathsf{F}}$)

For any set S of clauses, and for every set S' of formulae representing S, we define a mapping Φ_F that maps:

- a clause tableaux (σ, T) for S with m open branches
- m substitutions $\sigma_1, \dots, \sigma_m$ such that $\sigma_0 = \mathsf{mgu}(\sigma, \mathsf{mgu}(\sigma_1, \mathsf{mgu}(\dots, \sigma_m)))$ exists

to an incomplete proof-tree concluding $S' \vdash^{\emptyset}_{\Sigma,\Omega} \;\Rightarrow \sigma_0$ with m open leaves,
with a bijection f from the open branches of (σ, T) to $1 \dots m$ such that, for all open branches $S \| \Gamma$ of (σ, T), the $f(S\|\Gamma)$th open leaf of the incomplete proof-tree is labelled with the sequent

$$S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega} \;\Rightarrow \sigma_{f(S\|\Gamma)}$$

We define $\Phi_F((\sigma, T), (\sigma_i)_i)$ by induction on the inductive construction of (σ, T), checking that the above property is indeed an invariant of the induction.

Initial rule: We define $\Phi_F((\emptyset, T), \sigma_0)$ as

$$\dfrac{}{S' \vdash^{\emptyset}_{\Sigma,[]} \;\Rightarrow \sigma_0}$$

Expansion rule: If (σ', T') is a weak connection tableau that is obtained, from a tableau (σ, T) with m open branches, by expanding the branch $S\|\Gamma$ of (σ, T) using a new instance $\{l_1 \dots l_n\}$ of a clause in S,
we define $\Phi_F((\sigma', T'), (\sigma'_i)_{i \in 1 \dots m+n-1})$ as follows:
Let $(\sigma_i)_{i \in 1 \dots m}$ be defined by

$$\begin{array}{lcll} \sigma_i & := & \sigma'_i & \text{for } i \in 1 \dots (m-1) \\ \sigma_m & := & \mathsf{mgu}(\sigma'_m, \mathsf{mgu}(\dots, \sigma'_{m+n-1})) & \end{array}$$

Consider the incomplete proof-tree $\Phi_F((\sigma, T), (\sigma_i)_{i \in 1 \dots m})$.
We replace its open leaf corresponding to $S\|\Gamma$ by the following incomplete proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{S', \Gamma, l_1 \vdash^{\emptyset}_{\Sigma,\Omega'} \;\Rightarrow \sigma'_m}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega'} {l_1}^{\perp} \Rightarrow \sigma'_m}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega'} [{l_1}^{\perp}] \Rightarrow \sigma'_m} \quad \cdots \quad \dfrac{\dfrac{S', \Gamma, l_n \vdash^{\emptyset}_{\Sigma,\Omega'} \;\Rightarrow \sigma'_{m+n-1}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega'} {l_n}^{\perp} \Rightarrow \sigma'_{m+n-1}}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega'} [{l_n}^{\perp}] \Rightarrow \sigma'_{m+n-1}}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega'} [\{{}^{X_1,\dots,X_p}/_{x_1,\dots,x_p}\} A^{\perp}] \Rightarrow \sigma_0}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega} [{C'}^{\perp}] \Rightarrow \sigma_m}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega} \;\Rightarrow \sigma_m}$$

where $C' = \forall x_1 \dots x_p A$ is the formula of S' representing C and $\Omega' = X_1 :: \dots X_p :: \Omega$.

Closing rule: If (σ', T') is a weak connection tableau that is obtained, from a tableau (σ, T) with m open branches, by closing its branch $S\|\Gamma$, detecting l and ${l'}^{\perp}$ in Γ, with $\sigma'' = \mathsf{mgu}(l, l')$ and $\sigma' = \mathsf{mgu}(\sigma, \sigma'')$, then we define $\Phi_F((\sigma', T'), (\sigma'_i)_{i \in 1 \dots m-1})$ as follows:
Let $(\sigma_i)_{i \in 1 \dots m}$ be defined by

$$\begin{array}{lcl} \sigma_i & := & \sigma'_i \text{ for } i \in 1 \dots (m-1) \\ \sigma_m & := & \sigma'' \end{array}$$

Consider the incomplete proof-tree $\Phi_F((\sigma, T), (\sigma_i)_{i \in 1 \dots m})$.
We replace its open leaf corresponding to $S\|\Gamma$ by the following complete proof-tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{S', \Gamma, l \vdash^{l}_{\Sigma,\Omega} [l'] \Rightarrow \sigma''}}{S', \Gamma, l \vdash^{l}_{\Sigma,\Omega} \;\Rightarrow \sigma''}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega} l^{\perp} \Rightarrow \sigma''}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega} [l^{\perp}] \Rightarrow \sigma''}}{S', \Gamma \vdash^{\emptyset}_{\Sigma,\Omega} \;\Rightarrow \sigma''}$$

※

5.5.4 The $\mathsf{LK}_{\mathsf{F}}^{pdp}$ system and the simulation of weak connection tableaux

As for the propositional case, we now extend $\mathsf{LK}_{\mathsf{F}}^{pd}$ with the (Pol) rule to simulate weak connection tableaux for first-order logic.

Definition 81 (System $\mathsf{LK}_{\mathsf{F}}^{pdp}$) We extend the sequent calculus $\mathsf{LK}_{\mathsf{F}}^{pd}$ with the following polarisation rule:

$$(\mathsf{Pol})\frac{\Gamma, l \vdash^{\mathcal{P},l'}_{\Sigma,X_1::\dots X_n::\Omega} \;\Rightarrow \sigma}{\Gamma, l \vdash^{\mathcal{P}}_{\Sigma,\Omega} \;\Rightarrow \sigma_{|\Omega}} \quad \begin{array}{l} l' \text{ renames } l \\ \mathsf{MV}(l') = \{X_1,\dots,X_n\} \\ \mathsf{MV}(l')\subseteq \mathsf{Dom}(\sigma)\end{array}$$

The resulting system is denoted $\mathsf{LK}_{\mathsf{F}}^{pdp}$. ※

In rule (Pol), we allow ourselves to polarise positively not a literal of the left-hand side, but the **renaming** of one of them using fresh meta-variables $X_1,\dots,X_n$ (which we then have to declare and add to Ω); these meta-variables should not be mapped to themselves by the output substitution σ (i.e. the literal that we declared positive must have been involved in some unification) and the values they are mapped to are then forgotten. We will see in the simulation of weak connection tableaux why this form of (Pol) is important.

Definition 82 (Encoding from weak connection tableau to $\mathsf{LK}_{\mathsf{F}}^{pdp}$)

The encoding of weak connection tableaux in $\mathsf{LK}_{\mathsf{F}}^{pdp}$ is given by a function ϕ_F satisfying exactly the same invariants as the function Φ_F for first-order clause tableaux (Definition 80), but we slightly tweak the definition to enforce weak connections:

Initial rule: If $\{l_1 \dots l_n\}$ is a new instance of $C \in S$, using fresh meta-variables $X_1,\dots,X_p$, then we define $\phi_F((\emptyset, T),(\sigma_i)_{i\in 1\dots n})$ as

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{S', l_1 \vdash^{\emptyset}_{\Sigma,\Omega} \;\Rightarrow \sigma_1}{S' \vdash^{\emptyset}_{\Sigma,\Omega} {l_1}^{\perp} \Rightarrow \sigma_1}}{S' \vdash^{\emptyset}_{\Sigma,\Omega} [{l_1}^{\perp}] \Rightarrow \sigma_1} \quad \cdots \quad \dfrac{\dfrac{S', l_n \vdash^{\emptyset}_{\Sigma,\Omega} \;\Rightarrow \sigma_n}{S' \vdash^{\emptyset}_{\Sigma,\Omega} {l_n}^{\perp} \Rightarrow \sigma_n}}{S' \vdash^{\emptyset}_{\Sigma,\Omega} [{l_n}^{\perp}] \Rightarrow \sigma_n}}{S' \vdash^{\emptyset}_{\Sigma,\Omega} [\{{}^{X_1,\dots,X_p}\!/\!_{x_1,\dots,x_p}\}A^{\perp}] \Rightarrow \sigma_0}}{S' \vdash^{\emptyset}_{\Sigma,[\,]} [{C'}^{\perp}] \Rightarrow \sigma_0}}{S' \vdash^{\emptyset}_{\Sigma,[\,]} \;\Rightarrow \sigma_0}$$

where $C' = \forall x_1 \dots x_p A$ is the formula of S' representing C and $\Omega = X_1 :: \dots X_p :: [\,]$ and $\sigma_0 = \mathsf{mgu}(\sigma_1, \mathsf{mgu}(\dots,\sigma_n))$.

Expansion rule: If (σ', T') is a weak connection tableau that is obtained, from a tableau (σ, T) with m open branches, by expanding the branch $S\|\Gamma$ of (σ, T) using a new instance $\{l_1 \dots l_n\}$ of a clause in S,
we define $\phi_F((\sigma', T'), (\sigma'_i)_{i\in 1\dots m+n-2})$ as follows:
Let l be the literal of Γ weakly connecting to some literal l_{i_0} of the new clause instance, and let $\sigma'' := \mathsf{mgu}(l, l_{i_0}^{\perp})$. We can assume without loss of generality that $i_0 = n$.
Let $(\sigma_i)_{i\in 1\dots m}$ be defined by

$$\begin{array}{lll} \sigma_i & := \sigma'_i & \text{for } i \in 1 \dots (m-1) \\ \sigma_m & := \mathsf{mgu}(\sigma'', \mathsf{mgu}(\sigma'_m, \mathsf{mgu}(\dots, \sigma'_{m+n-2}))) & \end{array}$$

Consider the incomplete proof-tree $\phi_F((\sigma, T), (\sigma_i)_{i\in 1\dots m})$.
We replace its open leaf corresponding to $S\|\Gamma$ by the following incomplete proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{S',\Gamma,l_1 \vdash^{\emptyset}_{\Sigma,\Omega'}\ \Rightarrow \sigma'_m}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega'} {l_1}^{\perp} \Rightarrow \sigma'_m}}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega'} [{l_1}^{\perp}] \Rightarrow \sigma'_m} \quad \cdots \quad \dfrac{\dfrac{S',\Gamma,l_{n-1} \vdash^{\emptyset}_{\Sigma,\Omega'}\ \Rightarrow \sigma'_{m+n-2}}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega'} {l_{n-1}}^{\perp} \Rightarrow \sigma'_{m+n-2}}}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega'} [{l_{n-1}}^{\perp}] \Rightarrow \sigma'_{m+n-2}} \quad \dfrac{}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega'} [{l_n}^{\perp}] \Rightarrow \mathsf{mgu}(l,l',l_n^{\perp})}}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega'} [\{{}^{X_1,\dots,X_p}\!/\!_{x_1,\dots,x_p}\} A^{\perp}] \Rightarrow \sigma_0}}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega''} [{C'}^{\perp}] \Rightarrow \sigma_0}}{S',\Gamma \vdash^{l'}_{\Sigma,\Omega''}\ \Rightarrow \sigma_0}}{S',\Gamma \vdash^{\emptyset}_{\Sigma,\Omega}\ \Rightarrow \sigma_m}$$

where $C' = \forall x_1 \dots x_p A$ is the formula of S' representing C,
l' is a fresh renaming of l using the meta-variables $Y_1, \dots, Y_q$,
$\Omega'' := Y_1 :: \dots Y_q :: \Omega$ and $\Omega' := X_1 :: \dots X_p :: \Omega''$,
$\sigma_0 := \mathsf{mgu}((\mathsf{mgu}(l, l', l_n^{\perp})), \mathsf{mgu}(\sigma'_m, \mathsf{mgu}(\dots, \sigma'_{m+n-2})))$ (so that $\sigma_{0|\Omega} = \sigma'_m$).
Closing Rule: As for clause tableaux (in Definition 80).

※

Now, the point of having introduced the complex rule (Pol) instead of the natural one:

$$\dfrac{\Gamma \vdash^{\mathcal{P},l}_{\Sigma,\Omega}\ \Rightarrow \sigma}{\Gamma \vdash^{\mathcal{P}}_{\Sigma,\Omega}\ \Rightarrow \sigma}\, l \in \Gamma$$

is to be able to add a fresh l' in the polarisation set instead of the literal l in Γ. Having done this, we have introduced some new meta-variables $Y_1, \dots, Y_q$ that *need* to be instantiated by the output substitution σ_0. In order to instantiate them, the *only way*[5] is to have one of the branches immediately close with (Init_1) (in our case, the branch on $l_n^{\perp}$), thus implementing the weak connection.

Had we used the above rule instead of our complex (Pol) rule, we would have added l itself to the polarisation set, and nothing would have prevented the branch on $l_n^{\perp}$ from applying ($\mathsf{Release}$) and having continued as in the other branches.

The above mechanism that forces the weak connection is designed to help identifying those incomplete $\mathsf{LK}_{\mathsf{F}}^{pdp}$-proof-trees that are the images of weak connection tableaux by ϕ_F, but this as a general aim is left as further work, owing to the complexity of the system. Even identifying those incomplete $\mathsf{LK}_{\mathsf{F}}^{pd}$-proof-trees that are the images of clause tableaux by Φ_F is left as future work.

Finally, we will not try to build a first-order version of the propositional calculus LK^{pdp+} in order to simulate first-order strong connection tableaux (mixing $\mathsf{LK}_{\mathsf{F}}^{pdp}$ with LK^{pdp+} would become very heavy). But we believe that the difficulties of strong connections and the difficulties of meta-variables are orthogonal problems, so in theory it could be done, to the same level of satisfaction as for weak connections (i.e. lacking so far a neat characterisation of those incomplete proof-trees representing connection tableaux).

[5]Imagine that instead of (Init_1) we used ($\mathsf{Release}$) and continued as in the other branches; then going back to an empty polarisation set with (DPol) would erase all traces of l' and therefore all traces of $Y_1, \dots, Y_q$, which would never be instantiated. Contradiction.

Chapter 6

Conclusion

The principal aim of this thesis is to build a general framework based on sequent calculus where automated reasoning techniques can be performed as proof-search.

Therefore at the beginning of the thesis, we reviewed the relevant literature, from Gentzen's sequent calculus LK to Liang-Miller's system LKF, a focused sequent calculus that is equipped with polarities (for literals and connectives). On the way, we reviewed linear logic, the LC sequent calculus that introduced the concept of polarities, and Andreoli's introduction of focusing.

Then we presented a new system $\mathsf{LK}^p(\mathcal{T})$ for classical logic, which extends the LKF system with on-the-fly polarisation and calls to decision procedures. We proved the meta-theory of the system, including the cut-elimination procedure, the property that changing the polarity of connectives does not change the provability of formulae, and finally the completeness of the $\mathsf{LK}^p(\mathcal{T})$ system. Moreover, we found an important feature of the $\mathsf{LK}^p(\mathcal{T})$ system, namely that in the presence of a non-trivial background theory, changing the polarity of literals does affect the provability of formulae.

Since our main motivation was to simulate automated techniques as proof-search in our focused sequent calculus, we reviewed the main techniques of SAT and SMT-solving, namely Classical DPLL, Abstract $\mathsf{DPLL}(\mathcal{T})$, etc. We also introduced a new system, Elementary $\mathsf{DPLL}(\mathcal{T})$, which is a restricted version of Abstract $\mathsf{DPLL}(\mathcal{T})$ but which provides strong simulation properties.

We presented two kinds of simulations for both Elementary and Abstract $\mathsf{DPLL}(\mathcal{T})$: (i) an indirect simulation via Tinelli's inference system, and (ii) a direct simulation from $\mathsf{DPLL}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$. This direct simulation provides the characterisation of an isomorphic image of the (Elementary) $\mathsf{DPLL}(\mathcal{T})$ system into $\mathsf{LK}^p(\mathcal{T})$, by a criterion that only involves the management of polarities.

A by-product of these chapters is the implementation of a proof-search tool for $\mathsf{LK}^p(\mathcal{T})$[1] called PSYCHE, and implemented in OCaml. The construction of a proof-tree for a given formula results from the interaction between a small kernel and plugins. The former is parameterised by a decision procedure for $\mathcal{T}$, and implements the rules in $\mathsf{LK}^p(\mathcal{T})$ (bottom-up), offering an API to apply synchronous rules (mainly, the choice of focus) while automatically applying asynchronous rules (which are invertible and therefore represent no backtrack points). Plugins can be used to import efficient automated reasoning techniques in this goal-directed framework by specifying in which order (and to which depth) the branches of the search-space should be explored. The direct simulation of Chapter 4 was implemented as a plugin for PSYCHE, i.e. a module implementing a strategy that drives the kernel, by calling its API functions, towards either a proof or the guarantee that no proof exists. An expanded description of PSYCHE can be found in [GL13].

We finally chose, among other automated reasoning techniques, clause and connection tableaux to be simulated in our $\mathsf{LK}^p(\mathcal{T})$ system. We presented simulations for both propositional and first-order logic of these tableaux. We also got an isomorphic image of these tableaux (in propositional logic) in the $\mathsf{LK}^p(\mathcal{T})$ system. Moreover, we presented a simulation of clause tableaux modulo

[1]in its version from Chapter 4

theories [Tin07] into $\mathsf{LK}^p(\mathcal{T})$. The simulation shows a parallel (characterisation) between $\mathsf{DPLL}(\mathcal{T})$ and clause tableaux modulo theories, via our target calculus $\mathsf{LK}^p(\mathcal{T})$. That will allow for instance the switch from one technique to another technique in the same run.

This thesis leaves interesting topics for future research. Since we did not obtain an isomorphic image of Abstract $\mathsf{DPLL}(\mathcal{T})$ in $\mathsf{LK}^p(\mathcal{T})$, in the future we would like to investigate whether the simulation can be improved, so that the target image of the simulation can be characterised by a simple criterion, just as it the case for the Elementary $\mathsf{DPLL}(\mathcal{T})$. This would also enable us to obtain a reverse simulation of that image back into Abstract $\mathsf{DPLL}(\mathcal{T})$.

Another direction for future work is to use our simulations to get new completeness proofs for automated reasoning techniques, by i) identifying the target image of these simulated techniques, and then ii) show that any proof of $\mathsf{LK}^p(\mathcal{T})$ can be transformed into a proof in this image.

We would also like to better understand, and improve, the simulation of first-order connection tableaux in system $\mathsf{LK}^p(\mathcal{T})$. That requires a better understanding of how to manage meta-variables in $\mathsf{LK}^p(\mathcal{T})$, especially regarding polarities. But this would open the door to (new) approaches to mix pure first-order reasoning with ground reasoning modulo theories, perhaps providing proof-theoretical grounds to the use of triggers.

As future work, it would be interesting to extend our simulations, in $\mathsf{LK}^p(\mathcal{T})$, to first-order versions of $\mathsf{DPLL}(\mathcal{T})$ [Bau00, BT08] and other extensions, for instance with equality [BT11]: Extending $\mathsf{LK}^p(\mathcal{T})$ with an in-built treatment of equality could perhaps allow the simulation of superposition or paramodulation techniques, and make them interact with the other techniques previously simulated.

Bibliography

[AFG$^+$11] M. Armand, G. Faure, B. Grégoire, C. Keller, L. Théry, and B. Werner. A modular integration of SAT/SMT solvers to Coq through proof witnesses. In *Proc. of the 1st Int. Conf. on Certified Programs and Proofs (CPP'11)*, volume 7086 of *LNCS*, pages 135–150. Springer, 2011. 11

[And92] J. M. Andreoli. Logic programming with focusing proofs in linear logic. *J. Logic Comput.*, 2(3):297–347, 1992. 9, 10, 23, 24, 25, 27

[AP89] J.-M. Andreoli and R. Pareschi. Logic programming with sequent systems, a linear logic approach. In P. Schroeder-Heister, editor, *Proc. of the Int. Work. on Extensions of Logic Programming*, LNCS, pages 1–30. Springer-Verlag, 1989. 23

[AP91] J.-M. Andreoli and R. Pareschi. Linear ojects: Logical processes with built-in inheritance. *New Generation Comput.*, 9(3/4):445–474, 1991. 23

[Bar92] H. P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabby, and T. S. E. Maibaum, editors, *Hand. Log. Comput. Sci.*, volume 2, chapter 2, pages 117–309. Oxford University Press, 1992. 9

[Bau00] P. Baumgartner. FDPLL - a first order Davis-Putnam-Longeman-Loveland procedure. In *Proc. of the 17th Int. Conf. on Automated Deduction (CADE'00)*, volume 1831 of *LNCS*, pages 200–219. Springer-Verlag, 2000. 130

[BBP11] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In *Automated Deduction*, volume 6803 of *LNCS*, pages 116–130. Springer-Verlag, 2011. 11

[BCP11] F. Besson, P.-E. Cornilleau, and D. Pichardie. Modular SMT proofs for fast reflexive checking inside Coq. In J.-P. Jouannaud and Z. Shao, editors, *Certified Programs and Proofs*, volume 7086 of *LNCS*, pages 151–166. Springer-Verlag, 2011. 11

[Bou97] S. Boutin. Using reflection to build efficient and certified decision procedure s. In M. Abadi and T. Ito, editors, *TACS'97*, volume 1281 of *LNCS*. Springer-Verlag, 1997.11

[BT08] P. Baumgartner and C. Tinelli. The model evolution calculus as a first-order DPLL method. *Artificial Intelligence*, 172(4-5):591–632, 2008. 130

[BT11] P. Baumgartner and C. Tinelli. Model evolution with equality modulo built-in theories. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Proc. of the 23rd Int. Conf. on Automated Deduction (CADE'11)*, volume 6803 of *LNCS*, pages 85–100. Springer-Verlag, 2011. 130

[CH00] P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. of the 5^{th} ACM SIGPLAN Int. Conf. on Functional Programming (ICFP'00)*, pages 233–243. ACM Press, 2000. 27

[Coq] The Coq Proof Assistant. Available at http://coq.inria.fr/ 9

[Dan90] V. Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation (et principalement du λ-calcul).* PhD thesis, Université de Paris VII, 1990. Thèse de doctorat en mathématiques. 20

[DJS95] V. Danos, J.-B. Joinet, and H. Schellinx. LKQ and LKT: sequent calculi for second order logic based upon dual linear decompositions of classical implication. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Proc. of the Work. on Advances in Linear Logic*, volume 222 of *London Math. Soc. Lecture Note Ser.*, pages 211–224. Cambridge University Press, 1995. 27

[DJS97] V. Danos, J.-B. Joinet, and H. Schellinx. A new deconstructive logic: Linear logic. *J. of Symbolic Logic*, 62(3):755–807, 1997. 27

[DLL62] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962. 10, 79

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. of the ACM Press*, 7(3):201–215, 1960. 10, 79

[Gaz10] I. Gazeau. Private communication. 2010. 87, 99

[Gen35] G. Gentzen. Investigations into logical deduction. In *Gentzen collected works*, pages 68–131. Ed M. E. Szabo, North Holland, (1969), 1935. 13, 16

[Gir87] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987. 10, 13, 17, 20

[Gir91] J.-Y. Girard. A new constructive logic: Classical logic. *Math. Structures in Comput. Sci.*, 1(3):255–296, 1991. 10, 21, 23

[Gir95] J.-Y. Girard. Linear logic: its syntax and semantics. In *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995. 17

[GL13] S. Graham-Lengrand. Psyche: a proof-search engine based on sequent calculus with an LCF-style architecture. In D. Galmiche and D. Larchey-Wendling, editors, *Proc. of the 22nd Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux'13)*, volume 8123 of *LNCS*, pages 149–156. Springer-Verlag, 2013. 129

[JS97] R. J. B. Jr. and R. Schrag. Using csp look-back techniques to solve real-world sat instances. In *AAAI/IAAI*, pages 203–208, 1997. 82

[Lau02] O. Laurent. *Etude de la polarisation en logique.* Thèse de doctorat, Université Aix-Marseille II, 2002. 27

[LC09] S. Lescuyer and S. Conchon. Improving Coq propositional reasoning using a lazy CNF conversion scheme. In *Proc. of the 7th Int. Conf. on Frontiers of combining systems (FroCoS'09)*, pages 287–303. Springer-Verlag, 2009. 11

[LDM11] S. Lengrand, R. Dyckhoff, and J. McKinna. A focused sequent calculus framework for proof search in Pure Type Systems. *Logic. Methods Comput. Science*, 7(1), 2011. 9

[LM09] C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoret. Comput. Sci.*, 410(46):4747–4768, 2009. 9, 10, 11, 13, 27, 29, 34, 49

[LQdF05] O. Laurent, M. Quatrini, and L. T. de Falco. Polarized and focalized linear and classical proofs. *Ann. Pure Appl. Logic*, 134(2-3):217–264, 2005. 27

[MNPS91] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Ann. Pure Appl. Logic*, 51:125–157, 1991. 9, 23, 27

[NOT05] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Abstract DPLL and abstract DPLL Modulo Theories. In F. Baader and A. Voronkov, editors, *Proc. of the the 11th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR'04)*, volume 3452 of *LNCS*, pages 36–50. Springer-Verlag, 2005. 80, 104

[NOT06] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. of the ACM Press*, 53(6):937–977, 2006. 10, 11, 82, 83, 84, 99, 108

[RV01] J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes).* Elsevier and The MIT Press, 2001. 10, 109, 110, 111, 123

[SS99] J. P. M. Silva and K. A. Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999. 82

[Sta78] R. Statman. Bounds for proof search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, 15:225–287, 1978. 108

[Tin02] C. Tinelli. A DPLL-based calculus for ground satisfiability modulo theories. In *Proc. of the 8th European Conf. on Logics in Artificial Intelligence*, volume 2424 of *LNAI*, pages 308–319. Springer-Verlag, 2002. 11, 87, 94, 99

[Tin07] C. Tinelli. An abstract framework for satisfiability modulo theories. In N. Olivetti, editor, *TABLEAUX*, volume 4548 of *LNCS*. Springer-Verlag, 2007. Invited talk, available at http://ftp.cs.uiowa.edu/pub/tinelli/talks/TABLEAUX-07.pdf. 10, 12, 109, 112, 130

[TS00] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Cambridge University Press, 2000. 14, 16

[Web11] T. Weber. SMT solvers: New oracles for the HOL theorem prover. *International Journal on Software Tools for Technology Transfer (STTT)*, 13(5):419–429, 2011. 11

Index

List of Figures

Résumé

Cette thèse développe un cadre théorique et général dans lequel la recherche de preuve peut interagir de façon modulaire avec des procédures spécifiques à certain domaines ou théories. Ce cadre repose sur le calcul des séquents focalisé pour la logique classique polarisée, avec quantificateurs. L'objectif est de capturer plusieurs techniques de raisonnement assisté par ordinateur qui sont utilisées dans la programmation logique, les systèmes de preuves dirigés par le but, les assistants de preuves et les prouveurs automatiques.

Nous donnons d'abord un aperçu du calcul des séquents focalisé pour la logique classique polarisée, du calcul des séquents à la Gentzen au système LKF de Liang-Miller.

Nous introduisons ensuite une extension du système LKF, appelée $\mathsf{LK}^p(\mathcal{T})$, qui permet la polarisation des littéraux *à la volée* et des appels à des procédures de décisions. Nous démontrons les propriétés habituelles de la méta-théorie : premièrement, l'élimination des coupures ; deuxièmement, le fait que les choix de polarité pour les connecteurs logiques n'affectent pas la prouvabilité des formules, et enfin la complétude de $\mathsf{LK}^p(\mathcal{T})$. Alors que le système originel de Gentzen était fortement non-déterministe lors de la recherche de preuve, la focalisation permet un contrôle fin de la largeur de l'espace de recherche. Associé à la polarisation à la volée des littéraux, le calcul des séquents $\mathsf{LK}^p(\mathcal{T})$ est particulièrement approprié au raisonnement assisté par ordinateur.

Par exemple, une technique très répandue pour résoudre la satisfaisabilité des formules propositionnelles (SAT), est DPLL. Le problème de satisfaisabilité-modulo-théories (SMT) généralise le problème SAT par l'ajout d'une théorie pour laquelle une procédure de décision est connue. Ce problème peut être résolu en généralisation l'algorithme DPLL en $\mathsf{DPLL}(\mathcal{T})$, ce que la plupart des prouveurs SMT implémentent.

Cette thèse explore ainsi de quelles façons chaque étape de la procédure $\mathsf{DPLL}(\mathcal{T})$ peut être émulée par des étapes standards de recherche de preuves dans $\mathsf{LK}^p(\mathcal{T})$, à savoir la construction incrémentale et dirigée par le but de l'arbre de preuve. Cela nous permet d'appliquer l'algorithme $\mathsf{DPLL}(\mathcal{T})$ jusqu'à un certain point, puis d'utiliser une autre technique de recherche de preuve (qui dépend du but à atteindre). Cette approche diffère des travaux existants où une procédure de décision SMT est exécutée jusqu'à ce qu'elle termine.

Le contrôle fin de la recherche de preuve, rendu possible par la focalisation et la polarisation à la volée, nous permet d'obtenir un résultat plus fort que la simple simulation de $\mathsf{DPLL}(\mathcal{T})$. Nous montrons que les preuves de $\mathsf{LK}^p(\mathcal{T})$ obtenues comme images des exécutions de $\mathsf{DPLL}(\mathcal{T})$ qui concluent à l'insatisfaisabilité d'une formule, peuvent être caractérisées par un critère simple. Ce critère ne porte que sur la manière dont la recherche de preuve focalise les formules. Nous pouvons utiliser ce critère pour obtenir une stratégie simple de recherche de preuve, bi-similaire à $\mathsf{DPLL}(\mathcal{T})$ qui consiste à faire une complétion en profondeur de l'arbre de preuve (en complétant d'abord la branche incomplète la plus à gauche) en utilisant toutes les règles d'inférence qui satisfont notre critère. Nous obtenons alors une stratégie de recherche de preuve dans $\mathsf{LK}^p(\mathcal{T})$ qui est aussi efficace que la procédure $\mathsf{DPLL}(\mathcal{T})$.

Enfin, nous explorons d'autres techniques classiques de raisonnement automatique, assez différentes par nature de DPLL : les *méthodes de tableaux* (clause tableaux, connection tableaux). Nous expliquons comment ces techniques peuvent être décrites comme des stratégies de recherche de preuve dans $\mathsf{LK}^p(\mathcal{T})$, pour la logique propositionnelle comme celle du premier ordre, ce qui ouvre de nouvelles perspectives pour la généralisation et la collaboration des techniques de tableaux et DPLL, même en présence d'une théorie.