# Chapter 3

# Proof Search

Linear logic as introduced by Girard and presented in the previous chapter is a rich system for the formalization of reasoning involving state. It conservatively extends intuitionistic logic and can therefore also serve as the logical basis for general constructive mathematics. Searching for proofs in such an expressive logic is difficult, and one should not expect silver bullets.

Depending on the problem, proof search in linear logic can have a variety of applications. In the domain of planning problems (see Section 2.3) searching for a proof means searching for a plan. In the domain of concurrent computation (see Section 2.7) searching for a proof means searching for possible computations. In the domain of logic programming (which we investigate in detail in Chapter **??**), searching for a proof according to a fixed strategy is the basic paradigm of computation. In the domain of functional programming and type theory (which we investigate in Chapter **??**), searching for a proof means searching for a program satisfying a given specification.

Each application imposes different requirements on proof search, but there are underlying basic techniques which recur frequently. In this chapter we take a look at some basic techniques, to be exploited in subsequent chapters.

## 3.1 Bottom-Up Proof Search and Inversion

The literature is not in agreement on the terminology, but we refer to the process of creating a derivation from the desired judgment on upward as *bottom-up* proof search. A snap-shot of a bottom-up search is a partial derivation, with undecided judgments at the top. Our goal is to derive all remaining judgments, thereby completing a proof.

We proceed by selecting a judgment which remains to be derived and an inference rule with which it might be inferred. We also may need to determine exactly how the conclusion of the rule matches the judgment. For example, in the $\otimes$R rule we need to decide how to split the linear hypotheses between the two premisses. After these choices have been made, we reduce the goal of deriving the judgment to a number of subgoals, one for each premiss of the selected rule. If there are no premisses, the subgoal is solved. If there are no subgoals left, we have derived the original judgment.

Using this simple intuition, the cut elimination theorem (Theorem 2.17) directly implies decidability of pure propositional linear logic as defined in Section 2.1, that is, linear logic without unrestricted resources and without the exponential connectives *of course* "$!A$" and intuitionistic implication "$A \supset B$".

**Theorem 3.1 (Decidability of Propositional Pure Linear Logic)** *Pure linear logic with connectives* $\multimap$, $\otimes$, $\mathbf{1}$, $\&$, $\top$, $\oplus$, *and* $\mathbf{0}$ *is decidable.*

**Proof:** We know by cut elimination and other results from Chapter 2.2 that $\cdot; \Delta \vdash A$ iff $\cdot; \Delta \Longrightarrow A$. Every premiss of every sequent rule in the pure fragment of linear logic without cut contains fewer connectives and quantifiers than the conclusion. Every branch must therefore be finite. Furthermore, there are only finitely many different inference rules which can be used to infer any given conclusion, and every rule has at most two premisses. Therefore, the space of possible cut-free sequent derivations of a purely linear judgment is finite and derivability is decidable. □

After Section **??** we see that this theorem still holds even if we admit quantifiers, but that it fails if we allow unrestricted hypotheses (even without quantifiers).

The second observation about bottom-up proof search is that some rules are *invertible*, that is, the premisses are derivable whenever the conclusion is derivable. The usual direction states that the conclusion is evident whenver the premisses are. Invertible rules can safely be applied whenever possible without losing completeness, although some care must be taken to retain a terminating procedure in the presence of unrestricted hypotheses. We also separate *weakly invertible* rules, which only apply when there are no linear hypotheses (besides possibly the principal proposition of the inference rule). For example, we cannot apply the $\mathbf{1}$R whenever the judgment is $\Gamma; \Delta \vdash \mathbf{1}$, although it is safe to do so when there are no linear hypotheses. Similarly, we cannot use the initial sequent rule to infer $\Gamma; \Delta, A \Longrightarrow A$ unless $\Delta = \cdot$. Strongly invertible rules apply regardless of any other hypotheses.

**Theorem 3.2 (Inversion Lemmas)** *The following table lists invertible, weakly invertible, and non-invertible rule in intuitionistic linear logic.*

| *Strongly Invertible* | *Weakly Invertible* | *Not Invertible* |
|---|---|---|
| $\multimap$R | | $\multimap$L |
| $\otimes$L, $\mathbf{1}$L | $\mathbf{1}$R | $\otimes$R |
| $\&$R, $\top$R | | $\&\mathrm{L}_1, \&\mathrm{L}_2$ |
| $\oplus$L, $\mathbf{0}$L | | $\oplus\mathrm{R}_1, \oplus\mathrm{R}_2$ |
| $\forall$R, $\exists$L | | $\forall$L, $\exists$R |
| $\supset$R, !L | !R | $\supset$L |
| | I | DL |

**Proof:** For invertible rule we prove that each premiss follows from the conclusion. For non-invertible rules we give a counterexample. The two sample case below are representative: for invertible rules we apply admissibility of cut, for non-invertible rules we consider a sequent with the same proposition on the left and right.

**Case:** $\multimap$R is invertible. We have to show that $\Gamma; (\Delta, A) \Longrightarrow B$ is derivable whenver $\Gamma; \Delta \Longrightarrow A \multimap B$ is derivable, so we assume $\Gamma; \Delta \Longrightarrow A \multimap B$. We also have $\Gamma; (\cdot, A, A \multimap B) \Longrightarrow B$, which follows by one $\multimap$L rule from two initial sequents. From the admissibility of cut (Theorem 2.16) we then obtain directly $\Gamma; (\Delta, A) \Longrightarrow B$.

**Case:** $\multimap$L is not invertible. Consider $\cdot; (\cdot, A \multimap B) \Longrightarrow A \multimap B$ for parameters $A$ and $B$. There is only one way to use $\multimap$L to infer this, which leads to $\cdot; \cdot \Longrightarrow A$ and $\cdot; (\cdot, B) \Longrightarrow A \multimap B$, neither of which is derivable. Therefore $\multimap$L is not invertible in general.

□

As a final, general property for bottom-up proof search we show that we can restrict ourselved to initial sequents of the form $\Gamma; (\cdot, P) \Longrightarrow P$, where $P$ is an atomic proposition. We write $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ for the restricted judgment whose rules are as for $\Gamma; \Delta \Longrightarrow A$, except that initial sequents are restricted to atomic propositions. Obviously, if $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ then $\Gamma; \Delta \Longrightarrow A$.

**Theorem 3.3 (Completeness of Atomic Initial Sequents)** *If* $\Gamma; \Delta \Longrightarrow A$ *then* $\Gamma; \Delta \overset{-}{\Longrightarrow} A$.

**Proof:** By induction on the the structure of $\mathcal{D} :: (\Gamma; \Delta \Longrightarrow A)$. In each case except initial sequents, we appeal directly to the induction hypothesis and infer $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ from the results. For initial sequents, we use an auxiliary induction on the structure of the formula $A$. We show only one case—the others are similar in that they follow the local expansions, translated from natural deduction to the setting of the sequent calculus. If local completeness did not hold for a connective, then atomic initial sequents would be incomplete as well.

**Case:** $\mathcal{D} = \dfrac{}{\Gamma; (\cdot, A_1 \otimes A_2) \Longrightarrow A_1 \otimes A_2}\,\mathrm{I}$, where $A = A_1 \otimes A_2$. Then we construct

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1'\\ \Gamma; (\cdot, A_1) \overset{-}{\Longrightarrow} A_1\end{array} \qquad \begin{array}{c}\mathcal{D}_2'\\ \Gamma; (\cdot, A_2) \overset{-}{\Longrightarrow} A_2\end{array}}{\Gamma; (\cdot, A_1, A_2) \overset{-}{\Longrightarrow} A_1 \otimes A_2}\,\otimes\mathrm{R}}{\Gamma; (\cdot, A_1 \otimes A_2) \overset{-}{\Longrightarrow} A_1 \otimes A_2}\,\otimes\mathrm{L}$$

where $\mathcal{D}_1'$ and $\mathcal{D}_2'$ exist by induction hypothesis on $A_1$ and $A_2$.

□

The theorems in this section lead to a search procedure with the following general outline:

1. Pick a subgoal to solve.
2. Decide to apply a right rule to the consequent or a left rule to a hypothesis.
3. Determine the remaining parameters (either how to split the hypotheses, or on the terms which may be required).
4. Apply the rule in the backward direction, reducing the goal to possibly several subgoals.

A lot of choices remain in this procedure. They can be classified according to the type of choice which must be made. This classification will guide us in the remainder of this chapter, as we discuss how to reduce the inherent non-determinism in the procedure above.

- Conjunctive choices. We know all subgoals have to be solved, but the order in which we attempt to solve them is not determined. In the simplest case, this is a form of *don't-care non-determinism*, since all subgoals have to be solved. In practice, it is not that simple since subgoals may interact once other choices have been made more deterministic. Success is a special case of conjunctive choice with no conjuncts.
- Disjunctive choices. We don't know which left or right rule to apply. Invertible rules are always safe, but once they all have been applied, many possibilities may remain. This is a form of *don't-know non-determinism*, since a sequence of correct guesses will lead to a derivation if there is one. In practice, this may be solved via backtracking, for example. Failure is a special case of a disjunctive choice with zero alternatives.
- Universal choices. In the $\forall$R and $\exists$L rules we have to choose a new parameter. Fortunately, this is a trivial choice, since *any* new parameter will work, and its name is not important. Hence this is a form of don't-care non-determinism.
- Existential choices. In the $\exists$R and $\forall$L rules we have to choose a term $t$ to substitute for the bound variable. Since there are potentially infinitely many terms (depending on the domain of quantification), this is a form of don't-know non-determinism. In practice, this is solved by *unification*, discussed in the next section.