Incorporating tables into proofs

Dale Miller and Vivek Nigam

INRIA & LIX/École Polytechnique, Palaiseau, France
`nigam at lix.inria.fr` `dale.miller at inria.fr`

Abstract. We consider the problem of automating and checking the use of previously proved lemmas in the proof of some main theorem. In particular, we call the collection of such previously proved results a *table* and use a partial order on the table's entries to denote the (provability) dependency relationship between tabled items. Tables can be used in automated deduction to store previously proved subgoals and in interactive theorem proving to store a sequence of lemmas introduced by a user to direct the proof system towards some final theorem. Tables of literals can be incorporated into sequent calculus proofs using two ideas. First, cuts are used to incorporate tabled items into a proof: one premise of the cut requires a proof of the lemma and the other branch of the cut inserts the lemma into the set of assumptions. Second, to ensure that lemma is not reproved, we exploit the fact that in *focused proofs*, atoms can have different *polarity*. Using these ideas, simple logic engines that do focused proof search (such as logic programming interpreters) are able to check proofs for correctness with guarantees that previous work is not redone. We also discuss how a table can be seen as a proof object and discuss some possible uses of tables-as-proofs.

1 Introduction

A sequence of well chosen lemmas is often an important part of presenting a proof in, at least, informal mathematics. In some situations, one might feel that the sequence of lemmas itself could constitute an actual proof, particularly if the reader of the proof has significant mathematical means to fill in the gaps between the lemmas. Of course, as lemmas at the beginning of the list are proved, they can be used to help prove lemmas later in the list.

Although generating lemmas is a well known and critical activity in mathematical proof, producing and using such lemmas can be important in, say, logic programming, deductive databases, and model checking. In such settings, the underlying proofs that such systems attempt to build are usually *cut-free* (that is, they lack the use of lemmas). That does not mean, however, that lemmas (and, hence, the cut-inference rule) do not have a role in improving the search for or the presentation of proofs.

Consider attempting to prove the conjunctive query $B \wedge C$ from a logic program Γ. This attempt can be reduced to first attempting to prove B from Γ and then C from Γ. It might well be the case that during the attempt to prove C, many subgoals might need to be proved that were previously established during the attempt to prove B. Of course, if proved subgoals can be remembered from the first conjunct to the second, then it might be possible to build smaller proofs and these might be easier to find and to check for correctness. Some implemented logic programming systems already use tables in this fashion: for example, in XSB [16] and in Twelf [15], it is possible to specify that some predicates should be *tabled*: that is, whenever an atomic formula with such a predicate is successfully proved, that atomic formula is remembered, so that, any other time a proof of that atom is attempted, the proof process can be stopped with a success.

In this paper, we consider a general notion of *table* and attempt to show how proof theory can account for the following two salient aspects of tables.
(i) *Entering tabled formulas into the proof context.* Proofs will be sequent calculus proofs, and tables will be partially ordered collections of formulas. In a straightforward fashion, the cut-inference rule is used to state the obligation to prove a tabled lemma as well as insert it into the main proof context.
(ii) *Avoiding reproving of tabled formulas.* It is easy to provide algorithmic means for making certain that formulas are not reproved (for example, prior to attempting a proof of a formula, check if that formula is in the table). More challenging is to find a purely proof theoretic solution in which the only proofs that can be built are those in which reproving cannot happen. We achieve this by first restricting tables to be literals (a typical assumption in implementations of tabling). Second, we exploit some recent developments in the understanding of *focused* proofs in intuitionistic logic that allow literals to be given different *polarity.* Polarity can be used to signal that a literal is in or out of the table. Focused proof search can then be organized so that a tabled literal is not reproved.

This paper is structured as follows. Section 2 presents a couple of examples that help to motivate particular connections between tables and proofs. Section 3 illustrates how tables can be inserted into proofs by using the multicut inference rule (a simple generalization of the cut rule). Section 4 presents the main technical background of our approach: namely, the notions of focusing and polarity in intuitionistic logic. In Section 5, we show how focusing and polarity can be exploited to ensure that reproving already proved atoms is avoided, and later, in Section 6, we extend this result to literals. Section 7 discusses the possible merits of considering tables as proof objects themselves.

2 Two motivating examples

Consider the graph depicted in Figure 1, and assume that its arcs are represented by atomic facts of the form $(arr\ N_1\ N_2)$, where N_1 and N_2 are adjacent nodes in the graph. Consider also the following two Horn clauses for describing a path in this graph: $\forall x(path\,x\,x)$ and $\forall x \forall y \forall z(arr\ x\ z \wedge path\ z\ y \supset path\ x\ y)$,

Now consider attempting a proof of the conjunctive query $path\ a_1\ a_4 \wedge path\ a_2\ a_4$. The usual goal-directed logic interpreter will attempt to prove the two conjuncts independently. After making suitable backchaining steps, both

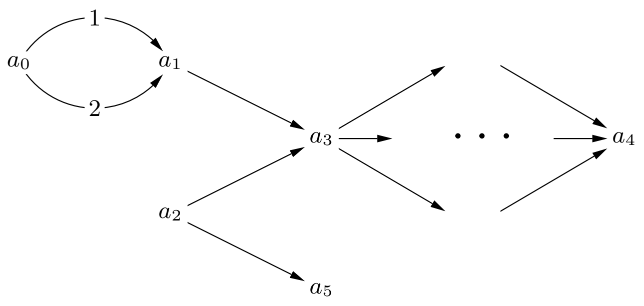

Fig. 1. Directed graph used to illustrate how tables can be used in proof search. The ellipses represent a section of the graph with a large number of paths from a_3 to a_4.

independent attempts will give rise to the same subgoal *path* a_3 a_4. The logic interpreter will then proceed to construct two (possibly identical) proofs of this subgoal. Clearly, a superior approach to proving this conjunctive goal would be to first prove the "lemma" *path* a_3 a_4, and then make that lemma available to the proof of the original conjunctive goal.

A basic problem still persists: how does one ensure that the assumed lemma is not reproved? If there are special algorithmic connections between the logic interpreter and the tabling mechanism, as exist in, say, XSB [16] and Twelf [15], then there are simple solutions to this problem of reproving lemmas. The question we are concerned with here, however, is whether or not there is an implementation independent and proof-theoretic solution to this problem.

For a second example, consider the following possible approach to *memoization* that one could attempt to use in logic programming languages (such as λProlog) that contain implicational goals [10]. Assume that the formula A is atomic and that we wish to prove the conjunction $A \wedge G$, for some general goal formula G. Since the attempt to prove G can reduce to several attempts to prove A, one might be tempted to rewrite the original conjunctive goal as the logically equivalent goal $A \wedge (A \supset G)$. During the attempt to prove G, the assumption A is available to establish any subgoal A immediately. Unfortunately, when moving from $A \wedge G$ to $A \wedge (A \supset G)$, one is making proof search more non-deterministic since for every proof that proves A by matching with the assumed version of A, there is another proof where A is, in fact, reproved. As a result, this naive approach to memoization has never been successfully used in λProlog.

This example also allows us to notice that our concern for not reproving previously proved formulas is different from the concerns of relevance logic [2], a logic in which the nature of implication is changed so that hypotheses are *necessary* for the proof of conclusions. In the example above, if the attempt to prove G succeeds without using the assumption A, the implication is still true even if the assumption A is not "relevant" to the conclusion G. The logic of this paper is intuitionistic.

Both of these examples illustrate a need for not only making proved atoms available for reuse but also enforcing that they are not reproved.

3 Tables as multicut derivations

In its most general form, we consider a table as a partially ordered finite set of formulas.

Definition 1. *A* table *is a tuple $\mathcal{T} = \langle \mathcal{A}, \preceq \rangle$, where $\mathcal{A}$ is some finite set of formulas, and $\preceq$ is a partial order relation over the elements of $\mathcal{A}$.*

The intended meaning of a table is that it is a structured collection of provable formulas (from some assumed context, say, Γ). The order relationship $B \preceq C$ denotes the fact that the proof of the formula B is available for reuse during a proof attempt of C: that is, if an attempt to prove the formula C results in the subgoal B then proof search can stop immediately since B has a proof.

The following inference rule, called the *multicut* rule, is often used as a technical generalization to the cut rule to help prove cut-elimination theorems (see, for example, [5, 19]).

$$\frac{\Delta_1 \longrightarrow B_1 \quad \cdots \quad \Delta_n \longrightarrow B_n \qquad B_1, \ldots, B_n, \Gamma \longrightarrow C}{\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow C}\ mc \quad (n \geq 0)$$

Notice that if $n = 1$, this rule reduces to the usual cut-rule (for a single conclusion calculus), and if $n = 0$, this rule is essentially a simple "repetition." If $n \geq 1$, then this rule can be seen as encoding n separate applications of the cut-rule.

The following definition describes how a table can be translated to a collection of multicut inference rules.

Definition 2. *Let $\mathcal{T} = \langle \mathcal{A}, \preceq \rangle$ be a table. The* multicut derivation *for $\mathcal{T}$ and the sequent $\mathcal{S} = \Gamma \longrightarrow G$, written as $mcd(\mathcal{T}, \mathcal{S})$, is defined inductively as follows: if $\mathcal{A}$ is empty, then $mcd(\mathcal{T}, \mathcal{S})$ is the derivation containing just the sequent $\Gamma \longrightarrow G$. Otherwise, if $\{A_1, \ldots, A_n\}$ is the collection of $\preceq$-minimal elements in $\mathcal{A}$ and if Π is the multicut derivation for the smaller table $\langle \mathcal{A} \setminus \{A_1, \ldots, A_n\}, \preceq \rangle$ and the sequent $\Gamma, A_1, \ldots, A_n \longrightarrow G$, then $mcd(\mathcal{T}, \mathcal{S})$ is the derivation*

$$\frac{\Gamma \longrightarrow A_1 \quad \cdots \quad \Gamma \longrightarrow A_n \qquad \begin{array}{c}\Pi \\ \Gamma, A_1, \ldots, A_n \longrightarrow G\end{array}}{\Gamma \longrightarrow G}\ mc$$

Multicut derivations are always open *derivations (that is, they contain leafs that are not proved). A* proof of a multicut derivation *is any (closed) proof that extends this open derivation.*

To illustrate this definition, consider the graph example in Section 2: let Γ contain the encoding of the original adjacency information as well as the specification of the *path* predicate, and consider the table that contains just the atomic formula *path* a_3 a_4. The following is the multicut derivation for $\Gamma \longrightarrow path\ a_1\ a_4 \wedge path\ a_2\ a_4$:

$$\frac{\Gamma \longrightarrow path\ a_3\ a_4 \qquad \Gamma, path\ a_3\ a_4 \longrightarrow path\ a_1\ a_4 \wedge path\ a_2\ a_4}{\Gamma \longrightarrow path\ a_1\ a_4 \wedge path\ a_2\ a_4}\ mc$$

By using the cut, it was possible to introduce the lemma *path* a_3 a_4 in the context of the rightmost branch. The left premise requires showing that there is, in fact, a path from a_3 to a_4 while the right branch attempts to show the original conjunctive goal under the assumption that the existence of that path is granted. Unfortunately, there are proofs of this right-most premise where this lemma is not used but is reproved. In the next section, we introduce the notions of *focusing* and *polarity* in order to provide means for enforcing reuse.

4 Focusing and polarities

In order to present a focused proof system, we first classify the connectives $\wedge$, $\exists$, *true* and $\perp$ as *synchronous* (their right introduction is not necessarily invertible) and the connectives $\supset$, and $\forall$ as *asynchronous* (their right introduction rules are invertible). This dichotomy must also be extended to atomic formulas: some atoms are considered asynchronous and the rest are considered synchronous. Since the terms "asynchronous" and "synchronous" do not apply well to atomic formulas, we shall instead use the slightly more general notions of *polarity* for a formula. In particular, a formula is positive if its main connective is synchronous or it is a *positive atom* and is negative if its main connective is asynchronous or it is a *negative atom*. The polarity of atoms is not necessarily fixed: we shall assign different polarities to atoms to achieve different purposes.

Although the notion of *focused proof* was originally given by Andreoli for linear logic [3], we shall use the recently designed LJF focused proof system for intuitionistic logic [7] displayed in Figure 2. This system has four types of sequents.

1. The sequent $[\Gamma]{-}_A{\rightarrow}$ is a *right-focusing* sequent (the focus is A);
2. The sequent $[\Gamma] \xrightarrow{A} [R]$: is a *left-focusing* sequent (with focus on A);
3. The sequent $[\Gamma], \Theta \longrightarrow \mathcal{R}$ is an *unfocused sequent*. Here, Γ contains negative formulas and positive atoms, and $\mathcal{R}$ is either in brackets, written as $[R]$, or without brackets;
4. The sequent $[\Gamma] \longrightarrow [R]$ is an instance of the previous sequent where Θ is empty.

As an inspection of the inference rules of LJF reveals, the search for a *focused* proof is composed of two alternating phases, and these phases are governed by polarities. The *asynchronous phase* applies invertible (asynchronous) rules until exhaustion: no backtracking during this phase of search is needed. The asynchronous phase uses the third type of sequent above (the unfocused sequents): in that case, Θ contains positive or negative formulas. If Θ contains positive formulas, then an introduction rule (either $\wedge_l, \exists_l$ or false_l) is used to decompose it; if it is negative, then the formula is moved to the Γ context (by using the $[]_l$ rule). The end of the asynchronous phase is represented by the fourth type of sequent. Such a sequent is then established by using one of the decide rules, D_r or D_l. The application of one of these decide rules then selects a formula for focusing and switches proof search to the *synchronous phase* or *focused phase*.

$$\frac{[N,\Gamma] \xrightarrow{N} [R]}{[N,\Gamma] \longrightarrow [R]}\ D_l \qquad \frac{[\Gamma] -_{P}\!\rightarrow}{[\Gamma] \longrightarrow [P]}\ D_r \qquad \frac{[\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{P} [R]}\ R_l \quad \frac{[\Gamma] \longrightarrow N}{[\Gamma] -_{N}\!\rightarrow}\ R_r$$

$$\frac{[\Gamma, N_a], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, N_a \longrightarrow \mathcal{R}}\ []_l \quad \frac{[\Gamma], \Theta \longrightarrow [P_a]}{[\Gamma], \Theta \longrightarrow P_a}\ []_r$$

$$\frac{}{[\Gamma] \xrightarrow{A_n} [A_n]}\ I_l \quad \frac{}{[\Gamma, A_p] -_{A_p}\!\rightarrow}\ I_r$$

$$\frac{}{[\Gamma], \Theta, \bot \longrightarrow \mathcal{R}}\ \mathit{false}_l \quad \frac{[\Gamma], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \mathit{true} \longrightarrow \mathcal{R}}\ \mathit{true}_l \quad \frac{}{[\Gamma] -_{\mathit{true}}\!\rightarrow}\ \mathit{true}_r$$

$$\frac{[\Gamma], \Theta, A, B \longrightarrow \mathcal{R}}{[\Gamma], \Theta, A \wedge B \longrightarrow \mathcal{R}}\ \wedge_l \quad \frac{[\Gamma] -_{A}\!\rightarrow \quad [\Gamma] -_{B}\!\rightarrow}{[\Gamma] -_{A \wedge B}\!\rightarrow}\ \wedge_r$$

$$\frac{[\Gamma] -_{A}\!\rightarrow \quad [\Gamma] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{A \supset B} [R]}\ \supset_l \quad \frac{[\Gamma], \Theta, A \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \supset B}\ \supset_r$$

$$\frac{[\Gamma], \Theta, A \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \exists y A \longrightarrow \mathcal{R}}\ \exists_l \quad \frac{[\Gamma] -_{A[t/x]}\!\rightarrow}{[\Gamma] -_{\exists x A}\!\rightarrow}\ \exists_r \quad \frac{[\Gamma] \xrightarrow{A[t/x]} [R]}{[\Gamma] \xrightarrow{\forall x A} [R]}\ \forall_l \quad \frac{[\Gamma], \Theta \longrightarrow A}{[\Gamma], \Theta \longrightarrow \forall y A}\ \forall_r$$

Fig. 2. The LJF system [7] originally has two conjunctions, $\wedge^+, \wedge^-$. In this paper, we only need one conjunction: we will drop $\wedge^-$ and write $\wedge$ for $\wedge^+$. Here A_n denotes a negative atom, A_p a positive atom, P a positive formula, N a negative formula, N_a a negative formula or an atom, and P_a a positive formula or an atom. All other formulas are arbitrary and y is not free in Γ, Θ or R.

This focused phase then proceeds by applying sequences of inference rules on focused formulas: in general, backtracking may be necessary in this phase of search.

As is pointed out in [7], if all atoms are given negative polarity, the resulting proof system models backwardchaining proof search and uniform proofs [11]. If positive atoms are permitted as well, then forwardchaining steps can also be accommodated.

We now present the LJF^t proof system that extends LJF by adding a multicut rule and by allowing atoms to have different polarity on the different branches of the multicut rule. In particular, occurrences of atoms in LJF^t proofs are assigned polarities in the following fashion: all atoms are initially given negative polarity: thus proof search with such atoms is the usual goal-directed search. When an atom is inserted into a proof context via a multicut inference rule, that atom's occurrences on the right-most branch will have positive polarity: in principle, a forwardchaining discipline is used on that atom on that branch, and it is this discipline that is used to implement the reuse policy on that part of the multicut derivation.

The sequents in LJF^t are the same four kinds of sequents except that we add a polarity declaration, $\mathcal{P}$, to all of them: if an atom appears in the set of atoms

$\mathcal{P}$, then it is considered positive; otherwise it is considered negative. Recall also that literals are either atomic formulas or negated atomic formulas (and that $\neg A$ is encoded as $A \supset \bot$). The multicut rule is the only rule that can change the declaration $\mathcal{P}$. In particular, the *polarized version* of the multicut rule is given as

$$\frac{\mathcal{P}; [\Gamma] \longrightarrow [L_1] \qquad \cdots \qquad \mathcal{P}; [\Gamma] \longrightarrow [L_n] \qquad \mathcal{P} \cup \Delta_P; [\Gamma \cup \Delta_L] \longrightarrow [R]}{\mathcal{P}; [\Gamma] \longrightarrow [R]}\ mc.$$

Here, $\Delta_L = \{L_1, \ldots, L_n\}$ is a set of literals and $\Delta_P = \{A \mid A \in \Delta_L \text{ or } \neg A \in \Delta_L\}$ is the set of all atoms in Δ_L. Notice that the literals in Δ_L are cut-formulas and that the atoms in Δ_P switch their polarity from negative in the conclusion of this rule to positive in the right-most premise. Whenever we use this multicut inference rule, we shall arrange things so that the sets Δ_P and $\mathcal{P}$ are disjoint.

As the notion of polarity of an atom is now declared via $\mathcal{P}$ instead of being globally fixed as in LJF, the inference rules in LJF^t must be adapted accordingly from LJF: for example, the LJF^t rule I_r^t will be derived from the LJF rule I_r as follows:

$$\frac{}{\mathcal{P}; [\Gamma, A_p] -_{A_p}\!\!\rightarrow}\ I_r^t, \text{ where } A_p \in \mathcal{P}.$$

In general, if the name of a rule is R in LJF, the corresponding rule in LJF^t is R^t. The following proposition can be proved by a simple induction on the depth of the cut free proofs.

Proposition 1. *LJF^t is sound and complete with respect to LJF.*

5 Tables of finite successes

In this section, we restrict our attention in two directions. First, we shall only consider tables containing atomic formulas. Such a restriction is familiar from such implemented tabling systems as [16, 15] where the only items placed in a table are atomic formulas. Second, we shall only allow logic specifications to be Horn clauses, which are defined as D-formulas in the following grammar.

$$G := true \mid A \mid G_1 \wedge G_2 \mid \exists x\, G \qquad\qquad D := A \mid G \supset A \mid D_1 \wedge D_2 \mid \forall x\, D$$

As a consequence of these restrictions, we shall only be tabling atoms if they are proved by "finite success": this contrasts with the situation addressed in the next section where tables can contain negated atoms if "finite failure" is successful to prove them. The restriction to Horn clause formulas is critical for the results here since such clauses ensure that the goal-reduction phase can be seen as completely synchronous. Goals with implications and universal quantifiers causes goal-reduction to mix synchronous and asynchronous phases. Therefore, allowing them can cause the focus of proof search to be broken before positive atomic formulas are encountered.

The following proposition states that a multicut derivation of a provable sequent (using the polarized version of the multicut rule) can be extended to a valid focused proof.

Proposition 2. *Let Γ be a collection of Horn clauses, G be a G-formula, and let $\mathcal{T}$ be a table of atoms, all of which are provable from Γ (the partial order is not restricted). The sequent $\Gamma \longrightarrow G$ is intuitionistically provable if and only if the open derivation $mcd(\mathcal{T}, \Gamma \longrightarrow G)$ can be extended to a proof in LJF^t.*

Proof The proof in the forward direction is by induction on the length of the longest path in the table's partial order. The converse is proved by forgetting the polarity information and using cut-elimination. □

The next proposition shows that polarities can be used to guarantee that any tabled atomic formula that has been proved once (and, hence, has positive polarity) will not be reproved. This proposition is proved by induction on the depth of the proof tree.

Proposition 3. *Let Γ be a set of Horn clauses, $A \in \mathcal{P} \cap \Gamma$, and Ξ be an arbitrary LJF^t proof tree for $\mathcal{P}; [\Gamma] -_{G}\rightarrow$. Then every occurrence of a sequent with right-hand side the atom A is the conclusion of an I_r^t rule.*

Since all the lemmas of a table are included as positive atoms in the right branch of its multicut derivation, all the proofs of any lemma in this branch will be composed of a single rule I_r^t.

Consider again the example in Section 2, where the subgoal *path* a_3 a_4 is tabled. Any proof of the rightmost branch of the multicut derivation obtained, will never reprove the lemma *path* a_3 a_4:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{path\ a_3\ a_4; [\Gamma, path\ a_3\ a_4] \xrightarrow{arr\, a_1\ a_3} [arr\, a_1\ a_3]}\ I_l^t}{path\ a_3\ a_4; [\Gamma, path\ a_3\ a_4] \longrightarrow [arr\, a_1\ a_3]}\ D_l^t}{path\ a_3\ a_4; [\Gamma, path\ a_3\ a_4] -_{arr\, a_1\ a_3}\rightarrow}\ R_r^t \quad \dfrac{}{path\ a_3\ a_4; [\Gamma, path\ a_3\ a_4] -_{path\ a_3\ a_4}\rightarrow}\ I_r^t}{\dfrac{path\ a_3\ a_4; [\Gamma, path\ a_3\ a_4] -_{arr\, a_1\ a_3 \wedge path\ a_3\ a_4}\rightarrow}{path\ a_3\ a_4; [\Gamma, path\ a_3\ a_4] \longrightarrow [path\ a_1\ a_4]}\ D_l^t, \forall_l^t, \forall_l^t, \forall_l^t, \supset_l^t}\ \wedge_r^t$$

The memoization example of Section 2 can be addressed similarly: instead of doing the goal reduction illustrated on the left below, use a multicut as is illustrated on the right:

$$\dfrac{\Gamma \longrightarrow A \qquad \Gamma \longrightarrow G}{\Gamma \longrightarrow A \wedge G} \quad \Longrightarrow \quad \dfrac{\mathcal{P}; [\Gamma] \longrightarrow [A] \qquad \mathcal{P} \cup \{A\}; [\Gamma, A] \longrightarrow [A \wedge G]}{\mathcal{P}; [\Gamma] \longrightarrow [A \wedge G]}\ mc.$$

In this way, all attempts to prove A on the right will be trivial applications of the initial rule.

When the asynchronous phase of proof search ends, that is, when all the invertible rules have been applied, the decide rules, namely D_l^t and D_r^t, chose a formula on which search should focus. Since logic programs generally contain many formulas, the choice made by these decide rules is a form of *don't know* non-determinism, which is a potential source of backtracking. For example, while the sequent $[A_1, A_1 \supset A_0, A_2 \supset A_0] \longrightarrow [A_0]$ has four formulas on which to focus, a valid LJF proof can be built on by focusing on the formula $A_1 \supset A_0$ (here, A_0, A_1, A_2 are atomic formulas).

While we are mainly interested in the use of tables and not with their discovery, we consider briefly one example of how a table can be built. In particular, a cut-free LJF proof Ξ of $\Gamma \longrightarrow G$ can be made into a table as follows. The table consists of all atoms that are on the right-hand side of some sequent in Ξ. The occurrences of proved atoms in Ξ can be ordered using postorder traversal (*i.e.*, process a node's premises before processing the node). The final order used for the table (which is on atomic formulas and not their occurrences) is then obtained from this postorder traversal by retaining only the first occurrence of any repeated atomic formula. The following proposition shows that it is trivial to extend a multicut derivation that is built in this way from a complete proof: the following definition helps to formalize what we mean as trivial here.

Definition 3. *The* decide-depth *of an LJF^t proof Ξ is the maximum number of occurrences of decide rules (*i.e.*, D_r and D_l) on any path from the root to a leaf in Ξ.*

Proposition 4. *Let Ξ be a LJF proof of $\Gamma \longrightarrow G$ and let $\mathcal{T}$ be a table obtained from Ξ using the postorder traversal described above. There exists a proof for $mcd(\mathcal{T}, [.]\Gamma \longrightarrow G)$ such that all of its added subproofs have decide-depth of one or less.*

Proof Proof by induction on the length of the table's longest chain. □

Given that it is simple to check if a table is derived from a cut-free proof, one might consider that the table is, in fact, a legitimate proof object. Within the proof carrying code framework [12], it might be more interesting to send an ordered collection of atoms in order to represent a proof than to send some more complex representation of a sequent calculus proof tree. We will return to this aspect of tables in Section 7.

6 Tables of finite failures

We now generalize LJF^t by including a proof theoretic notion of *fixed points* that is treated technically using a notion of *definitions*. A *definition* is a countable set of *clauses*, written as $\forall \bar{x}[p\,\bar{t} \stackrel{\Delta}{=} B]$: here p is a predicate, every free variable of B (the *body* of the clause) is also free in the atom $p\,\bar{t}$ (the *head* of the clause), and all variables free in $p\,\bar{t}$ are contained in the list $\bar{x}$ of variables. The symbol $\stackrel{\Delta}{=}$ is not a logical connective but is used to indicate a definitional clause. The left and right introduction rules for defined atoms, namely Def_l and Def_r, are shown in Figure 3. Notice that all free variables in a sequent are *eigenvariables* (no *logical* variables appear here). We shall call LJ^{Δ} the result of adding to Gentzen's LJ calculus the unpolarized versions of Def_l and Def_r (this logic is a first-order version of the logic $FO\lambda^{\Delta}$ in [8, 9]). The polarized version of this proof system $LJF^{\Delta t}$ results from adding the inference rules in Figure 3 to LJF^t.

As is shown in [18, 6], this notion of definition can yield a proof theoretic approach to negation-as-failure. We shall use this aspect of definitions to extend the notion of table of finite success in Section 5 to also contain finite failures.

$$\frac{\{\mathcal{P};[\Gamma\theta],\Theta\theta,B\theta \longrightarrow \mathcal{R}\theta \mid \theta = mgu(H,A) \text{ for some clause } H \stackrel{\Delta}{=} B\}}{\mathcal{P};[\Gamma],\Theta,A \longrightarrow \mathcal{R}}\ Def_l, A \notin \mathcal{P}$$

$$\frac{\mathcal{P};[\Gamma]-_{B\theta}\rightarrow}{\mathcal{P};[\Gamma]-_{A_n}\rightarrow}\ Def_r, A_n \notin \mathcal{P},\ \text{where } H \stackrel{\Delta}{=} B, \text{ and } H\theta = A_n$$

$$\frac{\mathcal{P};[\Gamma]-_{P_a}\rightarrow}{\mathcal{P};[\Gamma] \longrightarrow [P_a]}\ D_r^t$$

Fig. 3. The rules for introducing atomic formulas and for selecting a formula on the right. Remember that A_n denotes a negative atom and that P_a denotes a positive formula or an atom (positive or negative).

As a consequence, tables will now contain both atoms and negated atoms (*i.e.* literals). The literal $\neg A$ is always of negative polarity since it is defined by the asynchronous formula $A \supset \bot$: notice that the atom A can be either of positive or negative polarity.

The proof theoretic characterization of negation-as-failure is obtained by the Def_l rule. When this rule is used to introduce the atom A on the left of a sequent, a premise for each possible way that the definition could entail A is created in one step. Since all possible instances must be considered, this rule is part of the asynchronous phase of proof search. On the other hand, the Def_r rule's behavior is similar to that of the backchaining rule of a logic interpreter and, therefore, is applied only in the synchronous phase. We extend the idea of the previous section and consider that backchaining (that is the Def_r rules) is applied only to negative atoms and forwardchaining to positive atoms. Hence, we allow focusing on negative atoms, but do not allow Def_r to be applied on positive atoms.

Proposition 5. *$LJF^{\Delta t}$ is sound and complete with respect to LJ^{Δ}.*

Proof Soundness follows simply by dropping polarity information from sequents and by using cut-elimination. To prove completeness, assume that a sequent $\Gamma \longrightarrow B$ is provable in LJ^{Δ}. All we need to show is that $[\cdot]\Gamma \longrightarrow B$ (an unfocused sequent with no classified formulas) has an $LJF^{\Delta t}$ proof with an empty table (that is, without any occurrence of the multicut inference rule). As a result, completeness is proved by showing that any cut-free proof in LJ^{Δ} can be made into a focused proof by permutations of inference rules following standard argument lines, such as those in [7, 9]. □

Assume again here that all definitions are based on Horn clauses: in particular, all definition clauses are of the form $\forall\bar{x}[A \stackrel{\Delta}{=} G]$ where G is a goal formula defined as at the start of Section 5. For example, the specification of the *path* predicate in Section 2 is written as the two-clause definition

$$\forall x\forall y[path\ x\ y \stackrel{\Delta}{=} \exists z(arr\ x\ z \wedge path\ z\ y)] \quad \text{and} \quad \forall x[path\ x\ x \stackrel{\Delta}{=} true].$$

Since definitions are considered to be global, they are not included in sequents: as a consequence, the left-hand side of sequents contains only the formulas inserted by multicuts.

In the previous section, we used decide-depth as a measure of proof complexity (from the point-of-view of discovering the proof). In the logic considered in this section, it seem more sensible to use the following measure instead.

Definition 4. *The* Def_r-depth *of an* LJF^t *proof* Ξ *is the maximum number of occurrences of the* Def_r *rule on any path from the root to a leaf in* Ξ.

The next proposition, which can be proved by induction on proof trees, guarantees that a sequent that does a right-hand focusing on a literal built from a positive polarity atom yields proofs with small Def_r-depth. Notice that a proof with small Def_r-depth is not necessarily small since the Def_l inference rule can be used without bound: uses of the Def_l, however, are always invertible.

Proposition 6. *Let* $\mathcal{D}$ *be a set of definitions,* Γ *be a set of literals built on positive polarity atoms, and* $L \in \Gamma$. *If* Ξ *is an* $LJF^{\Delta t}$ *proof of* $\mathcal{P}; [\Gamma] -_G\rightarrow$ *then all occurrences of sequents in* Ξ *that have* L *as their right-focus formula are the conclusion of a proof with* Def_r*-depth at most 1.*

In particular, if L is $\neg A$ and Γ' is Γ with L removed, then an attempt to prove $\mathcal{P}; [\neg A, \Gamma'] -_{\neg A}\rightarrow$ can only yield an “immediate” proof: the proof of this sequent reduces to $\mathcal{P}; [\neg A, \Gamma', A] \longrightarrow \bot$ and this sequent is provable if and only if there is an atomic B such that $B \supset \bot$ and B are in $\Gamma \cup \{A\}$ (given that B has positive polarity).

If we know that a certain atom A is not intuitionistically provable from a set of assumptions Γ (using finite-failure, for example) then it is possible to organize focused proof search to fail immediately when an attempt to prove A is made. The following proposition, which is proved by induction on the depth of the proof tree, provides that conclusion since it states that such attempts on A are not the conclusion of any $LJF^{\Delta t}$ inference rule.

Proposition 7. *Let* A *be an atom such that* $\Gamma \longrightarrow A$ *is not provable in* LJ^{Δ} *and let* $A \in \mathcal{P}$. *Let* Ξ *be an arbitrary* $LJF^{\Delta t}$ *derivation for* $\mathcal{P}; [\Gamma] -_G\rightarrow$. *Then all sequents in* Ξ *with right-hand side* A *are open leafs.*

To illustrate this proposition, assume that we have proved the lemma $\neg A$ from Γ. On the right premise of the cut-rule used to insert $\neg A$ as an additional assumption, the atom A is given positive polarity. If one attempts to prove A (with left-hand side Γ) then Def_r cannot be applied. Similarly, the only other way to prove such a sequent is the I_r^t rule, but this implies that the positive atom A is in Γ, which is only possible if A was, in fact, proved from Γ, which is explicitly ruled out. Thus, using polarity, it is possible to “immediately” recognize a failure to prove A.

We can transplant the graph example in Section 2 to this section by mapping the Horn clause specifications for the *path*-atoms and *arr*-atoms into the corresponding definitions. Assume that the table contains only the literal $\neg path\, a_1\, a_5$.

The proof of the multicut derivation for the query $\neg path\ a_0\ a_5$ is illustrated below. Here, $\mathcal{P}$ is the set $\{path\ a_1\ a_5\}$ and A is an eigenvariable of the proof.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{P};[\Gamma, path\ a_1\ a_5] \xrightarrow[path\ a_1\ a_5]{}}\ I_r^t \qquad \dfrac{\dfrac{}{\mathcal{P};[\Gamma, path\ a_1\ a_5], \bot \longrightarrow [\bot]}\ \mathit{false}_l^t}{\mathcal{P};[\Gamma, path\ a_1\ a_5] \xrightarrow{\bot} [\bot]}\ R_l^t}{\mathcal{P};[\Gamma, path\ a_1\ a_5] \xrightarrow{\neg path\ a_1\ a_5} [\bot]}\ \supset_l^t}{\mathcal{P};[\Gamma, \neg\, path\ a_1\ a_5], path\ a_1\ a_5 \longrightarrow \bot}\ []_l^t, D_l^t}{\mathcal{P};[\Gamma, \neg\, path\ a_1\ a_5], arr\ a_0\ A, path\ A\ a_5 \longrightarrow \bot}\ \mathit{Def}_l \star}{\mathcal{P};[\Gamma, \neg\, path\ a_1\ a_5], \exists z[arr\ a_0\ z \wedge path\ z\ a_5] \longrightarrow \bot}\ \exists_l, \wedge_l^t}{\mathcal{P};[\Gamma, \neg\, path\ a_1\ a_5], path\ a_0\ a_5 \longrightarrow \bot}\ \mathit{Def}_l}{\mathcal{P};[\Gamma, \neg\, path\ a_1\ a_5] \longrightarrow \neg path\ a_0\ a_5}\ \supset_r^t
$$

7 Table as proof objects

We have illustrated how tables can be incorporated into proofs. To what extent can we think of tables as proofs themselves? Of course, this question is best addressed when one knows what one will do with a proof.

In the *proof carrying code* setting [12], proof objects are transmitted together with mobile codes to assure that some (safety) properties are satisfied by these programs. Before a client executes the transmitted code the client checks that the proof that that code is carrying proves the program's safety. Thus, proof objects must be engineered so that they are not too large (in order to reduce transmission costs) and not too complex to check (in order to reduce resource requirements on client proof checkers).

Tables might well be a good format for proofs in this setting for several reasons. First, tables represent declarative information and not procedural information: in particular, tables only describe what is provable and does not go into detail about how things are proved. Proof checking can then be organized around simple proof search engines that implement, for example, *LJF*. The trade-offs between proof size and proof checking time are fairly clear: if the producer of a proof tables all successfully proved atoms (as in Proposition 4) then tables can be large but proof checking can be simple (only proofs of decide-depth 1 must be considered in extending a multicut derivation). On the other hand, if some atomic formulas are not tabled, then the client may have to reprove them: clearly, reproving some atomic formulas might be rather straightforward and something that a client might be willing to do to help reduce the size of a transmitted proof.

In [17], Roychoudhury *et.al.* propose using tables to build *justifications* that can be seen as a kind of proof. In their setting, these proof objects serve to explain why a logic program can or cannot prove a given atom. They argue that their justification can be used within model checkers and parsers. It seems likely that our use of tables as proofs can be used in these settings as well.

We now consider two examples where tables relate to more than just proofs: they can also be simulations (Example 1) and winning strategies (Example 2). These examples also illustrate that non-Horn examples can also be used in the framework that was described above.

Example 1. Encode a noetherian abstract labeled transition system as a definition by writing the transition $P \xrightarrow{A} P'$ as the clause $one(p, a, p') \stackrel{\triangle}{=} true$. McDowell *et.al.* showed in [9] that the additional definition clause

$$\forall P, Q[sim(P, Q) \stackrel{\triangle}{=} \forall A, P'.one(P, A, P') \supset \exists Q'.one(Q, A, Q') \wedge sim(P', Q')]$$

can be used to compute the simulation relation. In particular, processes P is simulated by Q if and only if the atomic formula $sim(P, Q)$ is provable. (Bisimulation can be encoded using a slightly more complex definition.) Moreover, if Ξ is a cut-free proof of that atomic formula and if $\mathcal{S}$ is the set of all pairs $\langle t, s\rangle$ such that Ξ contains a subproof of $sim(t, s)$, then $\mathcal{S}$ is a simulation. Furthermore, let $\preceq$ be the postorder relation on $\mathcal{S}$ derived from Ξ as described in Section 5. Notice that it is now a simple matter to check that $\mathcal{S}$ is, in fact, a simulation by treating it as a table and considering extending its induced multicut derivation to a complete proof. In particular, let $\langle p, q\rangle \in \mathcal{S}$ and let $\mathcal{P} = \{sim(t, s) \mid \langle t, s\rangle \in \mathcal{S},$ and $sim(t, s) \prec sim(p, q)\}$. An attempt to extend the sequent $\mathcal{P}; [\mathcal{P}] \longrightarrow sim(p, q)$ yields a proof of the form

$$\dfrac{\dfrac{\cdots \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{P}; [\mathcal{P}] -_{true}\!\rightarrow}\; true^t_r}{\mathcal{P}; [\mathcal{P}] -_{one(q,a,q')}\!\rightarrow}\; \mathit{Def}_r \qquad \dfrac{}{\mathcal{P}; [\mathcal{P}] -_{sim(p',q')}\!\rightarrow}\; I^t_r}{\mathcal{P}; [\mathcal{P}] -_{one(q,a,q') \wedge sim(p',q')}\!\rightarrow}\; \wedge^t_r}{\mathcal{P}; [\mathcal{P}] -_{\exists Q'.one(q,a,p') \wedge sim(p',Q')}\!\rightarrow}\; \exists^t_r}{\mathcal{P}; [\mathcal{P}] \longrightarrow [\exists Q'.one(q, a, Q') \wedge sim(p', Q')]}\; D_r \qquad \cdots}{\mathcal{P}; [\mathcal{P}], one(p, A, P') \longrightarrow [\exists Q'.one(q, A, Q') \wedge sim(P', Q')]}\; \mathit{Def}_l}{\mathcal{P}; [\mathcal{P}] \longrightarrow \forall A, P'.\ one(p, A, P') \supset \exists Q'.\ one(q, A, Q') \wedge sim(P', Q')}\; \forall^t_l, \supset^t_l, []^t_r$$

The ellipses represents that there are other premises generated by the Def_l rule that introduces the atom $one(p, A, P')$: there is one premise for each pair $\langle a', p'\rangle$ such that $p \xrightarrow{a'} p'$ (if there is none, then the proof is completed at this point). Notice that the only Def_r rule in this proof is on the one-step transition and since these are given via a simple list of clauses, finding a q' such that $q \xrightarrow{a'} q'$ is a simple computation.

Example 2. Consider a game between two players, named 1 and 2, who alternate in playing (consider tic-tac-toe) and that one player wins when the other player cannot move. We assume that the state of the game is encoded as a term in the logic and that the binary predicate $move(P, Q)$ encodes the fact that there is move from position P to Q. Furthermore, assume that there are no infinite plays. Then there is a winning strategy from the position P if and only if the atom $win(P)$ is provable from a definition that includes the clause

$$\forall P[win(P) \overset{\Delta}{=} \forall P'.\ move(P,P') \supset \exists Q.\ move(P',Q) \wedge win(Q)]$$

as well as the (Horn clause) definition of $move(P,Q)$. As with the previous example, let Ξ be a proof of the atom $win(p)$, let let $\mathcal{W}$ be the set of atoms of the form $win(P)$ that are proved in subproofs of Ξ, and let $\preceq$ be the postorder traversal ordering of $\mathcal{W}$ based on Ξ. It is now a simple matter to verify that $\mathcal{W}$ encodes a winning strategy: simply build the multicut derivation associated to the table $\mathcal{W}$ and extend it to a complete proof. This later step is essentially the same kind of restricted proof search that is presented for the previous example based on simulation.

8 Conclusions and future work

This paper is part of a project to use focused proofs as a framework for relating a variety of proof representations. Here we showed a connection between tables and sequent calculus proofs. We expect that similar results will also allow us to relate sequent calculus proofs to other proof objects, e.g., the *oracles* of Necula and Rahul [13] and the fixpoints in the *Abstraction Carrying Code* [1].

Clearly, it should be possible to put more in tables than literals: for example, it seems easy to account for universally quantified literals in table. The Twelf system [14, 15] and the Bedwyr system [4] are two examples of implementation of a logic in which tables of atoms are used to improve proof search but where goals can be much richer, including specifically universal quantifiers and implications. It would be interesting to find a way to extend our work here to allow such goal formulas to be tabled as well.

In this paper, we investigated the problem of automating and checking the use of previously proved lemmas (or table) in the proof of some main theorem. After motivating the use of focusing and the polarity of atoms, we presented two focused systems; one involving Horn clauses, and another adding negation-as-failure. We also showed that by using a partial ordering relation over the elements of the table, we could define a multicut derivation which could be easily extended to a proof. With these systems, we were also able to give a declarative interpretation to the *memoization* procedure. And finally, we proposed to use tables as a proof objects and illustrated this with some examples.

Acknowledgments We thank David Baelde and the anonymous reviewers for their comments on an earlier draft of this paper. This work has been supported in part by INRIA through the “Equipes Associées” Slimmer and by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2005-015905 MOBIUS project.

References

1. Elvira Albert, Germn Puebla, and Manuel V. Hermenegildo. Abstraction-carrying code. In *LPAR 2004: Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3452 of *LNCS*, pages 380–397. Springer, 2004.
2. Alan R. Anderson and Nuel D. Belnap. *Entailment: The Logic of Relevance and Necessity.* Princeton University Press, Princeton, NJ, 1975.
3. Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.
4. David Baelde, Andrew Gacek, Dale Miller, Gopalan Nadathur, and Alwen Tiu. The Bedwyr system for model checking over syntactic expressions. To appear in CADE-21, 2007.
5. Gerhard Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1969.
6. Jean-Yves Girard. A fixpoint theorem in linear logic. An email posting to the mailing list linear@cs.stanford.edu, February 1992.
7. Chuck Liang and Dale Miller. Focusing and polarization in intuitionistic logic. To appear in CSL 2007, April 2007.
8. Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000.
9. Raymond McDowell, Dale Miller, and Catuscia Palamidessi. Encoding transition systems in sequent calculus. *Theoretical Computer Science*, 294(3):411–437, 2003.
10. Dale Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1-2):79–108, January 1989.
11. Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
12. George C. Necula. Proof-carrying code. In *Conference Record of the 24th Symposium on Principles of Programming Languages 97*, pages 106–119, Paris, France, 1997. ACM Press.
13. George C. Necula and Shree Prakash Rahul. Oracle-based checking of untrusted software. In *POPL*, pages 142–154, 2001.
14. Frank Pfenning and Carsten Schürmann. System description: Twelf — A meta-logical framework for deductive systems. In H. Ganzinger, editor, *16th Conference on Automated Deduction*, LNAI 1632, pages 202–206, Trento, 1999. Springer.
15. Brigitte Pientka. Tabling for higher-order logic programming. In *20th International Conference on Automated Deduction, Talinn, Estonia*, pages 54 – 69. Springer-Verlag, 2005.
16. Y. S. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, Scott A. Smolka, Terrance Swift, and David Scott Warren. Efficient model checking using tabled resolution. In *Proceedings of the 9th International Conference on Computer Aided Verification (CAV97)*, number 1254 in LNCS, pages 143–154, 1997.
17. Abhik Roychoudhury, C. R. Ramakrishnan, and I. V. Ramakrishnan. Justifying proofs using memo tables. In *PPDP*, pages 178–189, 2000.
18. Peter Schroeder-Heister. Definitional reflection and the completion. In R. Dyckhoff, editor, *Proceedings of the 4th International Workshop on Extensions of Logic Programming*, pages 333–347. Springer-Verlag LNAI 798, 1993.
19. John Slaney. Solution to a problem of Ono and Komori. *Journal of Philosophic Logic*, 18:103–111, 1989.