Fast Decision Procedure for Propositional Dummett Logic Based on a Multiple Premise Tableau Calculus

Guido Fiorino
Dipartimento di Metodi Quantitativi per le Scienze
Economiche ed Aziendali,
Università di Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1,
20126 Milano, Italy.
guido.fiorino@unimib.it

Abstract

We present a procedure to decide propositional Dummett logic. This procedure relies on a tableau calculus with a multiple premise rule and optimizations. The resulting implementation outperforms the state of the art graph-based procedure.

1 Introduction

In this note a tableau calculus, some optimizations and an implementation to decide propositional Dummett logic are described.

Dummett logic can be axiomatized by adding to any proof system for propositional intuitionistic logic the axiom scheme $(p \to q) \lor (q \to p)$. It has a well-known semantical characterization by linearly ordered Kripke models, thus Dummett logic is also known as Linear Chain logic. Gödel studied finite approximations of Dummett Logic ([15]), namely the sequence G_n, $n \geq 1$, of logics that are semantically characterized by linearly ordered Kripke models with at most n worlds. For this reason another name for the logic under consideration is Gödel-Dummett Logic. Dummett Logic has been considered by people interested in computer science [4, 5] and many valued logics [7, 8]. In [17] it has been recognized as an important fuzzy logic. We quote [9] for a thorough treatment of the subject.

In the late '90 were presented tableau [1, 12] and sequent calculi [11] for propositional Dummett logic. These calculi are believed to be highly inefficient to the purpose of performing practical theorem proving ([5, 20]), mainly because they contain a multiple premise rule that in the worse case analysis gives rise to tableau proofs having a factorial number of branches with respect to the number of formulas in the premise. To get rid of the multiple premise rule, paper [5] proposes to exploit the following logical equivalences: $A \to (B \lor C) \equiv (A \to B) \lor (A \to C)$, $A \to (B \land C) \equiv (A \to B) \land (A \to C)$, $(A \lor B) \to C \equiv (A \to C) \land (B \to C)$ and $(A \land B) \to C \equiv (A \to C) \lor (B \to C)$. In the recent [21] it is proved that the deduction in Dummett logic can be reduced to the construction of a graph and an implementation has been developed.

Recent investigations in propositional intuitionist logics have introduced optimization techniques that dramatically reduce the running time ([2]). Such techniques can be extended to other logics. The aim of this note is to show that the tableau calculi based on a multiple premise rule plus optimization techniques give rise to a fast decision procedure for propositional Dummett logic. Our fast implementation is based on three main ideas that are motivated in this note: (i) a new tableau calculus. Compared with [1, 12], our tableau calculus provides a new treatment of negated formulas and a new multiple premise rule; (ii) the optimization technique Simplification, first described in [22]. This optimization has been employed in [2] to improve automated deduction in propositional intuitionistic logic; (iii) the equivalences for Dummett

$$\frac{S,\mathbf{T}(A\land B)}{S,\mathbf{T}A,\mathbf{T}B}\mathbf{T}\land \quad \frac{S,\mathbf{F}(A\land B)}{S,\mathbf{F}A|S,\mathbf{F}B}\mathbf{F}\land \quad \frac{S,\mathbf{F_c}(A\land B)}{S,\mathbf{F_c}A|S,\mathbf{F_c}B}\mathbf{F_c}\land \quad \frac{S,\mathbf{T_{cl}}(A\land B)}{S,\mathbf{T_{cl}}A,\mathbf{T_{cl}}B}\mathbf{T_{cl}}\land$$

$$\frac{S,\mathbf{T}(A\lor B)}{S,\mathbf{T}A|S,\mathbf{T}B}\mathbf{T}\lor \quad \frac{S,\mathbf{F}(A\lor B)}{S,\mathbf{F}A,\mathbf{F}B}\mathbf{F}\lor \quad \frac{S,\mathbf{F_c}(A\lor B)}{S,\mathbf{F_c}A,\mathbf{F_c}B}\mathbf{F_c}\lor \quad \frac{S,\mathbf{T_{cl}}(A\lor B)}{S,\mathbf{T_{cl}}A|S,\mathbf{T_{cl}}B}\mathbf{T_{cl}}\lor$$

$$\frac{S,\mathbf{T}(\neg A)}{S,\mathbf{F_c}A}\mathbf{T}\neg \quad \frac{S,\mathbf{F}(\neg A)}{S,\mathbf{T_{cl}}A}\mathbf{F}\neg \quad \frac{S,\mathbf{F_c}(\neg A)}{S,\mathbf{T_{cl}}A}\mathbf{F_c}\neg \quad \frac{S,\mathbf{T_{cl}}(\neg A)}{S,\mathbf{F_c}A}\mathbf{T_{cl}}\neg$$

$$\frac{S,\mathbf{T}A,\mathbf{T}(A\to B)}{S,\mathbf{T}A,\mathbf{T}B}\mathbf{T}\to \quad \frac{S,\mathbf{F_c}(A\to B)}{S,\mathbf{T_{cl}}A,\mathbf{F_c}B}\mathbf{F_c}\to \quad \frac{S,\mathbf{T_{cl}}(A\to B)}{S,\mathbf{F_c}A|S,\mathbf{T_{cl}}B}\mathbf{T_{cl}}\to$$

$$\frac{S,\mathbf{T}((A\land B)\to C)}{S,\mathbf{T}(A\to(B\to C))}\mathbf{T}{\to}\land \quad \frac{S,\mathbf{T}(\neg A\to B)}{S,\mathbf{T_{cl}}A|S,\mathbf{T}B}\mathbf{T}{\to}\neg \quad \frac{S,\mathbf{T}((A\lor B)\to C)}{S,\mathbf{T}(A\to C),\mathbf{T}(B\to C)}\mathbf{T}{\to}\lor$$

$$\frac{S,\mathbf{T}((A\to B)\to C)}{S,\mathbf{F}(A\to p),\mathbf{T}(p\to C),\mathbf{T}(B\to p)|S,\mathbf{T}C}\mathbf{T}{\to}{\to}\ \text{with } p \text{ a new atom}$$

Figure 1: The invertible rules of $\mathbb{D}$.

logic quoted above. The equivalences are employed to reduce the number of branches generated by the multiple premise rule. They are used in the opposite style with respect to [5, 6, 21]. On the ideas described in this note (full details with proofs in [13]) we have developed a prolog implementation that is faster than the state of the art graph-based procedure of [21].

2 Basic Definitions

We consider the propositional language based on a denumerable set of propositional variables $\mathcal{PV}$, the boolean constants $\top$ and $\bot$ and the logical connectives $\neg, \land, \lor, \to$. In the following, formulas (respectively set of formulas and propositional variables) are denoted by letters A, B, C... (respectively S, T, U,... and p, q, r,...) possibly with subscripts or superscripts.

A well-known semantical characterization of Dummett logic (**Dum**) is by *linearly ordered Kripke models*. In this note *model* means a linearly ordered Kripke model, namely a structure $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$, where $\langle P,\leq,\rho\rangle$ is a linearly ordered set with minimum ρ and $\Vdash$ is the *forcing relation*, a binary relation on $P\times(\mathcal{PV}\cup\{\top,\bot\})$ such that: (i) if $\alpha\Vdash p$ and $\alpha\leq\beta$, then $\beta\Vdash p$; (ii) for every $\alpha\in P$, $\alpha\Vdash\top$ holds and $\alpha\Vdash\bot$ does not hold. Hereafter we denote the members of P with lowercase letters of the Greek alphabet.

The forcing relation is extended in a standard way to arbitrary formulas as follows: (i) $\alpha\Vdash A\land B$ iff $\alpha\Vdash A$ and $\alpha\Vdash B$; (ii) $\alpha\Vdash A\lor B$ iff $\alpha\Vdash A$ or $\alpha\Vdash B$; (iii) $\alpha\Vdash A\to B$ iff, for every $\beta\in P$ such that $\alpha\leq\beta$, $\beta\Vdash A$ implies $\beta\Vdash B$; (iv) $\alpha\Vdash\neg A$ iff, for every $\beta\in P$ such that $\alpha\leq\beta$, $\beta\Vdash A$ does not hold.

We write $\alpha\nVdash A$ when $\alpha\Vdash A$ does not hold. It is easy to prove that, for every formula A, the *persistence* property holds: if $\alpha\Vdash A$ and $\alpha\leq\beta$, then $\beta\Vdash A$. A formula A *is valid in a model* $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$ if and only if $\rho\Vdash A$. It is well-known (see e.g. [10]) that **Dum** coincides with the set of formulas valid in all models.

The rules of our calculus $\mathbb{D}$ for **Dum** are in Figures 1 and 2. The rules of $\mathbb{D}$ work on signed formulas, that is well-formed formulas prefixed with one of the *signs* $\{\mathbf{T},\mathbf{F},\mathbf{F_c},\mathbf{T_{cl}}\}$, and on sets of signed formulas (hereafter we omit the word "signed" in front of "formula" in all the contexts where no confusion arises).

The semantical meaning of the signs is explained by means of the *realizability* relation ($\triangleright$) defined as follows. Let $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$ be a model, let $\alpha\in P$, let H be a signed formula and let S be a set of signed formulas. We say that α *realizes* H, α *realizes* S and $\underline{K}$ *realizes* S, and

$$\frac{S, \mathbf{T_{cl}}A}{S_c, \mathbf{T}A}\,\mathbf{T_{cl}}\text{-Atom}$$

$$\frac{S, \mathbf{F}(A_1 \to B_1), \ldots, \mathbf{F}(A_n \to B_n)}{S_c, \mathbf{T}A_1, \mathbf{F}B_1, S^1_{\mathbf{F}\to} | S_c, \mathbf{T}A_2, \mathbf{F}B_2, S^2_{\mathbf{F}\to} | \ldots | S_c, \mathbf{T}A_n, \mathbf{F}B_n, S^n_{\mathbf{F}\to}}\,\mathbf{F}\!\to$$

$$\begin{array}{lcl} S_c & = & \{\mathbf{T}A | \mathbf{T}A \in S\} \cup \{\mathbf{F_c}A | \mathbf{F_c}A \in S\} \cup \{\mathbf{T_{cl}}A | \mathbf{T_{cl}}A \in S\}; \\ S^1_{\mathbf{F}\to} & = & \{\mathbf{F}(A_2 \to B_2), \ldots, \mathbf{F}(A_n \to B_n)\}, \\ S^i_{\mathbf{F}\to} & = & \{\mathbf{F}((A_1 \wedge B_i) \to B_1), \ldots, \mathbf{F}((A_{i-1} \wedge B_i) \to B_{i-1}), \mathbf{F}(A_{i+1} \to B_{i+1}), \ldots, \mathbf{F}(A_n \to B_n)\}, \\ & & \text{for } i = 2, \ldots, n. \end{array}$$

Figure 2: The non-invertible rules of $\mathbb{D}$.

we write $\alpha \rhd H$, $\alpha \rhd S$ and $\underline{K} \rhd S$, respectively, if the following conditions hold:

1. $\alpha \rhd \mathbf{T}A$ iff $\alpha \Vdash A$;
2. $\alpha \rhd \mathbf{F}A$ iff $\alpha \nVdash A$;
3. $\alpha \rhd \mathbf{F_c}A$ iff $\alpha \Vdash \neg A$;
4. $\alpha \rhd \mathbf{T_{cl}}A$ iff $\alpha \Vdash \neg\neg A$;
5. $\alpha \rhd S$ iff α realizes every formula in S;
6. $\underline{K} \rhd S$ iff $\rho \rhd S$.

Since the meaning of $\mathbf{T}$, $\mathbf{F_c}$ and $\mathbf{T_{cl}}$ is related to the forcing of a formula and since, by the persistence property, the forced formulas are preserved upwards, we call *stable* the formulas signed with $\mathbf{T}$, $\mathbf{F_c}$ or $\mathbf{T_{cl}}$. As discussed in the following, stable formulas have a central role in the organization of our deduction strategy. We point out that $\mathbf{F_c}A$ and $\mathbf{T_{cl}}A$ are synonym $\mathbf{T}\neg A$ and $\mathbf{T}\neg\neg A$ respectively. If $\mathbf{F_c}A$ holds in a world of a Kripke model, then A is *certainly not forced in the future.* If $\mathbf{T_{cl}}A$ holds in a world of a Kripke model $\underline{K}$, then A is forced in the maximum of $\underline{K}$, which is a world semantically behaving as a *classical model.* We could rewrite the rules of the calculus by using only the signs $\mathbf{T}$ and $\mathbf{F}$. In such a calculus the $\mathbf{F_c}$-rules are replaced by rules treating negated formulas and the rules for $\mathbf{T_{cl}}$ are replaced by rules treating double negated formulas. We prefer this object language because it makes the rules less cumbersome than the object language with two signs only.

From the meaning of the signs we get the conditions that make a set of formulas inconsistent. A set S is *inconsistent* iff $\{\mathbf{T}A, \mathbf{F}A\} \subseteq S$, $\{\mathbf{T}A, \mathbf{F_c}A\} \subseteq S$, $\{\mathbf{F_c}A, \mathbf{T_{cl}}A\} \subseteq S$, $\mathbf{T}\bot \in S$, $\mathbf{F}\top \in S$, $\mathbf{F_c}\top \in S$ or $\mathbf{T_{cl}}\bot \in S$. It is easy to prove the following result:

Theorem 1. *If a set of formulas S is inconsistent, then for every Kripke model $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ and for every $\alpha \in P$, $\alpha \not\rhd S$.*

We refer to [16] for a full presentation of tableaux systems. A *closed proof table* is a proof table whose leaves are all inconsistent sets. A closed proof table is a proof of the calculus and a formula A is provable iff there exists a closed proof table for $\{\mathbf{F}A\}$.

The calculus $\mathbb{D}$ has two non-invertible rules, namely $\mathbf{F} \to$ and $\mathbf{T_{cl}}$-Atom. Rule $\mathbf{F} \to$ is inspired to the rule of [1]. Rule $\mathbf{T_{cl}}$-Atom can be explained as follows: let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a model and let $\alpha \in P$ such that $\alpha \Vdash \neg\neg p$. Let ϕ the maximum with respect to $\langle P, \leq, \rho \rangle$.

Then $\phi \Vdash p$. We notice that nothing can be concluded about the forcing in α of p, whereas if $\alpha \Vdash \neg\neg A$, with A a non-atomic formula, we have information about the forcing in α of the subformulas of A. This explains the $\mathbf{T_{cl}}$-rules in Figure 1.

3 Correctness

To prove the correctness of $\mathbb{D}$ with respect to Dummett logic we need to prove that, if there exists a closed proof table for $\{\mathbf{F}A\}$, then A is a valid formula in Dummett logic. The main step is to prove that the rules of the calculus preserve realizability:

Proposition 1. *For every rule of $\mathbb{D}$, if a model realizes the premise, then there exists a model realizing at least one of the conclusions.*

Proof: we provide the proof for $\mathbf{F} \rightarrow$, the other cases being trivial. If the premise of $\mathbf{F} \rightarrow$ is realized, then there exist $\alpha_1, \dots, \alpha_n$ elements of a model $\underline{K}$ such that $\alpha_i \rhd \mathbf{T}A_i, \mathbf{F}B_i$, for $i = 1, \dots, n$. Let $\beta_i = \max\{\alpha | \alpha \rhd \mathbf{T}A_i, \mathbf{F}B_i\}$, for $i = 1, \dots, n$, thus, β_i is the maximal element such that $\beta_i \rhd \mathbf{T}A_i, \mathbf{F}B_i$. The structure $\underline{K}' = \langle\{\beta_1, \dots, \beta_n\}, \leq, \rho', \Vdash\rangle$, with $\rho' = \min\{\beta_1, \dots, \beta_n\}$, is a model such that $\rho' \rhd S_c, \mathbf{T}A_j, \mathbf{F}B_j, S_{\mathbf{F}\rightarrow}$ for some $j \in \{1, \dots, n\}$. Let $m = \min\{i \in \{1, \dots, n\} | \beta_i = \rho'\}$. If $m = 1$, then the leftmost conclusion of $\mathbf{F} \rightarrow$ is realized. If $m > 1$, then $\beta_1, \dots, \beta_{m-1} > \beta_m$. Let $\beta_k = \min\{\beta_1, \dots, \beta_{m-1}\}$. It follows that $\beta_k \rhd \mathbf{T}A_k, \mathbf{F}B_k, \mathbf{T}B_m$. Thus $\beta_m \rhd \mathbf{F}((A_i \wedge B_m) \rightarrow B_i)$, for $i = 1, \dots, m-1$ and this implies that the m-th conclusion of the rule is realized. □

The idea behind the rule $\mathbf{F} \rightarrow$ can be explained as follows: If the j-th conclusion of the rule **Dum** of [1] is realizable and no model realizes the first $j-1$ conclusions, then $\alpha_j > \alpha_1, \dots, \alpha_{j-1}$ holds. This also implies that $\alpha_1, \dots, \alpha_{j-1} \rhd \mathbf{T}B_j$, hence $\alpha_j \rhd \mathbf{F}((A_k \wedge B_j) \rightarrow B_k)$, for $k = 1, \dots, j-1$ and this proves that the j-th conclusion of $\mathbf{F} \rightarrow$ is realized.

Remark 1. We call *side information* the formula B_j added in the j-th conclusion of $\mathbf{F} \rightarrow$ as conjunct in $\mathbf{F}((A_i \wedge B_j) \rightarrow B_i)$, for $i = 1, \dots, j-1$. The side information arises from the knowledge that the left-hand side conclusions are not realizable. This information is correct but it is not necessary to get the completeness. The notion of side information is introduced to reduce the search space by means of the *simplification technique* which is described in Section 4.

Remark 2. The rule $\mathbf{F} \rightarrow$ can also be given in the following form

$$\frac{S, \mathbf{F}(A_1 \rightarrow B_1), \dots, \mathbf{F}(A_n \rightarrow B_n)}{S_c, \mathbf{T}A_1, \mathbf{F}B_1, S^1_{\mathbf{F}\rightarrow} | S_c, \mathbf{T}A_2, \mathbf{F}B_2, S^2_{\mathbf{F}\rightarrow} | \dots | S_c, \mathbf{T}A_n, \mathbf{F}B_n, S^n_{\mathbf{F}\rightarrow}} \mathbf{F}_{\rightarrow new}$$

$$\begin{array}{lcl} S_c & = & \{\mathbf{T}A | \mathbf{T}A \in S\} \cup \{\mathbf{F_c}A | \mathbf{F_c}A \in S\} \cup \{\mathbf{T_{cl}}A | \mathbf{T_{cl}}A \in S\}; \\ S^1_{\mathbf{F}\rightarrow} & = & \{\mathbf{F}(A_2 \rightarrow B_2), \dots, \mathbf{F}(A_n \rightarrow B_n)\}, \\ S^i_{\mathbf{F}\rightarrow} & = & \{\mathbf{T}(p \rightarrow B_i), \mathbf{F}(A_1 \wedge p \rightarrow B_1), \dots, \mathbf{F}(A_{i-1} \wedge p \rightarrow B_{i-1}), \mathbf{F}(A_{i+1} \rightarrow B_{i+1}), \dots, \mathbf{F}(A_n \rightarrow B_n)\}, \\ & & \text{with } p \text{ a new propositional variable,} \\ & & \text{for } i = 2, \dots, n. \end{array}$$

The correctness of $\mathbf{F} \rightarrow$-new can be easily obtained from $\mathbf{F} \rightarrow$ following the proof of correctness given in [12] for the rules $\mathbf{T} \rightarrow\rightarrow$ and $\mathbf{T} \rightarrow \vee$ (it can also be noticed that it is applied the indexing technique consisting in replacing a formula with a new propositional variable). This version highlights that the side information B_i is treated by the rules of the calculus once.

By the above proposition:

Theorem 2 (Soundness of $\mathbb{D}$)**.** *If there exists a proof of a formula A, then A is valid in every model.*

$$\frac{S, \mathbf{T}A}{S[A/\top], \mathbf{T}A}\text{Replace}\,\mathbf{T} \qquad \frac{S, \mathbf{F_c}A}{S[A/\bot], \mathbf{F_c}A}\text{Replace}\,\mathbf{F_c} \qquad \frac{S, \mathbf{T_{cl}}A}{S[\neg A/\bot], \mathbf{T_{cl}}A}\text{Replace}\,\mathbf{T_{cl}}$$

Figure 3: Replacement rules

4 Rules to Optimize the Proof Search

The tableau rules in Figures 1 and 2 are the core of $\mathbb{D}$. Now we discuss some rules, introduced to reduce the size of the proofs. We note that $\mathbb{D}$ improves the known multiple premise calculi [1, 12] by two aspects: the rule $\mathbf{F}\rightarrow$ and the rules to treat $\mathbf{T_{cl}}$-formulas.

Simplification Simplification is an effective optimization both in classical and intuitionistic logic. In the framework of tableau systems Simplification has been introduced in [22], where it is applied to classical and modal logics. Recently it has been fitted to intuitionistic logic ([2]). Simplification is based on the well-known replacement rules (see e.g. [19]) consisting in replacing a formula with a logically equivalent one. Our adaptation of Simplification to the object language of $\mathbb{D}$ consists in the replacement rules of Figure 3. It is an easy task to check that the replacement rules preserve the realizability and are invertible. The logical constant $\top$ and $\bot$ are replaced by means of the *simplification rules* consisting in the usual boolean simplification rules for conjunction and disjunction, plus the simplification rules for intuitionistic negation and implication.

The semantical meaning of $\top$ and $\bot$ implies that replacements affect neither the correctness nor the completeness, thus these replacements rules can be applied at any step of a tableau proof. Hereafter with Simplification we mean the set of rules described in this section, including the usual boolean simplification rules. We point out that the use of Simplification can reduce considerably the search-space ([22] and [2] give an account for propositional classical and intuitionistic logics, respectively).

Reducing the branching of the multiple premise rule $\mathbf{F}\rightarrow$ In [21] the equivalences

$$A \rightarrow (B \vee C) \equiv (A \rightarrow B) \vee (A \rightarrow C),\ \ A \rightarrow (B \wedge C) \equiv (A \rightarrow B) \wedge (A \rightarrow C)$$

$$(A \vee B) \rightarrow C \equiv (A \rightarrow C) \wedge (B \rightarrow C),\ \text{and}\ (A \wedge B) \rightarrow C \equiv (A \rightarrow C) \vee (B \rightarrow C)$$

are exploited to get the rules of Figure 4.

$$\frac{S, \mathbf{T}(A \rightarrow (B \vee C))}{S, \mathbf{T}(A \rightarrow B)|S, \mathbf{T}(A \rightarrow C)} \quad \frac{S, \mathbf{T}((A \vee B) \rightarrow C)}{S, \mathbf{T}(A \rightarrow C), \mathbf{T}(B \rightarrow C)} \quad \frac{S, \mathbf{T}((A \wedge B) \rightarrow C)}{S, \mathbf{T}(A \rightarrow C)|S, \mathbf{T}(B \rightarrow C)} \quad \frac{S, \mathbf{T}(A \rightarrow (B \wedge C))}{S, \mathbf{T}(A \rightarrow B), \mathbf{T}(A \rightarrow C)}$$

$$\frac{S, \mathbf{F}(A \rightarrow (B \wedge C))}{S, \mathbf{F}(A \rightarrow B)|S, \mathbf{F}(A \rightarrow C)} \quad \frac{S, \mathbf{F}(A \rightarrow (B \vee C))}{S, \mathbf{F}(A \rightarrow B), \mathbf{F}(A \rightarrow C)} \quad \frac{S, \mathbf{F}((A \vee B) \rightarrow C)}{S, \mathbf{F}(A \rightarrow C)|S, \mathbf{F}(B \rightarrow C)} \quad \frac{S, \mathbf{F}((A \wedge B) \rightarrow C)}{S, \mathbf{F}(A \rightarrow C), \mathbf{F}(B \rightarrow C)}$$

Figure 4: Rules of [6, 21] to treat implicative formulas.

The rules of Figure 4 allow to reduce implicative formulas to implicative atomic formulas, that is implicative formulas whose antecedent and consequent are propositional variables. The

$$\frac{\mathbf{F}(A \to B), \mathbf{F}(A \to C)}{\mathbf{F}(A \to (B \vee C))} \qquad \frac{\mathbf{F}(A \to C), \mathbf{F}(B \to C)}{\mathbf{F}((A \wedge B) \to C)}$$

Figure 5: The factorization rules of $\mathbb{D}$.

problem of deciding a set of formulas containing atomic implicative formulas and atomic formulas only is reducible to the problem of reachability on a graph [21]. To get sets whose implicative formulas are atomic a price has to be paid. First, it is always necessary to treat **T**-implicative formulas, and in the case of formulas of the kind $\mathbf{T}(p \to (B \vee C))$ and $\mathbf{T}((A \wedge B) \to C)$ branches are generated. Multiple premise calculi treat formulas of the kind $\mathbf{T}((A \wedge B) \to C)$ and $\mathbf{T}(p \to (B \vee C))$ by means of single-conclusion rules. Second, the **F**-rules above decompose the $\mathbf{F} \to$-formulas either by increasing their number in the conclusion or by introducing a new branch. Note also that such rules do not introduce in the conclusions any stable information. As we discussed in the previous section, we consider useful from a practical perspective to discover stable information in order to reduce the search space.

With regard to the $\mathbf{F} \to$-formulas, the calculi of [6, 21] and $\mathbb{D}$ give rise to decision procedures behaving in the opposite way. The calculi provided in [6, 21] employ the rules in Figure 4 to insert in a set all the possible $\mathbf{F} \to$-formulas. The calculus $\mathbb{D}$ employs the rules in Figure 5 to reduce the number of $\mathbf{F} \to$-formulas in the sets. In the case of $\mathbb{D}$ the reason to reduce this number is related to the presence of the multiple premise rule $\mathbf{F} \to$, whose number of conclusions depends on the $\mathbf{F} \to$-formulas in its premise. We aim to devise a decision procedure using the rules in Figure 5 in order to reduce as much as possible the number of $\mathbf{F} \to$-formulas before to apply the rule $\mathbf{F} \to$.

The rules to handle $\mathbf{T_{cl}}$-formulas The calculus $\mathbb{D}$ has rules for the formulas signed with $\mathbf{T_{cl}}$, this amounts to have ad-hoc rules to treat $\mathbf{T}\neg\neg$-formulas, a kind of formulas for which [1] does not have ad-hoc rules but takes back to the rule $\mathbf{F_c}\neg$.

To avoid backtracking still preserving the completeness, in [1] the application of the rule $\mathbf{F_c}$ has to be deferred until no **F**-rule is applicable. This means that the calculus defers to analyze the stable formulas of the kind $\mathbf{F_c}(\neg A)$ (in other words the calculus does not analyze the double negated formulas). On the contrary $\mathbb{D}$ analyzes the double negated formulas by means of $\mathbf{T_{cl}}$-rules. The advantage of analyzing formulas of the kind $\mathbf{F_c}(\neg A)$ for every case of A is that the discovery of new stable subformulas of $\mathbf{F_c}(\neg A)$ is not deferred to a stage when no other rule, included the rule $\mathbf{F} \to$, is applicable. The early discovery of stable formulas gives advantages since this information allows to shrink the search space by means of Simplification. Summarizing, the $\mathbf{T_{cl}}$-rules of $\mathbb{D}$ are the rules allowing to analyze $\mathbf{F_c}\neg$-formulas. Among them $\mathbf{T_{cl}}$-Atom is the only non-invertible rule. In Section 5 it is proved that to avoid backtracking it is sufficient to defer the application of $\mathbf{T_{cl}}$-Atom until no other rule is applicable.

We notice that the side information can also be characterized into the logic calculus by introducing appropriate connectives and signs along the ideas presented in Remark 4.

5 A Strategy to Decide Dummett Logic and Its Completeness

The implementation described in the next section uses the rules of the calculus according the following strategy. The non-invertible rules $\mathbf{F} \to$ and $\mathbf{T_{cl}}$-Atom are applied only when no invertible rule of Figures 1-3 is applicable. This strategy is necessary to avoid backtracking in proof search. Rule $\mathbf{F} \rightarrow$ is applied when no rule but $\mathbf{T_{cl}}$-Atom is applicable. This guarantees that the application of the non-invertible rule $\mathbf{F} \rightarrow$ is invertible, that is, no information necessary to the completeness is lost. Rules in Figure 5 allow to reduce the number of $\mathbf{F} \rightarrow$-formulas to be handled by $\mathbf{F} \rightarrow$, thus are applied before to apply $\mathbf{F} \rightarrow$. Finally rule $\mathbf{T_{cl}}$-Atom is applied. Every application of $\mathbf{T_{cl}}$-Atom is invertible because the $\mathbf{F}$-atomic formulas is the only information which is lost.

The decision procedure can be implemented in polynomial space by means of a depth-first strategy. Because to the side information introduced in the conclusion of $\mathbf{F} \rightarrow$, the depth of the deduction is quadratic in the formulas to be proved. As a matter of fact, to insert the side information, in every set S of the conclusion of $\mathbf{F} \rightarrow$ are introduced as many $\wedge$ connectives as the $\mathbf{F} \rightarrow$-formulas in the premise. This number is bounded by the size n of the formula to be decided. If the data structures of the implementation allow to store the formulas once, that is, independently of the number of their occurrences, then this is the new information that needs to be stored. Along a branch the number of applications of rule $\mathbf{F} \rightarrow$ is bounded by n. Since every application of $\mathbf{F} \rightarrow$ introduces in a set n new connectives $\wedge$ at most, it follows that to handle the side information, along a branch are introduced n^2 new $\wedge$ connectives at most. As regards the rules treating the side information, we point out that the sign of a side information is $\mathbf{T}$. This implies that all its occurrences in the set are replaced by $\top$. If the data structures represents the different occurrences of a formula once, then the replacement takes constant time. Every application of the simplification rules deletes one connective and takes constant time. The analysis of the other rules is straightforward, thus we conclude that, since along a branch the rule $\mathbf{F} \rightarrow$ introduces n^2 connectives $\wedge$, the depth of the deductions is at most quadratic. We also note that although the depth of a branch is $O(n^2)$, only $O(n)$ rules from Figures 1 and 2 are applied. This implies that as in the calculus of [1], the number of branches of the deductions is factorial in n.

Remark 3. An alternative to our approach is to decide Dummett logic via a decision procedure for propositional Intuitionistic logic. Paper [3] introduces the notion of Generalized Tableaux to decide intermediate logics. A Generalized Tableau is a tableau for propositional Intuitionistic logic plus a rule to be applied once as first rule of the deduction. The aim of this rule is to introduce formulas obtained by instantiating the axiom scheme of the logic under consideration. For the case of Dummett logic, to decide a given formula A, the special rule introduces the set of formulas obtained by instantiating in every possible way the propositional variables the axiom schemata $(p \rightarrow q) \vee (q \rightarrow p)$ with the formulas in $Rsf(A)=\{B|B$ is subformula of A and B is a propositional variable or $B \equiv C \rightarrow D$ or $B \equiv \neg C\}$. Since $|Rsf(A)| = O(|A|)$ and there are $|Rsf(A)|$ choices for p and q, it follows that the special rule introduces $O(|A|^2)$ formulas ($|A|$ denotes the cardinality of A). Thus the number of connectives to be handled in the deduction is $O(|A|^3)$. Paper [18] proves that propositional intuitionistic logic is decidable in $O(n \lg n)$-SPACE, hence this technique requires $O(n^3 \lg n)$-SPACE and the depth of the deductions is $O(|A|^3)$.

6 The Implementation and the Performances

We devote this section to give an account of our implementation EPDL[1]. The first issue we face is how EPDL handles the side information. We emphasize that the side information B_j has sign $\mathbf{T}$, this means that such instances of B_j will occur in the subsequent sets with sign $\mathbf{T}$. The

[1]EPDL is downloadable from `http://www.dimequant.unimib.it/~guidofiorino/epdl.jsp`.

formula B_j occurring in $\mathbf{F}((A_i \wedge B_j) \to B_i)$ conveys the information that when $\mathbf{T}A_i$ and $\mathbf{F}B_i$ are realized, also $\mathbf{T}B_j$ has to be realized. Since the rules of Simplification are used, we want use the stable information $\mathbf{T}B_j$ and possibly the information derivable from $\mathbf{T}B_j$ to reduce the size of the proofs. The side information is not necessary to get the completeness of the calculus, thus there are different ways to handle it, both in the logical calculus and in the implementation. It has to be noticed that if the side information is not treated properly, then there can be disadvantages. For example, in the implementation we can decide not to apply to the side information rules having two conclusions but only the rules having one, so useless branching is not introduced into the proof (thus, if B_j is of the kind $C \vee D$, then the rule $\mathbf{T}\vee$ is not applied; if B_j is of the kind $C \wedge D$, then the rule $\mathbf{T}\wedge$ is applied to deduce the two stable formulas $\mathbf{T}C$ and $\mathbf{T}D$). With respect to the efficiency, there is another remark that deserves attention and we have considered in the implementation but not in the presentation of the calculus. Let us consider the case that starting from the j-th conclusion of $\mathbf{F}\to$, $j > 1$, in a subsequent step the formula $\mathbf{T}A_i$ ($i \in \{1, \dots, n\}$ and $i \neq j$) is inserted. By Simplification the formula $\mathbf{F}((A_i \wedge B_j) \to B_i)$ becomes $\mathbf{F}(B_j \to B_i)$ and a useless branch arises. The branch is useless because to get the completeness only $\mathbf{F}B_i$ is necessary (note that this is the formula we get if we apply the rule **Dum**). In the presentation of the logical calculus we have decided not to be concerned about these aspects related to the efficiency. We have faced them at the development stage. To avoid the disadvantages quoted above the new connective % is introduced. The aim of % is to identify conjunctive formulas whose right operand is a side formula. Since the right operand of % is the side information, the truth value of % depends on its left operand. The rules treating formulas of the kind $A\%B$ behave as the rules for conjunctive formulas, thus $\dfrac{\mathbf{T}(A\%B)}{\mathbf{T}A, \mathbf{T}B}\mathbf{T}\%$ is an additional rule of the implementation. Beside the rule $\mathbf{T}\%$ there are the Simplification rules related to %, all behaving as the Simplification rules for $\wedge$ except for the case $\top\%B$ handled by the rule

$$\frac{S}{S[\top\%B/\top]}\text{Simp}\,\top\%.$$

Rules for the formulas of the kind $\mathcal{S}(\mathcal{A}\%\mathcal{B})$, with $\mathcal{S} \in \{\mathbf{F}, \mathbf{F_c}, \mathbf{T_{cl}}\}$, do not need to be implemented. Indeed, $A\%B$ always occurs in the antecedent of an $\mathbf{F}\to$-formula and this implies that in the proof table the formula $A\%B$ occurs with sign $\mathbf{T}$.

Remark 4. After the rule $\mathbf{T}\%$ is applied, EPDL treats the side information as a standard formula. The subsequent rules applied to side information can give rise to useless branches. To avoid the generation of branches, another implementation is possible along the following ideas: (i) introduce a new sign $\tilde{\mathbf{T}}$ to mark the side information; (ii) treat $\mathbf{T}(A\%B)$ by the rule $\dfrac{\mathbf{T}(A\%B)}{\mathbf{T}A, \tilde{\mathbf{T}}B}\mathbf{T}\%_{-new}$; (iii) treat the $\tilde{\mathbf{T}}$-formulas by the rule $\dfrac{\tilde{\mathbf{T}}(A \wedge B)}{\tilde{\mathbf{T}}A, \tilde{\mathbf{T}}B}\mathbf{T}\%$. We recall the side information is not necessary to preserve the completeness. Thus the $\tilde{\mathbf{T}}$-rules for the remaining connectives are not needed; (v) introduce the rule $\dfrac{S, \tilde{\mathbf{T}}A}{S[A/\top], \tilde{\mathbf{T}}A}\text{Replace}\,\tilde{\mathbf{T}}$ to exploit the stable information conveyed by the side information A.

Stable formulas convey information related to the preservation of the forcing relation. In order to exploit Simplification as much as possible, our strategy is to treat as soon as possible all the stable information. In particular, the choice of the rule $\mathbf{F}\wedge$ is delayed until no other rule in Figure 1 is applicable.

Formula	LC-models	EPDL
Nish.9	9	0.01
Nish.10	49	0.01
Nish.11	192	0.01
Nish.12	895	0.01
GdeJ.9	3	0.07
GdeJ.10	6	0.11
GdeJ.11	11	0.15
GdeJ.12	21	0.19
Fin.97	224	5.02
Fin.98	230	5.16
Fin.99	238	5.32
Fin.100	245	5.61
201.2	43.16	0.21
201.3	362	3.14
201.4	2183	35.96
201.5	12186	378

Formula	LC-models	EPDL
202.2	0.06	0.01
202.3	0.41	0.02
202.4	4.16	0.15
202.5	43.79	1.18
204.17	14.10	0.01
204.18	17.80	0.01
204.19	22.12	0.01
204.20	27.06	0.01
205.3	61.61	0.03
205.4	214	0.08
205.5	762	0.24
205.6	2932	0.68
206.2	0.16	0.01
206.3	28.74	0.01
206.4	1683	0.01
206.5	N.A.	0.01
(207.3)	46	0.31
(207.4)	200	3.71
(207.5)	805	40.23
(207.6)	3280	406

Formula	LC-models	EPDL
(208.1)	0.02	0.01
(208.2)	0.52	0.01
(208.3)	21.46	0.02
(208.4)	442	0.06
(210.17)	16.98	0.01
(210.18)	21.20	0.02
(210.19)	26	0.02
(210.20)	31.48	0.02
(211.17)	381	0.20
(211.18)	458	0.24
(211.19)	522	0.28
(211.20)	589	0.32
(212.1)	0.03	0.01
(212.2)	2.00	0.01
(212.3)	133	0.01
(212.4)	1589	0.01

Figure 6: Time comparison between LC-models and EPDL.

In Figure 6 we compare LC-models [2] based on [21] with our implementation EPDL[3]. The formulas considered come from two sources. The first three families are formulas characterizing intermediate logics (see [10, 14])[4]. Nish stands for Nishimura formulas, defined as follows: $\text{Nish}_1 = p$; $\text{Nish}_2 = \neg p$; $\text{Nish}_3 = \neg\neg p$; $\text{Nish}_4 = \neg\neg p \to p$; $\text{Nish}_k = \text{Nish}_{k-1} \to (\text{Nish}_{k-3} \vee \text{Nish}_{k-4})$, $(k \geq 5)$. GdeJ refers to Gabbay de-Jongh formulas, semantically characterized by k-ary trees Kripke models and whose definition is the following: $\text{GdeJ}_k = \bigwedge_{i=0}^{k}((p_i \to \bigvee_{i \neq j} p_j) \to \bigvee_{i \neq j} p_j) \to \bigvee_{i=0}^{k} p_i$ $(k \geq 1)$. Finally, the shortening Fin stands for the following sequence of formulas: $\text{Fin}_1 = \neg p_1 \vee \neg\neg p_1$, $\text{Fin}_2 = \neg p_1 \vee (\neg p_1 \to \neg p_2) \vee (\neg p_1 \to \neg\neg p_2)$, $\text{Fin}_k = \neg p_1 \vee (\neg p_1 \to \neg p_2) \vee (\neg p_1 \wedge \neg p_2 \to \neg p_3) \vee \cdots \vee (\neg p_1 \wedge \cdots \wedge \neg p_{k-1} \to \neg\neg p_k)$ $(k \geq 3)$, which is valid on Kripke models whose poset has at most k maximal elements.
The other families are formulas of ILTP library [23] (in the tables the names have been shortened to save space and brackets around benchmark names denote unprovable formulas in **Dum**). Considering the timings, EPDL is a clear winner in all the families. It is interesting to analyze the growing ratio within each family. The growing ratio of EPDL is higher than LC-models on the families 201 and 207, lower on the families Nish, GdeJ, 205, 206, 208 and 212, and equal in the remaining families. As a further experiment the two provers have been run on two sets containing 40000 randomly generated formulas. The formulas in the first set were built on 23 connectives and 3 variables, the formulas in the second set consisted of 49 connectives and 5 variables[5]. To decide all the formulas in the first set EPDL took 18.75 seconds and LC-models took 18531. As regard the formulas of the second kind, EPDL took 66 whereas LC-models took 969079 seconds. These results show that EPDL is faster than LC-models and, more importantly, that EPDL scales better since its increasing factor between the two kinds of families formulas is lower than LC-models.

[2]LC-models is downloadable from `http://www.loria.fr/~larchey/LC`.

[3]Experiments performed on Intel(R) Xeon(TM) CPU 3.00GHz, RAM 2GB. Timings expressed in seconds. On 206.5 the computation was stopped after some hours because memory occupation was more than 90%. Names of unprovable formulas are in parenthesis.

[4]Other formulas characterizing intermediate logics have been considered in the experiments. Results are disregarded since both the provers decide them in few seconds.

[5]The formula SYJ201+1.001 is built on 23 connectives and 3 variables, whereas SYJ201+1.002 contains 49 connectives and 5 variables.

Formula	Basic	+Fact	+Side	EPDL
201.2	0.36	0.22	0.39	0.21
201.3	8.25	2.94	7.45	3.14
201.4	191	39.91	116	35.96
201.5	4830	502	1629	378
203.7	1.94	0.01	0.09	0.00
203.8	17.17	0.01	0.24	0.00
203.9	171.63	0.01	0.59	0.01
203.10	1885	0.00	1.42	0.01

Formula	Basic	+Fact	+Side	EPDL
205.3	0.03	0.03	0.03	0.03
205.4	0.12	0.13	0.08	0.08
205.5	0.76	0.80	0.23	0.24
205.6	5.80	5.92	0.68	0.68
(207.3)	0.54	0.27	0.56	0.31
(207.4)	11.51	3.63	9.70	3.71
(207.5)	266	46.63	139	40.23
(207.6)	6692	573	1827	406

Figure 7: Time comparison between different versions of EPDL.

The Figure 7 provides an account both of the formulas on which the optimizations work and a comparison between them. From left to right, "Basic" refers to EPDL lacking of the factorization rules and of the side information in the conclusion of $\mathbf{F} \rightarrow$; "+Fact" stands for Basic extended with the factorization rules; "+Side" denotes to Basic extended with the side information in the conclusion. The comparison clearly evidences that the optimizations are effective both individually and together. In particular, on SYJ201 and SYJ207 families of the ILTP library both the optimizations contribute to improve the performances. As regard SYJ203, the side information allows to reduce the timing. Finally, the factorization of $\mathbf{F} \rightarrow$-formulas improves the performances of EPDL on SYJ205.

Despite EPDL is a decision procedure whose time complexity is exponential, figures emphasize that by adding to the logical rules some optimizations, the result is a decision procedure which is effective on a wide range of formulas and outperforms LC-models which implements a polynomial time decision procedure.

References

[1] A. Avellone, M. Ferrari, and P. Miglioli. Duplication-free tableau calculi and related cut-free sequent calculi for the interpolable propositional intermediate logics. *Logic Journal of the IGPL*, 7(4):447–480, 1999.

[2] A. Avellone, G. Fiorino, and U. Moscato. Optimization techniques for propositional intuitionistic logic and their implementation. *Theoretical Computer Science*, 409(1):41–58, 2008.

[3] A. Avellone, P. Miglioli, U. Moscato, and M. Ornaghi. Generalized tableau systems for intermediate propositional logics. In D. Galmiche, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: Tableaux '97*, volume 1227 of *LNAI*, pages 43–61. Springer-Verlag, 1997.

[4] A. Avron. Hypersequents, logical consequence and intermediate logics for concurrency. *Annals for Mathematics and Artificial Intelligence*, 4:225–248, 1991.

[5] A. Avron. A tableau system for Gödel-Dummett logic based on a hypersequent calculus. In Roy Dyckhoff, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 98–111. Springer, 2000.

[6] A. Avron and B. Konikowska. Decomposition proof systems for Gödel-Dummett logics. *Studia Logica*, 69(2):197–219, 2001.

[7] M. Baaz. Infinite-valued Gödel logics with 0-1 projections and relativizations. In *Proceedings of Gödel '96 - Kurt Gödel's Legacy*, volume 6 of *Lecture Notes Logic*, pages 23–33, 1996.

[8] M. Baaz and C.G. Fermüller. Analytic calculi for projective logics. In Neil V. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX '99*, volume 1617 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 1999.

[9] M. Baaz, A. Ciabattoni, and C. G. Fermüller. Hypersequent calculi for Gödel logics – a survey. *J. of Logic and Computation*, 13(6):835–861, 2003.

[10] A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford University Press, 1997.

[11] R. Dyckhoff. A deterministic terminating sequent calculus for Gödel-Dummett logic. *Logic Journal of the IGPL*, 7(3):319–326, 1999.

[12] G. Fiorino. An $O(n \log n)$-SPACE decision procedure for the propositional Dummett Logic. *Journal of Automated Reasoning*, 27(3):297–311, 2001.

[13] G. Fiorino. Fast decision procedure for propositional dummett logic based on a multiple premise tableau calculus. Technical Report 164, Dipartimento di Metodi Quantitativi per le Scienze Economiche ed Aziendali, Università degli Studi di Milano-Bicocca, 2008.

[14] D.M. Gabbay. *Semantical Investigations in Heyting's Intuitionistic Logic*. Reidel, Dordrecht, 1981.

[15] K. Gödel. On the intuitionistic propositional calculus. In S. Feferman et al, editor, *Collected Works*, volume 1. Oxford University Press, 1986.

[16] R. Hähnle. Tableaux and related methods. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 100–178. Elsevier and MIT Press, 2001.

[17] P. Hajek. *Metamathematics of Fuzzy Logic*. Kluwer, 1998.

[18] J. Hudelmaier. An $O(n \log n)$-SPACE decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3(1):63–75, 1993.

[19] S.C. Kleene. *Introduction to Metamathematics*. Van Nostrand, New York, 1952.

[20] D. Larchey-Wendling. Combining proof-search and counter-model construction for deciding Gödel-Dummett logic. In Andrei Voronkov, editor, *CADE*, volume 2392 of *LNCS*, pages 94–110. Springer, 2002.

[21] D. Larchey-Wendling. Graph-based decision for Gödel-Dummett logics. *J. Autom. Reasoning*, 38(1-3):201–225, 2007.

[22] F. Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In Harrie de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, Oosterwijk, The Netherlands*, volume 1397 of *LNCS*, pages 217–232. Springer-Verlag, 1998.

[23] Thomas Raths, Jens Otten, and Christoph Kreitz. The iltp problem library for intuitionistic logic. *J. Autom. Reasoning*, 38(1-3):261–271, 2007.