Writing Proofs for Propositional Logic using rules of Natural Deduction in DC Proof

ICS 280: Introduction to Computational Logic

Project-Assignment

Sushil K Bajracharya

sbajrach@ics.uci.edu

Abstract

This document serves as my project assignment for *ICS 280: Introduction to Computational Logic* I took in Spring, 2004. In this brief report I discuss how DC Proof, a proof writing software can be used to write proofs about theorems in Propositional Logic using the Rules of Natural Deduction. Most of my discussions are based on several examples of Theorem Proofs I created in DC Proof itself. I follow the examples and procedures for Natural Deduction as outlined in [1].

Introduction

DC Proof [2] is a proof writing software developed by Dan Christensen. It is not a proof generator or an automated theorem prover, rather it can be taken as a neat and simple proof editor that eases the arduous task of writing long logical proofs. DC proof has support for both Propositional and Predicate Logic and its other features allow it to be used in creating proofs in other domains like Group and Number theory. In this document I will limit the discussion to using DC Proof for writing proofs for propositional logic using natural deduction.

The greatest benefits that come with DC proof is the set of Logic re-write rules and rules for deduction built inside the software itself. This allows a proof writer to simplify any propositional formula stated in the system. I will not list all the features built in the system, neither will I present each and every step required to use the software. Rather, I will present a brief discussion on rules of natural deduction and illustrate how each rules of deduction is implemented or can be applied using DC Proof with examples in DC Proof. Finally, I will give some examples of interesting proofs written in DC Proof.

Conventions used in DC Proof

DC proof is a simple text based proof writer, this means all the logical symbols and connectives are represented using normal ASCII characters. Table 1 shows how the standard logical symbols map to the symbols used in DC Proof (for working with Propositional Logic).

Operator	DC Proof Symbol	Logic Symbol
implies	=>	→
AND	&	Λ
OR	|	V
NOT	~	¬
brackets for precedence	[,]	(,)
IFF	<=>	↔

Table 1: DC Proof Symbols for Propositional Logic

For convenience I will use symbols used in DC proof to maintain consistency in examples and only use Logical Symbols when required.

Writing Proofs in DC Proof

Although it is far easier to proceed with logic rewrite rules in DC Proof, initially it takes a while to get used to it. The simplest explanation for writing proof in DC Proof goes as follows –

- Enter Premises or Assumptions as any valid propositional statement after clicking on the 'Premise' button in the tool bar
- Simplify the written formula using the options available from the 'Logic' menu. Some options simply take a statement and produce a new statement, others take more than one statements and apply deduction rules to derive new statements or propositional formulas.

For detailed explanation on each menu option I refer the reader to the online help in DC Proof. However, in following sections I present some of the options available in more details

Other features like putting comments in the proof before any active statement, automatic inclusion of text about rules applied to generate new statements below the statements and the facility to fold/unfold sections in proof (like folding/unfolding of source code in modern programming environments) are helpful.

Figure 1 shows a snapshot of a Proof written in DC Proof with appropriate Labels[1] depicting the elements of the proof, such as - comments, statements and information on rules applied. The first line is a comment. Lines labeled with numbers are propositional statements. Text immediately below the statements are auto-generated text about the rule applied to get the statement above. If it is an assumption or premise the text shows 'Premise' below the statement.

[1] Labels appear as text with black background and are not part of the proof itself

```
PROOF, Theorem: ~(~A|B)=>A <----COMMENT

        1       ~[~A | B] <----STATEMENT
                Premise
                    ↑------RULE APPLIED TO REACH STATEMENT 1

                Assumption: (Assuming the negation for proof by contradiction)

                2       ~A
                        Premise

                3       ~A | B
                        Arb Or, 2

                4       [~A | B] & ~[~A | B]
                        Join, 3, 1

        5       ~~A
                Conclusion, 2
                  ↑------RULE APPLIED TO REACH STATEMENT 5
        6       A
                Rem DNeg, 5

7       ~[~A | B] => A
        Conclusion, 1
```

Figure 1. An example proof in DC Proof

Rules for Natural Deduction

In this section I present the rules for Natural Deduction (introduced by Gerhard Gentzen) as outlined in [1]. I present a mapping of these rules to the facilities available in DC Proof in a tabulated form in Table 2. For alternative discussions of these rules I refer the reader to [2].

<table>
<tr><th>No.</th><th>Rule</th><th>Logical Equivalent</th><th>DC Proof Option[2]</th></tr>
<tr><td>1</td><td>$\wedge$ - introduction ($\wedge$ I)</td><td>A B

$A \wedge B$</td><td>Join</td></tr>
<tr><td>Ex[3]:</td><td colspan="3">AND – Introduction

1 A
Premise

2 B
Premise

3 A & B
Join, 1, 2</td></tr>
<tr><td>2, 3</td><td>$\wedge$ - elimination ($\wedge$ E)</td><td>$A \wedge B$, $A \wedge B$
-------- , -------
A , B</td><td>Split</td></tr>
<tr><td>Ex:</td><td colspan="3">AND - Elimination

1 A & B
Premise

2 A
Split, 1

3 B
Split, 1</td></tr>
<tr><td>4, 5</td><td>$\vee$ - introduction ($\vee$ I)</td><td>A , B
-------- , ---------
$A \vee B$, $A \vee B$</td><td>Arbitrary OR
(LHS/RHS)</td></tr>
<tr><td>Ex:
i, ii</td><td colspan="3">OR-Introduction (i)

1 A
Premise

2 A | B
Arb Or, 1

OR-Introduction (ii)

1 B
Premise

2 A | B
Arb Or, 1</td></tr>
</table>

[2] Available in the 'Logic' menu or the toolbar in DC Proof software
[3] Example

<table>
<tr><td>6</td><td>$\vee$ - elimination ($\vee$ E)</td><td>$$\dfrac{A \vee B \quad \begin{matrix} A \\ \vdots \\ C \end{matrix} \quad \begin{matrix} B \\ \vdots \\ C \end{matrix}}{C}$$</td><td>Cases</td></tr>
<tr><td>Ex:</td><td colspan="3"><code>Or - Elimination</code>

1 <code>A | B</code>
<code>Premise</code>

2 <code>A => C</code>
<code>Premise</code>

3 <code>B => C</code>
<code>Premise</code>

4 <code>[A => C] & [B => C]</code>
<code>Join, 2, 3</code>

5 <code>C</code>
<code>Cases, 1, 4</code></td></tr>
<tr><td>7</td><td>$\rightarrow$ - introduction
($\rightarrow$ I)</td><td>$$\dfrac{\begin{matrix} A \\ \vdots \\ C \end{matrix}}{A \rightarrow C}$$</td><td>Not a direct option available</td></tr>
<tr><td>Ex:</td><td colspan="3"><code>=> - Introduction</code>
<code>(Illustrating how => - introduction works)</code>

Assumption 1

<code>1 B => C</code>
<code>Premise</code>

Assumption 2

<code>2 A => B</code>
<code>Premise</code>

Assumption A is TRUE

<code>3 A</code>
<code>Premise</code>

<code>4 B</code>
<code>Detach, 2, 3</code>

...C can be infered TRUE</td></tr>
</table>

	5 C Detach, 1, 4 If 'C' can be derived from assumption 'A', then we can drop, or discharge, the assumption, and conclude that A=>C 6 A => C Conclusion, 3		
8	→ E (→ - elimination) Modus Ponens	$\dfrac{A \quad A \to C}{C}$	Detachment
Ex:	=> E (=> - elimination) Modus Ponens **1** A Premise **2** A => C Premise 3 C Detach, 2, 1		
9	⊥ From *falsum* any conclusion can be drawn	$\dfrac{\bot}{C}$	Not a direct option, 'Arbitrary Consequent' option can be used
Ex:	From 'falsum', any conclusion C can be drawn Assuming B is TRUE 1 B Premise ...then conclude ~B (FALSE) => anything 2 ~B => C Arb Cons, 1		
10	**Reductio Ad Absurdum** (RAA)	$\dfrac{\begin{matrix}\neg A\\ \vdots\\ \bot\end{matrix}}{A}\ ,\ \dfrac{\begin{matrix}A\\ \vdots\\ \bot\end{matrix}}{\neg A}$	Contrapositive

Ex (i)	`Reductio Ad Absurdum (RAA)`
	`.. the basis for Proof By Contradiction`
	`(First Version/Example)`
	
	Intial Argument.
	Starting with negation of the premise (~A) IMPLIES (B)
	
	`1 ~A => B`
	`  Premise`
	
	Asserting B as FALSE,
	that means the negation of the premise leads to FALSE
	
	`2 ~B`
	`  Premise`
	
	If 'B' is FALSE in the Initial Argument, this leads to conclusion '~~A' holds TRUE.
	
	`3 ~B => ~~A`
	`  Contra, 1`
	
	Applying Modus Ponens
	
	`4 ~~A`
	`  Detach, 3, 2`
	
	Removing double negation
	
	`5 A`
	`  Rem DNeg, 4`

Ex (ii)	`Reductio Ad Absurdum (RAA)`
	`- the basis for Proof By Contradiction`
	`(Second Version/Example)`
	
	Initial Argument
	
	`1 A => B`
	`  Premise`
	
	Asserting 'B' is false
	
	`2 ~B`
	`  Premise`
	
	If 'B' is FALSE in the Initial Argument, this leads to conclusion '~A' holds TRUE.
	
	`3 ~B => ~A`
	`  Contra, 1`
	
	Since 'A' leads to 'B' which is FALSE as we asserted,

	it leads to the conclusion that '~A' holds true.
	Again we are applying 'Modus Ponens' here.
	
	`4 ~A`
	`  Detach, 3, 2`		

11	**Identity** (Id)	A --- A	Conclusion (not a direct option)
Ex	A simple way to use this rule is to repeat any pre-stated assumption, statement or premise in DC Proof. Although seemingly simple the importance of this operation is not to be neglected as would be visible in longer proofs I will present in following sections.		

Table 2: Rules for natural deduction

Table 2 includes almost all the options available in DC Proof for Propositional Logic. One useful option is the 'Conclusion' menu item, that allows in deducing conclusion from the last statement and the 'Premise' immediately preceding that statement.

Understanding the information Table 2 conveys is the core of this document itself. Important thing to note is the logical equivalent of deduction rule given for each of the rules in DC Proof and observe how they are manifested in the examples that follow. All examples are well commented and labels and description for Figure 1 in an earlier section explains how to understand the examples given in the format of DC Proof. The best way to use this document is to understand the logic rules from Table 2 (as implemented in DC Proof) and work out the examples above and given below in DC Proof.

Proofs – More Examples

In this section I list further examples of proofs written in DC Proof. These proofs are well commented so that they could be understood easily.

PROOF: |- $A \wedge B \rightarrow B \wedge A$

```
PROOF, Theorem: A & B => B & A

  1       A & B
          Premise

  2       A
          Split, 1
```

```
    3       B
            Split, 1

    4       B & A
            Join, 3, 2

  5 A & B => B & A
    Conclusion, 1
```

PROOF: |- (┐B → ┐A) → (A → B)

```
PROOF, Theorem: |- (~B => ~A) => (A => B)
```

Assumption 1

```
1 ~B => ~A
  Premise
```

Assumption 2

```
2 A
  Premise
```

Assumption 3

```
  3       ~B
          Premise

  4       ~A
          Detach, 1, 3

  5       ~A & A
          Join, 4, 2

6 ~~B
  Conclusion, 3
```

This says Assumption 3 was wrong

```
7 B
  Rem DNeg, 6
```

Repeating Assumption 1

```
  8       ~B => ~A
          Premise

  9       A => B
          Arb Ante, 7

10        ~B => ~A => [A => B]
```

```
Conclusion, 8
```

PROOF: |- $(A \rightarrow B) \rightarrow (\neg A \vee B)$

```
PROOF, Theorem: (A=>B) => (~A|B)
```

Steps 1 - 7, constitute a mini-proof of the therorem ~(~A|B) => A

```
1       ~[~A | B]
        Premise
```

Assumption: Contradiction for the sub-proof

```
2       ~A
                Premise

        3       ~A | B
                Arb Or, 2

        4       [~A | B] & ~[~A | B]
                Join, 3, 1

5       ~~A
        Conclusion, 2

6       A
        Rem DNeg, 5

7 ~[~A | B] => A
  Conclusion, 1
```

Assumption: Contradiction for this theorem

```
8 ~[~A | B]
  Premise

9 A
  Detach, 7, 8
```

Assumption: Premise for this theorem

```
10      A => B
        Premise

11      B
        Detach, 10, 9

12      ~A | B
        Arb Or, 11
```

Assumption: Repeating the contradiction for this theorem

```
            13      ~[~A | B]
                    Premise

            14      [~A | B] & ~[~A | B]
                    Join, 12, 8

    15      ~~[~A | B]
            Conclusion, 13

  16       A => B => ~~[~A | B]
    Conclusion, 10

  17       A => B => ~A | B
          Rem DNeg, 16
```

Conclusion

In this short document I presented how DC Proof can be used to construct proofs for Propositional Logic. Explanation of each DC Proof option for logical operations and rewrite rules is available elsewhere [2]. Thus without being redundant I presented a mapping of rules of natural deduction to DC Proof features and presented several interesting examples of proof I took from [1] and constructed in DC Proof.

In conclusion, I would say using DC Proof make proof construction easier and less error prone. It definitely is not an automated theorem prover so one cannot expect features of such tools in DC proof. However, DC proof does not come with an exhaustive documentation that demonstrates all its capabilities. Neither it documents how fundamental logic theory is manifested with its use nor does it have a rich set of examples to help a novice get started. I hope this document makes an attempt to serve the latter two purposes.

References

[1] John Kelly. *The Essence Of Logic*. Prentice Hall Essence of Computing Series.
[2] www.dcproof.com, DC Proof Online Help, DC Proof Web Site.