CS270 - Week 7 Lecture Notes

Mark Boady

November 8, 2017

1 Derived Rules

The rules we looked at last week are the minimum set of axioms needed to do proofs by deduction. There are many other rules that are useful, but not required. Derived Rules are rules that can be created only using the basic set of rules.

1.1 Repetition

The repetition is just repeating that a statement is true. This is useful if we need to move a statement into a subproof.

1.	A	Premise
2.	A	R 1

The repetition rule is labeled with an R and has the line number that we are repeating.

Using only the basic rules we can also prove this.

1.	A	Premise
2.	$A \wedge A$	$\wedge I 1, 1$
3.	A	$\wedge E 2$

1.2 Disjunctive Syllogism

If we have an OR statement, but we know one side is false, then the other side **must** be true.

$$(A \vee B) \wedge \sim A \rightarrow B$$

This is written in a deduction proof as

1.	$A \vee B$	Premise
2.	$\sim A$	Premise
3.	B	DS 1,2

It can be used on either side of the OR statement.

1.	$A \vee B$	Premise
2.	$\sim B$	Premise
3.	A	DS 1,2

We can derive this from the basic rules.

1.	$A \vee B$	Premise
2.	$\sim A$	Premise
3.	A	Assume
4.	$\bot$	$\bot$ I 2,3
5.	B	X 4
6.	B	Assume
7.	B	$\vee$ 1, 3-5, 6-6

1.	$A \vee B$	Premise
2.	$\sim B$	Premise
3.	A	Assume
4.	B	Assume
5.	$\bot$	$\bot$ I 2,4
6.	A	X 5
7.	B	$\vee$ 1, 3-3, 4-6

1.3 Modus Ponens

If we know that $A \rightarrow B$ is true and we also know that B is false, then A must also be true because $T \rightarrow F = F$.

1.	$A \rightarrow B$	Premise
2.	$\sim B$	Premise
3.	$\sim A$	MT 1,2

The derivation for this rule is

1.	$A \rightarrow B$	Premise
2.	$\sim B$	Premise
3.	A	Assume
4.	B	$\rightarrow$ E 1,3
5.	$\bot$	$\sim$ I 2,4
6.	$\sim A$	$\sim$ I 3-5

1.4 Double Negative Elimination

A double negative is a positive.

1.	$\sim\sim A$	Premise
2.	A	DNE 1

This comes from a basic contradiction.

1.	$\sim\sim A$	Premise
2.	$\sim A$	Assume
3.	$\perp$	$\sim$ E 1, 2
4.	A	IP 2-3

1.5 Law of the Excluded Middle

If we know $A \to B$ and $\sim A \to B$ are both true, then the only option for B is for it to be true.

1.	$A \to B$	Premise
2.	$\sim A \to B$	Premise
3.	A	Assume
4.	B	$\to$ E 1,3
5.	$\sim A$	Assume
6.	B	$\to$ E 2,5
7.	B	LEM 3-4, 5-6

This used multiple contradictions. We assume that $\sim B$ is true and prove that is impossible.

1.	$A \to B$	Premise
2.	$\sim A \to B$	Premise
3.	$\sim B$	Assume
4.	A	Assume
5.	B	$\to$ E 1,4
6.	$\perp$	$\sim$ E 3,5
7.	$\sim A$	$\sim$ E 4-6
8.	B	$\to$ E 2,7
9.	$\perp$	$\sim$E 3,8
10.	B	IP 3-9

1.6 Hypothetical Syllogism

This one is not supported by our proof checker, but I like it.

1.	$A \to B$	Premise
2.	$B \to C$	Premise
3.	$A \to C$	HS 1,2

We can do it the long way using derived rules.

1.	$A \to B$	Premise
2.	$B \to C$	Premise
3.	A	Assume
4.	B	$\to$ E 1,3
5.	C	$\to$ E 2,4
6.	$A \to C$	$\to$ I 3-5

1.7 DeMorgan's Laws

You will prove $\sim A \vee \sim B = \sim (A \wedge B)$ in lab.

1.	$\sim A \vee \sim B$	Premise
2.	$\sim (A \wedge B)$	DeM 1

1.	$\sim (A \wedge B)$	Premise
2.	$\sim A \vee \sim B$	DeM 1

1.	$\sim A \wedge \sim B$	Premise
2.	$\sim (A \vee B)$	DeM 1

1.	$\sim (A \vee B)$	Premise
2.	$\sim A \wedge \sim B$	DeM 1

Here will will only justify $\sim A \wedge \sim B = \sim (A \vee B)$

1.	$\sim A \wedge \sim B$	Premise
2.	$A \vee B$	Assume
3.	A	Assume
4.	$\sim A$	$\wedge$ E 1
5.	$\perp$	$\sim$E 3,4
6.	$\sim (A \vee B)$	X 5
7.	B	Assume
8.	$\sim B$	$\wedge$ 1
9.	$\perp$	$\sim$E 3,4
10.	$\sim (A \vee B)$	X 9
11.	$(A \vee B)$	$\vee$ 2, 3-6, 7-10
12.	$\perp$	$\sim$ 2,11
13.	$\sim (A \vee B)$	$\sim$I 2-12

1.	$\sim (A \vee B)$	Premise
2.	A	Assume
3.	$A \vee B$	$\vee$ I 2
4.	$\perp$	$\sim$ E 1,3
5.	$\sim A$	$\sim$I 2-4
6.	B	Assume
7.	$A \vee B$	$\vee$ I 6
8.	$\perp$	$\sim$ E 7,1
9.	$\sim B$	$\sim$ I 6-8
10.	$\sim A \wedge \sim B$	$\wedge$ I 5,9

2 Normal Forms

When working with Boolean expressions, it is useful to have them in specific formats. Every well-formed boolean expression can be written in every one of the following normal forms.

2.1 Negative Normal Form

Negative Normal Form (NNF) makes it clear which variables need to be true and which need to be false.

NNF follows certain rules

- Not Symbols may only be applied to variables.
- At most 1 Not symbol may be on a variable.

We have already see that $\sim\sim A = A$. If we have a variable with multiple negative signs, we can remove them.

To make an expression using NOT, OR, and AND into NNF we need the following rules.

$$\sim\sim A = A \tag{1}$$
$$\sim (A \vee B) \sim A \wedge \sim B \tag{2}$$
$$\sim (A \wedge B) \sim A \vee \sim B \tag{3}$$

We can make any boolean expression into NNF using just these three rules.

$$\begin{aligned}\sim\sim\sim (A \wedge (B \vee \sim C)) &= \sim (A \wedge (B \vee \sim C)) && (4)\\ &= \sim A \vee \sim (B \vee \sim C)) && (5)\\ &= \sim A \vee (\sim B \wedge \sim\sim C) && (6)\\ &= \sim A \vee (\sim B \wedge C) && (7)\end{aligned}$$

2.2 Disjunctive Normal Form

Disjunctive Normal Form is a Disjunction of Conjunctions.

It meets the following rules:

- The expression is in NNF
- All Disjunctions are above Conjunctions

The following are in DNF.

$$\begin{aligned}&A && (8)\\ &A \vee B && (9)\\ &A \wedge B && (10)\\ &A \vee (B \wedge \sim C) \vee (\sim C \wedge B \vee X) && (11)\end{aligned}$$

In general, DNF expressions look like

$$C_1 \vee C_2 \vee C_3 \vee C_4 \vee \cdots \tag{12}$$

Where C_i is called the i-th clause.

Each Clause looks like

$$C_i = A \wedge B \wedge C \wedge \cdots \tag{13}$$

Once an expression is in NNF, it can be converted to a DNF using only one additional rule.

$$A \wedge (X \vee Y) = (A \wedge X) \vee (A \wedge Y) \tag{14}$$

By applying distribution repeatedly, we organize all the operators the way we want.

To make a DNF true, successfully set one clause to be true. We can look at the clauses independently. To make a DNF false, we need to set every clause false.

2.3 Conjunctive Normal Form

Conjunctive Normal Form is a Conjunction of Disjunctions.

It meets the following rules:

- The expression is in NNF
- All Conjunctions are above disjunctions

The following are in CNF.

$$A \tag{15}$$
$$A \vee B \tag{16}$$
$$A \wedge B \tag{17}$$
$$A \wedge (B \vee \sim C) \wedge (\sim C \vee B \vee X) \tag{18}$$

In general, CNF expressions look like

$$C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge \cdots \tag{19}$$

Where C_i is called the i-th clause.

Each Clause looks like

$$C_i = A \vee B \vee C \vee \cdots \tag{20}$$

Once an expression is in NNF, it can be converted to a CNF using only one additional rule.

$$A \vee (X \wedge Y) = (A \vee X) \wedge (A \vee Y) \tag{21}$$

By applying distribution repeatedly, we organize all the operators the way we want.

To make a CNF true, successfully set every clause to be true. To make a CNF false, we need to set one clause to false.

Trying to Satisfy a CNF is nice because looking at a clause tells us exactly what to do.

3 3-CNF

Any expression can be converted to a 3-CNF formula. These formula are not equivalent, but

$$\text{SAT(3-CNF)} \rightarrow \text{SAT(CNF)} \tag{22}$$

If we can make the 3-CNF formula true, then we can also make the CNF formula true.

If a clause has 1 variable, then we can write is as

$$A \text{ is SAT iff } (A \vee B \vee \sim B) \tag{23}$$

If a clause has 2 variables, then we can write is as

$$(A \vee B) \text{ is SAT iff } (A \vee B \vee X) \wedge (A \vee B \vee \sim X) \tag{24}$$

If a clause has 3 variables, no changes are needed.

If a clause has 4 variables, we can write it as

$$(A \vee B \vee C \vee D) \text{ is SAT iff } (A \vee B \vee X) \wedge (\sim X \vee C \vee D) \tag{25}$$

If a clause has more then 4 variables, we can repeatedly apply the rule for 4 variables.

$$\begin{aligned}(A \vee B \vee C \vee D \vee E) &\text{ is SAT iff } (A \vee B \vee X) \wedge (\sim X \vee C \vee D \vee E) && (26)\\ &\text{ is SAT iff } (A \vee B \vee X) \wedge (\sim X \vee C \vee Y) \wedge (\sim Y \vee D \vee E) && (27)\end{aligned}$$

3-CNF is useful from an automation standpoint. If we know for a fact there will be exactly 3 variables in each clause, we can optimize the SAT algorithm.

4 Terminology

Satisfiable: Can be Set to True

Falsifiable: Can be Set to False