CISC 204 Class 12

Auxiliary Variables and the Tseitin Encoding

Text Correspondence: $\sim$

Main Concepts:

- *Auxiliary variable: rewriting a complex formula as a single "fresh" variable*
- *Tseitin encoding: rewriting a complex formula with an auxiliary variable for each sub-formula*

Often, we want to represent the concept of an entire formula being true or false *as a single variable*. This can help with the clarity of a logical encoding, but can also help with the efficiency of translating the encoding itself to CNF (as will be seen in the next section).

12.1 Auxiliary Variables

For an arbitrary propositional formula Φ, we can define an auxiliary variable x representing the formula by using the constraint $x \leftrightarrow \Phi$. As a concrete example, let's consider introducing an auxiliary variable x for the formula $p \vee q$:

$$x \leftrightarrow (p \vee q)$$

This can be split into two formulae corresponding to each direction:

$$x \rightarrow (p \vee q)$$
$$\neg x \vee p \vee q$$

$$(p \vee q) \rightarrow x$$
$$\neg(p \vee q) \vee x$$
$$(\neg p \wedge \neg q) \vee x$$
$$(\neg p \vee x) \wedge (\neg q \vee x)$$

We have converted each direction to CNF, and the final complete formula is:

$$(\neg x \vee p \vee q) \wedge (\neg p \vee x) \wedge (\neg q \vee x)$$

Finally, we can inspect the truth table for the formula:

p	q	x	$p \vee q$	$x \leftrightarrow (p \vee q)$
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Notice that the $x \leftrightarrow (p \vee q)$ formula holds only when the columns x and $p \vee q$ are equal. This is precisely what we want from an auxiliary variable that represents the true/false nature of another formula.

12.2 Tseitin Encoding

Building on the idea of auxiliary variables, we can repeatedly apply the technique to transform any arbitrary formula to CNF. The general idea of the Tseitin encoding is to introduce an auxiliary variable for every sub-formula. This allows us to convert a formula to CNF without suffering the combinatorial explosion that repeated application of de Morgan's Laws might cause.

To demonstrate the technique, consider the following formula:

$$\neg((p \vee q) \wedge (r \vee s)) \wedge t \tag{12.1}$$

The parse tree of formulae (12.1) is shown in Figure 12.1.

Recall that every inner node of the parse tree corresponds to one of the sub-formula of the original formula. Starting with those nodes further down, we wind up with the following auxiliary variable definitions:

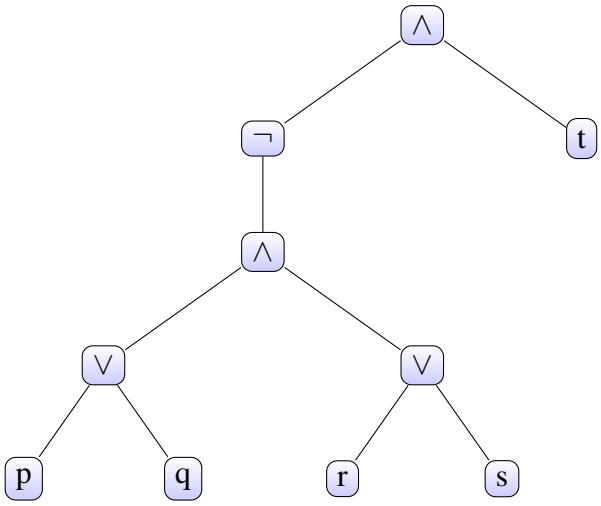

Figure 12.1: Parse tree for $\neg((p \vee q) \wedge (r \vee s)) \wedge t$.

$$x_1 \leftrightarrow (p \vee q) \tag{12.2}$$

$$x_2 \leftrightarrow (r \vee s) \tag{12.3}$$

$$x_3 \leftrightarrow (x_1 \wedge x_2) \tag{12.4}$$

$$x_4 \leftrightarrow \neg x_3 \tag{12.5}$$

$$x_5 \leftrightarrow (x_4 \wedge t) \tag{12.6}$$

Each of these auxiliary variable definitions will correspond to a CNF, as demonstrated in Section (12.2). For example, Formula 12.4 would be converted as:

$$x_3 \leftrightarrow x_1 \wedge x_2$$

$$(x_3 \rightarrow (x_1 \wedge x_2)) \wedge ((x_1 \wedge x_2) \rightarrow x_3)$$

$$(\neg x_3 \vee (x_1 \wedge x_2)) \wedge (\neg(x_1 \wedge x_2) \vee x_3)$$

$$(\neg x_3 \vee x_1) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

The final Tseitin encoding of Formula (12.1) will include the CNF conversion of each of Formulae (12.2)-(12.6), plus the single unit clause (x_5). This final clause essentially states that we want the entire Formula (12.1) to hold, which is what x_5 represents (i.e., the root of the parse tree).

12.3 Computational Efficiency of Tseitin Encoding

To understand why we would want to use the Tseitin encoding for converting a formula to CNF, consider the following formula:

$$(p_{1,1} \wedge p_{1,2}) \vee (p_{2,1} \wedge p_{2,2})) \vee \cdots \vee (p_{k,1} \wedge p_{k,2})$$

Converting to CNF naïvely would require us to consider every combination from each of the k terms. This would result in 2^k clauses in the CNF through the use of de Morgan's Laws.

On the other hand, when using the Tseitin encoding, each of the k terms would have a unique variable added and correspond to 3 clauses corresponding to:

$$x_i \leftrightarrow (p_{i,1} \wedge p_{i,2})$$

The final auxiliary variable for the entire formula would require an additional $k + 1$ clauses corresponding to:

$$x_{k+1} \leftrightarrow (x_1 \vee x_2 \vee \cdots \vee x_k)$$

This leads us to a total of $3 \cdot k + k + 1 = 4 \cdot k + 1$ clauses in the final CNF using the Tseitin encoding. Compared to the naïve one with 2^k clauses, this represents *substantial* savings.

The trade-off for using the Tseitin encoding is that we have many more variables introduced in the theory. Most reasoning approaches will systematically explore the assignment to variables, and every new variable effectively doubles the number of possible solutions. Nonetheless, the Tseitin encoding (or similar variations) are the state of the art for encoding large and complex formulae so that solvers that work only with CNF can process them.