Logic for Computer Science - Week 7
Tseitin's Algorithm

1 Introduction

Example 1.1. *Consider the class of formulae*

$$\varphi_n = (\mathtt{p}_1 \wedge \mathtt{q}_1) \vee (\mathtt{p}_2 \wedge \mathtt{q}_2) \vee \ldots \vee (\mathtt{p}_n \wedge \mathtt{q}_n),$$

where n ranges over the positive naturals.

Following the algorithm in the previous lecture, we obtain the following conjunctive φ'_n normal form of φ:

$$\varphi'_n = \bigwedge_{a_i \in \{p,q\}} (a_1 \vee a_2 \vee \ldots \vee a_n).$$

In other words, φ'_n is a conjunction of 2^n clauses. This constitutes what is called an exponential growth, and exponential growths are generally bad when they can be avoided. For example, when $n = 10$, we have that φ'_{10} has 1024 clauses, when $n = 20$ we have that φ'_{20} has 1048576 clauses, etc. In other words, when n increases by 1 unit, the resulting formula doubles its size.

Unfortunately, due to the exponential blowup explained in the previous example, it is not possible to expect the algorithm in the previous lecture to be fast.

2 Tseitin's Transformation

In order to avoid the exponential blowup, Tseitin (sometimes spelled Tseytin) proposed an algorithm that avoid the exponential blowup. Tseitin's transformation (or Tseitin's algorithm) takes as input a formula φ and produces a formula φ' that is in CNF. However, unlike the previous transformation, there is a weaker connection between φ and φ' in general: φ is only guaranteed to be equisatisfialble to φ', not equivalent. Two formulae are equisatisfiable if both are satisfiable or if both are not satisfiable.

Example 2.1. *The formulae $\mathtt{p}$ and $\neg\mathtt{q}$ are equisatisfiable (both are satisfiable).*

The formulae $\mathtt{p} \wedge \neg\mathtt{p}$ and $\neg(\mathtt{q} \vee \neg\mathtt{q})$ are equisatisfiable (both are unsatisfiable).

The formulae $\mathtt{p}$ *and* $\mathtt{p} \wedge \neg \mathtt{p}$ *are not equisatisfiable (the first is satisfiable, but the second is not satisfiable).*

Remark 2.1. *Any two formulae that are equivalent are also equisatisfiable (justify this claim).*

The exact guarantees are described in the following theorem:

Theorem 2.1. *For any formula* $\varphi \in PL$*, there exists a formula* $\varphi' \in PL$ *such that:*

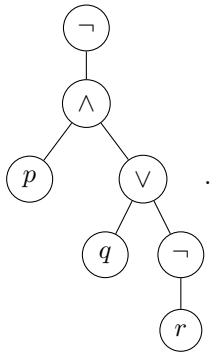
- φ *and* φ' *are equisatisfiable;*
- φ' *is in CNF;*
- $size\Big(\varphi'\Big) \leq 16 \times size\Big(\varphi\Big)$ *(the resulting formula is not much bigger);*
- *any model of* φ' *is also a model of* φ*.*

Recall that $size\Big(\varphi\Big)$ is the number of nodes of the abstract syntax tree of φ and that the function *size* was defined in the second lecture.

As with the previous theorem, we will explain the algorithm for finding φ' from φ on an example.

Step 1 The first step of Tseitin's algorithm is to start with the abstract syntax tree of the formula φ and associate to every node a propositional variable.

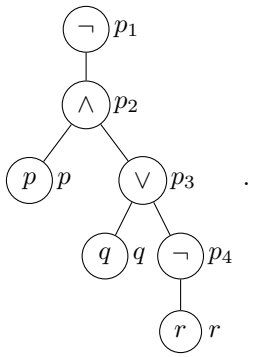
Let $\varphi = \neg(\mathtt{p} \wedge (\mathtt{q} \vee \neg \mathtt{r}))$. We have that

$ast\Big(\varphi\Big) =$ 

.

The first step of the algorithm it to associate to every internal node (node that is not a leaf) a fresh propositional variable (i.e. a propositional variable not seen before). To the leaves, we associate the variables that are already in the leaves. Here is the resulting tree, where next to each node we write the

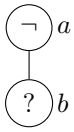
propositional variable associated with the node:



The idea of the fresh propositional variables $\mathtt{p}_1, \mathtt{p}_2, \mathtt{p}_3, \mathtt{p}_4, \mathtt{p}_5$ is to construct the formula φ' in such a way so that, in any truth assignment τ satisfying φ', the value of the new propositional variables will be the same as the value of the subformula rooted in the node to which the variable is associated. For example, $\hat{\tau}\Big(p_3\Big)$ should be equal to $\hat{\tau}\Big(q \vee \neg r\Big)$.

Step 2 The formula φ' will be a conjunction of clauses and for each internal node we will have two or three clauses, as follows:

- if the node is a negation of the following form:

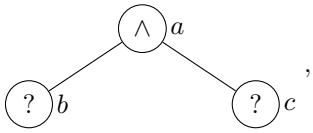


  where $a, b \in A$ are the propositional variables associated to the nodes, then we add to φ' clauses equivalent to the formula $a \leftrightarrow \neg b$. In this way, in any truth assignment satisfying φ', we have that a and $\neg b$ have the same truth value. How do we find the clauses? It is sufficient to rearrange the formula $a \leftrightarrow \neg b$ as follows:

$$\begin{aligned} \mathtt{a} \leftrightarrow \neg \mathtt{b} &\equiv (a \rightarrow \neg b) \wedge (\neg b \rightarrow a) \\ &\equiv (\neg a \vee \neg b) \wedge (\neg \neg b \vee a) \\ &\equiv (\neg a \vee \neg b) \wedge (b \vee a). \end{aligned}$$

  Therefore, we will add the clauses $\neg a \vee \neg b$ and $b \vee a$ to φ'.

- if the node is a conjunction of the following form:

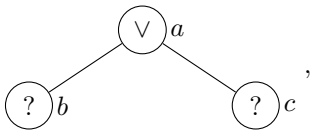


where $a, b, c \in A$ are the propositional variables associated to the nodes, then we add to φ' clauses equivalent to the formula $a \leftrightarrow b \wedge c$. In this way, in any truth assignment satisfying φ', we have that a and $b \wedge c$ have the same truth value. How do we find the clauses? It is sufficient to rearrange the formula $a \leftrightarrow b \wedge c$ as follows:

$$\begin{aligned} a \leftrightarrow b \wedge c &\equiv (a \rightarrow b \wedge c) \wedge (b \wedge c \rightarrow a) \\ &\equiv (\neg a \vee b \wedge c) \wedge (\neg(b \wedge c) \vee a) \\ &\equiv (\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c \vee a). \end{aligned}$$

Therefore, we will add the clauses $(\neg a \vee b)$, $(\neg a \vee c)$ and $(\neg b \vee \neg c \vee a)$ to φ'.

- if the node is a disjunction of the following form:



where $a, b, c \in A$ are the propositional variables associated to the nodes, then we add to φ' clauses equivalent to the formula $a \leftrightarrow b \vee c$. In this way, in any truth assignment satisfying φ', we have that a and $b \vee c$ have the same truth value. How do we find the clauses? It is sufficient to rearrange the formula $a \leftrightarrow b \vee c$ as follows:

$$\begin{aligned} a \leftrightarrow b \vee c &\equiv (a \rightarrow b \vee c) \wedge (b \vee c \rightarrow a) \\ &\equiv (\neg a \vee b \vee c) \wedge (\neg(b \vee c) \vee a) \\ &\equiv (\neg a \vee b \vee c) \wedge ((\neg b \wedge \neg c) \vee a) \\ &\equiv (\neg a \vee b \vee c) \wedge (\neg b \vee a) \wedge (\neg c \vee a). \end{aligned}$$

Therefore, we will add the clauses $(\neg a \vee b \vee c)$, $(\neg b \vee a)$ and $(\neg c \vee a)$ to φ'.

- if the node is an implication (exercise);
- if the node is a double implication (exercise).

In our example, we would add the following clauses to φ':

1. $\mathtt{p}_1 \vee \mathtt{p}_2$, $\neg\mathtt{p}_1 \vee \neg\mathtt{p}_2$;
2. $\neg\mathtt{p}_2 \vee \mathtt{p}_1$, $\neg\mathtt{p}_2 \vee \mathtt{p}_3$, $\neg\mathtt{p} \vee \neg\mathtt{p}_3 \vee \mathtt{p}_2$;
3. $\neg\mathtt{q} \vee \mathtt{p}_3$, $\neg\mathtt{p}_4 \vee \mathtt{p}_3$, $\neg\mathtt{p}_3 \vee \mathtt{q} \vee \mathtt{p}_4$;
4. $\mathtt{p}_4 \vee \mathtt{r}$, $\neg\mathtt{p}_4 \vee \neg\mathtt{r}$.

Step 3 Finally, the last step consists of adding a clause to φ' consisting of just the variable associated to the root node.

In our example, we obtain

$$\begin{aligned}\varphi' = \quad & (\mathtt{p}_1 \vee \mathtt{p}_2) \wedge (\neg\mathtt{p}_1 \vee \neg\mathtt{p}_2) \wedge \\ & (\neg\mathtt{p}_2 \vee \mathtt{p}_1) \wedge (\neg\mathtt{p}_2 \vee \mathtt{p}_3) \wedge (\neg\mathtt{p} \vee \neg\mathtt{p}_3 \vee \mathtt{p}_2) \wedge \\ & (\neg\mathtt{q} \vee \mathtt{p}_3) \wedge (\neg\mathtt{p}_4 \vee \mathtt{p}_3) \wedge (\neg\mathtt{p}_3 \vee \mathtt{q} \vee \mathtt{p}_4) \wedge \\ & (\mathtt{p}_4 \vee \mathtt{r}) \wedge (\neg\mathtt{p}_4 \vee \neg\mathtt{r}) \wedge \\ & (p_1)\end{aligned}$$

By construction, we have that:

1. any model of the formula φ' is also a model of φ – meaning that if φ' is satisfiable then φ must also be satisfiable;
2. any model τ of the formula φ can be changed into a model of φ by setting $\tau[\mathtt{p}_i]$ (assuming $\mathtt{p}_i$ are the fresh variables) to the truth value of the corresponding subformula of φ in τ – therefore if φ is satisfiable then φ' must also be satisfiable;
3. for every internal node of φ, φ' contains 2 or 3 clauses, each clause having an ast of at most 16 nodes – this means that the size of the resulting formula φ' is at most 16 times larger than the size of the initial formula;
4. as φ' is a conjunction of clauses, it is obviously in CNF.

Therefore φ' fulfils all requirements stated in the theorem above.