Proof systems for #SAT based on restricted circuits

Florent Capelli[1]

Université de Lille, CRIStAL, INRIA/CNRS

Abstract. In this work, we extend a well-known connection between linear resolution refutation and read-once branching program by constructing proof systems based on syntactically restricted circuits studied in the field of Knowledge Compilation. While our approach only yield proof systems that are weaker than resolution, they may be used to define proof systems for problems such as #SAT or max SAT. This is a work in progress.

Extended abstract. It is well-known that a linear resolution refutation of a CNF formula F can be seen as a read-once branching program B whose sinks are labelled by clauses of F and such that every path from the source of B to a sink labelled with $c \in F$ corresponds to an assignment of the variables of F that does not satisfy c, as depicted on Figure 1.

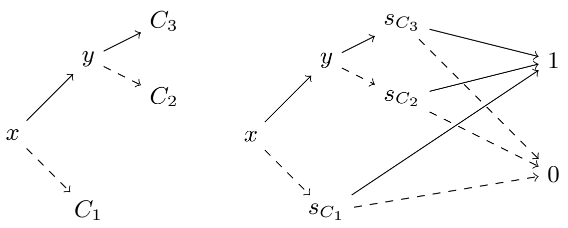

Fig. 1. Refutation of $F = C_1 \wedge C_2 \wedge C_3 = x \wedge y \wedge (\neg x \vee \neg y)$ and its transformation. Solid arrows correspond to fixing the value of the variable to 1, dashed arrows to 0.

By slightly changing the point of view, we can see this type of branching programs as a propositional proof system in the Cook–Reckhow sense as one can check in polynomial time if a given branching program B is indeed a refutation of a CNF formula. It is enough to check that the branching program is read-once, that is, there is no source-sink path containing twice the same variable and that for every clause $c \in F$ and every literal $\ell \in c$, every path from the source of B to a c labelled clause tests $\neg\ell$.

Another way of interpreting this result is to consider the following transformation of a CNF-formula F:

$$\tilde{F} = \bigwedge_{c \in F} c \vee s_c$$

where s_c is a fresh *selector variable* for every clause $c \in F$. Now, we can see B as a circuit computing some Boolean function b on variables $\{s_c \mid c \in F\} \cup \mathsf{var}(F)$ by turning every sink of B labelled with c into a node testing s_c and returning 1 if $s_c = 1$ and 0 otherwise, as depicted on Figure 1. It turns out that B is a refutation of F if and only if we have $\tilde{F} \Rightarrow b$ and that if we plug all s_c to 0 in b, then we have a Boolean function identically equal to 0. Both properties can easily be checked in polynomial time on branching programs.

All this reasoning can be abstracted independently from the representation of b. Indeed, assume we are given a Boolean function b on variables $\mathsf{var}(F) \cup \{s_c \mid c \in F\}$. If the representation of b is good enough, we may be able to check in polynomial time whether $\tilde{F} \Rightarrow b$. Now, if we are also able to check in polynomial time where b evaluates to 0 when we plug all s_c to 0, then we can consider b as a proof that F is not satisfiable. We can actually do much more. Assume we can count the number M of models of b when we plug all s_c to 0, then F has at most M models. In this case, b can be seen as a *proof* that F has at least M models in the sense that it can be checked in polynomial time by a third party.

Such good representations actually already exist in the literature in the field of Knowledge Compilation. They naturally induce proof systems for SAT but it seems that they are not much stronger than linear resolution. However, they also naturally induce proof systems for $\#\mathsf{SAT}$ or $\max\mathsf{SAT}$ and other aggregation problems for which there do not seem to exist much alternative. It turns out that many existing tools for solving $\#\mathsf{SAT}$ actually already implicitly construct a circuit that can be seen as a proof on the number of models of the input CNF.

In this talk, we propose to present this generic construction of proof systems for aggregation problems using circuits and develop the example on a particular family of circuits known as decision DNNF that strictly generalises branching programs and explain how such proofs could be easily extracted from the existing tools for $\#\mathsf{SAT}$.

State of the work. This is an ongoing unpublished work. I have trouble judging the relevance of these observations for the proof complexity community and I would enjoy the opportunity of this workshop to discuss it.