Formal Proofs and their Lengths V

Resolution

We identify CNF formulas with set of *clauses* C_i, where each C_i is a set of *literals* l_j and a literal is either a variable or its negation.

Definition 1. Given a CNF $\mathcal{C} = \{C_1^{init}, ..., C_m^{init}\}$ a **resolution refutation** of C is a sequence of clauses $C_1, ..., C_k$ so that

- C_k is an empty set,
- each C_j is either an initial clause C_i^{init} or is derived from the previous clauses via the **resolution rule**

$$\frac{C \cup \{p\} \quad C' \cup \{\neg p\}}{C \cup C'}$$

For a general formula ϕ its resolution refutation is defined as a resolution refutation of $L(\phi)$.

We have already seen that resolution is sound, i.e. given a CNF $\mathcal{C}$ with a valid resolution refutation it follows that $\mathcal{C}$ is unsatisfiable. We have also established that resolution can be viewed as a proof system, where in order to prove ϕ one considers a refutation of $\neg\phi$ and in order to transform $\neg\phi$ into an equisatisfiable CNF one applies Tseitin's limited extension procedure. It remains to show that resolution is complete.

Theorem 2. Resolution is a complete refutation system, i.e. if ϕ is unsatisfiable, then there is a resolution refutation of ϕ.

Exercise 3 (*towards 2*)**.** Let us fix a variable p and divide a CNF $\mathcal{C}$ into four parts:

- $\mathcal{C}_{00} = \{C \in \mathcal{C} \mid p, \neg p \notin C\}$;
- $\mathcal{C}_{01} = \{C \in \mathcal{C} \mid p \notin C, \neg p \in C\}$;
- $\mathcal{C}_{10} = \{C \in \mathcal{C} \mid p \in C, \neg p \notin C\}$;
- $\mathcal{C}_{11} = \{C \in \mathcal{C} \mid p, \neg p \in C\}$.

Let $\tilde{\mathcal{C}}$ contain $\mathcal{C}_{00}$ and all clauses acquired by application of the resolution rule on all pairs of clauses from $\mathcal{C}_{01}$ and $\mathcal{C}_{10}$, where p is the resolved variable. Show that if $\mathcal{C}$ is unsatisfiable, then $\tilde{\mathcal{C}}$ is unsatisfiable, as well.

Exercise 4 (*towards 2*)**.** Apply the previous exercise inductively to finish the proof of 2. What is the upper bound on the size of the refutation?

Exercise 5. Conclude that resolution is a Cook-Reckhow proof system.

Exercise 6. Using resolution show that 2-CNF-SAT is in **P**.

Proofs as games

Current exercise session is based upon the paper of P. Pudlák "Proofs as games". It can be found here: https://www.jstor.org/stable/2589349.

Exercise 7. Recall the *pigeonhole principle* which states that there is no injective mapping between $n+1$ different pigeons and n different holes. We denote this statement is PHP_n^{n+1}. Express $\neg PHP_n^{n+1}$ as a CNF of size polynomial in n.

The above CNF (or more precisely a family of CNFs) is the one for which we will derive exponential lower bounds for resolution refutations. The major part of today's session is devoted to the analysis of a certain combinatorial game. Near the end we will show the correspondence between *space complexity* of the mentioned game and size of resolution refutations of $\neg PHP_n^{n+1}$.

A game

Definition 8. We consider the following **game** played between two players Alice (*falsifier* or *spoiler* denoted simply as A) and Bob (*prover* or *delayer* denoted simply as B). A pretends there is some mapping f contradicting PHP_n^{n+1} and B tries to convict her of lying. B can query A questions of the following form "does pigeon p goes to hole h" and A must answer positively or negatively. B stores A's answers in the form $(p, h, \text{yes / no})$, but through the course of the game he might erase particular records from his list. A sees the actual list stored by B.

B **wins** iff there is a *direct contradiction* among the records stored in his list. This means that either for all holes h B's list contains records (p, h, no) for a particular pigeon, or B's list contains records (p', h, yes) and (p'', h, yes) for $p' \neq p''$, or B's list contains records (p, h', yes) and (p, h'', yes) for $h' \neq h''$, or B's list contains records (p, h, yes) and (p, h, no).

Strategy (for B) is a function from the set of all the lists of records into possible next moves which specify what to erase from the current list and what to query next.

We will consider only the *winning strategies* which we will call just strategies. We can also express any strategy as a function from the set of only the *admissible* lists, i.e. lists which can actually appear through the course of the game.

Exercise 9. Try to come up with some strategy for which the set of all lists and the set of all admissible lists do not equal.

Definition 10. Suppose a strategy is fixed. We say the the **complexity of the strategy** is the number of different records that can appear in all possible games.

Exercise 11. Recall that we identify a strategy with the function describing the next move based on the current list, and we care only about the admissible lists. Show that the strategy's complexity equals the number of *strategy's rules*, i.e. it the size of the domain of the function as above.

We emphasize that complexity manifests itself only through the course of many different games played against the strategy.

Exercise 12. Try to come up with a strategy whose complexity is exponential in n, although in any single game only poly of n different lists may appear.

Our goal is to prove that any strategy is necessarily of exponential complexity. For that we will define a *superstrategy* (for A) which would force B into using a lot of different lists. Here, superstrategy means a parameterized family of strategies of A to play against B. We will also see that this superstrategy doesn't actually use the a priori knowledge of B's strategy.