CS 2429 - Propositional Proof Complexity

Lecture #7: 24 October 2002

Lecturer: Toniann Pitassi

Scribe Notes by: Philipp Hertel

1 Today's Topics:

- Feasible Interpolation for Resolution
- Monotone Feasible Interpolation for Resolution
- Feasible Interpolation for Cutting Planes
- Monotone Feasible Interpolation for Cutting Planes

2 Definitions: Feasible, Interpolation, Monotone, Uniform

Recall the definitions of feasible interpolatin and monotone feasible interpolation from last class.

Definition A proof System V has *feasible interpolation* iff for all unsatisfiable formulas of the form $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$ there is a boolean circuit of size at most polynomial in s computing an interpolant C for $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$, where s is the size of shortest V-proof of $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$.

Definition A proof system V has *monotone feasible interpolation* iff $\overrightarrow{p}$ occuring only positively in A implies that C as described above is monotone.

We now give a new, uniform version of feasible interpolation suggested by Steve Cook. It turns out that all propositional proof systems that we currently know admitting feasible interpolation also admit uniform feasible interpolation.

Definition A proof system V has *uniform feasible interpolation* iff there exists a uniform polynomial time algorithm $M(\alpha, P)$ where P is a V-proof of $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$ and α is an assignment to $\overrightarrow{p}$, and M outputs:

0 only if $A(\alpha, \overrightarrow{q})$ is unsatisifiable
1 only if $B(\alpha, \overrightarrow{r})$ is unsatisfiable

A claim that a proof system has uniform feasible interpolation is a stronger claim than it having feasible interpolation. The condition that it has a uniform polynomial time algorithm requires that every instance, regardless of its size, must be computed using the same algorithm. Feasible interpolation, if it exists for a propositional proof system, is usually uniform. The two proof systems discussed in this lecture, Resolution and Cutting Planes both have uniform feasible interpolation.

Recall from the last lecture that if a proof-system V has monotone feasible interpolation, then unconditional super-polynomial lower bounds follow for V. If V has feasible interpolation, then superpolynomial lower bounds for V hold, assuming that NP is not contained in $P/poly$.

In the next section, we will show that Resolution has feasible interpolation and monotone feasible interpolation, and thus we obtain unconditional superpolynomial lower bounds for Resolution. Next we will show that Cutting Planes has (uniform) feasible interpolation. However, we will not show that Cutting Planes has monotone feasible interpolation, but we will nonetheless be able to prove unconditional lower bounds for Cutting Planes as well by showing that there is a feasible monotone real circuit for any $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$ with an efficient Cutting Planes proof. By extending Razborov's theorem for monotone circuits, lower bounds for monotone real circuits can also be shown, and thus we obtain unconditional superpolynomial lower bounds for Cutting Planes as well.

3 Interpolation for Resolution

Theorem 1 *Resolution has feasible interpolation.*

Theorem 2 *Resolution has uniform feasible interpolation.*

Proof To show that Resolution has uniform feasible interpolation (and therefore feasible interpolation) we will give a general procedure for constructing a circuit from a resolution proof P of a formula F of the form $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$. We will then modify the procedure slightly to to deal with the monotone case.

Let F be an arbitrary formula of the form $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$.

Let P be an arbitrary DAG resolution proof of F. Use the topology of P to construct the circuit corresponding to P, ie. each node in the DAG corresponds to a node in the circuit.

The circuit examines each clause starting with the intial clauses and works towards ϕ. The value at a node of the circuit reflects whether the clause resolved at that node was derived from the $\overrightarrow{q}$ side or the $\overrightarrow{r}$ side of F. The value is 0 if the clause at the node was derived from the $\overrightarrow{q}$ side and 1 if the clause was derived from the $\overrightarrow{r}$ side under the restriction of an assignment α to $\overrightarrow{p}$.

We will say that a clause is a q-clause if it contains only variables from $\overrightarrow{q}$ or $\overrightarrow{p}$. If a clause contains only variables from $\overrightarrow{p}$ then it is a $q - clause$ only if its ancestors are q-clauses.

Similarly, we will say that a clause is a r-clause if it contains only variables from $\overrightarrow{r}$ or $\overrightarrow{r}$. If a clause contains only variables from $\overrightarrow{r}$ then it is a $r - clause$ only if its ancestors are r-clauses.

Replace each clause by a subclause which is either a q-clause or an r-clause using the following rules:

- Each initial clause is already either a q-clause or an r-clause so it remains unchanged.

- **Case 1:** Clauses within the proof of the form $\Gamma \vee \Delta$ derived from previous clauses of the form $\Gamma \vee p_k$ and $\Delta \vee \neg p_k$.

$$\frac{(\Gamma \vee p_k) \quad (\Delta \vee \neg p_k)}{(\Gamma \vee \Delta)}$$

Inductively $\Gamma \vee p_k$ is replaced by Γ' and $\Delta \vee \neg p_k$ is replaced by Δ', and we replace $\Gamma \vee \Delta$ by Γ' if $p_k = 0$ and Δ' if $p_k = 1$.

- **Case 2:** Clauses within the proof of the form $\Gamma \vee \Delta$ derived from previous clauses of the form $\Gamma \vee q_k$ and $\Delta \vee \neg q_k$.

$$\frac{(\Gamma \vee q_k) \quad (\Delta \vee \neg q_k)}{(\Gamma \vee \Delta)}$$

Inductively $\Gamma \vee q_k$ is replaced by Γ' and $\Delta \vee \neg q_k$ is replaced by Δ'. We then replace $\Gamma \vee \Delta$ with the result of resolving Γ' and Δ' on q_k unless either Γ' or Δ' does not contain q_k, for example if one of them is an r-clause. If one does not contain q_k then we replace $\Gamma \vee \Delta$ with it.

- **Case 3:** Clauses within the proof of the form $\Gamma \vee \Delta$ derived from previous clauses of the form $\Gamma \vee r_k$ and $\Delta \vee \neg r_k$.

$$\frac{(\Gamma \vee r_k) \quad (\Delta \vee \neg r_k)}{(\Gamma \vee \Delta)}$$

This case is the dual of case 2.

Now if we apply a restriction α to $\overrightarrow{p}$ and remove every clause which contains a variable from $\overrightarrow{p}$ that is assigned 1 and delete all the remaining members of $\overrightarrow{p}$ from the other clauses, we will have a valid derivation of ϕ. Furthermore, if ϕ is a q-clause, then P, α has a subproof using only the restricted clauses from $A(\overrightarrow{p}, \overrightarrow{q})$ and if ϕ is an r-clause then P, α has a subproof using only the restricted clauses from $B(\overrightarrow{p}, \overrightarrow{r})$. In effect, this procedure has split P.

We can now construct the circuit as follows.

Define the sel function as:

$$sel(p_k, x, y) = \begin{cases} x & \text{if } p_k = 0 \\ y & \text{if } p_k = 1 \end{cases}$$

This function corresponds to the formula $(p_k \wedge x) \vee (\bar{p}_k \wedge y)$. A sel gate can therefore be implemented in a circuit as a combination of two AND-gates, a NOT-gate, and an OR-gate.

Put constant 0 gates on every initial q-clause and constant 1 gates on every initial r-clause. The only other cases to consider are the three described above. They can be implemented in the following manner:

- **Case 1:** If the output of the gate corresponding to $\Gamma \vee p_k$ gets the value x and the gate corresponding to $\Delta \vee \neg p_k$ gets the value y then the gate corresponding to $\Gamma \vee \Delta$ should have the output $sel(p_k, x, y)$. We therefore put a sel gate on $\Gamma \vee \Delta$.

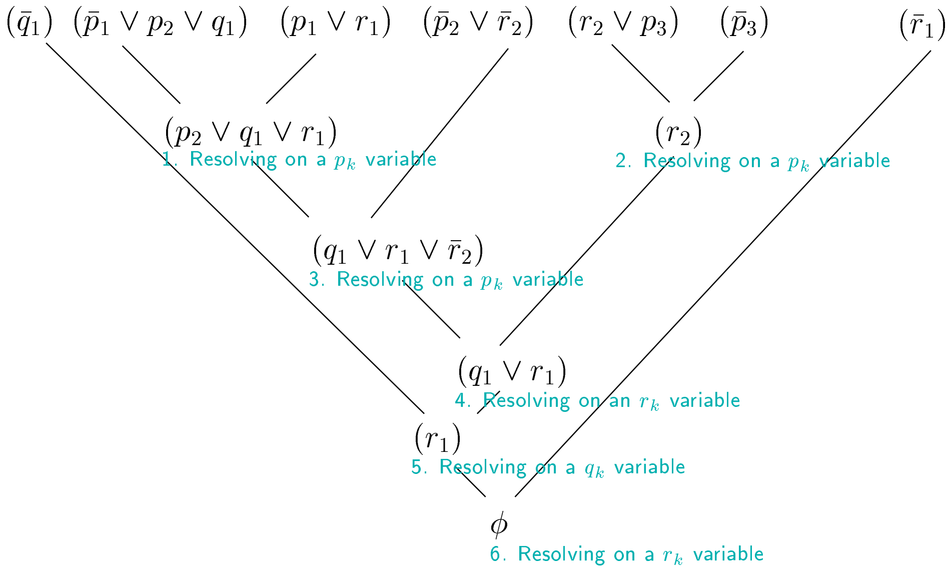
- **Case 2:** If the output of the gate corresponding to $\Gamma \vee q_k$ gets the value x and the gate corresponding to $\Delta \vee \neg q_k$ gets the value y then the gate corresponding to $\Gamma \vee \Delta$ should output 0 only if both Γ' and Δ' are q-clauses and 1 if either of the two are r-clauses. We therefore put a OR-gate corresponding to $x \vee y$ on $\Gamma \vee \Delta$.

- **Case 3:** If the output of the gate corresponding to $\Gamma \vee r_k$ gets the value x and the gate corresponding to $\Delta \vee \neg r_k$ gets the value y then the gate corresponding to $\Gamma \vee \Delta$ should output 1 only if both Γ' and Δ' are r-clauses and 0 if either of the two are q-clauses. We therefore put a AND-gate corresponding to $x \wedge y$ on $\Gamma \vee \Delta$.

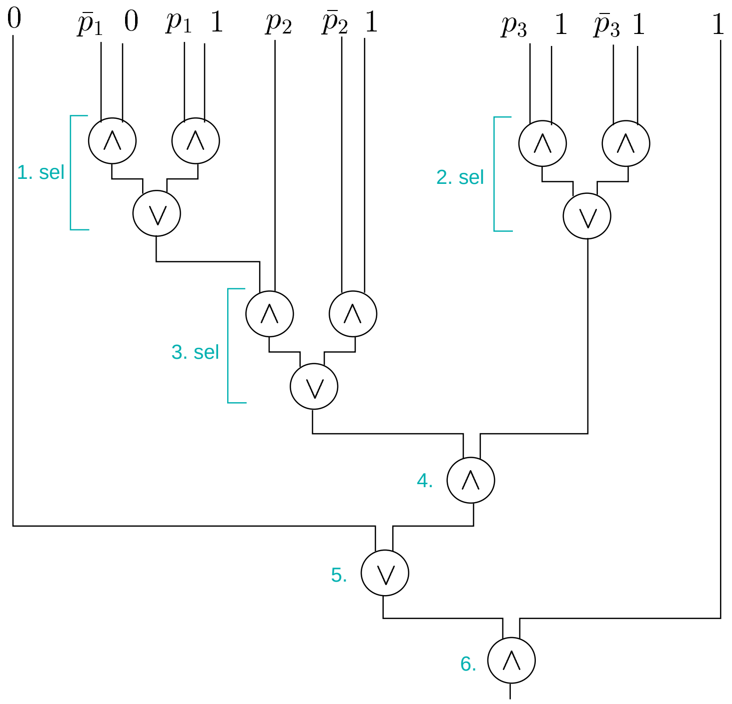
Note that in cases 2 & 3 the number of gates used is the same as the number of clauses in the proof P and case 1 only adds a linear number more, so the size of the circuit is linear in the size of P.

Example:

Let F be the unsatisfiable formula $(\neg q_1) \wedge (\neg p_1 \vee p_2 \vee q_1) \wedge (p_1 \vee r_1) \wedge (\neg p_2 \vee \neg r_2) \wedge (r_2 \vee p_3) \wedge (\neg p_3) \wedge (\neg r_1)$. Notice that F has the form $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$. The figure below shows the Resolution proof of F which will be used to construct a circuit.



To construct the circuit we replace each initial q-clause with a constant 0 and each initial r-clause with a constant 1. Then we replace each clause where we resolved on a variable from $\overrightarrow{p}$ with a sel gate, each clause where we resolved on a variable from $\overrightarrow{q}$ with an OR-gate, and each clause where we resolved on a variable from $\overrightarrow{r}$ with an AND-gate.



Theorem 3 *Resolution has monotone feasible interpolation.*

Proof The proof of this theorem is very similar to the proof above. The construction of the circuit differs only in case 1. In the monotone case variables of $\overrightarrow{p}$ occur only positively in $A(\overrightarrow{p}, \overrightarrow{q})$ or negatively in $B(\overrightarrow{p}, \overrightarrow{r})$. Without loss of generality let us assume that they occur only positively in $A(\overrightarrow{p}, \overrightarrow{q})$. Any derived q-clause will therefore retain this property. We can therefore replace $\Gamma \vee \Delta$ with Δ' in case 1 whenever Δ' is a q-clause, regardless of the assignment to p_k, since $\neg p_k$ will not occur due to the monotonicity.

The sel gate can therefore be replaced by a new monotone gate equivalent to $(p_k \vee x) \wedge y$. This new gate is monotone and differs from sel only on the input $p_k = 1$, $x = 1$, $y = 0$ which corresponds to the situation ruled out above.

4 Interpolation for Cutting Planes

The idea behind the proof of the interpolation theorem for cutting planes proof is similar to that of resolution. We want to split the proof into q-inequalities and r-inequalities by restricting the variables $\overrightarrow{p}$ which are common to both parts and by doing so we want to derive a refutation of either the q-side or the r-side.

The only cutting planes rule which allows q-clauses and r-clauses to mix is the addition rule. We will therefore not do additions when the result would be a mixed clause, but rather keep two separate inequalities. This would pose problems in an unrestricted cutting planes proof due to division. But given an assignment α to $\overrightarrow{p}$ this problem disappears since we can treat $\alpha(\overrightarrow{p})$ as part of the constant term.

Given an assignment α to $\overrightarrow{p}$, we then replace each inequality in P by two inequalities and such that the following property holds:

$\sum e_i p_i + \sum b_i q_i + \sum c_i r_i \geq D$ is replaced by:

$\sum b_i q_i \geq D_0$ and $\sum c_i r_i \geq D_1$

where $D_0 + D_1 \geq D - \sum e_i \alpha(p_i)$

This is done inductively, starting at the leaves of the Cutting Planes proof. An initial inequality from the A side of the form $\sum e_i p_i + \sum b_i q_i \geq D$ is transformed into the two inequalities $\sum b_i q_i \geq D - \sum e_i \alpha_i$ and $0 \geq 0$. Similarly an initial inequality from the B side of the form $\sum e_i p_i + \sum c_i r_i \geq D$ is replaced by the two inequalities $0 \geq 0$ and $\sum c_i r_i \geq D - \sum e_i \alpha_i$. If some inequality L_3 is derived by addition from L_1 and L_2, where L_1 and L_2 are transformed into (L_1^1, L_1^2) and (L_2^1, L_2^2) respectively, then the two inequalities associated with L_3 are obtained by summing each separately. That is, (L_3^1, L_3^2) is $(L_1^1 + L_2^1,\ L_1^2 + L_2^2)$. Similarly, if L_3 is derived from L_2 by multiplication (division), then the pair (L_3^1, L_3^2) is obtained by multiplying (dividing) by each separately.

This transformation will break the proof into two disjoint sections so that the last line no longer looks like $0 \geq 1$, but instead it is replaced by two sums, $\sum 0 q_i \geq D_0$ and $\sum 0 r_i \geq D_1$. Since $D_0 + D_1 \geq 1$ and D_0 and D_1 are integer valued, it follows that at least one of D_0 or D_1 is greater than or equal to 1. If D_0 is 1, then we can conclude that $A(\alpha, \overrightarrow{q})$ is unsatisfiable, and otherwise we can conclude that $B(\alpha, \overrightarrow{r})$ is unsatisfiable.

We have to check that the above transformation process satisies the property mentioned above. This involves checking that each of the rules preserves the inequality $D_0 + D_1 \geq D - \sum e_i \alpha(p_i)$.

The only slightly tricky case is division. But if $D_0 + D_1 \geq D - \sum e_i \alpha(p_i)$, then $\lceil \frac{D_0}{2} \rceil + \lceil \frac{D_1}{2} \rceil \geq \lceil \frac{D}{2} \rceil - \sum \frac{e_i}{2} \alpha(p_i)$ since rounding up makes numbers larger.

Since $+$, $-$, $\div$, and $\cdot$ can all be computed with polynomial time algorithms, they can also be computed with polynomial circuits. We can therefore build a polynomial circuit to compute the interpolant of cutting planes refutations. We also know that D_is have polynomial upper-bounds from a theorem by Buss and Clote which states that a cutting planes proof P can be transformed into another proof P' such that P' is at most polynomially larger than P and all the coefficients in P' have polynomially bounded binary length.

The monotone version of the above requires the use of monotone real circuits. We only sketch the argument here. For details and proofs, consult Pudlak's paper.

Definition A *monotone real circuit* is a circuit which computes with real numbers and uses arbitrary non-decreasing real unary and binary functions as gate. We say that a monotone real circuit computes a boolean function if for all 0/1 inputs the circuit outputs 0 or 1.

Let P be a Cutting Planes refutation of $A(\overrightarrow{p}, \overrightarrow{q}) \wedge B(\overrightarrow{p}, \overrightarrow{r})$, where $\overrightarrow{p}$ occurs only positively in A. Then it is possible to obtain a monotone real circuit that computes an interpolant for this formula of size polynomial in the size of P, and the number of variables. The idea is as follows. The circuit only needs to compute the constant $-D_0$ corresponding to the final inequality. This can be done by computing successively $-D_0$ from each pair. Again this can be done by using the graph of the proof to obtain the monotone real circuit. The gate produces the new $-D_0$ value from the previous ones. The input to the monotone real circuit is 0, 1, and the variables of $\overrightarrow{p}$. The gates need to be able to add two integers, and divide by 2 with downwards rounding. It can be shown that these operations are both monotone, and thus the entire circuit can be computed by a monotone real circuit.