Introduction to Quantum Branching Programs

Chris Pollett

(based on joint work with Farid Ablayev, Aida Gainutdinova, Marek Karpinski, and Cristopher Moore)

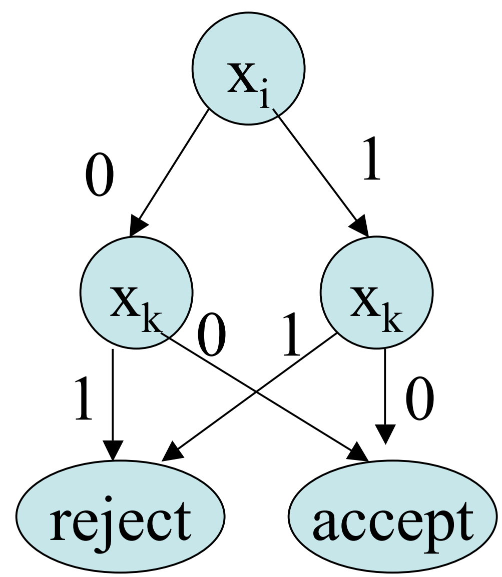
Apr. 4, 2006.

Outline

- Classical Branching Programs
- Quantum and Stochastic Branching Programs
- The Power of Width 2 Programs
- Classical Simulations.

Classical Branching Programs

- Directed acyclic graphs
- Nodes labeled with variables, have two outgoing edges 0, 1.
- Have a single source node
- Have distinguished sink nodes labeled accept.
- Given a setting to the variables follow path of 0's and 1's according to the variable values to see if accept.



More on Branching Programs

- Restricted versions of branching programs have been used for hardware verification, and other CAD applications.
- We will be interested in programs where the number of nodes at each level is the same and the variable queried at each node at a level is the same.
- Recall a **monoid** is a set M with an associative operation * on it as well as some identity element: 1*a =a*1 = a.
- We can view the operation that maps us from one level of the program to the next as coming from a monoid.

Barrington's Result

- The **size** of a program is the number of nodes in it.
- The **width** is the maximum number of nodes at a level.
- A family of branching programs $\{B_n\}$ can be viewed as computing a function, if the n-th member of the family can computes the values of the function for n-bit inputs.
- **Barrington** showed that the languages recognized by constant-width, polynomial-sized families of permutation branching programs are precisely the languages in NC^1, those computed by polynomial size, log depth circuit.
- He showed only need width 5 programs to get this class.

Quantum and Stochastic Programs

- As mentioned above, Barrington considered programs where the level to level transition is given by a permutation.
- It is natural to ask what happens when one uses a unitary operator or a stochastic operator to do the level to level transition.
- The answer is one gets Quantum and Stochastic Branching Programs!

More on Quantum and Stochastic Branching Programs

- A **branching program** of width k is a triple P=(T, |s>, Accept).
- T is a sequence of instructions (i_j, $A_i(0)$, $A_i(1)$)
- |s> is the start state (assume have a k-state system)
- Accept is a subset of values in {0, …, k-1} which are accepting.
- The program computes on input $x_1,..,x_n$ the vector

$$|m(\vec{x})\rangle = \prod_{j=L}^{1} A_j(x_{i_j})|s\rangle$$

- For stochastic programs the A_i are dim k matrices with column summing to 1.
- For quantum programs the A_i are dim k unitary matrices.
- The acceptance probabilities for the deterministic, stochastic and quantum case are defined as respectively

$$Pr(m) = \sum_{i\in\text{Accept}} \langle i|m\rangle \qquad Pr(m) = \sum_{i\in\text{Accept}} |\langle i|m\rangle|^2$$

Yet More on Branching Programs

- So we assume we only measure once.
- We will consider the usual possible acceptance criteria: **bounded-error** (accept if probability is >= 1/2 +e), **unbounded-error** (accept if probability is >=1/2), and **exact acceptance** (accept if probability is 1).
- A computation path in a program is **inconsistent** if on an input when a variable is queried more than once we use a different answer than the original one at some point.; otherwise it is **consistent**.
- Let A = {|m> : |m> is the result of a path Pr(|m>) >1/2+e} and R = {|m> : |m> is the result of a path Pr(|m>) < 1/2 -e}
- A branching program is called **syntactic** if the the A and rejecting R of the program form a partition of all the final possible states reachable on any path through the program (consistent or inconsistent).

The Power of Width 2 Programs

Theorem NC^1 is precisely the class of languages recognized by polynomial size, width 2 syntactic, quantum branching programs with exact acceptance criteria.

Proof Idea. Barrington's proof needed the levels of programs to come from a nonsolvable group. In his case, A_5. Notice U(2) is a double cover of SO(3) (ask any computer graphics person). SO(3) contains the group of three-dimensional rotations of the icosahedron which is A_5. So NC^1 is contained in the above quantum programs. For the other direction we need results about simulating quantum programs.

Classical Simulations

Theorem Let P be a width k, syntactic stochastic or quantum program of length l that recognizes a language L with probability 1/2 + e. Then there exists a determinstic program P'of width k' and length l that recognizes L where k' in the stochastic and quantum case are respectively: $k' \le (1/e)^{k-1}$ and $k' \le (2/e)^{2k}$.

Idea of the simulation

- We define an equivalence relations on the states V_t of level t of the program.
- Two states are called equivalent if they lead to the same outcome.
- We then use the fact that stochastic and unitary matrices do not increase distances.
- We prove that two states at level j in different equivalence classes must be at least 4e apart in the stochastic case and 2e apart in the quantum case.
- We then count the number of disjoint balls of these size that can fit in the sphere of size 1 according to the appropriate metric to get the bound.

Example

- The NC^1 result means there are syntactic, width-2, polynomial size quantum branching programs for multiplication.
- On the other hand, Ablayev and Karpinski have shown an exponential lower bound on read-once, randomized OBDDs.
- This can be used to show that width-2 doubly stochastic programs need exponential size for multiplication.

Conclusion

- It would be interesting to remove the syntactic condition from our upper bound results.
- Branching programs are also connected to resolution refutation systems. It would be interesting to use this to come up with “quantum proof systems” (different from Arthur Merlin setting).
- Upper bound results might be useful in analysing consistent histories?