Chapter 6: Formal Hardware Verification

July 8, 2003

1 Global Extremum Theorems

Lemma 1.

$$\min \mathrm{E[C]}(\mathbf{X}) \leq \mathrm{mean}\, \mathrm{E[C]}(\mathbf{X}) \leq \max \mathrm{E[C]}(\mathbf{X}) \tag{1}$$

Theorem 1. $\min \mathrm{E[C]}(\mathbf{X})$ *can always be computed as a function of the bounds* $x_{1\mathrm{l}}$, $x_{1\mathrm{r}}$, $x_{2\mathrm{l}}$, $x_{2\mathrm{r}}$, ... , x_{nl}, x_{nr},

Proof. Suppose, we assume that $\mathrm{E}_{\min} = \min \mathrm{E[C]}(\mathbf{X})$ is not a function of the bounds, in variable x_i. We do not make any assumptions of the other variables. Now,

$$\mathrm{E}_{\min} = \mathrm{E}(x_i = 0) + x_i \frac{\partial \mathrm{E}_{\min}}{\partial x_i} \tag{2}$$

If $\frac{\partial \mathrm{E}_{\min}}{\partial x_i} > 0$, but x_i is not x_{il}, it is not the global minimum, since if x_i was replaced by x_{il},it would be even lesser. If $\frac{\partial \mathrm{E}_{\min}}{\partial x_i} < 0$, but x_i is not x_{ir}, it is not the global minimum, since if x_i was replaced by x_{ir}, the function would be even lesser.
If $\frac{\partial \mathrm{E}_{\min}}{\partial x_i} = 0$, the minimal output can be a function of any x_i. Since, we are proving that output bound can always be computed as a function x_{ir}, x_{il}, as opposed to, proving that the output bound, is always a function of x_{ir}, x_{il}, our proof holds, for the general case. □

Theorem 2. $\max \mathrm{E[C]}(\mathbf{X})$ *can always be computed as a function of the bounds* $x_{1\mathrm{l}}$, $x_{1\mathrm{r}}$, $x_{2\mathrm{l}}$, $x_{2\mathrm{r}}$, ... , x_{nl}, x_{nr}.

Proof. The proof is similar to the above, except with conditions reversed. □

2 Bounds of Functions

Lemma 2. *Any circuit function can be writtern as*

$$\mathrm{C}(\mathbf{Y}) = a_0 + \sum_i b_i y_j + \sum_{i \neq j} c_{ij} y_i y_j + \sum_{i \neq j \neq k} d_{ijk} y_i y_j y_k + ... \tag{3}$$

where a_0, b_i, c_{ij} , d_{ijk},etc can be either -1,0 or 1.

To compute the expectation value of the function, for weightset $\mathbf{X}$. We just take

$$\mathrm{E[C]}(\mathbf{X}) = a_0 + \sum_i b_i x_j + \sum_{i \neq j} c_{ij} x_i x_j + \sum_{i \neq j \neq k} d_{ijk} x_i x_j x_k + ... \tag{4}$$

The trival proof follows from Parker-Mcklucsky alegbra.
Notice, that suppose, if we replace $x_i = \frac{x_{il}+x_{ir}}{2}$. Then, take

$$\mathrm{E[C]}(\mathbf{X}) = a_0 + \sum_i b_i \frac{x_{il}+x_{ir}}{2} + \sum_{i \neq j} c_{ij} \frac{x_{il}+x_{ir}}{2} \frac{x_{jl}+x_{jr}}{2} + \tag{5}$$

$$\sum_{i \neq j \neq k} d_{ijk} \frac{x_{il}+x_{ir}}{2} \frac{x_{jl}+x_{jr}}{2} \frac{x_{kl}+x_{kr}}{2} + ... \tag{6}$$

Directly, from that expression, we can see that the expected value of average input bound probability is also average of circuit probabilities computed by taking all permutation of input probability bounds.

3 Lemmas of Distributions

Lemma 3. *Given that a log-normal distribution of variable y, such that* $|\max\log(y) - \min\log(y)| < \delta$*, the distribution is also approximately gaussian.*

Proof. Take

$$p(y, \langle lt \rangle, \sigma_{lt}) = \frac{1}{\sqrt{2\pi}\sigma_{lt} y} \exp\left(-\frac{(\log(y) - \langle lt \rangle)^2}{2\sigma_{lt}^2}\right) \tag{7}$$

Pick a $\log(c)$ in the interval $[\min\ \log\ y, \max\ \log : y]$. Now ,

$$\log(c+y) = \log(c) + log(1 + y/c) \tag{8}$$

$$\approx \log(c) + y/c \tag{9}$$

The expected value, of $\log(c+y)$ is about $\log(c) + \frac{\langle y \rangle}{c}$. And variance is about $\frac{1}{c^2}\left(\langle y^2 \rangle - \langle y \rangle^2\right)$. And $\frac{1}{c+y}$ is about $\frac{1}{c}(1 - y/c)$. □

Lemma 4. *Distribution of* $\mathrm{E[C]}(\mathbf{X})$ *evaluated with each permutation of the input probability bounds is gaussian towards the center and log-normal asymtopically towards the ends.*

4 Algorithm for Probabilistic ATG

Suppose we want to find a binary input vector $\mathbf{X}$ that sets the circuit C($\mathbf{X}$) to 1. The following algorithm will find a solution,if one exists. It is terribly inefficient, for problems, where $\mathbf{X}$ could be guessed easily. However, when $\mathbf{X}$ is not easy to guess, the algorithm, combined with PREDICT, is most likely, to be a faster method for finding a solution.

4.1 The Algorithm

Set $x_i = 0.5$.If E[C]($\mathbf{X}|x_i = 1$) > E[C]($\mathbf{X}|x_i = 0$), set $x_i = 1$, else $x_i = 0$. And Move onto the next x_i.

Proof. For any $\mathbf{X}$, if E[C]($\mathbf{X}$) > 0, then, weightset $\mathbf{X}$ will contain atleast one binary vector, the sets C to 1.
Remember that every other variable is fixed, except for x_i. When E[C]($\mathbf{X}|x_i = 1$) > E[C]($\mathbf{X}|x_i = 0$) implies that there are more 1s, if $x_i = 1$, than when $x_i = 0$. This follows from the fact that E[C]($\mathbf{X}|x_i = a$) is the probability of getting 1, at the output, given that $x_i = a$.
Setting $x_i = 1$, is guaranteed, to raise the probability of finding a 1. On the other hand, When E[C]($\mathbf{X}|x_i = 1$) < E[C]($\mathbf{X}|x_i = 0$), implies that there are more 1s, if $x_i = 0$. Setting $x_i = 0$, is guaranteed, to raise the probability of finding a 1. □

The method can be used to set C($\mathbf{X}$) to 0, as follows. Set $x_i = 0.5$.If E[C]($\mathbf{X}|x_i = 1$) < E[C]($\mathbf{X}|x_i = 0$), set $x_i = 1$, else $x_i = 0$. And Move onto the next x_i. The proof is similar to the above.

5 Generalized Approximate Cutting Algorithm

Suppose $[x_{1\mathrm{l}}, x_{1\mathrm{r}}]$, $[x_{2\mathrm{l}}, x_{2\mathrm{r}}]$ ... $[x_{\mathrm{nl}}, x_{\mathrm{nr}}]$ are uncertainty bounds in the inputs, and suppose function E[C]($\mathbf{Y}$) is exactly calculatable for an aribitarty weight set $[y_1, y_2, ..., y_n]$,the uncertainty bounds in the output of the function can be estimated very accurately by the following procedure.

Take the input set $[x_{1\mathrm{l}}, x_{1\mathrm{r}}]$, $[x_{2\mathrm{l}}, x_{2\mathrm{r}}]$,all the way, upto $[x_{\mathrm{nl}}, x_{\mathrm{nr}}]$ and find the average of each bound,

$$\overline{x}_{\mathrm{i}} = \frac{x_{\mathrm{il}} + x_{\mathrm{ir}}}{2} \tag{10}$$

And calculate

$$m = \mathrm{E[C]}(\overline{\mathbf{X}}) \tag{11}$$

Now, change one $\overline{x}_i$ to either x_{il} or x_{ir} without changing the others. And calculate

$$m_{ir} = \mathrm{E[C]}(\overline{\mathbf{X}}|\overline{x}_i = x_{ir}) \tag{12}$$

and

$$m_{il} = \mathrm{E[C]}(\overline{\mathbf{X}}|\overline{x}_i = x_{il}) \tag{13}$$

To find the 'near global' minimum, take the least m_{iy} and set the $\overline{x}_i$ to x_{iy}. That is to say that if the least m_{iy} was m_{1r}, then $\overline{x}_i = x_{1r}$. And recurse.
To find the 'near global' maximum, take the greatest m_{iy} and set the $\overline{x}_i$ to x_{iy}, without changing the other . That is to say that if the greatest m_{iy} was m_{2l}, then $\overline{x}_i = x_{2l}$. And recurse.

Corollary 1. *Given the gaussian assumption, then the least m_{ix}. must be atleast less than half of the the permutations, at atmost greater than or equal to the greatest value of the least 1/4th of the total population. Call the cut off value c.*

Proof. Evidently since m_{ix} is less than m_i, from the gaussian assumption, m_{ix} is less than half of those permutations.
To prove the second part, suppose, we assume to the contrary that m_{ix} is less than c. The number of permutations used in computing m_{ix} is half of the permutation used for calculating m_i. And the number of values less than m_{ix} in calculating is, half of that population. So, there are already 1/4th of the terms below m_{ix}. But by suppostition, m_{ix} is less than the c. From the gaussian assumption, there must be terms between the cut off and m_{ix}. If that were true, that the cutoff isnt really, at 1/4 of the population, but contains more terms. Since, this is a contradiction, m_{ix} has to lie above the cutoff. □

Proof remains the same for each successive step.

Corollary 2. *Now, suppose, the gaussian at each step has the mean and standard deviation parameters u_j and σ_j, the number of implicit comparision,performed by this procedure at this step is atleast*

$$2^n \cdot 0.5^j \Phi\left(\frac{u_j - u_{j-1}}{\sigma_j}\right) \tag{14}$$

Proof. This can be seen, in the following light. $2^n \cdot 0.5^j$ is the amount of the population, the procedure acts on, each time. Every time, we succeed, in finding an m_{ix}, we have implicity compared values of the population between the previous mean u_{j-1} and the current one, u_j. From the gaussian assumption, we get the size, of the population inbetween, to be

$$\Phi\left(\frac{u_j - u_{j-1}}{\sigma_j}\right) \tag{15}$$

□

In the end, the procedure, implicitly does about

$$2^n \cdot 0.5 + \sum_j 2^n \cdot 0.5^j \Phi\left(\frac{u_j - u_{j-1}}{\sigma_j}\right) \tag{16}$$

comparisions,at least.
Something important to note is that, the number of comparision will never exceed 2^n. It is surprising to note that, for a circuit with 32-input, procedure, implicity compares billions of permutations, in $O(n^2)$ time. One thing to note,is that, athough it does compare most of them, in short swipes, it doesnt compare them all.
However, the number of comparisions, is much greater than that. The total number of compared values, between the final and the intial state is $0.5 \cdot 2^n + 2^n \Phi\left(\frac{u_n - u_0}{\sigma_0}\right)$.

6 Cutting Formulas

Here are some proof for some basic gates.

6.1 AND gate

$$\mathrm{E[C]}(\mathbf{X}) = x_1 x_2 x_3 \cdots x_n \tag{17}$$

without any difficulty, u must see that x_i for the lower bound

$$x_i = x_{il} \tag{18}$$

because $\overline{x}_1 \overline{x}_2 \cdots x_{il} \cdots \overline{x}_n$ is less than $\overline{x}_1 \overline{x}_2 \cdots x_{ir} \cdots \overline{x}_n$. and for the upper bound,

$$x_i = x_{ir} \tag{19}$$

6.2 OR gate

$$\mathrm{E[C]}(\mathbf{X}) = 1 - (1 - x_1)(1 - x_2) \cdots (1 - x_n) \tag{20}$$

Now $1-(1-\overline{x}_1)\cdots(1-x_{\mathrm{il}})\cdots(1-\overline{x}_n) < 1-(1-\overline{x}_1)\cdots(1-x_{\mathrm{ir}})\cdots(1-\overline{x}_n)$, so the lower bound is given by

$$x_i = x_{il} \tag{21}$$

And the upper bound is given by

$$x_i = x_{ir} \tag{22}$$

6.3 NOT gate

$$\mathrm{E[C]}(X) = 1 - x \tag{23}$$

Now, $1-x_r$ is the left bound and $1-x_l$ is the right bound given by our procedure.

7 Algorithm Complexity Hypothesis

Let's say we like to solve a boolean satisfiability $\mathrm{E[C]}(x_1, x_2, ..., x_n) = c$ where x_n is binary. Let's say that, you use a probabilistic method, i.e, we randomly start inside the space [0,1]x[0,1]....[0,1] and try to converge on to a solution.

Suppose the probability of naturally guessing the solution is really small, then nature of problem having P or NP-complete order depends on this one condition. For a moment, let us denote the exact $\mathrm{E[C]}(y_1, y_2, ..., y_n)$ by p. For any given real y_1,y_2...,y_n, if we can calculate the $\mathrm{E'[C]}(y_1, y_2, ..., y_n)$ approximately, in $O(N^k)$ always such that

$$\mathrm{E'[C]}(y_1, y_2, ..., y_n) = p \cdot (1 \pm e) \tag{24}$$

where $e \leq 1/2$, then that problem is no longer NP complete, and can always be solved in P time. If the error is always additive,

$$\mathrm{E'[C]}(y_1, y_2, ..., y_n) = p \pm e \tag{25}$$

Our hypthosis, does not apply to that problem.

Note: This is a propostion, without proof.

On the other hand, for any given y_1,y_2...,y_n, if we can calculate the $\mathrm{E'[C]}(y_1, y_2, ..., y_n)$ approximately, in $O(N^k)$, *on average* such that

$$\mathrm{E'[C]}(y_1, y_2, ..., y_n) = p \cdot (1 \pm e) \tag{26}$$

where $e \leq 1/2$, then that problem can *on average* be solved in P-time.If the error is additive, on average,

$$\mathrm{E'[C]}(y_1, y_2, ..., y_3) = p \pm e \tag{27}$$

Our hypthosis, does not apply to that problem.

7.1 Uses

We speculate that the result could be very useful in proving P or NP-completeness of factoring algorithms. The factoring problem is a boolean statisfability problem in disguise. Here, we have a multipler with (3/2)n-inputs [1] and n-outputs

[1] one n-bit multiplicand and another n/2 -bit multiplicand

[2]. Now we would like to find input values(factors) that generate the number that requires factoring, at the output. The way we do it, is by simulatenously requiring all output bits to match the bits of the result. Now, if we can, on average, interpolate the functional space of the multiplier in $O(N^k)$, then factoring ,on average, could be done in polynomial time.

[2]the n-bit number that requires factoring