Why McMillan's construction is an interpolant.

The resolution proof that $A(x,y)B(y,z)=0$, contains a set of sub-clauses of *A*, $\hat{A}(x,y)$, and a set of sub-clauses of *B*, $\hat{B}(y,z)$, so that $\hat{A}(x,y)\hat{B}(y,z)=0$ has been proved by SAT. Now form the AND DAG with the same structure as the resolution DAG; its leaves are the individual clauses of $\hat{A}(x,y)$ and $\hat{B}(y,z)$ and its internal nodes are all AND nodes. Each node is associated a pivot variable of the resolution proof. Take this DAG and replace the *A* clauses with their global parts $g_k(y)$, the *B* clauses with 1, and the AND operator at the nodes where the pivot variable was an *x* variable with OR. This gives the computational DAG as given by McMillan for the interpolant $I(y)$. Further if we put back the clauses of $\hat{B}$, then we have formed a computational DAG which computes some function $g(y,z)$, which we will use later.

Since $A(x,y)\subseteq\hat{A}(x,y)$, we first prove that $\hat{A}(x,y)\subseteq I(y)$. Note that one computation DAG for $\hat{A}$ is obtained from the AND DAG, associated with the resolution proof, of $\hat{A}(x,y)\hat{B}(y,z)$, where the leaves are the original clauses of $\hat{A}$ but with all $\hat{B}$ clauses set to 1. Each internal node of this DAG represents a function and each is a unate function of the leaves. Thus if we replace any internal function with a larger one, then the function at the root of the DAG increases too. Consider a node where x_i was the pivot variable associated with the node, let the child function associated with the positive literal be denoted by *p*, and the other child function by *n*. (Recall at a pivot, a resolution was done where one side had the positive literal and the other the negative literal.) The function of the AND node is $f=pn=x_i p_{x_i} n_{x_i}+\overline{x}_i p_{\overline{x}_i} n_{\overline{x}_i}\subseteq n_{x_i}+p_{\overline{x}_i}$. Thus the AND node can be upper-bounded by the OR of two other functions. We argue that these two functions are precisely the ones that are created by McMillan's construction. Since the cofactor operator commutes with the AND and OR operators, the cofactor can be pushed all the way to the leaves of the DAG. Suppose we are following the n_{x_i} side. At a leaf with an $\hat{A}$ clause, the local part $l_k(x)$ of the clause at the leaf, cannot have the x_i literal in it (since n_{x_i} is being computed). Thus at such a leaf, after all cofactors have been pushed down to the leaves, the clause $l_k(x)+g_k(y)$ becomes only its global part, $g_k(y)$, which is what is given by the McMillan construction for $I(y)$ at each $\hat{A}$ leaf. Thus the computational DAG for $\hat{A}(x,y)$ has been transformed into the one for $I(y)$ by operations which only increased some functions at internal nodes. Thus, $\hat{A}(x,y)\subseteq I(y)$.

To show that $I(y)\hat{B}(y,z)=0$, consider the computational DAG where $\hat{A}(x,y)$ clauses have been replaced by $\hat{A}(y)$ clauses, the OR nodes have been inserted, and the $\hat{B}(y,z)$ have been left alone. This DAG computes some function $g(y,z)$. We first show that $g(y,z)\equiv 0$ and then use this to prove $I(y)\hat{B}(y,z)=0$. Again, the DAG is unate in the leaves. We will increase this function by increasing the functions computed at its internal

AND nodes. At an AND node, the pivot variable was a y or z variable, denoted by w_i. We increase the function at every such node as follows:

$$f = np = w_i(np)_{w_i} + \overline{w}_i(np)_{\overline{w}_i} \subseteq w_i n_{w_i} + \overline{w}_i p_{\overline{w}_i}.$$

Again the cofactoring operation can be pushed down to the leaves, implying that each literal appearing in any leaf is evaluated to 0. Thus each leaf evaluates to 0 since all variables are eventually resolved out, because resolution derived the empty clause. This implies that $g(x, y) \equiv 0$, since the larger function computes is identically 0.

Now assume that $(\hat{y}, \hat{z})$ is an assignment such that $\hat{B}(\hat{y}, \hat{z}) = 1$. Then every clause of $\hat{B}$ is 1. Plugging in 1 for each $\hat{B}$ clause in the $g(y, z)$ DAG results in $I(y)$, and thus this computes $I(\hat{y})$. However, since $g(y, z) \equiv 0$, then $I(\hat{y})$ must be 0. Thus $\hat{B} \Rightarrow \overline{I}$, or $I\hat{B} \equiv 0$.