Logic Synthesis & Verification, Fall 2021

National Taiwan University

Programming Assignment 2

submission period: 2021/12/24 11:00-13:00

Submission Guidelines. Please send a pull request to the branch named with your student ID during the submission periods. Please develop your code under "`src/ext-lsv`" (For those who develop your PA1 in "`ext_<your_student_ID>`", please move your code to "`src/ext-lsv`" and remove "`ext_<your_student_ID>`").

1 [OR Bi-Decomposition of Functions]

(100%)

Overview. Write a procedure in ABC that decides whether each circuit PO $f(X)$ is OR bi-decomposable. Integrate this procedure into ABC, so that after reading in a circuit by the command "`read`", running the command "`lsv_or_bidec`" would invoke your code.

Preliminaries. A function $f(X)$ is OR bi-decomposable under a variable partition of its support $X = \{X_A|X_B|X_C\}$ if f can be written as $f(X) = f_A(X_A, X_C) \vee f_B(X_B, X_C)$. A variable partition is *non-trivial* if $X_A \neq \emptyset$ and $X_B \neq \emptyset$. In the following, only non-trivial variable partition is of concern. Proposition 1 states the sufficient and necessary condition for a function $f(X)$ to be OR bi-decomposable under a given support partition.

Proposition 1. *A function $f(X)$ can be written as $f_A(X_A, X_C) \vee f_B(X_B, X_C)$ for some function f_A and f_B if and only if the Boolean formula $f(X_A, X_B, X_C) \wedge \neg f(X'_A, X_B, X_C) \wedge \neg f(X_A, X'_B, X_C)$ is unsatisfiable, where X'_A, X'_B are renamed versions of X_A, X_B, respectively.*

Proposition 1 assumes a given variable partition $X = \{X_A|X_B|X_C\}$. To automate the process of finding a variable partition, two controlling variables α_i, β_i are introduced, for each $x_i \in X$. Consider the formula,

$$f(X) \wedge \neg f(X') \wedge \bigwedge_i ((x_i \equiv x'_i) \vee \alpha_i) \wedge \neg f(X'') \wedge \bigwedge_i ((x_i \equiv x''_i) \vee \beta_i), \qquad (1)$$

where $x'_i \in X'$ and $x''_i \in X''$ are renamed versions of $x_i \in X$. Note that $(\alpha_i, \beta_i) = (0,0), (0,1), (1,0), (1,1)$ indicates $x_i \in X_C$, $x_i \in X_B$, $x_i \in X_A$, and x_i can be in either of X_A and X_B, respectively.

Your goal is to find an unit assumption on the controlling variables α_i, β_i that makes formula 1 unsatisfiable, i.e. a way of partitioning X so that f is

OR bi-decomposable under this variable partition. Under an unsatisfiable unit assumption, the SAT solver will return a final conflict clause consisting of only the controlling variables. Every literal in the final conflict clause is of positive phase because the conflict arises from a subset of the controlling variables set to 0. It reveals that setting the variables in the final conflict clause to 0 is sufficient to make formula 1 unsatisfiable. For example, the final conflict clause $(\alpha_1 + \beta_1 + \alpha_2 + \beta_3)$ indicates that setting $\alpha_1 = \beta_1 = \alpha_2 = \beta_3 = 0$ is sufficient for unsatisfiability, so settting $\beta_2 = \alpha_3 = 1$ doesn't affect the unsatisfiability, which suggests the variable partition $x_1 \in X_C$, $x_2 \in X_B$, $x_3 \in X_A$.

To avoid finding a trivial partition, you can initially specify two distinct variables x_a, x_b and force $x_a \in X_A$, $x_b \in X_B$. That is, set the unit asssumption $(\alpha_a, \beta_b) = (1, 0)$, $(\alpha_b, \beta_b) = (0, 1)$ and $(\alpha_i, \beta_i) = (0, 0)$, for $i \neq a, b$. This is called a seed variable partition. If formula 1 is unsatisfiable under a seed partition, then the corresponding bi-decomposition is successful. Otherwise, if the seed partition fails, you should try another one. For a given function $f(X)$ with $|X| = n$, the existence of non-trivial OR bi-decomposition can be checked with at most $(n-1) + \ldots + 1 = n(n-1)/2$ different seed partitions. For more details, you may refer to [1] `https://ieeexplore.ieee.org/document/4555896`.

Output Format. The format of your printing message is as follows.

```
PO <po1-name> support partition: 1
210200001101
PO <po2-name> support partition: 0
PO <po3-name> support partition: 1
212111000
```

Print lines of "`PO <po-name>...`" according to the order of `Abc_NtkForEachPo()`. For each PO, use "`Abc_NtkCreateCone()`" to extract the cone of the PO and its support set. In each line of "`PO <po-name> support partition:`", print the names of POs returned by function `Abc_ObjName()`. Print "0" after "support partition: " if there is no valid non-trivial partition; print "1" if there is a valid non-trivial partition, and print the partition you find in the next line.

We use an integer string to represent the variable partition. Let 0 represents $x \in X_C$, 1 represents $x \in X_B$, 2 represents $x \in X_A$. For example, the string `212111000` indicates that the first support variable is in X_A, the second in X_B, the third in X_A , . . . , the last in X_C.

References

1. R.-R. Lee, J.-H. R. Jiang, and W.-L. Hung. Bi-decomposing large Boolean functions via interpolation and satisfiability solving. In *ACM/IEEE Design Automation Conference*, pages 636–641, 2008.