Efficient Cancer Therapy using Boolean Networks and Max-SAT-based ATPG

Pey-Chang Kent Lin, Sunil P. Khatri
Department of ECE, Texas A&M University, College Station TX 77843
kentlin@tamu.edu, sunilkhatri@tamu.edu

Abstract— **Cancer and other gene related diseases are usually caused by a failure in the signaling pathway between genes and cells. These failures can occur in different areas of the gene regulatory network, but can be abstracted as faults in the regulatory function. For effective cancer treatment, it is imperative to identify faults and select appropriate drugs to treat the fault. In this paper, we present an extensible Max-SAT based automatic test pattern generation (ATPG) algorithm for cancer therapy. This ATPG algorithm is based on Boolean Satisfiability (SAT) and utilizes the stuck-at fault model for representing signalling faults. A weighted partial Max-SAT formulation is used to enable selection of the most effective drug. Several usage cases as presented for fault identification and drug selection. These include the identification of testable faults, optimal drug selection for single/multiple known faults, and optimal drug selection for overall fault coverage. Experimental results on growth factor (GF) signaling pathways demonstrate that our algorithm is flexible, and can yield an exact solution for each feature in much less than 1 second.**

I. Introduction

In all organisms, cell function is supported by the interaction of genes and protein products, forming a interconnected network called the gene regulatory network (GRN) [1]. The interaction or communication between genes and cells is a highly complex and mutivariate. Cancer and gene-related diseases are often the result of a failure in the signalling, leading to incorrect gene regulation and its associated functions.

The modeling of the gene interactions is thus highly important for understanding the mechanism and therapy of cancer. Because genes are observed to have a switch-like expression (active or inactive), the Boolean network model [2] has become popular for representing the GRN. In the Boolean network, the genes and biochemical pathways are represented as logic gates, much like in an integrated circuit (IC). This network can be extended to include signaling failures and defects in the GRN, which are represented as faulty lines in the circuit [3].

The issue of faults in circuits is well understood in electronic testing. For example, in chip manufacturing, circuits are typically tested to check that the IC is defect free before vendoring. Manufacturing defects manifest themselves as logical faults modeled as lines (wires) stuck-at '1' or '0'. Using this *stuck-at fault model*, automatic test pattern generation (ATPG) algorithms determine a set of tests (bit vectors on the inputs of the circuit) to test for stuck-at faults in the circuit.

In this paper, we use the fault model for the GRN [3] and employ ATPG techniques to determine a drug vector (set of drugs) to rectify the fault. The ATPG algorithm is developed as a Boolean satisfiability (SAT) based method, where the Boolean network is transformed into a conjunctive normal form (CNF) expression and solved for satisfiability to find the drug vector. In therapy, the goal is to treat the cancer (represented by one or more faults) using drugs with the least negative impact on the patient, ideally by prescribing the fewest number of drugs necessary to avoid unnecessary side-effects and cost. The SAT method is further extended by assigning weights to the circuit outputs and drug vectors, and solved with a weighted partial Max-SAT to find the optimal set of drugs to fix or rectify the fault.

The key contributions of this paper are:

- In constrat to previous approaches [3] which performs a explicit search, we develop an implicit SAT-based ATPG approach to model and identify detectable faults (single and multiple) in a Boolean network.
- By assigning weights to model output and drug vectors, we use a weighted parital Max-SAT formulation to determine the optimum selection of drugs to rectify a specific fault.
- Our approach can be trivially extended to handle multiple faults.
- We utilize the above techniques for drug therapy to select the minimum set of drugs to provide the best coverage across all single/multiple faults.

The remainder of this paper is organized as follows. Section II discusses previous work in this area. In Section III we introduce fault-modeling and Boolean satisfiability and describe our approach for drug therapy. Section IV presents experimental results from a applying our methods to a biological example, while conclusions are drawn in Section V.

II. Previous Work

In the actual GRN, the gene expression or protein concentration is continuous. However, in this paper, the Boolean network (BN) [2] is chosen as preferred network for modeling the GRN for several reasons. First, it has been observed that many genes exhibit a switch-like ON/OFF activity in terms of their expression [4]. Second, a discrete model like the BN is relatively simple and easy to analyze and simulate. And lastly, there are many logic synthesis and test algorithms already developed in circuit design and testing that can be applied to the Boolean network.

In [3], the authors proposed modeling cancer as faults in the signaling network and applied fault analysis for drug intervention. Cancer is a disease that arise from fault(s) in the network leading to loss of cell cycle control and uncontrolled cell proliferation. Therapy involves both identification of the fault and a suitable drug combination. This paper considered the growth factor (GF) signaling pathways which are often associated with proliferation of cancer. The GRN is modeled using Boolean logic gates and all possible single faults are enumerated. Drugs were also simulated to determine the effectiveness of drug combinations towards each fault.

The method proposed in [3] is an ATPG technique in principle. Our approach is similar to [3] in using the BN and modeling cancer as faults in the network. However, the differences are several. Instead of explicit enumeration of the BN, we use an extensible, implicit SAT-based ATPG approach to efficiently model and identify faults, and perform drug selection. Further, unlike [3], we include weighted clauses for outputs and drugs in the SAT formulation. Using this, the algorithm can implicitly and efficiently determine the drug combination which is maximally effective. Finally, our approach can handle multiple faults easily. The runtimes of our approach are typically much less than a second per fault.

In the past, ATPG has been extensively studied in research and industry. One such ATPG technique is the SAT-based ATPG [5] which translates the testing condition into a SAT instance that retains the circuit structure. A test for the fault can then be found by invoking a SAT solver. In the context of cancer therapy, we extend the SAT based approach to handle drugs and multiple faults.

III. Our Approach

A. Fault Terminology

Definition 1: A manifestation of a defect at the abstracted function level is called a **fault**.

In an IC, the difference between a defect and a fault can be explained as imperfections in the hardware and function, respectively. While in genomics, examples of biological defects can include mutations in the gene activation site, malformation of the protein folding, and problems in the gene product transport. Likewise, an example of a biological fault is a modification of the logical function representing a gene, producing the incorrect output.

Definition 2: A **stuck-at fault** is modeled by assigning a fixed (0 or 1) value to a signal line (input or output of a logic gate) in the circuit.

Definition 3: An **untestable fault** is a fault which no test can detect. Untestable faults appear in two situations.

- Faults that are *redundant*, whose presence does not change the output behavior of the circuit.

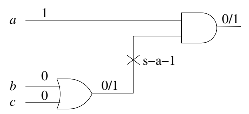

Fig. 1. Circuit with stuck-at fault

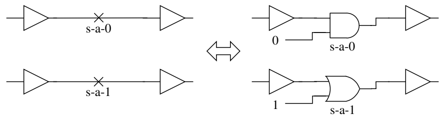

Fig. 2. Fault modeling and injection

- Faults that change the output behavior of the circuit, but no test (drug vector in the context of cancer therapy) can be generated to propagate or rectify the fault.

B. Stuck-at Fault Modeling

In the Boolean network model for a GRN, the activity of genes are modeled as a Boolean circuit. We assume the circuit is modeled as an interconnection of Boolean gates. A stuck-at fault is assumed to only affect interconnections (wires or nets) between gates. Each net can have one of two types of faults: stuck-at-1 or stuck-at-0 (s-a-1 and s-a-0, respectively). Thus, a net with a stuck-at-0 fault will always have a logic value 0, irrespective of the correct logic output of the gate (gene) driving the net.

As an example, consider the circuit of Figure 1 comprising of an OR gate driving an AND gate. Also consider a stuck-at-1 fault at the output of the OR gate, which means that the faulty line remains 1 irrespective of the input state of the OR gate. If the normal (good) output of the OR gate is 1 (in the case where its inputs were $<bc>=01,10,11$), then this fault will not affect any signal in the circuit. However, the input $<bc>=00$ to the OR gate should produce a 0 output in the good circuit. The good (faulty) value 0 (1) is applied to the AND gate. If the input vector $<abc>=100$, the good circuit output (true response) and faulty output would differ.

A stuck-at-0 fault is modeled by inserting a two-input AND gate at the fault site as shown in Figure 2. The side input of the gate is driven by a signal which is set to 1 to simulate a fault-free site, or set to 0 to inject the s-a-0 fault. Similarly, the circuit with a s-a-1 fault is modeled by inserting an OR gate at the site. The side input of this OR gate is set to 0 to simulate a fault-free site, or set to 1 to inject the s-a-1 fault. These gates are inserted at every net (wire), allowing the simulator to inject faults at any site.

Note that drugs are modeled the same as stuck-at faults, wherein a drug that inhibits a gene is modeled as a s-a-0 ”fault”, while a drug that activates a gene is modeled as s-a-1 ”fault”. The gates for drug injection are inserted at the nets of the genes that they target.

C. Boolean Satisfiability

Several ATPG algorithms [5], [6], [7], including the method proposed in this paper are based on Boolean Satisfiability (SAT) and utilize the stuck-at fault model. We begin with an overview of SAT, followed by a SAT-based formulation of the ATPG problem.

Definition 4: A **literal** or a **literal function** is a binary variable x or its negation $\overline{x}$.

Definition 5: A **clause** is a disjunction (logical OR) containing literals (example: $(x+\overline{y}+\overline{z})$).

Definition 6: A **Conjunctive Normal Form (CNF)** expression consists of a conjunction (AND) of m clauses $c_1 \dots c_m$. Each clause c_i consists of disjunction (OR) of k_i literals. A CNF S is said to be **satisfied** if it evaluates to 1. Satisfying S is equivalent to satisfying all $c_i \in S$

Definition 7: **Boolean satisfiability (SAT)**. Given a Boolean formula S (on a set of binary variables X) in CNF, the objective of SAT is to identify an assignment of the binary variables in X that satisfies S, if such an assignment exists.

For example, consider the formula $S(a,b,c) = (a+\overline{b}) \cdot (a+b+c)$. This formula consists of 3 variables, 2 clauses, and 4 literals. This particular formula is satisfiable, and a satisfying assignment is $(a,b,c) = (0,0,1)$ or $\overline{a}\overline{b}c$.

Definition 8: There may exist many satisfying assignments for the formula in question. An extension of the SAT problem is **All-SAT** which finds *all* satisfying assignments.

Definition 9: **Weighted partial Max-SAT** (WPMS). Another extension of SAT aims to satisfy a partial set of clauses. Each clause in the CNF is identified as a hard clause or soft clause. Each soft clause is associated with a weight. The problem is to identify an assignment that satisfies *all* hard clauses while maximizing the total weight of the satisfied soft clauses.

D. SAT-based formulation for stuck-at fault model

In the SAT based ATPG method, we first generate a formula in CNF to represent tests for the fault. To do so, the circuit from the stuck-at fault model must be converted to a CNF. Every gate (g_i) of the circuit has CNF formula (G_i) associated with it, which represent the function performed by the gate. The formula is true iff the variables representing the gate's inputs and outputs take on values consistent with its truth table.

For example, consider a 2-input AND gate (g_j) with the lines x and y as inputs and z as output. The CNF formula (G_j) for the AND gate is written as:

$$G_j = (\overline{z}+x)\cdot(\overline{z}+y)\cdot(z+\overline{x}+\overline{y})$$

A CNF formula for the entire circuit S is obtained by forming the conjunction of the CNF formulas for all the gates of the circuit. If there are n gates in the circuit, then the CNF formula S for the entire circuit is written as:

$$S = \prod_{i=1}^{n} G_i$$

When all the s-a-0 and s-a-1 variables are set to false (0), the CNF formula S describes the good (fault-free) circuit behavior. The faulty circuit is a copy of the fault-free circuit, with faults (s-a-0 or s-a-1 variables) set true for the gates affected by faults.

E. Implementation of Fault and Drug Simulation

1) Case 1: Single Stuck-at Fault Identification: In this method, all single stuck-at faults are simulated one at a time, to determine which faults are non-redundant, and the faulty output they generate. This is useful for identify which genes are faulty, by comparing the faulty output from the stuck-at model to the measured output in expression data. The information can also be used to target genes for potential drug development, avoiding genes that are untestable.

With the primary input set to a particular value, the circuit is first simulated using SAT in the fault-free and drug-free mode, and the primary output values for the true response are saved. Next while maintaining the drug-free mode, faults are simulated one by one. This is done by modifying the circuit description for a target fault (setting the variable assigned to the fault to 1) and then simulating the network. The output values of the faulty circuit are compared with the saved true response. All faults that generate a different output from the true response are non-redundant, and are flagged for drug simulation using any of the next three cases.

2) Case 2: Fault Rectification with Fewest Drugs: In the presence of a particular fault, the problem is determining whether a selection of drugs can rectify the circuit, i.e. change the faulty output to the correct output. If this is not possible, we want to obtain the ”best” or ”closest” output to the correct output, by using drugs. To do this, we guide the WPMS solver by assigning weights to the output states. For example, in the GF network used in our experiments, the fault-free output is assigned the highest weight (80) and remaining output states are assigned decreasing weights (70, 60, 50, etc.) based on increasing hamming distance (1, 2, 3, etc.) from the fault-free output. We assume that faulty states that are far in hamming-distance have a more cancer proliferative effect.

Additionally, the selection of drugs to achieve the best output should use fewest number of drugs to minimize the side-effects on the patient. To incorporate this in the WPMS solver, each drug that is *not* selected is given a weight of 1. The GF network example has 6 drugs, thus if no drugs are selected, then the cumulative drug weight is 6. Likewise, if all drugs are selected, the drug weight is 0.

Note that the output and drug weights are assigned in such a way as to avoid the situation where a less-desirable output (with few drugs) is chosen over a higher weight output with many drugs. We assume that

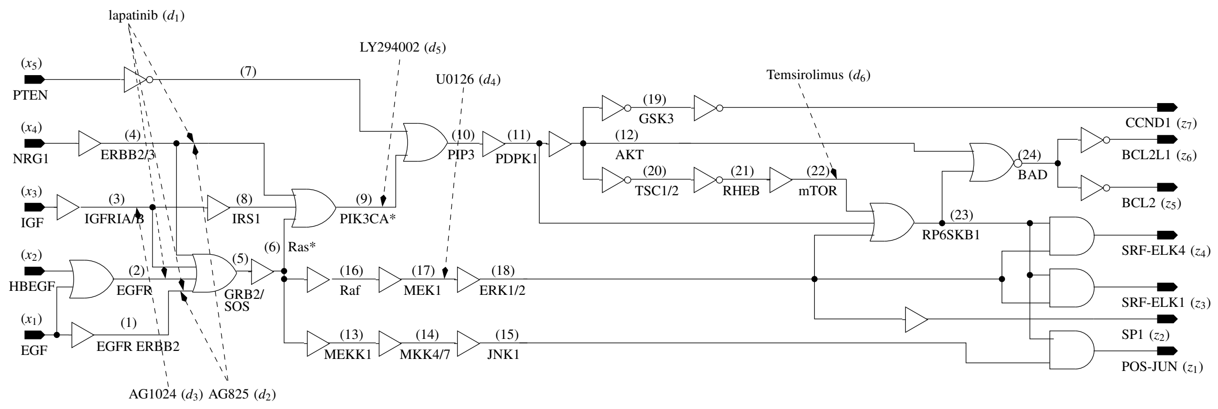

Fig. 3. Logic circuit stuck-at fault model for GF signaling pathways

from a clinical standpoint, the priority is to first determine the best possible output, and then the fewest drugs required for that output.

All faulty circuits with non-redundant faults from Case 1 are augmented with the output and drug weights and simulated using WPMS. The WPMS solver will implicitly and deterministically find the assignment of drugs that achieves the best possible output and with the fewest drugs. The output values, selected drugs, and highest weight of the fault+drug circuits are recorded and compared with the drug-free circuits. All fault+drug circuits with best solutions using zero drugs are untestable fault, wherein no drug combination can improve the output.

In general, several stuck-at faults can be simultaneously present in the circuit. A circuit with n lines can have $3^n - 1$ possible stuck line combinations. This is because each line can be in one of the three states: s-a-1, s-a-0, or fault-free. All combinations (except one which has all lines in their fault-free state) are counted as faulty. In our implementation, multiple stuck-at faults can easily be modeled for rectification, by setting one or more lines to their faulty state.

3) Case 3: Fault Rectification with Minimal Drug Cost: In the previous case, all drugs are equal in terms of their weight. However, there may be a situation where we would want to differentiate the drugs based on some cost function based on characteristics such price, number of side-effects, or ease of availability. For example, two drugs with few side-effects may be more desirable than one drug with many side-effects, if both drug selections have the same output. As such, in the presence of a particular faulty circuit and desired output, the problem is determining a selection of drugs with lowest total cost.

Each drug that is not selected is given a weight proportional to its cost. In our example, we use the number of side-effects as the drug's cost. All faulty circuits with detectable faults from Case 2 are modified with the new drug weights. In addition, the output of the circuit is fixed to the best output as determined in Case 2. These circuits are then solved using WPMS to obtain the selected drugs with lowest cost.

4) Case 4: Determining Therapy with Fewest Drugs and Best Coverage: From Case 2, we identify the drug selection that best rectifies a *certain* fault. However, in drug therapy, the fault location is likely unknown. In this situation, a drug selection that rectifies all faults (or as many faults as possible) with the fewest drugs, is desirable.

For each faulty circuit (with a single fault), we find all combinations of 1, 2, and 3 drugs that yield the best output from Case 2. This is done by performing a WPMS to find *all* satisfying drug selections with drug weight greater than or equal to $d-3$, where d is the total number of drugs. Each drug selection (or vector) is analyzed to see how many testable faults are rectified or covered by it. The drug vector with the highest coverage and fewest drugs is recorded as a best candidate for therapy.

IV. Experimental Results

A. Model Implementation

We evaluate the WPMS-based ATPG methods on the GRN that models growth factor (GF) pathways [3]. In multicellular organisms, cell growth and replication is tightly controlled by the cell cycle control. This system receive signals from other cells which are used to decide whether the cell should grow. A failure in these signals can lead to unwanted or unregulated cell growth, leading to cancer. These signaling pathways are well studied, and several drugs have been developed to target different pathways for cancer therapy.

We begin with a BN model of the GF pathways as derived in [3]. In this model, pathways are converted to an equivalent BN logic gate. Each interconnection (net) between logic gates is then numbered.

As stated in Section III, defects in the GRN are represented as stuck-at faults that permanently sets a signal net to 1 or 0. At each net, the logic gates for injecting a s-a-0 or s-a-1 are inserted. If there is a drug that targets the net, the appropriate logic gates are also inserted. The conversion of the faults and drug locations in Figure 3 to a logic netlist is discussed in Section III. The final circuit is then converted to CNF for further analysis.

In the results, stuck-at faults will be referred by the net numbers that are affected (i.e. net 7 s-a-0, means that the signal corresponding to net 7 is stuck-at 0). The network has 5 primary input (PI) signals and 7 primary output (PO) signals. The PIs will be defined as a 5-bit binary vector $X = [EGF, HBEGF, IGF, NRG1, PTEN]$, while the POs will be defined as a 7-bit binary vector $Z = [FOS-JUN, SP1, SRF-ELK1, SRF-ELK4, BCL2, BCL2L1, CCND1]$. In all tests, the PIs are fixed to $X = 00001$.

For this network, six drugs are available, defined as a 6-bit vector. Each bit corresponds to a drug, such that a value of 1 on the i^{th} bit indicates that drug i is selected, and a value of 0 indicates that drug i is not selected. The drug vector is $D = [lapatinib, AG825, AG1024, U0126, LY249002, Temsirolimus]$.

All the methods (Case 1 through 4 of Section III) methods were implemented using an open-source weighted partial Max-SAT solver called Maxsatz [8], [9]. Our procedure consists of scripts which take the initial CNF, selects desired fault variables, sets output and drug weights, and solves the CNF using Maxsatz. The satisfying assignments are then parsed for the output and drug vectors, and reported in the results. In all examples listed in this section, the WPMS runtime was significantly less than 1 second per CNF.

B. Simulation Results

1) Case 1: Single Stuck-at Fault Identification: In the single stuck-at fault model, each net was simulated for s-a-0 and s-a-1 with no drugs, and results compared with the fault-free circuit. For fault-free circuit with $X = 00001$, the output vector is $Z = 0000000$. All single stuck-at faults which have a output different from the fault-free circuit are recorded and shown in Table I. In this table, the first three columns show the affected net, the stuck-at value, and the faulty output, respectively.

From this table, we observe that nets 13, 14, and 15 are not listed. The presence of a fault (s-a-0 or s-a-1) on these nets does not generate an incorrect PO, and as such, these are redundant faults. From a therapy standpoint, these faults can be ignored.

2) Case 2: Fault Rectification with Fewest Drugs: From the results in Case 1, all non-redundant faults are simulated with drugs. The outputs are first weighted where the fault-free output $Z = 0000000$ has a maximum weight of 80 as it represents a non-proliferative output. All remaining output vectors are given weights of $80 - 10h$, where h is their hamming distance from the fault-free output. The drugs are

Net	s-a	Faulty PO	Best PO	Drug Vector	Score
1	1	1111111	0000000	010000	85
2	1	1111111	0000000	100000	85
3	1	1111111	0000000	001000	85
4	1	1111111	0000000	010000	85
5	1	1111111	0000000	000110	84
6	1	0000111	0000000	000110	84
7	1	0000111	0000111	000000	56
8	1	1111111	0000000	000010	85
9	1	0000111	0000000	000010	85
10	1	0000111	0000111	000000	56
11	1	0000111	0000111	000000	56
12	1	0000111	0000111	000000	56
16	1	0111110	0000000	000100	85
17	1	0111110	0000000	000100	85
18	1	0111110	0111110	000000	36
19	0	0000001	0000001	000000	76
20	0	0000110	0000000	000001	85
21	1	0000110	0000000	000001	85
22	1	0000110	0000000	000001	85
23	1	0000110	0000110	000000	66
24	0	0000110	0000110	000000	66

TABLE I
DRUG SELECTION FOR SINGLE STUCK-AT FAULTS

Net	s-a	Faulty PO	Best PO	Drug Vector	Score
1,21	1,1	1111111	0000000	010001	84
5,19	1,0	1111111	0000001	000110	74
13,16	1,1	1111110	0000000	000100	85
2,14,20	1,1,0	1111111	0000000	100001	84
4,7,17	1,1,1	1111111	0000111	010100	54
8,9,11	1,1,1	0000111	0000111	000000	56

TABLE II
DRUG SELECTION FOR MULTIPLE STUCK-AT FAULTS

also given weights where the non-selection of a drug has a weight of 1. With six drugs, the maximum score is therefore $80+6=86$.

Table I shows for each non-redundant stuck-at fault, the best output (Column 4), the drug vector to achieve such output (Column 5), and the score (Column 6). We observe that for many faults, there exists a drug vector that can completely rectify the fault, and produce a fault-free circuit. Additionally, the corresponding reported drug vector is minimal in the number of drugs used, which is desirable in therapy usage. We also determine that faults on nets 7, 10-15, 18, 19, 23, and 24 are untestable, as no combination of drugs can produce a change in the output. This can be explained as there are no drugs on the fan-out of these nets to rectify the fault.

To demonstrate the adaptability of our algorithm, we test it on a few examples of multiple stuck-at faults. Table II shows for a circuits with multiple stuck-at faults, the best drug selection for fault rectification (when possible). The columns of Table II have the same meaning as in Table I.

3) Case 3: Fault Rectification with Minimal Drug Cost: When selecting drugs, there may be multiple drug combinations that may rectify a fault, but where each drug has a different associated cost. We first assign weights to drugs, according to their cost. For this paper, we use the number of side-effects as the drug's cost. Drugs AG825, lapatinib, Temsirolimus are assigned weights of 10, 15, and 35, respectively, which correspond to their approximate number of side-effects [10], [11]. However, drugs AG1024, U0126, and LY294002 have yet to under go clinical trial and the number of side-effects are unknown. As such, these drugs are assigned a weight 20, which is an average of the 3 previous weights.

In this GF example, Case 3 simulation provides the same results as in Case 2. This is due to a lack of drugs that share paths in the circuit. In fact, for almost every non-redundant fault, the best output state can only be achieved through one drug vector.

4) Case 4: Determining Therapy with Fewest Drugs and Best Coverage: Using the results from Case 2, we observe that the GF network has 13 testable faults. For these 13 faults, we perform an All-SAT to find the top three scoring drug combinations yielding the best output. All drug combinations are analyzed across all single faults and presented in Table III showing drug vector, count of faults rectified, and fault coverage. Drug vectors are ordered in increasing

Drug Vector	Count	Coverage	Drug Vector	Count	Coverage
000001	**3**	**23%**	**000111**	**13**	**100%**
000010	2	15%	001011	6	46%
000100	2	15%	001101	6	46%
001000	1	8%	001110	10	77%
010000	2	15%	010011	7	54%
100000	**3**	**23%**	010101	7	54%
000011	5	38%	010110	10	77%
000101	3	23%	011001	6	46%
000110	**10**	**77%**	011010	5	38%
001001	4	31%	011100	5	38%
001010	3	23%	100011	8	62%
001100	3	23%	100101	8	62%
010001	5	38%	100110	10	77%
010010	4	31%	101001	7	54%
010100	4	31%	101010	6	46%
011000	3	23%	101100	6	46%
100001	6	46%	110001	6	46%
100010	5	38%	110010	5	38%
100100	5	38%	110100	5	38%
101000	4	31%	111000	4	31%
110000	3	23%			

TABLE III
DRUG SELECTION COUNT AND FAULT COVERAGE

order of number of drugs selected.

From these results, we observe that with only 1 drug, lapatinib (d_1) or Temsirolimus (d_6), the best coverage is only 23% of faults. When allowing for 2 drugs, coverage increases to 77% using the drug combination of U0126 (d_4) and LY294002 (d_5). Finally, we achieve 100% coverage of all testable faults when using the 3 drug combination of U0126 (d_4), LY294002 (d_5), and Temsirolimus (d_6). When the single stuck-at fault location is unknown, these selected drug combinations will be most effective for therapy and preventing the proliferation of cancer.

V. CONCLUSIONS

In this paper, we have presented an efficient and extensible SAT-based ATPG methodology towards the design of cancer therapy. We approach this problem by representing the BN and cancer as a logic circuit stuck-at fault model. This circuit, along with the testing conditions, are converted into a CNF. The CNF is then augmented with output and drug vectors weights and solved using a weighted partial Max-SAT solver for four different usage cases: (1) single stuck-at fault identification, (2) fault rectification with fewest drugs, (3) fault rectification with minimum drug cost, and (4) determining therapy with fewest drugs and best coverage. We demonstrate these methods on the growth factor signaling pathway, and have presented results that are applicable to cancer therapy. While the GF network example in this paper is a combinational network, our algorithm can easily be extended to address seqential networks, like those found in transcriptional GRNs, by simply unfolding the sequential circuit in time and applying the same methods. Furthermore, all nets, inputs, outputs, and drugs can be assigned weights which can be made variable, allowing the user to fine-tune the network or design therapies for any number of test situations.

REFERENCES

[1] N. Guelzim et al., "Topological and causal structure of the yeast transcriptional regulatory network," *Nature Genetics*, vol. 31, pp. 60–63, 2002.
[2] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437 – 467, 1969.
[3] R. Layek, A. Datta, M. Bittner, and E. R. Dougherty, "Cancer therapy design based on pathway logic," *Bioinformatics*, vol. 27, no. 4, pp. 548–555, 2011.
[4] F. Jacob and J. Monod, "Genetic regulatory mechanisms in the synthesis of proteins," *Journal of Molecular Biology*, vol. 3, no. 3, pp. 318–356, 1961.
[5] T. Larrabee, "Efficient Generation of Test Patterns Using Boolean Difference," in *Proc. of the Intl. Test Conf.*, pp. 795–801, 1989.
[6] P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, pp. 1167–1176, sep 1996.
[7] N. Saluja and S. Khatri, "Efficient sat-based combinational atpg using multi-level don't-cares," *Test Conference, International*, vol. 0, pp. 10 pp.–1038, 2005.
[8] "Maxsatz." http://home.mis.u-picardie.fr/˜cli/EnglishPage.html.
[9] C. Li, F. Manya, N. Mohamedou, and J. Planes, "Exploiting cycle structures in max-sat," in *Theory and Applications of Satisfiability Testing - SAT 2009* (O. Kullmann, ed.), vol. 5584 of *Lecture Notes in Computer Science*, pp. 467–480, Springer Berlin / Heidelberg, 2009.
[10] "Santa Cruz Biotechnology, Inc." http://www.scbt.com/.
[11] "PubMed Health Home." http://www.ncbi.nim.nih.gov/pubmedhealth/.