# 회로에서 생성된 CNF 에서 회로 정보 추출

남명진*, 성창훈**, 최진영*
*고려대학교 컴퓨터학과
**육군사관학교 전산학과
e-mail : mjnam@formal.korea.ac.kr

# Extracting Structural Information from CNF

Myoung-Jin Nam*, Chang-Hun Sung**, Jin-Young Choi*
*Dept. of Computer Science and Engineering, Korea University
**Dept. of Computer Science, Korea Military Academy

**요 약**

Boolean Satisfiability (SAT)는 전산학의 중요한 문제로서 SAT problem 을 풀기 위한 많은 알고리즘과 도구들이 제안되어 왔다. 특히, 지난 몇 년 간 많은 발전을 이루어왔고, 하드웨어 검증과 모델 체킹 등의 분야에서 많이 적용되어 왔다. 여러 문제들을 Conjunctive Normal Form (CNF)로 표현하여 CNF 의 특성을 이용하여 SAT 알고리즘이 발전되어 왔다. 그런데, 회로를 CNF 로 표현할 때 몇 가지 문제점이 발생하는데 특히 CNF 는 회로의 structural information 을 잃어버린다는 것이 큰 문제점이다. 이를 보완하기 위하여 회로의 structural information 을 이용하기 위한 많은 연구가 진행되어 왔다. 이러한 대부분의 연구는 회로의 정보를 가지고 있다는 경우에 한정된다. 그러나, 하드웨어 검증에서 회로의 정보 없이 검증해야 하는 경우들도 발생한다. 이 논문은 회로의 정보를 가지고 있지 않을 때 CNF 만으로 회로의 structural information 을 추론하는 방법을 제시한다.

## 1. 서론

SAT(Boolean Satisfiability) 문제는 하나의 이진 표현식 (Boolean Formula)이 TRUE 값을 가지게 하는 Truth assignment 가 존재하는지를 검사하는 문제이다. 만약 이진 표현식이 참이 되는 경우가 존재하면 그 표현식은 “satisfiable”하다고 하고, 줄여서 SAT 이라고 하기도 한다. 그러한 경우가 존재하지 않으면 “unsatisfiable”하다고 하고, 줄여서 UNSAT 이라고 한다. 잘 알려진 바와 같이 SAT 문제는 NP-complete [1] 문제로 전산학에서 중요한 문제이다.

지난 10 년간 이 문제에 대한 활발한 연구[2][3]가 진행되어 이 문제를 효율적으로 풀기 위한 많은 기법들이 제안되어 왔다. 또한, 현재 SAT 를 자동으로 검사하는 SAT solver 가 많이 개발되어 사용되고 있다. zchaff[4], Berkmin[5], GRASP[6] 등이 그 예이다. 이 solver 들은 CNF 를 입력으로 받아서, SAT/UNSAT 여부를 출력해주고, solver 에 따라서 satisfying assignment 를 출력하기도 한다. 어떠한 전자 회로가 언제나 주어진 속성을 만족하는지를 알아보는 하드웨어 검증 [7]등에 사용되며, 또한 모델 체킹 [8] 등에도 사용된다. 이와 같은 방식으로 하드웨어 나 소프트웨어 시스템의 정확성 (correctness)을 확인하여 보다 안정된 시스템을 구현하는데 도움이 된다.

SAT 를 풀기 위한 알고리즘들은 대부분은 CNF (conjunctive normal form)를 입력 형식으로 받는다. 특별한 형식을 지니지 않은 일반 표현식보다 형식을 가진 입력 형태가 규칙을 적용하고 SAT 를 검사하기에 더 효율적이다. CNF 를 이용하면 더 효율적인 SAT 검사가 가능하지만, 단점이 있다. SAT 는 특히 하드웨어 검증에 많이 사용되는데 회로를 CNF 로 변환하였을 때 회로의 structural information 을 잃어버린다. 회로를 CNF 로 변환하면 회로의 structural information 없이 SAT 의 기법만을 이용하여 검사를 하게 된다. 따라서 불필요한 search space 까지 검사해야 하기 때문에, 더 효율적인 SAT 검사를 위해서 많은 연구[9][10]가 진행되고 있다.

특히, 이 논문은 기존 연구들에서 더 나아가 CNF 에서 회로의 정보를 추측하는 방법을 포함하고 있다. 회

로의 structural information 을 이용하는 기존 연구들의 방향은 주로 CNF 가 어떠한 boolean function 에서 변환된 것인지를 아는 경우를 위주로 진행되어 왔다. 이러한 경우는, 회로의 정보를 가진 상태에서 그 회로를 이용하여 CNF 의 SAT 검사를 하게 된다. 하지만 회로의 정보를 가지고 있지 않은 상태에서, 회로에서 변환된 CNF 만 가지고 검증을 해야 하는 상황도 발생한다. 이러한 경우에는 기존의 연구들을 적용시킬 수 없다. 따라서, 더 효율적인 SAT 검사를 하기 위해서는 CNF 에서 회로의 structural information 을 추출하는 방법이 필요하다. 이 논문은 회로에서 변환된 CNF 만으로 실제 회로의 structural information 을 추출하는 방법을 제시한다.

## 2. Boolean Satisfiability

Satisfiability problem(SAT)는 어떠한 표현식이 있을 때, 그 표현식을 " true" 가 되게 하는 값의 조합이 존재하면 그 표현식은 " satisfiable" 이라고 얘기하고, 없으면 " unsatisfiable" 이라고 얘기한다. 그리고 " true" 가 되도록 하는 변수값의 조합은 모델(*model*) 또는 satisfying assignment 라고 한다. 그런데, 특별한 형식이 없는 일반 표현식은 SAT 임을 검사하기 힘들다. 그래서 일반 표현식은 특별한 형태로 바꾸어서 SAT 임을 증명하는데 대부분 CNF 를 많이 사용한다.

X = A1 & A2 & A3 & A4 …

A1 = (a1 + a2 + a3 + …)

A2 = (b2 + b2 + b3 + …) …

위와 같은 CNF 가 있을 때 A1, A2,...,An 을 *clause* 라고 하고, clause 내의 a1, a2, b1,b2...를 *literal* 이라고 한다. 여기서 literal 은 positive/negative proposition 모두를 가리킨다.

CNF 는 이처럼 clause 들의 set 으로 이루어지는데, 그 중에서 하나의 literal 로만 이루어진 clause 를 unit clause 라고 한다. 예를 들어, (~a)&(~b + c) 라는 CN F 가 있을 때 (~a)가 unit clause 이다. CNF 에서 이 unit clause 는 중요한 역할을 하는데, unit clause 가 있을 때 decision(변수에 1 또는 0 을 임의로 할당하는 것)을 하지 않고 unit clause 에 true 를 할당한다. 각 clause 는 and-operator 로 연결되어 있어서 하나의 clause 라도 false 가 된다면 전체가 false 가 되기 때문이다.

decision 을 하거나, unit propagation 을 했을 때 literal 에 값이 할당되는데, 값이 할당된 literal 에 대해서는 boolean constraints propagation(이하 BCP)를 적용한다. 만약 literal 에 true 가 할당되면 CNF 에서 그 literal 이 속한 clause 전체를 지운다. 이유는 각 clause 내에서 literal 은 or-operator 로 연결되어 있기 때문에 한 literal 이라도 true 가 되면, 전체 clause 가 true 가 된다. 그런데 clause 들은 and-operator 로 연결되어 있으므로, 전체 CNF 의 true/false 여부는 다른 clause 들의 값에 의존하기 때문이다. 반대로, literal 에 false 가 할당되면 그 clause 에서 그 literal 만 지운다.

## 3. 회로에서 CNF 변환 방법

하나의 회로가 주어졌을 때 회로의 Primary Output (PO)을 F 라 하면 CNF 로 변환하는 방법은 다음과 같다.

$$\bigwedge_{\substack{\psi \in Sub(\Phi) \\ \psi \equiv \psi_1 \oplus \psi_2}} (p_\psi \leftrightarrow (p_{\psi_1} \oplus p_{\psi_2})) \wedge \bigwedge_{\substack{\psi \in Sub(\Phi) \\ \psi \equiv \neg\psi_1}} (p_\psi \leftrightarrow \neg p_{\psi_1}).$$

**그림 1 CNF 변환**

여기서 Sub(F) 는 F 의 subformula 들의 set 을 가리키고, ' ∘' 은 binary operator 를 가리킨다. 이 변환 방법은 Tseitin[11]에 의해서 제안된 방법으로, unary operator 나 binary operator 로 연결된 sub-formula 를 다른 변수로 substitution 하는 방법이다. unary 또는 binary operator 를 새로운 변수로 치환한 식을 CNF 로 바꾸면 clause 내 최대 3 개의 literal 을 가지는 clause 들만 생성된다. binary 또는 unary operator 로 연결된 subformula 를 새로운 변수로 substitute 한 다음 형태를 triple 이라고 한다.

A = p ∘ q ( ∘은 binary operator)

A = ㄱ p

모든 subformula 를 triple 형태로 바꾼 뒤, triple 들을 전부 and-operator 로 연결한다. 각 triple 을 2 개 혹은 3 개의 clause 로 변환이 가능하다.

이 방식을 이용하여 변환을 하면 새로운 변수들이 생성되므로, 원래의 formula 와 이 변환 방식에 의해 생성된 formula 는 equivalent 하지 않다. 하지만 원래의 formula 가 satisfiable 하면, CNF 도 satisfiable 하다. 이 성질을 가르켜 두 표현식은 equisatisfiable 하다고 한다. 원래 한 표현식을 CNF 로 변환하는데 필요한 시간은 polynomial 한 시간이 필요하다. 하지만, 위 방법을 이용하면 최악의 경우에도 linear time 이 걸린다. 더 시간을 줄일 수 있기 때문에, 회로를 CNF 로 변환할 때 대부분 이 방법을 사용한다.

대부분의 SAT 연구들은 CNF 를 기반으로 하고 있기 때문에, 회로를 검증할 때 회로에 대한 boolean function 을 CNF 로 변환하게 된다. 이때 CNF 로 변환할 때의 큰 문제점이 있는데, 회로의 structural information 이 전부 소실된다는 점이다. 위의 변환 방법을 이용하여 생성된 CNF 로 마찬가지이다. SAT 를 효율적으로 풀기 위한 많은 연구가 이루어져 많은 instance 들이 빠른 시간에 풀리고 있다. 하지만, structural information 을 이용하지 않을 경우에는 불필요한 search space 까지도 검사해야 하는 경우가 발생한다. 이러한 불필요한 작업을 줄이기 위하여, 몇 가지 연구가 진행되고 있다. 이들 대부분의 연구들은 회로에서 생성된 CNF 의 실제 회로의 정보를 가지고 있을 때에 한정된다. 하지만, SAT 를 이용하여 하드웨어 검증을 할 때, 회로 정보를 가지지 않은 상태에서 회로에서 생성된 CNF 의 SAT 검사를 해야 하는 경우도 발생한다. 이러한 경우에는 structural information 에 초

점을 맞추지 않고, 기존 SAT 기법들만을 이용하여 SAT 검사를 해야 한다. 하지만, CNF 에서 원래의 회로를 완벽하게 복원할 수 없어도 어느 정도 추측할 수 있다면 더 효율적인 SAT 검사가 가능하다.

## 4. CNF 에서 Structural Information 추출 방법

이 논문에서는 CNF 의 clause 의 개수를 이용하여 structural information 을 추출하는 방식을 제안한다. Structural information 중에서, Primary input (PI)과 PI 에 가까운 variable 에 초점을 맞춘다. 회로의 관련된 대부분의 SAT 문제의 PI 의 값에 의존적이기 때문이다.

### Primary Input 추출 방법

회로에서 CNF 를 생성할 때 앞에서 언급한 Tseitin translation 을 이용한다. 이 translation 방식은 하나의 gate 를 새로운 변수로 치환한다. 하나의 gate 에 해당하는 boolean function 을 하나의 변수로 치환하여 이를 equivalent operator 로 나타낸 식을 triple 이라고 하자. 하나의 triple 에서 여러 개의 clause 가 생성된다. 이때 각 clause 내의 속한 literal 의 개수가 n 가 일 때 이 clause 를 n-size 의 clause 라고 하자. 입력이 I 개인 gate 를 cnf 로 변환하면 I+1-size 의 clause 하나와 I 개의 2-size 의 clause 들이 생성된다. 이 pattern 은 and, or, nand, nor gate 에 대해 적용된다. 이 때 gate 의 output 은 생성된 모든 clause 에서 나타나고, gate 의 input 들은 I+1-size 의 clause 에 한번, 그리고 2-size 의 clause 에 한번 나타난다. 회로에서 CNF 생성시 위의 방법이 많이 사용되므로, CNF 는 이러한 형태를 띄고 있다고 가정한다.

PI 는 gate 의 output 이 아니므로 한 gate 의 반드시 긴 clause 와 2-size 의 clause 에 한번씩 나타난다. 이 논문에서는 이점에 착안하여 PI 의 guess 한다. CNF 전체를 대상으로 size 3 이상의 clause 에 나타나는 횟수와 size 2 의 clause 에 나타나는 횟수가 같은 variable 을 찾아서 PI 라고 가정한다. 단 xor, xnor, buffer, inverter 는 이 pattern 에 해당되지 않는다. 한 PI 가 나열한 gate 의 입력으로 들어가는 경우에 못 찾아질 수도 있고, 실제 intermediate variable 이 이들 gate 의 입력으로 들어가면서 PI 로 찾아 질 수도 있다. 하지만 이 논문에서는 이런 경우는 많지 않다고 가정한다. 실험결과 실제로 이러한 경우는 일부에 불과하다. 만약 PI 가 xnor, xor gate 의 입력으로 들어가지 않는 경우에는 모든 실제 PI 가 찾아진 PI candidate 의 set 에 포함된다.

예를 들어, [표 1]의 경우 간단한 회로를 verilog 로 표현한 것이다. [표 1]의 회로를 DIMACS CNF file 로 변환하면, [표 2]와 같다. [표 1]의 회로에서 PI 에 해당되는, CNF 내의 실제 변수는 {1,2,3,4} 이다. 앞에서 제시된 방법을 이용하여 찾아진 PI 의 candidate 들은 {1,2,3,4}이다. [표 1]의 회로의 경우에는 PI 가 오직 XNOR, XOR gate 의 input 으로만 들어가기 때문에 실제의 PI 와 이 논문에서 제시된 방법을 이용하여 찾아진 PI 들이 정확하게 일치한다.

```
// IWLS benchmark module
module ₩lif/9symml_1 (₩1 , ₩2 , ₩3 , ₩4 , ₩5,
₩7, ₩8 );

input  ₩1, ₩2, ₩3, ₩7;
output ₩4 ;
wire  ₩5, ₩8;
          assign ₩5 = (₩1 | ₩2) & ₩7;
          assign ₩8 = ~₩5 & ₩2;
          assign ₩4 = (₩3 & ₩8) | ₩7;
end;
```

**[표 1] Verilog**

```
p  cnf  15  35
-9  1  2  0
9  -1  0
9  -2  0
7  -9  -4  0
-7  9  0
-7  4  0
8  7  -2  0
-8  -7  0
-8  2  0
10  -3  -8  0
-10  3  0
-10  8  0
-5  10  4  0
5  -10  0
5  -4  0
-13  1  2  0
13  -1  0
13  -2  0
11  -13  -4  0
-11  13  0
-11  4  0
12  11  -2  0
-12  -11  0
-12  2  0
14  -3  -12  0
-14  3  0
-14  12  0
-6  14  4  0
6  -14  0
6  -4  0
-15  -5  6  0
-15  5  -6  0
15  -5  -6  0
15  5  6  0
-15  0
```

**[표 2] DIMACS**

### 그 외의 Structural Information 추출 방법

앞에서 설명된 방법을 이용하여 선택된 PI 를 바탕으로 다른 structural information 까지 추출이 가능하다. 회로에서 생성된 CNF 가 주어졌을 때, PI 에 해당하는 variable 들을 알면 완벽한 회로의 복원이 가능하다. 하지만, PI 를 모르는 경우에는 회로를 정확히 복원하기 위해서 매우 많은 경우들이 존재하기 때문에 완벽한 복원은 불가능에 가깝다. 하지만, 앞의 방법을 이용하면 정확한 PI 의 집합에 가까운 PI 들을 찾아낼 수 있기 때문에, 이 방법을 이용하여 다른 structural information 을 찾는 방법을 제시한다.

PI 에서부터 path 길이가 n 인 variable 을 n-depth variable 이라고 하자. n-level variable 을 구하는 방법은 위의 PI 찾는 방법을 이용한다. N-level variable 을 구하기 위하여 우선 PI 의 candidate 를 찾는다. 각 PI 에 대하여, PI 가 속한 2-size clause 를 찾는다. Tseitin translation 을 이용하여 생성된 CNF 에서는 size 가 2 인 clause 에서 한 literal 은 gate 의 output 이고, 나머지 literal 은 gate 의 input 이다. 따

라서 찾아진 2-size clause 에서 PI candidate 가 아닌 variable 이 PI 에서 1 depth 만큼 들어간 variable 이 된다. 다른 n-depth variable 도 마찬가지이다. 물론 PI 추측 방법과 마찬가지로 xnor, xor, buffer 그리고 inverter 에 대해서 한계를 가지고 있다. 이 논문에서는 이 4 경우에 대한 gate 에 대해서는 무시하고, structural information 을 추출한다.

## 5. 결론 및 향후 연구

이 논문에서는 회로의 정보가 없는 상태에서 CNF 로부터 회로의 structural information 을 추출하는 방법을 제시하였다. SAT 를 이용한 하드웨어 검증 문제에서 회로의 structural information 이용 문제를 중요하게 다뤄져 왔다. 이러한 연구들은 회로의 정보를 가지고 있다는 가정 하에 이루어져 왔다. 이 논문은 지금까지의 연구와는 달리 회로의 정보가 없는 상태에서 CNF 만으로 회로의 structural information 을 추측하는 방법을 제시한다. 이 structural information 방법은 매우 간단하기 때문에 구현이 간단하고, structural information 을 찾는 시간이 매우 짧다. 회로의 정보없이 CNF 를 가지고 하드웨어 검증을 할 필요가 있을 때, 이 방법은 기존의 SAT 알고리즘을 바꿀 필요없이 structural information 을 이용하는 다른 많은 기법을 적용하기 전에 전처리로 쓸 수 있다.

## 6. 참조 문헌

[1] S.A. Cook, The Complexity of Theorem-Proving Procedures, The Third ACM Symposium on Theory of Computing, pp. 151-158, 1971.

[2] M.Davis, H.Putnam, A Computing Procedures for Quantification Theory, J. of ACM, 7:201-215, 1960

[3] Mary Sheeran, and Gunnar Stålmarck. A Tutorial on Stålmarcks's Proof Procedures for Propositoinal Logic. In *International Conference on Formal Methods in Computer-Aided Design,* Springer LNCS, pages 82-100, 1998.

[4] M.W.Moskewicz, C.F. Madigan, Y.Zhao, L.Zhang, and S.Malik, Chaff:Engineering an Efficient SAT Solver,38th Design Automation Conference(DAC), pp.530-535, 2001.

[5] E.Goldgerg and Y.Nivikov, BerkMin : A Fast and Robust SAT solver, Design Automation and Test in Europe(DATE), pp.142-149,2000.

[6]J.P. Marques-Silva, K.A. Sakallah, GRASP:A Search Algorithm for prepositional Satisfiability, IEEE Transaction on computers, vol.48,pp.506-521,1999.

[7] E. Goldberg, M. Prasad and R. Brayton. Using SAT for Combinational Equivalence Checking, *International Workshop on Logic Synthesis*, pages 185-191, 2000.

[8] K.L. McMillan, Interpolation and SAT-based Model Checking, Computer Aided Verification, 2003.

[9] J.Marques-Silva and L.G. e Silva, Algorithms for Satisfiability in Combinational Circuits Based on Backtrack Search and Recursive Learning. In Workshop notes of The International Workshop on Logic Synthesis, pages 227-241, June, 1999.

[10] S. Safarpour, A. Veneris, R. Drechsler, and J. Lee, Managing Don’t Cares in Boolean Satisfiability, Design, Automation, and Test in Europe(DATE), 2004.

[11] G.S. Tseitin. On the complexity of derivation in propositional calculus. Studies in Constructive Mathematics and Mathematical Logic part 2, pp. 115-125, 1968. Reprinted in J. Siekmann and G. Wrightson (editors), Automation of reasoning vol. 2, pp. 466-483. Springer-Verlag Berlin, 1983.