DOI: 10.3745/KIPSTD.2008.15-D.6.777

만족가능성 처리기를 이용한 이진 변수 서브시퀀스 추출

박 사 천[†] · 권 기 현[††]

요 약

최근 정형 검증 분야에서 상태 폭발 문제를 극복하기 위해 만족가능성(Satisfiability) 처리기를 사용하는 방법이 많이 연구되고 있다. 만족가능성 처리기를 사용하려면 대상을 CNF 식으로 변환해야 하는데, 이진 기수 제약 조건은 시스템을 CNF 식으로 변환하기 위해 많이 사용되는 기법이다. 그러나 이진 기수 제약 조건은 이진 변수들의 집합을 다루기 때문에 이진 변수들의 순서 정보는 변환할 수 없었다. 본 논문에서는 이진 변수의 시퀀스에서 길이가 k인 서브시퀀스 추출 문제에 대한 CNF 변환 방법을 제안한다. 또한 실험을 통해 제안된 방법이 순서정보를 고려치 않고 적용한 변환 방법보다 훨씬 더 좋은 결과를 얻을 수 있었다.

키워드 : 정형 검증, 만족성, 이진 기수 제약 조건, 서브시퀀스 추출, 퍼즐 풀이

Extracting Subsequence of Boolean Variables using SAT-solver

Sachoun Park[†] · Gihwon Kwon[††]

ABSTRACT

Recently in the field of model checking, to overcome the state explosion problem, the method of using a SAT-solver is mainly researched. To use a SAT-solver, the system to be verified is translated into CNF and the Boolean cardinality constraint is widely used in translating the system into CNF. In BCC it is dealt with set of boolean variables, but there is no translating method of the sequence among Boolean variables. In this paper, we propose methods for translating the problem, which is extracting a subsequence with length k from a sequence of Boolean variables, into CNF formulas. Through experimental results, we show that our method is more efficient than using only BCC.

Keywords : Formal Verification, Satisfiability, Boolean Cardinality Constraints, Extracting Subsequence, Solving Puzzle

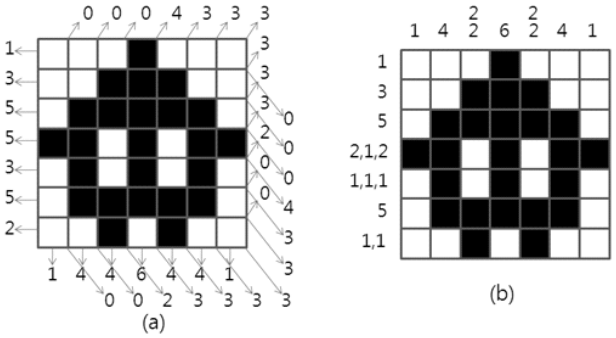
1. 서 론

유비쿼터스 환경 및 분산 환경에서 개발되는 소프트웨어 및 하드웨어의 오류는 그 파급효과가 막대하기 때문에 철저한 검증을 필요로 한다. 시스템을 철저히 검증하는 방법 중 가장 많이 사용하는 것이 모델검증[1]이다. 그런데 최근 모델 검증 분야에서 상태폭발 문제를 해결하기 위해서 명제 논리 만족가능성 처리기를 사용하는 연구가 활발하다. 논리식의 만족가능성을 결정하는 알고리즘은 60년대에 소개되었으나[2] 만족가능성 처리기의 성능은 최근 10년 동안에 급속히 발전했다[3, 4, 5]. 따라서 인공 지능 분야의 자동 계획 수립[6]이나 하드웨어 및 소프트웨어 검증[7, 8], 그리고 퍼즐 풀이[9] 등의 여러 분야에서 문제들을 해결하기 위해서 만족가능성 처리기를 많이 사용한다. 그런데 문제를 만족가능성 처리기로 다루기 위해서는 해당 문제를 CNF(Conjunctive Normal From) 형태의 명제 논리식으로 변환해야 한다. CNF는 절(Clause)들의 논리곱으로 이루어진 식이고 절은 리터럴들의 논리합으로 이루어진 식이다. 리터럴은 원소 명제 혹은 원소 명제의 부정을 의미한다. 다음은 CNF 식 ϕ에 대한 구문을 BNF로 나타낸 것이다. 아래에서 C는 절을 나타내고 L은 리터럴, p는 원소 명제를 대표한다.

$$\phi ::= C \mid C \wedge \phi$$
$$C ::= L \mid L \vee C$$
$$L ::= p \mid \neg p$$

어떤 문제가 주어졌을 때 이 문제를 한번에 CNF 식으로 변환하는 것은 매우 어렵다. 따라서 하나의 문제는 여러 개의 더 작은 문제로 나누어서 변환한다. 그 중 빈번하게 사용되는 문제 패턴이 "n개에서 인접된 $k \leq n$개 선택"하는 이진 기수 제약 조건이다[10, 11]. [8]에서는 모델 검증을 만족가능성 문제로 간주하고, 루프에 대한 CNF 변환에 이진 기수 제

※ 본 연구는 경기도의 경기도지역협력연구센터사업의 일환으로 수행하였음. [2008-111-11, 웹 기반 소프트웨어 모델 체킹 도구 개발]
† 준 회 원 : 경기대학교 전산학과 박사과정
†† 종신회원 : 경기대학교 정보과학부 교수
논문접수 : 2008년 1월 7일
수 정 일 : 1차 2008년 8월 29일
심사완료 : 2008년 10월 28일



(그림 1) 이미지 복원 문제 : (a) 네 방향 이산 단층 촬영 (b) 일본 퍼즐

약 조건을 사용했다. 그리고 [9]에서는 수도쿠 퍼즐 풀이를 만족가능성 문제로 간주하고 문제를 CNF로 변환하는데 이진 기수 제약조건을 사용했다. 또한 [10]에서는 이산 단층촬영의 이미지 복원 문제를 만족가능성 문제로 간주했다.

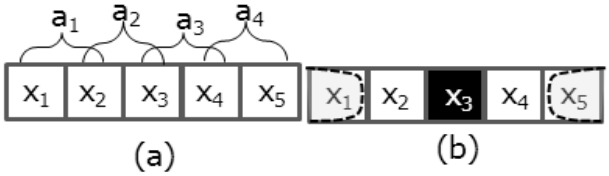
(그림 1)의 (a)는 가로, 세로, 그리고 양쪽 대각선의 4방향에서 X-선을 투영해서 얻은 정보를 바탕으로 이미지를 복원하는 예이다. [10]에서는 이 문제를 CNF로 변환하기 위해서 각 블록을 하나의 이진 변수로 표현된다. 그런 후 4 방향에 대해서 이진 기수 제약 조건을 적용하면 문제에 대한 CNF 식을 얻을 수 있다. 그런데 (그림 1)의 (b)와 같이 일본 퍼즐[13, 14]형태의 이미지 복원 문제가 주어진다면, 연속되는 블록에 대한 정보를 표현해야 하기 때문에 변수들 간의 순서 정보도 CNF 식으로 표현되어야 한다. 즉, 주어진 이진 변수의 시퀀스에서 서브시퀀스를 추출하는 문제가 된다. 시퀀스와 서브시퀀스는 전산학에서 빈번하게 사용되는 개념이지만, 이에 대한 CNF 변환은 우리가 아는 한 아직까지 연구된 적이 없었다.

본 논문에서는 서브시퀀스 제약 조건에 대한 개념과 CNF 변환 방법을 제안한다. 또한 실험을 통해 제안된 방법이 더 작은 크기의 CNF 식을 만들고, 더 빠르게 처리됨을 보인다. 특히 무작위 데이터 중 30×30의 경우 이진 제약 조건만을 사용했을 때는 처리시간이 179초가 넘었는데, 서브시퀀스를 사용했을 때는 0.5초 미만으로 처리할 수 있었다. 그리고 일본 퍼즐 문제 중 25×25에서는 79배의 속도 향상이 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 이진 변수의 서브시퀀스를 CNF로 변환하는 방법에 대해서 설명한다. 3장에서는 이전 문헌에서 소개된 효율적인 이진 기수 제약조건들[10, 11]을 서브시퀀스 추출에 활용하는 방법을 설명하고, 4장에서는 실험 결과를 보이고, 5장에서 결론 맺는다.

2. 서브시퀀스의 CNF 변환

본 논문에서 다루는 서브시퀀스 추출은 순서 있는 n개의 이진 변수 중에 서로 인접한 k개를 선택하는 것이다. 시퀀스 X에서 길이가 k인 서브시퀀스 X'을 추출하는 문제를 생각해 보자. 예를 들어 (그림 2)와 같이 시퀀스 $X=\langle x_1, x_2, x_3, x_4, x_5\rangle$가 있다고 하자. 길이가 2인 서브시퀀스 들을 나열하면 $\langle x_1, x_2\rangle$, $\langle x_2, x_3\rangle$, $\langle x_3, x_4\rangle$, $\langle x_4, x_5\rangle$ 4개의 서브시퀀스들이 존재한다. 즉 전체 시퀀스의 길이가 n이고 서브시퀀스의 길이가 k라면 $n-k+1$개의 서브시퀀스들이 존재하게 된다.



(그림 2) 서브시퀀스

이제 서브시퀀스 추출을 CNF로 변환한다면, 직관적으로 생각해 볼 때, (그림 2)의 (a)와 같이 $(x_1 \wedge x_2)$, $(x_2 \wedge x_3)$, $(x_3 \wedge x_4)$, $(x_4 \wedge x_5)$ 중 정확하게 하나만 참인 것으로 나타내면 된다. 이것을 CNF로 변환하면 아래 식 (1)과 같다. 여기서 $Exactly1(\bullet)$는 이진 변수의 집합을 받아서 해당 집합에서 정확하게 하나의 변수만 참이 되는 조건을 CNF로 변환한 식을 돌려주는 함수이다. 이때 $n-k+1$개의 보조변수와 $(n-k+1)(k+1)+\alpha$개의 절이 필요하다. 여기서 α는 보조변수 $\{a_1, ..., a_{n-k+1}\}$ 중 정확히 하나만 참이라는 조건을 표현하는데 사용된 절의 수이다.

$$\bigwedge_{i=1}^{n-k+1}\left(a_i \Leftrightarrow \bigwedge_{j=i}^{i+k-1} x_j\right) \wedge Exactly1(\{a_1, ..., a_{n-k+1}\}) \qquad (1)$$

그러나 $(x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5)$의 경우 $(x_1 \wedge x_2)$가 참 이므로 위의 식 (1)은 만족하지만, x_4가 참이므로 모두 연속한 서브시퀀스도 아니고, 서브시퀀스의 개수도 2가 아니다. 이때, 추출되는 변수의 개수는 2 이하라는 제약 조건을 추가하면 CNF 변환을 완성할 수 있다. n개의 이진 변수 중에 k개 이하가 참이라는 것의 일반적인 변환 방법은 크기가 $n-k+1$인 X의 모든 부분 집합 $X'_1, ..., X'_{\binom{n}{k-1}}$에 대해서, 아래 식으로 표현된다.

$$\bigwedge_{i=1}^{\binom{n}{k-1}_{n-k+1}} \bigvee_{j=1} x_j \in X'_i \qquad (2)$$

위의 방법이 긍정적인 접근법이라면, 그림 2의 (b)와 같이 만일 x_3이 참일 경우 x_1과 x_5는 결코 선택될 수 없다고 제한하는 부정적인 접근법을 사용할 수 있다. 이것은 $x_1 \Rightarrow \neg x_3 \wedge \neg x_4 \wedge \neg x_5$로 표현될 수 있고 다른 모든 변수에 대해서 적용해 보면 아래와 같은 식을 얻을 수 있다.

$$x_1 \Rightarrow \neg x_3 \wedge \neg x_4 \wedge \neg x_5$$
$$x_2 \Rightarrow \neg x_4 \wedge \neg x_5$$
$$x_3 \Rightarrow \neg x_1 \wedge \neg x_5$$
$$x_4 \Rightarrow \neg x_1 \wedge \neg x_2$$
$$x_5 \Rightarrow \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$

위와 같은 방법으로 식을 일반화 하면, $\frac{(n-k+1)(n-k)}{2}$

개의 절이 요구되고, 아래와 같다.

$$\bigwedge_{i=1}^{n-k}\bigwedge_{j=i+k}^{n}(\neg x_i \vee \neg x_j) \tag{3}$$

그러나 위 식의 모든 변수가 거짓이라도 식 (3)은 만족된다. 왜냐하면 거짓이 되는 변수만을 표현을 뿐 참인 변수는 표현하지 않았기 때문이다. 따라서 추출된 변수의 수는 2 이상이라는 조건을 추가해야 한다. n개의 이진 변수 중에 k개 이상이 참이라는 제약사항의 일반적인 변환 방법은 크기가 $k+1$인 X의 모든 부분 집합 $X''_1,...,X''_{\binom{n}{k+1}}$에 대해서, 아래 식으로 표현할 수 있다.

$$\bigwedge_{i=1}^{\binom{n}{k+1}}\bigvee_{j=1}^{k+1}\neg x_j \in X''_i \tag{4}$$

정리하면, 길이가 n인 시퀀스 $X=\langle x_1,\dots,x_n\rangle$에서 길이가 k인 서브시퀀스를 CNF로 인코딩하는 직관적인 방법은 (1)∧(2)와 (3)∧(4)의 두 가지가 있다.

3. 효율적인 이진기수 제약 조건을 사용한 서브 시퀀스의 CNF 변환

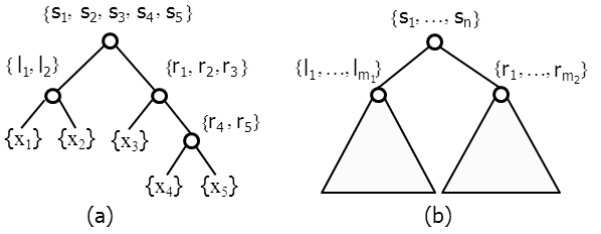
효율적인 인코딩 방법은 두 가지 기준으로 정리될 수 있다. 즉, 만족가능성 검사기의 처리시간을 단축시켜주는 인코딩 방법과 인코딩된 CNF 식의 길이가 짧은 것이라고 할 수 있다. 비록 짧은 식이 경우에 따라서 만족가능성 검사기의 속도를 저하시킬 수도 있지만, 일반적인 경우 식의 길이가 짧으면 처리 시간 또한 짧다. 이번 장에서는 이진 기수 제약 조건을 효율적으로 변환하는 두 방법을 적용해서 서브시퀀스를 추출하는 인코딩 방법에 대해서 제안한다.

3.1 이진 트리구조를 이용한 변환

이진 변수의 집합 $X=\{x_1,...,x_5\}$에서 참인 변수의 수를 단항 표현으로 나타내려면 5개의 보조 변수가 필요하다. 추가적인 보조 변수의 집합을 $S=\{s_1,...,s_5\}$라고 하자. 만일 X에서 참인 변수가 x_1와 x_4 둘뿐이라면 집합 S에 의한 단항 표현은 $s_1\wedge s_2\wedge\neg s_3\wedge\neg s_4\wedge\neg s_5$가 된다. 따라서 이진 변수 $s_1,s_2,...,s_n$로 정수 $0\le k\le n$을 나타낼 때, 단항 표현을 사용하면 $s_1\wedge...\wedge s_k\wedge\neg s_{k+1}\wedge...\wedge\neg s_n$이 된다. 이진 변수의 집합 $S=\{s_1,...,s_n\}$으로 범위 $l\le k\le m$의 단항 표현을 나타내면 아래와 같다.

$$\bigwedge_{i=1}^{l} s_i \quad\wedge\quad \bigwedge_{j=m+1}^{n}\neg s_j \tag{5}$$

[10]은 트리 구조와 단항 표현을 사용해서 길이가 n인 집합 X에서 부분 집합 X'을 선택하는 방법을 인코딩했다. (그림 3-(a))를 보면, 트리의 비 단말 노드에 보조 변수 l_1,l_2와 r_1,r_2,r_3 그리고 r_4,r_5가 각각 배정되고 루트 노드에는 S를



(그림 3) 단항 표현을 위한 이진 트리

배정하고 트리의 단말 노드에 X의 원소가 하나씩 배정된다. 따라서 모든 노드들은 각자 자신의 하위 트리에서 선택된 개수의 합을 나타내는 단항 표현이라고 할 수 있다. 따라서 앞의 예에서처럼 참인 변수가 x_1,x_4 뿐이라면, 각 중간 노드의 변수들은 한 개씩 선택됨을 나타내기 위해 각각 $l_1\wedge\neg l_2$와 $r_4\wedge\neg r_5$ 그리고 $r_1\wedge\neg r_2\wedge\neg r_3$가 되어야 한다.

결론적으로 이진 트리 구조를 이용해서 조건 $l\le|X'|\le m$을 표현하기 위해서는 (그림 3-(b))와 같이 먼저 전체 원소의 집합 $X=\{x_1,...,x_n\}$에 대한 이진 트리를 만들어야 한다. 앞의 예에서처럼 이 트리의 각 단말 노드에는 X의 원소가 하나씩 배정된다. 그리고 트리의 모든 비 단말 노드에는 해당 자식 노드들의 합을 단항 표현으로 나타내기 위한 추가적인 보조변수의 집합이 배정된다. 마지막으로 루트 노드에는 X에서 참인 변수의 합을 표현하기 위한 보조변수 $S=\{s_1,...,s_n\}$가 배정된다. 또한 트리의 균형을 맞추기 위해서 루트의 왼쪽 자식 노드에는 $\left\lfloor\frac{n}{2}\right\rfloor$ 개의 이진 변수를, 오른쪽 자식 노드에는 $n-\left\lfloor\frac{n}{2}\right\rfloor$ 개의 이진 변수를 배정한다. 따라서 (그림 3-(b))에서 $m_1=\left\lfloor\frac{n}{2}\right\rfloor$ 이고 $m_2=n-\left\lfloor\frac{n}{2}\right\rfloor$ 이며 $m_1+m_2=n$이다. 이와 같은 방식을 모든 서브 트리에 재귀적으로 적용해서 전체 트리를 완성할 수 있다.

이진 트리에서 임의의 비 단말 노드에 이진 변수 집합 $A=\{a_1,...,a_p\}$가 배정되고 그의 왼쪽과 오른쪽 자식 노드에는 각각 $L=\{l_1,...,l_{p_1}\}$와 $R=\{r_1,...,r_{p_2}\}$가 배정되었다고 하자. 이들 간의 관계는 다음과 같이 CNF로 변환된다. 아래에서 $a_0=l_0=r_0=1$ 이고 $a_{p+1}=l_{p_1+1}=r_{p_2+1}=0$ 이다.

$$\bigwedge_{i=0}^{p_1}\bigwedge_{j=0}^{p_2}(l_i\wedge r_j\Rightarrow a_{i+j})\wedge(a_{i+j}\Rightarrow l_{i+1}\vee r_{j+1}) \tag{6}$$

모든 비 단말 노드에 대해서 위와 같은 식으로 나타내면 전체 트리 구조를 표현한 CNF 식을 얻게 된다. 따라서 조건 $l\le|X'|\le m$을 변환하는 것은 (3)과 (4)의 논리곱으로 나타낼 수 있다. 이진 트리를 이용한 변환에는 $O(n\log n)$ 개의 추가 변수들과 $O(n^2)$개의 절이 필요하다. 그런데 그림 3-(a)의 트리 구조를 이용하면 (3)과 같은 의미의 식을 보다 간단하게 $(x_1\Rightarrow\neg r_1)\wedge(x_2\Rightarrow\neg r_4)\wedge(x_3\Rightarrow\neg x_5)$로 나타낼 수 있다. 왜냐하면 $\neg r_1$은 $\neg x_3\wedge\neg x_4\wedge\neg x_5$를 함축하고 $\neg r_4$는 $\neg x_4\wedge\neg x_5$를 함축하기 때문이다. 이와 같이 트리구조를 이용하는 방법을 일반화 하면 아래 식과 같다.

$$\bigwedge_{i=1}^{n-k} \bigwedge_{a \in LCA(i)} x_i \Rightarrow \neg r_1^{right(a)} \quad (7)$$

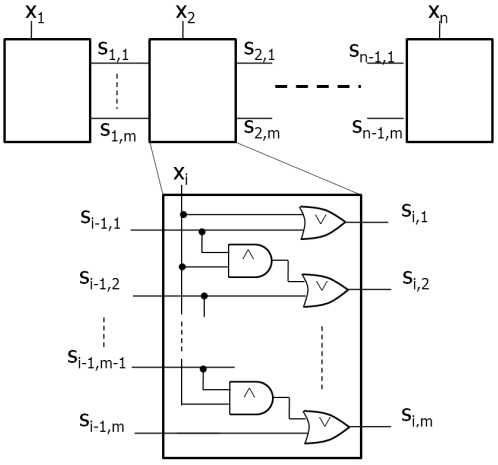
여기서 $LCA(i) = \{lca(i,j) \mid i+k \le j \le n\}$을 의미한다. 여기서 함수 $lca()$는 두 단말노드의 노드번호를 받아서 두 노드의 최소 공통 조상 노드의 노드 번호를 돌려주는 함수이다. 최소 공통 조상 노드는 두 노드의 공통 조상이면서 깊이가 가장 깊은 노드이다. 또한 $right()$는 비 단말 노드의 노드 번호를 입력으로 받아서 그 노드의 오른쪽 자식 노드의 노드 번호를 돌려주는 함수이다. 따라서 $r_1^{right(a)}$은 a의 오른쪽 자식 노드에 첫 번째 배정된 변수를 의미한다. 이진 트리를 이용한 변환은 (5)∧(6)∧(7)로 완성된다. 이러한 변환 방식은 트리를 이용하기 때문에 최악의 경우 $(n-k) \cdot \log n$개의 절이 사용된다.

3.2 순차 계수기를 이용한 변환

본 절에서는 순차 계수기를 사용하는 변환 방법을 설명한다. 순차 계수기는 (그림 4)과 같이 이진 입력 변수 $x_1, ..., x_n$을 차례로 더해서 그 합을 단항 표현으로 나타낸다. 그런데 이 서킷에서 계산할 수 있는 최대값은 m이 된다. 이것은 i 번째 부분합 $s_i = \sum_{j=1}^{i} x_j$를 $s_{i,1}, ..., s_{i,m}$의 단항 표현으로 나타내는 서킷이다. 따라서 덧셈을 위해서 $(n-1) \times m$ 개의 추가적인 이진 변수가 필요하다[11].

조건 $|X'| \le m$ 를 의미하는 (그림 4)의 순차 계수기를 CNF 논리식으로 변환하면 다음과 같다.

$$\begin{aligned}
&x_1 \Leftrightarrow s_{1,1} \quad (a)\\
&\text{for } 1 < j \le m\\
&\quad \neg s_{1,j}\\
&\text{for } 1 < i < n\\
&\quad (x_i \vee s_{i-1,1}) \Leftrightarrow s_{i,1} \quad (b)\\
&\quad \text{for } 1 < j \le m\\
&\qquad \left((x_i \wedge s_{i-1,j-1}) \vee s_{i-1,j}\right) \Leftrightarrow s_{i,j} \quad (c)\\
&\quad x_i \Rightarrow \neg s_{i-1,m}\\
&x_n \Rightarrow \neg s_{n-1,m}
\end{aligned} \quad (8)$$



(그림 4) 순차 계수기

[11]에서는 위의 서킷을 CNF로 변환하기 위해서 (a), (b), (c)의 논리식에 함축(⇒)기호를 사용했었다. 이러한 방법은 이진 변수들의 합이 최대한 m라는 것, 즉 조건 $|X'| \le m$에 대한 변환을 최적화 할 수 있다. 그러나 함축 기호 대신에 동등(⇔)기호를 사용하면, (8)∧(9)함으로써 적은 절수를 유지하면서도 조건 $l \le |X'| \le m$을 표현할 수 있다. 아래 식 (9)는 $l \le |X'|$ 부분이다.

$$s_{l,l} \vee ... \vee s_{n-1,l} \vee (x_n \wedge s_{n-1,l-1}) \quad (9)$$

식 (8)은 변수로 입력 변수들의 진위 값을 제약할 수 있다. 따라서 각각의 입력변수에 대해서 식 (3)과 같은 의미를 나타내면 아래의 식 (10)과 같다. 이식은 추가적인 변수 없이 $2(n-m)$개의 절로 표현된다. 따라서 (8)∧(9)∧(10)함으로써 시퀀스 X에서 길이가 m인 서브시퀀스를 얻을 수 있다.

$$\bigwedge_{i=1}^{n-m} x_i \Rightarrow s_{i+m-1,m} \quad \wedge \bigwedge_{i=m+1}^{n} x_i \Rightarrow \neg s_{i-m,1} \quad (10)$$

4. 실 험

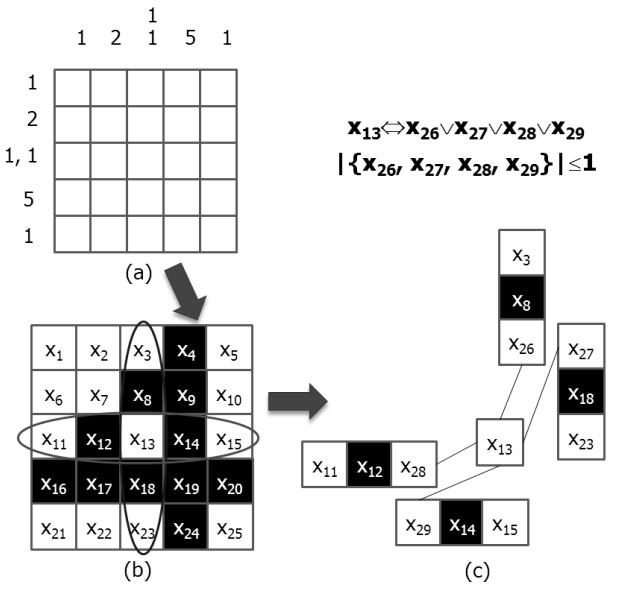
2장과 3장에서 서브시퀀스 추출을 CNF식으로 변환하는 방법을 설명했다. 이번 장에서는 각각의 변환 방법 중 어떤 것이 효율적인지 실험을 통해서 알아본다.

4.1 문제 설명

실험에서 다룰 문제는 이미지 복원 문제이다. 우리는 (그림 1)의 (a) 방법과 (b) 방법을 모두 실험하였다. (a)는 검사할 대상에 X-선을 주사하고 반대편에서 감쇠된 양을 측정해서 해당 위치에 대한 밀도를 투영(project) 하는 그림이다. 이미지를 재구축하기 위해 가로, 세로, 그리고 두 대각선 등 네 방향의 투영 결과에 대해서 이진 트리를 이용한 이진 기수조건을 CNF로 변환했다. 이것은 [10]에서 제시된 방법과 동일하다. 본 논문에서는 이 실험을 위해 [10]의 알고리즘을 재구현 하였고 이것을 실험 결과의 각 표에 4D-DT라고 명명했다.

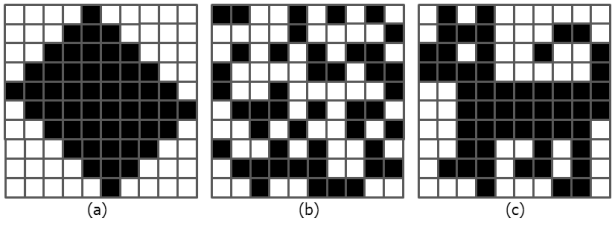
두 번째로 (그림 1)의 (b) 방법은 소위 일본 퍼즐이라고 불린다. 일본 퍼즐은 각 행과 각 열에 검은색 박스가 연속해서 나타나야 할 숫자들이 주어지고, 주어진 숫자의 개수만큼 빈 칸을 채워나가는 퍼즐이다. 단, 각 숫자에 해당하는 연속된 검은 박스들은 하나 이상의 흰색 박스에 의해서 구별되어야 한다. 여기서 하나의 박스는 검은 색이면 참이되고, 흰색이면 거짓이 되는 이진 변수로 표현된다. 그리고 연속된 박스는 이진 변수의 서브시퀀스를 의미하게 된다.

이에 대한 변환 방법을 요약하면 다음과 같다. (그림 5)의 (a)과 같이 문제가 주어지면 (b)와 같이 각 칸에는 이진 변수를 하나씩 배치한다. 그리고 각 행과 열에서 주어진 숫자의 개수만큼 서브시퀀스를 CNF 식으로 변환한다. 예를 들어, (그림 5-(a))의 두 번째 행에 주어진 숫자는 2이다. 이것



(그림 5) 일본퍼즐 CNF 변환 : (a) 문제 (b) 변수 배정 (c) 가로 세로 제약

은 (그림 5-(b))의 두 번째 행과 같이 $\langle x_6, x_7, x_8, x_9, x_{10} \rangle$의 이진 변수의 시퀀스에서 길이가 2인 서브시퀀스 $\langle x_8, x_9 \rangle$를 찾는 문제로 변환된다. 모든 행과 열에 대해서 이와 같이 변환하면 (그림 5-(a))에 주어진 숫자로 이미지를 얻을 수 있다. 그런데 세 번째 행은 두 개의 숫자 1, 1이 주어졌다. 이것은



(그림 6) 실험 데이터 유형: (a) c10 (b) r10 (c) 10×10 일본퍼즐

다시 두 개의 문제로 분해된다. 즉, 시퀀스 $\langle x_{11}, x_{12}, x_{13} \rangle$에서 길이가 1인 서브시퀀스를 찾는 것과 $\langle x_{13}, x_{14}, x_{15} \rangle$에서 길이가 1인 서브시퀀스를 찾는 문제로 나누어서 변환해야 한다. 그런데, 변수 x_{13}의 경우는 가로와 세로에서 각각 나뉘는 시퀀스에 포함되므로 4번 중첩된다. 중첩된 변수를 그대로 사용하면, 올바른 해를 구할 수 없게 된다. 따라서 (그림 5-(c))와 같이 추가적인 변수를 사용해서 중첩을 해결한다.

4.2 실험결과 및 분석

우리는 동일한 이미지에 대해서 4D-DT 방식과 (1)∧(2) 방식, (3)∧(4) 방식, 이진 트리를 이용한 (5)∧(6)∧(7) 방식, 그리고 순차계수기를 적용한 (8)∧(9)∧(10) 방식으로 이미지 복원 문제를 실험했다. 알고리즘 구현 언어는 Java 언어이고, 유분투 리눅스 상에서 듀얼 코어 1.86GHz의 CPU와 2G 메모리를 가지고 실험했다. 만족성 처리기는 Minisat 1.14를

〈표 1〉 변환된 CNF의 절 수

Problem	(1) ∧ (2)	(3) ∧ (4)	4D-DT	(5) ∧ (6) ∧ (7)	(8) ∧ (9) ∧ (10)
c10	12,688	2,932	5,476	3,500	3,576
c20	23,071,572	2,102,036	37,364	23,740	30,156
c30	Mem	Mem	116,084	73,632	103,736
c60	Mem	Mem	831,516	526,404	846,476
c120	Mem	Mem	6,162,824	3,897,648	6,840,956
c160	Mem	Mem	Mem	9,027,840	Mem
c180	Mem	Mem	Mem	12,746,724	Mem
c200	Mem	Mem	Mem	17,363,328	Mem
c220	Mem	Mem	Mem	Mem	mem
r10	2,787	2,293	5,736	3,537	2,473
r15	16,702	10,269	17,712	12,627	8,613
r20	91,154	33,968	39,508	30,065	19,688
r25	338,450	108,459	73,650	66,845	41,932
r30	703,316	292,222	122,932	120,641	72,755
r35	4,354,665	826,837	189,982	200,403	1,184,52
r45	Mem	8,019,382	388,062	480,998	2,694,96
r55	Mem	Mem	687,188	957,871	5,171,91
r65	Mem	Mem	1,107,834	1,758,808	9,151,09
10x10	2,646	2,711	5,764	4,251	3,060
18x10	6,621	2,782	12,876	5,343	6,132
15x20	1,219,921	89,136	25,796	23,399	18,485
20x20	3,245,382	410,610	39,226	37,453	26,206
25x25	Mem	12,849,614	73,464	66,835	52,076
40x40	Mem	Mem	274,484	345,895	200,919
45x50	Mem	Mem	446,824	417,924	328,336
40x79	Mem	Mem	718,988	1,030,041	575,821
50x100	Mem	Mem	1,398,164	1,752,110	1,142,766

사용했다.

실험 데이터는 (그림 6)과 같이 볼록면체(Convex) 형태의 데이터와 무작위(Random) 형태의 데이터, 그리고 인터넷에 공개된 일본 퍼즐 문제들을 가지고 실험했다. c10과 r10은 10×10의 데이터를 의미한다. 그리고 표에서 r65 이후의 데이터는 인터넷에서 구할 수 있는 일본퍼즐들이다.

<표 1>은 각 변환에 의해서 만들어진 CNF 식의 절수를 나타낸 것이다. 표에서 Mem으로 표시된 항목은 메모리 에러를 의미한다. 즉, 생성할 CNF 식이 너무 커서생성을 못한 경우이다. 그리고 Time은 만족성 처리기가 600초가 지나도록 처리하지 못한 경우이다. 절수의 결과를 살펴보면, 볼록면체 데이터는 이진 트리를 이용한 (5)∧(6)∧(7) 방식이 가장 적음을 알 수 있다. 그러나 무작위 데이터와 일본퍼즐의 경우는 순차 계수를 활용한 (8)∧(9)∧(10) 방식이 가장 적은 절수로 변환되었다.

<표 2>는 변환에 사용된 변수 수를 나타낸 것이다. 비록 (3)∧(4) 방식이 가장 적은 변수를 가지고 변환되더라도, 생성되는 절수가 이미지의 크기에 지수적으로 증가하기 때문에 큰 문제를 다룰 수 없었다. 볼록면체 데이터의 경우 순차계수기를 적용한 방식이 가장 많은 변수로 변환되었다. 순차계수기는 보조변수를 사용하는데 길이 n의 시퀀스에서 길이가 k인 서브시퀀스를 선택할 때 k의 크기에 따라서 보조변수의 숫자가 증가하기 때문이다.

<표 3>은 만족성 처리기의 처리 시간을 나타낸 것이다. 비록 4D-DT의 CNF 변환 방식이 (5)∧(6)∧(7) 방식이나 (8)∧(9)∧(10) 방식에 비해 비슷하거나 더 적은 수의 변수와 절수를 갖는 식으로 변환 되었지만 처리속도는 현저하게 늦었다. 볼록면체의 데이터의 경우는 (5)∧(6)∧(7) 방식이 가장 작은 식을 만들어 냈으며, 속도 또한 가장 좋았다. (5)∧(6)∧(7) 방식은 무작위 데이터에서도 제일 좋은 결과를 보였다.

일본 퍼즐의 경우는 이진 트리를 이용한 방식과 순차계수기를 활용한 방식이 비슷한 결과를 보였다. 그러나 25×25의 경우는 이 두 방식이 거의 70배 이상 짧은 시간에 이미지를 복원할 수 있었다. 비록 4D-DT와 일본퍼즐 형태의 투영 데이터는 서로 다른 의미의 값을 나타내지만, 실험을 통해서 시퀀스 정보를 활용해서 CNF 식을 변환하는 것이 문제 풀이에 효과적으로 활용될 수 있음을 알았다.

〈표 2〉 변환된 CNF의 변수 수

Problem	(1)∧(2)	(3)∧(4)	4D-DT	(5)∧(6)∧(7)	(8)∧(9)∧(10)
c10	2,148	272	1,228	780	1,000
c20	2,097,552	1,056	6,312	3,920	8,000
c30	Mem	Mem	15,962	9,780	27,000
c60	Mem	Mem	76,448	46,320	216,000
c120	Mem	Mem	356,192	214,080	1728,000
c160	Mem	Mem	Mem	404,480	Mem
c180	Mem	Mem	Mem	523,400	Mem
c200	Mem	Mem	Mem	657,600	Mem
c220	Mem	Mem	Mem	Mem	Mem
r10	704	514	1,303	861	633
r15	3,609	1,636	3,453	2,777	1,999
r20	15,461	3,478	6,827	6,117	4,243
r25	51,756	6,787	11,458	12,449	8,299
r30	118,588	11,226	17,393	21,017	13,683
r35	508,284	17,574	24,784	33,263	20,984
r45	Mem	35,187	44,170	70,503	43,329
r55	Mem	Mem	69,320	130,001	75,304
r65	Mem	Mem	100,419	218,600	129,079
10x10	688	584	1,312	994	763
18x10	1,387	514	2,628	1,281	1,721
15x20	106,379	1,737	4,703	4,023	4,284
20x20	338,022	3,394	6,757	6,777	5,733
25x25	Mem	4,976	11,397	11,069	11,677
40x40	Mem	Mem	33,077	42,917	37,369
45x50	Mem	Mem	48,912	52,160	70,184
40x79	Mem	Mem	71,211	90,420	99,630
50x100	Mem	Mem	120,732	175,199	202,626

〈표 3〉 만족성 처리기 처리시간

Problem	(1) ∧ (2)	(3) ∧ (4)	4D-DT	(5) ∧ (6) ∧ (7)	(8) ∧ (9) ∧ (10)
c10	0.008	0.004	0.004	0.003	0.003
c20	9.236	51.251	0.016	0.015	0.015
c30	Mem	Mem	0.076	0.045	0.059
c60	Mem	Mem	0.544	0.321	0.832
c120	Mem	Mem	3.844	2.441	1.320
c160	Mem	Mem	Mem	6.320	Mem
c180	Mem	Mem	Mem	9.740	Mem
c200	Mem	Mem	Mem	16.165	Mem
c220	Mem	Mem	Mem	Mem	Mem
r10	0.004	0.000	0.004	0.004	0.004
r15	0.028	0.008	0.012	0.016	0.012
r20	0.200	0.044	9.968	0.032	0.056
r25	2.640	0.248	369.080	0.140	0.176
r30	21.937	1.532	179.790	0.496	0.816
r35	Time	7.428	Time	10.668	2.188
r45	Mem	Time	Time	40.234	25.985
r55	Mem	Mem	Time	85.585	121.604
r65	Mem	Mem	Time	Time	Time
10x10	0.004	0.000	0.008	0.004	0.004
18x10	0.008	0.000	0.008	0.004	0.004
15x20	30.625	0.120	0.020	0.028	0.028
20x20	99.418	0.476	1.604	0.032	0.028
25x25	Mem	22.101	71.008	0.084	0.128
40x40	Mem	Mem	Time	0.652	0.540
45x50	Mem	Mem	Time	1.188	1.860
40x79	Mem	Mem	Time	2.224	1.964
50x100	Mem	Mem	Time	4.144	4.084

5. 결론 및 향후 연구

전산학의 여러 문제들을 명제 논리의 만족가능성 문제로 간주하는 연구들은[6, 7, 8, 9] 최근 만족가능성 처리기들[3, 4, 5]의 높은 성능에서 비롯되었다. 이진 기수 제약 조건은 실세계의 문제들을 명제 논리의 만족가능성 문제로 변환하는데 자주 쓰이는 기법이다[9, 10, 11]. 본 논문은 이러한 연구의 연장선상에서 이진 기수 제약조건과 같이 사용해서 서브시퀀스 추출 문제를 CNF로 변환하는 방법을 제안했다. 즉, 이미지 복원 문제에서 시퀀스 정보를 활용하면 이진 기수 제약조건만을 사용한 변환보다 더 작고 빨리 해결되는 CNF 식을 만들어 낸다.

본 논문에서는 서브시퀀스를 CNF로 변환하는 4가지 방법을 제시했다. 그리고 다양한 형태의 데이터로 처리 시간을 실험한 결과 대체적으로 이진 트리를 이용한 (5)∧(6)∧(7) 방식이 가장 효율적임을 알 수 있었다. 이러한 방식은 이산 단층촬영 문제와 퍼즐 풀이 및 소프트웨어 모델 검증 분야에 활용될 것으로 기대된다.

참 고 문 헌

[1] E. M. Clarke, O. Grumberg and D. Peled, Model Checking, MIT Press, 1999.

[2] M. Davis, G. Logemann, and D. Loveland, “A Machine Program for Theorem Proving.” Communications of the ACM 5 (7): 394 - 397, 1962.

[3] Marques-Silva, J. P., and Sakallah, K. A., “GRASP: A Search Algorithm for Propositional Satisfiability,” IEEE Transactions on Computers, Vol.48, pp.506-521, 1999.

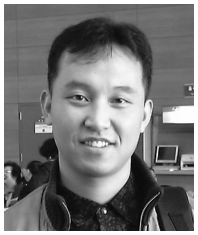
[4] M.W. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik, “Chaff: Engineering an Efficient SAT Solver,” In Proceedings of Design Automation Conference, 2001.

[5] http://www.cs.chalmers.se/Cs/Research/For-malMethods/MiniSat/

[6] H. Kautz and B. Selman. “Planning as satisfiability,” In Proceedings of the ECAI’92, pages 359 - 363. John Wiley & Sons, Inc., 1992.

[7] A. Biere, A. Cimatti, E. Clarke, Ofer Strichman, and Y. Zhu,

"Bounded Model Checking," Vol.58 of Advances in Computers, 2003.
[8] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple Is Better: Efficient Bounded Model Checking for Past LTL," in the Proceedings of VMCAI 2005, Vol.3385, LNCS, 2005.
[9] G. Kwon and H. Jain, "Optimized CNF Encoding for Sudoku Puzzles," in The Proceedings of LPAR06, 2006.
[10] O.Bailleux and Y. Boufkhad, "Efficient CNF encoding of Boolean cardinality constraints," in Proceedings of the CP 2003, Vol.2833, LNCS, 2003.
[11] C. Sinz, "Towards an optimal CNF encoding of Boolean cardinality constraints," In Proceedings of the CP 2005, Vol.3709, LNCS, 2005.
[12] K. J. Batenburg, An evolutionary algorithm for discrete tomography, Discrete Applied Mathematics, 2004.
[13] B. J. Batenburg and W. A. Kosters, "A discrete tomography approach to Japanese puzzles," in Proceedings of the Belgian-Dutch Conf. Artificial Intelligence, 2004.
[14] http://en.wikipedia.org/wiki/Nonogram



박 사 천
e-mail : sachem@kgu.ac.kr
2001년 경기대학교 전산학과(학사)
2003년 경기대학교 전산학과(석사)
2004년~현 재 경기대학교 전산학과 박사과정
관심분야 : 소프트웨어 설계 및 분석, 정형 기법, 만족가능성 문제

권 기 현
e-mail : khkwon@kgu.ac.kr
1985년 경기대학교 전산학과(학사)
1987년 중앙대학교 전산학과(이학석사)
1991년 중앙대학교 전산학과(공학박사)
1998년~1999년 독일 드레스덴 대학 전자계산학과 방문교수
1999년~2000년 미국 카네기 멜론 대학 전자계산학과 방문교수
2006년~2007년 미국 카네기 멜론 대학 전자계산학과 방문교수
1991년~현 재 경기대학교 정보과학부 교수
관심분야 : 소프트웨어 모델링, 소프트웨어 분석, 정형 기법, 소프트웨어공학