2002 년 한국정보처리학회 춘계학술발표논문집 제 9 권 제 1 호

상태 불변식의 투영

전승수, 권기현
경기대학교 전자계산학과
{dabins, khkwon}@kyonggi.ac.kr

Projections of State Invariant

Seungsu Chun, Gihwon Kwon
Dept. of Computer Engineering, Kyonggi University

요 약

상태 불변식은 도달 가능한 모든 상태에서 항상 만족되는 식이다. 이는 복잡한 시스템의 이해와 분석에 활용될 뿐만 아니라 시스템의 안전성, 일관성, 완전성 검사에 사용된다. 이와 같은 이유로, 상태 불변식의 추론은 폭 넓게 연구되었다. 기존의 연구에서는, 시스템의 가능한 모든 상태를 고려하여 상태 불변식을 생성했다. 이러한 방법으로 생성된 상태 불변식은 매우 복잡하며 사용자가 이를 이해하고 활용하기가 어렵다. 본 연구에서는 두 가지 투영 방법을 통해 단순화된 상태 불변식을 생성한다. 첫째, 지정 범위를 통한 투영으로 고려해야 할 상태들을 줄인다. 둘째, 해당 범위에서 사용된 명제들로 상태 불변식을 투영하여 최소 부정 불변식을 찾고 이를 제거한다. 그 결과 사용자의 관심을 특정 범위와 명제에 집중시킬 수 있었으며 단순화된 상태 불변식을 통해 사용자의 이해와 활용도를 높일 수 있었다.

1. 서론

정리 증명[1] 또는 모델 체킹 기법[2]을 이용하여 시스템을 정형 검증할 때, 검사될 대부분의 속성은 상태 불변식(이하 불변식)과 밀접한 관련이 있다. 예를 들어, 안전성 속성은 불변식의 일종이다. 따라서 효과적인 불변식의 생성이 정형 검증의 효과를 크게 좌우한다. 본 연구에서는 추상화 수준이 높은 요구 명세에서 불변식을 생성하고 투영한다. 기존의 연구에서는, 시스템의 도달 가능한 상태를 모두 고려하였다. 일반적으로 도달 가능한 상태의 수가 적지 않기 때문에, 생성된 상태 불변식은 복잡하고 이해하기 어려웠다. 우리는 다음 두 가지 방법을 사용하여 단순화된 상태 불변식을 투영한다. 첫째, 도달 가능한 상태 공간을 모두 고려하기 보다는 특정한 범위내의 상태만을 고려한다. 고려해야 할 상태들의 범위를 제한하였기 때문에, 결과적으로 단순화된 불변식을 얻을 수 있었다. 둘째, 단순 명제들을 모두 고려하기 보다는, 해당 범위에서 사용된 원소 명제들로 상태 불변식을 투영한다. 이렇게 함으로서, 불변식의 크기를 줄일 수 있을 뿐만 아니라 범위에서 항상 만족될 수 없는 최소 부정 불변식 구할 수 있었다.

그 결과 사용자의 관심을 특정 범위와 명제에 집중시킬 수 있었으며 불변식에 대한 이해와 활용을 높일 수 있다.

2. 불변식 생성과 CTL

본 논문에서는 상태 불변식을 다루며, 그 정의는 '도달 가능한 모든 상태에서 만족되는 식'이다[8]. 먼저 명제식의 구문은 다음과 같다:

$$\phi ::= x \mid \neg\phi \mid \phi_1 \wedge \phi_2$$

여기서 $x \in X$ 는 단순 명제를 나타낸다. '상태 s 에서 ϕ 가 만족' 되는 경우를 $s \vDash \phi$ 라고 하자. 만족성 관계는 아래와 같이 귀납적으로 정의된다:

$$s \vDash x \quad \text{iff} \quad x \in L(s), \quad s \vDash \neg\phi \quad \text{iff} \quad s \nvDash \phi$$
$$s \vDash \phi_1 \wedge \phi_2 \quad \text{iff} \quad s \vDash \phi_1 \text{ and } s \vDash \phi_2$$

정리: Y 를 상태 s 에서 만족되는 단순 명제들의 집합이라 하자. 즉 $Y=L(s)$. $Y \subseteq X$ 일 때, 아래 식은

$$\bigwedge_{x \in Y} x \wedge \bigwedge_{x \in X \setminus Y} \neg x$$

상태 s 에서 만족된다. 이와 같은 방식으로 각각의 상태에서 만족되는 식을 결정할 수 있다. 편의상, 도달 가능한 상태들의 집합을 Q 라고 하자. 불변식 I 이란 도달 가능한 모든 상태에서 만족되는 식으로 정의되었기 때문에, 다음과 같이 각 상태에서 만족되어야 하는 식을 이접(disjunction)하면 불변식이 된다.

$$I = (p \wedge q \wedge \neg r \wedge \neg t) \vee (\neg p \wedge \neg q \wedge r \wedge \neg t) \vee (p \wedge \neg q \wedge \neg r \wedge t) \vee (\neg p \wedge q \wedge r \wedge \neg t)$$

이렇게 생성된 불변식은 다른 모든 불변식을 대표하는 가장 강한 불변식이다. 본 논문에서는 CTL[2] 모델 체킹에서 사용되는 크립키 구조를 상태 기계 모델로 사용하고, CTL 질의[3]을 통해 불변식을 투영한다.

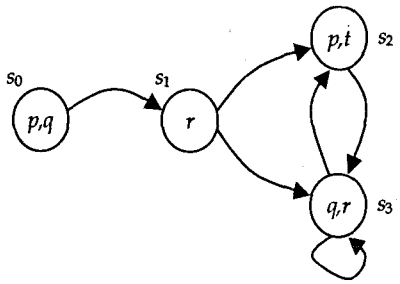

그림 1 모델의 예

분기 구조에 기반을 둔 CTL(Computation Tree Logic) 시제 논리는 속성 표 현 및 범위 지정에 효과적이다. CTL 식의 구문은 전 장의 명제식의 구문에다 몇 개의 시제 연산자를 추가한 것이다.

$$\phi ::= x \mid \text{true} \mid \text{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \text{AG}\,\phi \mid \text{A}(\phi_1 \,\text{W}\, \phi_2)$$

연산자 A 는 '모든 경로'를 나타내며, G 는 '항상'을 의미한다. 또한 W 는 '약한 까지(weak until)'로서, $\text{A}(\phi_1 \,\text{W}\, \phi_2)$의 의미는 '$\phi_2$ 가 발생하기까지 ϕ_1 이 참이 거나, 또는 항상 ϕ_1'이다. 다른 CTL 연산자 및 의미는 [4]에 자세히 소개되어 있다.

3. 범위 지정을 통한 투영

시제 시스템에서 일반적으로 사용될 수 있는 범위는 그림 2 와 같이 5 가지로 구분된다.

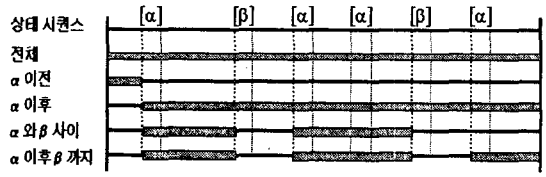

그림 2 CTL 식 계산에서의 상태 포함

지정된 범위 내에 있는 상태들을 생성하기 위하여, CTL 식에 있는 ϕ를 true 로 대치한다면, 해당 식을 만족하는 상태들의 집합을 구할 수 있다. 예를 들어, '전체 상태'는 AG(true)로 나타낸다. 다시 말해서, AG (true) 는 '참'이 만족되는 모든 상태를 의미한다. 단순 명제를 통하여 범위를 지정하고, 범위에서 만족되는 상태 불변식을 추출하는 방법을 설명한다. 입력, 처리, 그리고 출력 부분은 다음과 같다:

-입력 : 범위 지정 CTL 식

-처리 : 범위를 만족하는 상태의 집합.
상태에서 만족되는 명제식의 집합

-출력 : 범위에서 만족되는 불변식

한편, 불변식의 생성을 위해서는 상태 공간 탐색이 요구된다. 주어진 상태들의 집합으로부터 도달되는 다음 상태 집합과 이전 상태들의 집합을 구하기 위하여 아래와 같은 두 가지 함수를 정의한다:

$$\text{post}_\exists(P) = \{s \in S \mid \exists s \bullet s \in P \wedge (s, s') \in R\}$$

중요한 점은 불변식의 생성은 어떤 상태의 도달성 뿐만 아니라 범위에서 만족되는 상태를 모두 얻어야 한다는 것이다. 한편, 전방향 탐색은 초기 상태로 부터 ϕ를 만족하는 상속자 (successors)의 집합 R_ϕ 을 얻을 때 사용 된다. ϕ를 만족하면서 전방향 탐색으로 도달 가능한 모든 상태를 얻기 위한 것이다. R_ϕ 은 다음과 같이 정의된다.

$$R_\phi = \mu Z.((S \cup \text{post}_\exists(Z)) \cap [\phi])$$

불변식 생성에 있어 우리는 도달성이 보장되면서 그림 4 와 같은 식 계산의 실제 범위에서의 상태만을 얻고자 한다. 이를 위해 우리는 전 방향 탐색만을 이용한다. 여기서는 제시된 알고리즘을 그림 2 의 모형에 적용한다. 먼저 지정된 범위에서 만족되는 상태의 집합을 보인 후에, 불변식을 얻는다. 고정점(fixpoint) 계산의 단계는 Fix^0 부터 Fix^n 으로 나타낸다.

1.전체 범위

입력 : AG true

처리 : $\text{AG true} = S \cup R_{\text{true}} \cup \text{post}_\exists(R_{\text{true}})$

Fix^0 = true, $S \mapsto Y$,

Fix^1 = $(\text{true} \cup (\text{true} \mapsto Y)) = \{s_0\}$

Fix^2 = $(\{s_0\} \cup \{s_0\} \mapsto Y) = \{s_0, s_1\}$

Fix^3 = $(\{s_0, s_1\} \cup \{s_0, s_1\} \mapsto Y) = \{s_0, s_1, s_2, s_3\}$

Fix^4 = $(\{s_0, s_1, s_2, s_3\} \cup \{s_0, s_1, s_2, s_3\} \mapsto Y) = \{s_0, s_1, s_2, s_3\}$

출력 : $I = (p \wedge q \wedge \neg r \wedge \neg t) \vee (\neg p \wedge \neg q \wedge r \wedge \neg t) \vee (p \wedge \neg q \wedge \neg r \wedge t) \vee (\neg p \wedge q \wedge r \wedge \neg t)$

2. 'r 이전' 범위

입력 : $\text{A}([\text{true}] \vee \text{AG}(\neg r))\ W\ r)$

처리 : $\text{A}((\text{true} \vee \text{AG}\ \neg\alpha\ W\ \alpha) = S \cup R_{\neg\alpha} \cup \text{post}_\exists(R_{\neg\alpha})$

Fix^0 = false, $\varnothing \mapsto Y$

Fix^1 = $\{s_0\} \cup \{s_0\} \cup \varnothing \mapsto Y = \{s_0\}$

Fix^2 = $\{s_0\} \cup \{s_0\} \cup \{s_0\} \mapsto Y = \{s_0\}$

Fixpoint = $\{s_0\}$, $<s_0 = \{p, q\}>$

출력 : $I = (p \wedge q \wedge \neg r \wedge \neg t)$

3. 'r 이후' 범위

입력 : $\text{AG}(r \Rightarrow \text{AG}[\text{true}])$

처리 : $\text{AG}(r \Rightarrow \text{AG}[\text{true}]) = \text{AG}(\neg\alpha \vee \text{AG}\ \phi)$

$= S \cup R_{\text{true}} \cup \text{post}_\exists(R_{\text{true}}) \nearrow S \cup R_{\neg r} \cup \text{post}_\exists(R_{\neg r})$

$S \cup R_{\text{true}} \cup \text{post}_\exists(R_{\text{true}}) = A'$

Fix^0 = false, $\varnothing \mapsto Y$

Fix^1 = $\{s_0\} \cup \{s_0\} \cup \varnothing \mapsto Y = \{s_0\}$

Fix^2 = $\{s_0\} \cup \{s_0, s_1\} \cup \{s_0\} \mapsto Y = \{s_0, s_1\}$

Fix^3 = $\{s_0, s_1\} \cup \{s_0, s_1, s_2, s_3\} \cup$

$\{s_0,s_1\}\mapsto Y = \{s_0,s_1,s_2,s_3\}$

$Fix^4 = \{s_0,s_1,s_2,s_3\}\cup\{s_0,s_1,s_2,s_3\}\cup$
$\{s_0,s_1,s_2,s_3\}\mapsto Y = \{s_0,s_1,s_2,s_3\}$

Fixpoint A' = $\{s_0,s_1,s_2,s_3\}$

$S\cup R_{\neg r}\cup post_\exists(R_{\neg r})$ = B'

= r 이전과 같다.

Fixpoint B' = $\{s_0\}$

A' ↗B'= $\{s_0,s_1,s_2,s_3\}$↗$\{s_0\}$= $\{s_1,s_2,s_3\}$

혹은, $AG(r\Rightarrow AG[true])=S\cup R_{true}\cup post_\exists(post_\exists(R_{\neg\alpha}))$

$Fix^0 = \{s_1\}, R_{\neg\alpha}\mapsto Y$

$Fix^1 = \{s_1\}\cup\{s_1\}\mapsto Y = \{s_1,s_2,s_3\}$

$Fix^2 = \{s_1,s_2,s_3\}\cup\{s_2,s_3\}\mapsto Y = \{s_1,s_2,s_3\}$

Fixpoint = $\{s_1,s_2,s_3\}$

$<s_1 = \{r\}, s_2 = \{p, t\}, s_3 = \{q, r\}>$

결과 : $I = (\neg p\wedge\neg q\wedge r\wedge\neg t)\vee(p\wedge\neg q\wedge\neg r\wedge t)$
$\vee(\neg p\wedge q\wedge r\wedge\neg t)$

4. 'q 와 t 사이' 범위

입력 : $AG(q\wedge\neg t\Rightarrow A([true]\vee AG(\neg t))\ W\ t))$, M

처리 : $AG(q\wedge\neg t\Rightarrow A([true]\vee AG(\neg t))\ W\ t)$
$= S\cup R_{q\wedge\neg t}\cup(\neg t\cap post_\exists(post_\exists(R_{q\wedge\neg t})))$

$Fix^0 = \{s_0,s_3\}, R_{q\wedge\neg t}\mapsto Y$

$Fix^1 = \{s_0,s_3\}\cup\{s_0,s_3\}\mapsto Y = \{s_0,s_1,s_3\}$

$Fix^2 = \{s_0,s_1,s_3\}\cup\{s_0,s_1,s_3\}\mapsto Y = \{s_0,s_1,s_3\}$

Fixpoint = $\{s_0,s_1,s_3\}$

$<s_0 = \{p,q\}, s_1 = \{r\}, s_3 = \{q, r\}>$

결과 : $I = (p\wedge q\wedge\neg r\wedge\neg t)\vee(\neg p\wedge\neg q\wedge r\wedge\neg t)$
$\vee(\neg p\wedge q\wedge r\wedge\neg t)$

5. 'q 이후 t 까지' 범위

입력 : $AG(q\wedge\neg t\Rightarrow A(true\ W\ t))$, M

처리 : $AG(q\wedge\neg t\Rightarrow A([true]\ W\ t)$
$= S\cup R_{q\wedge\neg t}\cup(\neg\alpha\cap post_\exists(post_\exists(R_{q\wedge\neg t})))$

$Fix^0 = \{s_0,s_3\}, R_{q\wedge\neg t}\mapsto Y$

$Fix^1 = \{s_0,s_3\}\cup\{s_0,s_3\}\mapsto Y = \{s_0,s_1,s_3\}$

$Fix^2 = \{s_0,s_1,s_3\}\cup\{s_0,s_1,s_3\}\mapsto Y = \{s_0,s_1,s_3\}$

Fixpoint = $\{s_0,s_1,s_3\}$

$<s_0 = \{p,q\}, s_1 = \{r\}, s_3 = \{q, r\}>$

결과 : $I = (p\wedge q\wedge\neg r\wedge\neg t)$
$\vee(\neg p\wedge\neg q\wedge r\wedge\neg t)\vee(\neg p\wedge q\wedge r\wedge\neg t)$

한편, 이러한 범위를 서로 결합시켜 지정 범위를 더욱 좁힐 수 있다. 예를 들어, 'q 와 t 사이에서 r 이후의 불변식'을 얻고자 할 때 CTL 식은

$AG(q\wedge\neg t\Rightarrow A[((r\Rightarrow A[\neg t\ U\ (true\wedge\neg t)])\vee AG(\neg t))\ W\ t])$

이다. 이는 기존의 q 와 t 사이 범위와 r 이후 범위를 결합하여 얻을 수 있으며 얻고자 하는 범위의 도식적 해석은 그림 3 과 같다. 이는 '$\neg t\wedge q$'를 만족하는 상태로부터 '$\neg t$'를 통해 도달 가능한 모든 상태들의 집합에서 '$\neg t\wedge\neg r$'을 통해 도달 가능한 모든 상태들의 집합을 제거하는 것이다.

그림 3 에서 (1)의 범위는 q 와 t 사이의 범위이다. 그리고 (2)는 q 와 t 사이에서 ¬r 의 범위이다. 따라서 (1)의 범위에서 도달 가능한 모든 상태들의 집합에서 (2)의 범위를 제거하면 원하는 q 와 t 사이에서 r 이후에 만족되는 불변식을 얻을 수 있다.

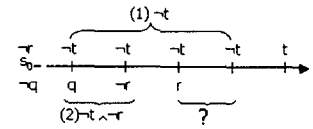

그림 3 'q 와 t 사이 범위에서 r 이후'

(1) $S\cup R_{q\wedge\neg t}\cup(\neg t\cap post_\exists(post_\exists(R_{q\wedge\neg t})))$
Fixpoint(1) = $\{s_0,s_1,s_3\}$
(2) $S\cup R_{q\wedge\neg t}\cup(\neg r\cap post_\exists(post_\exists(R_{q\wedge\neg t})))$
Fixpoint(2) = $\{s_0\}$
$AG(q\wedge\neg t\Rightarrow A[((r\Rightarrow A[\neg t\ U\ (true\wedge\neg t)])\vee AG(\neg t))\ W\ t])$
=(1) ↗(2)=$\{s_0,s_1,s_3\}$↗$\{s_0\}$ = $\{s_1,s_3\}$

4. 원소 명제의 투영

본 장에서는, 지정된 범위 내에서 사용되지 않는 단순 명제, 즉 도달 가능한 모든 상태에서 항상 만족될 수 없는 식인 최소 부정 불변식을 찾아내어, 이를 생성된 불변식에서 제거함으로써 불변식의 길이를 줄인다. 불변식 I 가 주어졌을 때, I 를 y 에 투영한다는 것을 $I\downarrow y$ 으로 나타내고, 그 정의는 $I\wedge y \Leftrightarrow (I\downarrow y)\wedge y$ 이다. 즉 I 에서 y 를 제외한 것과 같다. 아래의 대치 오퍼레이션으로 구현할 수 있다[5].

$I\downarrow y = I[false/y]$

집합 $Y=\{y_1,\ldots,y_n\}$로 일반화하는 경우를 보자.

$I\wedge Y \Leftrightarrow (I\downarrow Y)\wedge(y_1\wedge\ldots\wedge y_n)$, $I\downarrow Y = I[false/y_1, \ldots, false/y_n]$

Y 는 두 곳으로부터 얻을 수 있다. 첫째, 사용자가 관심 없어 하는 변수들을 지정할 수 있다. 둘째, 도달 가능한 상태 집합 Q 로부터 아래와 같이 구할 수 있다:$Y = X - \cup_{s\in Q}L(s)$ 따라서 $Y\subseteq X$ 이다.

후자의 경우를 살펴보자. 앞에서와 같이 질의로 $AG(q\wedge\neg t\Rightarrow A([true]\vee AG(\neg t))\ W\ t))$를 사용하였을 때, 도달 가능한 상태와 불변식은

$P = \{s_0,s_1,s_3\}$

$I = (p\wedge q\wedge\neg r\wedge\neg t)\vee(\neg p\wedge\neg q\wedge r\wedge\neg t)\vee(\neg p\wedge q\wedge r\wedge\neg t)$

이었다. 생성된 불변식에서 제거될 Y 는 생성 불변식에서 얻어진다.

$(p\wedge q\wedge\neg r\wedge\neg t)\vee(\neg p\wedge\neg q\wedge r\wedge\neg t)\vee(\neg p\wedge q\wedge r\wedge\neg t)$
$=\neg t\wedge((p\wedge q\wedge\neg r)\vee(\neg p\wedge\neg q\wedge r)\vee(\neg p\wedge q\wedge r))$ -∨ 제거
$=\neg t$ -긍정 제거

두 상태에서 사용된 변수들은 {p,q,r} 이기 때문에, $Y = \{t\}$이다. 그러므로

$I\downarrow Y = I[false/t]$
$= (p\wedge q\wedge\neg r\wedge\neg false)\vee(\neg p\wedge\neg q\wedge r\wedge\neg false)$
$\vee(\neg p\wedge q\wedge r\wedge\neg false)$
$= (p\wedge q\wedge\neg r\wedge true)\vee(\neg p\wedge\neg q\wedge r\wedge true)\vee(\neg p\wedge q\wedge r\wedge true)$
$= (p\wedge q\wedge\neg r)\vee(\neg p\wedge\neg q\wedge r)\vee(\neg p\wedge q\wedge r)$

이다.

5. 최소 불변식

특정 범위에서 만족되는 불변식은 주어진 범위에서 만족되는 모든 불변식을 대표하는 가장 강한 불변식이다. 우리는 이러한 가장 강한 불변식을 원소 명제에 의해 다시 투영했다. 따라서, 원소 명제에 의해 투영된 불변식은 가장 강한 불변식이 아니다. 하지만 만족되지 않는 원소 명제만을 제거하기 때문에 범위에서 존재하는 원소 명제의 모든 불변식은 대표한다. 우리는 이를 '최소 불변식'이라 한다.

한편, 얻어진 Y 의 원소 명제를 연언(conjunction)하면 최소 부정 불변식을 얻을 수 있다. 얻어진 Y 는 모든 부정 불변식을 대표하지는 않지만 매우 정확하다. 또한 모형 검사에서 효율적으로 사용될 수 있다. 얻어진 Y 는 주어진 범위의 모든 상태에서 만족될 수 없는 원소 명제이다. 예를 들어 주어진 범위에서 항상 만족될 수 없는 원소 명제의 집합 Y 의 원소 명제가 {q,p} 였다면 주어진 범위는 항상 '$\neg p \wedge \neg q$'를 만족해야 한다. 따라서 CTL 식의 $AG\neg(p \vee q)$로 나타낼 수 있다. 예를 들어, 얻어진 $Y = \{q,p\}$가 r 과 q 사이에서 얻어진 최소 부정 불변식이라면 지정된 범위는 다음과 같이 CTL 로 명세될 수 있다.

$$AG(r \wedge \neg q \rightarrow A((AG\neg(p \vee q) \vee AG\neg q)\ W\ q))$$

원소 명제를 통한 투영 방법은 범위에서의 상태 불변식의 크기를 줄일 뿐만 아니라 특정 범위에서의 원소 명제를 분석하고 이해할 때도 적용 가능하다.

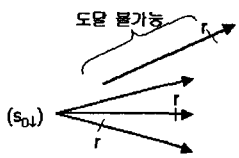

그림 4 'r 이 만족되는 상태'에서의 탐색 문제

6. 관련 연구 및 비교

프로그램으로부터 불변식을 추론하고 사용하는 기존의 연구[6,7]들과는 다르게 개발 초기 단계의 요구 명세로부터 얻은 불변식은 분석과 생성이 용이하고 비용 효과적이다. 이러한 이유로, SCR(Software Cost Reduction) 또는 RSML(Requirements State Machine Language)명세와 같은, 상위 단계의 모델로부터 불변식을 추출하는 연구들이 진행되었다[8,9]. 또한 추출된 불변식은 모델의 문서화, 일관성 검사, 그리고 완전성 검사등에 활용됐다[8,9]. 기존의 연구에서는 전체 상태만을 고려했다. 따라서 생성된 불변식이 복잡하고 이해하기 어려웠다. 우리는 CTL 질의[3]를 확장하여 특정 범위에서의 불변식을 얻을 수 있는 투영 방법과 원소 명제 투영을 통해 최소 부정 불변식 얻고 이를 제거하여 불변식을 줄이는 방법을 보였다.

한편, 불변식은 일반적으로 기호적 모형 검사 기술인 전 방향 혹은 역 방향 탐색을 통해 자동화된다. 전 방향 탐색의 경우 얻어진 상태의 도달성은 보장되지만 가능한 모든 경로와 상태를 탐색해야하는 문제가 있다. 반면, 역 방향 탐색의 경우 그림 4 와 같이 어떤 원소 명제가 만족되지 않는 범위의 탐색은 피할 수 있지만 그림과 같이 초기 상태로부터 도달되지 못하는 상태까지 탐색하게 된다. 불변식 생성에 있어 우리는 도달성이 보장되면서 식 계산의 실제 범위에서의 상태만을 얻고자 한다. 이를 위해 우리는 전 방향 탐색만을 이용한다.

7. 결론

본 연구에서는 불변식을 단순화하기 위해 범위 지정과 원소 명제를 통한 투영 방법을 사용했다. 이를 통해 복잡한 불변식과 원소 명제를 효과적으로 줄일 수 있었으며 범위에서의 최소 부정 불변식을 얻을 수 있었다. 이는 시스템에 대한 사용자의 이해를 높일 뿐만 아니라 보다 효율적인 모형 검사를 위한 단순화된 불변식을 생성했다. 향후 연구 과제는 상태도[10]와 같이 고 수준의 상태 기계 모델로부터 불변식을 생성하는 것이며 자동화를 위해 BDD[11](Binary Decision Diagram) 자료구조를 이용할 것이다.

참고문헌

[1]Z. Manna, et. al., "STeP: The Stanford temporal prover," Technical Report STAN-CS-TR-94-1518, Computer Science Department,, Stanford University, 1994.

[2]E.M. Clarke, et. al., "Automatic verification of finite state concurrent systems using temporal logic specifications," ACM Transactions on Programming Languages and Systems, Vol.8, No.2, pp. 244 - 263, 1986.

[3]William Chan, "Temporal Logic Queries", Proceedings of CAV 2000, LNCS 1855, Springer, 2000.

[4]E.M. Clarke, et. al., Model Checking, MIT Press, 1999.

[5]M. Fitting, First-Order Logic and Automated Theorem Proving, 2nd Edition, Springer, 1996.

[6]S. Bensalem, et. al., "Automatic generation of invariants," Formal Methods in System Design, Vol.15, pp.75-92, 1999.

[7]S. Bensalem, et al., "Powerful techniques for the automatic generation of invariants", Proceedings of CAV' 96, LNCS 1102, Springer, 1996.

[8]R. Jeffords, C. Heitmeyer, "Automatic generation of state invariants from requirements specifications," Proceedings of ACM SIGSOFT Symposium on Foundations of Software Engineering, 1998.

[9]C. Heitmeyer, et. al., "Using abstraction and model checking to detect safety violations in requirements specifications", IEEE Transactions on Software Engineering, Vol.24, No.11, 1998.

[10]D.Harel et. Al., "The STATMATE semantics of statecharts",ACM Transactions on Software Engineering and Methodology, Vol.5, No.4, pp.293 - 333, 1996.

[11]R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computer, Vol. 35, No.8, pp.677-691, 1986