2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

계층형 크립키 구조를 위한 CTL 모형검사 알고리즘

박사천, 권기현
경기대학교 정보과학부 소프트웨어공학 연구실
{sachem, khkwon}@kyonggi.ac.kr

CTL Model Checking Algorithm for Hierarchical Kripke Structure

Sachoun Park, Gihwon Kwon
Software Engineering Laboratory, Dept. of Computer Science, Kyonggi University

요 약

시스템의 안전성을 검사하기 위한 연구가 활발히 진행중이다. 그 중에서도 모형검사를 이용하는 방법이 가장 일반적이라고 할 수 있는데, 이 방식에는 몇 가지 문제점이 지적되고 있다. 그 중 가장 심각한 문제가 상태 폭발 문제이다. 상태 폭발은 모형 검사기가 큰 규모의 시스템들을 다루지 못하게 하는 주요 원인이다. 본 연구는 평탄화에 의해 발생하는 상태폭발의 문제를 극복하기 위하여, 계층형 크립키 구조를 정의하고 그 구조에서 CTL 속성을 검사하는 모형검사 알고리즘을 제시한다.

1. 서론

정형검증은 시스템의 개발과정에서 오류를 방지하거나 개발된 시스템에서 발견되지 않은 오류를 찾아내는 연구이다. 정형검증기법은 수학과 논리에 기반하여 검색할 시스템을 철저히 분석한다. 그 중에 모형검사 기법이 가장 많이 쓰이는데, 시스템을 적당한 모형으로 표현하고 반드시 만족되어야 할 속성들을 시제 논리식으로 표현하여 시스템이 요구된 속성을 만족하는지 기계적으로 검사하는 방법이다. 80 년대 초반에 E. Clarke 에 의해서 CTL 모형검사 방법이 소개된 이후 McMillan 이 OBDD 를 이용한 SMV 를 개발하고 몇 가지 사례에서 성공적으로 오류들을 발견함으로써 산업현장에서 모형검사기법에 대한 적용가능성을 열어주었다[1]. 그러나 CTL 모형검사는 본질적으로 평탄한 구조에서 해석되기 때문에 실제의 시스템에 적용하기에는 많은 문제점을 안고 있다. 검색해야 할 상태공간이 처리할 수 없이 커지는 것을 상태폭발이라고 하는데 이를 극복하기 위한 연구로는 합성을 이용하는 방법과 추상화를 이용하는 것 대칭성과 반 순서를 이용하는 것들이 있다. 또 다른 방향의 연구로는 계층성을 이용한 방법인데 이것은 평탄화를 피하고 시스템의 계층성을 고려해서 탐색해야 할 상태공간을 축소시키려는 방법이다[2].

본 논문에서는 먼저 계층형 크립키 구조를 정의하고 CTL 식이 계층형 크립키 구조에서 어떻게 해석되는지 그 알고리즘을 보인다. 우리가 제시한 계층형 크립키 구조는 계층간의 전이를 허용하지 않으며 상위 구조에서 기술된 속성이 하위 구조로 상속되는 특성이 있다. 속성 상속은 동일한 특성을 지니는 상태들을 묶어서 계층으로 표현한 상태도의 기본원리를 응용한 것이다[3]. 본 논문에서 제시하는 계층은 단순한 크립키 구조의 중첩으로 표현했고 이것은 실세계를 단순하고 명확하게 표현할 수 있다. 또한 계층을 뛰어 넘는 전이를 허용하지 않음으로써 보다 미래 지향적인 모형의 성격을 부여했다.

2. 관련연구

상태도의 전이는 계층을 넘나들며 발생하지만 Argo[4]나 모드[5]같은 구조에서는 계층간의 전이를 허용치 않는다. 프로그램이 명확하게 서브프로그램으로 분해되기 위해서는 계층간의 전이는 억제되어야 하며, [5]에서도 계층을 모드라는 핵심 컴포넌트의 중

첩으로 표현했는데 이때 서브 모드를 블랙박스처럼 여기고 외부로는 입·출력 전이만을 허용하고 있다.

Alur[2]는 같은 계층에서 동일한 개체들이 중복적으로 나타나는 계층형 크립키 구조와 이에 대한 모형검사 알고리즘을 소개하고 있는데 본 논문에서는 보다 일반적인 구조에서 해석되는 알고리즘을 보였다. HSEM(Hierarchical State/Event Model)에서는 병행적인 요소를 다루고 있지만 일반적인 시제속성이 아닌 도달성 검사에 국한한 연구였다[6]. 본 논문에서는 병행적인 모형보다는 일반적인 시제속성에 대한 검사를 중점으로 다루었다.

3. 계층형 크립키 구조와 CTL 정의

3.1 계층형 크립키 구조

정의 1. 계층형 크립키 구조 HK = (N, M, γ, L)는 네 개의 항목으로 이루어지며 M 은 크립키 구조들의 집합이고, N 은 M 에 포함된 최상위 크립키 구조이다. $M = \{M_1, M_2, \ldots, M_n\}$, M 에 속한 각각의 M_i는 $M_i = (S_i, in_i, R_i)$의 세 개의 항목으로 구성되는데 각 항목들은 다음과 같이 정의된다:

- S_i: M_i에 속하는 모든 상태들의 집합
- $in_i \in S_i$: M_i의 시작 상태
- 전이관계 $\forall s \in S_i.\ \exists t \in S_i.\ (s, t) \in R_i$는 전체 관계이다.

$\Sigma = \bigcup S_i$ (i=1 .. n)일 때, $\gamma: \Sigma \rightarrow 2^{\Sigma}$ 계층을 정의하는 함수 γ는 상태 $s \in \Sigma$를 상태집합 $\Sigma' \subset \Sigma$으로 사상 시킨다. Σ'은 s 의 자식 상태들의 집합이다. $\gamma^+(s)$는 s 자신을 포함하지 않는 s 의 모든 자손 상태들의 집합이고, $\gamma^*(s) = \gamma^+(s) \cup \{s\}$이다. 라벨함수 L: $\Sigma \rightarrow 2^{AP}$는 임의의 상태 s 를 기본 명제의 집합 AP 의 부분 집합으로 사상한다

내부에 어떤 상태를 포함하는 상태로 전이 되었을 경우 그 하위 상태들 중 시작 상태로부터 출발한다. 전이관계 R_i 는 같은 i-계층의 상태들 간에 정의된다. 라벨함수 L 은 모든 상태에 라벨 할 수 있음을 보여준다. 속성 상속은 예를 들어 라벨함수에 의해서 상태 s 에 기본명제 p 가 라벨 되었다면 상태 s 의 모든 자손 상태에도 동일하게 라벨 된다고 간주한다. 이것은 알고리즘을 전개해가는 과정에서도 동일하게 적용된다.

3.2 CTL 구문과 의미

대표적인 분기시제 논리인 CTL 은 크립키 구조를 무한 트리의 관점에서 해석한 것이다. CTL 은 명제논리를 포함하며 경로 한정자(path quantifier)와 시제 연산자(temporal operator)로 구성된다. 두개의 경로 한정자 **A**, **E** 는 각각 "모든 경로"와 "어떤 경로"로 해석된다. 네 개의 시제 연산자(**X**, **F**, **G**, **U**)는 모두 현재 상태를 기준으로 해석된다. **Xp** 는 "다음 상태에서 p 가 만족된다."를 의미하고 **Fp** 는 "언젠가는 p 를 만족한다.", **Gp** 는 "모든 상태에서 p 가 만족된다."는 의미로 해석된다. **pUq** 는 "q 가 만족 될 때 까지 p 가 만족된다."는 의미이다. **EGp** 는 "현재 상태로부터 모든 상태에서 p 를 만족하는 경로가 적어도 하나 존재한다."는 의미이다.

정의 2. CTL 구문을 BNF 형식으로 나타내면 다음과 같다:

$\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathbf{EX}\phi \mid \mathbf{EG}\phi \mid \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$.

CTL 식의 다른 구문은 동치 관계로 모두 표현될 수 있다: $\phi_1 \wedge \phi_2 \equiv \neg\phi_1 \vee \neg\phi_2$, $\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$, $\phi_1 \Leftrightarrow \phi_2 \equiv \phi_1 \Rightarrow \phi_2 \wedge \phi_2 \Rightarrow \phi_1$, $\mathbf{A}\pi \equiv \neg\mathbf{E}\neg\pi$, $\mathbf{F}\phi \equiv \mathit{true}\mathbf{U}\phi$, $\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$.

정의 3. CTL 의 의미(semantics)는 크립키 구조 M 에서 아래와 같이 귀납적으로 정의된다:

$M, s_0 \models p$ iff $p \in L(s_0)$
$M, s_0 \models \neg\phi$ iff $M, s_0 \not\models \phi$
$M, s_0 \models \phi_1 \vee \phi_2$ iff $M, s_0 \models \phi_1 \vee M, s_0 \models \phi_2$
$M, s_0 \models \mathbf{EX}\phi$ iff $\exists t.(s_0, t) \in R, M, t \models \phi$
$M, s_0 \models \mathbf{EG}\phi$ iff $\exists \pi = s_0, s_1, s_2, \ldots, \forall i \geq 0, M, s_i \models \phi$
$M, s_0 \models \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$ iff $\exists \pi = s_0, s_1, s_2, \ldots, \exists k \geq 0,$ $M, s_k \models \phi_2 \wedge \forall i.0 \leq i < k, M, s_i \models \phi_1$.

p 는 기본명제 이고 π는 M 에서 상태들의 무한 연속인 경로이다: $\pi = s_0, s_1, \ldots, i \geq 0, (s_i, s_{i+1}) \in R$. 정의하지 않은 CTL 식 **AX**φ, **EF**φ, **AF**φ, **AG**φ, **A**[φ**U**φ]등에 대한 의미는 위의 정의를 이용하여 유도 될 수 있다.

4. 계층형 구조에서 CTL 모형검사 알고리즘

아래의 [그림 1]은 계층형 크립키 구조에서의 모형검사 알고리즘이다. In 은 HK 에 포함된 초기 상태들의 집합으로 정의되고 S_M 은 임의의 크립키 구조 M 에 대한 상태집합, γ(s)는 $s \not\in M$ 일 때 S_M 로 정의된다. 어떤 상태 s 에 포함된 계층의 초기상태를 구하는 함수 I 는 다음과 같이 정의된다: $I(\{s\}) = \gamma(s) \cap In$, $I^+(\{s\}) = I(\{s\}) \cup I(I(\{s\})) \cup \ldots$, $I^*(\{s\}) = I^+(\{s\}) \cup \{s\}$. 초기상태에

```
function HSat(φ: Formula) : set of State;
begin
  if φ = true ⟶ return Σ
  [] φ = false ⟶ return ∅
  [] φ ∈ AP ⟶ return {s:Σ | ∃s'∈Σ·φ∈L(s') ∧∀s∈γ*(s') }
  [] φ = ¬φ ⟶ return {s:Σ | γ*(s) ∩ HSat(φ) ≠ ∅}
  [] φ = φ ∧ ψ ⟶ return (HSat(φ) ∩ HSat(ψ))
  [] φ = EXφ ⟶
       return {t:Σ | ∃s∈Σ·(s, s')∈R∧I*(s') ∩ HSat(φ) ≠ ∅·∀t∈γ*(s)}
  [] φ = EGφ ⟶ return HSatEG(SN, φ)
  [] φ = E[φUψ] ⟶ return HSatEU(SN, φ, ψ)
  fi
end
```

Model Checking problem: $I^*(\{in_N\}) \cap HSat(\phi) \neq \emptyset$.

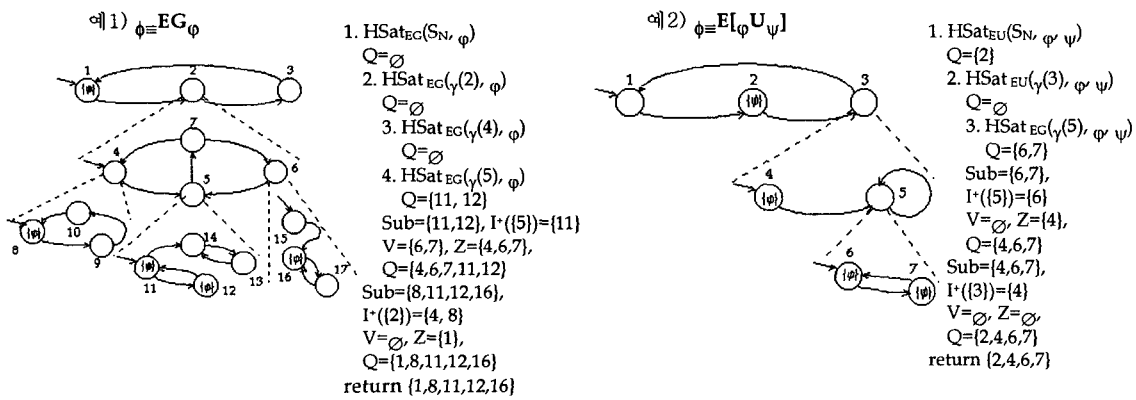
연속은 평탄화 된 구조에서 동일한 상태가 된다.

[그림 1] 계층형 크립키 구조의 모형검사 알고리즘

```
function HSatEG(SM: set of State, φ: Formula) : set of State;
begin var Q, Q', Sub, V, Z, Z': set of State;
  Q, Q' := SM ∩ ¢φ•, SM;
  do Q ≠ Q' ⟶
    Q' := Q;
    Q := (Q∩{s | ∃s'∈Q.(s, s')∈R}) ∪ Sφ
  od;
  foreach s ∈ (SM/Q)
    if γ(s) ≠ ∅ ⟶
      Sub := HSatEG(γ(s), φ);
      Z, V := ∅, ∅;
      if Sub ∩ I+({s}) ≠ ∅ ⟶
        Z, Z' := {s}, ∅;
          do Z ≠ Z' ⟶
            Z' := Z;
            V := V ∪ ({s | ∃s'∈Z.(s, s')∈R}/¢φ•);
            Z := Z ∪ ({s | ∃s'∈Z.(s, s')∈R} ∩ ¢φ•)
          od;
        Z := (Z/{s}) ∪ V
      fi;
      Q := Q ∪ Z ∪ Sub
    fi;
  return {s:Σ | ∃s'∈Q.s∈(γ*(s') ∩ HSat(φ))}
end
```

```
function HSatEU(SM: set of State, φ, ψ: Formula) : set of State;
begin var Q, Q', Sub, V, Z, Z': set of State;
  Q, Q' := SM ∩ ¢ψ•, ∅;
  do Q ≠ Q' ⟶
    Q' := Q;
    Q := Q∪({s | ∃s'∈Q.(s, s')∈R} ∩ ¢φ•)
  od;
  foreach s ∈ (SM/Q)
    if γ(s) ≠ ∅ ⟶
      Sub := HSatEU(γ(s), φ, ψ);
      Z, V := ∅, ∅;
      if Sub ∩ I+({s}) ≠ ∅ ⟶
        Z, Z' := {s}, ∅;
          do Z ≠ Z' ⟶
            Z' := Z;
            V := V ∪ ({s | ∃s'∈Z.(s, s')∈R}/¢φ•);
            Z := Z ∪ (({s | ∃s'∈Z.(s, s')∈R} ∩ ¢φ•)/Q)
          od;
        Z := (Z/{s}) ∪ V
      fi;
      Q := Q ∪ Z ∪ Sub
    fi;
  return {s:Σ | ∃s'∈Q.s∈(γ*(s') ∩ (HSat(φ) ∪ HSat(ψ)))}
end
```

[그림 2] EG_φ와 $E[\varphi U_\psi]$를 라벨 하는 프로시져



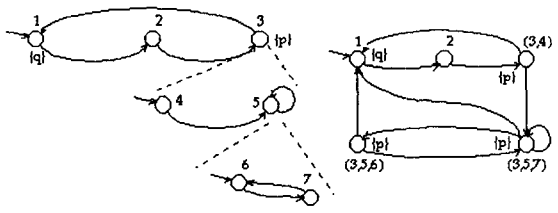
[그림 3] EG_φ와 $E[\varphi U_\psi]$를 알고리즘 적용예제

계층형 크립키 모형에서 CTL 모형검사의 만족성 (HK $\models \phi$)문제는 $I^*(\{in_N\}) \cap HSat(\phi) \neq \emptyset$가 참이 될 때 만족하는 것으로 판명된다. 즉 CTL 식으로 표현된 속성 ϕ를 만족하는 상태들의 집합이 계산되고 최상위 상태의 초기상태나 그 상태에 포함된 연속된 초기 상태들 중에 ϕ로 라벨 되는가를 확인하면 된다. 즉 속성 ϕ가 만족되는 상태 집합에 루트계층의 초기상태 연속이 포함되어 있으면 계층형 크립키 구조 HK 에서 속성 ϕ가 만족하는 것이다.

[그림 2]는 각각 **EG**φ와 **E[**φ**U**ψ**]**를 계산하는 부분이다. 위의 알고리즘은 전통적인 라벨방식의 CTL 알고리즘을 응용한 것으로 계층성의 이점을 살리기 위해 상위 계층의 전이를 최대한 활용하도록 하고 있다. 알고리즘을 시작할 때는 최상위 계층에 해당하는 크립키 구조의 상태집합을 대상으로 하고 재귀적으로 하위 구조들을 검사하게 된다. 중복 검사를 피하기 위해 먼저 검사된 상태는 다음 번 반복에서 제외된다. [그림 3]의 두 예제는 하위 구조에 부분식 φ나 ψ가 라벨 되어 있을 때 **EG**φ와 **E[**φ**U**ψ**]**를 계산하는 알고리즘의 사례를 보인 것이다. 만일 [그림 3]의 예 1)에서 3 번 상태에 φ가 라벨 되었더라면 단 한번의 $Hsat_{EG}(S_N, \varphi)$ 호출로 **EG**φ에 대한 만족성 검사가 완료되었을 것이다. 다시 말해 부분식이 보다 상위의 상태에 라벨 된다면 처리속도는 더욱 빨라질 것이다. 위의 예제에서 첫번째 예제는 **EG**φ가 라벨 된 상태 집합에 최상위

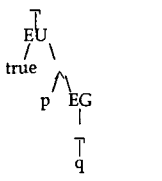
계층의 초기상태가 포함되므로 속성이 만족함을 알 수 있다. 예 2)는 4, 6번 상태에 φ가 라벨 되고 2, 7번 상태에 ψ가 라벨 된 경우에 **E[φUψ]**를 검사하는 예이다. 초기상태에 ϕ가 라벨 되지 않기 때문에 식이 만족하지 않는다.

AP={p, q}이고 주어진 계층형 크립키 구조가 [그림 4]의 1)과 같을 때 AG(p⇒AFq)의 속성이 만족하는가를 검사해보자. 먼저 시제 연산자들의 동치관계에 의해서 다음과 같은 식을 얻는다: ¬E[true U (p ∧ EG¬q)]. 아래의 그림과 같이 식이 분해되고 AP 로부터 라벨링 알고리즘이 적용되면서 전체 식을 만족하는 상태집합을 얻을 수 있다. 예를 들어 HSat(EG¬q)를 계산하는 단계는 알고리즘에 의해 S_N 계층만을 검사하면 된다. 그러나 [그림 4]의 2)는 1)의 계층형 크립키 구조와 동일한 크립키 구조이다. 여기에서 주어진 식을 계산하려면 전체 전이관계와 상태들을 고려하게 된다. 평탄화 된 구조에서 3 개로 표현된 전이들(3,4→1, 3,5,6→1, 3,5,7→1)이 계층형 구조에서는 하나의 전이(3→1)로 줄어 들고 기본명제에 대한 라벨도 상위상태에 되어있기 때문이다. 이러한 이점은 라벨을 진행하면서도 얻을 수 있는데 예를 들어 EXφ에 대한 라벨을 보면 φ가 만족되는 상태를 발견한 후 그 이전상태에 EXφ를 라벨하므로써 그 상태의 하위의 모든 상태들을 탐색하는 수고를 덜 수 있다. 이것은 다른 식에 대해서도 같은 결과를 보여준다.



1) 계층형 크립키 구조 2) 평탄화 된 크립키 구조

$\phi \equiv AG(p \Rightarrow AFq) \equiv \neg E[true\ U\ (p \wedge EG\neg q)]$
$HSat(q) = \{1\}$
$HSat(\neg q) = \{2,3,4,5,6,7\}$
$HSat(EG\neg q) = \{2,3,4,5,6,7\}$
$HSat(EG\neg q) \cap HSat(p) = \{3,4,5,6,7\}$
$HSat(E[true\ U\ (p \wedge EG\neg q)]$
$= HSat_{EU}(S_N, true, p \wedge EG\neg q) = \Sigma$
$HSat(\neg E[true\ U\ (p \wedge EG\neg q)] = \varnothing$



[그림 4] 계층형과 평탄화 된 구조의 속성검사 비교

위 [그림 4]에서는 주어진 식¬E[true U (p ∧ EG¬q)]로 라벨된 상태가 없기 때문에 1)의 계층형 구조에서 주어진 속성이 만족하지 않음을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 계층성을 보다 적극적으로 이용하여 모형검사의 문제점인 상태폭발에 대응하였다. [2]에서 보인 계층형 구조 보다는 일반적인 계층형 크립키 구조를 정의했고 계층형 구조 상에서 해석되는 CTL 알고리즘을 제시했다. [2]는 기본 상태들에만 AP를 라벨했었지만 우리가 제시한 계층형 크립키 구조에서는 모든 상태에 라벨 할 수 있었고, 상위상태에 라벨 된다는 것은 라벨된 상태가 포함하고 있는 하위의 모든 상태에 라벨 된다는 의미로 정의했다. 이로써 최상위 계층으로부터 모형검사를 시작하여서 하위상태로 내려가면서 더 이상 하위를 검사할 필요가 없는 것들은 생략함으로써 모형검사의 처리 속도를 높였다.

하지만 우리가 제시한 모형은 실제 산업에서 사용되는 상태도와 같은 명세언어와는 많은 차이를 보인다. 그 중에서도 계층형 모형에 병행적인 요소를 표현할 수 없는 단점이 있었다. [6]은 모형에 병행성을 표현하고 계층성과 재사용성을 이용한 도달성 검사를 수행했다. 우리는 보다 일반적인 시제속성 표현에 집중했으나 병행성 또한 중요한 특성이기 때문에 앞으로의 연구에서는 모형에 병행성을 추가하는 연구가 이루어져야 할 것이다. 또한 본 논문에서 제시한 알고리즘을 바탕으로 계층형 CTL 모형검사 도구를 구현하는 과제가 남았다.

참고문헌

[1] E. Clarke, D. Long, "Verification tools for finite-state concurrent systems," in Proceedings of A Decade of Concurrency, Lecture Note in Computer Science 803, pp. 124-175, 1994.

[2] R. Alur, M. Yannakakis, "Model Checking of Hierarchical State Machines," in Proceedings of Foundations of Software Engineering, 1998.

[3] D. Harel, A. Naamad, "The STATEMATE semantics of Statecharts," ACM Transactions on Software Engineering and Methodology, Vol.5, No.4, pp.293-333, 1996.

[4] F. Maraninchi, "Operational and Compositional Semantics of Synchronous Automaton Compositions," In Proceedings of CONCUR'92, Lecture Notes in Computer Science 630, pp.550-564, 1992.

[5] M. McDougall, R. Alur, R. Grosu, "Efficient reachability analysis of hierarchical reactive machines," In Proceedings of CVA'00, Lecture Notes in Computer Science 1885, 2000.

[6] G. Behrmann, et. al., "Verification of hierarchical state/event systems using reusability and compositionality," In Proceedings of TACAS'99, 1999.

[7] J.P. Katoen, "Concepts, Algorithms, and Tools for Model Checking," Arbeitsberichte der Informatik, Friedrich-Alexander-Universitaet Erlangen-Nuernberg, Gruner Druck GmbH, 1999.