Symbolic Incremental CTL 모델 검증

채정욱° 인호
고려대학교 정보통신대 컴퓨터학과 ({bravejw,hoh_in}@korea.ac.kr)

On Symbolic Incremental CTL Model Checking

Jung-Wook Chae°, Hoh Peter In
Department of Computer Science and Engineering Korea University

요 약

최근 유비쿼터스 컴퓨팅 환경이 부각되면서 유비쿼터스 컴퓨팅 환경에서의 모델 검증을 위해 기존의 심볼릭 검증 모델을 응용한 매우 크고 여러 콤포넌트로 구성된 모델 검증 대상이 빈번히 변화하는 상황을 가정한 모델 검증 기술이 요구되었다. 본 논문에서는 빈번히 변화하는 모델 검증을 위해 새로운 알고리즘을 제시하고 완전히 증명하였으며 기존의 모델과 새로운 모델간의 성능 비교 실험으로 새로운 모델의 효율성을 증명하였다.

1. 서 론

유비쿼터스 건강 관리 시스템은 병원과 집이 연결된 건강 검진 유비쿼터스 컴퓨팅 시스템이다. 이러한 시스템에서는 시스템의 오류가 인간에게 직접적인 해를 끼칠 수 있기 때문에 시스템은 완전 무결하여야 한다. 뿐만 아니라 시스템의 유지 보수로 인한 시스템의 변화에 대해 무결성이 실시간으로 검증 되어야 한다. 하지만 시스템이 대단위의 분산 시스템이기 때문에 한 번 바뀔 때마다 전부를 검증하면 그 때마다 시간이 많이 소요되었다. 따라서 모델이 변함에 따라 즉시 그 변화된 시스템에 대해 검증하기 위해 기존의 모델 검증 방법인 모든 부분을 다시 계산하는 것 보다 빠른 방법이 요구 되었다. 본 논문의 알고리즘은 변화된 부분과 그 변화된 부분에 의해 영향받는 부분과 기존의 바뀌지 않은 부분을 고려하여 모델 검증 시간 복잡도를 줄인다.

모델 검증에는 두 가지 주 패러다임이 있다. 명시적 모델 검증과 심볼릭 모델 검증이 두 패러다임이다. 심볼릭 모델 검증은 명시적 모델 검증보다 많은 상태 수를 검증할 수 있도록 Kripke 구조를 최적화하고 그 결과인 Ordered Binary Decision Diagram(OBDD)에 μ-calculus를 알고리즘을 적용하여 검증을 하는 방식이다. 본 논문의 연구는 심볼릭에 기반한다.

Incremental Model Checking은 [1]에서 처음 논의되었다. 그리고 [3]에서는 시스템의 변화를 나타내는 Δ를 통해 Kripke 구조의 변화에 따른 알고리즘의 복잡도를 설명하였다.

[2]에서는 Computation Tree Logic(CTL)의 reduce 연산자인 RP와 expand 연산자 EO가 모델 검증을 위해 제안되었다. 이러한 접근법은 공간 복잡도를 줄이기 위해 심볼릭 표현법을 사용하고 있으며 fixpoint 계산에서 빠른 수렴이 되도록 starting point(sp)를 최적화한다.

기존의 연구[2]는 완전히 입증되지 않았으며 본 논문의 알고리즘과 비교해 시간 복잡도가 컸다.

본 논문은 기존의 연구[2]와 비슷한 디자인을 기반으로 Incremental 모델 검증에 적합한 새로운 fixpoint(FP) 계산 방식인 "above estimation"과 "below estimation"을 제시한다. 또한 본 논문은 제시한 알고리즘을 완전히 증명하고 실험함으로써 어떻게 제시한 알고리즘이 EU, EG 연산의 복잡도를 줄이는지 보여준다. 우리의 접근법은 어떠한 시스템의 변화들에도 적용가능하며 기존의 Boolean formulae를 표현하고 다루기 위한 기존의 스키마들과 독립적이다.

2. Baseline CTL Model Checking

CTL 모델 검증에서는 시스템이 Kripke 구조로 모델링 되고 검증하고자 하는 속성은 CTL formulae로 표현된다. Kripke 구조 $K=(S, T, L)$는 유한 상태 기계를 레이블링한다(S는 상태들의 집합, T는 전이 관계).

CTL formula f 는 atomic Boolean propositions 과 기본 연산자인 '~', '∨', 'EX', 'EG', 'EU'의 접합이며 [4]에서와 같이 재귀적 문법으로 정의된다.

본 논문에서 논의되고 있는 기존 모델 검증은 심볼릭 모델 검증이다. 이러한 표현에서, 상태들과 전이들의 집합은 boolean formula로 인코딩된다. Kripke 구조에서의 수행은 이러한 수식을 다룸으로써 구현된다. MC 알고리즘을 설명하기 전에 중요한 부분 중의 하나인 을 설명한다. F가 상태들의 집합일 때, FP알고리즘 FP($_{\tau}$ F)는 함수 $_{\tau}$와 시작 점 F를 입력받는다. 이 알고리즘은 결과가 바뀌지 않을 때까지 $_{\tau}$를 반복적으로 적용한다. τ의 fixpoint라 불리는 이 마지막 결과를 FP의 호출자에게 리턴한다.

Algorithm MC(K, f)
compute the semantics of f's direct sub-formula(e);
case
f=a∈A: f:=L(a);
f=~p: f:=~p;
f=$p \vee q$ f:=$p \vee q$;
f=EXp: f:=EX(T,p);
f=EGp: f:=EG(T,p);
f=EpUq: f:=EU(T,p,q);
end case
return f;

Algorithm EG(T,p)
sp: = $true$;
τ_{EG}: =$\lambda Z.p \wedge EX(T, Z)$;
return FP($_{\tau_{EG}}$ sp);

Algorithm EU(T,p,q)
sp: = $false$;
τ_{EU}: =$\lambda Z.q \vee p \wedge EX(T, Z)$;
return FP($_{\tau_{EU}}$ sp);

3. Incremental CTL Model Checking

Baseline과 Incremental 모델 검증을 구별하기 위해 우리는 Baseline을 "^{o}'으로 표시 할 것이다. f는 K의 semantic이다. 추가로 $\Delta^+ f = f - f^o$ 이고 $\Delta^- f = f^o - f$ 이다.

Sub(f)는 f의 sub-formulae의 집합이다. 우리는 $\forall g \in$ Sub(f), g^o를 이미 안다고 가정한다. K^o, g^o, $\forall g \in$Sub(f)는 과거 지식이다.

K와 f가 주어졌다고 할 때 f는 MC알고리즘에 의해 바로 구해질 수 있다. 본 논문의 접근법 IMC는 과거 정보를 이용하여 MC에 비해 더 효율적으로 계산된다.

알고리즘 IMC(K, f)는 EG, EU대신 IEG, IEU알고리즘을 사용한다는 것만 다르다.

Algorithm IEG(T,p)
sp =$f^+ = f^o \vee \delta^+$;
τ_{EG}: =$\lambda Z.p \wedge EX(T, Z)$;
return FP($_{\tau_{EG}}$ sp);

Algorithm IEU(T,p,q)
sp =$f^- = f^o - \delta^-$;
τ_{EU}: =$\lambda Z.q \vee p \wedge EX(T, Z)$;
return FP($_{\tau_{EU}}$ sp);

δ^+=EU (T,p,$\Delta^+ p \vee$TO ($\Delta^+ T$)$\wedge p^o$),
δ^-=EU (T^o,p^o,$\Delta^- q \vee$FROM ($\Delta^- T$)$\wedge f^o \vee \Delta^- p \wedge f^o$)
FROM($T(\bar{x}, \bar{y})$) = $\exists \bar{y}. T(\bar{x}, \bar{y})$
TO($T(\bar{x}, \bar{y})$) = $\exists \bar{x}. T(\bar{x}, \bar{y})$
f^+는 "above estimation"이고, f^-는"below estimation"이다.

정리1. f = EG(T,p) = IEG(T,p), where f = EGp
증명. EG와 IEG는 각기 다른 시작점을 가지고 $_{\tau_{EG}}$의 greatest fixpoint를 계산한다. $_{\tau_{EG}}$가 단조 감소할 때 IEG가 f로 수렴한다는 것은 $f \subseteq f^+$를 보이면 된다.

$s_1 \in f - f^o$ 를 고려해보자. $\forall i, s_i \in p, (s_i, s_{i+1}) \in T^o$이고 $s_1 \in f$일 때 $T:p$ 경로 $\pi = s_1, s_2, \ldots, s_i, \ldots$가 존재한다. $s_1 \notin f$일 때, π는 $T^o:p^o$ 경로가 아니다. 따라서 다음 두 경우 중 한 가지가 참이 되어야 한다.

1. $s_i \notin p^o$
π가 $T:p$ 경로이다. 따라서 $s_i \in$p이고 $s_i \in \Delta^+ p$이다. s_1부터 s_i까지 π의 영역은 $T:p \rightarrow \Delta^+ p$ 영역이다. 따라서 $s_1 \in$ EU(T,p, $\Delta^+ p$)이다.

2. $(s_i, s_{i+1}) \notin T^o$
π는 $T:p$ 경로 이다. 따라서 $(s_i, s_{i+1}) \in T$, $s_{i+1} \in p$이다. $(s_i, s_{i+1}) \in T$에 따라 우리는 $(s_i, s_{i+1}) \in \Delta^+ T$를 얻을 수 있다. $s_{i+1} \in p$에따라 우리는 $s_{i+1} \in p^o$또는 $s_{i+1} \in \Delta^+ p$를 얻을 수 있다. 만약 $s_{i+1} \in p^o$이면 $s_{i+1} \in$TO ($\Delta^+ T$)$\wedge p^o$ 이다. s_1에서 s_{i+1}까지 π 영역은 $T:p \rightarrow$TO ($\Delta^+ T$)$\wedge p^o$ 영역이다. 따라서 $s_1 \in EU(T, p,$TO ($\Delta^+ T$)$\wedge p^o$)이다.

$s_{i+1}\in\Delta^+p$이라면 s_1에서 s_{i+1}인 π의 영역이 $T:p\rightarrow\Delta^+p$영역이다. 따라서 $s_1\in EU(T,p,\Delta^+p)$이다.
1과 2에따라 $\forall s_1\in f-f^o, s_1\in EU(T,p,\Delta^+p)\vee TO(\Delta^+T)\wedge p^o)=\delta^+$ 따라서 $f\subseteq f^+$이다.□□

정리 2. f=EU(T,p,q)=IEU(T,p,q), where f = EpUq
증명. 증명1에서와 같은 방법으로 $f^-\subseteq f$를 보이면 된다.
임의의 상태 $s_1\in f^o-f$에서, $s_n\in q^o$, $\forall i<n, s_i\in p^o$, $s_1\in f^o$일 때, $T^\wedge:p^o\rightarrow q^o$ 영역 θ= $s_1,s_2,\ldots,s_n$가 존재한다.
$s_1\notin f$일때, θ는 $T:p\rightarrow q$영역이 아니다. 따라서 s_i가 아래 세 조건에서 첫 상태일 때, 세 가지 조건을 생각할 수 있다.
1. $i<n$일 때 $s_i\notin p$이다.
θ는 $T^\wedge:p^o\rightarrow q^o$영역이다. 따라서 $s_i\in p^o$이고 $s_i\in f^o$이다. $s_i\in p^o$에 따라 우리는 $s_i\in\Delta^-p$임을 알 수 있다. 또 $s_i\in f^o$이므로 $s_i\in\Delta^-p\wedge f^o$이다.
s_1에서 s_i까지의 θ의 영역은 $T^\wedge:p^o\rightarrow(\Delta^-p\wedge f^o)$이다. 따라서 $s_1\in EU(T^o,p^o,\Delta^-p\wedge f^o)$이다.
2. $i<n$일 때 $(s_i,s_{i+1})\notin T^o$이다.
θ는 $T^\wedge:p^o\rightarrow q^o$영역이다. 따라서 $(s_i,s_{i+1})\in T^o$, $s_i\in f^o$ 이 것은 $(s_i,s_{i+1})\in\Delta^-T$, $s_i\in$FROM (Δ^-T)이다. 따라서 $s_i\in$FROM $(\Delta^-T)\wedge f^o$ s_1에서 s_i까지의 θ의 영역은 $T^\wedge:p^o\rightarrow$(FROM $(\Delta^-T)\wedge f^o$)이다.
따라서 $s_1\in EU(T^o,p^o,$FROM $(\Delta^-T)\wedge f^o)$이다.
3. $i=n$일 때, $s_i\notin q^o$이다.
θ는 $T^\wedge:p^o\rightarrow q^o$영역이다. 따라서 $s_n\in q^o$이고 $s_n\in\Delta^-q$이다. 따라서 θ는 $T^\wedge:p^o\rightarrow\Delta^-q$영역이다. 따라서 $s_1\in EU(T^o,p^o,\Delta^-q)$이다.
1,2,3에 의해 우리는 $\forall s_1\in f^o-f, s_1\in EU(T^o,p^o,\Delta^-q\vee$ FROM $(\Delta^-T)\wedge f^o\vee\Delta^-p\wedge f^o)=\delta^-$ 따라서 $f^-\subseteq f^o$이다.□□

정리3. MC(K,f) = IMC(K,f)
정리1과 정리2에 의해 정리 3은 참이다.□□

4. 실험

Carnegie Mellon Univ.(CMU) Symbolic Model Verifier(SMV)에 있는 동기 디지털 카운터를 통해 실험을 해봤다. 그 결과 보통의 경우에 Kripke 구조가 바뀌었을 때 IMC가 MC에 비해 크게 시간 비용을 줄일 수 있었다. 하지만 Kripke 구조가 완전히 바뀌었을 때에는 IEG, IEU에서 sp를 계산하는 비용이 IEG나 IEU를 사용하여 얻는 시간 비용보다 크므로 오히려 MC가 시간 비용이 작았다.

5. 결론 및 향후 연구

본 논문에서는 그동안 완전히 증명 되지 않은 알고리즘인 "above estimation"과 "below estimation" 을 완전히 증명하였고 간단한 실험으로 많이 변하지 않는 모델에서 알고리즘의 시간 복잡도가 기존의 알고리즘에 비해 낮다는 것 을 보였다.
이제는 복잡도가 크고 다양한 유비쿼터스 컴퓨팅 모델을 MC와 IMC로 모델 검증을 하여 상황에 따라 어떤 시간 비용이 요구되는지 이론적으로, 실험적으로 완전히 검증하는 것이 필요하다.
본 논문의 접근법을 Linear Temporal Logic(LTL)에 적용한다면 LTL에서의 연산 비용을 줄일 수 있을 것이라 생각된다.

Acknowledgment

이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2004-003-D00285).

참고 문헌

[1] O.Sokolsky and S.smolka. Incremental Model Checking in the Modal Mu-Calculus. In Proceedings of the 6th International Conference on Computer Aided Verification, LNCS 818:351-363, 1994.
[2] G. Swamy. Incremental Methods for Formal Verification and Logic Synthesis. PhD thesis, University of California at Berkeley, 1996. UMI publication 9723211
[3] R. Cleaveland, Tableau-based model checking in the propositional mu-calculus. Acta Informatica, 27(9):725-747, 1990.
[4] E. Clarke, E. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, In proceedings of the Workshop on Logic of Programs, LNCS 131:52-71, 1981