모델체커를 검증하기 위한 모델체커*

어현준 Nikolay V. Shilov 이광근†

한국과학기술원 프로그램 분석 시스템 연구단

Model Checker for Validating Other Model Checkers

Hyunjun Eo, Nikolay V. Shilov, and Kwangkeun Yi

Research On Program Analysis System, KAIST

요 약

모델체커(model checker)의 정확성을 판단하기 위한 방법으로 이미 엄밀하게 검증된 믿을만한 모델체커의 결과를 바탕으로 한 테스팅의 방법이 있을 수 있다. 우리가 구현한 모델체커는 게임 이론을 바탕으로하여 설계되었기 때문에 그 정확성을 쉽게 증명할 수 있고, 프로그램의 의미가 명확하게 드러나 있기 때문에 믿을 만한(reliable) 모델체커이다. 이 모델체커가 주는 결과들은 다른 모델체커들의 정확성을 판단하는 기준이 될 수 있을 것이다.

1 서론

모델체커(model checker)는 하드웨어나 소프트웨어 시스템의 정확성을 검증하는 중요한 도구중의 하나이다. 따라서, 모델체커 자체의 정확성은 그 모델체커를 사용하여 검증된 시스템의 정확성에 영향을 미칠 수 있는 중요한 문제이다.

일반적으로, 모델체커의 정확성을 입증하는 방법에는 테스팅(testing)의 방법과 정리증명(theorem proving)을 이용한 엄밀한 검증(formal verification)의 두가지 방법을 생각해 볼 수 있다. 그러나, 테스트 데이타를 수동으로 만들어 내는 것은 쉬운 일이 아니다. 또한 정리증명을 이용한 엄밀한 검증은 많은 수의 전문가를 필요로 하고 오랜 시간을 요하는 작업이다.

우리가 제시하는 방법은 엄밀하게 검증된 믿을 만한 모델체커를 만들고 그 모델체커로부터 테스트 데이타를 자동으로 생성하여 다른 모델체커들의 정확성을 테스트하는 데 사용할 수 있도록 하는 것이다.

게임을 이용하여 만든 모델체커는 쉽게 그 정확성을 증명할 수 있는 장점이 있다. 모델체커의 입력으로 주어지는 모델과 검증하고자 하는 성질을 기술한 논리식을 바탕으로 모델체킹게임을 만들어 그 게임을 풀어가는 방법으로 모델을 검사할 수 있다. 이렇게 만들어진 모델체커는 간단하고, 명확하기때문에 쉽게 검증 가능하다[1].

이 논문에서는 게임을 이용한 모델체커를 소개하고, 그 모델체커가 테스트 데이타를 만들어낼 수 있을 만큼 효율적임을 실험을 통해 보이고자 한다.

2 μ-계산법 : μC

μC는 명제논리(propositional logic)에 실행 기호(action symbol)에 연관된 양상 $[a]$, $\langle a\rangle$, 부동점(fixpoint) 연산 μ, ν를 확장한 것이다.

μC의 문법구조(syntax)는 다음과 같다.

$$\phi \quad ::= \quad p \mid \neg\phi \mid \phi\wedge\phi \mid \phi\vee\phi \mid [a]\phi \mid \langle a\rangle\phi \mid \mu p.\phi \mid \nu p.\phi$$

μC 논리식의 의미(semantics)는 모델 $M = (D_M, I_M)$에 의해 정의된다. 도메인 D_M의 원소들은 모델이 가질 수 있는 상태(state)들을 의미한다. 해석 $I_M(p)$는 명제변수 p에 대한 해석으로 p가 참이되는 상태들의 집합을 의미하고, $I_M(a)$는 실행기호 a에 대한 해석으로 a의 실행으로인해 변화되는 전후 상태들의 관계를 의미한다.

모델 M에서의 논리식 ϕ의 의미 $M(\phi)$는 모델 M에서 ϕ가 참이되는 상태들의 집합으로, 다음과 같이 귀납적으로 정의된다.

$$\begin{array}{ll} M(p) = I_M(p) & M(\neg\phi) = D_M \backslash M(\phi) \\ M(\phi_1\wedge\phi_2) = M(\phi_1)\cap M(\phi_2) & M(\phi_1\vee\phi_2) = M(\phi_1)\cup M(\phi_2) \\ \multicolumn{2}{l}{M([a]\phi) = \{s\in D_M \mid \exists s'\in D_M.s \xrightarrow{a} s' \wedge s'\in M(\phi)\}} \\ \multicolumn{2}{l}{M(\langle a\rangle\phi) = \{s\in D_M \mid \forall s'\in D_M.s \xrightarrow{a} s' \Rightarrow s'\in M(\phi)\}} \\ \multicolumn{2}{l}{M(\mu x.\phi) = \cap\{S\subseteq D_M \mid M_{S/x}(\phi)\subseteq S\}} \\ \multicolumn{2}{l}{M(\nu x.\phi) = \cup\{S\subseteq D_M \mid S\subseteq M_{S/x}(\phi)\}} \end{array}$$

$true$와 $false$는 $\nu x.x$와 $\mu x.x$로 표현될 수 있다.

3 게임

A, B 두 사람에 의한 게임 G는 다음의 튜플(tuple) $(P_A, P_B, M_A, M_B, F_A, F_B)$로 정의되어진다.

- $P_{A,B}$: A와 B의 위치(position), $P_A\cap P_B = \emptyset$
- $M_{A,B} \subseteq (P_{A,B}\backslash(F_A\cup F_B))\times(P_A\cup P_B)$: A와 B의 움직임(move)
- $F_{A,B} \subseteq (P_A\cup P_B)$: A와 B의 승리위치(winning position)

게임은 한 시점에 하나의 위치에서 진행되어진다. 그 위치가 $P_A(P_B)$에 해당하면 $A(B)$는 $M_A(M_B)$에 따라 게임을 다음 위치로 진행시킬 수 있다. 만약, 게임이 $F_A(F_B)$에 이르게 되면 $A(B)$가 그 게임에서 승리하게 된다. $A(B)$의 필승 전략(winning strategy)이란 언제나 $A(B)$가 승리할 수 있게 이끌어낼 수 있는 $A(B)$의 움직임이다.

유한(finite) 게임 G는 다음과 같이 유한 모델 M_G로 나타내어질 수 있다.

*본 연구는 과학기술부 창의적연구진흥사업 추진으로 얻어진 결과임.

†e-mail: {poisson; shilov; kwang}@ropas.kaist.ac.kr

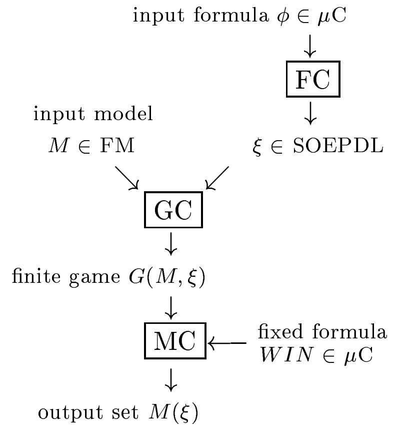

그림 1: 모델체커 Winner

- $D_M = P_A \cup P_B$
- $I_M(a) = M_A$ and $I_M(b) = M_B$
- $I_M(p) = F_A$ and $I_M(q) = F_B$

명제 1 *두 사람 A, B에 의한 모든 유한 게임 G에서 A가 필승 전략을 가진다는 것은 μC 논리식 $WIN \equiv \mu x.(p \vee \langle a\rangle x \vee (\langle b\rangle true \wedge [b]x)$ 가 만족한다는 것과 같다.*

4 SOEPDL

SOEPDL(Second Order Elementary Propositional Dynamic Logic)은 EPDL에 양상(modality) $\Box$, $\Diamond$와 이차한정기호(second order quantifier) $\forall$, $\exists$를 확장한 것으로 다음의 문법구조와 의미구조를 가진다.

$$\begin{array}{lll} \phi & ::= & true \mid false \mid p \mid \neg\phi \mid \phi\wedge\phi \mid \phi\vee\phi \\ & \mid & [a]\phi \mid \langle a\rangle\phi \mid \Box\phi \mid \Diamond\phi \mid \forall p.\phi \mid \exists p.\phi \end{array}$$

$$\begin{array}{lll} M(\Box\phi) & = & D_M \text{ if } \forall s' \in D_M.s' \in M(\phi) \\ M(\Diamond\phi) & = & D_M \text{ if } \exists s' \in D_M.s' \in M(\phi) \\ M(\forall x.\phi) & = & \{s \in D_M \mid \forall S \subseteq D_M.s \in M_{S/x}(\phi)\} \\ M(\exists x.\phi) & = & \{s \in D_M \mid \exists S \subseteq D_M.s \in M_{S/x}(\phi)\} \end{array}$$

명제 2 *모든 μC 논리식은 다음과 같이 SOEPDL 논리식으로 표현가능하다.*

$$\mu x.\phi \leftrightarrow \forall x.(\Box(\phi \to x) \to x) \quad \nu x.\phi \leftrightarrow \exists x.(\Box(x \to \phi) \wedge x)$$

5 모델체커 Winner

모델체커 Winner는 그림1과 같이 FC, GC, MC의 세부분으로 이루어진다.

- 논리식 컴파일러(formula compiler) FC는 입력으로 주어진 μC 논리식을 명제 2에서와 같이 SOEPDL 논리식으로 변환한다. 변환된 SOEPDL 논리식은 드모르간의 법칙에 따라 역($\neg$)을 최대한 논리식 안쪽으로 파고들게 하여 역($\neg$)은 명제변수 앞에만 붙도록 한다.

$$\begin{array}{ll} (\neg(\neg\phi)) \leftrightarrow \phi & \\ (\neg(\phi\wedge\psi)) \leftrightarrow ((\neg\phi)\wedge(\neg\psi)) & (\neg(\langle a\rangle\phi)) \leftrightarrow ([a](\neg\phi)) \\ (\neg(\Diamond\phi)) \leftrightarrow (\Box(\neg\phi)) & (\neg(\exists p.\phi)) \leftrightarrow (\forall p.(\neg\phi)) \end{array}$$

```
BEGIN
  x := EMPTY; y := wna;
  WHILE x<>y DO
    x := y;
    y := x U (<mva> x) U ((<mvb> all) & ([mvb] x));
  OD
END
```

그림 2: 논리식 WIN에 특화된 μC 모델체커 MC

- 게임 컴파일러(game compiler) GC는 입력 모듈(M)과 변환된 입력 논리식(ξ)를 입력으로해서 모델체킹 게임 $G(M,\xi)$를 만들어낸다. 게임 $G(M,\xi)$의 위치(position)는 s가 M에서의 한 상태, $S_x,...,S_z$가 한정변수(bound variable) $x,...,z$에 대한 해석, ψ가 ξ의 하위논리식(subformula)일때 $(s,S_x,...,S_z,\psi)$가 된다. A는 상태 s, 해석 $S_x,...,S_z$에서 논리식 ψ가 참이되는 방향으로 게임을 진행하고, B는 그것을 방해하기 위해 게임을 진행한다. A와 B의 움직임은 다음과 같고,

$$\begin{array}{l} (s,S_x,...,S_z,(\psi_1 \frac{\vee}{\wedge} \psi_2)) \to (s,S_x,...,S_z,\psi_i) \ \ \forall i \in \{1,2\} \\ (s,S_x,...,S_z,(\frac{\langle a\rangle}{[a]}\psi)) \to (t,S_x,...,S_z,\psi) \ \ \forall (s,t) \in I_M(a) \\ (s,S_x,...,S_z,(\frac{\Diamond}{\Box}\psi)) \to (t,S_x,...,S_z,\psi) \ \ \forall t \in D_M \\ (s,S_x...S_y...S_z,(\frac{\exists}{\forall}y.\psi)) \to (s,S_x...S...S_z,\psi) \ \ \forall S \subseteq D_M \end{array}$$

A와 B의 승리위치는 다음과 같다.

$$\begin{array}{l} (s,S_x,...,S_z,\frac{true}{false}) \\ (s,S_x,...,S_z,p) \text{ where } s \frac{\in}{\notin} M(p) \\ (s,S_x,...,S_z,\neg p) \text{ where } s \frac{\notin}{\in} M(p) \\ (s,S_x,...,S_z,y) \text{ where } s \frac{\in}{\notin} S_y \\ (s,S_x,...,S_z,\neg y) \text{ where } s \frac{\notin}{\in} S_y \\ (S,S_x,...,S_z,\frac{[a]}{\langle a\rangle}) \text{ where } \{t \mid (s,t) \in I_M(a)\} = \emptyset \end{array}$$

- 모델체커(model checker) MC는 GC를 통해 만들어진 게임 $G(M,\xi)$에 대해서 필승전략 $WIN \equiv \mu x.(p \vee \langle a\rangle x \vee (\langle b\rangle true \wedge [b]x)$이 존재하는 위치들을 찾는다. 일반적인 모델체커에 비해 MC는 고정된 하나의 논리식 WIN에 대해서만 모델을 조사하면 되므로 그림 2와 같이 매우 간단하며 쉽게 검증 가능하다.

정리 1 *모든 유한 모델 $M = (D_M, I_M)$과 모든 SOEPDL 논리식 ξ에 대해 두 사람 A와 B의 유한 게임 $G(M,\xi)$가 존재하고, A가 위치 $(s,S_x,...,S_z,\psi)$에서 필승전략을 가진다는 것은 논리식 ψ가 상태 s에서 참이된다는 것과 동치이다. d가 M에 있는 상태들의 수, p가 ξ에 있는 부동점들의 수, f가 논리식 ξ의 크기라 할때, 이 게임은 $O(2^{d\times p} \times d \times f)$에 만들어질 수 있다.*

정리 2 *MC가 모든 유한 모델에 대해서 μC 논리식 WIN을 검사하는 모델체커라면, 이 MC는 그림 1과 같이 모든 유한 모델에 대해 모든 μC 논리식을 검사하는 모델체커로 사용가능하다.*

이 게임은 $d \times 2^{d\times p} \times f$만큼의 위치를 가진다. 따라서, 모델의 상태가 단지 10개이더라도 고정점이 2개, 논리식의 크기가 20인 논리식에 대해서 이 게임이 가질 수 있는 위치는 $10 \times 2^{20} \times 20 = 209715200$개가 되어 시간, 공간적인 면에서 매우 비효율적이다.

6 Winskel의 μ-계산법 : $\mu \mathrm{C}_W$

Winskel의 μ-계산법 $\mu\mathrm{C}_W$[2]은 다음의 문법구조를 가진다.

$$\phi ::= p \mid \neg\phi \mid \phi\wedge\phi \mid \phi\vee\phi \mid [a]\phi \mid \langle a\rangle\phi \mid \mu p\{u\}.\phi \mid \nu p\{u\}.\phi$$

일반적인 μC와 달리 $\mu\mathrm{C}_W$는 부동점 계산을 할 때, 현재까지의 계산 역사를 논리식에 가지고 다니면서 다음 계산에 사용한다. 모델 M에서의 $\mu\mathrm{C}_W$ 논리식 $\mu x\{u\}.\phi$와 $\mu x\{u\}.\phi$의 의미는 다음과 같다.

$$\begin{array}{rcl} M(\mu x\{u\}.\phi) & = & \cap\{S\subseteq D_M \mid M_{S/x}(\phi)\backslash u\subseteq S\} \\ M(\nu x\{u\}.\phi) & = & \cup\{S\subseteq D_M \mid S\subseteq u\cup M_{S/x}(\phi)\} \end{array}$$

7 모델체커 Revenger

SOEPDL을 이용한 게임이 비효율적이었던 가장 큰 이유는 항상 모든 한정변수(bound variable)들에 대한 해석을 가지고 다녀야 했기 때문이다.

$\mu\mathrm{C}_W$를 이용하면 변수에 대한 해석을 가지고 다니는 것이 아니라, 치환(substitution)을 사용하기 때문에 부동점이 서로 얽혀있지 않을 경우 효율적인 계산이 가능하다.

- FC는 입력으로 주어진 μC 논리식을 $\mu\mathrm{C}_W$ 논리식으로 변환한다. 크기가 f인 μC 논리식은 크기의 상한이 $O(2^{d\times p}\times f)$인 $\mu\mathrm{C}_W$ 논리식으로 변환된다.

- GC는 입력 모듈(M)과 변환된 입력 논리식(φ)를 입력으로해서 모델체킹게임 $G(M,\varphi)$를 만들어낸다. 게임 $G(M,\varphi)$의 위치는 s가 M에서의 한 상태, ψ가 φ의 하위논리식일때 (s,ψ)가 된다. A는 상태 s에서 논리식 ψ가 참이되는 방향으로 게임을 진행하고, B는 그것을 방해하려고 한다. A와 B의 움직임은 다음과 같고,

  $$\begin{array}{l} (s,(\psi_1 \frac{\vee}{\wedge} \psi_2) \to (s,\psi_i) \ \text{ for every } i\in\{1,2\} \\ (s,(\frac{\langle a\rangle}{[a]}\psi)\to(t,\psi) \ \text{ for every } (s,t)\in I_M(a) \\ (s,(\frac{\mu}{\nu}x\{u\}.\psi)\to(s,\psi(\frac{\mu}{\nu}x\{u+s\}.\psi)) \ \text{ iff } s\notin u \end{array}$$

  A와 B의 승리위치는 다음과 같다.

  $$\begin{array}{l} (s,\frac{true}{false}) \\ (s,p) \text{ where } s\frac{\in}{\notin}M(p) \\ (s,\neg p) \text{ where } s\frac{\notin}{\in}M(p) \\ (S,\frac{[a]}{\langle a\rangle}) \text{ where } \{t \mid (s,t)\in I_M(a)\}=\emptyset \\ (s,(\frac{\mu}{\nu}x\{u\}.\psi)) \text{ where } s\in u \end{array}$$

- MC는 Winner에서와 같다.

정리 3 *모든 유한 모델 $M=(D_M,I_M)$과 모든 μC 논리식 ϕ에 대해 두 사람 A와 B의 유한 게임 $G(M,\phi)$가 존재하고, A가 위치 (s,ψ)에서 필승전략을 가진다는 것은 논리식 ψ가 상태 s에서 참이된다는 것과 동치이다. 논리식 ϕ의 크기가 f, 부동점의 개수가 p, 모델의 상태가 d개라면, 이 게임은 $O(d\times 2^{d\times p}\times f)$에 만들어질 수 있다.*

정의 1 *부동점 $\mu p.\phi(\nu p.\phi)$가 부동점 $\nu q.\psi(\mu q.\psi)$의 하위논리식이고, 변수 q가 ϕ에서 자유롭게 사용되지 않는다면 부동점이 서로 얽혀있지 않다고 한다.*

		M_1		M_2	
		ϕ_1	ϕ_2	ϕ_1	ϕ_2
W	시간(FC)	0	0	0	0
	시간(MC)	33831	34336	26853	29915
	반복회수	12	12	9	10
	게임크기	8912896	9437184	8912896	9437184
R	시간(FC)	3864	2	3811	2
	시간(MC)	19521	34	16795	11
	반복회수	56	29	45	8
	게임크기	1564712	12272	1564712	12272

W:Winner, R:Revenger, 시간단위:초, 게임크기:위치개수

표 1: 실험 결과

CTL, LTL과 같은 시간논리(temporal logic)들은 부동점이 서로 얽혀있지 않은 μ-계산법의 하위논리(sublogic)라고 생각할 수 있다. 이렇게 부동점이 서로 얽혀있지 않을 경우에 $\mu\mathrm{C}_W$를 이용할 경우 다음정리와 같이 부동점의 수에 상관없이 효율적으로 모델체킹게임을 만들 수 있다.

정리 4 *상태가 d개인 모델 M과 부동점이 서로 얽혀있지 않고 크기가 f인 μC 논리식 ϕ에 대해, 모델체킹 게임은 부동점의 개수 p에 상관없이 $O(d\times 2^d\times f)$에 만들어진다.*

8 구현 및 실험결과

모델체커 Winner와 Revenger는 nML로 구현되었다. Winner와 Revenger의 크기는 600줄 정도의 nML코드로 이루어져 있으며, `Int32`, `Array`, `List`, `Set`, `Map`, `Hashtbl` 라이브러리를 사용하였다.

다음의 모델 M_1, M_2에 대해 ϕ_1, ϕ_2를 조사해 보았다.

$$\begin{array}{ccccccccc} & \bigcirc & & \bigcirc & \qquad & & \bigcirc & & \bigcirc \\ & \downarrow & & \updownarrow & & & \downarrow & & \updownarrow \\ & \bigcirc & \leftrightarrow & \textcircled{p} & & & \textcircled{p} & \leftrightarrow & \bigcirc \\ M_1 & \downarrow & & \downarrow & & M_2 & \downarrow & & \downarrow \\ & \textcircled{p} & \leftarrow & \bigcirc & & & \bigcirc & \leftarrow & \textcircled{p} \\ & \updownarrow & & \downarrow & & & \updownarrow & & \downarrow \\ & \bigcirc & & \bigcirc & & & \bigcirc & & \bigcirc \end{array}$$

$$\begin{array}{rcl} \phi_1 & = & \nu q.\big(p \ \wedge \ [a]\mu r.(q\vee[a]r)\big) \\ \phi_2 & = & p \ \wedge \ \nu q.\big([a]q \ \wedge \ \mu r.(p\vee[a]r)\big) \end{array}$$

논리식 ϕ_1은 두 부동점 νq와 μr이 서로 얽혀있는 경우이고, 논리식 ϕ_2는 그렇지 않은 경우이다. 논리식 ϕ_1과 ϕ_2는 동치관계이며 $M_1(\phi_1)=I_{M_1}(p)$ 이고, $M_2(\phi_1)=\emptyset$ 이다.

표 1에서 볼 수 있듯이, 부동점이 서로 얽혀있지 않은 ϕ_2의 경우 Revenger는 매우 긍정적인 결과를 보였다.

9 향후 계획

Winner와 Revenger의 정확성을 정리증명(theorem proving)을 통해서 검증하고, 다른 모델체커들의 정확성을 판단할 수 있는 테스트 데이타들을 만들어 낼 계획이다.

참고 자료

[1] Nikolay V. Shilov and Kwangkeun Yi. Easy-to-verify model checker for propositional *mu*-calculus in finite models. (draft), 2000.

[2] Glynn Winskel. A note on model checking the modal μ-calculus. *Theoretical Computer Science*, 83:157–167, 1991.