2003-

质量保证的属性分为"safety"和"liveness"属性。虽然前者说明不应该发生什么（或者换种说法，应该总是发生什么），后者说明最终应该发生的事情。

BMC 的基本想法是寻找一个反例来执行,其边界长度是整数 k 的倍数。如果没有发现错误,边界长度则会增加 k 直到发现错误(问题变得难以解决),或者达到一些预先已知的上界

BMC 的独特性质：①用户必须提供应探索的周期数的界限，这意味着如果界限不够高，该方法是不完整的。②它使用 SAT 技术，而不是 BDDs。

模型检测：

模型检测有三个基础特性：①自动化 ②检查的有限系统 ③用时间逻辑来说明系统属性

Kripke 结构 M 是四 M=（S，I，T，L），其中 S 是一组状态，I⊆S 是初始状态集，T⊆S×S 是过渡关系，L:S → P(A)是标记函数，其中 A 是原子命题集，P（A）表示 A 上的幂集。标记是将观察结果附加到系统上的一种方式：对于一个属于 S 的状态 s，集合 L(s)由包含在 s 中的原子命题构成。

抽象层次上每个进程有两个程序计数器位置 0 和 1，其中 1 代表临界段。它可以被编码为一个长度为 2 的二进制向量 s ∈ S = {0,1}2。 因此，S={0,1}2 是系统的状态集。

转换关系由几个可能的转换组成，按照以下两个规则:①除非当前状态已经是初始状态，否则下一个状态 s'是初始状态 00; ②初始状态可以来回转换到 01 和 10。 因此，转换关系 T⊆S2 = {0,1}4 可以用以下一组位串表示:

{0100, 1000, 1100, 0001, 0010}

根据 Kripke 结构的路径定义时态公式的形式语义。设π 是一个 Kripke 结构 M 的无限路径，设 f 是一个时间公式。 当 f 保持在π 上时，我们递归地定义π |= f：

$$
\begin{array}{lll}
\pi \models p & \text{iff} & p \in L(\pi(0)) \\
\pi \models \neg f & \text{iff} & \pi \not\models f \\
\pi \models f \wedge g & \text{iff} & \pi \models f \text{ and } \pi \models g \\
\pi \models \mathbf{X} f & \text{iff} & \pi_1 \models f \\
\pi \models \mathbf{G} f & \text{iff} & \pi_i \models f \text{ for all } i \geq 0 \\
\pi \models \mathbf{F} f & \text{iff} & \pi_i \models f \text{ for some } i \geq 0 \\
\pi \models f \mathbf{U} g & \text{iff} & \pi_i \models g \text{ for some } i \geq 0 \text{ and } \pi_j \models f \text{ for all } 0 \leq j < i \\
\pi \models f \mathbf{R} g & \text{iff} & \pi_i \models g \text{ if for all } j < i, \pi_j \not\models f
\end{array}
$$

BMC：

Definition 1. 对于 l≤k，当 T(π (k),π (l))且 π = u•vw 其中 u = (π (0),...,π (l −1))且 v = (π (l),...,π (k))[3]。如果存在 k ⩾ l ⩾ 0 我们称π 为一个 k-loop，其中π 是(k, l) -loop。

Definition 2 (Bounded Semantics for a Loop). 设 k⩾0,π 是 k-loop。当且仅当π |= f(π $|=_k$f)那么 LTL 公式 f 在边界为 k 的路径π 上是有效的。

Theorem 1. 设 f 为 LTL 公式，M 为 Kripke 结构。那么 M |= Ef 如果存在 k⩾0 可使 M|=kEf。

2015-

介绍：

①大多数优化技术只考虑 safety 属性，忽略了同样重要的 liveness 属性。 ②许多 BMC 优化技术直接重用现有的显式抽象技术，通过引入额外的控制变量将他们编码到 BMC 实例中，只得到了性能增强。 ③关于基于多核的 BMC 优化，经常发生的是单片 BMC 实例直接交付给内部支持多核计算的 SMT 求解器。但是核之间交流的通常与否会明显影响到整体表现。

对 BMC 优化来说，简单胜过复杂

关键思想是利用 SESE 遍历目标系统的状态空间，并在每个时间步骤(即执行深度)记住可执行的转换，直到用户指定的边界

BMC：

系统可以用 Kripke 结构表示，型如 M = (S，I，T，L)，S 是系统状态集，I 是初始状态集并 I ⊆ S，T ⊆ S × S 是转换关系（不完全），L 是标记函数。

定义 Kripke 结构 M，LTL 公式 f 以及界限 k，BMC 的基本思想是构造命题式 |[M, f, k]|，当且仅当在 k 步中有一 m 违反了 f 这命题式是可满足的。

[|M, f, k]|定义为|[M]|$_k$ ∧ |[¬ f]|$_k$

$$|[M]|_k \triangleq I(S_0) \wedge \bigwedge_{i=1}^{k} T_i(S_{i-1}, S_i)$$

$I(S_0)$ 是一个状态变量的谓词，定义初始状态 S_0；$T_i(S_{i-1}, S_i)$ 是 M 的转换关系，是一个命题公式。因此，|[M]|$_k$表示系统从初始状态到指定边界 k 的所有执行路径。

2008-

FSMActor：

四元组　Afsm = (Qfsm, Pfsm, Parafsm, Tfsm)

Qfsm 是状态集，其中具有参数(initial,true)的状态 q0 = {φ, Pq0, Paraq0,φ} ∈ Qfsm 为 FSMActor 的初始状态。

Pfsm 是端口集。

Parafsm 内部变量及其对应初始值的集合，包含元素形为(Vari, IniVali)。

Tfsm 是迁移集，Tfsm = {(ps, Parasd, pd)| ps ∈ Ps, pd ∈ Pd, qs = {φ, Ps, Paras, φ} ∈ Qfsm,

qd = {φ, Pd, Parad, φ} ∈ Qfsm}　qs∈Qfsm 是源状态，qd∈Qfsm 是目标状态。

迁移所满足性质

当警戒条件 expguard 满足时，迁移会触发；

输出行为 actoutput 指定了目标端口和值；

设定行为 actset 更新了内部变量的值。

初始化：preinitialize() 和 initialize()，每次执行调用一次

执行：　prefire(), fire(), 和 postfire()，每次 SR 导演执行一次

显示：　wrapup()，在每次执行的最后调用一次来产生结果

当我们将 FSMActor 转化为 Kripke 结构，所得的状态数量不能为|Qfsm|

guardExpression 决定了可能触发转换的变量值

setAction 决定了转换后内部变量值

转换 FSMActor 到 Kripke：

原子命题集

$$AP = \underbrace{(\cup_{q\in Q_{fsm}}[state_{fsm} == q])}_{\text{FSM 所有状态并集}} \cup \underbrace{(\cup_{v\in Para_{fsm}.name, val\in GVD(A_{fsm},v,span)}[v == val])}_{\text{FSM 变量 count 的并集}};$$

状态集

$$S = (\wedge_{q\in Q_{fsm}} 2^{\{[state_{fsm}==q]\}}) \wedge (\wedge_{v\in Para_{fsm}.name, val\in GVD(A_{fsm},v,span)} 2^{\{[v==val]\}})$$

2 的幂？

$$\begin{aligned}&([state_{fsm} == state_1], \ldots, [state_{fsm} == state_n],\\&[v_1 == val_{1,1}], \ldots, [v_1 == val_{1,|GVD(A_{fsm},v_1,span)|}],\\&\ \ldots\\&[v_{|Para_{fsm}|} == val_{|Para_{fsm}|,1}], \ldots,\\&[v_{|Para_{fsm}|} == val_{|Para_{fsm}|,|GVD(A_{fsm},v_{|Para_{fsm}|},span)|}])\end{aligned}$$

标记函数 L

L：S→S

$$\begin{aligned}&([state_{fsm} == state_1], \ldots, [state_{fsm} == state_{|Q_{fsm}|}],\\&[v_1 == val_{1,1}], \ldots, [v_1 == val_{1,|GVD(A_{fsm},v_1,span)|}],\\&\ \ldots\\&[v_{|Para_{fsm}|} == val_{|Para_{fsm}|,1}], \ldots,\\&[v_{|Para_{fsm}|} == val_{|Para_{fsm}|,|GVD(A_{fsm},v_{|Para_{fsm}|},span)|}])\end{aligned}$$

迁移关系 R

对于$\forall$ $r = (s, s0) \in S \times S$,若满足下述条件时，有 $R = R \cup \{r\}$

①状态 s 满足其中一个关于 $state_{fsm}$的原子，即

$\exists 1$ $i, 1 \leqslant i \leqslant |Q_{fsm}|$ ，s.t. 在 s 中,有[$state_{fsm}$ == $state_i$] == 1

②状态 s'满足其中一个关于 $state_{fsm}$的原子，即

$\exists 1$ $i', 1 \leqslant i' \leqslant |Q_{fsm}|$ ，s.t. 在 s'中,有[$state_{fsm}$ == $state_i$] == 1

③状态 s 中和 s'中所有变量都各自满足其中一个关于其取值的原子：

$\forall$ $j, 1 \leqslant j \leqslant |Para_{fsm}|$,$\exists 1$ $k, k', 1 \leqslant k, k' \leqslant |GVD(A_{fsm}, v_j, span)|$,s.t.

在 s 中,有[v_j == $val_{j,k}$] == 1;

在 s'中，有[v_j == $val_{j,k'}$] == 1,

④T_{fsm}中存在符合下述条件的迁移，即$\exists$ $t = (p_s, Para_{sd}, p_d) \in T_{fsm}$:

(a) $p_s \in P_s$, $p_d \in P_d$, $state_i = \{\varphi, Para_s, P_s, \varphi\} \in Q_{fsm}$,

$state_{i'} = \{\varphi, Para_d, P_d, \varphi\} \in Q_{fsm}$,

(b) $\forall$ $j, 1 \leqslant j \leqslant |Para_{fsm}|$, $exp_{guardj}(val_{j,k})$ == 1,and $act_{setj}(val_{j,k})$ == $val_{j,k'}$;

SR 模型转化：

假设有两个 FSMActor：$Afsm_1$ 和 $Afsm_2$

原子命题集 $AP = AP_1 \cup AP_2$

状态集 $S_{12} = S_1 \wedge S_2$

标记函数 $S_{12} \rightarrow S_{12}$

迁移关系 R 前提：相连接的两个端口的名字相同

都得满足 $state_{fsm1}$ 和 $state_{fsm2}$ 中的原子