中国科学院研究生院课程讲稿

§5 模型检测方法

模型检测主要有两类方法：一类基于状态的分析，另一类基于路径的分析。基于状态的分析的出发点是计算模型的哪些状态满足哪些性质。基于路径的分析的出发点是确定模型的每条路径是否满足模些性质。

§5.1 基于状态分析的模型检测

基于状态分析的模型检测适用于 CTL 性质。一个系统的正确性依赖于系统可达的状态的性质。每个 CTL 公式都是状态公式。给定一个模型，我们可以判断其中的任何状态是否满足这个公式。

§5.1.1 状态标号算法

由于 $\{\neg, \vee\}$ 构成命题逻辑联结词的完全集，我们只考虑这些命题逻辑联结词。由于 $\{EX, EG, EU\}$ 构成 CTL 模态算子的完全集，我们只考虑这些模态算子。

给定 AP 上的 Kripke 结构 $\langle S, R, I, L\rangle$ 和一个 CTL 公式 φ，该标号算法的主要思想是逐步扩充 L 使得 $L(s)$ 包含每个在 s 满足的 φ 的子公式。

假设每个状态是否满足 φ 的子公式已经计算过。那么为计算每个状态是否满足 φ 我们有以下情况。

（ 1 ）若 $\varphi = p$ 且 p 是命题，则对每个 s， p 是否在 $L(s)$ 已经有系统给定，无需计算。

（ 2 ）若 $\varphi = \neg\varphi_0$，则对每个 s，若 $\varphi_0 \notin L(s)$，则将 φ 加入 $L(s)$。

（ 3 ）若 $\varphi = \varphi_0 \vee \varphi_1$，则对每个 s，若 $\varphi_0 \in L(s)$ 或 $\varphi_1 \in L(s)$，则将 φ_0 加入 $L(s)$。

（ 4 ）若 $\varphi = EX\varphi_0$，则对每个 s，若存在 t 使得 $R(s,t)$ 且 $\varphi_0 \in L(t)$，则将 φ 加入 $L(s)$。

（ 5 ）若 $\varphi = E(\varphi_0 U \varphi_1)$，则

（ 5a ）对每个 s，若 $\varphi_1 \in L(s)$，则将 φ 加入 $L(s)$。

（ 5b ）对每个 s，若存在 t 使得 $R(s,t)$ 且 $\varphi \in L(t)$，且 $\varphi_0 \in L(s)$，则将 φ 加入 $L(s)$。

（ 5c ）重复 (5b) 直至 L 不起变化。

（ 6 ）若 $\varphi = EG\varphi_0$，则

（ 6a ）计算 $S' = \{s \mid \varphi_0 \in L(s)\}$。

（ 6b ）计算 $\langle S, R\rangle$ 的导出子图 $\langle S', R'\rangle$ 的非平凡强连通分图的顶点集合 S''，并对每个 $s \in S''$，将 φ 加入 $L(s)$。

（ 6c ）对每个 $s \in S'$，若存在 t 使得 $R(s,t)$ 且 $\varphi \in L(t)$，则将 φ 加入 $L(s)$。

（ 6d ）重复 (6c) 直至 L 不起变化。

然后我们可以直接根据已经计算出的 L 求得系统的初始状态是否满足 φ，即

$$M, s \models \varphi \quad \text{当且仅当} \quad \varphi \in L(s)。$$
$$M \models \varphi \quad \text{当且仅当} \quad \varphi \in \bigcap_{s\in I} L(s)。$$

§5.1.2 符号模型检测原理

符号模型检测是建立在对状态集合的操作上。主要是计算一个公式在哪些状态上成立。基于这个思路，一个公式可以看作是满足这个公式的系统状态的集合。一个系统满足某个状态公式，就是说系统的初始状态包含于这个公式所代表的状态集合。

给定一个 AP 上的 Kripke 结构 $M = \langle S, R, I, L\rangle$ 。定义函数 $ex : 2^S \to 2^S$ 如下：

$$ex(A) = \{s \mid (s, s') \in \Delta, s' \in A\}$$

用 $[[p]]$ 表示 M 中满足 p 的状态的集合，即 $[[p]] = \{s \mid M, s \models p\}$ 。我们分 6 种情况计算一个公式 p 在 M 中的哪些状态上成立。

(1) p 是原子命题,	则 $[[p]] = \{s \mid p \in L(s)\}$
(2) p 是 $\neg p_0$,	则 $[[p]] = \sim[[p_0]]$
(3) p 是 $p_0 \vee p_1$,	则 $[[p]] = [[p_0]] \cup [[p_1]]$
(4) p 是 EXp_0 ,	则 $[[p]] = ex([[p_0]])$
(5) p 是 EGp_0 ,	则 $[[p]] = \nu Z.([[p_0]] \cap ex(Z))$
(6) p 是 $E(p_0 U p_1)$,	则 $[[p]] = \mu Z.([[p_1]] \cup ([[p_0]] \cap ex(Z)))$

我们有

$$\begin{array}{ll} M, s \models \varphi & \text{当且仅当} \quad s \in [[\varphi]] \text{ 。} \\ M \models \varphi & \text{当且仅当} \quad I \subseteq [[\varphi]] \text{ 。} \end{array}$$

符号模型

记 $\Phi(V)$ 为自由命题变量集合是 V 的子集的命题逻辑公式的集合。若 V 为变量集合，则 V' 为变量集合，定义如下： $v' \in V'$ 当且仅当 $v \in V$ 。

Definition 5.1 给定一个命题集合 AP 。一个 AP 上的符号模型是一个四元组

$$\langle V, R, \Theta, N\rangle$$

其中 V 为命题变量集合， R 为 $V \cup V'$ 上的公式， Θ 为 V 上的公式， $N : AP \to \Phi(V)$ 为 AP 到 V 上公式集合 $\Phi(V)$ 的映射。

系统状态由变量 $x_1, ..., x_n$ 决定。一个系统状态用 n 元组 $(a_1, ..., a_n)$ 表示。 $(a_1, ..., a_n) \models q$ 当且仅当 $q^{a_1,...,a_n}_{x_1,...,x_n} = 1$ 。

系统状态的集合为 $\{0, 1\}^n$ 。迁移关系由 $R(x_1, ..., x_n, x'_1, ..., x'_n)$ 决定。系统的初始状态集合由 $\Theta(x_1, ..., x_n)$ 决定。设 $AP = \{p_1, ..., p_k\}$ 。满足 p_i 的状态由 $N(p_i)$ 决定。

路径

我们用 $s{\to}s'$ 表示存在从 s 到 s' 的迁移，即 $(s, s') \models R$ 。符号模型 $\langle V, R, \Theta, N\rangle$ 上的一条路径是状态集 $\{0, 1\}^n$ 上的一个无穷序列

$$s_0 s_1 s_2 ...$$

其中对任意 $i \geq 0$, $s_i{\to}s_{i+1}$ 。

运行

符号模型 $\langle V, R, \Theta, N\rangle$ 上的一次运行是该符号模型上的第一个状态满足 Θ 的一条路径。

符号模型与 Kripke 结构

给定符号模型 (V,R,Θ,N) 。该模型对应于 $AP=\{p_1,...,p_k\}$ 上的 Kripke 结构 $M=\langle S,\Delta,I,L\rangle$ 其中 S,Δ,I,L 定义如下：

$$
\begin{aligned}
&S=\{(a_1,...,a_n)\mid a_i\in\{0,1\}\}\\
&\Delta=\{((a_1,...,a_n),(a'_1,...,a'_n))\mid (a_1,...,a_n,a'_1,...,a'_n)\models R(V,V')\}\\
&I=\{(a_1,...,a_n))\mid (a_1,...,a_n)\models\Theta\}\\
&L((a_1,...,a_n))=\{p_i\in AP\mid (a_1,...,a_n)\models N(p_i)\}
\end{aligned}
$$

Kripke 结构与符号模型

给定 Kripke 结构 $M=\langle S,\Delta,I,L\rangle$ 。我们可以将 S 编号。若 S 有 $\le 2^n$ 个元素，则可以用 n 个命题 $x_1,...,x_n$ 来表示。 S 中的第 i 个元素记为 $s(i-1)$ 。

用 $(a_1,...,a_n)$ 表示 $x_1,...,x_n$ 取值的 n 元组。则该 n 元组表示 $s(a_1\cdot 2^n+\cdots+a_{n-1}\cdot 2+a_n)$ 。

一个 $x_1,...,x_n$ 上的公式表示一个集合，即满足该公式的 $x_1,...,x_n$ 取值的 n 元组的集合。

状态的转化关系的集合一样可以用公式表示。我们需引进 n 个新变量，记作 $x'_1,...,x'_n$ 。用 $(a_1,...,a_n,a'_1,...,a'_n)$ 表示 $x_1,...,x_n,x'_1,...,x'_n$ 取值的 $2n$ 元组。则该 $2n$ 元组表示 $(s(a_1\cdot 2^n+\cdots+a_{n-1}\cdot 2+a_n),s(a'_1\cdot 2^n+\cdots+a'_{n-1}\cdot 2+a'_n))\in\Delta$ 。

一个 $x_1,...,x_n,x'_1,...,x'_n$ 上的公式表示一个迁移集合，即满足该公式的 $a_1,...,a_n,a'_1,...,a'_n$ 取值的 $2n$ 元组的集合。

对于标号函数 $L:S\to 2^{AP}$ ，我们定义 $N:AP\to 2^S$ 如下： $s\in N(p)$ 当且仅当 $p\in L(s)$ 。则 L 和 N 作为系统模型中描述状态满足哪些基本命题的作用是等价的。由于 $N(p)$ 是一个状态集合， $N(p)$ 可以用一个公式表示。

给定 $AP=\{p_1,...,p_k\}$ 上的 Kripke 结构 $M=\langle S,\Delta,I,L\rangle$ 。该模型对应于符号模型 (V,R,Θ,N) ，其中 V,R,Θ,N 定义如下：

$$
\begin{aligned}
&V=\{a_1,...,a_n\}\\
&(a_1,...,a_n,a'_1,...,a'_n)\models R(V,V')\text{ 当且仅当 }(s(a_1,...,a_n),s(a'_1,...,a'_n))\in\Delta\\
&(a_1,...,a_n)\models\Theta\text{ 当且仅当 }s(a_1,...,a_n)\in I\\
&N(p_i)=\Theta_i\text{ 且 }\Theta_i\text{ 定义如下：}(a_1,...,a_n)\models\Theta_i\text{ 当且仅当 }p_i\in L(s(a_1,...,a_n))
\end{aligned}
$$

符号模型检测

一个 (V,R,Θ,N) 表示的系统是否满足一个时序逻辑公式，即系统所对应的 Kripke 结构 M 是否满足一个时序逻辑公式。

定义 $\exists x.\varphi=(\varphi_x^0\vee\varphi_x^1)$ 。定义函数 $ex:\Phi(V)\to\Phi(V)$ 如下：

$$ex(\varphi)=\exists x'_1...x'_n.(R(x_1,...,x_n,x'_1,...,x'_n)\wedge\varphi_{x_1,...,x_n}^{x'_1,...,x'_n})$$

用 $[[q]]$ 表示满足 q 的状态的集合。我们分 6 种情况计算一个公式 q 在 M 中的哪些状态上成立。

(1) $q=p_i$ 是原子命题，	则 $[[q]]=N(p_i)$
(2) q 是 $\neg q_0$ ，	则 $[[q]]=\neg[[q_0]]$
(3) q 是 $q_0\vee q_1$ ，	则 $[[q]]=[[q_0]]\vee[[q_1]]$
(4) q 是 EXq_0 ，	则 $[[q]]=ex([[q_0]])$
(5) q 是 EGq_0 ，	则 $[[q]]=\nu Z.([[q_0]]\wedge ex(Z))$
(6) q 是 $E(q_0Uq_1)$ ，	则 $[[q]]=\mu Z.([[q_1]]\vee([[q_0]]\wedge ex(Z)))$

我们有

$$M \models \varphi \quad \text{当且仅当} \quad \Theta \to [[\varphi]]$$

§5.1.3 二元决策图

Definition 5.2 给定一个变量集合 V 。一个 V 上的二元决策图 *(BDD)* 是一个四元组

$$\langle N, n_0, E, L\rangle$$

其中 N 为节点的集合； $E : N \to N \times N$ 为定义节点出边的偏函数，若 $E(n) = \langle n', n''\rangle$ ，则称 $\langle n, n'\rangle$ 为 n 的 *0* 边， $\langle n, n''\rangle$ 为 n 的 *1* 边； n_0 为图的初始点； $L : N \to V \cup \{0, 1\}$ 为点的标号函数，满足 $L(n) \in \{0, 1\}$ 当且仅当 $E(N)$ 没有定义。

有序二元决策图 (OBDD)

给定一个 V 上变量的排序 $x_1, ..., x_m$ ，即 $x_1 < x_2 < \cdots < x_m$ 。约定 $x_i < 0$ 且 $x_i < 1$ 。一个二元决策图 $\langle N, n_0, E, L\rangle$ 是有序的，当且仅当 $E(a) = \langle b, c\rangle$ 则 $L(a) < L(b)$ 且 $L(a) < L(c)$ 。

给定 V 上的一个公式 φ 。该公式可以用一个有序二元决策图 $\langle N, n_0, E, L\rangle$ 表示，其构造如下。

---

1. $N = \{n_0\}, L(n_0) = x_1, L'(n_0) = \varphi.$
2. 对任意 $n \in N$ 且 $E(n)$ 尚无定义，

若 $L(n) = x_i$ ，则在 N 中添加两个新的点 n', n'' 且令 $E(n) = \langle n', n''\rangle$ 。

若 $i < m$ ， 则令 $L(n') = L(n'') = x_{i+1}, L'(n') = (L'(n))^0_{x_i}, L'(n'') = (L'(n))^1_{x_i}$ 。

若 $i = m$ ， 则令 $L(n') = L'(n') = (L'(n))^0_{x_m}, L(n'') = L'(n'') = (L'(n))^1_{x_m}$ 。

若两个公式是等价的，则它们的这样构造的有序二元决策图是同构的。

最简有序二元决策图 (ROBDD)

以上构造的 OBDD 的结构和大小相当于一个真值表，有很多冗余部分。二元决策图化简的规则如下：

---

u 和 v 是叶节点， $L(u) = L(v)$ ，则合并这两个节点。

u 和 v 不是叶节点， $E(u) = E(v)$ 且 $L(u) = L(v)$ ，则合并这两个节点。

u 和 v 不是叶节点， $E(v) = \langle a, a\rangle$ ，则删除 v 点，将指向 v 的边 (或 n_0) 指到 a 。

将冗余部分用以上规则化简直到不能再简化时得到的决策图称为最简有序二元决策图。若两个公式是等价的，则它们的最简有序二元决策图是同构的。用最简有序二元决策图表示的公式的等价性容易判定。若一个公式为永真式，则该公式的 ROBDD 为只有一个叶节点的图且该叶节点的值为 1 。若一个公式为永假式，则该公式的 ROBDD 为只有一个叶节点的图且该叶节点的值为 0 。

变量排序

变量排序对 ROBDD 的大小有很大的影响,对于一些问题,不同的变量排序所产生的 ROBDD 的大小的区别是指数的。

设 $\varphi = \bigvee_{i=1}^{n}(x_{2i-1} \wedge x_{2i})$ 。给定变量的排序 $x_1, x_3, ..., x_{2n-1}, x_2, x_4, ..., x_{2n}$ 。则 φ 的 ROBDD 节点数为 2^{n+1} 。给定变量的排序 $x_1, x_2, ..., x_{2n}$ 。则 φ 的 ROBDD 节点数为 $2(n + 1)$ 。 由于寻找最佳排序是 NP 难的问题。可用启发式的算法计算一个较好的变量排序。

有序二元决策图和布尔函数

一个公式可以看成是一个布尔函数，因此一个有序二元决策图对应于一个布尔函数。

用 $F|_N$ 表示 F 将定义域限制在 N 的子函数。定义 $\pi_i(\langle a_0, a_1\rangle) = a_i$ 。用 E_x^i 表示满足以下性质的通过修改 E 得到的函数 E' ：

若 $L(\pi_0(E(n))) \neq x$ 且 $L(\pi_1(E(n))) \neq x$	则 $E'(n) = E(n)$
若 $L(\pi_0(E(n))) = x$	则 $\pi_0(E'(n)) = \pi_i(E(\pi_0(E(n))))$
若 $L(\pi_1(E(n))) = x$	则 $\pi_1(E'(n)) = \pi_i(E(\pi_1(E(n))))$

用 $f_{OBDD}(x_1, ..., x_n)$ 表示与布尔函数 $f(x_1, ..., x_n)$ 对应的有序二元决策图。设 $f_{OBDD}(x_1, ..., x_n) = \langle N, n_0, E, L\rangle$ 且 $x_1 < \cdots < x_n$ 。 则

$f_{OBDD}(0, x_2, ..., x_n) = \langle N', n_0', E', L'\rangle$ 其中

N' 是由 $\pi_0(E(n_0))$ 可达的所有节点的集合； $n_0' = \pi_0(E(n_0))$ ； $E' = E|_{N'}$ ； $L' = L|_{N'}$ 。

$f_{OBDD}(x_1, ..., x_{i-1}, 0, x_{i+1}, x_n) = \langle N', n_0, E', L'\rangle$ 其中

N' 是由 n_0 通过 $E_{x_i}^0$ 可达的所有节点的集合； $E' = E_{x_i}^0|_{N'}$ ； $L' = L|_{N'}$ 。

$f_{OBDD}(1, x_2, ..., x_n) = \langle N', n_0', E', L'\rangle$ 其中

N' 是由 $\pi_1(E(n_0))$ 可达的所有节点的集合； $n_0' = \pi_1(E(n_0))$ ； $E' = E|_{N'}$ ； $L' = L|_{N'}$ 。

$f_{OBDD}(x_1, ..., x_{i-1}, 1, x_{i+1}, x_n) = \langle N', n_0, E', L'\rangle$ 其中

N' 是由 n_0 通过 $E_{x_i}^1$ 可达的所有节点的集合； $E' = E_{x_i}^1|_{N'}$ ； $L' = L|_{N'}$ 。

量词消去

设 f, b, c 是 $OBDD$ 。设 n 是 $OBDD$ 的节点。用 $top(f)$ 代表 f 的顶点。用 $L(f)$ 代表 $L(top(f))$ 。用 $f = (n, b, c)$ 表示 $top(f) = n$ 且 $E(n) = \langle top(b), top(c)\rangle$ 。 设 f 为 OBDD 且其上变量的序为由小（根节点）到大（叶节点）。 则 $f|_{x=0}$ 和 $f|_{x=1}$ 的算法如下：

若 f 是叶节点，则 $f|_{x=0} = f$
若 $L(f) = x$ 且 $f = (n, s_0, s_1)$ ，则 $f|_{x=0} = s_0$
若 $L(f) \neq x$ 且 $f = (n, s_0, s_1)$ ，则 $f|_{x=0} = (n, s_0|_{x=0}, s_1|_{x=0})$
若 f 是叶节点，则 $f|_{x=1} = f$
若 $L(f) = x$ 且 $f = (n, s_0, s_1)$ ，则 $f|_{x=1} = s_1$
若 $L(f) \neq x$ 且 $f = (n, s_0, s_1)$ ，则 $f|_{x=1} = (n, s_0|_{x=1}, s_1|_{x=1})$

我们有 $\exists y_1 ... y_n . \exists x . f = \exists y_1 ... y_n . (f|_{x=0} \vee f|_{x=1})$ 。 OBDD 上的或运算以及其他运算定义如下。

OBDD 上的运算

若 $s = (n, s', s'')$ 且 $L(s) = x \in V$, 则将 s, s', s'' 分别表示为 $(\neg x \wedge s') \vee (x \wedge s''), s|_{x=0}, s|_{x=1}$ 。若 $L(s) = 0$ 或 $L(s) = 1$ ，则用 0 或 1 直接表示该节点。

命题逻辑连接符可以看成是 OBDD 上的运算。设 $\circ$ 为二元运算符， f 和 g 为 OBDD 且其上变量的序为由小（根节点）到大（叶节点）。则 $f \circ g$ 的算法如下：

若 f 和 g 都是叶节点，则 $f \circ g = L(f) \circ L(g)$
若 $L(f) = L(g) = x$ 则 $f \circ g = ((\neg x \wedge (f|_{x=0} \circ g|_{x=0}) \vee (x \wedge (f|_{x=1} \circ g|_{x=1}))$
若 $x = L(f) < L(g)$ 则 $f \circ g = ((\neg x \wedge (f|_{x=0} \circ g) \vee (x \wedge (f|_{x=1} \circ g))$
若 $L(f) > L(g) = y$ 则 $f \circ g = ((\neg y \wedge (f \circ g|_{y=0}) \vee (y \wedge (f \circ g|_{y=1}))$

常用的二元运算有 $\wedge, \vee, \rightarrow, \leftrightarrow$ 。一元运算 $\neg$ 原则上可用 $\rightarrow$ 实现，即 $\neg p \Leftrightarrow (p \rightarrow \text{false})$ ，其计算结果就是将叶结点的 0 和 1 的值互换。公式 φ_0 和 φ_1 是否等价可以根据 $\varphi_0 \leftrightarrow \varphi_1$ 的 ROBDD 是否为 1 来判断。

量词消去的优化策略

用 $\bar{x}$ 表示 $x_1,...,x_n$ 。则 $ex(\varphi)=\exists x'_1...x'_n.(R(\bar{x},\bar{x'})\wedge\varphi^{\bar{x'}}_{\bar{x}})$ 。

计算 $ex(\varphi)$ 的 ROBDD 需要进行量词消去，是符号模型检测中计算量很大的操作。我们应用等价变换，尽可能将存在量词作用于较小的子公式。迁移关系 $R(\bar{x},\bar{x'})$ 可能是一组公式的析取或合取。以下分别对这两种请况进行分析。

析取关系

设 $R(\bar{x},\bar{x'})=R_1(\bar{x},\bar{x'})\vee\cdots\vee R_m(\bar{x},\bar{x'})$ 。

我们应用等式 $\exists x.(p(x)\vee q(x))=\exists x.p(x)\vee\exists x.q(x))$ 来简化 ROBDD 的计算。我们有

$$\exists x'_1...x'_n.(R(\bar{x},\bar{x'})\wedge\varphi^{x'_1,...,x'_n}_{x_1,...,x_n})=\exists x'_1...x'_n.(R_1(\bar{x},\bar{x'})\wedge\varphi^{\bar{x'}}_{\bar{x}})\vee\cdots\vee\exists x'_1...x'_n.(R_n(\bar{x},\bar{x'})\wedge\varphi^{\bar{x'}}_{\bar{x}})$$

合取关系

设 $R(\bar{x},\bar{x'})=R_1(\bar{x},\bar{x'})\wedge\cdots\wedge R_m(\bar{x},\bar{x'})$ 。

若 x 不在 $p(x)$ 中出现，我们有 $\exists x.(p(x)\wedge q(x))=p(x)\wedge\exists x.q(x))$ 。我们应用该等式来简化 ROBDD 的计算。为方便起见，设 $R_0(\bar{x},\bar{x'})=\varphi^{x'_1,...,x'_n}_{x_1,...,x_n}$ 。则

$$\exists x'_1...x'_n.(R(\bar{x},\bar{x'})\wedge\varphi^{x'_1,...,x'_n}_{x_1,...,x_n})=\exists x'_1...x'_n.(R_0(\bar{x},\bar{x'})\wedge\cdots\wedge R_m(\bar{x},\bar{x'})$$

我们将 $x'_1,...,x'_n$ 分成 k 个不相交子集 $d_1,...,d_k$ ，将 $R_0,....,R_m$ 分成 k 个不相交非空子集 $D_1,...,D_k$ 。如果对于所有 $1\le i<j\le k$ ， d_j 中的变量不在 D_i 中出现，则

$$\exists x'_1...x'_n.(R_0(\bar{x},\bar{x'})\wedge\cdots\wedge R_m(\bar{x},\bar{x'})=\exists d_1.(D_1\wedge\exists d_2.(D_2\wedge\cdots\wedge\exists d_k.(D_k)\cdots))$$

其中 d_i 表示 d_i 中变量的列举， D_i 表示 D_i 中公式的合取。对于给定的公式集合 $\{R_0,...,R_m\}$ 和变量集合 $\{x'_1,...,x'_n\}$ ，我们可以依据不同的标准来计算 $d_1,...,d_k$ 和 $D_1,...,D_k$ 。原则上我们应该有外层的 d_i 越小而 D_i 越大越好。

§5.2 基于路径分析的模型检测

基于状态分析的模型检测适用于 PLTL 性质。一个系统可以看作是其可能运行的路径的集合。一个系统满足某个路径公式，就是说系统的所有运行都满足这个公式。设 $M=\langle S,\Delta,I,L\rangle$ 是 AP 上的标号 Kripke 结构， φ 是 AP 上的 PLTL 公式。

$$M\models\varphi \text{ 当且仅当 } [[M]]\subseteq[[\varphi]] \text{ 。}$$

§5.2.1 Kripke 结构与 Büchi 自动机

设 $M=\langle S,\Delta,I,L\rangle$ 是 AP 上的标号 Kripke 结构。定义

$\Sigma=2^{AP}$
$\Delta'=\{(s,a,s')\mid(s,s')\in\Delta,a=L(s)\}$
$F=S$

令 $A_M=(\Sigma,S,\Delta',I,F)$ 。则 $[[A_M]]=[[M]$ 。

称 A_M 为 AP 上的 Kripke 结构 $M=\langle S,\Delta,I,L\rangle$ 所对应的 Büchi 自动机。

§5.2.2 模型检测原理

由于 Kripke 结构和 PLTL 公式都可以转换成对应的 Büchi 自动机。模型检测问题可转化为两个自动机的语言的包含关系或一个自动机的语言是否为空的问题。用 A_φ 表示公式 φ 对应的 Büchi 自动机。 一个给定的系统 M 满足一个 AP 上的时序逻辑公式 φ，即：

$$M \models \varphi$$

即 $[[M]] \subseteq [[\varphi]]$
即 $[[M]] \cap ((2^{AP})^\omega \setminus [[\varphi]]) = \emptyset$
即 $[[M]] \cap [[\neg\varphi]] = \emptyset$
即 $[[A_M]] \cap [[A_{\neg\varphi}]] = \emptyset$
即 $A_M \cap A_{\neg\varphi}$ 的语言为空。模型检测问题转化为自动机非空问题。

双深度优先搜索算法

为简单起见，我们使用栈作为算法的基本数据结构。栈的三个基本操作为压栈、退栈和读取栈最上面的一个元素。设 W 为一个栈， q 为一个元素。这三个操作分别记作 push(q,W) 、pop(W) 、 top(W) 。记空栈为 [ ] 。 给定 Büchi 自动机 $\langle \Sigma, S, \Delta, I, F\rangle$ ，其语言是否为空可以用以下算法计算。定义 $\Delta(q) = \{s \mid \exists a \in \Sigma.(q, a, s) \in \Delta\}$ 为 q 的所有后继状态的集合。该算法由两个嵌套的深度优先搜索算法实现。

```
start()
{
  W = A = B = [ ];
  for each s ∈ I, if (s ∉ A) { push(s,A); push(s,W); dfs1(s); pop(W); }
  report("empty");
}
```

```
dfs1(q)
{
  for each s ∈ Δ(q), if (s ∉ A) { push(s,A); push(s,W); dfs1(s); pop(W); }
  if (q ∈ F) { push(s,B); dfs2(s); }
}
```

```
dfs2(q)
{
  for each s ∈ Δ(q), { if (s ∈ W) report("nonemp"); if (s ∉ B) { push(s,B); dfs2(s); } }
}
```

给定一个 Büchi 自动机，算法报告 "nonemp" 当且仅当该自动机的语言非空。该判定算法的状态搜索次数相对于系统规模是线性的。

动态模型检测原理

给定一个隐式迁移系统描述和一个 PLTL 性质描述。由于计算效率的问题，验证这样的一个问题并不先构造系统的 Kripke 结构。计算 $A_M \cap A_{\neg\varphi}$ 是否为空的算法可以在 $A_{\neg\varphi}$ 已经存在的情况下，边构造 A_M 边计算 $A_M \cap A_{\neg\varphi}$ 。一条有利的因素是在根据系统模型的高层描述构造新的系统状态时，可以判断这些新的状态是否与 $A_{\neg\varphi}$ 有共同部分，决定是否需要对这些状态进一步扩展。另外就是我们只要判定 $A_M \cap A_{\neg\varphi}$ 是否为空，因此找到一条满足 $A_M \cap A_{\neg\varphi}$ 的运行就可退出，不必一定等到完成 A_M 的构造。

踏步等价与偏序归约

两条无穷路径 π 和 π' 是踏步等价的当前仅当存在两条整数上的无穷序列 $0 = i_0 < i_1 < i_2 < \cdots$ 和 $0 = j_0 < j_1 < j_2 < \cdots$ 使得对任意 $k \geq 0$,

$$L(\pi_{i_k}) = L(\pi_{i_k+1}) = \cdots = L(\pi_{i_{k+1}-1}) = L(\pi'_{j_k}) = L(\pi'_{j_k+1}) = \cdots = L(\pi'_{j_{k+1}-1})$$

一个公式 φ 是踏步不变的当前仅当对任意踏步等价的 π 和 π' ,

$$\pi \models \varphi \Leftrightarrow \pi' \models \varphi$$

不出现 O 算子的 PLTL 公式的集合记为 $PLTL_{-X}$ 。该集合的公式都是踏步不变的。

设要验证的性质为 $\varphi \in PLTL_{-X}$ 且 φ 中的命题变量的集合为 AP' 。假设系统的运行集合包含 $\pi = s_0...s_is_{i+1}s_{i+2}...$ 和 $\pi' = s_0...s_is'_{i+1}s_{i+2}\cdots$ 。

如果 $L(s_i) \cap AP' = L(s_{i+1}) \cap AP'$ 且 $L(s'_{i+1}) \cap AP' = L(s_{i+2}) \cap AP'$, 则 $\pi \models \varphi$ 当且仅当 $\pi' \models \varphi$ 。因此对于验证系统是否满足 φ , 我们可以不考虑 π' 。

偏序归约的主要思想就是根据一些判断条件, 在动态模型检测的使用中避免类似于 s'_{i+1} 这样的状态的生成以及对 π' 的检查。

§5.3 限界模型检测

限界模型检测的主要思路是对参与检测的路径做限制, 如果在参与检测的路径长度较短时能知道一个公式是否满足, 则该方法有较好的效率。

这一节所有的公式都必须是 NNF 范式。若出现的公式不是 NNF 范式, 则将其看成是与其等价的公式的 NNF 范式。

§5.3.1 CTL 公式的限界模型检测

给定 AP 上的 Kripke 结构 $M = \langle S, R, I, L\rangle$ 。M 的 k 界模型, 简称 k 模型 $M_k = \langle S, P_k, I, L\rangle$ 其中 P_k 为 M 中长度为 $k+1$ 的路径的集合。设 $\pi \in P_k$ 。这样的路径称为 k 路径。定义

$$rs(\pi) = \bigvee_{i=0}^{|\pi|-2} \bigvee_{j=i+1}^{|\pi|-1} \pi_i = \pi_j$$

CTL 公式的限界语义

NNF 范式的 CTL 公式 φ 的限界语义 $M_k, s \models \varphi$ 如下:

$M_k, s \models p$	若 $p = \top$, 或 $p \in AP$ 且 $p \in L(s)$
$M_k, s \models \neg p$	若 $M_k, s \not\models p$
$M_k, s \models \varphi \vee \psi$	若 $M_k, s \models \varphi$ 或 $M_k, s \models \psi$
$M_k, s \models \varphi \wedge \psi$	若 $M_k, s \models \varphi$ 且 $M_k, s \models \psi$
$M_k, s \models A\varphi$	若对于所有 P_k 中以 s 为起点的路径 π ： $M_k, \pi \models \varphi$
$M_k, s \models E\varphi$	若存在 P_k 中以 s 为起点的路径 π ： $M_k, \pi \models \varphi$
$M_k, \pi \models X\varphi$	若 $k \geq 1$ 且 $M_k, \pi_1 \models \varphi$
$M_k, \pi \models G\psi$	若 $rs(\pi)$ 且 $\forall i \leq k, M_k, \pi_i \models \psi$
$M_k, \pi \models F\varphi$	若 $\exists i \leq k, M_k, \pi_i \models \varphi$
$M_k, \pi \models \varphi U \psi$	若 $\exists i \leq k, M_k, \pi_i \models \psi$ 且 $\forall j < i, M_k, \pi_j \models \varphi$
$M_k, \pi \models \varphi R \psi$	若 $\forall i \leq k$, 若 $\forall j < i, M_k, \pi_j \not\models \varphi$ 则 $M_k, \pi_i \models \psi$, 且若 $\forall i \leq k$,$M_k, \pi_i \not\models \varphi$ 则 $rs(\pi)$

限界语义的正确性和完备性

我们有

$$M, s \models \varphi \text{ 当且仅当存在 } k \geq 0 \text{ 使得 } M_k, s \models \varphi \text{ 。}$$

定义 $M_k \models \varphi$ 当且仅当

$$\text{对所有 } s \in I \text{ ，} \quad M_k, s \models \varphi \text{ 。}$$

我们有

$$M \models \varphi \text{ 当且仅当存在 } k \geq 0 \text{ 使得 } M_k \models \varphi \text{ 。}$$

限界模型检测算法

给定模型 M 和公式 φ 。限界模型检测算法如下：

(1) $k = 0$
(2) 若 $M_k \models \varphi$ 成立，则 $M \models \varphi$ ，算法结束
(3) 若存在 s 且 $M_k, s \models \neg\varphi$ 成立，则 $M \not\models \varphi$ ，算法结束
(4) $k = k + 1$ ，回到 (2)

注意：这里的 $\neg\varphi$ 代表与其等价的 CTL 公式的 NNF 范式。前面的性质保证了算法一定能够终止和获得正确答案。

§5.3.2 PLTL 公式的限界模型检测

对于 PLTL ，我们的关注是否存在一个运行满足给定的公式。对于 k 路径，我们需要考虑这条路径的终点是否有可能回到已经经过的点形成一个环。这样的环隐含一条无穷路径，可以当成无穷路径对待。定义

$$s(i, k, n) = [if\ (i < k)\ then\ (i + 1)\ else\ (n)]$$

PLTL 公式的限界语义

定义 $M_k, \pi \models_n^i \varphi$ 如下：

$M_k, \pi \models_n^i p$	若 $p = \top$, 或 $p \in AP$ 且 $p \in L(\pi_i)$
$M_k, \pi \models_n^i \neg p$	若 $M_k, \pi \not\models_n^i p$
$M_k, \pi \models_n^i \varphi \wedge \psi$	若 $M_k, \pi \models_n^i \varphi$ 且 $M_k, \pi \models_n^i \psi$
$M_k, \pi \models_n^i \varphi \vee \psi$	若 $M_k, \pi \models_n^i \varphi$ 或 $M_k, \pi \models_n^i \psi$
$M_k, \pi \models_n^i X\varphi$	若 $M_k, \pi \models_n^{s(i,k,n)} \varphi$
$M_k, \pi \models_n^i G\varphi$	若 $\forall j \in \{\min(i, n), ..., k\}, M_k, \pi \models_n^j \varphi$
$M_k, \pi \models_n^i \varphi U \psi$	若 $\exists j \in \{\min(i, n), ..., k\}, M_k, \pi \models_n^j \psi$ 且 $i \leq j \wedge \forall m \in \{i, ..., j-1\}, M_k, \pi \models_n^m \varphi$ 或 $i > j \wedge \forall m \in \{n, ..., j-1, i, ..., k\}, M_k, \pi \models_n^m \varphi$

定义 $M_k, \pi \models_n^{-1} \varphi$ 为永假。定义 $R(\pi_k, \pi_{-1})$ 为永真。 NNF 范式的 LTL 公式 φ 的限界语义 $M_k, \pi \models \varphi$ 如下：

$$M_k, \pi \models \varphi \text{ 当且仅当存在 } n \in \{-1, 0, ..., k\} \text{ ，使得 } R(\pi_k, \pi_n) \text{ 且 } M_k, \pi \models_n^0 \varphi \text{ 。}$$

限界语义的正确性和完备性

我们有

$$\exists \pi \in M, \pi_0 \in I.(M, \pi \models \varphi) \text{ 当且仅当存在 } k, \exists \pi \in M_k, \pi_0 \in I.(M_k, \pi \models \varphi) \text{ 。}$$

定义 $M_k \models \varphi$ 当且仅当

$$\text{不存在始于 } I \text{ 的路径 } \pi \in P_k \text{ 满足 } M_k, \pi \models \neg\varphi \text{ 。}$$

我们有

$$M \models \varphi \text{ 当且仅当 } \forall k \geq 0, M_k \models \varphi \text{ 。}$$

限界模型检测算法

我们有 $M_k \not\models \varphi$ 则对所有 $i \geq 0$, $M_{k+i} \not\models \varphi$ 。定义 (M, φ) 的完备阈值 $ct(M, \varphi)$ 为最小的满足以上条件的 k 。对任何 M 和 φ ，完备阈值 $ct(M, \varphi)$ 存在。进而我们有

$$M \models \varphi \text{ 当且仅当对 } k = ct(M, \varphi), M_k \models \varphi \text{ 。}$$

给定模型 M ， φ 和 $m \geq ct(M, \varphi)$ 。限界模型检测算法如下：

(1) $k = 0$
(2) 若 $M_k \not\models \varphi$ ，则 $M \not\models \varphi$ ，算法结束
(3) 若 $k = m$ ，则 $M \models \varphi$ ，算法结束
(4) $k = k + 1$ ，回到 (2)

计算 $M_k \not\models \varphi$ 即根据限界语义计算是否存在初始状态出发的 k 路径 π 使得 $M_k, \pi \models \neg\varphi$ 。假定我们通过估算能够正确地获得 $m \geq ct(M, \varphi)$ ，前面的性质保证了算法一定能够终止和获得正确答案。

§5.4 公平性

本章前面提到的系统模型以 Kripke 结构为基础，并不能体现公平性约束。一般而言，公平性可以用状态集合或用状态集合的二元组表示，而状态集合可用公式表示。对有些特殊的公平性可以用特殊的申明表示。

关于给定公式集合的弱公平性

给定公式集合 $\{\varphi_1, ..., \varphi_n\}$ 。一个运行满足弱公平性当且仅当对每个 i ，该运行无限次满足 φ_i 。

关于给定公式二元组集合的强公平性

给定公式二元组集合 $\{(\varphi_1, \psi_1), ..., (\varphi_n, \psi_n)\}$ 。一个运行满足强公平性当且仅当对每个 i ，该运行如果从某状态起无限次地满足 φ_i ，则在该状态后有状态满足 ψ_i 。

关于给定公式二元组集合的弱公平性

给定公式二元组集合 $\{(\varphi_1, \psi_1), ..., (\varphi_n, \psi_n)\}$ 。一个运行满足弱公平性当且仅当对每个 i ，该运行如果从某状态起总是满足 φ_i ，则在该状态后有状态满足 ψ_i 。这个弱公平性等价于关于公式集合 $\{(\neg\varphi_1 \vee \psi_1), ..., (\neg\varphi_n \vee \psi_n)\}$ 的弱公平性。

关于进程和迁移的公平性

假设系统由若干进程组成，进程由若干迁移组成。我们可以申明

进程弱公平性：	若某进程总是可执行的，则该进程必须执行一次。
迁移弱公平性：	若某迁移总是可执行的，则该迁移必须执行一次。
进程强公平性：	若某进程总有可执行的时候，则该进程必须执行一次。
迁移强公平性：	若某迁移总有可执行的时候，则该迁移必须执行一次。

公平系统的模型检测

以上所述公平性在系统模型中具有一定的普遍性，很多系统模型描述方法与模型检测工具提供描述一种或多种以上所述公平性的方法，并实现相关模型检测算法。

§5.5 模型检测工具

基于 OBDD 的状态分析的模型检测的工具有 SMV 和 NuSMV 等。 SMV 最早由 CMU 大学开发，主要应用领域包括电子电路验证等方面。 基于路径分析的模型检测工具有 SPIN 等。 SPIN 由贝尔实验室开发，主要应用领域包括通信协议验证等方面。 基于限界语义的模型检测工具有 VERDS 等。 VERDS 由中国科学院软件研究所计算机科学国家重点实验室开发，是有穷状态并发迁移系统的模型检测与限界模型检测工具。

§5.6 说明

本章介绍模型检测方法和工具。符号模型检测和模型检测工具 SMV 可参考 [McM93] 。基于自动机原理的模型检测和模型检测工具 SPIN 可参考 [Hol90, Hol03] 。基于限界语义的模型检测和模型检测工具 VERDS 可参考 [BCCZ99, Zha09] 。

参考文献

[BCCZ99] A. Biere, A. Cimmatti, E. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. LNCS 1579:193-207. TACAS 99.

[McM93] K. L. McMillan. Symbolic Model Checking. Kluwer Publisher, 1993.

[Hol90] G. J. Holzmann. Design and Validation of Computer Protocols. New Jersey: Prentice Hall, 1990.

[Hol03] G. J. Holzmann. The SPIN Model Checker. Boston: Addison-Wesley, 2003.

[Zha09] W. Zhang. Bounded Semantics of CTL and SAT-based Verification. Lecture Notes in Computer Science 5885 (ICFEM 2009):286-305. Springer-Verlag. 2009.