PSL的有界模型检验

虞 蕾[1,2], 赵宗涛[2]
(1. 国防科学技术大学计算机学院博士后流动站, 湖南长沙 410073; 2. 第二炮兵工程学院计算机系, 陕西西安 710025)

摘 要: 基于SAT的有界模型检验被视为是基于OBDD的符号化模型检验技术的重要补充, 是并行反应式系统的一种有效验证方法. 然而, 直至现在, 有界模型检验已验证的属性逻辑还十分有限. PSL是一种用于描述并行系统的属性规约语言(IEEE-1850), 包括线性时序逻辑FL和分支时序逻辑OBE两部分. 通过模型检验可验证系统的PSL属性, 本文提出了PSL的有界模型检验方法及其算法框架. 首先, 定义PSL逻辑的有界语义, 而后, 将有界语义进一步简化为SAT, 分别将PSL性质规约公式和系统 M 的状态迁移关系转换为SAT命题公式, 最后验证上述两个SAT命题公式合取式的可满足性, 这样就将时序逻辑PSL的存在模型检验转化为一个命题公式的可满足性问题, 并用一个队列控制电路实例具体解释算法执行过程.

关键词: PSL (property specification language); 有界模型检验 (bounded model checking, BMC); SAT (propositional satisfiability); OBDD (ordered binary decision diagram)
中图分类号: TP301 文献标识码: A 文章编号: 0372-2112 (2009) 03-0614-08

Bounded Model Checking of PSL

YU Lei[1,2], ZHAO Zong-tao[2]
(1. *Postdoctoral Station of Computer School, National University of Defense Technology, Changsha, Hunan 410073, China;*
2. *Department of Computer Science, Second Artillery Engineering College, Xi'an, Shaanxi 710025, China*)

Abstract: SAT-based bounded model checking (BMC) is introduced as a complementary technique to OBDD-based symbolic model checking, and is a verification method for parallel and reactive systems. However, until now the properties verified by bounded model checking are very limited. Temporal logic PSL is a property specification language (IEEE-1850) describing parallel systems and is divided into two parts, i.e. the linear time logic FL (Foundation Language) and the branch time logic OBE (Optional Branching Extension). The specification checked by BMC is extended to PSL and its algorithm is also proposed. Firstly, define the bounded semantics of PSL, and then reduce the bounded semantics into SAT by translating PSL specification formula and the state transition relation of the system M into the propositional formulas, respectively. Finally, verify the satisfiability of the conjunction of the two propositional formulas. The algorithm results in the translation of the existential model checking of the temporal logic PSL into the satisfiability problem of propositional formula. An example of a queue controlling circuit is used to interpret detailedly the executing procedure of the algorithm.

Key words: PSL (property specification language); BMC (bounded model checking); SAT (propositional satisfiability); OBDD (ordered binary decision diagram)

1 引言

PSL[1]是由EDA界的标准化组织Accellera提出, 并于2005年确定为IEEE工业标准(IEEE-1850)的用于描述并发系统的属性规约语言. IBM的Sugar语言是形成PSL的基础[2,3]. PSL是一种分层语言, 包括4层: 布尔层(Boolean Layer)、时序层(Temporal Layer)、验证层(Verification Layer)、建模层(Modeling Layer). 时序层是语言的核心层, 用于实现语言的主要功能, 即描述设计模块的时序属性, 可分为FL(Foundation Language, 基础语言)和OBE(Optional Branching Extension, 可选分支扩展)两部分. FL是线性时序逻辑, 而OBE就是CTL, 是一种分支时序逻辑. 由于PSL易于读写, 语法精简, 语义清晰, 表达能力强, 因此作为一种功能描述语言得到了广泛的应用.

模型检验是形式化验证的常用方法[4,5], 在验证技

收稿日期: 2007-01-23; 修回日期: 2007-12-03
基金项目: 国家高技术研究发展计划(863计划)课题(No. 2007AA010301); 国家自然科学基金(No. 60503032); 中国博士后科学基金(No. 20080431401)

术方面, 主要利用 Tableau、自动机理论、OBDD[4] (Ordered Binary Decision Diagram, 排序的二进制决策图) 等技术. 从目前已有的文献看[6-9], PSL 的模型检验主要是采用自动机理论, 通过构造 PSL 的自动机来实现自动验证, 但构造自动机的过程需要引入 $|M|\cdot|\alpha|\cdot 2^{|\alpha|}$ 个状态 (M 为系统模型, α 为要验证的时序逻辑公式); 而 OBDD 验证技术主要用来符号化地表示状态集合和状态的迁移关系, OBDD 缺点是对系统的类型和大小非常敏感, 采用 OBDD 的模型检验技术可能会随着模型规模的扩大而产生状态的指数爆炸问题. Biere 等人[10] 于 1999 年提出一种新的符号化模型检验方法即有界模型检验 (Bounded Model Checking, BMC) 方法. 这种方法基于 SAT 技术, 其思想是: 已知一 Kripke 结构 M 和一个属性规约 f, 首先构造一个命题公式, 而后用 SAT 求解器验证此公式, 从初始状态开始, 若在 k 次以内的状态迁移可达的状态空间中使得命题公式不满足, 则寻找到使规约不成立的反例, 验证过程结束. SAT 过程不出现如 OBDD 的潜在的状态爆炸问题, 可以更快地产生反例, 所需的内存空间也较小, 并且不需要手工定义变量或费时的动态重排序, 因此基于 SAT 的 BMC 是基于 OBDD 符号化模型检验技术的一个重要补充. 然而, 直到现在, 有界模型检验已经验证的属性逻辑还十分有限, 许多文献[10,11] 中系统的规约 (specification) 仅局限于是 LTL. Penczek 等人[12] 在 2002 年提出 ACTL (CTL 的全称片断) 的 BMC 方法, 并实现了一个 BMC 工具. 而文献[13] 也仅将规约扩展为 ACTL*. 骆翔宇等人[14] 研究了时态认知逻辑的有界模型检验问题. 本文将有界模型检验验证的规约再次扩展到时序逻辑 PSL, 提出 PSL 的有界模型检验方法及其算法框架. 首先, 定义 PSL 逻辑的有界语义, 而后, 将有界语义进一步简化为 SAT, 分别将 PSL 性质规约公式 α、系统 M 的状态迁移关系转换为命题公式 $[[\alpha]]_{M_k}$ 和 $[[M,\alpha]]_k^s$, 最后验证命题公式 $[[M,\alpha]]_k^s \wedge [[\alpha]]_{M_k}$ 的可满足性, 这样就将时序逻辑 PSL 的存在模型检验转化为一个命题公式的可满足性问题, 并通过例举了一个队列控制电路系统实例具体解释算法执行过程.

2　Kripke 结构与 PSL (Property Specification Language) 逻辑

2.1　Kripke 结构[10]

令 AP 是原子公式集, AP 上定义的 Kripke 结构 M 是一个 4 元序偶 $M=(S,T,s_0,L)$. 其中, (1) S 是有限状态集; (2) s_0 是初始状态; (3) $T\subset S\times S$ 是完全的 (total) 迁移关系, 也即, 对于每个状态 $s\in S$, 总存在状态 $s'\in S$ 有 $T(s,s')$; (4) $L: S\to 2^{AP}$, 状态属性标志函数, 表征每个状态对应的指派为 true 的原子命题公式. 作以下定义:

•Kripke 结构 M 中的一条路径是指一个状态的无穷序列 $\pi=s_0s_1\cdots$, 对任一 $i\in N$, 有 $(s_i,s_{i+1})\in T$.

•Kripke 结构 M 中的一条有界路径是指一个状态的有穷序列 $\pi=s_0s_1\cdots s_k$, 对任一 $0\leqslant i\leqslant k-1$ 有 $(s_i,s_{i+1})\in T$.

•对一条路径 π, 用 $\pi(i)$ 表示 π 的第 i 个状态, 用 π^i 表示 π 从状态 $\pi(i)$ 开始的后缀.

•当 $l\leqslant k$ 时, 称一路径 π 是 (k,l) loop (循环) 的, 若 $(\pi(k),\pi(l))\in T$, 且 π 可表示成 $\pi=u\cdot v^{\omega}$, 其中 $u=(\pi(0),\cdots,\pi(l-1))$, $v=(\pi(l),\cdots,\pi(k))$; 称 π 为一 k-loop, 若存在一整数 l, $k\geqslant l\geqslant 0$, π 是一 (k,l) loop; 一条 k-path 是一条长为 k 的路径 $\pi_k=\{s_0,\cdots s_k\}$; $\mathrm{loop}(\pi_k):=\{l\mid(\pi(k),\pi(l))\in T\}$.

•M 的 k 模型是一个 4 元序偶 $M_k=(S,\mathrm{Path}_k,s_0,L)$, 其中 Path_k 是 M 的所有 k-path 的集合.

2.2　PSL 语法与有界语义

2.2.1　PSL 语法[1]

定义 1 (连续扩展正规表达式 **Sequential Extended Regular Expression, SERE**)

每个布尔表达式 b 是一个 SERE; 如果 r, r_1, r_2 是 SERE, 则以下各项都是 SERE:

•$\{r\}$　•$r_1;r_2$　•$r_1:r_2$　•$r_1|r_2$　•$r_1\&\&r_2$　•$[*0]$　•$r[*]$

定义 2 (基础语言公式 **Foundation Language Formulas, FL formulas**)

如果 b 是布尔表达式, 则 b 和 $b!$ 都是 FL formulas;

如果 φ 和 ψ 是 FL formulas, r, r_1, r_2 是 SEREs, b 是布尔表达式, 则以下各项都是 FL formulas:

•(φ)　•$\neg\varphi$　•$\varphi\wedge\psi$　•$r!$　•r　•$X!\varphi$

•$[\varphi U\psi]$　•φ abort b　•$r|\rightarrow\varphi$

定义 3 (可选分支扩展公式 **Optional Branching Extension Formulas, OBE formulas**)

每个布尔表达式是一个 OBE 公式; 如果 f, f_1, f_2 是 OBE 公式, 则以下是 OBE 公式:

•(f)　•$\mathrm{EX}f$　•$\neg f$　•$E[f_1Uf_2]$　•$f_1\wedge f_2$

•$\mathrm{EG}f$

定义 4 (**Accellera PSL 公式**)

每个 FL 公式都是一个 Accellera PSL 公式; 每个 OBE 公式都是一个 Accellera PSL 公式.

2.2.2　PSL 的有界语义

(1) 非时钟 SEREs 的语义

$w\models r$ 表示 w 紧满足 (tightly satisfies) r, 其中, w 是定义在 $\Sigma=2^p\cup\{\top,\bot\}$ 上的有穷字, b 为布尔表达式,

r, r_1, r_2 是非时钟 SEREs, 则:

(1) $w \models \{r\} \Leftrightarrow w \models r$

(2) $w \models b \Leftrightarrow |w| = 1$ 且 $w^0 \models b$

(3) $w \models r_1; r_2 \Leftrightarrow \exists w_1, w_2$, s.t. $w = w_1 w_2$, $w_1 \models r_1$ 且 $w_2 \models r_2$

(4) $w \models r_1 : r_2 \Leftrightarrow \exists w_1, w_2, l$, s.t. $w = w_1 l w_2$, $w_1 l \models r_1$ 且 $l w_2 \models r_2$

(5) $w \models r_1 | r_2 \Leftrightarrow w \models r_1$ 或 $w \models r_2$

(6) $w \models r_1 \&\& r_2 \Leftrightarrow w \models r_1$ 且 $w \models r_2$

(7) $w \models [* 0] \Leftrightarrow w = \varepsilon$

(8) $w \models r[*] \Leftrightarrow w \models [* 0]$ 或 $\exists w_1, w_2$, s.t. $w_1 \neq \varepsilon$, $w = w_1 w_2$, $w_1 \models r$ 且 $w_2 \models r[*]$

(2) 非时钟 FL 公式的语义

式 $w \models \varphi$ 表示 w 满足 φ, 其中, w 是一个有穷或无穷字, b 是布尔表达式, r, r_1, r_2 是非时钟 SEREs, φ 和 ψ 是非时钟 FL 公式, 则:

(1) $w \models (\varphi) \Leftrightarrow w \models \varphi$ (2) $w \models \neg \varphi \Leftrightarrow w \not\models \varphi$ (3) $w \models \varphi \wedge \psi \Leftrightarrow w \models \varphi$ 且 $w \models \psi$

(4) $w \models b! \Leftrightarrow |w| > 0$ 且 $w^0 \models b$ (5) $w \models b \Leftrightarrow |w| = 0$ 或 $w^0 \models b$ (6) $w \models r! \Leftrightarrow \exists j < |w|$, s.t. $w^{0..j} \models r$

(7) $w \models r \Leftrightarrow \forall j < |w|$, s.t. $w^{0..j} T^{\omega} \models r!$ (8) $w \models X! \varphi \Leftrightarrow |w| > 1$ 且 $w^{1..} \models \varphi$

(9) $w \models [\varphi U \psi] \Leftrightarrow \exists k < |w|$, s.t. $w^{k..} \models \psi$, 且 $\forall j < k$, $w^{j..} \models \varphi$

(10) $w \models \varphi$ abort $b \Leftrightarrow w \models \varphi$ 或 $\exists j < |w|$ s.t. $w^j \models b$ 且 $w^{0..j-1} T^{\omega} \models \varphi$

(11) $w \models r \overrightarrow{|} \varphi \Leftrightarrow \forall j < |w|$, s.t. $w^{0..j} \models r$, $w^{j..} \models \varphi$

(3) PSL 的分支时序逻辑 OBE 就是 CTL, 采用文献[4]所使用的标准语义.

(4) PSL 的有界语义

令 M_k 是一个 k 模型, $\alpha, \alpha_1, \alpha_2$ 是 PSL 公式, b 是布尔表达式, r, r_1, r_2 是非时钟 SEREs, φ 和 ψ 是非时钟 FL 公式, $M_k, [(\pi_k, l), n] \models \alpha$ 表示 α 在 M_k 的 k-path π_k 的后缀中为 true, π_k 的后缀开始于位置 n, 这里省略 M_k.

(1) $[(\pi_k, l), n] \models p$ iff $p \in \pi_k(n)$;

(2) $[(\pi_k, l), n] \models \neg p$ iff $p \notin \pi_k(n)$;

(3) $[(\pi_k, l), n] \models \alpha_1 \&\& \alpha_2$ iff $[(\pi_k, l), n] \models \alpha_1$ and $[(\pi_k, l), n] \models \alpha_2$;

(4) $[(\pi_k, l), n] \models \alpha_1 | \alpha_2$ iff $[(\pi_k, l), n] \models \alpha_1$ or $[(\pi_k, l), n] \models \alpha_2$;

(5) $[(\pi_k, l), n] \models X! \alpha$ iff $l \in \text{loop}(\pi_k)$, then if $n < k$, $[(\pi_k, l), n+1] \models \alpha$ else $[(\pi_k, l), l] \models \alpha$ And when $l \notin \text{loop}(\pi_k)$ if $n < k$ then $[(\pi_k, l), n+1] \models \alpha$, else false;

(6) $[(\pi_k, l), n] \models \alpha_1 U \alpha_2$ iff $l \in \text{loop}(\pi_k)$, then $\exists n \leqslant i \leqslant k$, $([(\pi_k, l), i] \models \alpha_2$ and $\forall n \leqslant j \leqslant i-1 [(\pi_k, l), j] \models \alpha_1)$ or $\exists l \leqslant i \leqslant n-1$, $([(\pi_k, l), i] \models \alpha_2$ and $\forall n \leqslant j \leqslant k [(\pi_k, l), j] \models \alpha_1$ and $\forall l \leqslant j \leqslant i-1 [(\pi_k, l), j] \models \alpha_1)$ otherwise $\exists n \leqslant i \leqslant k$, $([(\pi_k, l), i] \models \alpha_2$ and $\forall n \leqslant j \leqslant i-1 [(\pi_k, l), j] \models \alpha_1)$;

(7) $[(\pi_k, l), n] \models r \overrightarrow{|} \varphi$ iff $l \in \text{loop}(\pi_k)$, then $\forall n \leqslant i \leqslant k$ $([(\pi_k, l), i] \models \varphi$ and $\forall n \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r)$ or $\forall l \leqslant i \leqslant n-1$ $([(\pi_k, l), i] \models \varphi$ and $\forall n \leqslant j \leqslant k$ $[(\pi_k, l), j] \models r$ and $\forall l \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r)$ Otherwise $\forall n \leqslant i \leqslant k$ $([(\pi_k, l), i] \models \varphi$ and $\forall n \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r)$;

(8) $[(\pi_k, l), n] \models r_1 : r_2$ iff $l \in \text{loop}(\pi_k)$, then $\exists n \leqslant i \leqslant k$ $([(\pi_k, l), i] \models r_2$ and $\forall n \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r_1)$ or $\exists l \leqslant i \leqslant n-1$ $([(\pi_k, l), i] \models r_2$ and $\forall n \leqslant j \leqslant k$ $[(\pi_k, l), j] \models r_1$ and $\forall l \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r_1)$ Otherwise $\exists n \leqslant i \leqslant k$ $([(\pi_k, l), i] \models r_2$ and $\forall n \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r_1)$;

(9) $[(\pi_k, l), n] \models r_1; r_2$ iff $l \in \text{loop}(\pi_k)$, then $\exists n+1 \leqslant i \leqslant k$ $([(\pi_k, l), i] \models r_2$ and $\forall n \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r_1)$ or $\forall n \leqslant i \leqslant k [(\pi_k, l), i] \models r_2$ or $\forall n \leqslant i \leqslant k$ $[(\pi_k, l), i] \models r_1$ or $\exists l \leqslant i \leqslant n-1$ $([(\pi_k, l), i] \models r_2$ and $\forall n \leqslant j \leqslant k$ $[(\pi_k, l), j] \models r_1$ and $\forall l \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r_1)$ or $\forall l \leqslant j \leqslant k$ $[(\pi_k, l), i] \models r_2$ or $\forall l \leqslant j \leqslant k$ $[(\pi_k, l), i] \models r_1$ Otherwise $\exists n+1 \leqslant i \leqslant k$ $([(\pi_k, l), i] \models r_2$ and $\forall n \leqslant j \leqslant i$ $[(\pi_k, l), j] \models r_1)$ or $\forall n \leqslant i \leqslant k$ $[(\pi_k, l), i] \models r_2$ or $\forall n \leqslant i \leqslant k$ $[(\pi_k, l), i] \models r_1$;

(10) $[(\pi_k, l), n] \models r[*]$ iff $l \in \text{loop}(\pi_k)$, then $(\forall n \leqslant i \leqslant k$ $[(\pi_k, l), i] \models r$ or $\exists n+1 \leqslant i \leqslant k$ $([(\pi_k, l), i] \models r[*]$ and $\forall n \leqslant j \leqslant i-1$ $[(\pi_k, l), j] \models r))$ or $(\forall l \leqslant i \leqslant k$ $[(\pi_k, l), i] \models r$ or $\forall l+1 \leqslant i \leqslant n-1$ $([(\pi_k, l), i] \models r[*]$ and $\forall n \leqslant j \leqslant k$ $[(\pi_k, l), j] \models r$ and $\forall l \leqslant j \leqslant i-1$ $[(\pi_k, l), j] \models r))$ Otherwise $[(\pi_0, l), n] \models r[*]$ or $\forall n \leqslant i \leqslant k$ $[(\pi_k, l), j] \models r$ or $\exists n+1 \leqslant i \leqslant k$ $([(\pi_k, l), i] \models r[*]$ and $\forall n \leqslant j \leqslant i-1$ $[(\pi_k, l), j] \models r)$;

(11) $[(\pi_k, l), n] \models \varphi$ abort b iff $l \in \text{loop}(\pi_k)$, then $\forall n \leqslant i \leqslant k$ $[(\pi_k, l), i] \models \varphi$ or $\exists n+1 \leqslant i \leqslant k$ $([(\pi_k, l), i] \models b$ and $\forall n \leqslant j \leqslant i-1$ $[(\pi_k, l), j] \models \varphi)$ or $\exists l+1 \leqslant i \leqslant n-1$ $([(\pi_k, l), i] \models b$ and $\forall n \leqslant j \leqslant k$ $[(\pi_k, l), j] \models \varphi$ and $\forall l \leqslant j \leqslant i-1$ $[(\pi_k, l), j] \models \varphi)$ Otherwise $\forall n \leqslant i \leqslant k$ $[(\pi_k, l), i] \models \varphi$ or $\exists n+1 \leqslant i \leqslant k$ $([(\pi_k, l), i] \models b$ and $\forall n \leqslant j \leqslant i-1$ $[(\pi_k, l), j] \models \varphi)$;

(12) $[(\pi_k, l), n] \models E\alpha$ iff $\exists \pi_k^{[m]} \in \text{Path}_k$ (if $n < l$,

$\pi_k(n) = s^{[m][n]}$ and $\exists\, n \leqslant l \leqslant k\ [(\pi_k, l), n] \mid = \alpha$ otherwise $\pi_k(l) = s^{[m][l]}$ and $\exists\, l \leqslant n \leqslant k\ [(\pi_k, l), n] \mid = \alpha$);

(13) $[(\pi_k, l), n] \mid = G\alpha$ iff $l \in \mathrm{loop}(\pi_k)$ then $\wedge_{i=\min(n,l)}^{k}[(\pi_k, i), n] \mid = \alpha$ Otherwise false.

定义 5　(**PSL** 逻辑有界语义的有效性)

一个 PSL 公式 α 在 k-model M_k 是有效的(记为 $M_k \mid = \alpha$)当且仅当存在某个 $0 \leqslant l \leqslant k$ 使得 $M_k, [(\pi_k, l), 0] \mid = \alpha$, 其中 $\pi_k(0)$ 等于 s_0 对应的状态, 即 α 在 M_k 的初始状态为 true.

定理 1　令 M 是一 Kripke 结构, α 为 PSL 公式, 若对每条 k-path π_k 有 $\mathrm{loop}(\pi_k) = \Phi$, 则 $M \mid = E\alpha$ 成立当且仅当存在 $k \geqslant 0$, 有 $M_k \mid = E\alpha$.

定理 2　令 M 是一 Kripke 结构, α 为 PSL 公式, 若 $M \mid = E\alpha$, 则存在 $k \in N$, 有 $M_k \mid = E\alpha$.

定理 3　令 M 是一 Kripke 结构, α 为 PSL 公式, $\mid M \mid$ 表示 M 的状态个数, 则 $M \mid = E\alpha$ 当且仅当存在 $k \leqslant \mid M \mid \times 2^{\mid \alpha \mid}$ 有 $M_k \mid = E\alpha$.

定理 4　令 M 是一 Kripke 结构, α 为 PSL 公式, $k = \mid M \mid + 1$. 则 $M_k \mid = E\alpha$ 可满足的当且仅当 $M \mid = E\alpha$.

3　将 PSL 的有界模型检验转化(简化)为 SAT(可满足性问题)

上一节, 具体给出了 PSL 的有界语义, 本节, 将有界语义进一步简化(转换)为 SAT, 即将一个 PSL 的有界模型检验问题转化为一个命题公式的可满足性问题. 并给出有界模型检验 PSL 的算法框架. 算法的基本思路是: PSL 公式 α 的有效性可通过验证一个命题公式 $[[M, \alpha]]_k := [[M, \alpha]]_k^s \wedge [[\alpha]]_{M_k}$ 的可满足性的方法来确定, 其中, $[[M, \alpha]]_k^s$ 表示当前的 k 模型, 而 $[[\alpha]]_{M_k}$ 是 α 在 M_k 中成立的必要约束条件, 定义 6 和 7 给出了 $[[\alpha]]_{M_k}$ 的转换公式.

首先, 给出以下定义. 其中 b 是布尔表达式, r, r_1, r_2 是非时钟 SEREs, φ 和 ψ 是非时钟 FL 公式, α 是 PSL 公式, k, l, m, $n \in N$, M 是一 Kripke 结构, $s_k^{[m][n]}$ 表示 S 中一状态.

- 对于两状态 w, v, $H(w, v) := (w = v)$;
- $\mathrm{Path}_k^{[m]} := (\forall w, v\ (\vee_{i=0}^{k-1}(w \leftrightarrow s_k^{[m][i]} \wedge v \leftrightarrow s_k^{[m][i+1]}) \rightarrow T(w, v)))$;
- $L_{k,l}^{[m]} := T(s_k^{[m][k]}, s_k^{[m][l]})$;
- $LL_k^{[m]} := \vee_{l=0}^{k} L_{k,l}^{[m]}$;
- $\exists\, S_k^{[m]}$ 表示序列 $\exists\, s_k^{[m][0]} \ldots \exists\, s_k^{[m][k]}$;
- (k, l)-loop 中, 当 $n < k$ 时 n 的后继 $\mathrm{succ}(n)$ 就是 $\mathrm{succ}(n) := n + 1$, 当 $n = k$ 时, $\mathrm{succ}(n) := l$.

定义 6　(不具有(**k, l**) **loop** 时 **PSL** 公式向 **SAT** 的转换)

(1) $[[b]]_k^{[m][n]} := b(s_k^{[m][n]})$, 即 $b \in l(s_k^{[m][n]})$

(2) $[[\neg b]]_k^i := \neg b(s_k^{[m][n]})$, 即 $b \notin l(s_k^{[m][n]})$

(3) $[[r_1 \&\& r_2]]_k^{[m][n]} := [[r_1]]_k^{[m][n]} \wedge [[r_2]]_k^{[m][n]}$

(4) $[[r_1 \mid r_2]]_k^{[m][n]} := [[r_1]]_k^{[m][n]} \vee [[r_2]]_k^{[m][n]}$

(5) $[[X!\ \varphi]]_k^{[m][n]} =$ if $n < k$ then $[[X!\ \varphi]]_k^{[m][n+1]}$ else false

(6) $[[\varphi U \psi]]_k^{[m][n]} := \vee_{i=n}^{k}([[\psi]]_k^{[m][i]} \wedge \wedge_{j=n}^{i-1}[[\varphi]]_k^{[m][j]})$

(7) $[[r \mid \rightarrow \varphi]]_k^{[m][n]} := \wedge_{i=n}^{k}([[\varphi]]_k^{[m][i]} \wedge \wedge_{j=n}^{i}[[r]]_k^{[m][j]})$

(8) $[[r_1 : r_2]]_k^{[m][n]} := \vee_{i=n}^{k}([[r_2]]_k^{[m][i]} \wedge \wedge_{j=n}^{i}[[r_1]]_k^{[m][j]})$

(9) $[[r_1 ; r_2]]_k^{[m][n]} := \vee_{i=n+1}^{k}([[r_2]]_k^{[m][i]} \wedge \wedge_{j=n}^{i}[[r_1]]_k^{[m][j]}) \vee \wedge_{i=n}^{k}[[r_2]]_k^{[m][i]} \vee \wedge_{i=n}^{k}[[r_1]]_k^{[m][i]}$

(10) $[[r[*]]]_k^{[m][n]} := (k == 0) \vee \wedge_{i=n}^{k}[[r]]_k^{[m][i]} \vee \vee_{i=n+1}^{k}([[r[*]]]_k^{[m][i]} \wedge \wedge_{j=n}^{i-1}[[r]]_k^{[m][j]})$

(11) $[[\varphi\ \mathrm{abort}\ b]]_k^{[m][n]} := \wedge_{i=n}^{k}[[\varphi]]_k^{[m][i]} \vee \vee_{i=n+1}^{k}([[b]]_k^{[m][i]} \wedge \wedge_{j=n}^{i-1}[[\varphi]]_k^{[m][j]})$

(12) $[[E\alpha]]_k^{[m][n]} := \exists\, S_k^{[m+1]}(\mathrm{Path}_k^{[m+1]} \wedge H(s_k^{[m][n]}, s_k^{[m+1][0]}) \wedge (([[\alpha]]_k^{[m+1][0]}) \wedge \neg LL_k^{[m+1]}) \vee \vee_{i=0}^{k}(L_{k,i}^{[m+1]} \wedge [[\alpha]]_{k,i}^{[m+1][0]}))$

(13) $[[G\alpha]]_k^{[m][n]} :=$ false

定义 7　(具有(**k, l**) **loop** 时 **PSL** 公式向 **SAT** 的转换)

(1) $_l[[p]]_k^{[m][n]} := p(s_k^{[m][n]})$, 即 $p \in l(s_k^{[m][n]})$

(2) $_l[[\neg p]]_k^i := \neg p(s_k^{[m][n]})$, 即 $p \notin l(s_k^{[m][n]})$

(3) $_l[[r_1 \&\& r_2]]_k^{[m][n]} := {}_l[[r_1]]_k^{[m][n]} \wedge {}_l[[r_2]]_k^{[m][n]}$

(4) $_l[[r_1 \mid r_2]]_k^{[m][n]} := {}_l[[r_1]]_k^{[m][n]} \vee {}_l[[r_2]]_k^{[m][n]}$

(5) $_l[[X!\ \varphi]]_k^{[m][n]} := {}_l[[\varphi]]_k^{[m][\mathrm{succ}(n)]}$

(6) $_l[[\varphi U \psi]]_k^{[m][n]} := \vee_{i=n}^{k}({}_l[[\psi]]_k^{[m][i]} \wedge \wedge_{j=n}^{i-1}{}_l[[\varphi]]_k^{[m][j]}) \vee \vee_{i=l}^{n-1}({}_l[[\psi]]_k^{[m][i]} \wedge \wedge_{j=n}^{k}{}_l[[\varphi]]_k^{[m][j]} \wedge \wedge_{j=l}^{i}{}_l[[\varphi]]_k^{[m][j]})$

(7) $_l[[r \mapsto \varphi]]_k^{[m][n]} := \wedge_{i=n}^{k}({}_l[[\varphi]]_k^{[m][i]} \wedge \wedge_{j=n}^{i}{}_l[[r]]_k^{[m][j]}) \vee \wedge_{i=l}^{n-1}({}_l[[\varphi]]_k^{[m][i]} \wedge \wedge_{j=n}^{k}{}_l[[r]]_k^{[m][j]} \wedge \wedge_{j=l}^{i}{}_l[[r]]_k^{[m][j]})$

(8) $_l[[r_1 : r_2]]_k^{[m][n]} := \vee_{i=n}^{k}({}_l[[r_2]]_k^{[m][i]} \wedge \wedge_{j=n+1}^{k}{}_l[[r_1]]_k^{[m][j]}) \vee \vee_{i=l}^{n-1}({}_l[[r_2]]_k^{[m][i]} \wedge \wedge_{j=n}^{k}{}_l[[r_1]]_k^{[m][j]} \wedge \wedge_{j=l}^{i}{}_l[[r_1]]_k^{[m][j]})$

(9) $_l[[r_1 ; r_2]]_k^{[m][n]} := \vee_{i=n+1}^{k}({}_l[[r_2]]_k^{[m][i]} \wedge \wedge_{j=n}^{i}{}_l[[r_1]]_k^{[m][j]}) \vee \wedge_{i=n}^{k}{}_l[[r_2]]_k^{[m][i]} \vee \wedge_{i=n}^{k}$

$_l[[r_1]]_k^{[m][i]} \vee (\vee_{i=l}^{n-1}({}_l[[r_2]]_k^{[m][i]} \wedge \wedge_{j=n}^{k}{}_l[[r_1]]_k^{[m][j]} \wedge \wedge_{j=l}^{i}{}_l[[r_1]]_k^{[m][j]}) \vee \wedge_{i=n}^{k}{}_l[[r_2]]_k^{[m][i]} \vee \wedge_{i=n}^{k}{}_l[[r_1]]_k^{[m][i]})$

(10) $_l[[r[*]]]_k^{[m][n]} := {}_l[[r[*]]]_{k=0}^{[m][n]} \vee (\wedge_{i=n}^{k}{}_l[[r]]_k^{[m][i]} \vee \vee_{i=n+1}^{k}({}_l[[r[*]]]_k^{[m][i]} \wedge \wedge_{j=n}^{i-1}{}_l[[r]]_k^{[m][j]}))$ $\vee ((\wedge_{i=l}^{k}[[r]]_k^{[m][i]} \vee \vee_{i=l+1}^{n-1}({}_l[[r[*]]]_k^{[m][i]} \wedge \wedge_{j=n}^{k}{}_l[[r]]_k^{[m][j]} \wedge \wedge_{j=l}^{i-1}{}_l[[r]]_k^{[m][j]}))$

(11) $_l[[\varphi \text{ abort } b]]_k^{[m][n]} := \wedge_{i=n}^{k}{}_l[[\varphi]]_k^{[m][i]} \vee \vee_{i=n+1}^{k}({}_l[[b]]_k^{[m][i]} \wedge \wedge_{j=n}^{i-1}{}_l[[\varphi]]_k^{[m][j]}) \vee \vee_{i=l+1}^{n+1}({}_l[[b]]_k^{[m][i]} \wedge \wedge_{j=n}^{k}{}_l[[\varphi]]_k^{[m][j]} \wedge \wedge_{j=l}^{i-1}{}_l[[\varphi]]_k^{[m][j]})$

(12) $[[E\alpha]]_{k,l}^{[m][n]} := \exists S_k^{[m+1]}(\text{Path}_k^{[m+1]} \wedge H(s_k^{[m][n]}, s_k^{[m+1][0]}) \wedge (([[\alpha]]_k^{[m+1][0]}) \wedge \neg LL_k^{[m+1]}) \vee \vee_{i=0}^{k}(L_{k,i}^{[m+1]} \wedge [[\alpha]]_{k,i}^{[m+1][0]}))$

(13) $[[G\alpha]]_{k,l}^{[m][n]} := \wedge_{i=\min(l,n)}^{k}[[\alpha]]_{k,l}^{[m][i]}$

定义 6 和 7 中的公式(12)、(13)分别参考文献[15]得到.

下面定义函数 depth_k, 以确定在 k 模型 M_k 中验证 PSL 公式过程中需构造 k-path 的条数:

定义 **8(PSL 公式长度 depth$k$)**

α, β 为 PSL 公式, $k \in N$, 则各公式长度定义如下:

(1) 若 $p \in AP$, 则 $\text{depth}_k(p) = \text{depth}_k(\neg p) = 0$;

(2) 若 $f = \alpha | \beta$, 则 $\text{depth}_k(f) = \max\{\text{depth}_k(\alpha), \text{depth}_k(\beta)\}$;

(3) 若 $f = \alpha \&\& \beta$, 则 $\text{depth}_k(f) = \text{depth}_k(\alpha) + \text{depth}_k(\beta)\}$;

(4) 若 $f = E\alpha$, 则 $\text{depth}_k(f) = \text{depth}_k(\alpha) + 1$;

(5) 若 $f = G\alpha$, 则 $\text{depth}_k(f) = (k+1)\text{depth}_k(\alpha)$;

(6) 若 $f = X!\alpha$, 则 $\text{depth}_k(f) = \text{depth}_k(\alpha)$;

(7) 若 $f = \varphi \text{ abort } b$, 则 $\text{depth}_k(f) = \text{depth}_k(\varphi)$ 或 $k \cdot \text{depth}_k(\varphi) + \text{depth}_k(b)$

(8) 若 $f = \alpha U \beta$, 或 $f = \alpha : \beta$, 或 $f = \alpha ; \beta$, 或 $f = \alpha | \rightarrow \beta$ 则 $\text{depth}_k(f) = k \cdot \text{depth}_k(\alpha) + \text{depth}_k(\beta)$;

(9) 若 $f = r[*]$(即是 r 的传递闭包), 则 $\text{depth}_k(f) = 0$ 或 $\text{depth}_k(r) + k$;

已知全局状态变量 w, 可用全局状态变量的有限序列$(w_0, \cdots, w_k)$来表示一条 k-path, 为构造命题公式 $[[M, \alpha]]_k^s$, 需要构造 $\text{depth}_k(\alpha)$ 条这样的 k-path.

定义 **9(状态迁移关系的公式转化)** 对一 Kripke 结构 M, $k \in N$

$$[[M, \alpha]]_k^s := I_s(w_{0,0}) \wedge \wedge_{j=1}^{\text{depth}_k(\alpha)} \wedge_{i=0}^{k-1} T(w_{i,j}, w_{i+1,j})$$

其中, $I_s(w_{0,0})$ 表示用全局状态变量 $w_{0,0}$ 表示状态 s; 对于 $T(w, v)$, 令 s 和 s' 分别是 w 和 v 的一个赋值, 则 $T(w, v)$ 为 true 当且仅当 $(s, s') \in T$; 对于任意的 $0 \leqslant i \leqslant k-1$, $1 \leqslant j \leqslant \text{depth}_k(\alpha)$, $w_{i,j}$ 是全局状态变量, 表示第 j 条 k-path 的第 i 个状态. 通过定义 6 和 7, 将 PSL 公式 α 转换为一个命题公式$[[\alpha]]_{M_k}$, 则整体转换命题公式 $[[M, \alpha]]_k$ 的构造如定义 10.

定义 **10**(整体转化 **General Translation**) 令 α 为 FL 公式, M 是一 Kripke 结构, 且 $k \in N$

$$[[M, \alpha]]_k := [[M, \alpha]]_k^s \wedge [[\alpha]]_{M_k} := [[M, \alpha]]_k^s \wedge [[\alpha]]_k^{[0][0]}$$

定理 **5** 令 M 是一 Kripke 结构, α 为 PSL 公式, $[[M, \alpha]]_k$ 是满足的当且仅当 $M_k | = E\alpha$.

定理 **6** 令 M 是一 Kripke 结构, α 为 PSL 公式, 若对每条 k-path π_k 有 $\text{loop}(\pi_k) = \Phi$, 则$[[M, \alpha]]_k$ 蕴含 $M_k | = E\alpha$.

定理 **7** 令 M 是一 Kripke 结构, α 为 PSL 公式, $|M|$ 表示 M 的状态个数, 则$[[M, \alpha]]_k$ 是可满足当且仅当存在 $k \leqslant |M| \leqslant 2^{|\alpha|}$ 有 $M_k | = E\alpha$.

定理 **8** 令 M 是一 Kripke 结构, α 为 PSL 公式, $k = |M|$, 则$[[M, \alpha]]_k$ 可满足的当且仅当 $M_k | = E\alpha$.

通过定义 6、7 和 9、10 的公式转换构造命题公式 $[[M, \alpha]]_k := [[M, \alpha]]_k^s \wedge [[\alpha]]_{M_k}$, 再根据上述定理, 公式 α 在 M_k 的有效性可通过命题公式$[[M, \alpha]]_k$ 的可满足性来验证. 令 M 是一个模型, α 是一个 PSL 公式, 则 PSL 的有界模型检验算法框架如下:

定义 **11** (**PSL** 的有界模型检验算法框架)

```
0  Procedure BMC(M, α)
1  for k = 1 to |M|
2      选择 M 的 k 模型 M_k, 构造满足迁移关系的命题公式 T(w, v), 其中 w, v 是全局状态变量;
3      根据定义 8, 计算 depth_k(α);
4      根据 Step3 及定义 9, 将 M 的状态迁移关系转化为一个命题公式[[M, α]]_k^s;
5      根据定义 6, 7, 在 M 中将公式 α 转换为一个命题公式[[α]]_{M_k} := [[α]]_k^{[0][0]};
6      根据定义 10, 构造命题公式[[M, α]]_k := [[M, α]]_k^s ∧ [[α]]_{M_k}, 并验证其满足性:
7      if ([[M, α]]_k := [[M, α]]_k^s ∧ [[α]]_{M_k} 是可满足的)
8      报告 M| = α 是不合法的(invalid);
9      当 k = |M|, 则报告 M| = α 是合法的.
```

4 实例 —队列控制电路

图 1 描述了队列控制电路图, 包括队列和相应的控制芯片. 队列具有两个指针: qFirst 指示第一数据单元位置, qLast 指示最后数据单元位置. 控制芯片 cntrl 包含两个输入信号 qInsert(插入数据), qRemove(取数据), 以及三个输出信号 qFull(队列已满)、qEmpty(队列空)

和 qError(队列上溢或下溢) . 对于队列两种情况必须避免: (1) 队列已满, 仍执行插入数据操作, 称为上溢; (2) 队列已空, 仍执行取数操作, 称为下溢.

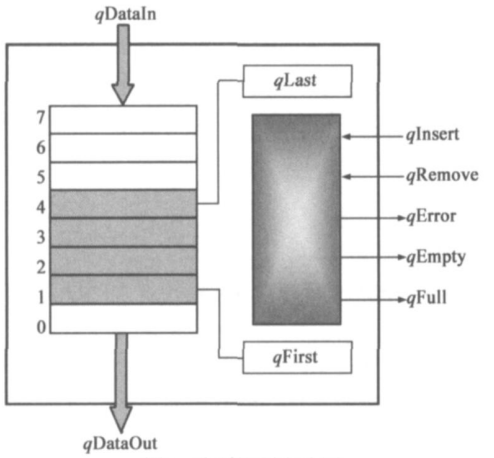

图1　队列控制电路图

```
function [3:0] qNext
        input [3:0] p;
        qNext=((p+1) mod qSize);
end function
assign  qFull=(qNext(qLast)= =qFirst);
assign  qEmpty=(qLast= =qFirst);
assign  qInsert=(qNext= =qNext+1);
assign  qRemove=(qNext= =qNext-1);
assign  qError=(qFull && qInsert)|(qEmpty&& qRemove)
```

图2　队列控制电路部分代码及各信号指派

图 2 给出队列控制电路部分代码及各信号指派.

分析以上代码, 得到电路的两则用 PSL 表达的属性:

(1) (qFull && qInsert $\rightarrow$ X! (¬ qEmpty)) abort (¬rstN) ; (2) (qEmpty && qRemove $\rightarrow$ X! (¬ qFull)) abort (¬rst$N$) . rstN 是表征电路重初始化操作符号. 语句(1) 表示不执行初始化操作时, 性质“队列满并且插入数据,则下一步队列非空”不要求必需成立; 语句(2) 表示不执行初始化操作时, 性质“队列空且取数据, 则下一步队列非满”不要求必须成立.

以(1) 为例:

α= (qFull && qInsert $\rightarrow$ X! (¬ qEmpty)) abort (¬rstN) : = (¬(qFull && qInsert) | X! (¬ qEmpty)) abort (¬rstN)

设 L= { L_1, L_2} , 其中 L_1, L_2 分别表示队列和控制芯片, 其局部状态集合分别是: L_1= { e1234(表示字节 1, 2, 3, 4 均空) , e123f4(字节 1, 2, 3 空, 4 满, 后类同) , e12f34, e1f234, f1234} , L_2= { insert, remove, rst$N$} . 整个电路的基本状态集合 S_c= { empty, full, error} . 对一个状态 sc$\in S_c$, 定义其前置条件 pre(sc) 和后置条件 post(sc) 均为一个局部或全局状态集合, 分别表示 sc 的先决条件和以后的执行状态. 例如 pre(error) = { f1234 ∧ insert} ∪ { e1234 ∧ remove} , post(error) = { rst$N$} . 另外, b(sc) 表示电路基本状态为 sc 时, 可能会改变局部状态变量的 L 集合. 下面描述 k= 2 时公式[[M, α]] $_k$ 的构造过程.

4.1　命题公式$[[M^{\alpha, s_0}]]_2$的构造

首先对局部状态进行二进制编码. 由公式转换过程可知 L_1 需要 n_1= 3 个二进制位表示 5 个局部状态: (0, 0, 0) = e1234, (0, 0, 1) = e123f4, (0, 1, 0) = e12f34, (0, 1, 1) = e1f234, (1, 0, 0) = f1234. L_2 需要 n_2= 2 个二进制位表示 3 个局部状态: (0, 0) = ¬ rst$N$, (0, 1) = insert, (1, 0) = remove. 因此, 一个全局状态变量需要 m= n_1+ n_2 = 5 个二进制位来表示. L 的局部命题变量索引集合分别是I_1= { 0, 1, 2} , I_2= { 3, 4} . 令 w= (w[0], …w[4]) 和 v= (v[0], …v[4]) 是两个全局状态变量, 构造下列命题公式:

• 对 L_1 与 L_2 的每个局部状态, 根据其二进制编码构造相应的公式, 如 $P_{\text{remove}}(w)$: = w[4] ∧ ¬ w[5] .

• 系统的初始状态由命题公式 $Is_0(w)$: = $\wedge_{i\in\{0,\ldots4\}}$ ¬ w[i] 表示.

• $T(w, v)$: = $\vee_{sc\in S_c}(\wedge_{i\in \text{pre}(sc)} P_i(w) \wedge \wedge_{i\in \text{post}(sc)} P_i(v) \wedge \wedge_{i\in B}(w[i] \Leftrightarrow v[i])) \vee (\wedge_{sc\in S_c} \vee_{i\in \text{pre}(sc)} (\neg w_i) \wedge \wedge_{i\in\{0,\ldots4\}}(w[i] \Leftrightarrow v[i]))$

其中, B: = $\cup_{i\in(L-b(sc))} I_i$ 是电路状态不能改变的全局变量的局部状态的索引集合. 令 s, s' 分别是由全局状态变量w 和v 表示的状态, 则 $T(w, v)$ 表示从 s 到s' 是可达的. $T(w, v)$ 由两个析取式组成: 第 1 个表示在状态 s 存在一组电路的基本状态S_c, post(sc) 后状态变为 s', 并且 s' 中电路基本状态不能修改的局部状态位保持不变; 第 2 个表示如果在状态 s 不存在任何电路的基本状态的先决条件, 则 s= s', 这保证了状态迁移关系 T 是完全的(total) . 这样, $T(w, v)$ 为 true 当且仅当(s, s') ∈T. 由定义 8 计算 depth$_2$(α) = 5, 再据定义 9 构造状态迁移关系命题公式:

$$[[M^{\alpha, s_0}]]_2 := I_{s0}(w) \wedge \wedge_{1\leqslant j\leqslant 5\wedge 0\leqslant i< 2} T(w_{i,j}, w_{i+1,j})$$

4.2　命题公式[[α]] $_{M_2}$的构造

令 w= (w[0], …w[4]) 是一个全局状态变量. 下面根据 PSL 公式转换定义 6 和 7 构造公式$[[\alpha]]_{M_2}^{[0,0]}$(文中省略非 k- loop 语义下的公式转换过程) :

α= (¬ (qFull && qInsert) | $X!$ (¬ qEmpty)) abort (¬ rst$N$)

• $_l[[\alpha]]_2^{[0][0]}$: = $_l[[($ ¬ (qFull && qInsert) | $X!$ (¬ qEmpty)) abort(¬ rst$N$)]]$_2^{[0][0]}$

: = $\wedge_{i=0}^{2}{}_l[[$ ¬ (qFull && qInsert) | $X!$ (¬ qEmpr

$ty)]]_2^{0/[i}\vee\vee_{i=1}^{2}({}_l[[\neg rstN]]_2^{0/[i}\wedge\wedge_{j=0}^{i-1}{}_l[[\neg(qFull\ \&\&\ qInsert)|X!(\neg qEmpty)]]_2^{0/[j}))\vee\vee_{i=l+1}^{1}({}_l[[\neg rstN]]_k^{0/[i}\wedge\wedge_{j=0}^{2}{}_l[[\neg(qFull\ \&\&\ qInsert)|X!(\neg qEmpty)]]_2^{0/[j}\wedge\wedge_{j=l}^{i-1}{}_l[[\neg(qFull\ \&\&\ qInsert)|X!(\neg qEmpty)]]_2^{0/[j}) \quad (1)$

$\cdot\ {}_l[[qFull\ \&\&\ qInsert]]_k^{m/[n}:={}_l[[qFull]]_k^{m/[n}\wedge{}_l[[qInsert]]_k^{m/[n} \quad (2)$

$\cdot\ {}_l[[qFull]]_k^{m/[n}:=qFull(w_k^{m/[n}):=(w_k^{m/[n}[0]\wedge\neg w_k^{m/[n}[1]\wedge\neg w_k^{m/[n}[2]) \quad (3)$

$\cdot\ {}_l[[qInsert]]_k^{m/[n}:=qInsert(w_k^{m/[n}):=(\neg w_k^{m/[n}[3]\wedge w_k^{m/[n}[4]) \quad (4)$

$\cdot\ {}_l[[\neg(qFull\ \&\&\ qInsert)]]_k^{m/[n}:=\neg w_k^{m/[n}[0]\vee w_k^{m/[n}[1]\vee w_k^{m/[n}[2]\vee w_k^{m/[n}[3]\vee\neg w_k^{m/[n}[4] \quad (5)$

$\cdot\ {}_l[[\neg qEmpty]]_k^{m/[n}:=\neg qEmpty(w_k^{m/[n}):=w_k^{m/[n}[0]\vee w_k^{m/[n}[1]\vee w_k^{m/[n}[2] \quad (6)$

$\cdot\ {}_l[[X!(\neg qEmpty)]]_k^{m/[n}:={}_l[[\neg qEmpty]]_k^{m/[succ(n)}:=w_k^{m/[succ(n)}[0]\vee w_k^{m/[succ(n)}[1]\vee w_k^{m/[succ(n)}[2] \quad (7)$

$\cdot\ {}_l[[\neg(qFull\ \&\&\ qInsert)|X!(\neg qEmpty)]]_k^{m/[n}:=\neg w_k^{m/[n}[0]\vee w_k^{m/[n}[1]\vee w_k^{m/[n}[2]\vee w_k^{m/[n}[3]\vee\neg w_k^{m/[n}[4]\vee w_k^{m/[succ(n)}[0]\vee w_k^{m/[succ(n)}[1]\vee w_k^{m/[succ(n)}[2] \quad (8)$

$\cdot\ {}_l[[\neg rstN]]_k^{m/[n}:=\neg rstN(w_k^{m/[n}):=\neg w_k^{m/[n}[3]\wedge\neg w_k^{m/[n}[4] \quad (9)$

将公式(2)~(9)代入式(1)得到最后的命题公式$[[\alpha]]_{M_2}$. 根据3节的算法框架进行有界模型检验. 令k从1到$|M|$依次构造命题公式$[[M,\alpha]]_k$, 每次构造过程类似于上述$k=2$的情况, 然后用SAT解算器PROVER可验证$[[M,\alpha]]_k$的满足性. 结果$k=6$是满足的, 算法结束.

5 结论

基于SAT的有界模型检验自1999年提出以来, 被公认为是基于OBDD的符号化模型检验的重要补充技术. 然而, 直到现在, 有界模型检验已经验证的属性逻辑还十分有限. 本文首次提出时序逻辑PSL的有界模型检验方法. 首先, 定义PSL逻辑的有界语义, 而后, 将有界语义进一步简化为SAT, 分别将PSL性质规约公式α、系统M的状态迁移关系转换为命题公式$[[\alpha]]_{M_k}$和$[[M,\alpha]]_k^s$, 最后验证命题公式$[[M,\alpha]]_k^s\wedge[[\alpha]]_{M_k}$的可满足性, 这样就将时序逻辑*PSL*的存在模型检验转化为一个命题公式的可满足性问题, 该方法的优势在于简洁、易操作, 并且规范. 论文最后通过例举一个队列控制电路系统实例具体解释算法执行过程.

本文在构造状态迁移关系命题公式时, 考虑的k-path的个数是由$depth_k(\alpha)$确定的, 而实际上, 可以借鉴文献[17]的受限的转化(Restricted Translation)过程, 进一步减少要构造的k-path的条数, 使得转换效率更高. 另外, 还需要进一步研究并利用产生命题公式过程中的一些优化技术, 如文献[18]提出的路径优化方法等.

参考文献:

[1] Accellera Organization, Inc. Formal Semantics of Accellera Property Specification Language[S]. Appendix B of http://www.eda.org/vfv/docs/PSL-v1.1.pdf, 2004.

[2] Accellea. Property Specification Language Reference Manual, v1.0[S]. http://www.haifa.il.ibm.com/projects/verification/sugar, 2003.

[3] Beer I, Berr David S, Eisner C, Fisman D, Gringauze A, Rodeh Y. The temporal logic Sugar[A]. Proceedings of the 13th International Conference on Computer Aided Verification (CAV2001)[C]. LNCS2102, Paris(France): Springer Verlag, 2001. 363-367.

[4] Clarke EM, Grumber O, Peled DA. Model Checking[M]. Cambridge: MIT Press, 2000. 28-33.

[5] Berr Ari M, Manna A, Pnueli. The temporal logic of branching time[J]. Acta Information, 1983(20): 207-226.

[6] Doron Bustan, Dana Fisman, John Havlicek. Automata construction for PSL[EB/OL]. http://www.wisdom.weizmann.ac.il/~dana/publicat/automata_constructionTR.pdf, 2005.

[7] S Berr David, R Bloem, D Fisman, A Griesmayer, et al. Automata Construction Algorithms Optimized for PSL[R]. PROSYD, 2005.

[8] K Heljanko, T Junttila, M Kein?nen, et al. Bounded model checking for weak alternating Büchi automata[A]. Proceedings of the 18th International Conference on Computer Aided Verification (CAV 2006)[C]. LNCS 4144, Seattle(USA): Springer Verlag, 2006. 96-108.

[9] A Cimatti, M Roveri, S Semprini, S Tonetta. From PSL to NBA: a modular symbolic encoding[A]. Proceedings of the Formal Methods in Computer Aided Design(FMCAD'06)[C]. Washington DC, USA: IEEE Computer Society Press, 2006. 125-133.

[10] Biere A, Cimatti A, Clarke E M, Zhu Y. Symbolic model checking without BDDs[A]. Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)[C]. LNCS 1597, Amsterdam, Netherlands: Springer Verlag, 1999. 193-207.

[11] Latvala T, Biere A, Heljanko K, Junttila T. Simple is better: Efficient bounded model checking for past LTL[A]. International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'2005)[C]. LNCS 3385, Paris, FRANCE: Springer Verlag, 2005. 380-395.

[12] Penczek W, Wozna B, Zbrzezny A. Bounded model checking for the universal fragment of CTL[J]. Fundamental Informaticae, 2002, 51(1): 135–156.

[13] Wozna B. ACTL* properties and bounded model checking [J]. Fundamental Informaticae, 2004, 62(2): 1–23.

[14] 骆翔宇, 苏开乐, 杨晋吉. 有界模型检测同步多智体系统的时态认知逻辑[J]. 软件学报, 2006, 17(12): 2485–2498.
LUO Xiang-yu, SU Kai-le, YANG Jin-ji. Bounded model checking for temporal epistemic logic in synchronous multi-agent systems[J]. Journal of Software, 2006, 17(12): 2485–2498. (in Chinese)

[15] Zhi-Hong Tao, Cong-Hua Zhou, Zhong Chen, Li-Fu Wang. Bounded model checking of CTL* [J]. Journal of Comput Sci. &Technol, 2007, 22(1): 39–43.

[16] Wozna B. Bounded model checking for the universal of CTL* [R]. London: King's College London, 2004.

[17] Wenhui Zhang. Verification of ACTL properties by bounded model checking[A]. Proceedings of the 11th International Conference on Computer Aided Systems Theory (EUROCAST' 2007)[C]. LNCS 4739, Las Palmas de Gran Canaria, Spain: Springer-Verlag, 2007. 556–563.

[18] Latvala T, Biere A, Heljanko K, Junttila T. Simple bounded LTL model checking[A]. Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD' 2004)[C]. LNCS 3312, Texas, USA: Springer-Verlag, 2004. 186–200.

作者简介:

虞　蕾　女, 1978 年出生于浙江浦江, 博士后, 讲师, 主要研究领域为模型检验, 航迹规划.
Email: yuleizj163.com

赵宗涛　男, 1942 年出生于江苏徐州, 教授, 博导, 主要研究领域为决策支持系统, 软件工程.
Email: zztxian@163.com