软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(4):898−906 [doi: 10.13328/j.cnki.jos.005057]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

NuTL2PFG: νTL 公式的可满足性检查*

刘 尧[1,2], 段振华[1,2], 田 聪[1,2]

[1](西安电子科技大学 计算理论与技术研究所,陕西 西安 710071)

[2](综合业务网理论及关键技术国家重点实验室(西安电子科技大学),陕西 西安 710071)

通讯作者: 段振华,田聪, E-mail: {zhhduan,ctian}@mail.xidian.edu.cn

摘 要: 线性μ演算(linear time μ-calculus,简称νTL)语法简单,表达能力强,可用于验证并发程序的多种性质.然而,不动点操作符的嵌套使其判定问题难以有效解决.针对这一问题,开发了工具 NuTL2PFG,用以判定νTL 公式的可满足性.利用νTL 公式的当前-未来范式(present future form,简称 PF 式),该工具能够为一个给定公式构造其当前-未来范式图(present future form graph,简称 PFG),用以描述满足该公式的模型.通过在所得 PFG 中寻找一条ν-路径,即,不涉及最小不动点公式的无穷展开的路径,该工具便可判断出给定公式的可满足性.实验结果表明,NuTL2PFG 的执行效率优于已有工具.

关键词: 线性μ演算;当前-未来范式;当前-未来范式图;可满足性

中图法分类号: TP301

中文引用格式: 刘尧,段振华,田聪.NuTL2PFG:νTL 公式的可满足性检查.软件学报,2017,28(4):898−906. http://www.jos.org.cn/1000-9825/5057.htm

英文引用格式: Liu Y, Duan ZH, Tian C. NuTL2PFG: Checking satisfiability of νTL formulas. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):898−906 (in Chinese). http://www.jos.org.cn/1000-9825/5057.htm

NuTL2PFG: Checking Satisfiability of νTL Formulas

LIU Yao[1,2], DUAN Zhen-Hua[1,2], TIAN Cong[1,2]

[1](Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China)

[2](State Key Laboratory of Integrated Service Networks (Xidian University), Xi'an 710071, China)

Abstract: Linear time μ-calculus (νTL) is a formalism which has a strong expressive power with a succinct syntax. It is useful for specifying and verifying various properties of concurrent programs. However, the nesting of fix point operators makes its decision problem difficult to solve. To tackle the issue, a tool called NuTL2PFG for checking the satisfiability of νTL formulas is developed in this paper. Based on present future form (PF form) of νTL formulas, the tool is able to construct the present future form graph (PFG) for a given formula to specify the models that satisfy the formula. Further, the tool checks the satisfiability of a given formula by searching for a ν-path in its PFG free of infinite unfoldings of least fixpoints. Experimental results show that NuTL2PFG is more efficient than the existing tools.

Key words: linear time μ-calculus; present future form; present future form graph; satisfiability

线性μ演算(linear time μ-calculus,简称νTL)[1]是模态μ演算[2]的时序版本,它与线性时序逻辑(linear temporal logic,简称 LTL)[3]最大的不同在于最小和最大不动点操作符的引入.νTL 不仅语法简洁,而且具有完全正则的表达能力[4,5],可用于表达和验证并发程序的多种正则性质,这些性质是 LTL 无法表达的.因此,νTL 在过去的几十

* 基金项目: 国家自然科学基金(61322202, 61133001, 61420106004, 91418201)
Foundation item: National Natural Science Foundation of China (61322202, 61133001, 61420106004, 91418201)
收稿时间: 2015-12-25; 修改时间: 2016-02-24; 采用时间: 2016-03-08; jos 在线出版时间: 2016-05-03
CNKI 网络优先出版: 2016-05-04 08:44:10, http://www.cnki.net/kcms/detail/11.2560.TP.20160504.0844.005.html

年里受到了科学研究和工程技术领域的广泛关注.从实际应用的角度来看,为νTL 公式的可满足性提出一种有效的判定算法十分重要,这对获得有效的νTL 模型检测算法起着决定性作用.可满足性是指找出一种判定算法,用以判断一个给定公式是否可满足,它的复杂度是 PSPACE-complete[6,7],即,指数级时间和多项式空间复杂度.

针对νTL 的判定问题,目前主要存在两类方法:基于自动机的方法[6,8]和基于 Tableau 的方法[9−11].基于自动机的方法主要用到α和β两个自动机,其中,α用于检测预模型(pre-model)的一致性,β用于检测最小不动点的非良基性(non-well-foundedness).通过对α和β的补的乘积自动机进行判空,从而判断公式的可满足性,时间复杂度为 $2^{O(n^4)}$,其中,n 表示公式的长度(下同).基于 Tableau 的方法主要利用相关推导规则来构造给定公式的 Tableau,通过检查所得 Tableau 中各分支成功与否,进而判断公式的可满足性,时间复杂度为 $2^{O(n^2\log n)}$.此外,文献[12]证明,若一个公式可满足,则该公式能够生成一个好的 Hintikka 结构,进而将公式的可满足性问题转化为图的路径搜索问题.该方法的时间复杂度为 $2^{O(n^4)}$,空间消耗为指数级.然而,上述方法中涉及的判定过程较为复杂,并且缺少相应的支持工具.文献[13]为νTL 提出了一种简单的判定方法,其时间复杂度为 $2^{O(n^2\log n)}$,并且用 OCAML 实现了工具 PS4NuTL.该工具是当前存在的针对νTL 判定问题的唯一工具.

本文展示了原型工具 NuTL2PFG,用以检查νTL 公式的可满足性.NuTL2PFG 能够将νTL 公式转化为当前-未来范式(present future form,简称 PF 式)[14,15].当前部分是原子命题或它们的非的合取式,未来部分是给定公式的闭包中元素的合取式.基于 PF 式,NuTL2PFG 能够构造出一个公式的当前-未来范式图(present future form graph,简称 PFG),用以描述该公式的模型.在构造 PFG 的过程中,需要对某些边添加记号,用于记录不动点公式的展开情况.一个记号就是公式中出现的变量的一个子集.通过记号,能够找出 PFG 中的ν-路径,即,不存在最小不动点公式的无穷展开的路径.这样,对于一个给定的公式,NuTL2PFG 通过构造其 PFG 并在该 PFG 中寻找一条ν-路径,便可判断该公式的可满足性.基于 PFG 的判定方法的时间复杂度为 $2^{O(n)}$,空间消耗为指数级[15].实验结果显示,NuTL2PFG 的表现优于文献[13]中的工具 PS4NuTL.需要注意的是:本文方法的复杂度仅适用于受卫公式,而已有方法的复杂度适用于任意公式.

PF 式和 PFG 的概念受命题投影时序逻辑(propositional projection temporal logic,简称 PPTL)的范式(normal form)和范式图(normal form graph)[16,17]的启发.范式和范式图能够用于判定 PPTL 公式的可满足性[18−20].

本文第 1 节给出νTL 的语法、语义和其他基本概念.第 2 节介绍νTL 公式的 PF 式、PFG 以及 PFG 中的ν-路径.第 3 节详细描述工具 NuTL2PFG 的架构.第 4 节给出相关实验结果与分析.第 5 节对本文进行总结,并进一步指出今后的研究方向.

1 基础概念

1.1 νTL语法

令$\mathcal{P}$为一个原子命题集合,$\mathcal{V}$为一个变量集合.νTL 公式可根据以下语法进行构造:

$$\phi ::= p|\neg p|X|\phi\vee\phi|\phi\wedge\phi|\bigcirc\phi|\mu X.\phi|\nu X.\phi,$$

其中,$p\in\mathcal{P}$,$X\in\mathcal{V}$.

通常用σ来表示μ或ν.σX表示变量 X 被σ绑定.在一个公式中,如果变量 X 的出现位于σX的辖域之内,则称 X 为约束变量(bound variable);否则,称为自由变量(free variable).如果一个公式中不包含自由变量,则称该公式是封闭的(closed).通常以$\phi[\varphi/X]$表示将公式ϕ中所有变量 X 的自由出现替换为公式φ.假定在一个公式中每个变量至多可被绑定 1 次,那么按照上述语法构造出的公式都是正则式(positive normal form)[21].

对于公式φ中出现的任意约束变量 X,都存在一个形如$\sigma X.\varphi_X$ 的φ的子公式,称该公式为 X 在φ中的绑定式,记作$\Gamma_\varphi(X)$.给定φ中的两个约束变量X和Y,如果$\Gamma_\varphi(Y)$是$\Gamma_\varphi(X)$的子公式,则称X高于Y.如果X高于Y并且在$\Gamma_\varphi(Y)$中自由出现,则称 Y 依赖于 X,记作 $X\lhd Y$.在一个公式中,变量间的依赖关系具有可传递性.

例如,对于公式$\nu X.\mu Y.(p\wedge\bigcirc X\vee\nu Z.\bigcirc(Y\wedge q\vee Z\wedge r))$,可得 $X\lhd Y\lhd Z$.

一个公式是受卫的(guarded)当且仅当该公式中任意约束变量均处于○操作符的辖域之内.任何公式都能转

化为与之等价的受卫公式[22].对公式进行受卫转换时,在最坏情况下,可能导致公式长度的指数级增长[23].

对于一个给定的公式φ,基于文献[24],归纳定义其闭包 $CL(\varphi)$如下.

(1) φ,true$\in CL(\varphi)$;

(2) 若$\phi\vee\psi\in CL(\varphi)$或$\phi\wedge\psi\in CL(\varphi)$,则$\phi,\psi\in CL(\varphi)$;

(3) 若$\bigcirc\phi\in CL(\varphi)$,则$\phi\in CL(\varphi)$;

(4) 若$\sigma X.\phi\in CL(\varphi)$,则$\phi[\sigma X.\phi/X]\in CL(\varphi)$.

对于任意公式φ,$CL(\varphi)$中的公式个数为 $O(|\varphi|)$[24].这里,$|\varphi|$表示φ的长度.

1.2 νTL语义

νTL 公式的语义需要通过线性结构进行解释.一个关于$\mathcal{P}$的线性结构是一个函数$\mathcal{K}:\mathbb{N}\to 2^{\mathcal{P}}$,这里,$\mathbb{N}$表示自然数集.给定一个环境 $e:\mathcal{V}\to 2^{\mathbb{N}}$,公式$\phi$关于$\mathcal{K}$和 e 的语义定义如下:

$$[\![p]\!]_e^{\mathcal{K}}:=\{i\in\mathbb{N}\mid p\in\mathcal{K}(i)\},[\![\neg p]\!]_e^{\mathcal{K}}:=\{i\in\mathbb{N}\mid p\notin\mathcal{K}(i)\},[\![X]\!]_e^{\mathcal{K}}:=e(X),$$

$$[\![\varphi\vee\psi]\!]_e^{\mathcal{K}}:=[\![\varphi]\!]_e^{\mathcal{K}}\cup[\![\psi]\!]_e^{\mathcal{K}},[\![\varphi\wedge\psi]\!]_e^{\mathcal{K}}:=[\![\varphi]\!]_e^{\mathcal{K}}\cap[\![\psi]\!]_e^{\mathcal{K}},[\![\bigcirc\varphi]\!]_e^{\mathcal{K}}:=\{i\in\mathbb{N}\mid i+1\in[\![\varphi]\!]_e^{\mathcal{K}}\},$$

$$[\![\mu X.\varphi]\!]_e^{\mathcal{K}}:=\bigcap\{W\subseteq\mathbb{N}\mid[\![\varphi]\!]_{e[X\mapsto W]}^{\mathcal{K}}\subseteq W\},[\![\nu X.\varphi]\!]_e^{\mathcal{K}}:=\bigcup\{W\subseteq\mathbb{N}\mid W\subseteq[\![\varphi]\!]_{e[X\mapsto W]}^{\mathcal{K}}\},$$

其中,$e[X\mapsto W]$表示一个新的环境 e'.除 $e'(X)=W$ 之外,e'对其他变量的赋值与 e 相同.环境 e 用于对自由变量进行赋值,当ϕ封闭时可省略.

对于一个给定的公式φ,其在线性结构$\mathcal{K}$的状态 i 下为真,记作$\mathcal{K},i\vDash\varphi$,当且仅当 $i\in[\![\varphi]\!]_e^{\mathcal{K}}$.称$\varphi$是有效的,记作$\vDash\varphi$,当且仅当对于任意线性结构$\mathcal{K}$以及$\mathcal{K}$中的任意状态 j,均有$\mathcal{K},j\vDash\varphi$;称$\varphi$是可满足的,当且仅当存在一个线性结构$\mathcal{K}$以及$\mathcal{K}$中的一个状态 j,使得$\mathcal{K},j\vDash\varphi$.

2 PF 式和 PFG

从本节开始,只考虑封闭的受卫公式.同时,$\vee$操作符不得作为$\bigcirc$操作符辖域下的主操作符(可由等价性$\bigcirc(\varphi_1\vee\varphi_2)\equiv\bigcirc\varphi_1\vee\bigcirc\varphi_2$进行相关转换).

2.1 PF式

定义 1(PF 式). 给定一个公式φ,令$\mathcal{Q}$表示φ中出现的原子命题集合.φ的 PF 式定义为

$$\varphi\equiv\vee_{i=1}^{n}(\varphi_{p_i}\wedge\bigcirc\varphi_{f_i}),$$

其中,$\varphi_{p_i}\equiv\wedge_{j=1}^{h}\dot{p}_{ij},\varphi_{f_i}\equiv\wedge_{k=1}^{m}\varphi_{ik}$.对于任意 j,有 $p_{ij}\in\mathcal{Q}$(对于任意 $r\in\mathcal{Q}$,$\dot{r}$ 表示 r 或$\neg r$);对于任意 k,有$\varphi_{ik}\in CL(\varphi)$.

利用 PF 式可以将一个公式φ分解为当前和未来两部分,其中,当前部分 φ_{p_i} 是φ中的原子命题或它们的非的合取式,未来部分 φ_{f_i} 是 $CL(\varphi)$中的元素的合取式.这样,为了满足φ,当前状态需要满足当前部分,下一状态需要满足未来部分.通过把未来部分的公式分别转化为 PF 式,并且不断重复该转化过程,就能构造出描述φ的模型的图,即 PFG.

2.2 PFG

给定一个公式ϕ.它的 PFG,记作 G_ϕ是一个三元组(N_ϕ,E_ϕ,n_0),其中,N_ϕ是节点集,E_ϕ是边集,n_0 是根节点.N_ϕ中的每个节点都是 $CL(\phi)$中公式的合取式.E_ϕ中的每条边都是一个三元组(ϕ_i,ϕ_e,ϕ_j),其中,$\phi_i,\phi_j\in N_\phi,\phi_e$ 是边上的标签.此外,边上可能伴有记号.一个记号就是ϕ中出现的变量的一个子集.

定义 2(PFG). 给定公式ϕ.N_ϕ和 E_ϕ归纳定义如下.

(1) $n_0=\phi\in N_\phi$;

(2) 对于任意$\varphi\in N_\phi\backslash\{\text{false}\}$,若 $\varphi\equiv\vee_{i=1}^{k}(\varphi_{p_i}\wedge\bigcirc\varphi_{f_i})$, 则对任意 i,均有 $\varphi_{f_i}\in N_\phi,(\varphi,\varphi_{p_i},\varphi_{f_i})\in E_\phi$.

在一个 PFG 中,根节点用双圆形表示,其他节点用单圆形表示.每条边由一条有向弧表示,弧上伴有标签.此外,还可能伴有记号.为了表述方便,在节点中通常用变量来表示相应的不动点公式.

图 1 中展示了公式$\mu X.(p\vee\bigcirc\bigcirc X)$的 PFG.该 PFG 共有 3 个节点,其中,$n_0$ 为根节点.(n_0,p,n_1)是一条不带记号的边,其标签为 p.(n_0,true,n_2)是一条带有记号$\{X\}$的边,其标签为 true.

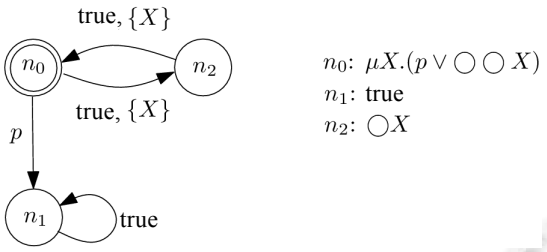

Fig.1 An example of PFG

图 1 PFG 实例

PFG 中的路径Π是指始于根节点的节点和边的无穷交替序列.每条路径均对应一个线性结构.

令 $Atom(\wedge_{i=1}^{m}\dot{q}_i)$ 表示公式 $\wedge_{i=1}^{m}\dot{q}_i$ 中的原子命题或它们的非的集合.对于一条路径$\Pi=\phi_0,\phi_{e0},\phi_1,\phi_{e1},\ldots$,可以得到一个对应的线性结构 $Atom(\phi_{e0}),Atom(\phi_{e1}),Atom(\phi_{e2}),\ldots$例如,与图 1 中路径 $n_0,p,(n_1,\text{true})^{\omega}$相对应的线性结构为 $\{p\},\{\text{true}\}^{\omega}$.

2.3 记 号

由图 1 可以看出:PFG 中的路径,例如$(n_0,\text{true},n_2,\text{true})^{\omega}$,可能产生于最小不动点公式的无穷展开.因此,在构造 PFG 的过程中,需要添加记号用以追踪不动点的展开情况.

定义 3(记号). 给定一个 PFG G_ϕ以及 G_ϕ中的一个节点φ,其中,$\varphi\equiv\vee_{i=1}^{k}(\varphi_{p_i}\wedge\bigcirc\varphi_{f_i})$, 边 $(\varphi,\varphi_{p_i},\varphi_{f_i})$ $(1\leqslant i\leqslant k)$的记号是一个变量集 M,并且满足以下两个条件.

(a) 对于 M 中的任意变量 X,与之对应的不动点公式$\sigma X.\varphi_X\in CL(\phi)$是 φ_{f_i} 的子公式;

(b) 在 PF 式的转化过程中,$\sigma X.\varphi_X$在未来部分的出现不是源于公式$\nu Y.\varphi_Y$(Y 高于 X)的展开.

2.4 ν-路径

给定一个公式ϕ.实际上,G_ϕ中的节点的每一条出边均对应于选择函数[6]的一种可能的选择.由于一个节点不能同时拥有多个选择,因此在 G_ϕ中,只需考虑结束于简单环的路径.令Π为 G_ϕ中的一条路径.为方便起见,用 $LES(\Pi)$来表示出现在Π中环的部分的边集,用 $Mark(e)$表示边 e 上的记号,用 $LMS(\Pi)$表示 $LES(\Pi)$中的边上的记号里所有μ型变量的集合.

定义 4(ν-路径). 给定一个 PFG 以及该 PFG 中的一条路径Π.称Π是一条 ν-路径当且仅当对于任意 $X\in LMS(\Pi)$,均能找到一条边 $e\in LES(\Pi)$,使得 $X\notin Mark(e)$,并且对于任意变量 $X'(X\lhd X')$,$X'\notin Mark(e)$.

由文献[15]中的定理 4 可知,一个公式是可满足的当且仅当其 PFG 中存在一条 ν-路径.因此,νTL 公式的可满足性问题得以转化为 PFG 中的 ν-路径搜索问题.本文已用 C++开发了基于 PFG 的 νTL 公式可满足性判定工具 NuTL2PFG,该工具的工作原理将在下一节进行介绍.

3 原型工具

3.1 工具架构

基于 PFG 的概念,我们开发了用于判定 νTL 公式可满足性的原型工具 NuTL2PFG,其架构如图 2 所示.

NuTL2PFG 主要包括 4 个模块,分别是语法树构造模块、PF 式转化模块、PFG 构造模块以及 ν-路径搜索模块.该工具以一个公式φ作为输入,格式为文本文件.在该文件中,公式各操作符的表示方法如下:~表示¬;|和&分别表示∨和∧;next 表示○;*mu* 和 *nu* 分别表示μ和ν.例如,公式$\nu X.(\bigcirc X\wedge\mu Y.(p\vee\neg q\wedge\bigcirc Y))$可表示为

nu X.(next *X* & *mu Y*.(*p*|~*q* & next *Y*)).

首先,语法树构造模块创建φ的语法树,在 PFG 中的所有相关操作都是在语法树上进行的;然后,PFG 构造模块开始创建φ的 PFG,在此过程中,需要不断调用 PF 式转化模块将节点转化为 PF 式,并据此生成新的节点和边,直至所有节点均已被处理;最后,ν-路径搜索模块检测所得 PFG 中是否存在ν-路径,以此来判定φ的可满足性.

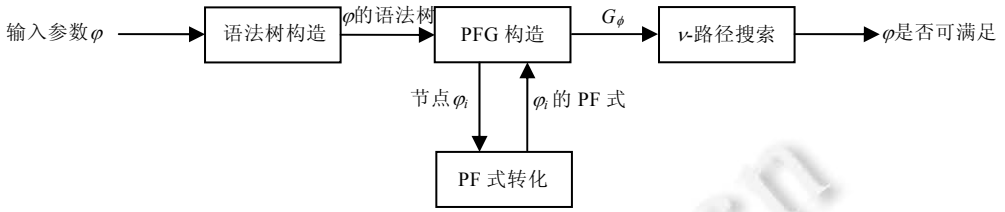

Fig.2 Architecture of NuTL2PFG

图 2 NuTL2PFG 架构

3.2 语法树构造

PFG 中所有关于节点和边的操作都是在语法树的概念上进行的.νTL 公式的语法树结构如图 3 所示.

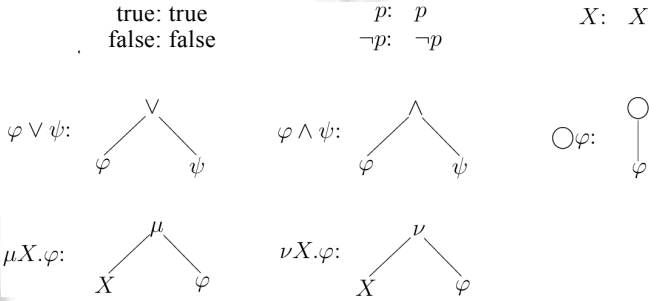

Fig.3 Syntax trees of νTL formulas

图 3 νTL 公式的语法树

由图 3 可以看出,语法树的根节点能够表示相应的节点类型.true、false、原子命题或它们的非以及变量的语法树均不含子节点.在公式$\varphi\vee\psi$(或$\varphi\wedge\psi$)的语法树中,左右孩子分别对应该公式的左右析取项(或合取项).公式$\bigcirc\varphi$的语法树只有一个孩子φ.在不动点公式$\sigma X.\varphi$的语法树中,左孩子为变量 X,右孩子为公式φ.

3.3 PF式转化

在构造 PFG 的过程中,PF 式转化模块不断被 PFG 构造模块调用,进而将 PFG 中的节点转化为 PF 式并返回给 PFG 构造模块.将一个公式φ转化为 PF 式的算法 PFTran 可递归描述如下.

(1) 当φ为 true 时,返回 true∧○true;

(2) 当φ为 false 时,返回 false;

(3) 当φ为φ_p时,其中,$\varphi_p\equiv\wedge_{k=1}^{m}\dot{p}_k$,返回$\varphi_p\wedge\bigcirc$true;

(4) 当φ为$\varphi_p\wedge\bigcirc\psi$时,返回$\vee_i(\varphi_p\wedge\bigcirc\psi_i)$;

(5) 当φ为$\bigcirc\psi$时,返回$\vee_i$(true$\wedge\bigcirc\psi_i$);

(6) 当φ为$\varphi_1\vee\varphi_2$时,返回 $PFTran(\varphi_1)\vee PFTran(\varphi_2)$;

(7) 当φ为$\varphi_1\wedge\varphi_2$时,返回 $AND(PFTran(\varphi_1),PFTran(\varphi_2))$;

(8) 当φ为$\sigma X.\psi$时,返回 $PFTran(\psi[\sigma X.\psi/X])$.

算法 AND 用于处理∧操作符.令ϕ和ψ为该算法的输入参数.由算法 PFTran 可以看出,ϕ和ψ均为 PF 式.若ϕ

形如$\vee_i(\phi_i\wedge\bigcirc\chi_i)$,$\psi$形如$\vee_k(\psi_k\wedge\bigcirc\gamma_k)$,那么算法 AND 返回$\vee_i\vee_k(\phi_i\wedge\psi_k\wedge\bigcirc(\chi_i\wedge\gamma_k))$.接下来给出用算法 PFTran 将公式 Ψ:$\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X)\vee\nu Y.(r\wedge\bigcirc\bigcirc Y)$转化为 PF 式的过程.

$$PFTran(\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X)\vee\nu Y.(r\wedge\bigcirc\bigcirc Y))\equiv$$
$$PFTran(\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X))\vee PFTran(\nu Y.(r\wedge\bigcirc\bigcirc Y))\equiv$$
$$AND(PFTran(\nu Z.(p\wedge\bigcirc Z)),PFTran(\mu X.(q\vee\bigcirc X)))\vee PFTran(r\wedge\bigcirc\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y))\equiv$$
$$AND(PFTran(p\wedge\bigcirc\nu Z.(p\wedge\bigcirc Z)),PFTran(q\vee\bigcirc\mu X.(q\vee\bigcirc X)))\vee r\wedge\bigcirc\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y)\equiv$$
$$AND(p\wedge\bigcirc\nu Z.(p\wedge\bigcirc Z),PFTran(q)\vee PFTran(\bigcirc\mu X.(q\vee\bigcirc X)))\vee r\wedge\bigcirc\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y)\equiv$$
$$AND(p\wedge\bigcirc\nu Z.(p\wedge\bigcirc Z),q\wedge\bigcirc true\vee true\wedge\bigcirc\mu X.(q\vee\bigcirc X))\vee r\wedge\bigcirc\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y)\equiv$$
$$p\wedge q\wedge\bigcirc\nu Z.(p\wedge\bigcirc Z)\vee p\wedge\bigcirc(\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X))\vee r\wedge\bigcirc\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y).$$

3.4 PFG构造

对于一个给定的输入ϕ,PFG 构造模块首先调用 PF 式转化模块,将ϕ转化为 PF 式,并据此生成新的节点和边.此后,继续调用 PF 式转化模块,将新生成的节点再分别转化为 PF 式,并重复上述过程,直至没有新的节点产生.此外,在向 PFG 中添加边时,通过检查 PF 式中未来部分的各个合取项,还需添加相应的记号,用以追踪不动点的展开情况.

值得注意的是:在构造 PFG 的过程中,可能会产生 false 节点,例如$p\wedge\neg p$.这些节点对应公式闭包的不一致性子集,公式的模型无法由这些节点产生.因此,在 PFG 构造过程的最后,还需删除这些节点以及相关的边.接下来给出构造公式 Ψ:$\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X)\vee\nu Y.(r\wedge\bigcirc\bigcirc Y)$的 PFG(如图 4 所示)的过程.

- 首先创建根节点$\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X)\vee\nu Y.(r\wedge\bigcirc\bigcirc Y)$,记作 n_0;
- 然后,将根节点转化为 PF 式:$n_0\equiv p\wedge q\wedge\bigcirc\nu Z.(p\wedge\bigcirc Z)\vee p\wedge\bigcirc(\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X))\vee r\wedge\bigcirc\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y)$,并据此生成新的节点:$\nu Z.(p\wedge\bigcirc Z)$,$\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X)$和$\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y)$,分别记作 n_1,n_2 和 n_3.与此同时,还要生成相应的边:$(n_0,p\wedge q,n_1)$,(n_0,p,n_2)和(n_0,r,n_3),它们的记号分别为$\{Z\}$,$\{Z,X\}$和$\{Y\}$;
- 之后,将节点 n_1 转化为 PF 式:$n_1\equiv p\wedge\bigcirc\nu Z.(p\wedge\bigcirc Z)$,并据此创建边$(n_1,p,n_1)$,记号为$\{Z\}$.需要注意的是,此时没有新节点产生.类似地,对于节点 n_2,可生成边(n_2,p,n_2)和$(n_2,p\wedge q,n_1)$,它们的记号分别为$\{Z,X\}$和$\{Z\}$;
- 在此之后,将节点 n_3 转化为 PF 式:$n_3\equiv$true$\wedge\bigcirc\nu Y.(r\wedge\bigcirc\bigcirc Y)$,并据此生成节点 n_4:$\nu Y.(r\wedge\bigcirc\bigcirc Y)$以及边$(n_3,$true$,n_4)$,记号为$\{Y\}$;
- 最后,对于节点 n_4,可生成边(n_4,r,n_3),记号为$\{Y\}$.此时无新节点产生.

至此,所有节点均已被处理,整个构造过程结束.

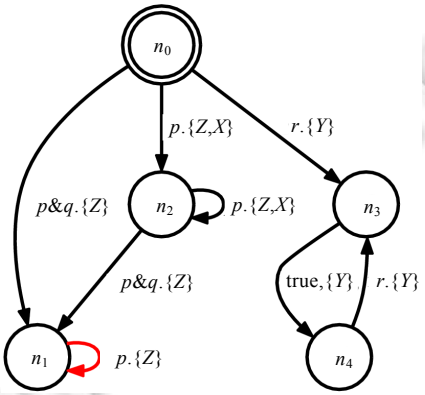

Fig.4 PFG of formula Ψ

图 4 公式Ψ的 PFG

3.5 ν-路径搜索

为了判断在一个 PFG 中是否存在ν-路径,需要储存一个边序列.在实际中,为节省空间消耗,通常先对该 PFG 用 Tarjan 算法[25]进行强连通分量分解,然后再从某个强连通分量中找出一个对应于ν-路径的环.

给定一个 PFG,令 *scc* 为该 PFG 中的一个强连通分量,*u* 为 *scc* 中一个任意节点.ν-路径搜索模块能够构造一条始于节点 *u* 的潜在ν-路径.在此过程中,需要使用全局变量 *ES*,用于存储 *scc* 中的一个边序列.该模块首先将 *scc* 中一条以 *u* 为源节点且未被访问过的边 *e* 加入至 *ES*.接着,按深度优先搜索方式继续向 *ES* 中添加其他未被访问的边.每当向 *ES* 中添加一条边后,都需要判断当前 *ES* 中是否存在一个环:若存在,且该环对应于一条ν-路径,则搜索过程结束;否则,需要删除 *ES* 中的最后一条边,然后搜索其他路径.当 *scc* 中的所有边都被访问过时,搜索过程结束.若一个 PFG 中的所有强连通分量里均不存在对应于ν-路径的环,则说明给定公式是不可满足的.

在由工具 NuTL2PFG 生成的 PFG 中,若在某个强连通分量里找到了一个对应于ν-路径的环,那么,任意终止于该环的路径(即ν-路径)均描述一个满足给定公式的模型.如果生成的 PFG 中不存在上述环,则表明给定公式是不可满足的.

对于公式 Ψ: $\nu Z.(p\wedge\bigcirc Z)\wedge\mu X.(q\vee\bigcirc X)\vee\nu Y.(r\wedge\bigcirc\bigcirc Y)$,由 NuTL2PFG 为其生成的 PFG 如图 4 所示.环$(n_1,p)^\omega$为对应于ν-路径的环,因此,该公式是可满足的.公式 Ψ表示原子命题 *p* 总是成立且 *q* 最终成立,或 *r* 在偶数时刻成立.图 4 中任意终止于环$(n_1,p)^\omega$的路径均对应满足公式 Ψ的模型.例如,路径 $n_0,p\wedge q,(n_1,p)^\omega$对应的模型为$\{p,q\},\{p\}^\omega$.由此可见:通过 PFG,能够更直观地反映出为什么一个公式是可满足的.

4 实验与分析

文献[13]中给出了针对 νTL 判定问题的当前唯一可用的工具 PS4NuTL.在该文献中,共对以下 3 类公式进行了有效性验证(实验环境为 1G 内存 PC):$Include_n$,$Nester_n$ 和 $Counter_n$.

$$Include_n \equiv \nu X.(\underbrace{q\wedge\bigcirc(q\wedge\bigcirc(...\bigcirc(q}_{2n\ times}\wedge\bigcirc(\neg q\wedge\bigcirc X))...)))\rightarrow\nu Z.\mu Y.(\neg q\wedge\bigcirc Z\vee q\wedge\bigcirc(q\wedge\bigcirc Y)),$$

$$Nester_n \equiv \psi\vee\neg\psi,\text{其中,}\psi\equiv\mu X_1.\nu X_2.\mu X_3....\sigma X_n.(q_1\vee\bigcirc(X_1\wedge(q_2\vee\bigcirc(X_2\wedge...(q_n\vee\bigcirc X_n)...)))),$$

$$Counter_n \equiv\vee_{i=0}^{n}\neg c_i\vee\mu X.(\bigcirc X\vee(c_0\leftrightarrow\bigcirc\neg c_0)\vee\vee_{i=1}^{n}(\bigcirc c_i\leftrightarrow c_i\wedge\neg c_{i-1}\vee c_{i-1}\wedge(\bigcirc c_{i-1}\leftrightarrow c_i))).$$

为了与文献[13]中的实验结果进行对比,我们用工具 NuTL2PFG 分别检查$\neg Include_n$,$\neg Nester_n$ 和$\neg Counter_n$的可满足性.实验环境为 1.73GHz,Genuine Intel(R) CPU T2080,1G 内存 PC.实验结果见表 1.在表 1 中,第 1 列表示公式中下标 *n* 的大小;节点数和边数所在列分别表示生成的相应公式的 PFG 中的节点数和边数,单位为个;时间所在列表示判断公式可满足性所需时间,单位为 ms.

Table 1 Experimental results

表 1 实验结果

n	$\neg Include_n$			$\neg Nester_n$			$\neg Counter_n$		
	节点数	边数	时间	节点数	边数	时间	节点数	边数	时间
0	6	18	0	–	–	–	2	2	0
1	17	39	31	1	1	0	4	4	0
2	28	64	63	10	30	31	8	8	16
3	39	85	124	73	386	1 185	16	16	156
4	50	106	218	601	4 640	128 559	32	32	889
5	61	127	328	5 401	55 419	18 075 924	64	64	5 117

文献[13]指出,用 PS4NuTL 检测各个公式有效性的运行时间均为几分钟.此外,在处理公式 $Nester_4$,$Nester_5$ 和 $Counter_5$ 时还遇到了内存溢出问题.这是因为,在文献[13]所用的方法中,除了需要构造公式的证明树之外,还要通过构造自动机来检测最小不动点的非良基性.然而,本文方法仅需为公式构造 PFG,并根据 PFG 中的记号便可检测最小不动点的非良基性.因此,NuTL2PFG 比 PS4NuTL 的执行效率更高.

5 结束语

作为一种语法简洁表达能力强的逻辑,νTL 在程序验证领域具有举足轻重的地位.然而,不动点操作符的嵌套使其判定问题难以得到有效解决,妨碍了其在实际中的进一步应用.针对这一问题,本文开发了νTL 公式的可满足性判定工具 NuTL2PFG.利用νTL 公式的 PF 式、PFG 以及 PFG 中的ν-路径等概念,NuTL2PFG 通过为一个给定公式构造 PFG 并从该 PFG 中找出一条ν-路径,进而判定该公式的可满足性.此外,本文通过实验证明了 NuTL2PFG 在实践中的可用性和高效性.未来有两方面的工作有待进一步研究:一方面,我们可以通过比较节点的所有出边的记号,从而更快地构造出 ν-路径,以此提升 NuTL2PFG 的性能;另一方面,我们将在 NuTL2PFG 的基础上开发基于 PFG 的νTL 模型检测工具.

References:

[1] Barringer H, Kuiper R, Pnueli A. A really abstract concurrent model and its temporal logic. In: Proc. of the POPL'86. New York: ACM Press, 1986. 173−183. [doi: 10.1145/512644.512660]

[2] Kozen D. Results on the propositional μ-calculus. Theoretical Computer Science, 1983,27(3):333−354. [doi: 10.1016/0304-3975(82)90125-6]

[3] Pnueli A. The temporal logic of programs. In: Proc. of the FOCS'77. New York: IEEE Computer Society, 1977. 46−57. [doi: 10.1109/SFCS.1977.32]

[4] Barringer H, Kuiper R, Pnueli A. Now you may compose temporal logic specifications. In: DeMillo RA, ed. Proc. of the STOC'84. New York: ACM Press, 1984. 51−63. [doi: 10.1145/800057.808665]

[5] Emerson EA, Clarke EM. Characterizing correctness properties of parallel programs using fixpoints. In: de Bakker JW, van Leeuwen J, eds. Proc. of the ICALP'80. LNCS 85, Heidelberg: Springer-Verlag, 1980. 169−181. [doi: 10.1007/3-540-10003-2_69]

[6] Vardi MY. A temporal fixpoint calculus. In: Ferrante J, Mager P, eds. Proc. of the POPL'88. New York: ACM Press, 1988. 250−259. [doi: 10.1145/73560.73582]

[7] Sistla AP, Clarke EM. The complexity of propositional linear temporal logics. Journal of the Association for Computing Machinery, 1985,32(3):733−749. [doi: 10.1145/3828.3837]

[8] Streett RS, Emerson EA. An automata theoretic decision procedure for the propositional *mu*-calculus. Information and Computation, 1989,81(3):249−264. [doi: 10.1016/0890-5401(89)90031-X]

[9] Stirling C, Walker D. CCS, liveness, and local model checking in the linear time mu-calculus. In: Sifakis J, ed. Proc. of the AVMFSS'89. LNCS 407, Heidelberg: Springer-Verlag, 1990. 166−178. [doi: 10.1007/3-540-52148-8_14]

[10] Kaivola R. A simple decision method for the linear time mu-calculus. In: Desel J Dr. rer. nat, ed. Proc. of the STRICT'95. London: Springer-Verlag, 1995. 190−204. [doi: 10.1007/978-1-4471-3078-9_13]

[11] Bradfield J, Esparza J, Mader A. An effective tableau system for the linear time *mu*-calculus. In: Meyer F, Monien B, eds. Proc. of the ICALP'96. LNCS 1099, Heidelberg: Springer-Verlag, 1996. 98−109. [doi: 10.1007/3-540-61440-0_120]

[12] Banieqbal B, Barringer H. Temporal logic with fixed points. In: Banieqbal B, Barringer H, Pnueli A, eds. Proc. of the Temporal Logic in Specification'87. LNCS 398, Heidelberg: Springer-Verlag, 1989. 62−74. [doi: 10.1007/3-540-51803-7_22]

[13] Dax C, Hofmann M, Lange M. A proof system for the linear time *mu*-calculus. In: Arun-Kumar S, Garg N, eds. Proc. of the FSTTCS 2006. LNCS 4337, Heidelberg: Springer-Verlag, 2006. 274−285. [doi: 10.1007/11944836_26]

[14] Liu Y, Duan ZH, Tian C, Liu B. Present-Future form of linear time *mu*-calculus. In: Liu SY, Duan ZH, eds. Proc. of the SOFL+MSVL 2013. LNCS 8332, Switzerland: Springer Int'l Publishing, 2014. 76−85. [doi: 10.1007/978-3-319-04915-1_6]

[15] Liu Y, Duan ZH, Tian C. An improved decision procedure for linear time mu-calculus. arXiv preprint arXiv:1507.05513, 2015.

[16] Duan ZH. An extended interval temporal logic and a framing technique for temporal logic programming [Ph.D. Thesis]. Newcastle upon Tyne: University of Newcastle upon Tyne, 1996.

[17] Duan ZH. Temporal Logic and Temporal Logic Programming. Beijing: Science Press, 2005.

[18] Duan ZH, Tian C. Decidability of propositional projection temporal logic with infinite models. In: Cai JY, Cooper SB, Zhu H, eds. Proc. of the TAMC 2007. LNCS 4484, Heidelberg: Springer-Verlag, 2007. 521−532. [doi: 10.1007/978-3-540-72504-6_47]

[19] Duan ZH, Tian C. An improved decision procedure for propositional projection temporal logic. In: Dong JS, Zhu HB, eds. Proc. of the ICFEM 2010. LNCS 6447, Heidelberg: Springer-Verlag, 2010. 90−105. [doi: 10.1007/978-3-642-16901-4_8]

[20] Duan ZH, Tian C. A practical decision procedure for propositional projection temporal logic with infinite models. Theoretical Computer Science, 2014,554:169−190. [doi: 10.1016/j.tcs.2014.02.011]

[21] Stirling C. Modal and temporal logics. Technical Report, ECS-LFCS-91-157, University of Edinburgh, Department of Computer Science, Laboratory for Foundations of Computer Science, 1991.

[22] Walukiewicz I. Completeness of Kozen's axiomatisation of the propositional μ-calculus. Information and Computation, 2000, 157(1):142−182. [doi: 10.1006/inco.1999.2836]

[23] Bruse F, Friedmann O, Lange M. On guarded transformation in the modal μ-calculus. Logic Journal of the IGPL, 2015,23(2): 194−216. [doi: 10.1093/jigpal/jzu030]

[24] Fischer MJ, Ladner RE. Propositional dynamic logic of regular programs. Journal of Computer and System Sciences, 1979,18(2): 194−211. [doi: 10.1016/0022-0000(79)90046-1]

[25] Tarjan R. Depth-First search and linear graph algorithms. SIAM Journal on Computing, 1972,1(2):146−160. [doi: 10.1137/0201010]

刘尧(1987－),男,河北唐山人,博士,CCF 学生会员,主要研究领域为时序逻辑,模型检测.

田聪(1981－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为形式化方法,时序逻辑,模型检测.

段振华(1948－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为网络计算,高可信软件理论和技术.