关于交替式下推系统可达性证明的研究与实现①

周 青[1,2]

[1](中国科学院软件研究所, 北京 100190)
[2](中国科学院大学, 北京 100190)

摘 要: 演绎推理是形式化验证中一种重要的方法, 具有可以处理无穷状态系统的优点. 本文研究与实现关于交替式下推系统中可达性的证明, 该系统可以将无穷证明树转换为有穷树. 文中首先利用迭代收敛思想实现了饱和算法, 并通过全排列算法实现了系统完备化; 然后采用余归纳方式对搜索证明进行了优化; 最后利用可视化技术对证明树在三维空间进行展示.

关键词: 推理验证; 形式化验证; 交替式下推系统; 无穷状态系统; 可视化

Research and Implementation of Reachability Proof for Alternating Pushdown System

ZHOU Qing[1,2]

[1](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Deductive reasoning is an important method in formal verification, which can address programs in infinite state systems. In this paper, a proof system is studied and implemented, where infinite proof trees transform into finite ones. Firstly, a saturation algorithm is implemented based on iterative convergence, and the completion of the saturated system is implemented using Full Permutation Algorithm; Secondly, an optimization of proof search is carried out by inductive method; At last, the proof tree is visualized in 3D space using visualization technology.

Key words: deductive reasoning; formal verification; alternating pushdown system; infinite state systems; visualization

本文在文献[1]的基础上实现了一个基于交替式下推系统[1]的证明系统. 该证明系统既可以应用于有穷状态系统, 也可以用于无穷状态系统, 并可以将无穷证明树转换为有穷树.

伴随着信息技术的高速发展, 计算机系统越来越复杂. 因此, 保证计算机系统的正确运行和安全显得尤为重要. 采用形式化方法验证系统的正确性是一种非常科学可靠的方法. 形式化方法可以用于证明一个系统不存在某个缺陷或符合某些属性. 模型检测[2-6]与逻辑推演是形式验证中非常重要的两个部分. 模型检测主要通过显式状态搜索或隐式不动点计算来验证有穷状态并发系统的模态/命题性质. 由于模型检测可以自动执行,并能在系统不满足性质时提供反例路径,因此在工业界很受推崇. 模型检测的优点是可以自动化地进行, 其中不需要人的参与, 缺点是在验证过程中随着系统复杂性增加可能会产生状态爆炸问题, 从而使验证变得低效. 相对而言, 逻辑推演在处理复杂系统的状态问题时更有优势.

有穷自动机、下推自动机、petri 网[7]等迁移系统常常被用于系统建模, 在这些系统中大部分系统分析问题可以规约到可达性问题. 因此, 可达性问题在形式化验证中是非常重要. 本文研究的是交替式下推系统[1,2,8,9]中的可达性问题. 这类问题传统的解决方法是通过在原本的下推系统之外构造一个新的有穷状态自动机, 使得一个状态在下推系统中可达当且仅当这个状态能被新构造的有穷状态自动机所接收. 文献[1]中, 作者提出了另外一种解决交替式下推系统可达性问题的方法, 即利用逻辑推演解决这些问题. 作者证明了

① 基金项目:中法 NSFC-ANR 共同资助合作研究项目 LOCALI(61161130530)
收稿时间:2016-03-11;收到修改稿时间:2016-04-08 [doi:10.15888/j.cnki.csa.005432]

一类基于交替式下推系统的推演系统的切消定理(cut elimination theorem), 并指出了在交替式下推系统中一个状态是否可达当且仅当它在交替式下推系统中具有一个不包含切(*Cut*)的证明. 文中提出的交替式下推系统是在原有的下推系统中对迁移规则做一个饱和运算, 使得一个状态是可达的当且仅当它在经过饱和后的下推系统中具有不含切(*Cut*)的证明树, 而省去了构造有穷状态自动机的过程.

本文证明系统的实现包括输入解析、下推系统的饱和过程, 饱和化[1](Saturation), 系统完备化, 利用余归纳[10]方式寻找证明树. 本文接下来, 首先在第一章介绍相关理论; 然后在第二章介绍系统框架和系统各个模块及其联系和系统关键算法的具体实现; 最后利用一个具体实例对系统进行展示.

1　理论基础

1.1 下推系统

本节基于文献[2]介绍下推系统相关理论.

1.1.1 下推系统

下推系统(Pushdown System，简写为PS)是一个三元组$\mathcal{P}$=(*P*,Γ,Δ)，其中P是控制部件的集合，Γ是一个有穷符号集，称为栈字母表，$\Delta\subseteq(P\times\Gamma)\times(P\times\Gamma^*)$是迁移规则的有穷集合.

下推系统$\mathcal{P}$的格局(configuration)是一个二元组$<p,\omega>$，其中$p\in P$是一个控制部件，$\omega\in\Gamma^*$是一个栈字符串(stack content).

如果$((q,\gamma),(q',\omega))\in\Delta$，那么对于每一个$\omega'\in\Gamma^*$，格局$<q,\gamma\omega'>$是$<q',\omega\omega'>$的直接前驱，$<q',\omega\omega'>$是$<q,\gamma\omega'>$的直接后继. 可达关系$\Rightarrow$定义为直接后继关系的自反传递闭包. $\mathcal{P}$的一个轨迹是格局的一个极大序列，其中对于任意两个连续的格局c_i和c_{i+1}，c_{i+1}是c_i的直接后继.

1.1.2 多元自动机

假设$\mathcal{P}=(P,\Gamma,\Delta)$是一个下推系统. $\mathcal{P}$的多元自动机是一个五元组$\mathcal{A}=(\Gamma,Q,\delta,I,F)$其中$Q$是有穷状态集合，$\delta\subseteq Q\times\Gamma\times Q$是迁移的集合，$I=\{s^1,...,s^m\}\subseteq Q$是初始状态的集合，$F\subseteq Q$是终止状态的集合.

迁移关系$\rightarrow\subseteq Q\times\Gamma^*\times Q$定义为满足下列条件的最小关系:

1) 如果$(q,\gamma,q')\in\delta$, 那么$q\xrightarrow{\gamma}q'$;

2) 对于任意$q\in Q$, $q\xrightarrow{\varepsilon}q$;

3) 如果$q\xrightarrow{\omega}q''$并且$q''\xrightarrow{\gamma}q'$，那么$q\xrightarrow{\omega\gamma}q'$.

对于任意$q\in F$, 如果满足$s^i\xrightarrow{\omega}q$那么称格局$<p^i,\omega>$被$\mathcal{A}$所接收或识别.

对于多元自动机$\mathcal{A}$，$\omega=\gamma_1...\gamma_n\in\Gamma^*$，满足$s^i\xrightarrow{\gamma_1}q_1...\xrightarrow{\gamma_n}q_n$的序列叫做多元自动机$\mathcal{A}$的$\omega$-轨迹.

1.1.3 交替式下推系统

在迁移系统中, 交替式下推系统(Alternating Pushdown System, 简写为 APS)是一个三元组$\mathcal{P}=(P,\Gamma,\Delta)$，其中$P$是控制部件的集合，Γ是一个有穷符号集，称为栈字母表，Δ是一个从集合$P\times\Gamma$到没有非操作的布尔公式集合的映射函数, 其中没有非操作的布尔公式是由集合$P\times\Gamma^*$中的元素构成的，Δ也可以等价的定义为迁移规则$(P\times\Gamma)\times 2^{P\times\Gamma^*}$的一个子集, 例如

$$\Delta(p,\gamma)=((p_1,\omega_1)\vee(p_2,\omega_2))\wedge(p_3,\omega_3)$$

在交替式下推系统中是如下形式:

$$\{((p,\gamma),\{(p_1,\omega_1),(p_3,\omega_3)\})((p,\gamma),\{(p_2,\omega_2),(p_3,\omega_3)\})\}\subseteq\Delta.$$

如果$\{((p,\gamma),\{(p_1,\omega_1),(p_3,\omega_3)\}),...((p,\gamma),\{(p_2,\omega_2),(p_3,\omega_3)\})\}\subseteq\Delta$那么对于任意$\omega\in\Gamma^*$，格局$<p,\gamma\omega>$是集合$\{<p_1,\omega_1\omega>,...,<p_n,\omega_n\omega>\}$的直接后驱, 并且这个集合是$<p,\gamma\omega>$的直接后继.

初始格局c在$\mathcal{P}$中的轨迹是以c为根的一棵有穷树, 树中的每个节点都是一个格局, 树中的节点c'的子节点是c'的直接后继中的格局(节点$<p,\varepsilon>$没有后继).

可达关系$\Rightarrow\subseteq(P\times\Gamma^*)\times 2^{P\times\Gamma^*}$是满足下列条件的最小子集:

1) 对于任意$c\in P\times\Gamma^*$, $c\Rightarrow\{c\}$;

2) 如果c是C的直接前驱, 那么$c\Rightarrow C$;

3) 如果$c\Rightarrow\{c_1,...,c_n\}$并且对任意$1\le i\le n$，$c_i\Rightarrow C_i$，那么$c\Rightarrow(C_1\cup...\cup C_n)$.

1.1.4 交替式多元自动机

在迁移系统中, 交替式多元自动机是一个五元组$\mathcal{A}=(\Gamma,Q,\delta,I,F)$，其中$Q$是有穷状态集合，$\delta\subseteq Q\times\Gamma\times Q$是迁移的集合，$I=\{s^1,...,s^m\}\subseteq Q$代表初始状态的集合，$F\subseteq Q$代表终止状态的集合, δ是一个从集合$P\times\Gamma$到正布尔公式集合的映射函数, δ也可以等价的定义为迁移$(Q\times\Gamma)\times 2^Q$的一个子集.

迁移关系 $\rightarrow \subseteq Q \times \Gamma^* \times 2^Q$ 是满足下列条件的最小关系:

1) 如果 $(q,\gamma,Q') \in \delta$, 那么 $q \xrightarrow{\gamma} Q'$;

2) 对于任意 $q \in Q$, $q \xrightarrow{\varepsilon} \{q\}$;

3) 如果 $q \xrightarrow{\omega} \{q_1,...,q_n\}$ 并且对于任意的 $1 \le i \le n$, $q_i \xrightarrow{\gamma} Q_i$, 那么 $q \xrightarrow{\omega\gamma} (Q_1 \cup ... \cup Q_n)$.

在 $\mathcal{A}$ 中, 对于任意格局 $< p^i, \omega >$, 如果 $s^i \xrightarrow{\omega} Q'$, 其中 $Q' \subseteq F$, 那么称 $< p^i, \omega >$ 被 A 识别. 给定一个有穷序列 $\omega \in \Gamma^*$ 和一个状态 $q \in Q$, $\mathcal{A}$ 中关于 ω 从 q 开始的轨迹是一棵有穷树, 其中树的节点用 Q 中的状态标记, 边用 Γ 中的符号标记, 根节点是用 q 标记.

1.2 证明系统

本节基于文献[1]介绍本文实现的关于交替式下推系统可达性证明的证明系统.

1.2.1 交替式下推系统和小步交替式下推系统

对于给定的语言, 交替式下推系统*(Alternating Pushdown System,* 简写为 *APS)*是一组推演规则(迁移规则)的集合, 这些推演规则具备如下的形式:

$$\frac{\mathrm{P}_1(\nu_1 x)...\mathrm{P}_n(\nu_n x)}{Q(\omega x)} \text{ 或 } \frac{}{Q(\varepsilon)}.$$

而在交替式下推系统中具备如下形式

$$\frac{\mathrm{P}_1(x)...\mathrm{P}_n(x)}{Q(\gamma x)} \text{ 或 } \frac{}{Q(\varepsilon)}$$

的规则称为引入规则; 具备如下形式

$$\frac{\mathrm{P}_1(\gamma x)\mathrm{P}_2(x)...\mathrm{P}_n(x)}{Q(x)}$$

的规则称为消去规则; 具备如下形式

$$\frac{\mathrm{P}_1(x)\mathrm{P}_2(x)...\mathrm{P}_n(x)}{Q(x)}$$

的规则称为中立规则; 其中 $\mathrm{P}_1,...,\mathrm{P}_n, Q$ 代表谓词符号, $\nu_1,...,\nu_n, \omega$ 代表栈符号, ε 代表空栈, n 为自然数.

推演系统中的证明是用格局标记的一棵有穷树, 其中对于每一个节点 N, 系统中存在一个推演规则

$$\frac{A_1...A_n}{B}$$

和一个替代函数 σ, 在 σ 中节点 N 用 σB 标记并且 N 的子节点用 $\sigma A_1,...,\sigma A_n$ 标记.

只包含引入规则、消去规则和中立规则三种形式的推演规则的交替式下推系统称为小步交替式下推系统*(Small Step Alternating Pushdown System,* 简称为 *SSAPS)*.

对于交替式下推系统中不是引入规则、消去规则和中立规则的推演规则 R, 可以在系统中引入新的谓词符号和相应的引入规则和消去规则, 并用新的谓词符号替换原有 的谓词符号从而得到一个新的中立规则 R',并用 R' 代替规则 R, 这样得到的等价系统中只包含引入规则、消去规则和中立规则三种形式的推演规则, 即得到了小步交替式下推系统. 具体转换规则的算法如下:

① 对于推演规则中具备这 $\mathrm{P}(\gamma_1...\gamma_n x)$ 形式的命题, 引入谓词符号 $\mathrm{P}^{\gamma_1}, \mathrm{P}^{\gamma_1\gamma_2}, ..., \mathrm{P}^{\gamma_1...\gamma_n}$, n 个引入规则

$$\frac{\mathrm{P}^{\gamma_1...\gamma_i\gamma_{i+1}}(x)}{\mathrm{P}^{\gamma_1...\gamma_i}(\gamma_{i+1}x)}$$

和 n 个消去规则

$$\frac{\mathrm{P}^{\gamma_1...\gamma_i}(\gamma_{i+1}x)}{\mathrm{P}^{\gamma_1...\gamma_i\gamma_{i+1}}(x)};$$

② 用 $\mathrm{P}^{\gamma_1...\gamma_n}(x)$ 代替规则 R 中的 $\mathrm{P}(\gamma_1...\gamma_n x)$, 得到一个新的中立规则 R', 并用 R' 替换 R;

③ 重复步骤①和②,直到不再产生新的规则.

1.2.2 饱和化

切消(Cut Elimination)是指消去一个证明中的切(Cut). 遗憾的是, 并非所有的交替式下推系统都具有切消性质(Cut Elimination property), 但可以在原有的系统上做一个饱和的操作, 使得饱和后的系统具有切消性质. 具体饱和算法如下:

① 如果系统系统包含一个引入规则

$$\frac{\mathrm{P}_1(x)...\mathrm{P}_m(x)}{Q_1(\gamma x)}\text{intro}$$

和一个消去规则

$$\frac{Q_1(\gamma x)Q_2(x)...Q_n(x)}{R(x)}\text{elim}$$

那么在系统中加入一个中立规则

$$\frac{\mathrm{P}_1(x)...\mathrm{P}_m(x)Q_2(x)...Q_n(x)}{R(x)}\text{neutral}$$

② 如果系统中包含一个或多个引入规则

$$\frac{\mathrm{P}_1^1(x)...\mathrm{P}_{m_1}^1(x)}{Q_1(\gamma x)}\text{intro}$$

...

$$\frac{\mathrm{P}_1^n(x)...\mathrm{P}_{m_n}^n(x)}{Q_n(\gamma x)}\text{intro}$$

并且包含一个这样的中立规则

$$\frac{Q_1(x)...Q_n(x)}{R(x)}\text{neutral}$$

那么在系统中加入一个引入规则

$$\frac{\mathrm{P}_1^1(x)...\mathrm{P}_{m_1}^1(x)...\mathrm{P}_1^n(x)...\mathrm{P}_{m_n}^n(x)}{R(\gamma x)}\text{intro}$$

③ 如果系统中包含一个或多个引入规则

$$\frac{}{Q_1(\varepsilon)}\text{intro}$$

...

$$\frac{}{Q_n(\varepsilon)}\text{intro}$$

和一个中立规则

$$\frac{Q_1(x)...Q_n(x)}{R(x)}\text{neutral}$$

那么在系统中加入一个引入规则

$$\frac{}{R(\varepsilon)}\text{intro}$$

④ 重复步骤①、②、③, 直到没有新的规则产生.

只包含引入规则的交替式下推系统叫做交替式多元自动机*(Alternating Multi-automaton,* 简写为 *AM)*. 从 SSAPS 去掉中立规则和消去规则得到交替式多元自动机.

对于交替式多元自动机, 我们采用自下向上的证明搜索方法. 在每一步证明中, 栈符号的数目都会减一, 因此, 在交替式多元自动机中可达性的证明是可判定的.

1.2.3 系统完备化

根据完备化定义[1], 我们将交替式多元自动机扩展成一个完备化系统, 使得对于一个命题 A, 要么 A 是可证的, 要么 A 是可证的. 在系统 L 中, 对于具有如下结论相同的推演规则

$$\frac{A_1^1...A_{m_1}^1}{B}$$

...

$$\frac{A_1^n...A_{m_n}^n}{B}$$

在系统中加入一个引入规则

$$\frac{A_{j_1}^1...A_{j_n}^n}{B}$$

其中 $i,n,j_1...j_n,m_1...m_i$ 都为自然数.

新加入的所有引入规则构成一个完备化系统. 在完备化系统中, 对于一个命题 A, 要么 A 是可证的, 要么 ¬A 是可证的.

1.2.4 证明树

推演系统中的一个证明就是一棵有穷证明树. 当生成的证明树比较大时, 直接将证明树输出不利于观察分析树的结构及证明过程. 本文基于蒋颖提出的定理证明三维可视化思想对证明树在三维空间的可视化进行了尝试.

实现过程中参考了信贤卫的图形显示与系统之间的交互方式[11]和一些可视化准则[12]. 系统采用 JOGL(Java Bindings for OpenGL)技术, 在三维空间渲染证明树. 采用斥力-引力模型算法[13]使树节点布局合理, 但该算法只考虑节点间的力的相互作用, 忽略了树的层次结构, 会破坏树的层次结构关系, 因此需要对算法进行改进. 系统只考虑具有父子关系的节点之间力的作用, 并通过固定特殊点坐标等方法使树的展示更加清晰, 避免了节点坐标重复等问题并保留了树的层次结构. 同时利用节点颜色的体现节点之间父子关系, 黑色代表根节点, 在每个树的分支上采用颜色插值方法, 实现从根节点到叶子节点颜色渐变的效果, 使得越靠近叶子节点的节点颜色越淡; 用户可以通过旋转从不同角度观察树的结构, 以更好的理解证明树; 用户也可以通过缩放和高亮功能, 观察某棵子树的详细信息; 事件相应功能使得用户可以通过单击右键, 选择查看节点的切(Cut)信息和该节点所表示的公式信息.

2 系统总体设计

2.1 系统设计

系统总体分成五部分: 解析输入文本、交替式下推系统到小步交替式下推系统的转化、饱和化、系统完备化以及生成证明树. 系统架构图如图 1 所示.

本系统采用 java 语言实现, 主要的类有 Configuration 类、Rule 类和 Parse 类. 在具体实现中, 用 Configuration 类代表命题(格局), Rule 类代表推演规则. 类 Configuration 的 state 和 word 属性分别代表状态和栈符号串. Rule 类的属性 premise 是一个 Configuration 对象的集合, 代表推演规则的前提, 属性 conclusion 代表推演规则的结论, 其类型为

Configuration. 用集合rule代表系统中所有规则的集合, 并用三个集合 introSet、elimSet, neutralSet 分别代表引入规则集合、消去规则集合、中立规则集合, 并用 map 结构对推演规则做分类处理. Parse 类是一个用于解析输入文本的工具类.

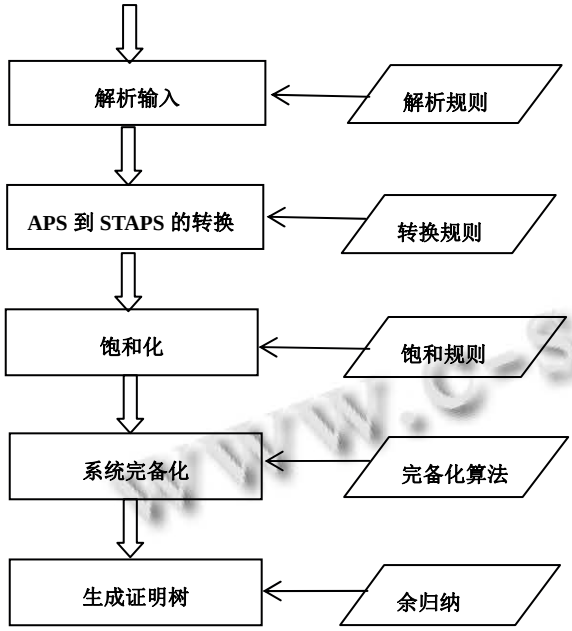

图 1 系统架构图

系统的输入文本是用文本格式表示的交替式下推自动机, 本文提出了一种直观通用的文本格式在文本中表示交替式下推系统. 输入文本中符号"—>"的左侧代表推演规则的前提, 右侧代表推演规则的结论, 在前提中命题(格局)用符号"&"连接, 栈符号使用英文逗号隔开, 并用符号"#"表示空栈 ε.

对于输入文本的处理如下:

1) 将文本按行读入, 并去除空格等无意义符号;

2) 检查字符串格式是否正确. 否则, 则解析失败;

3) 对字符串进行解析, 生成前提和结论;

4) 解析前提和结论中的命题, 并将所有的状态与栈符号分别加入状态集合与栈符号集合, 得到推演规则的前提和结论, 并构造出推演规则;

5) 根据推演规则的类别, 将其加入不同的集合.

解析输入文本得到交替式下推自动机后, 首先将其按照章节 1.2.1 介绍的转换规则算法, 将交替式下推自动机转化为小步交替式下推系统; 接下来利用章节 1.2.2 中的饱和算法, 在小步交替式下推系统下做一个饱和运算得到交替式多元自动机; 最后利用章节 1.2.3 中的完备化算法, 得到一个完备化的交替式多元自动机, 并利用章节 1.2.4 中的可视化理论对证明树进行三维展示.

2.2 关键算法实现

2.2.1 饱和算法

切消去过程是通过在原有系统中做一个饱和过程实现的.

饱和算法的实现分为以下三步:

① 遍历消去规则的集合, 在遍历消去规则过程中, 首先找到规则前提中包含消去栈符号的格局; 其次在引入规则集合中找到结论中含有该格局的引入规则; 然后用该引入规则的前提代替消去规则中包含消去栈符号的格局生成的新的前提; 最后用新的前提和消去规则的结论构造一个中立规则.

② 遍历包含键值对<栈符号\空栈符号, 引入规则>的 map, 对于每一个键 γ, 首先得到其对应的值——引入规则集合 introRuleSet; 其次, 取出该集合中规则的结论, 并用结论中的状态构成一个状态集合 introStateSet; 然后遍历中立规则集合, 找到一个前提中的状态集合是 introStateSet 的子集的中立规则 n; 最后以 introRuleSet 中的所有规则前提的并集作为新的前提, 以 n 中结论的状态 R 和键 γ 构成的新格局 R(γ x)作为新的结论, 并用新的前提和结论构成一个新的引入规则.

③ 首先利用包含键值对<栈符号\空栈符号, 引入规则>的 map, 取得空栈对应的所有引入规则; 然后利用规则中结论的状态, 构成一个状态集合 introStateSet; 最后遍历中立规则集合, 对于每一个前提中的状态的集合是 introStateSet 的子集的中立规则 n, 以 n 中结论中结构的状态 R 作为新规则中结论结构的状态, 构成新的格局 R(ε), 并用 R(ε)为结论、公理为前提构造新的引入规则.

不断重复上述步骤, 直到没有新的规则产生.

2.2.2 完备化算法

按键遍历包含键值对<栈符号\空栈符号, 引入规则>的 map, 对于含有同一结论 B 的所有推演规则, 计算出其前提中的所有结构集合的全排列, 然后以每一个排列作为前提, B 为结论, 构造一个引入规则. 其中全排列算法利用递归方式实现. 具体实现中, 一个排列是格局的集合, 并用一个链表 consList 存储所有的排列即所有的格局集合. 在递归过程中用一个栈结构

存储中间结果及一个变量 level 记录当前遍历的层次. 当当前遍历的层次 level 等于层数(即规则的数目)的时候, 栈中的内容即为一个排列, 将栈中的内容加入到结果集合 list 中, 并将栈中的内容清空, 并继续依此搜索.

递归的核心伪代码如下:

```
1    Permutation(list, level)
2      for configuration in consList.get(level)
3        stack.push(configuration)
4        if level == n - 1
5          then
6            list.add(stack) //栈中的内容加入 list
7              stack.empty()
8        else
9          Permutation(list,level + 1)
10       stack.pop()
11   End
```

2.2.3 深度优先搜索

在证明过程中采用余归纳的方式, 寻找证明树. 余归纳的思想在代码中体现为深度优先搜索. 具体算法流程图如图 2 所示.

对于给定命题, 在系统中找到能够推导出该命题的规则, 然后依次继续证明该规则前提中的所有命题. 一个命题证明的结束是指推导出该命题的规则的前提是公理. 当一个命题的前提全部证明结束时, 则该命题证明结束, 即证明树所有证明分支证明结束时, 该证明树的证明结束.

2.2.4 斥力-引力模型

在搜索生成证明树的过程中, 树的渲染模式是一边生成一边渲染的. 但当节点较多时, 会出现不同节点坐标重复, 树结构混乱等问题, 导致最终的生成树对用户来讲是难以理解的. 因此本文采用斥力-引力模型解决这个问题.

在斥力-引力模型中, 算法模拟物体间的斥力与引力, 假设所有节点间都具有斥力和引力, 根据计算每个节点的合力调整节点位置. 在该模型中引力计算公式为:

$$F_a(n_1,n_2)=d(n_1,n_2)$$

斥力计算公式为:

$$F_r(n_1,n_2)=k_r\frac{d'(n_1,n_2)}{d(n_1,n_2)^{3/2}}$$

其中 $d(n_1,n_2)$表示节点 n_1, n_2 的距离, $d'(n_1,n_2)$ 表示点在三维空间某一维度上的距离.

在具体实现中, 斥力因子设为 3, 引力因子设为 1, 整个算法分为三个部分, 第一部分是计算每个点的斥力, 第二部分计算每个点的引力, 第三部分根据点的斥力与引力调整点的坐标. 在具体实现中力与点的位置坐标一样用一个三维向量表示, 每个维度力的大小用上述公式计算.

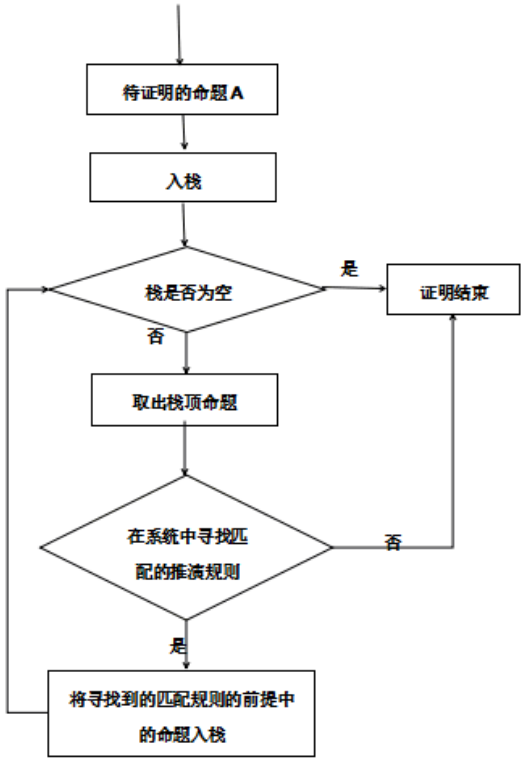

图 2　余归纳方式搜索算法流程图

在斥力(引力)的计算中, 对于每个点, 计算其与其他点的斥力(引力), 并对所有斥力(引力)求和即为该点所受到的斥力(引力). 对于最后点坐标的调整, 调整幅度为力与 0.01 的乘积, 其中 0.01 是通过实验不断调整得到.

3　实例展示

为了展示系统可以将一个在一般下推系统中的无穷证明转化为一个有穷证明的, 其证明效率要优于一般下推系统.

本文通过一个链表问题展示证明系统, 这个链表

问题描述如下: 对于一个链表, 定义一个插入和一个删除操作, 一次插入操作可以向链表中添加一个元素, 一个删除操作可以从链表中删除一个元素. 其状态图如图 3.

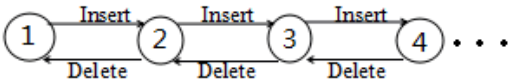

图 3 链表问题状态图

图 3 中, 状态 1 代表初始状态, 状态 2 代表插入一个元素后的状态, 其他状态依此类推, 迁移关系如图 3 所示, 且该图的状态的数目是无穷的.

将该链表问题抽象成下推系统, 得到一个链表系统 $\mathcal{P}$ 如下.

$\rightarrow s(\varepsilon)$
$s(x) \rightarrow s(ax)$
$s(ax) \rightarrow p(x)$
$p(ax) \rightarrow p(x)$
$p(x) \rightarrow s(ax)$

$\mathcal{P}$ 中的四条推演规则依次表示系统的初始状态为 $s(\varepsilon)$、插入一个元素、在插入一个元素后删除一个元素、在删除一个元素后再删除一个元素以及在删除一个元素后插入一个元素.

从图 3 与 $\mathcal{P}$ 中可以看出链表问题的状态是无穷的, 且该状态图中是存在环的, 所以对于一个命题的证明, 该证明会陷入一个无穷循环, 使得证明是一个无穷的状态迁移, 从而使得命题无法得到证明. 例如对于命题 $p(\varepsilon)$ 的证明, 如果系统 $\mathcal{P}$ 首先选择规则 $s(ax) \rightarrow p(x)$, 然后则需要证明 s(a), 如果系统 $\mathcal{P}$ 选择 $p(x) \rightarrow s(ax)$证明 s(a), 那么接下来需要证明 $p(\varepsilon)$, 这样就陷入了一个不断选择规则 $s(ax) \rightarrow p(x)$和规则 $p(x) \rightarrow s(ax)$导致不断证明 $p(\varepsilon)$和 s(a)的无穷循环.

在 $\mathcal{P}$ 中, 对于命题 $p(\varepsilon)$的证明输出进行可视化得到可视化结果如图 4,5,6 所示, 证明过程陷入了无限循环, 节点不断增多, 无法停止.

将 $\mathcal{P}$ 作为系统输入, 得到一个只包含引入规则的交替式多元自动机 $\mathcal{P}'$, 如下所示.

$\rightarrow p(\varepsilon)$
$s(x) \rightarrow s(ax)$
$s(x) \rightarrow p(ax)$
$\rightarrow s(\varepsilon)$
$p(x) \rightarrow s(ax)$

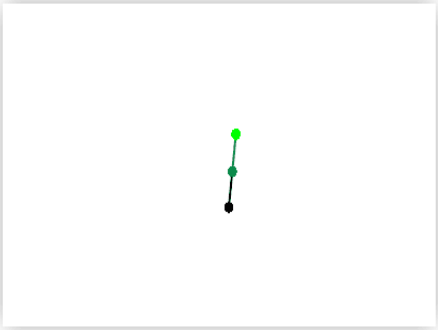

图 4 系统无穷证明 0.01 秒时可视化输出

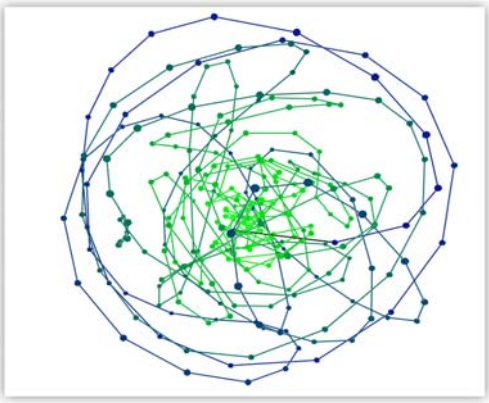

图 5 系统无穷证明 1 秒时可视化输出

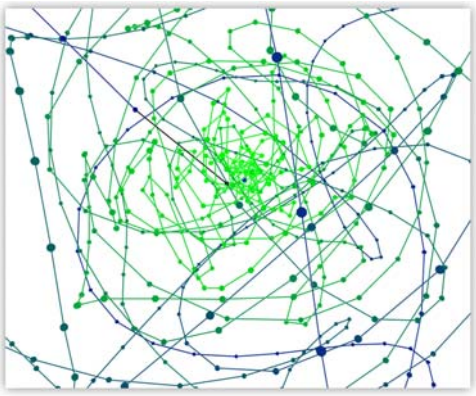

图 6 系统无穷证明 3 秒时可视化输出

对于命题 $p(\varepsilon)$的证明, 如图 7 所示, 系统 $\mathcal{P}'$ 会选择规则 $\rightarrow p(\varepsilon)$, 从而使得该命题得以证明, 证明过程终止. 这样就将命题 $p(\varepsilon)$的无穷证明树转换成了有穷树. 将用户可以右击节点以查看节点其他信息以及从不同角度查看该证明树. 其中符号#代表上文中的 ε.

由于系统中只包含引入规则, 每一步证明栈中的数目减一, 所以该系统是可终止的, 即没有循环.

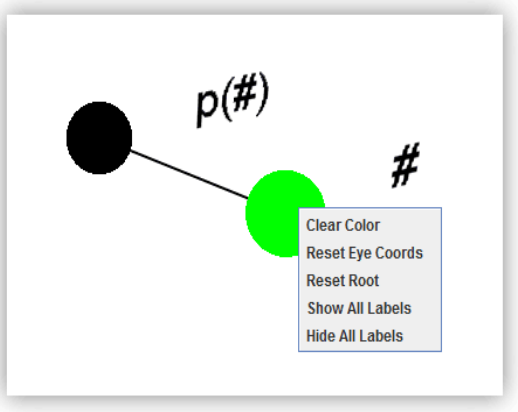

图 7 系统有穷证明可视化输出

4 结语

本文对利用自然推演方式研究交替式下推系统可达性问题的方法进行了实现. 首先, 系统读入文本格式的输入文件, 解析输入文件并得到交替式下推系统; 然后, 将交替式下推系统转换成小步交替式下推系统, 通过循环迭代的方式对系统进行饱和操作直至规则收敛, 得到多元自动机; 接下来利用全排列算法实现系统的完备化, 得到完备化的多元自动机, 并利用深度优先算法实现余归纳方式的搜索证明; 最后, 对搜索证明产生的证明树采用基于斥力-引力模型的算法在三维空间进行布局和展示. 对于交替式下推系统来说, 本文实现的系统可以将无穷证明树转化为有穷树, 从而提高验证效率.

接下来的工作主要是扩充和完善可视化部分, 进一步提高系统性能.

参考文献

1 Dowek G, Jiang Y. Cut-elimination and the decidability of reachability in alternating pushdown systems. Eprint Arxiv, 2014.

2 Bouajjani A, Esparza J, Maler O. Reachability analysis of pushdown automata: Application to model-checking. LNCS. 1997: 135–150.

3 Mcmillan KL. Symbolic model checking. States & Beyond Information & Computation, 1993, 1(1-2): 11–40.

4 Song F, Touili T. Pushdown model checking for malware detection. International Journal on Software Tools for Technology Transfer, 2012, 16(16): 110–125.

5 Finkel A, Willems B, Wolper P. A direct symbolic approach to model checking pushdown systems. Personal Communication, 1997.

6 Walukiewicz I. Pushdown processes: Games and model checking. CAV'96. LNCS1102. 1996.

7 Dalen DV. Logic and structure. Springer Berlin, 2013, (515): 344.

8 Ladner RE. Alternating pushdown and stack automata. Siam Journal of Computer, 1984, 13(1): 135–155.

9 Esparza J, Hansel D, Rossmanith P, et al. Efficient algorithms for model checking pushdown systems. CAV. Springer Berlin Heidelberg, 2000: 232–247.

10 Sangiorgi D. An introduction to bisimulation and coinduction. Introduction to Bisimulation & Coinduction, 2012, 31(4): 824–833.

11 信贤卫.具体反例生成与图形化显示系统.计算机系统应用,2013,22(11):51–57.

12 Libal T, Riener M, Rukhaia M. Advanced proof viewing in prooftool. Eprint Arxiv, 2014, 167.

13 Jacomy M, Venturini T, Heymann S, et al. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. Plos One, 2014, 9(6): e98679.