基于扩展 RED 图的概率时间自动机可达性分析①

纪 玮[1,2], 王 凡[3], 吴 鹏[1]

[1](中国科学院软件研究所 计算机科学国家重点实验室, 北京 100190)
[2](中国科学院大学, 北京 100080)
[3](国立台湾大学, 台北 10617)

摘 要: RED 图可以表示一个完整的时间自动机上的状态集, 包括其连续时间部分和离散部分. 在它基础上实现的模型检测工具 RED, 在时间自动机模型检测中表现出了优良的性能. 另一方面, 现有的概率时间自动机模型检测工具仍然使用不同的方法来分别表示概率时间自动机状态的连续时间和离散部分. 我们在复用原始 RED 图的数据结构的基础上, 对其做出了扩展, 以令其支持概率状态的表达, 同时保持其性能方面的优势. 我们又为此实现了一个概率时间自动机可达性分析工具原型, 并将其与两个概率模型检测工具(PRISM 和 Modest)就概率时间自动机可达性分析作实验对比, 来评估该工具原型的性能. 实验结果显示, 我们的集成表示概率状态空间的方式, 确实提高了概率时间自动机模型检测的时间效率和延展性.

关键词: 概率时间自动机; 可达性分析; RED 图; 扩展 RED 图

Reachability Analysis of Probabilistic Timed Automata Based on Extended RED Diagrams

JI Wei[1,2], WANG Fan[3], WU Peng[1]

[1](State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Science, Beijing 100190, China)
[2](University of Chinese Academy of Science, Beijing 100080, China)
[3](National Taiwan Univerty, Taipei 10617, China)

Abstract: RED diagrams represent states of a timed automata in a single and integrated diagram for both the dense-time and the discrete parts of the states. This integration contributes greatly to the considerable efficiency of the model checker RED in verification of timed automata. On the other hand, the state-of-the-art model checkers for probabilistic timed automata (PTAs) still use different representations for the dense-time and the discrete parts of probabilistic states. This paper proposes an elegant way to extend RED diagrams to represent probabilistic states, yet reuses the structures of original RED diagrams and hence preserves their efficiency. It also implements a prototype tool for reachability analysis of PTAs based on the extended RED diagrams, while the PTA benchmarks distributed within the probabilistic model checkers PRISM and Modest are used to evaluate its performance. Experimental results show that our integrated representation of probabilistic state space can indeed help improve the time efficiency and scalability for PTA reachability analysis.

Key words: probabilistic timed automata; reachability analysis; RED diagrams; extended RED diagrams; probabilistic models

RED 图在时间自动机的模型检测中显示了优良的性能[1-4]. 一个 RED 图包含了 clock restriction diagrams (CRDs)[5]和其他决策图, 分别用以表示时间自动机状态的连续时间和离散部分. 这种集成表示的方式使时间自动机状态之间可以高效地共享数据结构存储,同时保持了 CRD 在时钟变量处理上的优势[2]. 例如, 与 clock difference diagrams(CDDs)[6]和 difference decision diagrams(DDDs)[7]相比, CRD 在交、并运算上拥有更

① 基金项目:国家高技术研究发展计划(863)(2012AA010905);国家自然科学基金(61370081)
收稿时间:2016-02-23;收到修改稿时间:2016-03-28 [doi:10.15888/j.cnki.csa.005414]

具时空效率的算法[5,8]. 所有这些特性, 对于时间自动机模型检测工具 RED 的总体性能, 具有至关重要的作用[9].

本文, 我们将扩展 RED 图来对概率时间自动机进行可达性分析. 我们观察到, 现有的概率时间自动机模型检测工具仍然使用不同方法来分别表示概率时间自动机状态的连续时间和离散部分. 例如, PRISM[10,11]使用 difference bound matrices(DBMs)[12]来描述时钟, 用显式状态或符号表示如 multi-terminal Boolean decision diagrams(MTBDDs)[13]来描述离散状态.

我们在扩展 RED 图中, 引入一种新的结点来表示概率信息. 在分析概率时间自动机的过程中, 其状态可以被原始 RED 图很好的表示, 如同时间自动机状态一般. 我们在原始 RED 图上额外增加了一个特殊的概率类型的根结点, 以及从这个新的根节点直达原来的根结点的边. 在每一条通向原来根结点的新边上, 我们标记上到达原来根结点所代表的状态集的概率. 通过这种方式, 我们可以一种统一而简洁的方式来表达概率时间自动机的状态及其关联概率信息, 而不需要打破原始 RED 图的结构. 因此, 现有的支持 RED 图操作的编程库(如 REDLIB[14])也就可以被重用和扩展, 以支持扩展 RED 图的操作. 这一点可以被我们基于扩展 RED 图和 REDLIB 所实现的概率时间自动机可达性分析工具原型所证实.

我们利用知名的概率模型检测工具 PRISM[10,11]和 Modest[15,16]所发表的概率时间自动机测试用例来评估扩展 RED 图和所实现的概率可达性分析工具的性能. 我们利用来自网络及安全领域的概率时间自动机模型, 设计并实施了两类实验. 一类用于考查扩展 RED 图在模型参数或数据域增大情况下的表现; 另一类用于考查其在模块数量增多情况下的表现. 一个扩展 RED 图的大小, 在模型的状态变量增多或其参数/数据域增大的情况下, 都可能会增加. 实验结果显示, 扩展 RED 图在概率时间自动机的可达性分析方面拥有更好的时间效率. 和 PRISM 相比, 我们的工具原型分别在 CPU 时间和总时间的消耗上, 平均分别减少了 63.09%和 65.76%. 和 Modest 相比, 我们的工具原型在 CPU 时间和总时间的消耗上, 平均分别减少了 62.73% 和 65.23%. 与此同时, 扩展 RED 图在处理大规模概率时间自动机模型时, 表现出了良好的延展性(Scalability). 如果不在 PRISM 和 Modest 上采取优化技术的话, 我们的工具与之相比, 可以处理更大规模的概率时间自动机模型. 这意味着扩展 RED 图所采用的集成表示方法, 确实可以提高概率时间自动机可达性分析的时间效率和延展性.

状态空间符号表示, 如 Binary decision diagrams (BDDs)及其衍生结构, 在软硬件的验证工作中取得了巨大的成功. 我们的工作则是受到了时间自动机模型检测工具 RED 中的类 BDD 图的启示. 这种图在本文中被称为 RED 图. RED 图包含了用于表示连续时钟的 CRD 和用于表示离散状态的 multi-value decision diagrams (MDDs). CRD 与 BDD 类似, 区别在于在前者中, 结点是时钟差值, 而差值的上界则是边上的标记. MDD 同样和 BDD 类似, 区别在于在前者中, 变量可以是离散数据类型, 边上的标记可以是区间. RED 图的弹性和集成性的特点, 使其可以扩展后成为集成表示概率时间自动机状态及其关联概率信息的一种时空效率很高的选择.

DBM 是一种典型的用于表示时钟变量的符号表示方法, 它被广泛地应用于当今主流的时间自动机或概率时间自动机模型检测工具(如 UPPAAL[17], PRISM[10,11])中. 对于概率模型, PRISM 对相关概率信息使用不同的表示方法. 其中包括 MTBDD[13], 和我们的表示方法很接近. 它扩充了 BDD, 允许多个叶子结点(terminals)的存在. 在本文中, 我们会通过实验数据, 说明这种扩展方式对 RED 图来说并不适合.

本文后续部分的结构如下: 在第 1 节, 我们简述时间自动机、概率时间自动机和 RED 图的定义. 在第 2 节, 我们提出 RED 图的扩展. 在第 3 节, 我们阐述基于扩展 RED 图的可达性分析算法. 在第 4 节, 我们给出详细的实验数据, 并将我们的工具原型和 PRISM、Modest 作出详尽的对比. 最后在第 5 节我们总结全文并规划下一步工作.

1 准备知识

在这一部分我们简述时间自动机[8]的定义及其状态空间的符号表示, 以及概率时间自动机[18,19]和 RED 图的定义. 我们使用以下术语和记号.

Chan 是通道集, 其中通道用 *ch* 表示. *Act* 是动作集, 其中动作用 α 表示. α可以是 *ch?*或 *ch!*($ch \in Chan$), 其中, 动作 *ch?* 向通道 *ch* 发送一个信号, 动作 *ch!*从通道 *ch* 接收一个信号.

C 是实数值的时钟变量集, 其中时钟用 c 表示. 对时钟变量只能读值, 或者重置为 0. 一个时钟约束 b 是原子命题的布尔组合, 这些原子命题可以是 $c\sim v$ 或 $c_1-c_2\sim v$ 的形式, 其中, $v\in\mathbb{R}_0^+$且$\sim\in\{<,\le,>,\ge,=\}$. 令 $\mathcal{B}$ 表示时钟集合 C 上的时钟约束集.

定义 1(时间自动机). 一个时间自动机可以用四元组 $A=(L,T,l_0,I)$表示. 其中, L 是一个确定的位置集合, $l_0\in L$ 是初始位置, $I:L\to\mathcal{B}$将每个位置与一个时钟不变式相关联, $T\subseteq L\times\mathcal{B}\times Act\times 2^C\times L$ 是一个确定的迁移集合.

迁移$(l,b,\alpha,D,l')\in T$, 表示当处于位置 l 时, 只要时钟约束 b 成立, 时间自动机 A 就可以进行动作 α, 重置 D 中的时钟, 并在不违反时钟不变式 $I(l')$的情况下到达位置 l'. 当然, A 也可以停留在位置 l, 只要时钟不变式 $I(l)$持续被满足.

令 $Dist(S)$表示集合 S 上的概率分布函数集合, 即 $Dist(S)=\{\mu:S\to(0,1]|\Sigma_{s\in S}\mu(s)=1\}$. 对概率分布函数 $\mu\in Dist(S)$, 定义 $sup(\mu)=\{s\in S|\mu(s)>0\}$.

概率时间自动机在时间自动机上作出了扩充, 来描述实时系统中的概率和随机行为[18].

定义 2(概率时间自动机). 一个概率时间自动机可以用四元组$\mathcal{A}=(L,\ ,l_0,I)$表示. 其中, L, l_0 和 I 与在定义 1 中相同. $\subseteq L\times\mathcal{B}\times Act\times Dist(2^C\times L)$是一个确定的概率迁移集合.

概率迁移$(l,b,\alpha,\mu)\in$, 表示当处于位置 l 时, 只要时钟约束 b 成立, 概率时间自动机$\mathcal{A}$就可以进行动作 α, 重置 D 中的时钟, 并在不违反时钟不变式 $I(l')$的情况下, 以 $\mu(D,l')$的概率, 到达位置 l'. 同样, $\mathcal{A}$也可以停留在位置 l, 只要时钟不变式 $I(l)$持续被满足.

上面的定义中, 只描述了时钟变量. 数据变量可以像 UPPAAL[17], PRISM[10,11]和 Modest[15,16] 所支持的模型语言一样被引入. 我们用 ι 表示离散数据类型, 它可以是布尔型、整型或浮点数据类型. 令 Var_ι和 Val_ι 分别为 ι 类型的数据变量的集合和数值的集合. 我们用 Var 和 Val 分别表示变量的集合和数值的集合, 这里的变量可以是任意类型.

这样, 一个时间自动机 $A=(L,T,l_0,I)$上的混合状态就可以用三元组(l,η,ρ)来表示. 其中 $l\in L$; $\eta:C\to\mathbb{R}_0^+$是对时钟的赋值; $\rho:Var\to Val$ 是对数据变量的赋值. 不严格地说, η 构成了这个状态的连续时间部分, 而 l 和 ρ 共同构成了它的离散部分. 时钟通常被符号化表示(例如 CRD), 而离散状态也可以用各种决策图(decision diagrams)来符号化表示. RED 图包含了这些决策图, 因此可以在一个单一而集成的结构中表示时间自动机状态的连续时间部分和离散部分[1,20]. 令 AP 为时钟和数据变量上的原子命题的集合. 这样, RED 图就可以按以下的方式被形式定义.

定义 3(RED 图[2]). 一个 RED 图 G 是一个有向无环图(N,E,δ,γ). 其中, N 是确定的结点集合, E 是确定的、定义于 $N\times N$ 上的多重边集(multiset of edges), 且

$\delta:N\to Var\cup\{c,-c,c-c'|c,c'\in C\}\cup true$ 将结点 $v\in N$ 与一个变量, 一个时钟表达式, 或是布尔常量 $true$ 关联起来;

$\gamma:E\to 2^{AP}$ 在每条边$(n,n')\in E$ 上以如下形式标记原子命题: 1)若 $\delta(n)$是时钟表达式, 标记 $\delta(n)\le v$, 其中 $v\in\mathbb{R}$ 或 $\delta(n)<\infty$; 2)若 $\delta(n)$是一个 ι 类型的变量, 即 $\delta(n)\in Var$, 标记 $\delta(n)\sim v$, 其中 $v\in Val_\iota$ 且 $\sim\in\{<,\le,>,\ge,=\}$.

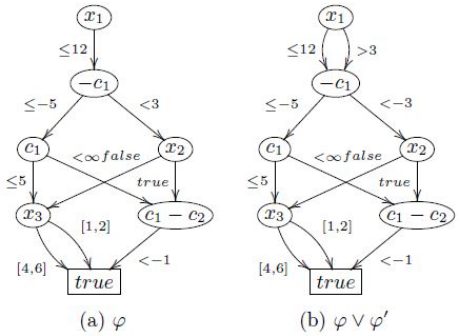
结点$n\in N$被称为G的一个根结点, 如果其入度为 0, 即不存在 $n'\in N$ 使得$(n',n)\in E$;

结点$n\in N$被称为G的一个叶子结点, 如果其出度为 0, 即不存在 $n'\in N$ 使得$(n,n')\in E$;

G 中唯一的叶子结点是与布尔常量 $true$ 相关联的结点.

给定 RED 图 $G=(N,E,\delta,\gamma)$, 令 $roots(G)\subseteq N$ 表示 G 的根结点的集合. 那么, G 所表示的符号约束 $\|G\|$ 则是每个根结点 $r\in roots(G)$表示的符号约束 $\|r\|$ 的析取式, 其中若 $\delta(n)=true$, 则 $\|n\|=true$; 否则 $\|n\|=\vee(\gamma(n,n')\wedge\|n'\|)$,其中$(n,n')\in E$.

例如, 图 1(a)展示了一个 RED 图, 该图表示的符号约束为 $\varphi\equiv(x_1\le 12)\wedge(\psi_1\vee\psi_2)$, 其中 $\psi_1\equiv c_1>3\wedge((x_2\wedge c_1-c_2<-1)\vee((\neg x_2)\wedge(4\le x_3\le 6\vee 1\le x_3\le 2)))$,$\psi_2\equiv c_1\ge 5\wedge(c_1-c_2<-1\vee(c_1\le 5\wedge(4\le x_3\le 6\vee 1\le x_3\le 2)))$.



(a) φ (b) $\varphi\vee\varphi'$

图 1 两个 RED 图

为了表示的简洁性, 我们在边上的标记中省略了时钟表达式和数据变量, 只保留了常量(代表等式约束)和区间(代表不等式约束).

一个图 G 实际上表示了对所有满足符号约束 $\|G\|$ 的时间自动机状态的集合. 在下文中, 我们用符号状态一词来代指这样的状态集.

2　RED图的概率扩充

概率时间自动机$\mathcal{A}=(L, \ ,l_0,I)$中的一个混合状态可以用三元组$(l,\eta,\rho)$表示. 其中, l, η 和 ρ 与时间自动机中混合状态的定义是一致的. 概率时间自动机中的混合状态需要额外的概率信息来对模型进行量化验证. 由于 RED 图可被用来表示时间自动机的符号状态, 我们可以扩展 RED 图, 使其成为一种集成而紧致的概率状态空间表示方法. 这需要将概率时间自动机状态的概率信息引入 RED 图.

假设符号状态 φ 所对应的概率值为 *0.6*. 这样的概率信息可以通过以下方式集成到符号状态 φ 对应的 RED 图上: 即为概率值引入一个特殊类型的结点. 下面是两种可能的解决方案, 以尽可能少地改变原始 RED 图共享结构:

方案一: 将原始图的叶子结点上的布尔常量 *true* 替换为概率值.

方案二: 在原始图上增加新的表达概率信息的根结点.

由于两种方案之间是对称的, 它们并没有本质的差别. 但在实际操作中, 两种方案在与原始图的兼容性上存在很大的差别. 这一点通过图 1(b)的例子可以被说明. 该例展示了一个符号状态 $\varphi\vee\varphi'$, 其中 $\varphi'\equiv(x_1>3)\wedge(\psi_1\vee\psi_2)$对应的概率值为 *0.3*.

显然, 如图 1(b)所示, $\varphi\vee\varphi'$以通过合并 φ 和 φ'对应的两个 RED 图来表示. 可以看出在 $\varphi\vee\varphi'$对应的 RED 图中, 几乎所有 φ 和 φ'的结点和边都是共享的. 当概率值被引入时, 我们需要在 φ 和 φ'的基础上, 用一种紧致的方式来描述集合$\{(0.6,\varphi), (0.3,\varphi')\}$.

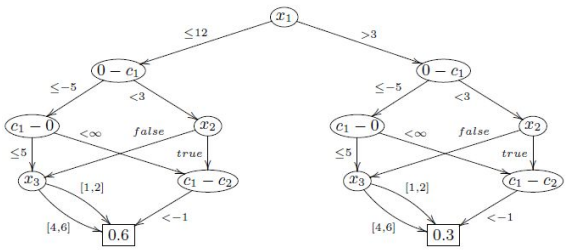


图 2　概率扩展方案一

在第一种方案中, 图 1(b)中的叶子结点 *true* 被替换为两个分别拥有独立的概率值的叶子结点, 如图 2 所示. 这和从 BDD 扩展到 MTBDD 的方式类似[13]. 由于所关联的概率值不同, 图 1(b)中共享的结点需要被复制分割. 该方案存在的问题是原始 RED 图的结构在扩展图中被改变了, 且原始的单一叶子结点也被替换了. 多个叶子结点的存在给扩展图中的结点的含义带来了不一致性. 在图 2 中, 符号状态结点 x_1 的含义事实上取决于所关注的概率叶子结点, 而图 1(b)中对应的结点则明确地代表 $\|x_1\|$.

在第二种方案中, 图 1(b)通过增加一个新的概率根结点 *PROB* 来进行扩展. 这样, 我们就可以把相关联的概率信息标记在连接新根结点和原根结点及其拷贝的新边上, 如图 3 所示. 可以看出, 图 1(a)和图 1(b)实质上是图 3 的子图. 因此, 这种方案可以很好地保持原始 RED 图的结构, 并且除了新增的根结点, 扩展图中所有结点的含义和在原始图中保持一致. 这使得我们可以复用已有的 RED 图的操作算法. 通过这种方案, RED 图原有的优势可以得到很好的保留.

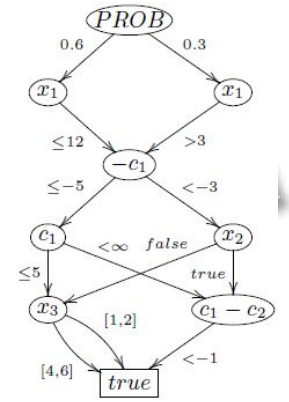


图 3　概率扩展方案二

根据以上的分析, 我们采用第二种方案来扩展 RED 图. 我们称扩展后的图为概率 RED 图, 并形式定义如下.

定义 4(概率 RED 图). 一个概率 RED 图是一个有向无环图 $=(N\cup\{r\},E\cup E_r,\delta,\gamma,v)$. 其中, N, E, δ, γ 和在定义 3 中相同. $r\notin N$ 是概率 *PROB* 类型的根结点, E_r 是 $\{r\}\times N$ 上确定的边集, $v:E_r\rightarrow(0,1]$为 E_r 中的每一条边关联一个概率值.

类似地, 一个概率 RED 图 可以看作一个<符号状态, 概率值>序偶的集合 ‖　‖, 即

$\| \quad \| = \{(v(n), \| n \|) | (r,n) \in E_r\}$

这样扩展之后, 我们就可以基于现有的 RED 图编程库 REDLIB[14,21], 来开发概率时间自动机的可达性分析和模型检测算法.

需要说明的是, 本节中给出的例子 $\varphi \vee \varphi'$并不意味着方案一总是比方案二带来的图更大. 结点的复制分割在方案二中仍然可能发生. 复制分割本质上是由变量的排序所决定的, 这在 BDD 及其类似的结构中是一个常见的问题. 方案一存在的问题也反映了 MTBDD 本身的局限性[22].

3 概率可达性分析

在 REDLIB 库的支持下, 本节阐述基于概率 RED 图的概率时间自动机概率可达性分析算法.

如算法 1 所示, 给定一个概率时间自动机 $\mathcal{A}$, $Prob_{max}(\varphi)$计算到达符号状态 φ 的最大概率值. 算法从目标状态 φ 开始实施后向分析. 它迭代地收集可以到达 φ 的所有状态, 以及它们到达 φ 的最大概率值, 直到到达一个最小不动点. 如果初始符号状态被包括在这个不动点中, 就返回对应的概率值; 否则, 就说明 φ 从初始符号状态开始是不可达的, 算法直接返回概率值 *0*. 这里, 我们主要展示符号状态及其对应的概率值是如何在概率 RED 图上表示并计算的. 类似地, 目标状态的最小可达概率可以通过对称的算法来计算得到.

算法 1. $Prob_{max}(\varphi, \varphi_{init})$

Require: 目标状态 φ, 初始状态 φ_{init}

```
1.  reachp={(1.0, φ)}
2.  reach={(1.0, true)}
3.  while (reach != reachp) do
4.    reach=reachp; reachp=∅
5.    for each (l,b,α,μ) ∈  do
6.      prob_pre=∅
7.      for each (D, l') ∈sup(μ) do
8.        pre=red_bck(reach, (D, l'))
9.        pre=pre∩¬φ
10.       pre=prob_multiply(pre, μ(D, l'))
11.       prob_pre=prob_add(prob_pre, pre)
12.     endfor
13.     reachp=prob_max(prob_pre, reachp)
14.   endfor
15.   reachp=reachp ∪{(1.0, φ)}
16. endwhile
17. if φinit ⊆reachp then
18.   return probability value of reachp
19. else
20.   return 0.0
21. endif
```

概率 RED 图 *reach* 和 *reachp* 分别代表每次迭代前后的符号状态和它们对应的可达概率. 从第 3 行开始, 每次迭代中, 我们把上次迭代的结果或是第一次迭代时的初始值{(1.0,φ)}存入 *reach*. 不失一般性地, 令 *reach* 代表一个集合$\{(p_i,\varphi_i) | 1 \le i \le k\}$, 所有 中的概率迁移在迭代中都被用于计算 *reach* 的最弱前件, 即所有可以一步到达 *reach* 所包含的符号状态的状态集. 对于一个概率迁移$(l,b,\alpha,\mu) \in$, *prob_pre* 表示那些可以通过这个迁移到达 *reach* 中某个符号状态的状态集. *prob_pre* 被初始化为空集, 如第 6 行所示. 第 8 行的 REDLIB 函数 *red_bck* 返回可以通过迁移分支(D,l')到达 *reach* 中某个符号状态的符号状态 *pre*[21].

假设某状态 φ_i 到达目标状态的概率为 p_i, 且 *pre* 可以通过某概率迁移$(l,b,\alpha,\mu) \in$ 的分支(D,l')到达 φ_i. 那么, *pre* 通过此分支到达目标状态的概率就是 $\mu(D,l')p_i$. 这一计算通过函数 *prob_multiply* 实现, 如算法 2 所示.

概率 RED 图 *pre* 在第 10 行根据 *pre* 到达目标状态的概率进行更新. 然后, 通过函数 *prob_add*, 更新的 *pre* 被合并入 *prob_pre*, 如第 11 行所示. 算法 3 显示了函数 *prob_add* 的实现机制. 它把可以通过同一个迁移 (l,b,α,μ)到达 *reach* 中同一个状态的状态合并起来. 给定两个概率 RED 图 d_1 和 d_2, 如果某状态 φ 只包含在 d_1 或 d_2 中, 那么在这两个图合并后的新图中, 它会带着其原有的概率值被包含进去. 否则如果 φ 被 d_1 和 d_2 共同包含, 那么在合并后的新图中, 它会将其在两个图中各自的概率值之和作为新的概率值并被包含进去.

在第 13 行, 我们利用之前计算的中间结果 *prob_pre* 来更新 *reachp*. 如算法 4 所示, 我们通过函数 *prob_max*, 来比较上一次的迭代结果与新的计算结果, 以计算每个被遍历过的状态的最大可达概率. 给定两个概率 RED 图 d_1 和 d_2, 如果某状态 φ 只包含在 d_1 或 d_2 中, 那么在这两个图合并后的新图中, 它会带着其原有的概率值被包含进去. 否则如果 φ 同时被两个图

包含，那么在合并后的新图中，它会将在原来两个图中较大的概率值作为新的概率值被包含进去.

第 15 行确保了目标状态 φ 被包含在当前迭代结果中. 如果两次迭代结果没有变化，即 *reachp=reach*，那么说明整个迭代循环到达了一个最小不动点. 然后，若初始状态被包含在这个不动点中，函数 $Prob_{max}$ 就返回对应的概率值；否则返回 *0*.

算法 2. *prob_multiply*

Require: RED 图 *pre*，概率值 *mult*

```
1.  result=∅
2.  for each (p, φ) in pre do
3.      result=result∪{(p*mult, φ)}
4.  endfor
5.  return result
```

算法 3. *prob_add*

Require: RED 图 d_1, d_2

```
1.  result=dy=conj=∅, φ''=∅
2.  if d1=∅ then
3.      return d2
4.  else if d2=∅ then
5.      return d1
6.  else
7.      result=d1
8.      for (p, φ) in d2 do
9.          for (p', φ') in result do
10.             conj=conj∪{(p+p', φ∧φ')}
11.         endfor
12.         φ''=φ
13.         for (p', φ') in result do
14.             φ''=φ''-φ'
15.         endfor
16.         dy={(p, φ'')}
17.         for (p', φ') in result do
18.             φ'=φ'-φ
19.         endfor
20.         result=result∪dy∪conj
21.     endfor
22.     return result
23. endif
```

算法 4. *prob_max*

Require: RED 图 *dx*, *dy*

```
1.  if dx=∅ then
2.      return dy
3.  else if dy=∅ then
4.      return dx
5.  else
6.      for (p, ∅) in dx do
7.          ddx[ip]={(p, ∅)}
8.      endfor
9.      for (p', ∅') in dy do
10.         ddy[i'p]={(p', ∅')}
11.     endfor
12.     for (p, ∅) in dx do
13.         for (p', ∅') in dy do
14.             if p'<p then
15.                 ddy[i'p]={(p', ∅'-∅)}
16.             endif
17.         endfor
18.     endfor
19.     for (p', ∅) in dy do
20.         for (p, ∅) in dx do
21.             if p<p' then
22.                 ddx[ip]={(p, ∅-∅')}
23.             endif
24.         endfor
25.     endfor
26.     for d in ddx do
27.         result=result∪d
28.     endfor
29.     for d' in ddy do
30.         result=result∪d'
31.     endfor
32. endif
```

我们用如图 4 所示的概率时间自动机模型来说明 $Prob_{max}$ 的计算过程. 这个自动机的定义如下：$\mathcal{A}=(L, ,a,I)$，其中 $L=\{a,b,c,g\}$，I 可以这样描述：$((l=a\&clock\leq 2)\&(l=b\&clock\leq 2)\&(l=c\&clock\leq 2))$，其中 *l* 是位置，*clock* 是时钟变量. 包含：

$$l=a\&clock\geq 1\rightarrow 0.3:l=b\&clock=0+0.7:l=g\&clock=0 \quad (1)$$

$$l=b\&clock\geq 1\rightarrow 0.5:l=b\&clock=0+0.1:l=c\&clock=0+0.4:l=g\&clock=0 \quad (2)$$

$$l=c\&clock\geq 1\rightarrow 1.0:l=g\&clock=0 \quad (3)$$

以式(1)为例, 它表示无论何时处于位置 *a*, 只要满足时钟约束 *clock*≥*1*, 概率时间自动机$\mathcal{A}$就可以 *0.3* 的概率迁移到位置 *b*, 或以 *0.7* 的概率迁移到位置 *g*, 然后分别将时钟 *clock* 重置为 *0*. 我们计算最终到达目标状态 φ:*l*=*g* 的最大概率.

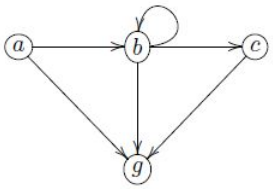


图 4 概率时间自动机

如算法 1 的第 1 行所示, Probmax 从 φ:l=*g* 出发来寻找一个最小不动点. 首先为不动点建立一个概率 RED 图来表示它, 并初始化为{(1.0,φ)}, 如图 5(a)所示. 然后开始第一次迭代, 遍历所有能够直接到达 φ 的迁移分支的前件, 如第 8、9 行所示. 我们假设迁移(3)首先被分析. 令 *X*→*Y*(*μ*(*Y*))表示一个从位置 X 到位置 Y, 概率值为 *μ*(*Y*)的迁移分支. 在计算式(3)的唯一分支 *c*→*g*(1.0)的前件后, 存入 *pre*. 在第 10 行更新概率并将其加入 *prob_pre.pre* 和 *prob_pre* 如图 5(b)所示. 此时, *reachp* 则如图 5(c)所示.

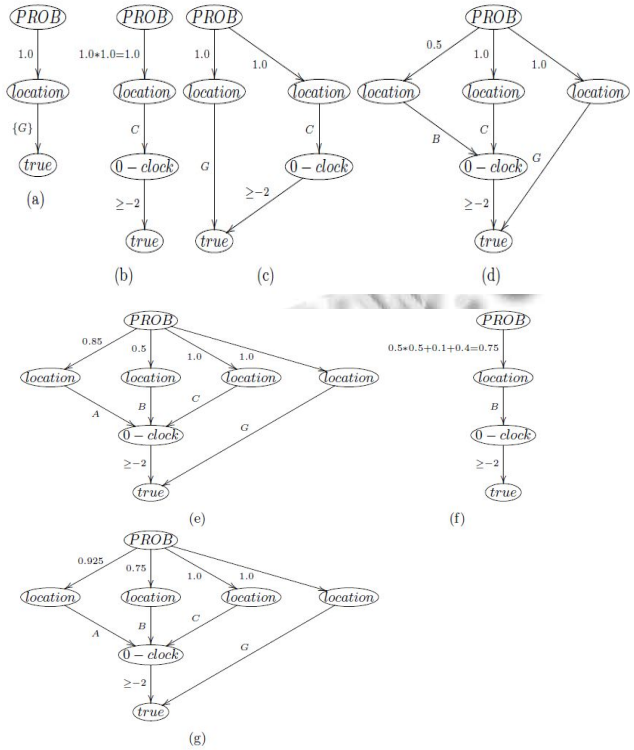


图 5 不动点迭代的过程

假设下一步进行分析的迁移为(2). 对其计算完成后, *reachp* 将如图 5(d)所示. 在第一轮迭代的最后, *reachp* 在第 15 行被更新, 如图 5(e)所示.

由于 *reach* 和 *reachp* 并不相等, 迭代还需要继续. 所有的迁移分支将会被重新遍历一遍. 可以看到, (2)被计算完成后, *prob_pre* 如图 5(f)所示. 这次迭代过后, *reachp* 再一次被更新, 如图 5(g)所示. 因此, 迭代仍然需要继续. 事实上, 因为这个例子中存在环边, *reachp* 在每次迭代后都会有新的更新. 为了解决这个问题, 当 *reachp* 中的状态没有被更新且概率值的更新足够小, 迭代将会停止. 由于初始状态被包含于不动点中, 可以得到到达目标状态 φ:*l*=*g* 的概率 *0.99…*, 近似为 *1.0*.

4 实例研究

我们基于模型检测工具 RED 实现了概率 RED 图及其可达性分析算法. 由于 RED 是时间自动机的模型检测工具, 自身并不支持概率模型, 我们需要对 RED 的输入语言进行扩充.

RED 的输入语言是一种基于模式(mode)的模型语言[4,14,21]. 一个 RED 模型定义了一个基于同步通道通讯的时间自动机的模板. 这个模板由离散变量、时钟和多个模式组成. 每个模式与一个时钟不变式和一组迁移规则相关联. 每条迁移规则的语法格式如下:

when <*seq_sync*> <*guard*> *may* <*seq_assign*> *goto B;*

其中, <*seq_sync*>是一个输入或输出动作的序列; <*guard*>是一个关于时钟和离散变量的布尔条件; <*seq_assign*> 是一个赋值序列; *B* 是模式的标识符. 这样一条迁移规则的含义是, 在当前模式, 只要<*seq_sync*>按指定顺序发生, <*guard*>被保持且不违反模式 *B* 的时钟不变式, 时间自动机就可以进行<*seq_assign*>赋值并迁移到模式 *B*. 每条迁移规则被隐式地用其在模型中被定义时的顺序来索引. 这样, 模型所定义的系统可以由模板的进程实例复合而成. 每个进程实例由一个初始模式和实例的整数编号初始化而得.

我们复用 RED 模型来描述概率时间自动机中的迁移分支, 同时将各分支对应的概率值分离出来并用以下形式进行输入:

<*proc_intl*>,<*#trans*>:(<$prob_1$>,<$trans_1$>),…,(<$prob_k$>, <$trans_k$>)

其中, <*proc_intl*>是进程编号区间, <*#trans*>是概率迁移中分支的数量, <$prob_i$>和<$trans_i$>分别是该迁移中第 *i* 个分支的序号和概率值($1 \le i \le k$).

通过上述对 RED 输入语言的扩充, 就可以把以其他模型语言(如 PRISM)所描述的概率时间自动机模型翻译为扩充的 RED 模型, 并用我们的工具原型进行分析. 例如, 对 PRISM 的概率时间自动机模型的模块(module)结构及其迁移规则的转换是很直接的, 同时需要引入一个 *Signal* 进程来模拟 PRISM 模型中基于动作标号的同步. PRISM 模型中的动作标号被翻译为对应 RED 模型中的同步通道. 我们用 *Signal* 进程和对应进程之间的显式同步, 来实现 PRISM 中拥有相同动作标号的模块之间的同步.

我们对 *7* 个概率时间自动机实例进行了分析. 其中 *6* 个来自 PRISM[23]: *firewire*, $firewire_{abst}$, $repudiation_{honest}$, $repudiation_{malicious}$, *zeroconf* 和 *csma*; 另外一个来自 Modest[24]: *wlan*. 我们的实验环境为 Core i7-2670QM 中央处理器, 4GB 内存及 Ubuntu 11.04 操作系统.

firewire 和 $firewire_{abst}$ 对 IEEE 1394 火线协议进行建模. 该协议定义了当网络中没有环时选出根(root)的规则, 并且在网络中被侦测到环的存在时报告错误[25]. $firewire_{abst}$ 是 *firewire* 的一个简化版本. 对于该实例, 我们分析其最终成功选出根的最大与最小概率. 两个模型都包含一个延时参数 *delay*, 在实验中被设定为 *30* 或 *360*.

$repudiation_{honest}$ 和 $repudiation_{malicious}$ 对不可否认协议进行建模[26]. 对于 $repudiation_{honest}$ 模型, 我们分析服务能够成功停止的概率. 对于 $repudiation_{malicious}$, 我们分析信息泄露的概率.

zeroconf 对 IPv4 零配置网络协议进行建模[27]. 我们分析 IPv4 地址冲突出现的概率.

csma 对 IEEE 802.3 CSMA/CD 协议进行建模[28]. 我们分析从基站传送信息会出现 *COL* 次冲突的最大概率. 这里, *COL* 是一个参数.

wlan 对 IEEE 802.11 局域网无线协议进行建模[29]. 我们分析其中任一基站的补偿计数到达 *K* 的最大和最小概率. 其中, *K* 也是一个参数.

我们从两个方面来评估概率 RED 图的性能: 参数值和进程数量. 显然, 参数值越大, 数值区间越大, RED 图中的结点就可能有越大的出度. 表 1、2 和 3 分别展示了我们的工具, 与 PRISM、Modest 所消耗的总时间(列 time) 和 CPU 时间(列 CPU)的平均值. 表中的每项实验数据都是经过 5 次实验后所获的平均值. M_{max} 和 M_{min} 分别代表了所分析的是模型 M 的最大和最小概率. parameter 列说明了相关参数的设置情况.

表 1 扩充的 RED 图在 PTA 模型检测中的测试结果

case	model	parameter	RED Performance		
			time(s)	CPU(s)	prob
1	$firewire_{max}$	*delay=30*	0.237	0.199	1.000000
2	$firewire_{max}$	*delay=360*	0.251	0.206	1.000000
3	$firewire_{min}$	*delay=30*	0.231	0.191	1.000000
4	$firewire_{min}$	*delay=360*	0.246	0.196	1.000000
5	$firewire\_abst_{max}$	*delay=30*	0.197	0.168	1.000000
6	$firewire\_abst_{max}$	*delay=360*	0.186	0.151	1.000000
7	$firewire\_abst_{min}$	*delay=30*	0.204	0.182	1.000000
8	$firewire\_abst_{min}$	*delay=360*	0.191	0.167	1.000000
9	$repudiation\_honest_{min}$	-	0.274	0.228	1.000000
10	$repudiation\_malicious_{max}$	-	0.301	0.264	0.105658
11	$zeroconf_{max}$	-	0.291	0.232	0.001298
12	$wlan_{max}$	*K=2*	5.214	3.542	0.183594
13	$wlan_{min}$	*K=2*	5.083	3.489	0.000000

表 2 PRISM 在 PTA 模型检测中的测试结果

case	model	parameter	PRISM Performance		
			time(s)	CPU(s)	prob
1	$firewire_{max}$	*delay=30*	0.351	0.314	1.000000
2	$firewire_{max}$	*delay=360*	0.335	0.312	1.000000
3	$firewire_{min}$	*delay=30*	0.339	0.304	1.000000
4	$firewire_{min}$	*delay=360*	0.349	0.318	1.000000
5	$firewire\_abst_{max}$	*delay=30*	0.243	0.228	1.000000
6	$firewire\_abst_{max}$	*delay=360*	0.249	0.231	1.000000
7	$firewire\_abst_{min}$	*delay=30*	0.272	0.255	1.000000
8	$firewire\_abst_{min}$	*delay=360*	0.261	0.233	1.000000
9	$repudiation\_honest_{min}$	-	0.487	0.442	1.000000
10	$repudiation\_malicious_{max}$	-	0.516	0.427	0.105658
11	$zeroconf_{max}$	-	0.317	0.267	0.001302
12	$wlan_{max}$	*K=2*	13.865	11.150	0.183594
13	$wlan_{min}$	*K=2*	13.031	10.550	0.000000

表 3 Modest 在 PTA 模型检测中的测试结果

case	model	parameter	Modest Performance		
			time(s)	CPU(s)	prob
1	$firewire_{max}$	*delay=30*	0.394	0.318	1.000000
2	$firewire_{max}$	*delay=360*	0.393	0.315	1.000000

3	$firewire_{min}$	*delay=30*	0.385	0.321	1.000000
4	$firewire_{min}$	*delay=360*	0.392	0.316	1.000000
5	$firewire\_abst_{max}$	*delay=30*	0.274	0.225	1.000000
6	$firewire\_abst_{max}$	*delay=360*	0.275	0.232	1.000000
7	$firewire\_abst_{min}$	*delay=30*	0.293	0.238	1.000000
8	$firewire\_abst_{min}$	*delay=360*	0.287	0.228	1.000000
9	$repudiation\_honest_{min}$	-	0.512	0.438	1.000000
10	$repudiation\_malicious_{max}$	-	0.533	0.454	0.105658
11	$zeroconf_{max}$	-	0.351	0.274	0.001302
12	$wlan_{max}$	*K=2*	14.236	11.323	0.183594
13	$wlan_{min}$	*K=2*	13.486	10.759	0.000000

另一方面，模型包含的进程越多，模型的符号状态就包含越多的变量，对应的概率 RED 图的深度也就越大. 我们对模型 csma 设置了不同基站的数量来分别进行分析. 实验结果如表 4 所示.

表 4 在可扩充的 PTA 模型 csma 上对性质 Pmax=?[Fc=COL]的检测结果(固定参数设置: COL=5)

Station amount		2	4	6	8
RED	time(s)	0.814	2.734	8.748	31.830
	CPU(s)	0.691	2.418	6.082	22.580
	prob	0.062805	0.009313	0.009313	0.009313
PRISM	time(s)	7.308	112.541	10186.092	-
	CPU(s)	4.686	100.983	7637.475	-
	prob	0.062805	0.009313	0.009313	-
Modest	time(s)	7.401	113.272	10842.465	-
	CPU(s)	4.785	101.563	7681.521	-
	prob	0.062805	0.009313	0.009313	-

通过对比我们可以看出，概率 RED 图在概率时间自动机可达性分析中比 PRISM 和 Modest 有着更好的时间效率. 在表 1 到表 3 列出的 13 个实验中，我们的工具相对于 PRISM，分别在 CPU 时间和总时间上平均减少了 63.19%和 57.36%. 和 Modest 相比，CPU 时间和总时间分别平均减少了 63.78%和 59.43%. 对于 csma，我们的工具在更大的模型规模上具有更好的时间表现.

5 结语

本文提出了 RED 图的一种概率扩展. 我们通过在原始 RED 图中引入概率根结点，来扩展 RED 图，并实现了基于概率 RED 图的概率时间自动机可达性分析算法和工具原型. 实验研究表明，与 PRISM 和 Modest 相比，扩展 RED 图在概率时间自动机可达性分析上拥有更好的时间效率和延展性.

下一步工作，我们将进一步尝试扩展 RED 图的应用，并探索与其它符号模型检测方法的结合.

参考文献

1 Wang F. Symbolic parametric safety analysis of linear hybrid systems with BDD-like data-structures. IEEE Trans. on Software Engineering, 2005, 31(1): 38–51.

2 Wang F. Region encoding diagram for fully symbolic verification of real-time systems. The 24th Annual International Computer Software and Applications Conference, 2000(COMPSAC 2000). IEEE. 2000. 509–515.

3 Wang F. Efficient model-checking of dense-time systems with tme-convexity analysis. Theoretical Computer Science, 2013, 467: 89–108.

4 Wang F, Yao LW, Yang YL. Efficient verification of distributed real-time systems with broadcasting behaviors. Real-Time Systems, 2011, 47(4): 285–318.

5 Wang F. Symbolic verification of complex real-time systems with clock-restriction diagram. Formal Techniques for Networked and Distributed Systems, 2002: 235–250.

6 Larsen KG, Weise C, Yi W, Pearson J. Clock difference diagrams. Nordic Journal of Computing, 1999, 6(3): 271–298.

7 Møller J, Lichtenberg J, Andersen HR, Hulgaard H. Difference decision diagrams. Computer Science Logic, 1999: 111–125.

8 Wang F. Efficient verification of timed automata with BDD-like data-structures. 4th International Conference on Verification, Model Checking, and Abstract Interpretation. 2003. LNCS. 2575. 189–205.

9 Wang F. Formal verification of timed systems: A survey and perspective. Proc. of the IEEE, 2004, 92(8): 1283–1305.

10 Kwiatkowska M, Norman G, Parker D. PRISM: Probabilistic symbolic model checker. Computer Performance Evaluation: Modelling Techniques and Tools. Springer Berlin Heidelberg. 2002: 200–204.

11 Kwiatkowska M, Norman G, Parker D. PRISM 4.0: verification of probabilistic real-time systems. Computer Aided Verification, 2011: 585–591.

12 Alur R, Courcoubetis C, Dill D. Model-checking in dense real-time. Information and Computation, 1993, 104(1):

2–34.

13 Fujita M, McGeer PC, Yang JY. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. Formal Methods in System Design, 1997, 10(2-3): 149–169.

14 Wang F. REDLIB for the formal verification of embedded systems. Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. ISoLA 2006. IEEE. 2006. 341–346.

15 Hartmanns A, Hermanns H. A modest approach to checking probabilistic timed automata. Sixth International Conference on the Quantitative Evaluation of Systems, 2009. QEST'09. IEEE. 2009. 187–196.

16 Hartmanns A. MODEST--A unified language for quantitative models. 2012 Forum on Specification and Design Languages (FDL). IEEE. 2012. 44–51.

17 Amnell T, Behrmann G, Bengtsson J, D'Argenio P, David A, Fehnker A, Hune T, Jeannet B, G. Larsen K, Oliver M, Pettersson P, Weise C. UPPAAL now, next, and future meet a family of model checkers. Proc. of Modelling and Verification of Parallel Processes (MOVEP'2k). LNCS Tutorial 2067. 2001. 100–125.

18 Beauquier D. On probabilistic timed automata. Theoretical Computer Science, 2003, 292(1): 65–84.

19 Duan S, Zhang J, Roe P, Wimmer J, Dong X, Truskinger A, Towsey M. Timed probabilistic automaton: A bridge between raven and song scope for automatic species recognition. Proc. of the Twenty-Fifth Innovative Applications of Artificial Intelligence Conference. AAAI. 2013. 1519–1524.

20 Wang F, Wu RS, Huang GD. Verifying timed and linear hybrid rule-systems with RED. Proc. of the 17th International Conference on Software Engineering & Knowledge Engineering (SEKE 2005). 2005. 448–454.

21 Wang F. REDLIB: A Library for Integrated BDD-like Diagrams. http://sourceforge.net/projects/redlib.

22 Clarke EM, Fujita M, Zhao X. Hybrid decision diagrams. Proc. of the 1995 IEEE/ACM International Conference on Computer-Aided Design. IEEE Computer Society (ICCAD-95). 1995. 159–163.

23 PRISM-Benchmarks. http://www.prismmodelchecker.org/benchmarks/models.php\#ptas.

24 Modest Case Studies. http://www.modestchecker.net/ CaseStudies.aspx.

25 Kwiatkowska M, Norman G, Sproston J. Probabilistic model checking of dead-line properties in the IEEE 1394 FireWire root contention protocol. Formal Aspects of Computing, 2003, 14(3): 295–318.

26 Markowitch O, Roggeman Y. Probabilistic non-repudiation without trusted third party. Second Conference on Security in Communication Networks, 1999, 99: 25–36.

27 Cheshire S, Aboba B, Guttman E. Dynamic configuration of ipv4 link-local addresses. IETF INTERNET DRAFT, zeroconf working group, June 2001. draft-ietf-zeroconf-ipv4-linklocal-03.txt.

28 Duflot M, Fribourg L, Herault T, Lassaigne R, Magniette F, Messika S, Peyronnet S, Picaronny C. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. Electronic Notes in Theoretical Computer Science, 2005, 128(6): 195–214.

29 Kwiatkowska M, Norman G, Sproston J. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. Lecture Notes in Computer Science, 2002, 17(3): 411–423.