DOI: 10.7641/CTA.2018.70684

基于搜索树的业务流程Petri网模型抽象化简方法

方 欢†, 何路路, 方贤文, 王丽丽

(安徽理工大学 数学与大数据学院, 安徽 淮南 232001)

摘要: 由于大型复杂的业务流程模型不便于用户形成快速的梗概理解, 因此模型的抽象化简方法研究至关重要. 已有的模型抽象化简方法大都考虑模型的控制流, 很少考虑模型的交互语义, 并且对于模型的待抽象区域定位模糊. 本文利用Petri网的行为关系理论, 结合深度优先搜索的思想, 提出了基于搜索树的业务流程模型抽象化简方法. 通过定义工作流网的3种块结构和边界库所的概念, 得到网系统的变迁关联搜索树, 进而利用变迁关联搜索树来识别模型中待抽象的区域, 从而实现模型的抽象化简. 最后, 通过一个具体的实例来验证所提出方法的有效性和可行性. 本文的主要贡献是: 在同时考虑模型行为交互语义和控制流依赖关系的基础上, 提出块结构的抽象化简方法和块结构的识别定位方法, 算法的时间复杂度控制在多项式难度.

关键词: 深度优先搜索; 变迁关联搜索树; 抽象化简; 行为轮廓; Petri网

引用格式: 方欢, 何路路, 方贤文, 等. 基于搜索树的业务流程Petri网模型抽象化简方法. 控制理论与应用, 2018, 35(1): 92 – 102

中图分类号: TP301 **文献标识码**: A

A search-tree-based abstraction method for business process Petri nets models

FANG Huan†, HE Lu-lu, FANG Xian-wen, WANG Li-li

(College of Mathematics & Big Data, Anhui University of Science and Technology, Huainan Anhui 232001, China)

Abstract: As it is difficult to form a quick overview understanding for large and complex business process models, so the studies about the technologies and methods of model abstraction and simplification are crucial. The related researches in existence have some limitations, such as only the control flows of the model are taken into account, whereas the interactive behavioral semantics are mostly neglected, and have little vague about the scopes orientation that to be abstracted in the model. A search-tree-based abstraction method for business process model is purposed in the paper, which is founded on the basis of behavioral relation theory of Petri nets and the Depth-First Search ideas. Firstly, the concepts of three kinds of block structures and boundary places in work-flow Petri nets are formalized, and the transitions association tree of the system is then obtained. The transition association tree is further used to identify and locate the areas that to be abstracted in the model, and the aim of model abstraction is then accomplished. Finally, a case example is applied to illustrate the validity and feasibility of the proposed method. Therefore, compared to the existing work, our main contributions can be stated as follows: we purpose a block based abstraction method and its corresponding block identification method, where the interactive semantics and control dependencies are both taken into consideration, and the purposed methods are in polynomial time complexity.

Key words: depth first search; transition association search tree; abstract simplification; behavioral profile; Petri nets

Citation: FANG Huan, HE Lulu, FANG Xianwen, et al. A search-tree-based abstraction method for business process Petri nets models. *Control Theory & Applications*, 2018, 35(1): 92 – 102

1 引言(Introduction)

业务流程管理(business process management, BPM)已经广泛应用于企业的日常业务管理中, 企业已经针对不同的事项开发出各种各样复杂的业务流程管理系统(business process management systems, BPMS)的构件资源. 然而, 复杂 BPMS 面临着如下的

收稿日期: 2017−09−19; 录用日期: 2018−01−26.

†通信作者. E-mail: fanghuan0307@163.com; Tel.: +86 15955489701.

本文责任编委: 赵千川.

国家自然科学基金项目(61472003, 61272153, 61340003, 61402011, 61572035), 安徽省自然科学基金项目(1608085QF149), 安徽省高校优秀青年人才支持项目(gxyqZD2018038), 淮南市科技计划项目(2016A23)资助.

Supported by the National Natural Science Foundation of China (61472003, 61272153, 61340003, 61402011, 61572035), the Natural Science Foundation of Anhui Province (1608085QF149), the Youth Talents Support Project in Universities of Anhui Province(gxyqZD2018038) and the Science & Technology Foundation of Huainan City (2016A23).

难题: 首先, 一个复杂的业务流程模型可能包含数十个或者数百个活动元素, 并且这些元素捕获了关于流程模型的各个不同方面的特征, 如结构、功能、资源或者数据方面[1]的信息. 从用户角度考虑, 用户很难去理解一个包含60个以上元素的业务流程模型[2]; 其次, 复杂的业务流程模型通常包含大量的不同的错误信息, 而这些错误信息会对业务流程管理带来负面的影响[3]; 最后, 实际中不同的用户(或利益相关者)需要定义各自不同的个性化流程视角, 比如业务经理更倾向接受流程的梗概视图, 而实际的流程参与者需要对其涉及的流程部分的详细视图[4]. 因此, 在实际的工业应用中, 为了克服这些问题, 需要提出一个关于复杂流程模型的“简单视图”, 即业务流程模型抽象(business process model abstraction, BPMA), 以便实现对业务流程模型不同方面的快速理解.

Petri网是实现BPMA的一种重要的形式化建模工具, 以其坚实的数学基础和图形化建模语言而著称. 将Petri网用于模型的抽象和化简的研究由来已久, 取得了很多的研究结论, 如文献[5]运用Petri网代数将Petri网化简和精细化转换进行了详细的论述, 并研究了相关的性质保性问题. 除了Petri网代数论研究, 更多的学者将 Petri 网的语义抽象用于工业控制中, 最典型的是 FMS 的监控器设计方法的相关研究. 文献[6–9]从关键活动和冗余约束的角度实现Petri网工业系统的控制.

以上Petri网的抽象研究[5–9]主要针对工业控制系统, 而针对软件业务流程系统展开分析讨论的主要是BPMA理论和方法[10–18], 并将这种层次化抽象思想应用在业务流程系统的演化适配分析中[19], 得到了很多有效的研究结论. 文献[11]提出多入多出(MEME)的模型分解技术进行模型抽象, 该方法具有二次时间的计算复杂度. 之后, 在文献[11]的基础, 文献[12] 提出了单入单出(SESE)的模型抽象技术, 使得时间的计算量变为线性时间, 减少了计算时间的复杂度, 进一步改进了文献[11]的研究结论. 文献[13]针对结构化的流程模型提出了两步法构建个性化的用户定制视图, 文献[14–15]中Polyvyanyy 等人分别在 2009 年和2010年提出了基于流程结构树(PST)的模型抽象方法, 但是该方法没有明确指出哪些元素应该被抽象放入一个更高级别的活动, 在模型语义描述方面存在部分局限. 文献[16]针对语义学提出了基于流程模型的语义信息进行模型的抽象技术, 文献[17]提出了基于行为轮廓的抽象化简算法, 该算法利用活动之间的行为轮廓关系来识别出被聚合的活动组之间的控制流关系. 这些模型抽象方法都是基于模型控制流结构而展开的, 而均未考虑模型的数据流结构. 为了将模型的控制流和数据流都纳入到BPMA中, 少数研究对此展开了一些研究探讨: 文献[18]提出了对活动带数据信息标签的模型抽象方法, 文献[19]提出了对模型带数据流信息的模型抽象技术.

综上所述, 已有的研究结论大都是基于单个模型内部活动的抽象方法研究, 而并没有考虑模型交互时的情形, 且已有研究均没有给出模型中待抽象区域的具体界定方法. 例如, 利用文献[17]的研究结论对图1的模型进行抽象操作, 可以得到如图2的结果. 进一步分析可以发现: 文献[17]的研究方法可以将变迁集$\{t_2, t_3\}$, $\{t_4, t_5, t_6\}$, $\{t_8, t_9\}$ 进行聚合操作. 然而, 从业务流程的现实角度出发, 两个不同部门的活动在抽象时大都不会被抽象为一个新的变迁, 那么$\{t_4, t_5, t_6\}$中的$\{t_5\}$和$\{t_6\}$是分属不同的两个模型的变迁, 但被聚合到一起, 则会引起模型交互的额外开销. 另外, 已有的关于模型抽象化简方面的研究很少涉及到待抽象区域的计算方法, 而这对复杂的、具有交互结构的业务流程化简至关重要. 因此, 本文在文献[17]的基础上, 利用工作流网和行为轮廓的相关理论, 提出基于搜索树的业务流程抽象化简方法, 该方法能在考虑模型交互语义的基础上, 完成计算和界定模型中待抽象区域的计算, 从而进一步完善模型的抽象化简方法的已有研究结论.

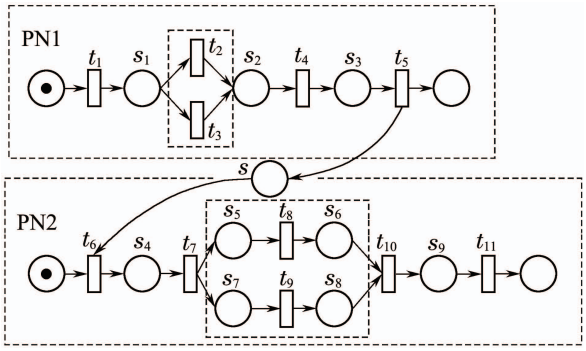


图 1 包含两个交互子模型的业务系统

Fig. 1 A business system that has two interactive components

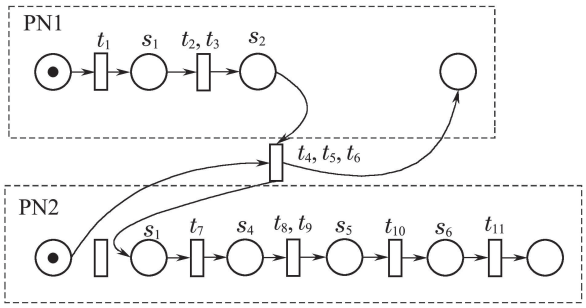


图 2 图1化简后的网系统

Fig. 2 The simplified system of Fig.1

本文的主要思路如下: 首先, 将业务流程模型转换为基于Petri网的工作流系统, 提出了基于搜索树的模型抽象化简(search tree-based abstraction)方法. 进而, 利用深度优先搜索策略计算出模型的活动行为轮廓

关系和边界库所; 再次, 根据深度优先搜索策略产生的节点遍历顺序, 构造模型相应的变迁关联搜索树; 最后, 利用变迁关联搜索树界定模型待抽象的区域, 完成模型的抽象化简.

本文结构如下: 第2部分给出了相关的基本概念; 第3部分在文献[18]的基础上, 定义了3种合理块结构, 并在此基础上提出了基于合理块的模型抽象化简算法; 为了实现对模型中待抽象区域的定位, 第4部分提出了待抽象区域的搜索树定位算法; 第5部分通过一个实际案例对所提出的方法进行试验分析; 第6部分对全文进行了总结, 并对未来研究方向进行了展望.

2 相关定义(Related concepts)

有关Petri网、Petri网系统的定义以及变迁发生规则的相关定义在此不做赘述, 可以参考文献[20], 下面主要介绍行为轮廓和模型抽象的一些相关概念.

定义 1(弱序[21]) 设$\Sigma=(N,M)$是一个网系统, τ是模型的可行迹集合, 弱序关系$\prec_\Sigma$包含了所有的变迁对 $x\prec y$, 使得在 τ 中存在一条迹 $\sigma=t_1,t_2,\cdots,t_m$, 满 足 $t_j=x$, $t_k=y$ ($j\in\{1,2,\cdots,m-1\}$, $j<k\leqslant m$).

定义 2(行为轮廓[21]) 设$\Sigma=(N,M)$是一个网系统, 初始标识为M_0, 对任意的变迁对$(t_1,t_2)\in T\times T$满足下面关系:

1) 若$t_1\prec t_2$且$t_2\not\prec t_1$, 则称t_1,t_2处于严格序关系, 记作$t_1\to t_2$;

2) 若$t_1\not\prec t_2$且$t_1\not\prec t_2$, 则称t_1,t_2处于排他序关系, 记作t_1+t_2;

3) 若$t_1\prec t_2$且$t_2\prec t_1$, 则称t_1,t_2处于交叉序关系, 记作$t_1||t_2$;

称所有的弱序关系的集合为流程模型Σ的行为轮廓, 记作$BP=\{\to,||,+\}$.

定义 3(工作流Petri网[22]: WF–PN) 一个网$N=(S,T;F,i,o)$称为WF–PN, 当且仅当满足条件:

1) 该Petri网有唯一的源库所$i:{}^\bullet i=\varnothing$;

2) 该Petri网有唯一的终止库所$o:o^\bullet=\varnothing$;

3) 该Petri网上的每一个节点都属于i到o的一条路径上, 即Petri网N是强连通的.

定义 4(抽象函数) 令 $N=(S,T;F)$ 为初始的WF–PN, $N'=(S',T';F')$为抽象化简的WF–PN, 则称函数 $h:N\to N'$为抽象函数, 其中 h 满足下列条件:

1) $\forall\, s\in S\Rightarrow h(s)\in S'\cup\{\xi\}$;

2) $\forall\, t\in T\Rightarrow h(t)\in T'\cup\{\xi\}$;

3) $s\in S$, $|{}^\bullet s|=0\wedge h(s)=s'\Rightarrow|{}^\bullet s'|=0$;

4) $h(n)\in S\cup T\Leftrightarrow h(n)=n$;

5) $h(n_1)=n'\wedge h(n_2)=n'\Rightarrow n_1=n_2$.

在定义4中, 条件1)保证N中的库所被映射为N中的一个库所或者被删除; 条件2)保证N中的变迁被映射为N'中的一个变迁或者被删除; 条件3)保证N中的源库所被映射为N中的源库所; 条件4)保证被映射到N'中的一个节点(库所或变迁)如果也是属于N中的节点, 则在抽象过程中该节点未发生变化; 条件5)保证N中有且仅有唯一的一个节点与N'中的节点相对应.

模型的抽象化简涉及3个基本操作: 节点不变、节点聚合和节点删除. 当节点保持不变时, 有$h(n)=n$; 当节点n_1,n_2,n_3被聚合成一个节点n'时, 有$h(n_1,n_2,n_3)=n'$; 当节点被删除时, 有$h(n)=\xi$.

3 基于合理块结构的模型抽象化简算法(Abstraction and simplification algorithm based on sound blocks)

3.1 合理块结构(Sound blocks)

本文在文献[17]行为轮廓的基础上, 提出了3种可以被抽象的合理块结构子片段, 每个片段均对应一个模型的抽象化简规则.

3.1.1 顺序块抽象化简操作(Abstraction simplification operation of sequential blocks)

定义 5(顺序块SB) 令$N=(S,T;F)$为WF–PN, $SB=(S_{\rm sb},T_{\rm sb};F_{\rm sb})$为$N$的子网, 若$SB$是一个顺序块, 当且仅当$|S_{\rm sb}|>1$或$\forall\, t_{i1},t_{i2}\in T_{\rm sb}\Rightarrow t_{i1}\to t_{i2}\vee t_{i2}\to t_{i1}$.

Rule 1 顺序块 SB 抽象化简操作 (sequential blocks reduction operation, SBRO).

原网为$N=(S,T;F)$, $SB=(S_{\rm sb},T_{\rm sb};F_{\rm sb})$是其中的一个顺序块, $N'=(S',T';F')$是抽象化简后的网模型.

1) 若$SB=(S_{\rm sb},T_{\rm sb};F_{\rm sb})$满足${}^\bullet SB\cap S-S_{\rm sb}\neq\varnothing\wedge SB^\bullet\cap T-T_{\rm sb}\neq\varnothing$, 则该顺序块$SB$在原网$N=(S,T;F)$中位于一个变迁和一个库所之间, 则SBRO将该顺序块SB删除, 即$\forall\, s_i,s_j\in S_{\rm sb}$, $t_n,t_m\in T_{\rm sb}$: $h(\{s_i,s_j,t_m,t_n\})=\xi$;

2) 若$SB=(S_{\rm sb},T_{\rm sb};F_{\rm sb})$满足${}^\bullet SB\cap S-S_{\rm sb}\neq\varnothing\wedge SB^\bullet\cap S-S_{\rm sb}\neq\varnothing$, 则该顺序块$SB$在原网$N=(S,T;F)$中位于两个库所之间, 则将该顺序块$SB$抽象为一个新变迁, 即$\forall\, s_i,s_j\in S_{\rm sb}$, $t_n,t_m\in T_{\rm sb}$, $t'\in T'$: $h(\{s_i,s_j,t_m,t_n\})=t'$;

3) 若$SB=(S_{\rm sb},T_{\rm sb};F_{\rm sb})$满足${}^\bullet SB\cap T-T_{\rm sb}\neq\varnothing\wedge SB^\bullet\cap T-T_{\rm sb}\neq\varnothing$, 则该顺序块$SB$在原网中位于两个变迁之间, 则将该顺序块$SB$抽象为一个新库所, 即$\forall\, s_i,s_j\in S_{\rm sb}$, $t_n,t_m\in T_{\rm sb}$, $s'\in S'$: $h(\{s_i,s_j,t_m,t_n\})=s'$.

图3(a)–(c)分别给出了当SB中含一个或两个变迁

时的抽象化简情况, 当SB中含有大于2个变迁时, 操作类似.

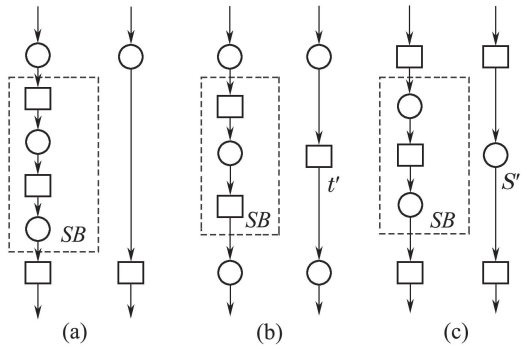


图 3 简单SB抽象化简的3种情况

Fig. 3 Three abstraction cases of simple SB

3.1.2 并发块抽象化简操作 (Abstraction simplification operation of concurrent blocks)

定义 6(并发块 CB) 令 $N=(S,T;F)$ 为 WF–PN, $CB=(S_{\rm cb},T_{\rm cb};F_{\rm cb})$为$N$的子网, 若$CB$是一个并发块, 当且仅当 ${}^\bullet CB\cap T-T_{\rm cb}\neq\varnothing\wedge CB^\bullet\cap T-T_{\rm cb}\neq\varnothing$ 和$\forall\, t\in T_{\rm cb}\Rightarrow\exists t'\in T_{\rm cb}, t||t'$同时成立.

Rule 2 并发块 CB 抽象化简操作 (concurrent blocks reduction operation, CBRO).

原网 $N=(S,T;F)$, $CB=(S_{\rm cb},T_{\rm cb};F_{\rm cb})$ 是其中的一个并发块, $N'=(S',T';F')$是抽象化简后的网模型.

1) 若CB满足$|{}^\bullet CB\cap T-T_{\rm cb}|=1\wedge|CB^\bullet\cap T-T_{\rm cb}|=1$, 则进行如下聚合操作: $\forall\, s_i,s_j\in S_{\rm cb},\ t_n,t_m\in T_{\rm cb},\ s'\in S':h(s_i,s_j,t_m,t_n)=s'$.

2) 若CB不满足$|{}^\bullet CB\cap T-T_{\rm cb}|=1\wedge|CB^\bullet\cap T-T_{\rm cb}|=1$, 则对并发块不做任何化简, 即$\forall\, n\in S_{\rm cb}\cup T_{\rm cb}\Rightarrow h(n)=n$.

图4对CBRO进行了图形说明, 在图4(a)中, 并发块CB可以化简成一个库所, 而在图4(b)中, 由于并发块CB不满足$|{}^\bullet CB\cap T-T_{\rm cb}|=1\wedge|CB^\bullet\cap T-T_{\rm cb}|=1$, 因此不能进行化简, 其每个分支可以根据顺序块的化简规则进行操作.

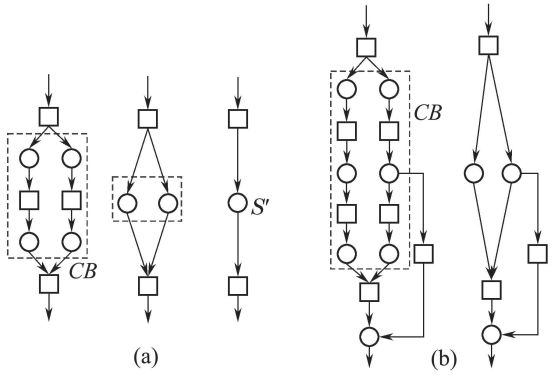


图 4 并发块抽象化简的两种情况

Fig. 4 Two abstraction cases of concurrent block

3.1.3 选择块抽象化简操作(Abstraction simplification operation of choice blocks)

定义 7(选择块 ChB) 令$N=(S,T;F)$为WF–PN, $ChB=(S_{\rm chb},T_{\rm chb};F_{\rm chb})$ 为 N 的子网, 若 ChB 是一个选择块, 当且仅当 ${}^\bullet ChB\cap T-T_{\rm chb}\neq\varnothing\wedge ChB^\bullet\cap T-T_{\rm chb}\neq\varnothing$ 和 $\forall\, t\in T_{\rm chb}\Rightarrow\exists t'\in T_{\rm chb}: t+t'$同时成立.

Rule 3 选择块 ChB 抽象化简操作 (choice blocks reduction operation, ChBRO).

原网 $N=(S,T;F)$, $ChB=(S_{\rm chb},T_{\rm chb};F_{\rm chb})$ 是其中的一个并发块, $N'=(S',T';F')$是抽象化简后的网模型.

1) 若ChB满足$|{}^\bullet ChB\cap S-S_{\rm chb}|=1\wedge|ChB^\bullet\cap S-S_{\rm chb}|=1$, 则进行如下聚合操作: $\forall\, s_i,s_j\in S_{\rm chb},\ t_n,t_m\in T_{\rm chb},\ t'\in T':h(s_i,s_j,t_m,t_n)=t'$;

2) 若ChB不满足$|{}^\bullet ChB\cap S-S_{\rm chb}|=1\wedge|ChB^\bullet\cap S-S_{\rm chb}|=1$, 则对选择块不做任何化简, 即$\forall\, n\in S_{\rm chb}\cup T_{\rm chb}\Rightarrow h(n)=n$.

与并发块的化简规则CBRO类似, 图5给出了选择块抽象化简操作的例示.

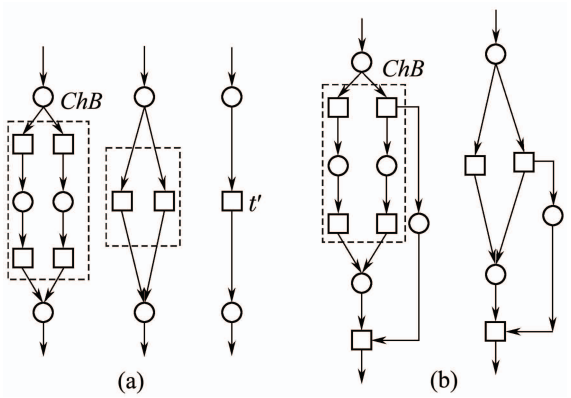


图 5 选择块抽象化简的两种情况

Fig. 5 Two abstraction cases of choice block

4 流程模型的块结构抽象化简算法 (Block based abstraction algorithm of process model)

第3.1节已经确定了模型的3类合理的块结构, 现根据这3类结构设计流程模型抽象化简算法, 具体的过程如算法1所示.

算法 1 基于合理块结构的抽象化简算法 (sound-block-based abstraction algorithm, SBA algorithm).

输入: 目标模型WF–PN.

输出: WF–PN的抽象化简模型N'.

步骤:

Step 1 计算出N中变迁之间的行为轮廓BP_N,

flag=1;

Step 2 While(flag);

Step 2.1 根据计算出的BP_N, 遍历网系统中的变迁, 如果存在多个变迁之间满足顺序块SB的关系, 则执行Rule 1; 否则执行Step 2.2;

Step 2.2 如果存在多个变迁之间满足并发块CB的关系, 则执行Rule 2; 否则执行Step 2.3;

Step 2.3 如果存在多个变迁之间满足选择块ChB的关系, 则执行Rule 3;

Step 2.4 如果网系统中没有满足条件的变迁可供抽象化简, 则flag=0;

Step 3 算法结束, 返回化简后的网系统N'.

5 模型待抽象化简区域的搜索树定位算法 (Search-tree-based algorithm for identifying regions to be abstracted)

在SBA算法中, 需要通过找出模型变迁之间的关系来实现行为轮廓的计算, 而变迁间的弱序关系都是建立在模型的行为迹基础上, 因此可以借助深度优先搜索(depth first search, DFS)的相关思路, 完成行为弱序关系的判定: 将Petri网模型看成是一个有向图, 根据定义5、定义6和定义7提出的3种块结构概念, 在模型运行期间选择执行一条行为迹遍历, 直到这条迹的某个变迁不符合条件走不下去, 再选择另一条迹, 以此类推.

在算法设计领域中, 深度搜索和广度搜索思想类似, 计算难度相当, 然而在基于Petri网的行为轮廓模型中, 一条行为迹的产生对应网系统的一次深度搜索, 但是网系统的执行不能同时启动若干个并发状态, 因而在串行算法中不能将深度搜索转换为同等概念下的广度搜索. 本节利用深度优先搜索树来构建模型的行为轮廓关系和模型边界库所的计算方法, 从而实现对顺序块、并发块和选择块这3个块结构的快速界定, 进一步完善SBA算法的中的Step 1和Step 2.

5.1 变迁关联 DFS 树(Transitions association DFS tree)

利用深度优先策略遍历模型中的每个变迁和库所(以下称为节点)的目的是构造深优先搜索树, 并以此计算出变迁之间的行为轮廓关系以及模型交互时的边界库所. 但以往在遍历网系统时得到的是模型的可行迹, 然而该可行迹并不包含库所, 所以要想在遍历时得到边界库所, 需要一个扩充的既包含变迁又包含库所的可行迹. 为此, 先给出扩充迹和边界库所的形式化定义.

定义 8(扩充可行迹) 设Σ为网系统, $\exists\,\sigma$是网系统Σ中的一条可行迹, $\sigma = t_{i1}, t_{i2}, \cdots, t_{in}$, 则称$\mathrm{ext}\,\sigma$为$\Sigma$的一条扩充可行迹当且仅当$\mathrm{ext}\,\sigma = s_{i1}, t_{i1}, \cdots, s_{ii}, \cdots, s_{in}, t_{in}, s_{i,n+1}$, 其中$s_{ij} \in S \wedge s_{ij} \in {}^{\bullet}t_{ij} \wedge s(i, j+1) \in t_{ij}^{\bullet}\,(1 \leqslant j \leqslant n)$.

为了叙述方便, 本文以下网系统中所提到的扩充可行迹简称为可行迹.

定义 9(边界库所) 设Σ_1和Σ_2为两个不同的网系统, 对任意的一个库所s, 则$s \in S_{\mathrm{boundary}}$为边界库所, 当且仅当$\exists\,\mathrm{ext}\,\sigma_1 \in \Sigma_1$, $\mathrm{ext}\,\sigma_2 \in \Sigma_2$: 使得 $s \in \mathrm{ext}\,\sigma_1 \wedge s \in \mathrm{ext}\,\sigma_2$.

如图1所示, PN_1中存在一条可行迹$\mathrm{ext}\,\sigma_1 = t_4 s_3 t_5 s$, 同时 PN_2 中存在一条可行迹 $\mathrm{ext}\,\sigma_2 = s t_6 s_4$, 则$s$为边界库所.

边界库所的识别方法如算法2所示.

算法 2 边界库所识别算法(boundary place identification algorithm: BPI algorithm)

输入: 网系统$\Sigma = \{\Sigma_1, \Sigma_2, \cdots, \Sigma_n\}$.

输出: 边界库所集S_{boundary}.

步骤:

Step 1 为每个库所定义访问函数数组visit[ ];

Step 2 初始化操作, 每个节点初始时都未被访问, 即$\mathrm{visited}[v] = \mathrm{False}$;

Step 3 从第1个节点开始依次访问, 如果节点v_i有 $\mathrm{visited}[v_i] = \mathrm{False}$, 则执行$BPI(\Sigma, v_i)$, 即对尚未被访问节点继续调用边界库所识别; 否则, 继续检查下一个节点;

Step 4 根据Step3, 得到整个网系统中节点(包括变迁和库所)的访问顺序, 即得到网系统中所有的扩充迹$\mathrm{ext}\,\sigma$, 对所有的$\mathrm{ext}\,\sigma$做比较, 如果($\forall\,\mathrm{ext}\,\sigma_i \in \Sigma_i$, $\forall\,\mathrm{ext}\,\sigma_j \in \Sigma_j (i \neq j)$, $\mathrm{ext}\,\sigma_i \cap \mathrm{ext}\,\sigma_j \cap S_i \cap S_j = s_{ij}$), 则$s_{ij}$为$\Sigma_i$和$\Sigma_j$的边界库所, 即$s_{ij} \subseteq S_{\mathrm{boundary}}$;

Step 5 算法结束, 返回边界库所集合

$$S_{\mathrm{boundary}} = \bigcup_{i=1, i\neq j}^{n} \bigcup_{j=1}^{n} s_{ij}.$$

当BPI算法结束时, 网系统中的所有节点都被遍历了一遍, 并且每个变迁之间的行为轮廓关系和模型之间的边界库所都被发现. 下面根据深度优先搜索的结果构建搜索树, 用来发现网系统的边界库所集合S_{boundary}.

5.2 变迁关联搜索树 (Transitions association search tree)

变迁关联搜索树是根据深度优先搜索算法求得的每个节点的遍历顺序来构造的, 搜索树中包含两个基本要素、节点和边, 其具体定义如定义10所示.

定义 10(变迁关联搜索树T_Σ-tree) 设Σ为网系统, 则符合如下条件的搜索树是Σ对应的变迁关联搜

索树, 记为T_Σ-tree:

1) 树中的每个节点为网系统中对应的变迁;

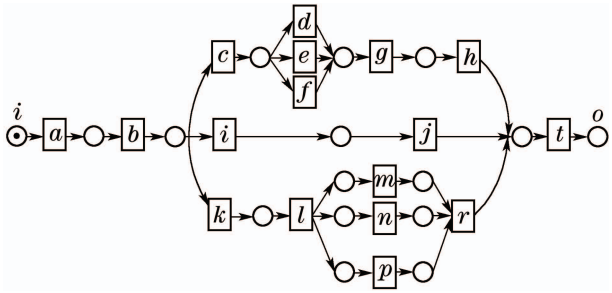
2) 树中的每个节点v有3个属性: v.parent为其父节点; v.left为其左孩子; v.brother为其兄弟节点, 若该节点没有父亲、儿子或兄弟, 则相应的属性为NIL;

3) 树中每条边都有两个值: 1或∞, 即对$\forall\, t_i, t_j \in T$,有

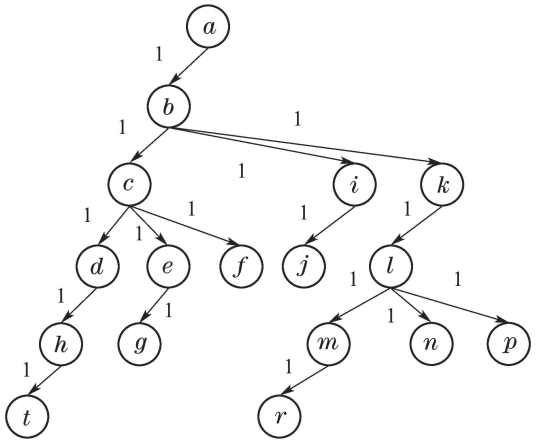
$$d(t_i,t_j)=\begin{cases}1, & s\in t_i^{\bullet}\wedge s\in {}^{\bullet}t_j\wedge s\notin S_{\text{boundary}},\\ \infty, & s\in t_i^{\bullet}\wedge s\in {}^{\bullet}t_j\wedge s\in S_{\text{boundary}}.\end{cases}$$

若一棵搜索树中∞若有m个, 则此变迁关联搜索树代表的网系统Σ内包含$\Sigma_1, \Sigma_2, \cdots, \Sigma_{m+1}$个服务不同对象的子模型.

如图6(a)的网系统, 根据深度优先搜索得到一个对应的T_Σ-tree, 如图6(b)所示. 进一步, 将得到一个k叉树转化为二叉树结构, 以节省算法的计算复杂度. 一个二叉链表由3个域组成, 数据域和左右指针域, 其中左右指针分别指向该节点的最左孩子和其兄弟节点.



(a) 一个简单网系统Σ

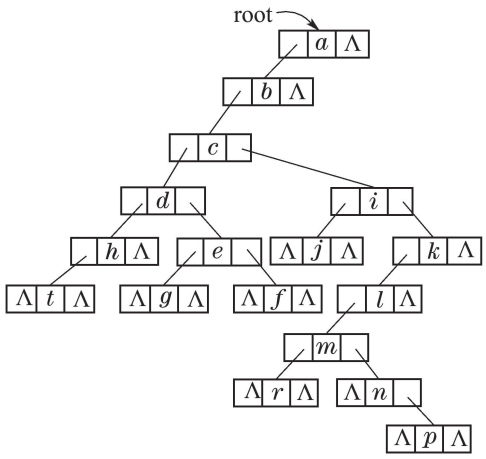


(b) 图(a)对应的T_Σ-tree

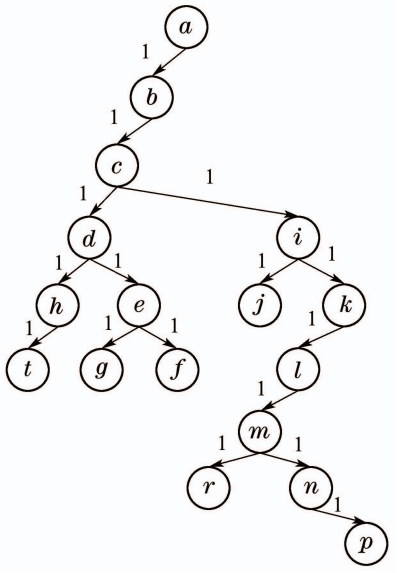
图 6 一个工作网系统及其T_Σ-tree

Fig. 6 A workflow net system and its T_Σ-tree

由此, 图6(b)可以转换成一个二叉链表, 进而相应的得到一个二叉树, 如图7所示.



(a) 图6(b)T_Σ-tree对应的二叉链表



(b) 图(a)对应的二叉树

图 7 图6的二叉树的结构

Fig. 7 The binary tree structure of Fig.6

5.3 二叉搜索树的合理块结构定位算法 (The sound blocks identification algorithm based on binary search tree)

本文将根据以下定义的两种规则用于发现变迁关联二叉搜索树中对应网系统中的块结构, 具体规则定义如下.

Rule 4　二叉搜索树的顺序块SB.

若对$\forall\, t_1, t_2, t_3 \in T_\Sigma$-tree, 当且仅当

1) t_1.left $= t_2 \wedge t_2$.right $= NIL$;

2) $d(t_1,t_2)\neq\infty$,

则称$\{t_1,t_2\}\in SB$.

若$\{t_1,t_2\}\in SB$并且$\{t_2,t_3\}\in SB$, 则有$\{t_1,t_2,t_3\}\in SB$.

Rule 5　二叉搜索树的选择块ChB、并发块CB.

若对$\forall\, t_1, t_2, t_3 \in T_\Sigma$-tree, 当且仅当

1) $t_1.\text{right} = t_2$;

2) $d(t_1,t_2) \neq \infty \Rightarrow \{t_1,t_2\} \in ChB \vee \{t_1,t_2\} \in CB$;

3) $\{t_1,t_2\} \in ChB \wedge \{t_2,t_3\} \in ChB \Rightarrow \{t_1,t_2,t_3\} \in ChB$ (或 $\{t_1,t_2\} \in CB \wedge \{t_2,t_3\} \in CB \Rightarrow \{t_1,t_2,t_3\} \in CB$).

Rule 6 二叉搜索树的选择块ChB、并发块CB的抽象顺序.

1) 若 $\forall\, t_1,t_2,t_3,t_4 \in T_\Sigma$-tree, $\{t_1,t_2\} \in ChB \wedge \{t_2,t_3\} \in ChB \wedge \{t_2,t_4\} \in SB$或$\{t_1,t_2\} \in CB \wedge \{t_2,t_3\} \in CB \wedge \{t_2,t_4\} \in SB$, 则此时先对顺序块进行抽象化简, 其后再对选择或并发(选择)块进行抽象化简;

2) 若$\forall\, t_1,t_2,t_3,t_4 \in T_\Sigma$-tree, 若$\{t_1,t_2\} \in SB \wedge \{t_2,t_3\} \in SB \wedge \{t_2,t_4\} \in ChB$或$\{t_1,t_2\} \in SB \wedge \{t_2,t_3\} \in SB \wedge \{t_2,t_4\} \in CB$, 则此时先对选择或并发(选择)块进行抽象化简, 其后再对顺序块进行抽象化简.

按照深度优先算法遍历网中的每个节点, 当搜索到网系统中的选择块和并发块时, 在所构造的T_Σ-tree里的形式都是一样的, 即两个节点其中一个是双亲节点, 另一个是右孩子节点, 并且两个节点边的权值不为∞, 符合这样的特点结构就是对应于网系统中的选择块或并发块. 需要说明的是, 本文并不具体要分辨出在搜索树中到底是哪种块, 因为本文的目的是只要在识别(界定)这种块结构, 就可以在网系统中进行相应的抽象化简. 在Rule4, Rule5和Rule6的基础上, 对算法1的SBA算法进行进一步的优化, 可以得到基于搜索树的业务流程抽象化简算法.

5.4 搜索树的模型抽象化简算法(The abstraction algorithm based on search tree)

算法 3 基于搜索树的抽象化简算法(search-tree-based abstraction algorithm, STA algorithm).

输入: 网系统 Σ 及其模块组成 $\Sigma = \{\Sigma_1, \Sigma_2, \cdots, \Sigma_n\}$.

输出: Σ的抽象化简网系统Σ'.

步骤:

Step 1 $BPI(\Sigma)$ //对网系统实行遍历搜索, 计算出行为轮廓和边界库所;

Step 2 利用Step 1访问的节点遍历顺序构造T_Σ-tree; //构造变迁关联搜索树;

Step 3 将T_Σ-tree存储为二叉链表形式, 转换为对应的二叉搜索树;

Step 4 遍历二叉搜索树, 如果当前节点符合Rule 4和Rule 6, 则执行Rule 1; 否则执行Step 5; //对顺序块进程抽象;

Step 5 如果当前节点符合Rule 5 和 Rule 6, 则执行Rule 2和Rule 3; //对选择和并发块进行抽象;

Step 6 如果当前还有未被访问的节点, 则转到Step 4继续寻找可抽象的块; 否则, 返回此网系统不可抽象化简;

Step 7 END(算法结束).

下面用一个例子来说明STA算法的执行过程.

例 1 图8给出了一个简单的网站支付业务流程模型: 首先, 买家选好商品后选择支付方式, 然后将钱打入第3方支付平台机构. 如果第3方支付机构收到的金额少于商品价格, 则会通知买家重新支付; 反之, 如果第3方支付机构收到大于等于商品定价的金额, 则会通知卖家已收到商品支付, 此后卖家会给买家发货图8中变迁具体代表的含义如表1所示.

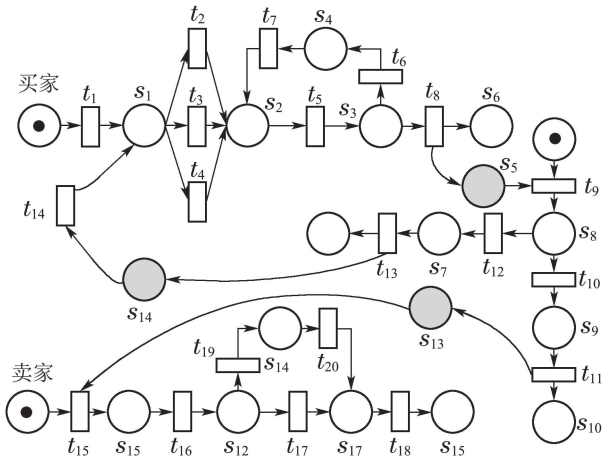


图 8 某购物平台支付环节模型

Fig. 8 A payment model of shopping platform

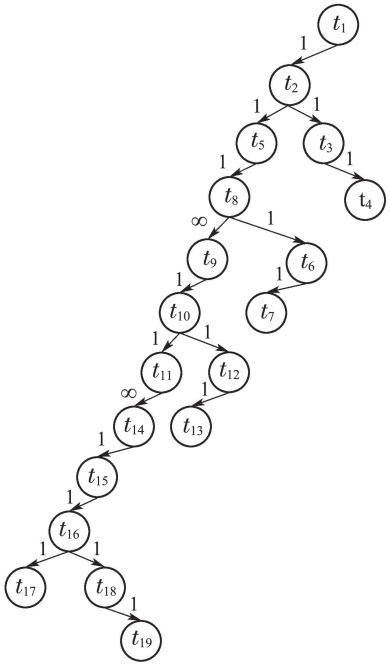
表 1 图8中的符号表示

Table 1 The symbols denotation of Fig.8

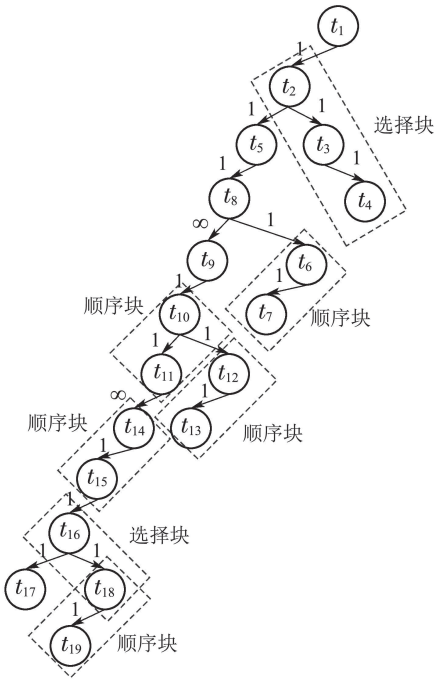
符号	语义	符号	语义
t_1	开始支付	t_{11}	向卖家发送已付款信息
t_2	选择支付宝支付	t_{12}	付款金额不足
t_3	选择微信支付	t_{13}	重新增量支付
t_4	选择网银支付	t_{14}	开始增量支付
t_5	输入密码	t_{15}	收到支付成功信息
t_6	密码不正确	t_{16}	核对收货地址等信息
t_7	重新输入密码	t_{17}	信息核对正确
t_8	密码正确	t_{18}	发货
t_9	收到买家付款信息	t_{19}	信息核对有误
t_{10}	付款金额大于等于商品定价金额	t_{20}	修改信息

根据算法3, 在执行Step 1时, 网系统中模型交互之间的边界库所被发现, 即s_5, s_{13}, s_{14}. 并且此时网系统的变迁行为轮廓关系和节点的遍历顺序都已经被计算出来. 再根据算法3的Step 2, Step 3得到对应的T_Σ-

tree二叉链表结构, 如图9(a)所示. 最后根据算法3的Step 4和Step 5, 利用规则Rule 4, Rule 5以及Rule 6 对网系统中的3个块结构进行定位, 此时得到5个顺序块$\{t_8, s_5, t_9\}$, $\{t_6, s_4, t_7\}$, $\{t_{18}, s_{12}, t_{19}\}$, $\{t_{12}, s_7, t_{13}, s_{14}, t_1\}$和$\{t_{10}, s_8, t_{11}, s_{13}, t_{14}, s_9, t_{15}\}$, 以及2个选择块$\{t_2, t_3, t_4\}$, $\{s_{10}, t_{18}, s_{12}, t_{19}, s_{11}\}$, 这7个定位块分别对应于网系统的7个待抽象化简区域. 根据算法3所得到第1次循环操作的结果如图9(b)所示.



(a) 图8网系统对应的T_Σ-tree



(b) 图8系统中的合理块

图 9　运用STA算法对图8进行块结构定位

Fig. 9 The sound blocks of Fig. 8 by STA algorithm

图9(b)中出现了两个∞, 代表该网系统包含3个不同的子模型, 并且子模型和子模型交互的区域分别在t_8和t_9, 以及t_{11}和t_{14}, 说明根据算法3在进行抽象时它们不能被聚合在一起. 但是, 如果根据算法1, 则t_8和t_9, t_{11}和t_{14}在抽象化简时就会被聚合在一起, 从而导致出现不符合实际意义情况.

6　抽象方法的性质分析(Properties analysis of the proposed abstraction method)

由于合理的工作流Petri网系统WF–PN具有恰当终止性 (proper termination)、活性 (liveness)、无死锁性(deadlock free)[22]等性质, 本文提出的抽象化简方法建立在WF–PN之上. 下面对本文提出的基于搜索树的抽象化简方法的相关性质进行分析.

定理 1　若$N = (S, T; F)$, $N' = (S', T'; F')$分别是使用STA方法抽象化简前后的网模型, M_0是N的初始标识, N'的初始标识满足:

$$\forall\, s' \in S' : M_0(s') = \sum_{\exists\ s_i \in S \wedge h(N_{s_i}, s_i) = s} M_0(s_i),$$

其中: N_{s_i}是N中的节点子集, $N_{s_i} \subset S \cup T$.

1) 若网系统$\Sigma = (N, M_0)$是合理的WF–PN, 则$\Sigma' = (N', M_0')$也是合理的WF–PN;

2) 若网系统$\Sigma = (N, M_0)$是活的(无死锁的), 则$\Sigma' = (N', M_0')$也是活的(无死锁的);

3) 若$\Sigma = (N, M_0)$是安全的, 则$\Sigma' = (N', M_0')$也是安全的.

证　1) 恰当终止性包含两方面的内容: 始终终止和无死变迁.

若(N, M_0)是恰当终止的, 则$\exists\, M_f \in R(M_0) : \nexists\, t \in T \wedge M_f[t >$, 且$M_f(o) = 1 (s \neq o : M_f(s) = 0)$. 而$(N', M_0')$是通过STA算法得到的, 由于STA算法仅包含3种合理的块结构: 顺序块、并发块和选择块, 那么根据定义5、定义6和定义7, 原网模型的终止库所o所对应的抽象库所o'一定也是唯一的, 并且$\exists\, M_f' \in R(M_0') : \nexists\, t' \in T' \wedge M_f'[t' >$, 且$M_f'(o') = 1 \wedge (\forall\, s' \neq o' : M_f'(s') = 0)$, 则满足始终终止性.

下面来证明若(N, M_0)中无死变迁, 则(N', M_0')中也无死变迁. 若$\exists\, t' \in T'$是(N', M_0')中的一个死变迁, 即$\nexists\, M_1' \in R(M_0') : M_1'[t' >$. 由于$\forall\, t' \in T' : \exists\, t \in T,\ N_s \subset S \cup T \Rightarrow h(t, N_s) = t'$, 若$t'$是死变迁, 则可知在$(N, M_0)$中至少存在一个死变迁$t$, 与假设矛盾.

综上, 定理1的1)得证.　　证毕.

2)和3)的证明与1)类似, 在此不做赘述.

定理 2(行为良好的保性性)　若$N = (S, T; F)$, $N' = (S', T'; F')$分别是使用STA方法抽象化简前后

的网模型, M_0是N的初始标识, N'的初始标识满足

$$\forall\, s' \in S' : M_0(s') = \sum_{\exists\ s_i \in S \wedge h(N_{s_i}, s_i) = s} M_0(s_i),$$

其中: N_{s_i}是N中的节点子集, $N_{s_i} \subset S \cup T$. 若网系统$\Sigma = (N, M_0)$是行为良好(well-performed)的WF–PN, 则$\Sigma' = (N', M_0')$ 也是行为良好的WF–PN.

证 当WF–PN系统$\Sigma = (N, M_0)$是合理的、活的、无死锁的并且是安全的, 则称 Σ 是行为良好的[23](well-performed). 根据定理1可知, 若Σ是行为良好的, 则通过STA抽象化简方法得到的Σ'也是行为良好的. 证毕.

定义 11(抽象行为包含) 设两个WF–PN: $\Sigma = (N, M_0)$, $\Sigma' = (N', M_0')$, 其中: $N = (S, T; F)$, $N' = (S', T'; F')$, M_0, M_0'分别是N, N'的初始标识, $\Gamma = \{\mathrm{ext}\,\sigma_i | \mathrm{ext}\,\sigma_i \in \Sigma\}$和$\Gamma' = \{\mathrm{ext}\,\sigma_i' | \mathrm{ext}\,\sigma_i' \in \Sigma'\}$分别为$\Sigma$和$\Sigma'$的迹集合. 若$\forall\, \sigma \in \Gamma, \forall\, x_i \in \sigma \Rightarrow \exists\, N_s \subset S \cup T, \exists(\sigma' \in \Gamma', y_j \in \sigma')$满足$h(x_i, N_s) = y_j$, 则称$\sigma$抽象行为包含$\sigma'$.

定义 12(精细行为包含) 设两个WF–PN: $\Sigma = (N, M_0)$, $\Sigma' = (N', M_0')$, 其中: $N = (S, T; F)$, $N' = (S', T'; F')$, M_0, M_0'分别是N, N'的初始标识, $\Gamma = \{\mathrm{ext}\,\sigma_i | \mathrm{ext}\,\sigma_i \in \Sigma\}$和$\Gamma' = \{\mathrm{ext}\,\sigma_i' | \mathrm{ext}\,\sigma_i' \in \Sigma'\}$分别为$\Sigma$和$\Sigma'$的迹集合. 若$\forall\, \sigma \in \Gamma$, $\forall\, x_i \in \sigma \Rightarrow \exists\, N_s' \subset S' \cup T'$, $\exists(\sigma' \in \Gamma', y_j \in \sigma')$ 满足 $h(y_j, N_s') = x_i$, 则称σ精细行为包含σ'.

定义 13(抽象层次的行为一致) 设两个WF–PN: $\Sigma = (N, M_0)$, $\Sigma' = (N', M_0')$, 其中: $N = (S, T; F)$, $N' = (S', T'; F')$, M_0, M_0'分别是N, N'的初始标识. 若σ抽象行为包含σ', 且σ'精细行为包含σ, 则称σ, σ'抽象层次的行为一致.

定理 3 若流程模型为$\Sigma = (N, M_0)$, $N = (S, T; F)$, 而 $\Sigma' = (N', M_0')$, $N' = (S', T'; F')$ 是通过STA算法得到的抽象模型, M_0'满足

$$\forall\, s' \in S' : M_0(s') = \sum_{\exists\ s_i \in S \wedge h(N_{s_i}, s_i) = s} M_0(s_i),$$

其中: N_{s_i}是N中的节点子集, $N_{s_i} \subset S \cup T$, 则σ, σ'满足抽象层次的行为一致性.

证 通过定义11、定义12和定义13可以直接得出. 证毕.

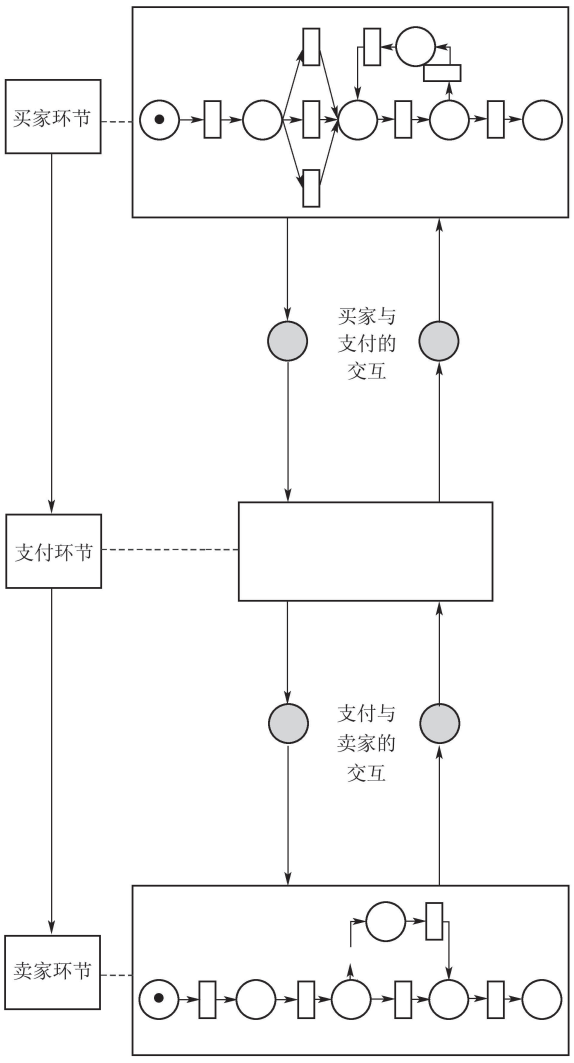
7 案例应用(Case study)

为了验证本文提出的行为抽象方法在大型模型系统中的可行性, 以BPI网站给出的银行支付交互的实际行为轨迹为基础 (http://data.4tu.nl/repository/collection:event_logs), 借助行为挖掘的方法构建的购物平台支付环节业务流程(如图10所示), 以此业务流程模型为例对本文提出的抽象化简方法进行验证, 图中具体的语义在此省略, 可以参考网站相关的说明.

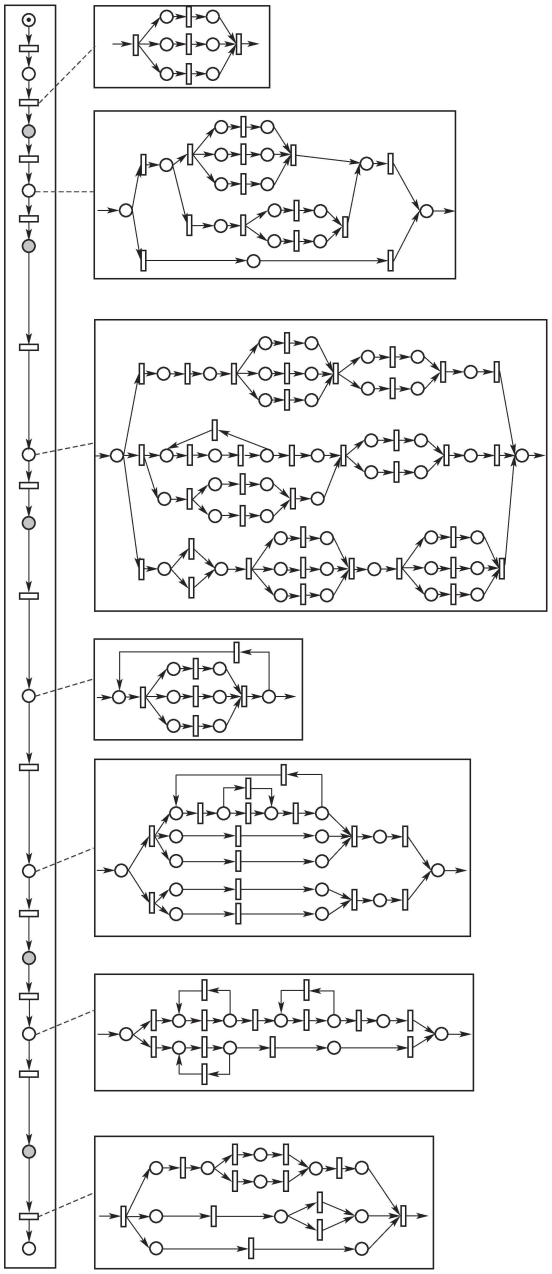
图10(a)给出了一个具有卖家、买家及第3方的支付模型框架, 该模型具有典型的交互语义, 图10(b)对其中的支付模块细节进行了刻画, 其中虚线代表对应关系, 而带阴影的圆圈表示使用STA算法计算得到的边界库所.

8 结论(Conclusions)

流程模型的复杂性对于用户进行快速理解和分析都带来了不同程度的困难, 因此, 业务流程模型的抽象化简是一个至关重要的问题. 本文在现有研究的基础上, 基于Petri网的行为轮廓关系理论, 结合深度优先搜索策略, 探讨了模型抽象化简方法以及待抽象区域的识别和定位问题.



(a) 一个存在多模块交互语义的支付场景



(b) 支付模型的细节

图 10 BPI网站由迹挖掘产生的Petri网模型

Fig. 10 A process Petri net system from BPI website through log mined method

本文的贡献如下: 1) 依据行为轮廓关系, 提出了3种合理的块结构, 分别是顺序块、并发块、选择块; 2) 依据深度优先搜索策略, 构造网系统对应的变迁关联搜索树, 识别模型交互过程中的边界库所, 并在抽象算法中与普通库所区别对待; 3) 根据变迁关联搜索树设计对应3种块结构的定位规则, 识别出网系统中待抽象化简的区域. 最后, 通过BPI挖掘的业务流程模型, 对所提出的抽象化简方法以及待抽象区域定位的相关算法进行了验证.

在今后的研究中, 重点对带数据信息的业务流模型抽象化简方法, 以及根据抽象化简方法对模型的合理性检验问题进行分析, 并进一步对寻找出的不合理结构进行结构处理, 以达到在系统集成过程中对业务逻辑的正确性进行论证分析.

参考文献(References):

[1] WEBER B, REICHERT M. Refactoring process models in large process repositories [C] //*International Conference on Advanced Information Systems Engineering*. Montpellier, France: IEEE, 2008, 6: 124 – 139.

[2] MENDLING J, REIJERS H A, VAN DER AALST W M P. Seven process modeling guidelines (7PMG) [J]. *Information and Software Technology*, 2010, 52(2): 127 – 136.

[3] MENDLING J. On the detection and prediction of errors in EPC business process models [J]. *EMISA Forum*, 2007, 27(2): 52 – 59.

[4] POLYVYANYY A, SMIRNOV S, WESKE M. The triconnected abstraction of process models [C] //*International Conference on Business Process Management*. Ulm, Germany: Springer, 2009, 9: 229 – 244.

[5] LI J, ZHOU M C, DAI X Z. Reduction and refinement by algebraic operations for Petri net transformation [J]. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 2012, 42(5): 1244 – 1255.

[6] HUANG B, ZHOU M C, ZHANG G X. Synthesis of Petri net supervisors for FMS via redundant constraint elimination [J]. *Automatica*, 2015, 61: 156 – 163.

[7] HUANG B, ZHOU M C, HUANG Y S, et al. Supervisor synthesis for FMS based on critical activity places [J]. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2017: DOI 10.1109/TSMC.2017.2732442.

[8] LI J, ZHOU M C, GUO T, et al. Robust control reconfiguration of resource allocation systems with Petri nets and integer programming [J]. *Automatica*, 2014, 50(3): 915 – 923.

[9] XU Shanshan, DONG Lida, ZHU Dan, et al. Redundancy detection and structure simplification for a class of liveness-enforcing Petri net supervisors [J]. *Control Theory & Applications*, 2013, 30(6): 673 – 682.
(徐珊珊, 董利达, 朱丹, 等. 一类活性Petri网控制器的冗余检测及结构简化 [J]. 控制理论与应用, 2013, 30(6): 673 – 682.)

[10] SMIRNOV S, REIJERS H A, MATHIAS W. Business process model abstraction: a definition, catalog and survey [J]. *Distributed and Parallel Databases*, 2012, 30(1): 63 – 99.

[11] ZERGUINI L. A novel hierarchical method for decomposition and design of workflow models [J]. *Journal of Integrated Design & Process Science*, 2004, 8(2): 65 – 74.

[12] VANHATALO J, VÖLZER H, LEYMANN F. Faster and more focused control-flow analysis for business process models through SESE decomposition [C] //*The 5th International Conference on Service-Oriented Computing*. Vienna, Austria: Lecture Notes in Computer Science, 2007, 9: 43 – 55.

[13] ESHUIS R, GREFEN P. Constructing customized process views [J]. *Data & Knowledge Engineering*, 2008, 64(2): 419 – 438.

[14] POLYVYANYY A, SMIRNOV S, WESKE M. On application of structural decomposition for process model abstraction [C] //*Business Process, Services Computing and Intelligent Service Management, BPSC 2009*. Leipzig, Germany: Gesellschaft fur Informatik, 2009, 3: 110 – 122.

[15] POLYVYANYY A, VANHATALO J, LZER H. Simplified computation and generalization of the refined process structure tree [C] //*International Conference on Web Services and Formal Methods*. Berilin: Springer, 2010, 9: 25 – 41.

[16] SMIRNOV S, REIJERS H A, WESKE M. A semantic approach for business process model abstraction [C] //*International Conference on Advanced Information Systems Engineering*. London, United kingdom: Springer, 2011, 6: 497 – 511.

[17] SERGEY S, MATTHIAS W, JAN M. Business process model abstraction based on synthesis from well-structured behavioral profiles [J]. *International Journal of Cooperative Information Systems*, 2012, 21(1): 55 – 83.

[18] MEYER A, WESKE M. Data support in process model abstraction [C] //*International Conference on Conceptual Modeling*. Florence, Italy: Springer, 2012, 10: 292 – 306.

[19] KOLB J, REICHERT M. Data flow abstractions and adaptations through updatable process views [C] //*ACM Symposium on Applied Computing*. Coimbra, Portugal: ACM, 2013, 3: 1447 – 1453.

[20] WU Zhehui. *Introduction to Petri* [M]. Beijing: Machinery Industry Press, 2006.
(吴哲辉. Petri导论 [M]. 北京: 机械工业出版社, 2006.)

[21] SMIRNOV S, WEIDLICH M, MENDLING J. *Business Process Model Abstraction Based on Behavioral Profiles* [M]. Berlin: Springer, 2010.

[22] WEIDLICH M, MENDLING J, WESKE M. Efficient consistency measurement based on behavioral profiles of process models [J]. *IEEE Transactions on Software Engineering*, 2010, 37(3): 410 – 429.

[23] GEROGE G, SHAMILA M, MAYER W, et al. Change propagation and conflict resolution for the co-evolution of business processes [J]. *International Journal of Cooperative Information Systems*, 2015, 24(1): 1 – 38.

作者简介:

方 欢 (1982–), 女, 博士, 副教授, 目前主要研究方向为Petri网理论与应用、智能控制, E-mail: fanghuan0307@163.com;

何路路 (1990–), 男, 硕士, 主要研究方向为Petri网理论与应用, E-mail: 47488398@qq.com;

方贤文 (1976–), 男, 博士, 教授, 主要研究方向为业务流程管理系统、行为轮廓、Petri 网理论与应用, E-mail: xianwenfang@hotmail.com;

王丽丽 (1982–), 女, 硕士, 副教授, 主要研究方向为业务流程管理系统、Petri网理论与应用, E-mail: llwang@aust.edu.cn.