利用令牌轨迹区域从标记 Petri 网合成 Petri 网

arXiv:2601.23130v1 [cs.FL] 30 Jan 2026

Robin Bergenthum

数学与计算机科学系，哈根远程大学，德国，邮箱：
robin.bergenthum@fernuni-hagen.de

Jakub Kov´aˇr

编程系统教研组，哈根远程大学，德国，邮箱：
jakub.kovar@fernuni-hagen.de

摘要。 合成会根据行为规约自动生成过程模型。当目标模型为佩特里网时，我们通过区域理论来研究合成。研究人员已针对基于状态的规约（如迁移系统和步迁移系统）以及基于语言的规约，对基于区域的合成进行了广泛研究。因此，文献中的区域理论分为两个主要分支：基于状态的区域和基于语言的区域。采用基于状态的区域时，行为规约是一组全局状态及其相关状态转移。这种表示能够自然地表达冲突以及全局状态的合并。然而，它会遭遇状态爆炸问题，且无法显式表达并发。采用基于语言的区域时，行为规约是一组示例运行，这些运行由事件的部分或全序集合定义。这种表示能够自然地表达并发和分支。然而，随着选择数量的增加，它会迅速膨胀，且无法表达冲突的合并。迄今为止，合成必须在这两种方法之间权衡。这两种区域定义都存在根本局限，因此合成始终需要折中。本文通过引入一种同时涵盖基于状态和基于语言输入的新区域理论，消除了这些局限。我们证明，新定义是一种融合了这两种概念的区域元理论。该理论使用由一组标记网构成的规约，从而能够自然地表达冲突、并发和局部状态的合并，并合成一个能够模拟输入规约中所有标记网的佩特里网。

通信地址：Universit¨atsstr. 1，58097 哈根，德国

关键词： 佩特里网、合成、区域理论、令牌轨迹、标记佩特里网

1. 引言

我们使用佩特里网 [1, 2, 3, 4] 对复杂系统建模。佩特里网具有形式语义、直观的图形表示，以及表达系统各项动作之间并发关系的能力。然而，为现实世界的流程构建佩特里网模型仍然成本高昂且容易出错 [1, 5]。

幸运的是，在对系统进行建模时，我们往往能够获得关于预期流程行为的描述或规约，例如事件日志、示例运行，或描述用例的产品规格。我们可以借助语言、迁移系统或偏序语言，将这类行为规范形式化。如果某一规约准确地刻画了预期行为，我们便可以自动合成合适的过程模型。

在关于基于状态的合成的文献中，合成通常涉及判定是否存在一个行为与输入规约完全匹配的模型。参见 [6] 以获得全面综述。在这一设定下，合成往往给出否定答案。对于基于语言的综合，其合成问题是计算出一个过程模型，使得：(A) 该规约属于所合成模型的行为，并且 (B) 所合成模型具有尽可能少的额外行为。

在本文中，我们采用后一种方法，综合出对指定行为的上近似。将完美拟合检查视为可选项后，综合过程始终会生成一个模型，并为建模和流程挖掘领域带来更多应用。

佩特里网综合的理论基础是区域理论 [7, 8]。研究者已针对迁移系统 [6], 、语言 [9, 10], 和部分语言 [11, 12], 对区域理论进行了广泛研究。相关文献报道了许多理论成果、概念、案例研究和工具支持，包括 APT[13], Genet [14], ProM [15, 16], Viptool [17], 以及 I Petri网 [18]。研究者还将区域理论扩展到广义网类（例如抑制网 [19, 20], ）、更精细的发射语义（例如区间语义 [21],），以及近期将这两者结合起来的方法 [22]。

尽管已有文献种类繁多，我们发现区域理论在文献中主要沿着两条路径发展。如果输入由全局状态构成，例如转移系统，则应用基于状态的区域。在这种情况下，区域是指定状态的一个多重集，并由所谓的最小区域构造出有限的库所集合。如果输入由有序事件集构成，例如事件日志、语言或部分语言，则应用所谓的基于语言的区域。在这里，区域是由语言前缀产生的令牌多重集，而有效库所则通过求解相关的整数线性规划来计算。为了获得有限结果，可以计算一个基，或者采用错误延续的概念 [23]。

尽管这两个分支有着共同的基础，但它们依赖于不同的定义和算法，因此我们必须预先选择一种行为表示。在实际应用中，这一选择将我们限制在两种技术中的一种，并且往往要求我们早早确定对系统行为的特定理解。因此，建模工作量和综合结果的质量都在很大程度上取决于最初如何捕捉行为。并发、冲突或状态合并等重要方面，根据所选表示的不同，可能更容易或更难表达。这促使我们寻求对特定行为表示不那么敏感、且能够灵活整合系统行为不同视角的综合方法。

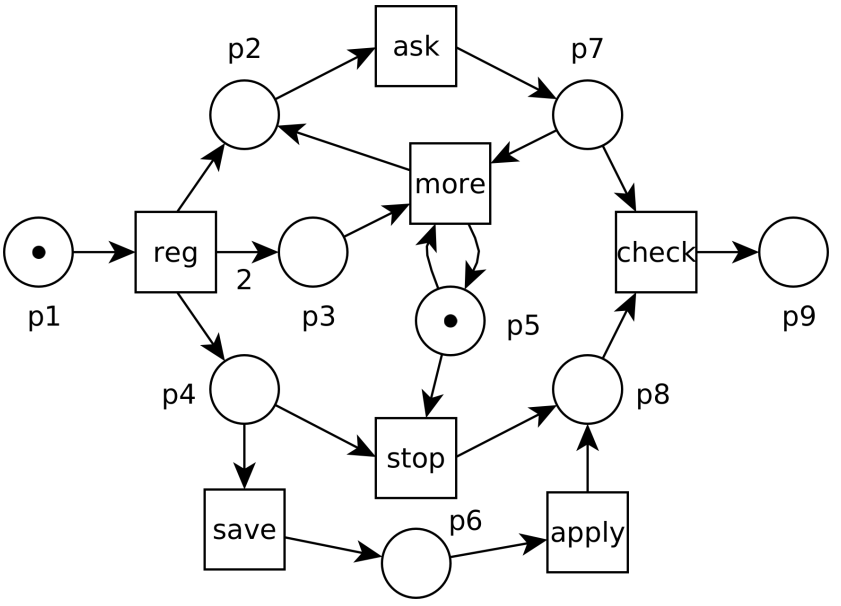


图1：示例工作流的 Petri 网模型，本文始终以此作为贯穿全文的示例，展示了冲突、并发、循环和弧权重。

让我们以一个简单的类工作流系统为例。注册后，我们会询问一次、两次或三次信息。与此同时，流程要么保存并应用信息，要么停止请求更多信息。收集完所有所需信息，且停止或 应用 操作发生后，将执行最终检查。图1展示了这一工作流的佩特里网模型。由于我们是经验丰富的建模人员，并且熟悉佩特里网，因此能够构建这样的模型。尽管如此，最终模型并不简单。它包含冲突、并发、分布式初始标识、短循环以及弧权重。

在许多实际应用中，从头开始对系统建模实在太困难。因此，一种替代方法是先描述系统的预期行为，再利用区域理论自动生成最适配的过程模型。我们假设，对行为进行建模（例如通过示例运行或单次执行来描述），要比构建一个完整且全面集成的系统模型容易得多。作为示例，现在假设我们无法直接获得图1，而是依靠合成技术来生成这一结果。

利用基于状态的区域理论，我们通过对可达状态集合及所有相关状态转移进行建模，来规定预期行为。图2展示了我们工作流的相应规约。尽管该模型构成了一个简单的状态机，但构建起来却很困难，因为即使在这个小型示例中，状态和变迁的数量也已经很多。系统中的并发在状态空间中形成了菱形结构，难以进行显式表示和推理。此外，我们必须区分各个状态，因为需要统计*ask*的发生次数。从建模角度来看，我们更希望直接构建图1，而不是对图2进行建模。

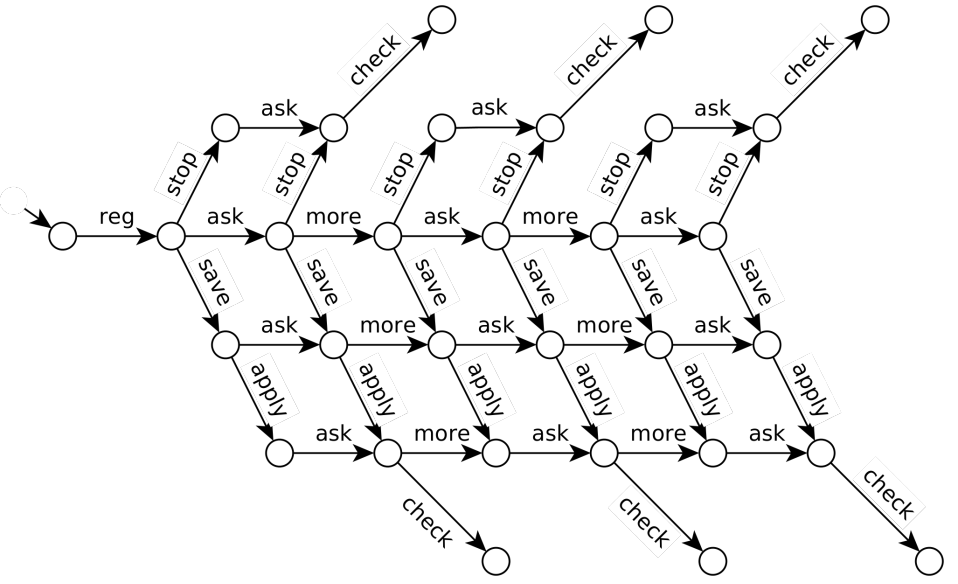


图2：示例工作流的基于状态的规约，表示所有可达的全局状态和可能的状态转移。

利用基于语言的区域理论，我们通过对系统的运行集合进行建模，来规定预期行为。运行是由一组无冲突的动作发生（称为事件）及其因果关系构成的。每个运行描述系统的一次执行，无法表达冲突。图3展示了我们工作流的相应规约。尽管每个运行都易于理解，但管理所有可能的替代执行却很困难。即使在这个小型示例中，我们也必须规定六个相互独立的运行，才能捕获该工作流的行为。在工作流的起点或终点仅增加一个替代选项，就会使运行数量翻倍。从建模角度来看，对于这个类似工作流的示例，我们或许更倾向于图3而不是图2。然而，即使系统模型复杂度仅小幅增加，转移系统的规模和运行数量也会迅速增长。

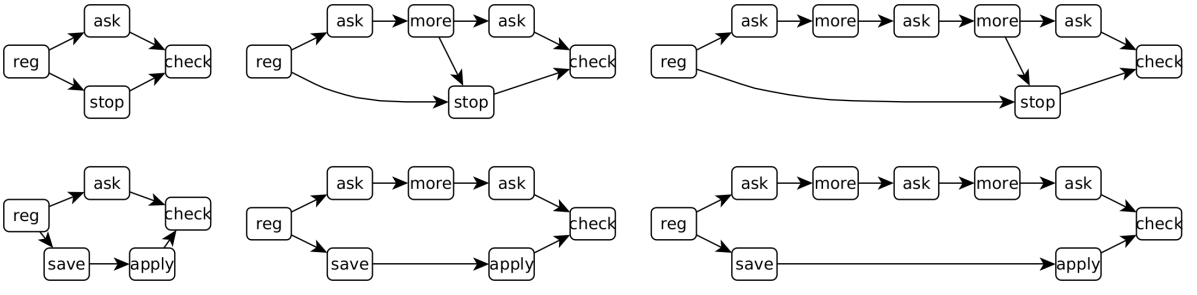


图3：示例工作流的基于语言的规约，将各次执行表示为无冲突的运行。

到目前为止，如果要应用合成，我们必须选择图2或图3作为起点。图2因并发而扩展，而图3则因冲突需要枚举备选运行而扩展。我们无法将这两种规约形式结合起来，也无法同时自然地表达并发与冲突。

本文通过构建一种统一的元区域理论来解决这一局限。我们设计这一理论，使其能够处理基于状态和基于语言的规约。此外，我们展示了这种方法能够在单个规约中混合并结合状态图与语言。我们还可以考虑基于网的输入。这实际上是区域定义中的关键新思路。它以标记佩特里网作为规约语言，使我们能够自然地表达冲突与并发，同时仍可分裂标签，并根据一组备选的、由示例行为构成的小部分建立完整规约。

回到我们的示例，假设我们使用图4所示的标记网来建模部分预期行为。注册使活动*ask*得以进行，并引入了*stop* 与*save*之间的选择。这并非自由选择，因为只有在*more*执行之后，才可能发生*stop* 。在这次执行中，活动*ask* 执行了两次，因此该标记网包含两个不同的、标签均为*ask*的事件。在*stop* 与*save* 之间进行选择不会影响后续行为，因为*save* 后接*apply* 以及*stop* 都会到达同一个局部状态，在该状态下变迁*check*处于启用状态。

因此，图4可以作为工作流示例规约的一部分。它描述了一组可能的执行，其中局部状态会发生分支与合并，同时有意只对图1所示整体行为的一个片段进行建模。通过标签拆分，可以直观地指定变迁*ask* 在这部分行为中发生两次。请注意，图4既不是状态图，也不是运行、过程网或分支过程。相反，它包含冲突、并发以及局部状态的合并。通过允许标签拆分，并聚焦于对预期行为的部分描述，图4比图1中的完整过程模型简单得多。

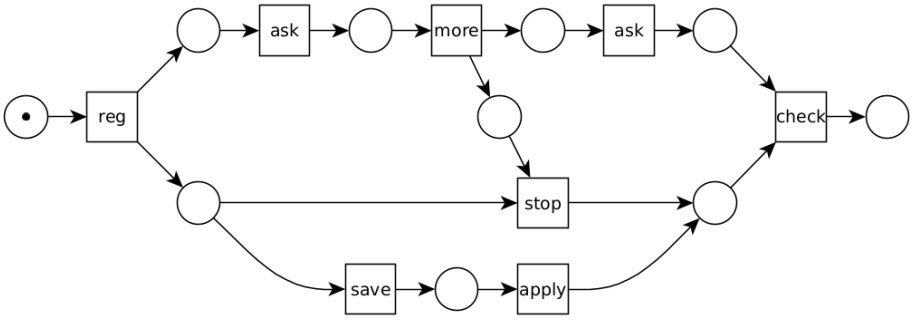


图4：该示例工作流的基于 Petri 网的规约，通过标签和局部状态表示部分预期行为，并体现冲突与并发。

如上所述，图 4 仅描绘了我们示例中预期行为的一部分。与图 3 类似，如果我们希望综合结果包含其他备选运行和执行，就必须对其加以规约

中。因此，我们构造了图 5 所示的组合规约。第一个标记网是一个简单的状态图，第二个标记网是一个简单的标记图。第三个标记网直接取自图 4，第四个则是一个类似的标记网，其中动作 *ask* 出现了三次。我们认为，与图 2 中完整的基于状态的规约或图 3 中完整的基于语言的规约相比，这个组合规约明显更容易设计；甚至比从头构造图 1 中佩特里网的蛮力方法还要简单。

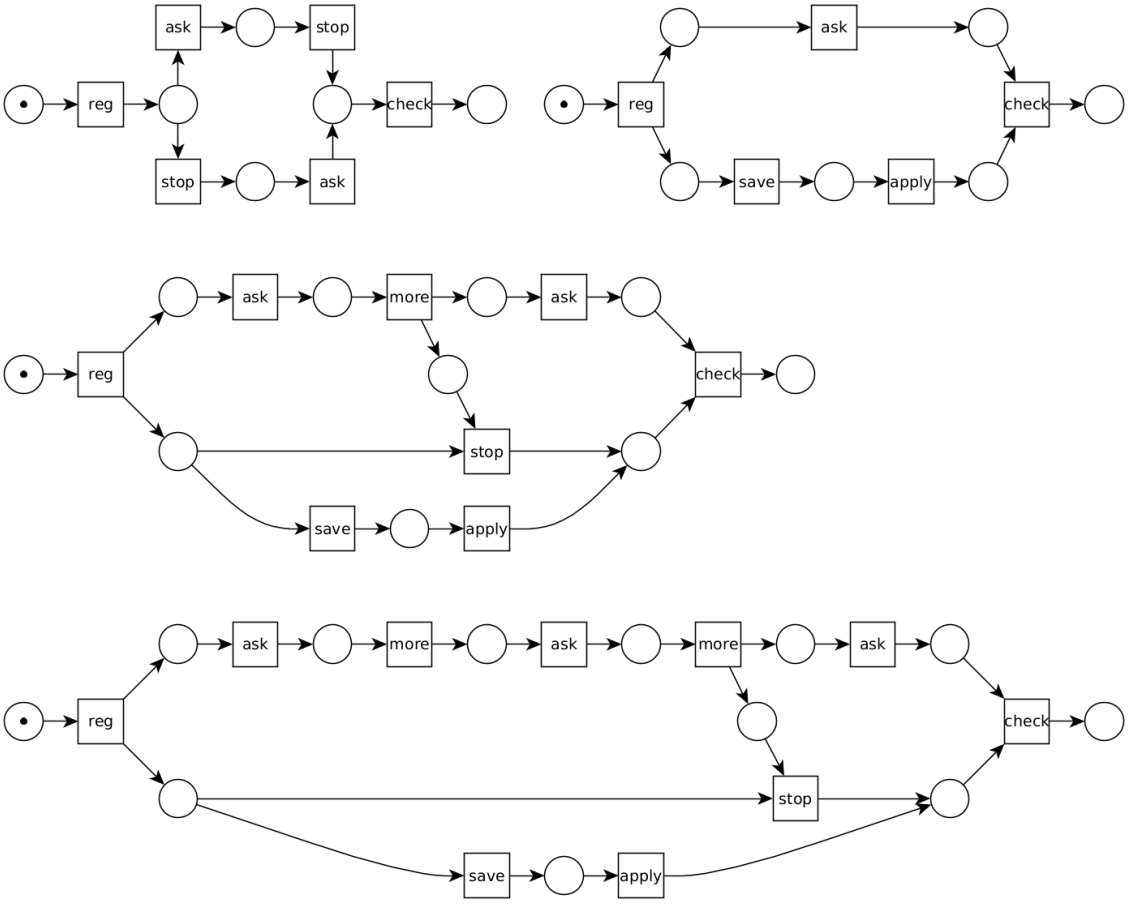


图 5：该示例工作流的完整规约，结合了基于状态、基于语言和基于网的模型。

我们邀请读者访问 www.fernuni-hagen.de/ilovepetrinets/horse/，并将标记为示例 *1*的框拖到按钮上。示例 *1* 对应于图 5 所示的规约。经过短暂计算后，该工具会合成图1所示的佩特里网。唯一的区别在于，变迁*reg* 会在短循环库所中生成令牌，从而减少初始标识。在界面中，综合结果显示在顶部，而规约的标记网则显示在下方的四个独立选项卡中。

让我们仔细看看图 5。第一个标记网是一个简单的状态图。事实上，它是将图 2 左上方的菱形一对一转换为标记网的结果。这个标记网只包含冲突，不包含并发。请注意，我们可以将整幅图 2 转换为一个大型标记网，从而获得对示例流程行为的完整规约。然而，这样做会遇到纯基于状态的方法所具有的相同缺点。在该工具中，这被作为示例*2*提供。如果我们重新加载该工具，并将示例*2* 拖到按钮上，就可以观察到规约规模的增加，以及综合结果不再能够识别出短循环库所。这是预期结果，因为这种短循环无法通过纯粹的全局基于状态的规约来描述。

图 5 中的第二个标记网是一个标记图。更准确地说，它是将图3中的第四个偏序运行一对一地转换为标记网的结果。这个标记网只包含并发，不包含冲突。同样，我们可以将整个图3转换为六个独立的无冲突标记网，从而完整规约示例流程的行为。然而，这样做仍会面临与纯粹基于语言的方法相同的缺点。在工具中，这以示例*3*的形式提供。如果重新加载工具，并将示例*3* 拖到按钮上，我们会发现现有又能够合成短循环库所了，但计算时间显著增加。在示例*3*中，由于只使用偏序，我们既无法利用冲突，也无法通过合并来保持输入简洁。

图 5 中第三个和第四个标记网是带有冲突、并发以及局部状态合并的标记网。在这个例子中，我们通过某种方式折叠图3中的偏序2和偏序5，得到第三个标记网。通过合并图3中的偏序3和偏序6，我们得到第四个标记网。不过，我们并不认为标记网在所有情况下都是描述行为的更好方式。在适合采用基于状态的建模时，我们只使用状态图；在更适合采用基于语言的建模时，我们只使用标记图。同时，我们也可以结合这些方法，并在适当时加入标记网。因此，我们有了更多选择。该工具的示例4提供了另一个例子，展示所提出的方法甚至能够处理包含循环和弧权重的标记网输入。

总的来说，我们认为，一组标记网为分布式系统的行为规约建模提供了一种灵活且非常合适的形式体系。本文从此类规约出发，探讨综合问题。请注意，在我们的设定中，我们综合出一个佩特里网，使其能够仿真规约中的 [24, 25, 26] 所有标记网，这对我们的贡献至关重要。在基于状态的综合中，输入是所综合佩特里网可达图的一个子图。在基于语言的综合中，输入是所综合佩特里网语言的一个子集。因此，二者所依据的语义有所不同。我们的区域定义通过要求所综合的网能够仿真所有输入，统一了这两种观点。也就是说，对于任意输入标记网，都存在一个线性映射，将其所有可达状态映射到综合结果的可达状态。该映射保持输入状态图中的初始状态和所有步迁移。若将规约限制为状态图或标记图，就可直接得到经典的基于状态或基于语言的综合表述。如果输入完全基于状态，则它是结果可达图的一个子图，仿真映射就是恒等映射。对于基于语言的综合，存在一个非平凡的结果：步语义与偏序语义相吻合 [27, 28]。如果模型能够像规约那样进行步迁移，那么输入就是综合结果语言的一部分。

请注意，我们要求的是仿真，而不是互模拟，因为每个输入网只对预期行为的一部分进行建模。因此，要求每个输入网都能够仿真综合结果是错误的。

新的区域定义建立在 [18, 29, 30]中引入的令牌轨迹语义之上。本文是会议论文 [31]的扩展版本。利用令牌轨迹，我们在第3章定义了相应的区域理论，并证明标记网为行为规范的建模提供了一种自然而紧凑的方式。在第4章中，我们表明该理论与基于状态的区域和基于语言的区域均一致，因此构成一种新的元区域理论。在第5章中，我们介绍如何使用整数线性规划计算综合结果，并介绍该工具中的相应实现。在第6章中，我们讨论该方法的应用，尤其侧重于过程发现。

2. 预备知识

设 $\mathbb{N}$ 为非负整数。设 f 为一个函数， B 为 f的定义域的一个子集。我们用 $f|_B$ 表示 f 到 B 的限制。照例，我们称函数 $m\colon A \to \mathbb{N}$ 为多重集，并用 $m = \sum_{a\in A} m(a) \cdot a$ 表示 m中元素的重数。设 $m'\colon A \to \mathbb{N}$ 为另一个多重集。我们记 $m \leq m'$ 当且仅当 $\forall a \in A\colon m(a) \leq m'(a)$ 成立。我们用佩特里网 [1, 2, 3, 4]对分布式系统进行建模。

定义 2.1. 佩特里网是一个元组 (P, T, W)，其中 P是一个有限的库所集合， T是一个有限的变迁集合，且满足 $P \cap T = \emptyset$， $W\colon (P \times T) \cup (T \times P) \to \mathbb{N}$是一个弧的多重集。$(P, T, W)$的一个标识是一个多重集 $m\colon P \to \mathbb{N}$。设 m_0是一个标识，我们称 $N = (P, T, W, m_0)$为一个标识佩特里网，并称 m_0 为 N的初始标识。

图1展示了一个标识佩特里网。变迁用矩形表示，库所用圆形表示，弧的多重集表示为一组带权弧，初始标识表示为一组黑点，称为令牌。佩特里网有一条发射规则。设 t 是标识佩特里网 (P, T, W, m_0) 的一个变迁。我们将 $\circ t = \sum_{p\in P} W(p, t) \cdot p$称为 t的加权前集。我们将 $t\circ = \sum_{p\in P} W(t, p) \cdot p$称为 t的加权后集。当且仅当满足 $m \geq \circ t$时，变迁 t才能在标识 m下发射。变迁 t发射后，佩特里网的标识从 m 变为 $m' = m - \circ t + t\circ$。在我们的示例标识佩特里网中，变迁 *reg* 可以在初始标识下发射。如果 *reg* 发射，它会从 p_1 中移除一个令牌，并在 p_2中产生一个新令牌，在 p_3中产生两个新令牌，并在 p_4 中产生一个新令牌。在这个新标识下，变迁 *ask*、*stop*和 *save* 可以发射。由于 p_1中不再有令牌，*reg* 不再被启用。

反复应用发射规则会产生所谓的发射序列。这些发射序列是佩特里网最基本的行为模型。例如，序列*reg askstop check* 在图1的标识佩特里网中是可触发的。序列*reg save apply ask check* 是另一个例子。设 N 为一个标识佩特里网， N 的所有可触发发射序列构成了N的顺序语言。

用于对佩特里网行为建模的另一种形式体系是可达图。如果存在一个触发序列能够产生某个标识，则该标识是可达的。

定义 2.2。设 $N = (P, T, W, m_0)$ 是一个标识佩特里网。N 的可达图是一个四元组 (R, T, X, m_0)，其中 R 是 N 的可达标识集合，T 是 N 的变迁集合，而 $X \subseteq R \times T \times R$ 是带标签状态转移的集合，使得 (m, t, m') 属于 X 当且仅当 t 在 (P, T, W, m) 中使能，且在 m 中发射 t 会到达标识 m'。若存在单射函数 $g\colon R' \to R$, $g(i) = m_0$, $T' \subseteq T$,$\forall (m, t, m') \in X' \colon (g(m)\ t, g(m')) \in X$，并且对于每个 $m' \in R'$，都存在一条从 i 到 m'、以 X' 中的元素为弧的有向路径，则称四元组 (R', T', X', i) 是在 N 中使能的状态图。我们称所有使能状态图构成的集合为 N 的状态语言。

粗略地说，状态图中的每个节点都对应一个可达状态；只要每个节点都能从初始节点到达，我们就不必纳入所有状态转移和状态。因此，状态图可以看作可达图的某种前缀。图2描绘了一个状态图，用于建模图1所示标识佩特里网的行为。该状态图包含31个状态和43条状态转移，这些状态转移以佩特里网的变迁为标签。状态图对佩特里网行为的描述如下：首先，我们必须触发变迁*reg*。然后，我们有几种选择：可以执行*stop*、*ask*或*save*。我们需要统计动作*ask* 出现的次数，而*stop* 之后就没有*more*了。最后，触发*check* 总会导致死锁。触发*check* 后，p_3中可能有两个、一个或零个令牌，而 p_5中可能有一个或零个令牌。状态语言将触发序列表示为图中所有路径的集合。从这个意义上说，状态图可以在共享状态处合并触发序列，也可能包含循环。不过，这些图无法直接表达并发。图1所示的初始标识中，触发*reg* 会产生标识 $p_2 + 2 \cdot p_3 + p_4 + p_5$。在该标识下，变迁*ask* 和*stop*，或*ask* 和*save* 可以并发触发，因为它们不共享令牌。触发序列和状态图都无法表达这种并发。因此，文献中还提出了其他能够明确表达并发的佩特里网语义，包括佩特里网的步语义 [32],、佩特里网的过程网语义 [33], 、佩特里网的令牌流语义 [28], 以及佩特里网的紧凑令牌流语义 [34]。幸运的是，这些语义是等价的 [35, 27, 28, 34] ，并且都定义了相同的部分语言。在部分语言中，每个所谓的运行都是一个事件偏序集。显然，运行能够表达并发，并为分布式系统的行为建模提供一种直观的方法。

使用紧凑令牌流语义，我们可以在多项式时间内判定某个运行是否属于佩特里网的部分语言 [36]。粗略地说，紧凑令牌流是运行的弧上的一种令牌分布，它确保每个事件都能获得足够的令牌、没有事件必须传递过多令牌，并且所有事件共享来自初始标识的令牌。

定义 2.3。设 T为一个标签集合。标记偏序是一个三元组（V、$\ll$、l），其中 V是事件的有限集合，$\ll \subseteq V \times V$是传递且反自反的关系，而标记函数 $l\colon V \to T$为每个事件分配一个标签。运行是一个三元组（V、$<$、l），带有关系 $< \subseteq V \times V$；当且仅当其反自反传递闭包（V、$<^*$、l）是一个标记偏序。设 $N =$（P、T、W、m_0）为一个标识佩特里网，且 $R =$（V、$<$、l）为一个运行，使得 l（V）$\subseteq T$成立。设 ▶、■ $\notin V$为两个符号。紧凑令牌流是一个函数 $x\colon ((\{\blacktriangleright\} \times V) \cup < \cup (V \times \{\blacksquare\})) \to \mathbb{N}$。设 $v \in V$为一个事件。我们记

$in_x(v) := x(\blacktriangleright, v) + \sum_{v'<v} x(v', v)$ 为 v的流入量，而 $out_x(v) := \sum_{v<v'} x(v, v') + x(v, \blacksquare)$ 为 v的流出量。我们定义： x 对于 $p \in P$有效，当且仅当以下条件成立：

(i) $\forall v \in V : in_x(v) \geq W(p, l(v))$,

(ii) $\forall v \in V : out_x(v) = in_x(v) + W(l(v), p) - W(p, l(v))$, and

(iii) $\sum_{v \in V} x(\blacktriangleright, v) = m_0(p)$.

R 在且仅在对于每个 $p \in P$都存在有效紧凑令牌流时，才可在 N 中执行。 N 的所有可执行运行的集合就是 N的部分语言。

图3描绘了六个不同的运行，用于对图1所示标识佩特里网的行为进行建模。每个运行都从执行变迁 *reg*开始。第一个运行对变迁*ask* 和*stop* 的并发执行进行建模，然后触发*check*。第二个运行对循环*askmore* 的执行进行建模，然后并发执行*ask* 和*stop*。由于库所 p_5的存在，变迁*stop* 只能在*more* 之后发生。第三个运行对循环执行两次的情况进行建模。第四、第五和第六个运行对*ask* 和*more*循环的执行与*saveapply*序列的执行并行进行的情况进行建模。图6描绘了图3最后一个运行中，针对图1中库所 p_1、 p_2、 p_3和 p_5 的四个有效紧凑令牌流。对于每个令牌流，相关库所定义了每个事件必须接收的令牌数量，以及事件可以传递给后续事件的令牌数量。紧凑令牌流是在运行的弧上分配令牌，同时遵循每个事件的需求和能力。此外，紧凑令牌流还会分配初始标识并收集所有多余令牌，以定义最终标识。

在图6中，第一个令牌流与库所 p_1相关，*reg* 从初始标识中消耗令牌。我们将该令牌显示在事件上方。第二个令牌流与库所 p_2相关，*reg* 为第一个*ask*产生一个令牌。之后，每次出现*more*，都会为后续的*ask*产生一个令牌。第三个令牌流与库所 p_3相关，*reg* 产生两个令牌，每次出现*more*时产生一个。在这里，我们可以清楚地看到令牌流如何传递给后续事件。第四个令牌流与库所 p_5相关，*more* 从初始标识中消耗令牌，并为第二次出现的*more*产生一个令牌。这个*more* 为最终标识产生一个令牌。我们将该令牌显示在事件下方。图3有六

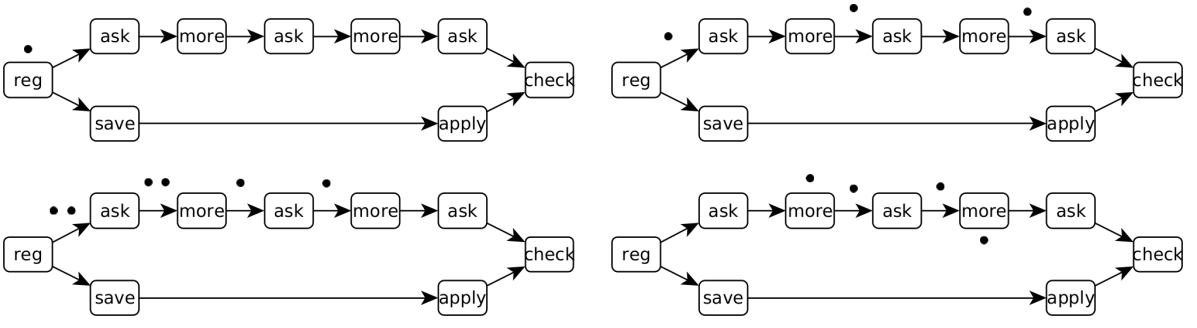


图6：图3最后一次运行中，分别对应库所 p_1、 p_2、 p_3 和 p_5 的四种有效紧凑令牌流。

次运行，而图1有九个库所。要证明图3属于图1的部分语言，我们需要54个有效紧凑令牌流。

总的来说，图2和图3共同刻画了图1中佩特里网的行为。图2无法表达变迁的并发性。图3需要六次独立的运行，因为运行不能包含冲突。因此，在选择其中一种语义时，总会存在某种权衡。

在文献中，我们还可以找到其他可能更先进的技术，用于对佩特里网的状态或执行进行建模。可选方案包括过程网 [33], 分支过程 [33, 4], 素事件结构 [37], Oclets [38], 和展开网 [39]。这些形式体系以状态图和运行作为基础，并加入步和/或条件与割，以增强建模技术的表达能力。然而，这也有一个缺点：技术越先进，对行为进行建模就越棘手。必须考虑如何指定所有可能的步、不同前置条件的组合以及其他因素。此外，这些技术虽然扩展了语义，却也使其变得更加复杂，因而成了另一种需要学习的高级建模语言。尽管这些语义通常能够明确表达冲突与并发，但仍无法直观地合并不同执行中的局部状态。例如，分支过程之所以称为分支过程，是因为它只能分支，不能合并。如果我们用分支过程来描述行为，模型就会不断分叉。因此，这类技术的主要应用是计算给定模型的完整行为。要用这类结构来描述行为并不容易。为简单说明这一问题，图7展示了图1的分支过程。显然，我们不希望用图7这样的形式来描述过程，从而合成图1。

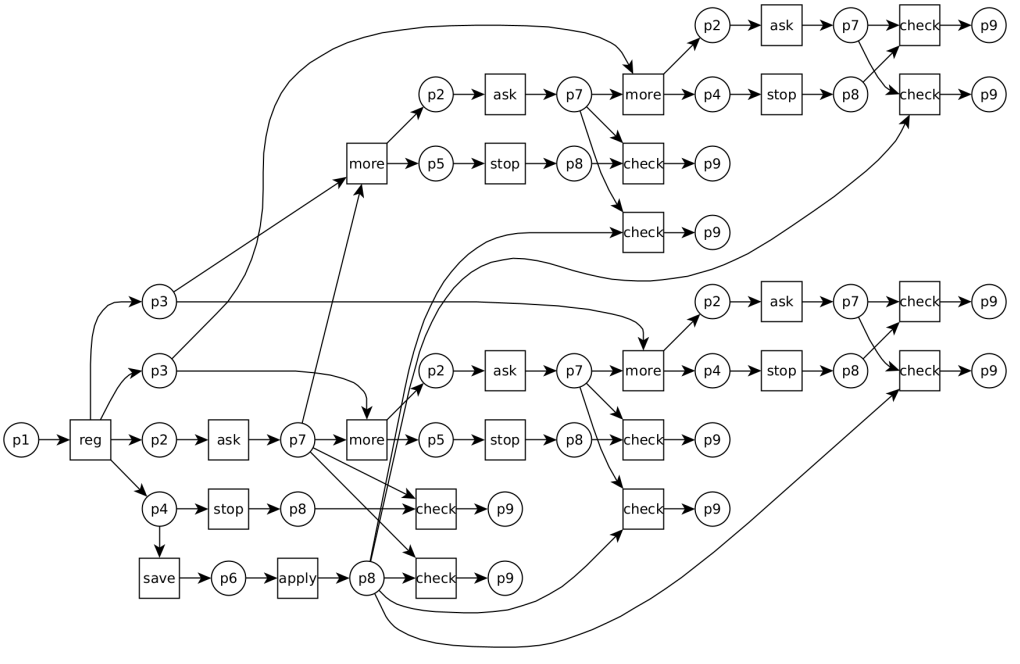


图7：图1的分支过程。

为了解决如何直观地对系统行为建模，并使其能够表达并发、冲突以及局部状态合并这一问题，我们引入了佩特里网的所谓网语言 [29, 30]。佩特里网模型的网语言，是所有满足以下条件的标记网的集合：该模型能够仿真其中的每一个标记网。在本文语境中，仿真意味着存在一个从语言中每个状态到模型中每个状态的映射；该映射遵循初始标识，并保持使能变迁多重集的状态变迁行为。粗略地说，模型能够表现得与规约完全一样。自然，网语言还包含所有使能触发序列、状态图、运行和分支过程。对于这些形式体系，都可以简单地逐一转换为标记网。因此，很容易证明，所得标记网也属于该网语言，这并不令人意外。另一方面，网语言并非不精确。网语言中所有网的步序列化集合，构成了模型的步语言。从这个意义上说，语言中不存在任何未经模型支持的额外行为。此外，我们已经知道如何用佩特里网建模，因此无需再学习另一种建模语言。详情请参阅 [29, 30], ，下面仅给出网语言的形式化定义。该定义的核心思想是引入证明仿真所需的线性映射，并将此映射表示为标记网的一个标识。与紧凑令牌流类似，该标识可以理解为规约各局部库所上的令牌分布。

定义 2.4. 设 T为一个标签集合。标记网是一个五元组 (C, E, F, i_0, l)，其中 (C, E, F, i_0) 是一个标识佩特里网，且 $l\colon E \to T$ 是一个标记函数，为标记网的每个变迁指定一个标签。设 $N = (P, T, W, m_0)$ 是一个标识佩特里网，$L = (C, E, F, i_0, l)$ 是一个标记标记网。令令牌轨迹 $x\colon C \to \mathbb{N}$ 为 L的一个标识。设 $e \in E$ 为一个标记变迁。我们将 $in_x(e) := \sum_{(c,e) \in F} F(c, e) \cdot x(c)$ 表示为 x中流入 e 的令牌流入量，并将加权和 $out_x(e) := \sum_{(e,c)\in F} F(e, c) \cdot x(c)$ 表示为 x中从 e 流出的令牌流出量。我们定义，x 对 $p \in P$有效，当且仅当满足以下条件：

(I) $\forall e \in E : in_x(e) \geq W(p, l(e))$,

(II) $\forall e \in E : out_x(e) = in_x(e) + W(l(e), p) - W(p, l(e))$, and

(III) $\sum_{c\in C} i_0(c) \cdot x(c) = m_0(p)$.

L 在且仅在对于每个 $p \in P$都存在有效托肯轨迹时，才在 N 中使能。N 的所有使能标号网的集合即为 N的网语言 $\mathcal{L}(N)$。

图 8 描绘了 4 条有效托肯轨迹，分别位于图 5 所示规约的每个标记网中，对应图1中的库所 p_3 。托肯轨迹是一种标识，与令牌流一样，用于描述一种有效的令牌分布，同时遵循每个事件的需求和能力。有效托肯轨迹证明，佩特里网中相应的库所能够模拟该标记网。在本例中，对于库所 p_3，只有带有 *reg* 标签的变迁能够生成令牌。因此，在每条托肯轨迹中，带有 *reg* 标签的变迁后集中的令牌数总和，减去同一变迁前集中的令牌数总和，恰好为 2。对于每个带有 *more* 标签的变迁，其后集中的令牌数比前集少 1，因为 *more* 会从佩特里网模型中的 p_3 消耗一个令牌。所有其他标签都不与 p_3 交互，输入托肯总数与输出托肯总数相同。

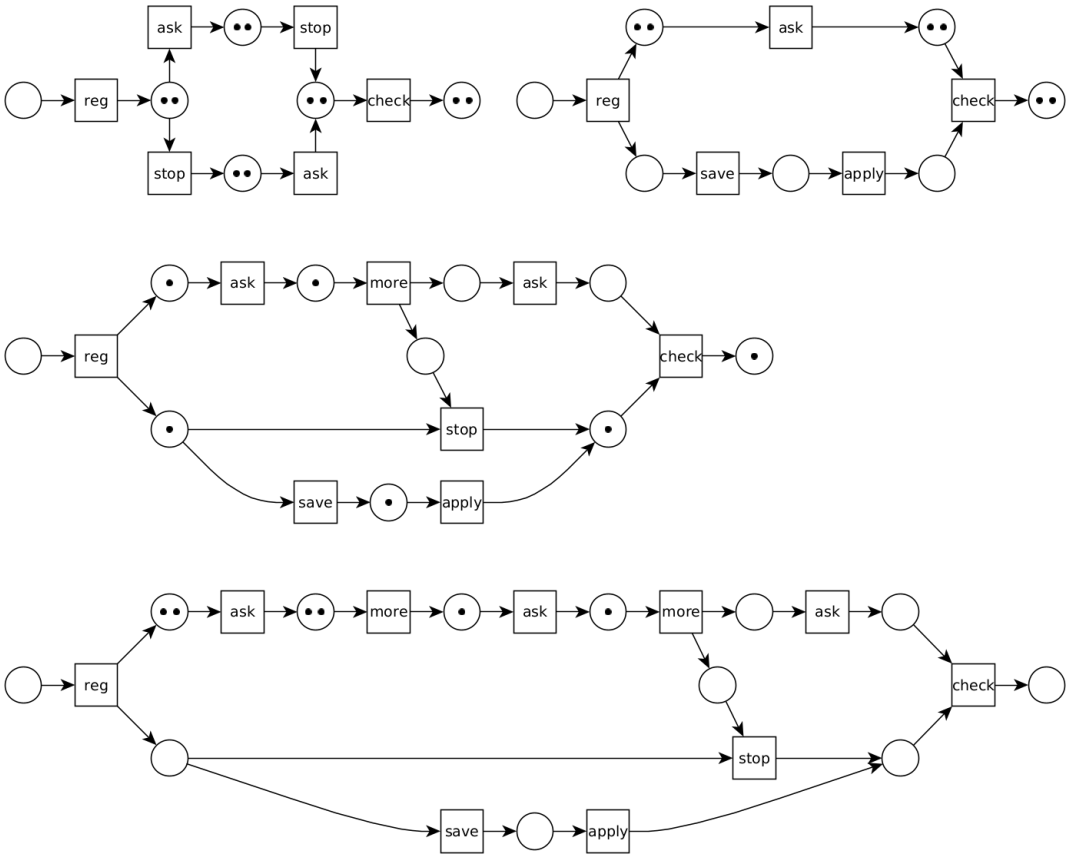


图8：图1的 p_3 在图 5的四个网中的四条有效托肯轨迹。

如果将托肯轨迹看作令牌分布，那么在图 8 的第一个网中可以看到，如果存在冲突，事件会以某种方式共享令牌。如果存在汇合，事件会对所传递的令牌数达成一致。在本例中，路径 *reg askstop* 和路径*regstop ask* 都会提供两个令牌。在图 8 的第二个网中可以看到，如果存在分裂，事件会分配令牌；如果存在汇合，事件则会收集令牌。不过，托肯轨迹中的令牌表示局部状态，它们不必对应于佩特里网中的单个令牌。这就是为什么用标记网而不是过程网或分支过程来描述行为要容易得多。在这里还可以看到，一个库所可能对应多条有效托肯轨迹，因为令牌可以通过 *ask* 或通过*save apply* 移动。在第三个网和第四个网中，令牌的分裂、汇合、冲突和合并交织在一起。在第三个网中，其中一个令牌被传递给 *more* 并被其消耗。在第四个网中，每个 *more* 都接收并消耗一个由*reg* 生成的令牌。

为了证明图 5 中的四个网属于图1的网语言，我们还必须为图1的其他八个库所找到有效托肯轨迹。这里请读者访问我们的网络工具 www.fernuni-hagen.de/ilovepetrinets/fox/，我们已准备好本例中的每个标记网，可将其拖放到图标上。点击佩特里网中的库所，即可显示相关的

标记网中的有效托肯轨迹。关于更多示例，以及令牌轨迹确实涵盖状态语言和部分语言、网语言中的每个网都能由佩特里网模拟的证明，请参阅 [29, 30]。请注意，网语言中的标记网也可能包含循环、弧权重和分布式初始标识。

3. 令牌轨迹区域

本节讨论一组标记网的综合问题。综合问题是根据一组标记网计算出一个 Petri 网，使得：(A) 每个标记 Petri 网都属于所合成 Petri 网的网语言；(B) 所生成的模型具有最少的额外行为。如上所述，为了解决这一问题，我们严格遵循区域理论的一般思路。

定义 3.1. 设 $S = \{(C_1, E_1, F_1, i_1, l_1) \ldots, (C_n, E_n, F_n, i_n, l_n)\}$ 为一组带标记的标记网。令牌轨迹区域是一个标识 $r : \bigcup_i C_i \to \mathbb{N}$，当且仅当满足以下两个条件：

(IV) $\forall \nu, \mu \in \mathbb{N}, e \in E_\nu, e' \in E_\mu, l_\nu(e) = l_\mu(e') : out_r(e) - in_r(e) = out_r(e') - in_r(e')$ and

(V) $\forall \nu, \mu \in \mathbb{N} : \sum_{c \in C_\nu} i_\nu(c) \cdot r(c) = \sum_{c' \in C_\mu} i_\mu(c') \cdot r(c')$.

对于每个标记变迁 e，我们将 $out_r(e) - in_r(e)$ 称为 e 的增量。利用这一概念，性质 (IV) 确保：相同标签，相同增量。对于每个标记网，我们将 $\sum_{c \in C_\nu} i_\nu(c) \cdot r(c)$ 称为初始令牌总数。性质 (V) 确保：所有网的初始令牌总数相同。

显然，定义 3.1 的目的是在一组标记网上定义一种令牌分布，使得标签相同的变迁对整体分布产生相同的影响（相同标签，相同增量）。如果满足这一条件，对于每一种这样的分布，我们都可以直接在待构造的佩特里网中定义一个对应的库所，使连接该库所与所有变迁的弧权重直接源自每个标签对整体分布的影响。初始令牌总数直接决定了这个有效库所的初始标识。根据构造，该区域是此类库所的有效令牌轨迹。因此，在综合结果中只添加由区域推导出的库所，就能保证 (A)。

定理 3.2。设 $S = \{(C_1, E_1, F_1, i_1, l_1) \ldots, (C_n, E_n, F_n, i_n, l_n)\}$ 是一组带标记的标记网，设 r 是 S 中的一个区域。我们记 $E = \bigcup_i E_i$ 和 $l = \bigcup_i l_i$。对于每个 $t \in \{l(e) | e \in E\}$，我们固定一个 $e_t \in E$，使得 $l(e_t) = t$ 且 $in_r(e_t) = min\{in_r(e) | l(e) = t\}$。佩特里网 $N = (P, T, W, m_0)$，其中 $P = \{p\}$, $T = \{l(e) | e \in E\}$, $W = \sum_{t \in T} in_r(e_t) \cdot (p, t) + \sum_{t \in T} out_r(e_t) \cdot (t, p)$，且 $m_0(p) = \sum_{c \in C_1} i_1(c) \cdot r(c)$，定义良好。对于每个 $(C_\nu, E_\nu, F_\nu, i_\nu, l_\nu) \in S$，$r|_{C_\nu}$ 是 N 的 p 的有效托肯轨迹。因此，$S \subseteq \mathcal{L}(N)$ 成立。

证明：
固定某个 ν，我们证明 $r_\nu := r|_{C_\nu}$ 是 $(C_\nu, E_\nu, F_\nu, i_\nu, l_\nu)$ 中针对 N 的 p 的有效令牌轨迹。 $\forall e \in E_\nu$： $W(p, l(e)) = in_r(e_{l(e)}) \leq in_{r_\nu}(e)$，因此 (I) 成立。 $\forall e \in E_\nu$： $W(l(e), p) - W(p, l(e)) = out_r(e_{l(e)}) - in_r(e_{l(e)}) = rise_r(e_{l(e)}) = rise_{r_\nu}(r)$，因此 (II) 成立。 $m_0(p) = \sum_{c \in C_1} i_1(c) \cdot r(c) = \sum_{c \in C_\nu} i_\nu(c) \cdot r_\nu(c)$，因此 (III) 也成立。区域 r 是 S 中每个标记网里针对 N 的 p 的有效令牌轨迹。因此，$S \subseteq \mathcal{L}(N)$ 成立。□

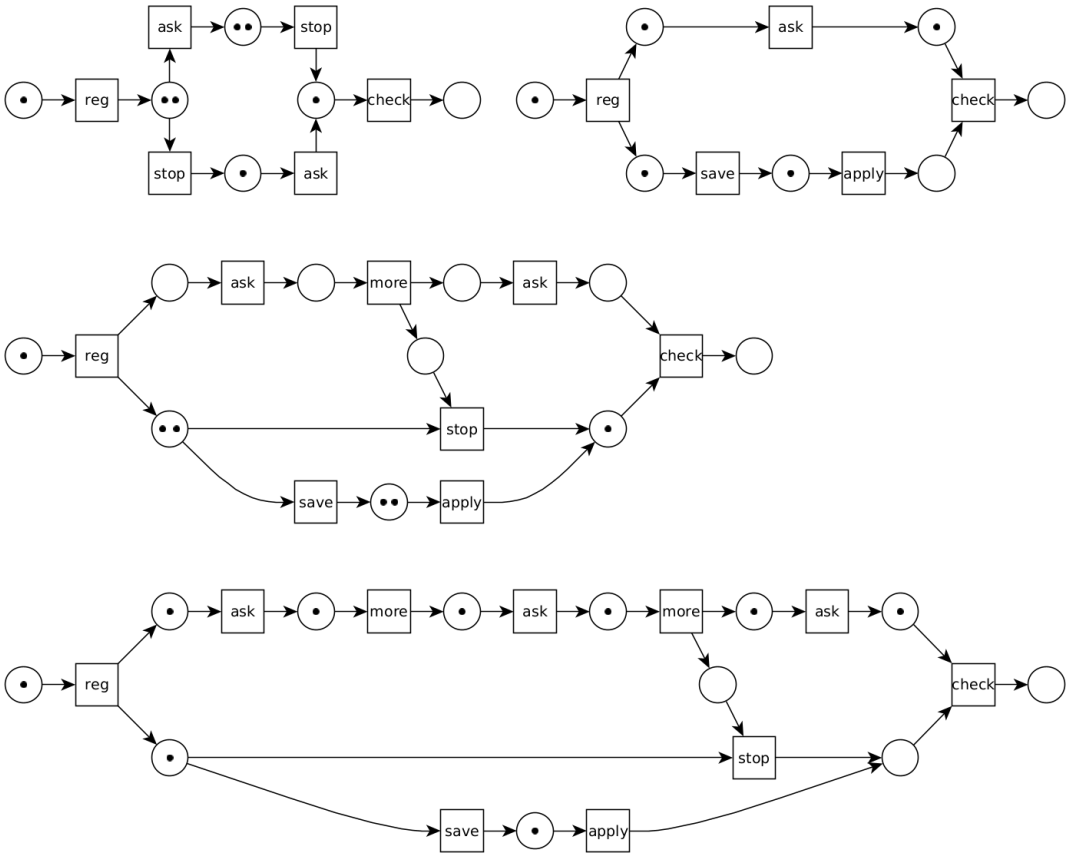


图9：图 5 中的一个令牌轨迹区域。

定理 3.2 表明，只要有一个区域，我们就可以构造一个相关的单库所 Petri 网，使该区域成为该库所的有效令牌轨迹，因此该网语言包含该规约。图9展示了我们规约中的标识，使得 (IV) 和 (V) 成立。图 5 中所有初始标识的库所都各有一个令牌。所有标记为 *reg* 的变迁，其增量均为 1。所有标记为 *stop*、*apply* 或 *check* 的变迁，其增量均为 -1。所有其他变迁的增量均为 0。

根据定理 3.2，该区域是图 10 所示库所的有效托肯轨迹。该库所初始时有一个令牌，触发 *reg* 将消耗此令牌，然后在该库所中产生两个令牌。触发*stop*、*apply*或*check* 将消耗一个令牌。变迁*save*在该库所上有一个短循环，所有其他变迁则未连接。将此库所添加到图1后，所得网的网语言仍将包含图 5。

推论 3.3. 设 S 是标记网的集合， R 是 S中的区域集合。根据定理 3.2 的定义，每个区域 $r \in R$ 都定义了一个单库所 Petri 网 N_r。令 N 为所有佩特里网 N_r的并集，根据定义 2.4，$S \subseteq \mathcal{L}(N)$成立。

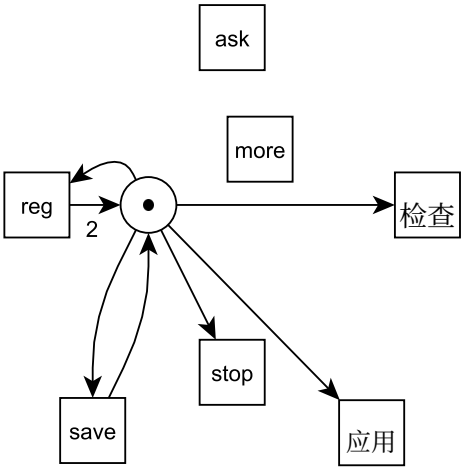


图 10：由图9的区域定义的单库所 Petri 网。

推论 3.3 为解决综合问题提供了第一个拼图片段。我们只将与区域相关的库所添加到综合结果中，以确保 (A)。接下来，我们将证明无需添加任何未由区域定义的库所。不过首先，我们需要讨论短循环。图 11 展示了下一个定理的核心思想。第一行中有一个佩特里网，以及该佩特里网的网语言中的一个标记网。因此，存在一条适用于 p的有效令牌轨迹，如图第二行所示。这条令牌轨迹自然形成一个区域，该区域定义了第二行中的佩特里网，其弧权重相匹配。每个区域定义的库所都会使弧权重最大化，但针对 p 的令牌轨迹仍然也是适用于 p' 的有效令牌轨迹。 p' 只是比 p限制更多，因此无需添加并非由区域直接定义的库所。

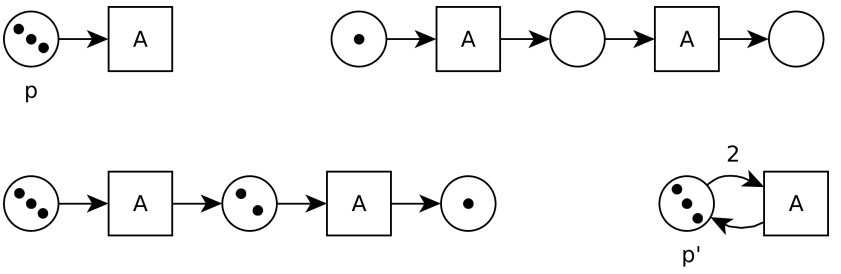


图 11：一个佩特里网、一个带标识的标记网、一条有效托肯轨迹，以及一个单库所 Petri 网。

定理 3.4。设 $S = \{(C_1, E_1, F_1, i_1, l_1) \ldots, (C_n, E_n, F_n, i_n, l_n)\}$ 为一组带标记的标记网。设 $N = (\{p\}, T, W, m_0)$ 为一个单库所 Petri 网。若 $S \subseteq \mathcal{L}(N)$ 成立，则存在 S 中的一个区域 r，定义一个单库所网 $N' = (\{p'\}, T, W', m_0')$，使得 $S \subseteq \mathcal{L}(N') \subseteq \mathcal{L}(N)$ 成立。

证明:
若 $S \subseteq \mathcal{L}(N)$ 成立，则 S 中每个标记网都存在一条适用于 p 的有效托肯轨迹。这些标识共同构成 S 中的一个区域 r，原因如下：(III) 对每个标记网都成立，因此 (V) 也成立。(II) 对所有标记变迁都成立，因此 (IV) 也成立。根据 (I)，在每条令牌轨迹中，每个标记变迁 t 至少有 $W(l(t)\ p)$ 个输入托肯，但也可能更多。根据定理 3.2，我们构造相关的单库所网 $(\{p'\}, T, W', m'_0)$。为定义 W'，对于每个 t，我们取所有标记为 t 的变迁的输入托肯数之和的最小值。因此，对于每个 t，$W(p', l(t)) \geq W(p, l(t))$ 成立。r 是 p' 的一条有效托肯轨迹（定理 3.2）。因此，p' 的每条有效托肯轨迹也是 p 的一条有效托肯轨迹，从而 $S \subseteq \mathcal{L}(N') \subseteq \mathcal{L}(N)$ 成立。库所 p' 就是 p，其中短环被最大化，以获得限制最严格的库所。□

结合定理 3.2 和定理 3.4 可知，为了解决综合问题，我们要计算区域，并组合相应的单库所 Petri 网。我们无需添加任何未由区域定义的库所。

推论3.5。设 S 为一组标记网，R 为 S中的一组区域。R中的每个区域 r 都定义一个单库所 Petri 网 N_r。由这些单库所 Petri 网 N_r构成的可能无限的并网 N 可解决综合问题。

证明:
假设我们有由单库所 Petri 网 N_r 组成的可能无限的并网 N，以及一个尚未属于 N的库所 p。如果将 p 添加到 N 后仍能确保 (A)，那么 S中就存在 p 的有效令牌轨迹。该令牌轨迹定义了一个区域，因此，相关库所 p' 已经属于 N。根据定理 3.4，无需添加 p。□

在实践中，为了解决综合问题，我们需要构造一个有限佩特里网，使其行为与并集佩特里网相同。回顾区域理论相关文献后可以发现，解决这一最后步骤有多种方法。对于大多数基于状态的区域，在构造不带弧权重的佩特里网时，区域的数量自然是有限的。如果考虑基于状态的区域以及带弧权重的佩特里网，我们通常会尝试在额外约束——不引入额外行为——下解决综合问题。如果存在这样的结果，那么所得网的状态空间就是规约，因此只需添加极小区域。对于基于语言的区域，我们旨在获得所规约行为的最佳上近似。因此，即使某个区域就规约行为而言更大，就未规约行为而言却可能更小。因此，极小区域可能并不足够。在这里，我们计算错误延续的集合，它是未规约行为与已规约行为之间的边界。对于基于偏序的规约，错误延续的集合是有限的，因此我们可以确定综合问题是否存在精确解。如果可行，在确保条件 (A) 的同时排除所有错误延续，即可解决综合问题。

在本文中，我们为一类标记网定义区域。因此，行为是根据佩特里网的网语言来规约的。网语言总是无限的。添加不会限制行为的多余附加库所，或者简单地复制一个网并使副本与原网发生冲突，始终是可能的，也理应如此。因此，由于规约总是有限的，而

每个综合结果都会有无限的网语言，从技术上说，精确解从来不存在。总的来说，由于弧权重的存在，区域的数量是无限的；而由于网语言的存在，错误延续的数量也是无限的。因此，尽管含有与所有区域相关库所的饱和网仍然是一个解，但有限子网可能并非如此。尽管这听起来不是好消息，但我们其实并不关心某个佩特里网完整的无限网语言。我们希望用状态图、部分语言和标记网来描述行为，并获得恰当的综合结果：既能模拟规约，又属于某个合理的网类。本文表明，一种非常实用的简化方法能够带来很好的结果。与基于状态的区域方法一样，我们只需将区域的每个局部标识的大小限制在某个数值 k 以内。这样，区域的数量便再次变为有限。在这一过程中，每个解都满足 (A)，并且不存在行为更少的 k 有界解。随着 k 增大，附加行为会减少。

4. 令牌轨迹区域作为元区域

本节将证明，令牌轨迹区域构成状态型区域和基于语言的区域的元理论。

将状态图转换为标记网很容易。每个状态都变为一个库所，每个状态变迁都变为一个标记变迁。与初始状态对应的库所在初始标识中带有一个令牌。根据构造，所得网的可达图就是原状态图。图 5 中的第一个网展示了这样一个标记网。它的可达图就是网本身。

定理 4.1. 设 $S = (R, T, X, i)$ 是一个状态图，r 是 S 中的一个区域。我们通过 $C = R, E = X$，$F = \sum_{(s,t,s')\in X}(s, (s, t, s')) + ((s, t, s') \ s')$，$m_0 = i$，并令 l 将每个 (s, t, s') 映射到 t，构造一个标记网 $N = (C, E, F, m_0, l)$。r 是 (C, E, F, m_0, l) 中的一个令牌轨迹区域。

证明：
r 是 S 中的一个区域，因此 r 是 R 的子集，并且每个标签相同的状态变迁要么进入 r，要么离开 r，要么不穿越 r。如果状态变迁（s, t, s'）离开 r，则 $s \in r$ 和 $s' \notin r$ 成立。在 N中，r是一个标识，s 和 s' 是库所。r 在 s 中标记一个令牌，在 s' 中不标记任何令牌。在 N中，（s, t, s'）是一个仅与 s 和 s'相连的变迁。（s, t, s'）在 r 中的增量为 -1。用相同的论证可知，如果状态变迁（s, t, s'）进入 r，则（s, t, s'）在 r 中的增量为 1。如果状态变迁（s, t, s'）不发生穿越，则变迁（s, t, s'）在 r 中的增量为 0。在 S中，标签相同的状态变迁具有相同的穿越行为，因此，在 N中，标签相同的变迁具有相同的增量，即（IV）成立。（V）成立，因为我们只考虑一个初始令牌总数为 1 的网。□

显然，我们可以将定理 4.1 扩展到带弧权重的基于状态的区域。这里，区域是状态的多重集，它们同样可以完美地映射为令牌轨迹区域的标识。图 12 左侧展示了基于状态的综合算法的输入，以及一个使用状态多重集的区域。浅色阴影的状态在区域中出现一次，深色阴影的状态则出现两次。*reg*进入区域一次，*stop* 离开区域一次，而其他所有状态变迁都不穿越该区域。右侧是相关的标记网及对应的令牌轨迹区域。这只是同一个概念。假设我们有一个标记网，它直接对状态图进行建模，并且有一个令牌轨迹区域。

使用定理 4.1 中定义的相同关系（图 12 中已突出显示），标记网中的令牌轨迹区域对应于状态图的一个区域。对于状态图而言，每个区域都是一个令牌轨迹区域，反之亦然。

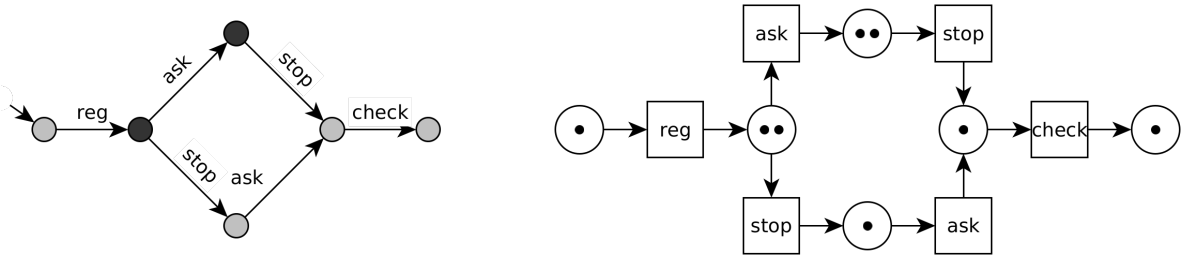


图 12：一个基于状态的区域和一个相同的令牌轨迹区域。

在定理 4.2 中，我们对基于语言的区域得到了相同的结果。这里，我们利用了部分语言上的令牌轨迹与令牌流之间的一一对应关系。这一结果已在 [29]中提出。我们重述该定理，以展示这两种相关的区域定义也是等价的。

定理 4.2。设 $S = \{(V_1, \ll_1, l_1) \ldots, (V_n, \ll_n, l_n)\}$ 为一组标记偏序，且 r 是 S中的一个紧致令牌流区域。我们通过 $C_i = (\{\blacktriangleright\} \times V_i) \cup \ll_i \cup (V_i \times \{\blacksquare\})$, $E_i = V_i$, $F_i = \sum_{(v,v') \in \ll_i} (v, (v, v')) + ((v, v') \ v') + \sum_{v \in V_i} ((\blacktriangleright, v) \ v) + (v, (v, \blacksquare))$, and $m_i = \sum_{v \in V_i} (\blacktriangleright, v)$构造一组标记网 $N = \{(C_1, E_1, F_1, m_1, l_1) \ldots, (C_n, E_n, F_n, m_n, l_n)\}$。 r 是 N 中的一个令牌轨迹区域。

证明:
我们证明部分语言上的令牌流与令牌轨迹之间存在一一对应关系。(i)、(ii)、(iii) 与 (I)、(II)、(III) 之间唯一的区别在于 in 和 out 的定义。在定义 2.4 中，各项求和时会乘以相应的弧权重。然而，要将标记偏序建模为标记网，并不需要弧权重。在这种情况下，所有权重均为 1。在定义 2.3 中，每个事件都可以消耗和产生初始令牌及多余令牌。在本定理中，我们将匹配库所加入每个标记偏序的前集和后集。因此， r 是 S 的令牌轨迹区域。□

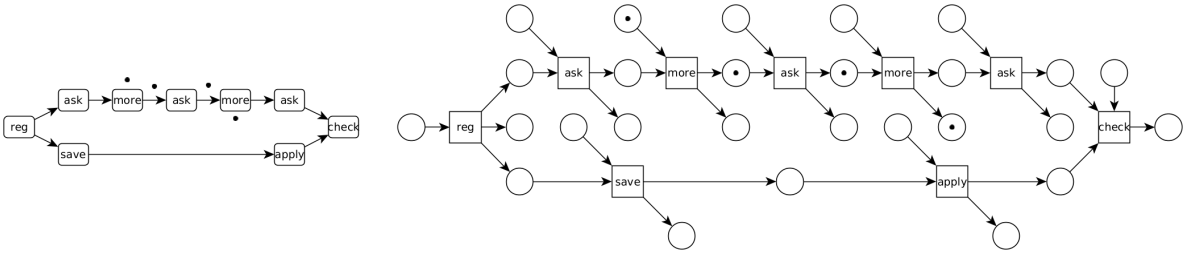


图 13：一个基于语言的区域及与之相同的令牌轨迹区域。

图 13 描绘了部分有序运行上有效紧凑令牌流与有效托肯轨迹之间的关系。不难看出，我们可以移除大量用于建模的附加库所

初始流与最终流。该网是一个偏序，因此初始标识中的令牌可以在极小库所中产生，并传递给每个事件。类似地，多余令牌可以传递到极大库所。因此，如果在这样的扩展网上存在令牌轨迹，那么在仅建模偏序、并为初始标识和最终标识添加极小库所与极大库所的网上也存在令牌轨迹。请参见图9中的第二个网作为示例。

假设我们有一组标记网，它们直接对部分语言建模，并且有一个令牌轨迹区域。使用定理 4.2 中定义的同一关系（图 13 中对此作了强调），标记网中的令牌轨迹区域定义了部分语言的一个区域。对于部分语言而言，每个区域都是一个令牌轨迹区域，反之亦然。

总结本节，正如 [18]，中所推测的，令牌轨迹区域是一种元区域定义。我们可以直接使用令牌轨迹区域，而不是基于状态的区域。偏序区域也同样如此。此外，我们可以使用极小区域或错误延续来获得有限结果。然而，本文的目标不仅是通过元区域定义统一这两个概念，还要引入一种适用于实际应用的新方法。采用这一新定义，我们不仅能够结合基于状态的输入和基于语言的输入，还能将标记网纳入规约。这些标记网可以具有分布式初始标识、循环，甚至弧权重。接下来，我们将介绍一种方法，用于计算上限为 k 的一组极小区域，以解决基于一组标记网的综合问题。

5. 计算令牌轨迹区域

本节介绍一种方法，用于计算给定上界k内的所有极小令牌轨迹区域。我们将介绍一种实现和一个 Web 工具，并提供综合结果示例，以证明该综合方法确实适用于实际场景。

通常，基于状态的区域采用自底向上方法构造。从一组所谓的激励区域开始，通过逐步添加更多状态来逐渐修复这些区域，直到形成一组有效区域。自底向上方法会生成一组最小区域。类似的过程也可以用于令牌轨迹区域。然而，由于标记网中的并发会在修复区域时引入许多替代方案，自底向上方法可能会变得相当繁琐。

大多数基于语言的综合算法会使用整数线性规划（ILP），并结合错误延续方法来识别所有可能的区域。例如，参见 [15]，这是最著名的算法，它运用基于语言的区域、ILP，以及基于输入的直接跟随关系的启发式方法来生成有限的区域集合。Kokosminer 基于 Karl Heinrichmeyer 的硕士论文 [40], , 通过 ILP 直接计算一组极小区域。在我们的实现中，我们将 Karl 的方法加以推广，以计算令牌轨迹区域。

整数线性规划易于使用专用求解器实现，而且具有很高的灵活性。它能直观地实现对目标区域类型的限制，例如单界区域，或对目标网类的限制，例如带有或不带弧权重和／或短循环的网。基于这些原因，本文结合两种经验证的最佳技术，并提出一种用于计算最小区域的新整数线性规划方法。

为计算区域（形式上即库所的多重集），我们将定义 3.1 中的条件实现为整数线性规划。规约中的每个库所都是整数线性规划中的一个变量。对于规约中的每个标签，我们固定一个具有相关标签的变迁。对于其他每个具有相同标签的变迁，我们在整数线性规划中引入一个等式，以确保两个变迁的增量相等。以图 5 为例。规约有 35 个库所，因此整数线性规划有 35 个变量。有四个标记为 *reg* 的变迁。我们将它们称为 reg_1, reg_2, reg_3, 和 reg_4。我们在整数线性规划中加入三个等式。一个等式确保 reg_1 的增量等于 reg_2 的增量。另一个等式确保 reg_1 的增量等于 reg_3 的增量。再一个等式确保 reg_1 的增量等于 reg_4 的增量。有八个标记为 *ask* 的变迁，因此我们再加入七个等式。总之，我们针对每个标签都加入这些等式。它们确保整数线性规划的每个解都满足 (IV)。

下一步，我们加入等式以确保 (V)。同样，我们将规约中的所有标记网简单地记作 网$_1$, 网$_2$, ...网$_n$。我们构造一个等式，使 网$_1$ 和 网$_2$ 的初始令牌之和相等。另一个等式确保 网$_1$ 和 网$_3$ 的初始令牌之和相等，依此类推。它们确保整数线性规划的每个解都满足 (V)。

我们为每个变量添加一个约束，确保其小于或等于 k，从而仅计算不超过 k 的区域。

总而言之，该区域整数线性规划的每个非负整数解都是不超过 k 的令牌轨迹区域，反之亦然。这个构造确实相当直接，因为定义 3.1 本身就是一个整数线性规划。

接下来，我们搜索一个非空的极小区域。我们添加一个不等式，要求所有变量之和大于 0，并将同一个和（即所有变量之和）作为目标函数，交由整数线性规划求最小值。现在，每个最优解都对应一个极小区域，因为不存在其他区域能在每个库所中都拥有更少或相同数量的令牌；否则，该区域的目标函数值会更小。

利用第一个极小区域，我们按照定理 3.2 的定义，为每个标签构造一个变迁，以及弧和一个库所。现在，我们必须找到其他极小区域及相关库所。

我们找到的每个区域都是一个解 $(s_1, \ldots, s_n)$。我们就扩展整数线性规划。假设我们刚找到一个极小区域$r_1 = s_1 \cdot p_1 + \ldots + s_n \cdot p_n$。对于 r_1的每个重数 $s_i > 0$，我们都在整数线性规划中添加一个变量 x_{r1pi}。我们的想法是添加不等式，使得当变量 p_i 小于 s_i时， x_{r1pi} 取值为 1，否则取值为 0。因此，我们添加以下不等式：

$$0 \leq (p_1 - s_1) + k \cdot x_{r1p1} \leq k - 1$$
$$\ldots$$
$$0 \leq (p_n - s_n) + k \cdot x_{r1pn} \leq k - 1$$

如果变量 p_i 小于前一个解 s_i，则项 $(p_i - s_i)$ 小于 0，因此 x_{r1pi}必须为 1。如果 p_i 大于或等于 s_i，则项 $(p_i - s_i)$ 非负，因此 x_{r1pi} 必须为 0。最后，我们将$x_{r1p1} + \ldots + x_{r1pn} > 0$ 添加到整数线性规划中，以确保每个新解至少在一个分量上小于前一个解。

通过反复构建、求解和更新整数线性规划，我们可以为规约计算出截至 k 的最小区域集合，随后构造相关的佩特里网。

这一综合方法作为 I Petri网 网站的一个新模块实现。该 ❤ 🐴

模块位于 www.fernuni-hagen.de/ilovepetrinets/horse，通过使用 GLPK 求解器 [41, 42] 求解一系列整数线性规划，实现了本章介绍的算法。为此，我们准备了四个示例，读者可以尝试使用我们新的区域方法来求解。

我们已在引言中提及示例 1 至示例 3。示例 1（图 5 中基于网的规约）在 30 秒内合成出图1。示例2（图2中的基于状态的规约）无需 p_5，瞬间便合成出图1。基于状态的输入无法描述短循环。示例3使用图3中基于语言的规约的四个运行，在 6 分钟内合成出图1。基于语言的区域无法在合理时间内处理全部六个运行。示例4展示了令牌轨迹方法的另一种应用。显然，每个佩特里网也是一个标记网。因此，合成的输入可以是一个完整的过程模型。在示例4中，我们输入了两个一般的标记网，一个具有分布式初始标记，另一个具有弧权重。我们在转瞬之间便合成出图1。这是令牌轨迹区域的一项令人振奋的特性。我们可以输入序列、状态图、部分有序运行、工作流网和完整的系统模型，合成出能够模拟所描述行为的佩特里网。

我们鼓励读者尝试更多示例。如果删除示例 1 中最长的运行，其中*more* 出现两次，结果将不包含弧权重。如果输入一个未标记网，结果将与输入完全相同。我们可以将示例4中的两个网添加到示例 1，而不会改变结果。可以使用 *PNML* 标准上传输入，也可以使用 I ❤ Petri Nets 网站提供的编辑器创建输入，网址为 www.fernuni-hagen.de/ilovepetrinets/。

6. 使用令牌轨迹区域进行过程发现

在前几节中，我们介绍了令牌轨迹区域，它是一种统一的区域理论，可根据一组标记网所构成的行为规约综合出佩特里网。迄今为止，我们的重点一直是理论基础和综合问题本身；现在，我们将转向应用。具体而言，我们将讨论过程发现，这是基于区域的合成技术最重要的实际应用。

过程发现 [1, 43, 44] 旨在根据观察到的示例行为自动生成过程模型。在大多数实际场景中，这类示例行为都以事件日志的形式记录。事件日志由轨迹的多重集构成，每条轨迹都是一个有限的活动标签序列，用于描述该流程的一次已观察到的执行。根据抽象层次的不同，还可能包含时间戳、资源或数据属性等附加信息，但经典的发现方法侧重于控制流视角。

在基于区域的发现方法中，ILP Miner [15] 是最具代表性的算法。它将基于语言的区域应用于由事件日志诱导的前缀闭语言，并将综合问题表述为一个整数线性规划，以计算工作流网的库所。基于区域的发现算法（例如 [45, 16, 46, 47] ）以能够生成精确且拟合良好的模型而著称，这与往往以牺牲精确度换取性能的启发式方法形成对比。然而，这些优势也伴随着代价。基于综合的发现算法可能运行时间较长，而且结果的质量在很大程度上取决于输入日志的质量。

从合成的角度来看，基于区域的技术总是假定观察到的行为足够完整且正确。也就是说，输入行为被隐式地视为规约

而非含有噪声或不完整的样本。如果日志中缺少相关行为，或者因噪声而包含不期望的行为，基于区域的合成往往会过度泛化，或无法生成有意义的模型。这一观察也适用于我们的贡献。当观察到的行为可被视为对预期系统行为的规约时，令牌轨迹区域尤其适用。

令牌轨迹区域为基于区域的过程发现提供了一种自然的泛化。我们不再将输入限定为从事件日志中提取的序列或偏序，而是允许将规约表示为一组标记网。这使得经典发现场景能够得到多种扩展。

首先，可以直接处理经典事件日志。事件日志中的每条轨迹都可以转换为一个简单的顺序带标签网。或者，也可以将轨迹分组并折叠为共享公共前缀、后缀或局部状态的标记网。这可以显著缩小输入规约的规模，从而缩短运行时间。

其次，令牌轨迹区域使我们能够纳入原始事件数据之外的其他行为知识。例如，可以使用标记网显式编码从时间戳、领域知识或先前分析中得出的并发关系。同样，也可以用小型标记网描述已知的冲突、循环或同步模式，并将其加入规约。这对于事件日志不完整、但存在部分模型或约束的过程发现情境尤其有用。

第三，令牌轨迹区域支持组合异质的行为信息来源。规约可以包含从事件日志中提取的轨迹、通过并发分析获得的偏序、从遗留系统推导出的基于状态的片段，甚至现有的过程模型。所有这些产物都可以统一表示为标记网，并由综合算法共同处理。生成的佩特里网能够模拟所有给定的输入，因此可以将不同视角整合到单一模型中。

从计算角度来看，基于整数线性规划的令牌轨迹区域计算与现有的发现算法十分相似。通过参数 k限制区域的局部标识，可获得有限且可控的搜索空间。与经典的基于区域的发现方法一样，增大 k会减少额外行为，而减小 k 则会得到更一般化的模型。这提供了一种自然的机制，可在精确度与泛化之间取得平衡。

总之，令牌轨迹区域以一种保守而有力的方式扩展了基于区域的过程发现。它们涵盖了经典的基于语言的发现技术，同时支持超越事件日志的、更丰富的规约。因此，令牌轨迹区域尤其适用于这样的发现场景：观察到的行为由结构知识、部分模型或人工指定的行为片段加以补充。

在过程发现中，令牌轨迹区域的另一项优势是能够消除所发现过程模型中的标签拆分。若干发现算法（例如 Alpha++[48]和 Split Miner [49], ）明确允许标签拆分，以表示活动的不同流程情境。虽然标签拆分能够显著提高所发现模型的精确度和结构清晰度，但代价是复杂性增加。具体而言，一致性检查、性能分析或进一步合成等分析技术应用于标记网时，运行时间往往会更长。原因在于，过程模型中的路径与触发序列之间不再一一对应。因此，在这类应用中，许多

分析技术必须基于过程模型的可达图运行。

借助令牌轨迹区域，标签拆分可以被视为一种建模上的便利手段，而不是最终模型的固有特征。通过标记网发现的行为本身可以视为一种规约。令牌轨迹综合确保，标签拆分引入的行为差异在语义上通过库所和弧权重得以保留，而非通过重复标签来体现。如此一来，令牌轨迹区域使我们能够在发现过程中将标记网用作表达能力强的中间表示，同时仍能将紧凑且易于分析的 Petri 网模型作为综合结果。

本文所述方法的另一个有趣特点是，令牌轨迹区域通过整数线性规划计算，并显式枚举一组最小区域。这使我们能够控制添加到综合结果中的每个库所。例如，在流程发现中，我们通常希望综合出工作流网。工作流网是一种具有特定初始库所 i 和终止库所 o 的佩特里网，且网中的每个节点都位于从 i 到 o 的某条有向路径上。对于工作流网，健全性 [50] 是另一项理想属性。如果从初始标识出发，总能最终到达最终标识，那么该工作流网就是健全的。粗略地说，健全的工作流网从库所 i 中的一个令牌开始；当执行到达库所 o 中的一个令牌时，即表示执行完成。因此，在每次执行开始和结束时，工作流网的所有内部库所都应为空。

为了使令牌轨迹综合更适用于发现健全的工作流网，我们首先将规约限制为标注工作流网。例如，将事件日志转换为一组顺序排列的标注网时，每个这样的轨迹网都已经具有唯一的初始库所和唯一的终止库所。同样，我们可以要求规约中包含的每个偏序或标记网都恰好具有一个初始库所和一个终止库所。在为预期工作流网综合结果中的内部库所计算区域时，我们要求某个区域的令牌轨迹不会标记规约中任何标记网的终止库所。这样，对于每个输入网，我们都会得到一个简单的附加不等式，从而确保每新增一个区域，对应的库所在每次指定运行之后都为空。因此，执行任何指定运行都不会在综合结果中留下令牌。这类库所不会妨碍正确终止，因此也不会导致生成的网不健全。

需要注意的是，这项限制并不能保证综合得到的网是健全的，因为它无法确保结果连通或不存在死锁。不过，它提供了一种有效机制，可引导综合过程趋向预期的网类。通过排除那些在观察到的行为结束时可能仍有令牌的库所，生成的模型在泛化规约的同时，也能更接近健全的工作流网。

作为示例，该工具提供了一个额外选项，用于启用此整数线性规划扩展。将模式从合成切换为发现后，相应的结束时为空约束便会启用。将此设置应用于示例 *1*，即可得到图1所示的佩特里网，但不包含库所 p_3和 p_5。规约中的所有四个标记网都是工作流网。因此，我们可以轻松添加一项约束，要求每个输入网的终止库所在每个计算出的区域中均未标记。库所p_3 用于计数循环迭代，因此在图 5 所示的前三次运行后并非空置。只有在变迁停止 发生后，库所 p_5 才为空。此示例表明，令牌轨迹区域可用于引导过程发现，使其生成具有更强终止性质的工作流网。

通过这种技术，我们可以得到一个能够模拟指定行为、但并未对其施加最大限制的佩特里网。在给定示例中，生成的网不再像不受限综合结果那样精确地表示观察到的行为。然而，如果输入仅包含完整行为的一个片段，那么这种方法就能得到一个有效且可靠的解，完整表示观察到的行为。这体现了行为精确度与结构保证之间的根本性权衡。根据具体应用场景，令牌轨迹区域使我们能够系统地引导综合过程。通过在整数线性规划中强化或放宽约束，我们可以在行为保持综合与更具泛化性的发现之间进行调整，从而使结果符合所需的网类和分析目标。

7. 结论

本文提出了令牌轨迹区域，作为一种新的区域定义，用于从以标记佩特里网形式给出的行为规约中合成佩特里网。令牌轨迹区域基于令牌轨迹语义，将区域定义为一组标记网中遵循标签的局部状态分布。这使我们能够合成佩特里网，在模拟所规约行为的同时，尽可能少地引入额外行为。

本研究的一项核心贡献是统一了基于状态的区域理论与基于语言的区域理论。我们从形式上证明，令牌轨迹区域涵盖了经典的基于状态的区域和基于语言的区域，因此构成一种元区域理论。由此，以状态图、语言、偏序或它们的组合形式给出的行为规约，都可以采用统一的方式处理。此外，令牌轨迹区域进一步拓展了这种统一，使标记佩特里网也能作为一等规范产物。这使建模者能够在单个规约中显式地对冲突、并发以及局部状态的合并进行建模，而无须事先选定某一种特定的行为表示。

从建模的角度来看，标记网提供了一种自然且直观的行为描述方式。建模人员可以将复杂行为分解为更小的片段，在必要时通过标签拆分展开行为，并根据可用信息的性质结合不同的建模风格。与分支过程或事件结构等表达能力更强的行为形式化方法相比，标记网并未引入新的建模语言，而是直接建立在成熟的佩特里网概念之上。令牌轨迹区域在保留这种建模便利性的同时，也提供了严谨的综合语义。

在合成方面，我们证明了每个令牌轨迹区域都定义了综合结果中的一个有效库所，并且无需添加区域所诱导的附加库所以外的其他库所。这一结果泛化了经典的基于区域的综合，并确保综合总是可以归结为区域的计算。尽管所有区域的集合通常是无限的，我们证明，将局部标识的大小限制在上界 k 以内，就能得到有限且在实践中易于处理的近似。增大 k 会单调地减少附加行为，从而提供了一种有原则的方法，在精确度与泛化之间取得平衡。

我们提出了一种基于整数线性规划的方法，用于在给定上界内计算最小令牌轨迹区域，并将该方法实现为一个基于网络的工具。实验结果表明，尽管基于区域的综合本身具有复杂性，该方法在实践中仍然可行，并且

在多种示例上表现良好。具体而言，令牌轨迹区域使我们能够将异构规约（包括基于状态的输入、基于语言的输入、工作流网以及完整的过程模型）整合到单个综合任务中。

我们进一步证明，令牌轨迹区域天然适用于过程发现场景。将观察到的行为视为规约，而非含噪样本时，只要信息充足，令牌轨迹综合便能提供精确的发现结果。同时，整数线性规划表述允许将综合过程引导至特定网类，例如具有终止保证的工作流网。因此，令牌轨迹区域非常适用于需要谨慎权衡精确度、健全性与可分析性的发现情境。

未来的工作将聚焦于若干方向。在理论方面，我们计划进一步研究综合结果的形式化保证，尤其是在附加约束下的健全性与行为最小性。在算法方面，我们希望探索整数线性规划表述的优化技术，以及处理更大规模规约的增量方法。从应用角度来看，我们打算研究将令牌轨迹区域与流程挖掘中的噪声处理和抽象技术相结合，以放宽对规约完整且正确的假设。总而言之，令牌轨迹区域为佩特里网综合提供了灵活且统一的基础，连接理论与实践，并为并发系统的建模、综合与发现开辟了新途径。

参考文献

[1] van der Aalst WMP，van Dongen BF。从事件日志发现佩特里网。《佩特里网及其他并发模型汇刊》，2013 年。**7**：372–422。doi:10.1007/978-3-642-38143-0 10。[2] Desel J，Juh´as G。“什么是佩特里网？”。载于：《统一佩特里网》，《计算机科学讲义》第 2128 卷，第 1–25 页。Springer，2001 年。[3] Peterson JL。《佩特里网理论与系统建模》。普伦蒂斯·霍尔，1981 年。[4] Reisig W。《理解佩特里网》。Springer，2013 年。doi:10.1007/978-3-642-33278-4。[5] Mayr HC，Kop C，Esberger D。业务流程建模与需求建模。载于：第一届数字社会国际会议（ICDS 2007），2007 年 1 月 2–6 日，瓜德罗普，法属加勒比地区。IEEE计算机学会，2007 年，第 8 页。doi:10.1109/ICDS.2007.9。URL https://doi.org/10.1109/ ICDS.2007.9。[6] Badouel E，Bernardinello L，Darondeau P。《佩特里网综合》。Springer，2015 年。doi:10.1007/978-3-662-47967-4。[7] Ehrenfeucht A，Rozenberg G。部分（集合）二结构。第一部分。《信息学学报》，1990 年。**27**：315–342。doi:10.1007/BF00264611。[8] Ehrenfeucht A，Rozenberg G。部分（集合）二结构。第二部分。《信息学学报》，1990 年。**27**：343–368。doi:10.1007/BF00264612。[9] Darondeau P。从形式语言推导无界佩特里网。载于：CONCUR’98，《计算机科学讲义》第 1466 卷。Springer，1998 年，第 533–548 页。doi:10.1007/BFb0055646。[10] Lorenz R，Mauser S，Juh´as G。如何从语言综合网——综述。载于：2007 年冬季仿真会议论文集。IEEE，2007 年，第 637–647 页。doi:10.1109/WSC.2007.4419657。

[11] Bergenthum R，Desel J，Lorenz R，Mauser S。从有限部分语言综合佩特里网。*FundamentaInformaticae*，2008 年。**88**（4）：437–468。[12] Bergenthum R。从哈塞图综合佩特里网。载于：BPM 2017，《计算机科学讲义》第 10445 卷。Springer，2017 年，第 22–39 页。[13] Best E，Schlachter U。佩特里网与转移系统分析。载于：第八届交互与并发经验会议，ICE 2015，《EPTCS》第 189 卷。2015 年，第 1–5 页。doi:10.4204/EPTCS.189.6。[14] Carmona J，Cortadella J，Kishinevsky M。Genet：用于佩特里网综合与挖掘的工具。载于：第九届并发在系统设计中的应用国际会议，ACSD 2009，德国奥格斯堡，2009 年 6 月 23–26 日。IEEE计算机学会，2009 年，第 181–185 页。doi:10.1109/ACSD.2009.6。[15] van der Werf JMEM，van Dongen BF，Hurkens CAJ，Serebrenik A。使用整数线性规划进行过程发现。载于：PETRI NETS 2008 论文集，《计算机科学讲义》第 5062 卷。Springer，2008 年，第 368–387 页。[16] Bergenthum R。Prime Miner——使用素事件结构进行过程发现。载于：ICPM 2019。IEEE，2019 年，第 41–48 页。[17] Bergenthum R，Desel J，Lorenz R，Mauser S。使用 VipTool 从场景综合佩特里网。载于：《佩特里网的应用与理论》，《计算机科学讲义》第 5062 卷。Springer，2008 年，第 388–398 页。doi:10.1007/978-3-540-68746-7 25。[18] Bergenthum R，Kov´aˇr J。初探佩特里网区域。载于：ATAED 2022，《CEUR研讨会论文集》第 3167 卷。2022 年。[19] Pietkiewicz-Koutny M。从步变迁系统综合带抑制弧的基本网系统。*FundamentaInformaticae*，2002 年。**50**（2）：175–203。[20] Lorenz R，Mauser S，Bergenthum R。根据场景综合抑制网的区域理论。载于：佩特里网及其他并发模型——ICATPN 2007，《计算机科学讲义》第 4546 卷。Springer，2007 年，第 342–361 页。doi:10.1007/978-3-540-73094-1 21。[21] Pietkiewicz-Koutny M，Koutny M。以区间序语义综合基本网系统。载于：ATAED & PN4TT 2023，《CEUR研讨会论文集》第 3424 卷。2023 年。[22] Koutny M，Pietkiewicz-Koutny M。以区间序语义综合 ENI系统。载于：PNSE 2024 论文集，《CEUR研讨会论文集》第 3730 卷。2024 年。[23] Bergenthum R，Desel J，Mauser S。从部分语言综合佩特里网的不同算法比较。载于：《佩特里网及其他并发模型汇刊 III》，_《计算机科学讲义》第 5800 卷，第 216–243 页。Springer，2009 年。doi:10.1007/978-3-642-04856-2 9。[24] Milner R。程序间仿真的代数定义。载于：Cooper DC（编），第二届国际人工智能联合会议论文集。英国伦敦，1971 年 9 月 1–3 日。William Kaufmann，1971 年，第 481–489 页。[25] Park D MR。无穷序列上的并发与自动机。载于：Deussen P（编），理论计算机科学，第五届 GI 会议，德国卡尔斯鲁厄，1981 年 3 月 23–25 日，论文集，《计算机科学讲义》第 104 卷。Springer，1981 年，第 167–183 页。doi:10.1007/BFB0017309。[26] Joyal A，Nielsen M，Winskel G。由开映射得到的互模拟。*Inf. Comput.*，1996 年。**127**（2）：164–185。doi:10.1006/INCO.1996.0057。[27] Vogler W。佩特里网的模块化构造与偏序语义。Springer，1992 年。

[28] Juh´as G, Lorenz R, Desel J. 我能在你的网中执行我的场景吗？载于：ICATPN 2005，《计算机科学讲义》第3536卷。Springer，2005，第289–308页。[29] Bergenthum R, Folz-Weinstein S, Kov´aˇr J. 令牌轨迹语义——使用标记佩特里网对佩特里网的行为进行建模。载于：PETRI NETS 2023，《计算机科学讲义》第13929卷。Springer，2023，第286–306页。[30] Kov´aˇr J, Bergenthum R. 令牌轨迹语义 II——佩特里网及其网语言。载于：PETRI NETS 2024，《计算机科学讲义》。Springer，2024。[31] Bergenthum R, Kov´aˇr J. 从标记 Petri 网合成 Petri 网。载于：PETRI NETS 2025，《计算机科学讲义》。Springer，2025。[32] Grabowski J. 关于部分语言。《Fundamenta Informaticae》，1981。*Fundamenta Informaticae*，1981。**4**(1):427–498。[33] Goltz U, Reisig W. 佩特里网的非顺序行为。《信息与控制》，1983。**57**(2/3):125–147。[34] Bergenthum R, Lorenz R. 使用紧凑令牌流验证佩特里网中的场景。《Fundamenta I nformaticae》，2015，第117–142页。[35] Kiehn A. 佩特里网同步与非同步行为之间的相互关系。《信息处理与控制论杂志》，1988，第3–18页。[36] Bergenthum R. 佩特里网中的发射偏序。载于：PETRI NETS 2021，《计算机科学讲义》第12737卷。Springer，2021，第399–419页。[37] Winskel G. 事件结构。载于：《佩特里网：应用及其与其他并发模型的关系》，《计算机科学讲义》第255卷。Springer，1986，第325–392页。[38] Fahland D. Oclets——基于佩特里网的场景建模。载于：《佩特里网的应用与理论》，《计算机科学讲义》第5606卷。Springer，2009，第223–242页。doi:10.1007/ 978-3-642-02424-5 14。[39] Pinna GM, Fabre E. 扩展网：一种统一的展开方法。《逻辑与代数编程方法杂志》，2020。**111**:100519。doi:10.1016/j.jlamp.2019.100519。[40] Heinrichmeyer K. 通过持续组合最小区域进行过程发现。硕士论文，FernUniversit¨at in Hagen，2020。[41] Makhorin A. GLPK——GNU线性规划工具包。https://www.gnu.org/software/glpk/，2000–2012。[42] Vaillant J, Rasulkhani S, Regue R, Jones A. glpk.js。https://www.npmjs.com/package/glpk.js，2014–2023。[43] van der Aalst WMP. 流程挖掘——《行动中的数据科学》。Springer，2016。doi:10.1007/ 978-3-662-49851-4。[44] van der Aalst WMP, Carmona J（编）。《流程挖掘手册》，《商业信息处理讲义》第448卷。Springer，2022。doi:10.1007/978-3-031-08848-3。[45] Carmona J, Cortadella J, Kishinevsky M. 一种从事件日志发现佩特里网的基于区域算法。载于：BPM 2008，《计算机科学讲义》第5240卷。Springer，2008，第358–373页。

[46] Folz-Weinstein S, Bergenthum R, Desel J, Kov´aˇr J. ILP^2 Miner——使用整数线性规划从部分有序事件日志中发现流程。载于：PETRI NETS 2023，《计算机科学讲义》第13929卷。Springer，2023，第59–76页。[47] Folz-Weinstein S, Rennert C, Mannel LL, Bergenthum R, van der Aalst WMP. eST^2 Miner——基于发射偏序的过程发现。载于：CAiSE 2025，论文集，第二部分，《计算机科学讲义》第15702卷。Springer，2025，第59–75页。[48] Wen L, Wang J, van der Aalst WMP, Wang J, Sun J. 挖掘包含非自由选择结构的过程模型。《数据挖掘与知识发现》，2007。**15**(2):145–180。[49] Augusto A, Conforti R, Dumas M, Rosa ML. Split Miner：从事件日志中发现准确且简洁的业务流程模型。《IEEE Trans. Knowl. Data Eng.》，2020。**32**(1):57–73。[50] van der Aalst WMP, Verbeek HMW, van Hee KM, ter Hofstede AHM, Sidorova N, Voorhoeve M, Wynn MT. 工作流网的健全性：分类、可判定性与分析。《Formal Aspects of Computing》，2011。**23**(3):333–363。