深度规格挖掘

Tien-Duy B. Le

信息系统学院

新加坡管理大学，新加坡

摘要

正式规范是必要的，但通常在软件系统中不可用。此外，编写这些规范成本很高，需要开发人员的技能。最近，许多自动化技术被提出来挖掘各种格式的规范，包括有限状态自动机(FSA)。然而，规范挖掘还需要更多的工作来进一步提高推断规范的准确性。

在这项工作中，我们提出了深度规范挖掘器(DSM)，这是一种为挖掘基于 FSA 的规范执行深度学习的新方法。我们提出的方法使用测试用例生成来生成更丰富的执行轨迹集，用于训练基于递归神经网络的语言模型。从这些执行轨迹中，我们构造了一个前缀树接受器(PTA)，并使用所学习的 RNNLM 来提取许多特征。这些特征随后被聚类算法用来合并 PTA 中相似的自动机状态，以构建多个 FSa。然后，我们的方法执行一个模型选择启发式算法来估计金融服务协议的金融服务度量，并返回具有最高估计金融服务度量的金融服务协议。我们执行需求侧管理来挖掘 11 个目标库类的规范。我们的实证分析表明，需求侧管理的平均离差为 71.97%，比最佳绩效基线高出 28.22%。我们还展示了帝斯曼在沙盒安卓应用中的价值。

CCS 概念

软件及其工程→动态分析；

关键词

规范挖掘，深度学习

ACM 参考格式:

天都乐和卢大伟。2018.深度规格挖掘。在第 27 届美国计算机学会软件测试与分析国际研讨会(ISSTA'18)的代表中。美国纽约州纽约市 ACM，12 页。https://doi . org/10 . 1145/3213846 . 3213876

1 简介

由于满足客户需求的快速发展，软件应用程序和库经常在没有文档记录的情况下发布

卢大伟

信息系统学院

新加坡管理大学，新加坡大卫路

规格。即使有正式的规范，随着软件系统在短时间内快速发展，它们也可能会过时。最后，编写正式的规范需要开发人员具备必要的技能和动力，因为这是一个昂贵且耗时的过程，[23]。此外，缺乏规范会对系统的可维护性和可靠性产生负面影响。由于没有文档化的规范，开发人员可能会发现很难理解一段代码，并且由于错误的假设，软件更有可能有错误。此外，开发人员不能利用需要正式规范作为输入的最先进的错误发现和测试工具[10，37]。

最近，许多自动化方法被提出来帮助开发人员降低手工起草正式规范的成本，[6，13，24，26，33]。在这项工作中，我们着重于规范挖掘算法家族，这些算法从执行轨迹中推断基于有限状态自动机的规范。克尔卡等人，[24]和许多其他研究人员提出了各种金融服务分析挖掘方法，与以前的解决方案相比，这些方法提高了推断的金融服务分析模型的质量。然而，挖掘出的规范的质量还不完善，需要做更多的工作来使规范挖掘更好。事实上，基于 FSA 的规范挖掘者仍然面临许多问题。例如，如果输入执行迹线中的方法经常以特定顺序出现，或者输入迹线的数量太小，则由 k-tails [7]和许多其他算法推断的 FSA 可能会返回未被一般化和过度写入输入执行迹线的 FSA。

为了挖掘更精确的 FSA 模型，我们提出了一种新的对执行轨迹进行深度学习的特定挖掘算法。我们将我们的方法命名为 DSM，它代表深度规格矿工。我们的方法将目标库类 C 作为输入，并使用自动化测试用例生成工具生成数千个测试用例。这个测试用例生成过程的目标是捕获一组更丰富的调用方法的有效序列。接下来，我们对生成的测试用例的执行轨迹进行深度学习，以训练递归神经网络语言模型(RNNLM)[39]。在这一步之后，我们从执行轨迹中构造一个前缀树接受者(PTA)，并利用所学的语言模型从 PTA 的节点中提取一些有趣的特征。然后，这些特征被输入到聚类算法中，用于合并相似的状态(即 PTA 的节点)。聚类算法应用程序的输出是反映训练执行轨迹的更简单和更一般化的 FSA。最后，我们的方法预测了构造的模糊聚类算法(由考虑不同设置的不同聚类算法生成)的准确性，并输出了具有最高预测值的模糊聚类算法。

我们评估了我们为 11 个目标库类提出的方法，这些目标库类以前用于评估许多以前的工作[24，26]。对于每个输入类，我们首先运行 Randoop 来生成数千个测试用例。然后，我们使用通过运行生成的执行跟踪

这些测试用例来推断 FSAs。实验表明，需求侧管理的平均离差为 71.97%。与其他现有的规范挖掘算法相比，我们的方法优于所有根据执行轨迹构建 FSAs 的基线(例如，k-tails [7、SEKT [24、TEMI [24 等)。)至少 28.22%。一些基线首先使用 Daikon 来学习不变量，然后用来推断更好的 FSA。我们的方法在 FSAs 的推理中没有使用 Daikon 不变量。排除使用 Daikon 不变量的基线，我们的方法在平均 F 度量方面可以比剩余的表现最好的挖掘器高出 33.24%。

此外，我们还评估了 DSM 挖掘的 FSA 在检测安卓应用中恶意行为的适用性。我们提出了一种利用 DSM 挖掘算法输出的 FSA 作为行为模型来构建安卓沙箱的技术。我们的技术输出了一个全面的沙箱，它考虑了敏感的应用程序接口方法的执行上下文，以更好地保护应用程序用户。我们的对比评估发现，

我们的技术可以将最先进的沙箱挖掘方法 Boxsate[21 的真阳性率提高 15.69%，而仅将假阳性率提高 4.52%。用性能最佳的适用基线替换需求侧管理，可以获得相似的真实阳性率(如需求侧管理)，但假阳性率明显更低(即假阳性率增加近 10%)。结果表明，利用帝斯曼挖掘的金融服务协议来创建更有效的安卓沙盒是有希望的。

我们工作的贡献如下:(1)我们提出了深度规格挖掘，一种新的规格

利用测试用例生成、深度学习、聚类和模型选择策略来推断基于 FSA 的规范的操作挖掘算法。据我们所知，我们是第一个将深度学习用于采矿规范的人。

(2)我们评估了需求侧管理在 11 个不同目标图书馆类别中的有效性。我们的结果表明，我们的方法在平均离差度量方面远远优于最佳基线。

(3)我们提出了一种技术，该技术使用由需求侧管理推断的有限状态机来构建一个更全面的沙箱，该沙箱考虑了敏感应用编程接口方法的执行上下文。我们的评估表明，无论是真阳性率还是假阳性率，我们提出的技术都可以大大超过几个基线。

本文的其余部分结构如下。第 2 节重点介绍背景材料。第 3 节和第 4 节介绍需求侧管理及其评估。第 5 节介绍了我们提出的技术，该技术使用 DSM 推断的 FSA 来检测安卓应用程序中的恶意行为，并对其进行评估。我们分别在第 6 节和第 7 节讨论有效性威胁和相关工作。最后，我们在第 8 节中总结并提及未来的工作。

2 背景

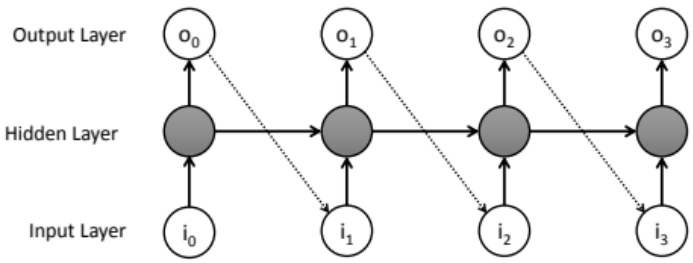
统计语言模型:统计语言模型是预言一个句子 s = w1，w2，.。。，wn 出现在一种语言中。简而言之，统计语言模型认为序列 s 是单词 w1、w2，...通过计算单词的联合概率:P(w1，.。。，wn)=μn I = 1 P(wi | w1，.。。，wi-1)。因为挑战在于

STN

新界

HMTF

<结束>



<开始>

STN

新界

HMTF

t0 t1 t2 t3 时间

图 1:一个从时间 t0 到 t3 展开的递归神经网络，用于预测下一个可能的方法，给出了调用 java.util.StringTokenizer 的方法序列。

计算条件概率 ρ(wi | w1，.。。每个不同的语言模型都有自己的假设来近似计算。N-gram 模型是一个流行的语言模型家族，其近似方式是一个单词 wk 有条件地只依赖于它之前的 N 个单词(即 wk-N+1，.。。，wk 1)。例如，单图模型简单地估计了 P(wi |w1，.。。二元模型近似为 P(wi | w1，.。。，wi-1)作为 P(wi | wi-1)等。在这项工作中，我们利用语言模型的能力来计算 P(wi | w1，.。。wi-1)用于估计自动机状态的特征。我们认为每个方法调用都是一个单词，对象的执行轨迹是一个句子(即方法调用的序列)。给定一系列先前调用的方法，我们使用语言模型输出下一个调用方法的概率。

基于递归神经网络的语言模型:最近，一个使用神经网络的语言模型家族被证明比 n-gram[38]更有效。这些模型被称为基于神经网络的语言模型(NNLM)。如果一个 NNLM 有许多隐藏层，我们称之为深层神经网络语言模型或简称深层语言模型。在这些深层语言模型中，基于递归神经网络的语言模型(RNNLM)[39]以其使用内部记忆处理任意长度单词序列的能力而闻名。RNNLM 的底层网络结构是递归神经网络(RNN)，它在其隐藏层中存储输入单词序列的信息。图 1 展示了给定序列<开始>、STN、NT、HMTF、<结束>，RNN 是如何操作的。在图中，一个 RNN 展开成四个相连的网络，每个网络一次处理一种输入法。最初，隐藏层中的所有状态都被分配为零。在时间 tk，方法 mk 由输入层表示为单热向量 ik。接下来，隐藏层通过使用矢量和先前在时间 tk-1 计算的状态来更新其状态。然后，输出层估计所有方法的概率向量 ok，以便它们出现在下一个时间步骤 tk+1 中。在随后的时间步骤中重复该过程，直到处理完序列中的最后一个方法。

3 提议的方法

图 2 显示了我们提议的方法的总体框架。在我们的框架中，有三个主要过程:测试用例生成和跟踪收集、基于递归神经网络的语言模型学习和自动机构建。我们的方法将目标类和方法签名作为输入。然后，DSM 运行 Randoop [42]，为输入目标类生成大量测试用例。然后，我们记录这些测试用例的执行，并保留输入目标类的方法调用的跟踪作为训练数据集。接下来，我们的方法对收集到的轨迹进行深度学习，以推断出一个 RNNLM，该 RNNLM 能够在给定一系列先前调用的方法的情况下预测下一个可能执行的方法。我们选择 RNNLM 而不是传统的概率语言模型，因为过去的研究表明它的优越性[39，44]。

随后，我们采用启发式方法来选择最能代表整个训练数据集的轨迹子集。根据这些轨迹，我们构建了前缀树接受者；我们将每个 PTA 的节点称为自动机状态。我们选择迹线的子集，以便在构建 PTA 时优化性能，但仍然保持推断的 FSa 的准确性。利用推断的 RNNLM，我们从自动机状态中提取多个特征，并将特征值输入考虑不同设置(例如，不同数量的聚类)的多个聚类算法(即，k-means [35]和分层聚类[43)。聚类算法的输出是相似自动机状态的聚类。我们使用这些集群通过合并属于同一集群的状态来创建新的 FSA。具有特定设置的聚类算法的每个应用都会导致不同的 FSA。我们提出了一种模型选择策略，

通过预测精度、召回率和离差来启发式地选择最准确的模型。最后，我们输出具有最高预测 f 测度的金融稳定分析。

3.1 测试用例生成和跟踪收集

这一过程对我们的方法起着重要的作用，因为它决定了由深度学习过程推断的 RNNLM 的质量。先前在规范挖掘[24，26，27]中的研究工作收集了给定单元测试用例或研究者手工创建的输入的程序执行的轨迹。在这项工作中，我们利用对挖掘规范的深入学习。深度学习需要大量丰富的数据。训练输入越多，生成的 RNNLM 可以捕获的模式就越多。一般来说，很难按照以前的工作为任意目标库类收集足够丰富的执行跟踪集。首先，寻找所有使用目标库类的项目很有挑战性，尤其是来自新的或未发布的库的类。其次，现有的单元测试用例或手动创建的输入可能无法涵盖目标类中方法的许多可能的执行场景。

我们通过遵循 Dallmeier 等人的[11，12]来解决上述问题，以便为挖掘规范生成尽可能多的测试用例，并为后续步骤收集这些测试用例的执行轨迹。最近，已经提出了许多测试用例生成工具，例如兰多普[42、埃沃套件[16 等。在最先进的测试用例生成工具中，我们选择 Randoop 是因为

https://randoop.github.io/randoop/·http://www.evosuite.org/

它用途广泛，重量轻。此外，Randoop 维护良好，并经常更新新版本。作为未来的工作，我们计划将许多其他测试用例生成方法集成到我们的方法中。

Randoop 生成大量测试用例，这与其执行的时间限制成比例。为了提高测试中可能的方法序列的覆盖率，除了缺省值之外，我们还向 Randoop 提供了特定于类的文本。例如，对于 java.net.Socket，我们创建字符串和整数升，它们是主机的地址(例如，“localhost”、“127.0.0.1”等)。)和监听端口(例如 8888 等)。)。此外，我们创建包含静态方法的 dri-ver 类，这些静态方法调用目标类的构造函数来初始化新对象。这有助于加快 Randoop 创建新对象的速度，而无需花费时间为构造函数搜索合适的输入值。

3.2 学习面向规范挖掘的 RNNLM

3.2.1 训练方法序列的构建。我们收集的执行跟踪集是一系列方法序列。每个序列都以两个特殊符号开始和结束:<开始>和<结束>。这些符号用于分隔两个不同的序列。我们收集所有序列来创建用于训练递归神经网络的数据。此外，我们将方法序列 MAX_SEQ_FREQ 的最大频率限制为 10，以防止出现不平衡的数据问题，其中一个序列比其他序列出现得更频繁。

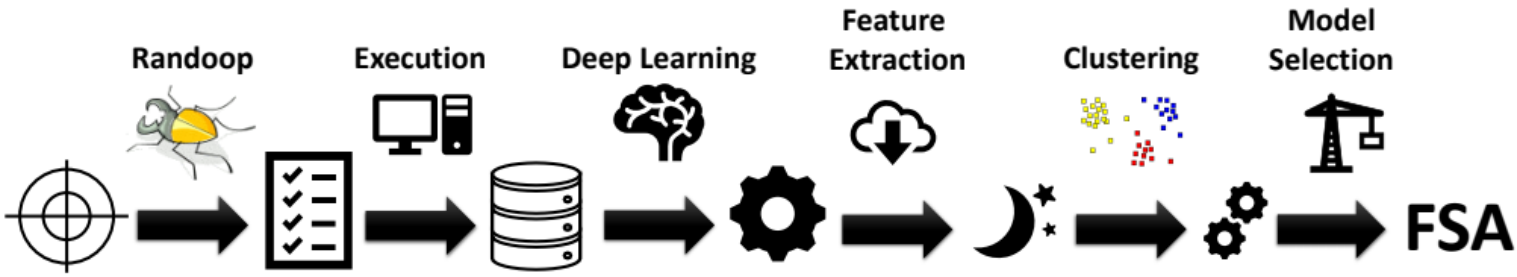
3.2.2 模型培训。我们对训练数据进行深度学习，以便为每个目标库类学习基于递归神经网络的语言模型。默认情况下，我们使用长短期内存(LSTM)网络[20 作为 RNNLM 的底层架构，这是最先进的 RNN 之一。与标准的 RNN 架构相比，LSTM 在学习长期依赖性方面更好。此外，LSTM 对于长序列是可伸缩的[44]。

3.3 自动机构造

在这个处理步骤中，我们的方法将训练执行轨迹集和推断的 RNNLM 作为输入(参见第 3.2 节)。这个步骤的输出是一个 FSA，它最好地捕获了相应目标类的规范。FSA 的构建经历了几个子步骤:跟踪采样、特征提取、聚类和模型选择。

首先，我们使用启发式方法来选择代表所有训练执行轨迹的方法序列子集。特征提取和聚类步骤使用这些选择的轨迹，而不是所有轨迹，以降低计算成本。我们从所选择的轨迹中构造一个前缀树接受器，并使用推断的神经网络模型提取每个前缀树接受器节点的特征。我们将每个 PTA 节点称为自动机状态。图

3 显示了从 java.security.Signature 的方法调用序列中构建的一个示例 PTA 的摘录。在聚类子步骤中，我们在具有不同设置的 PTA 节点上运行许多聚类算法来创建许多不同的 FSa。最后，在模型选择子步骤中，我们遵循一种启发式方法来预测构造的金融服务协议的金融服务度量(见第 4.2.1 节)，并输出具有最高值的金融服务协议



目标类和方法

测试用例

跟踪

总部在 RNN

语言模型

特征

FSA

候选人

图 2:需求侧管理的总体框架

算法 1:选择执行跟踪的子集

输入:S = {Si | 1 ≤ i ≤ N}:执行跟踪的集合，其中 N 是跟踪的数量输出:O:选定的执行跟踪 1 按跟踪长度的升序排序 S 2d ← 初始化字典类型 3 P ← 为 Si ∈ S do 空集 4

5 代表(a，b)，其中 a∈Si∈b∈Si∈a < b 做 6 维[(a，b)]+=[Si] 7 包括(a，b)到 P 8 末端 9 末端

S9

S1

<开始>

S2

<初始化> S6

更新

S7 S8

初始验证<结束>

S10 S12

初始验证<结束>

S11

初始验证

S3·S4 S5

验证<结束>

… …

图 3:前缀树接受器的例子包括从硅到氧。我们通过从磷(第 15 行)中移除来标记对(a，b)。我们一直在搜索序列，直到 0 覆盖输入执行轨迹中的所有同现对(a，b)。

表 1:自动机状态的提取特征

10 O ←空套 11 而∃(a，b←p 做

12 选择(a，b) ∈ P，使[(a，b)]的元素数量最少

13 找到 Si ∈ D[(a，b)]，使 Si 最短∈Si < o14 包括 Si 到 O，并从 P 中删除所有对(a，b)，其中

a∈Si∈b∈Si∈a < b15 端

16 返回 0

预测离差。该模型选择步骤中使用了全套轨迹。在下面的段落中，我们将描述该处理步骤中每个子步骤的细节:

跟踪采样:我们的训练数据包含大量序列。因此，使用它们来构建 FSa 是非常昂贵的。因此，跟踪采样的目标是创建一个更小的子集，它可能合理地很好地代表所有跟踪的整个集合。我们提出了一种启发式方法来寻找一个轨迹子集，该子集覆盖了所有训练轨迹中的所有共现方法对。

算法 1 展示了我们从整个执行轨迹集合中选择执行轨迹的启发式方法。首先，我们确定所有对(a，b)，其中 a 和 b 是在至少一个轨迹中一起出现在 S 中的方法，并将它们存储在 P 中(第 4 行到第 9 行)。接下来，我们创建一个包含所选轨迹的集合 0——最初 0 是一个空集合(第 10 行)。然后，我们迭代地选择一对(a，b)，它不出现在 0 中的任何轨迹中，而出现在最少数量的输入轨迹中(第 12 行)。给定选定的对(a，b)，我们寻找最短的迹线 S1 < 0，其中 a，b∈S1(第 13 行)。一旦找到踪迹，我们

如果 m1 和 m2 一起出现在至少一个轨迹中，则(m1，m2)是同现对。

特征标识值	
类型一:以前调用的方法	
调频	如果 m 在自动机状态之前被调用，则为 1。否则，0。
类型二:下一个要调用的方法	
下午	由学习的基于递归神经网络的语言模型为方法 m 计算的概率，方法 m 在达到特定的原子状态(0 ≤ Pm ≤ 1)后被调用。

特征提取:从采样的执行轨迹的方法序列中，我们构造了前缀树接受器。PTA 是一种树状确定性有限自动机，它是通过将序列的所有前缀作为状态来创建的，PTA 只接受构建它的序列。我们构建的 PTa 的最终状态是具有带<结束>标签的进入边缘的状态(见第 3.2 节)。图 3 显示了前缀树接收器(PTA)的一个例子。表 1 显示了提取特征的信息。对于 PTA 的每个状态，我们对两种类型的特征特别感兴趣:(1)类型一:这种类型的特征捕获以前的信息

在达到状态之前调用方法。状态 S 的第一类特征的值是在起始状态(即 PTA 的根)和状态 S 之间的路径上的方法的出现。例如，根据图 3，对应于节点 S3 的第一类特征的值是:F<START> = F<init> = FinitVerify =和 Fupdate = Fverify = F<END> =。

(2)第二类:这种类型的特性捕获了在达到一个状态后可能立即调用的方法。这些特征的值由

算法 2:给定一组方法序列，预测有限状态自动机的精度。

输入:M:有限状态自动机

数据:一组训练方法序列输出:序列∈数据 do 的 m1 PDat ← 空集 2 的预测精度

3 对于 0 ≤ i <长度(seq)1 d0 4 包括(seq[i，seq[i + 1])到 PDat 的 5 端 6 端

晚上 7 点←空套

8 EM 9 中 S1 M → S2∈EM∈S2 M → S3∈EM do 10 包括(m1，m2)到 PM 11 的一组转换结束

精度←| PDat a | | PDat a≈PM | 13 返回精度

深度学习步骤(参见第 3.2 节)。例如，在图 3 的节点 S3，initVerify 和< END >比之后调用的其他方法具有更高的概率。RNNLM 输出的节点 S3 的第二类特征及其值的例子如下:P <开始> = P <初始> = P 验证=化蛹日期=..

我们提取不同类型特征的直觉是为后续子步骤中的聚类算法提供足够的信息，以便更好地合并 PTA 节点。

聚类:我们运行 k-means [35]和分层聚类[43]算法对 PTA 的状态及其提取的特征。我们的目标是创建一个更简单、更一般化的自动机来满足目标库类的规范。由于 k 均值聚类和层次聚类都需要预定义的聚类数量输入，我们尝试使用从 2 到 MAX_CLUSTER 的许多值(参见第 4.2.2 节)来搜索最佳的 FSA。总的来说，聚类算法的执行会产生 2 ×(MAX _ CLUSTER 1)个 FSAs。

模型选择:我们提出了一种启发式算法来从聚类算法输出的结果中选择最佳的有限状态机。算法 2 描述了我们在给定所有轨迹数据集的情况下预测自动机精度的策略(参见第 3.1 节)。

我们通过首先构建包含所有对(m1，m2)的集 PDat a 来预测精度，其中 m1 和 m2 在数据的执行跟踪中连续出现(即 m1 正好在 m2 之前被调用)。然后，我们构造另一个包含自动机 M 生成的轨迹中连续出现的所有对(m1，m 2)的集合 PM。在算法 2 中，第 1 至 6 行计算数据中出现的所有对(m1，m2)，第 7 至 11 行收集输入自动机 M 生成的轨迹中出现的那些对。为了找到由 M 生成的轨迹中出现的所有对，我们寻找两个过渡 S1 M→S1 和 S2 M→S2，其中 S1 = S2。我们获取两个转换(即 m1 和 m2)的标签，并将一对方法(m1，m2)添加到集合 PM 中。第 12 行计算精度的预测值，精度是指在大约下午 1 点时，所有粒子对的数量与所有粒子对的数量之比。我们输入

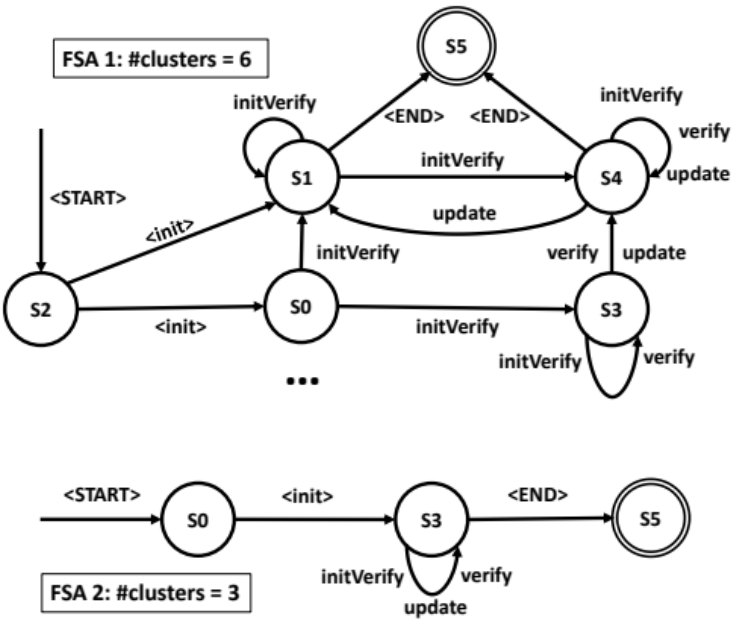


图 4:通过图 3 所示的 PTA 聚类算法输出的示例 FSAs。

由聚类算法创建的所有 FSA 和算法 2 的所有执行跟踪，用于估计 FSA 的精度。

接下来，我们通过计算给定自动机 m 接受的所有执行轨迹的百分比来近似召回的值。一旦预测了 FSA 模型的所有精度和召回，我们计算每个自动机的 F 度量的期望值(即精度和召回的调和平均值)。最后，我们的方法返回具有最高期望离差的金融稳定分析。

示例:从图 3 中部分显示的 PTA 中，我们对 PTA 的每个状态进行特征提取。然后，我们将这些具有提取特征的状态输入聚类算法(即 k 均值和分层聚类)以合并相似的状态。如果 MAX_CLUSTER 被识别为总共 20 个，那么将有 38 个由两个聚类算法构建的 FSA。图 4 显示了 k 均值输出的 19 个 FSa 中的 2 个。接下来，我们对这些金融服务协议进行模型选择，以选择具有最高预测金融服务措施的金融服务协议。

4 实证评估

4.1 数据集

4.1.1 目标库类。在我们的实验中，我们选择了 11 个目标库类作为基准来评估我们提出的方法的有效性。这些库类由先前在规范挖掘[24，26]中的研究工作调查。表 2 显示了所选库类的更多细节，包括收集的执行跟踪信息。在这些库类中，11 个中有 9 个来自 Java 开发工具包(JDK)；另外两个库类是 DataStructure。StackAr(来自 Daikon 项目)和 NumberFormatStringTokenizer(来自 Apache Xalan)。对于每一个图书馆类，我们都考虑由克尔卡等人[24]分析的方法。

4.1.2 基本事实模型。我们利用克尔卡等人[24]创建的地面真相模型。在调查的库类中，我们细化了五个基于 Java 集合的库类(即数组列表、链接列表、哈希表、哈希表和哈希表)的基本事实模型，以捕获“空”和“非空”

表 2:目标库类。“#M”代表被分析的类方法的数量，“生成的测试用例”是由 Ran-doop 生成的测试用例的数量，“记录的方法调用”是执行跟踪中记录的方法调用的数量，“NFST”代表 NumberFormatStringTokenizer。

目标库# M #生成#记录类测试用例方法调用数组列表 18 42，865 22，996 哈希表 11 53，396 67，942 哈希表 8 79，403 89，811 哈希表 8 23，181 257，428 链接列表 7 13，731 4，847 NFST 5 15，8998 95，149 签名 5 79，096 205，386 套接字

集合对象的状态。我们还修改了 NumberFormatStringTokenizer 和 Socket 的基本事实模型，包括原始模型的缺失转换。

4.2 实验设置

4.2.1 评估指标。我们遵循劳和霍的方法[32]来测量精确度和召回率，以评估我们建议的方法的有效性。洛和霍的方法已被[24，27]的许多先前规范采矿工程广泛采用。他们提出的方法将一个基本事实和一个推断的金融稳定分析作为输入。接下来，它从两个 FSa 生成句子(即轨迹)，以计算它们的相似性。推断的模糊推理的精确度是其相应的基本事实模型所接受的句子在该模糊推理生成的句子中所占的百分比。回忆一个推断的 FSA 是在由相应的基本事实模型产生的句子中，它自己接受的句子的百分比。简而言之，精确度反映了推断模型产生的正确感觉的百分比，而回忆反映了推断模型能够产生的正确句子的百分比。我们使用精确度和召回率的调和平均值——F-measure 作为评估规范挖掘算法的总结度量。f-测量定义如下:

f-测量= 2 倍

精确度×召回率

精确+召回(1)

为了精确地计算精确度、回忆和离差，从一个句子生成的句子必须完全覆盖它的状态和转换。为了实现这个目标，我们将最大生成句子数设置为 10，000，最大长度为 50，每个转换的最小覆盖范围等于 20。[24，26]以前的著作也采用了类似的策略。

4.2.2 实验配置和环境。

随机配置。在测试用例生成步骤中，对于每个目标类，我们用 20 个不同的初始种子重复执行 Randoop(3 . 1 . 2 版)，时间限制为 5 分钟。我们设置了

时间限制为 5 分钟，以确保后续收集的执行跟踪不太长也不太短。我们重复执行 Randoop 20 次，以最大限度地覆盖 Randoop 生成的测试用例中可能的程序方法序列。此外，我们关闭了 Randoop 生成错误揭示测试用例的选项(即，无错误揭示测试设置为真)，因为这些测试用例的执行通常被异常或错误中断，这导致后续深度学习过程的方法序列不完整。我们发现在这个设置中，生成的跟踪覆盖了 100%的目标应用编程接口方法；此外，平均而言，每个目标类的基本真理模型中 96.97%和 98.18%的边和状态都被覆盖。

RNNLM。我们使用基于张量流的 RNNLM 的公开实现。我们在英伟达 DGX-1 深度学习系统上执行这一实施。我们使用作为实现一部分的默认配置。

群集配置。在聚类步骤中，我们运行 k-means 和分层聚类，每个目标类的聚类数设置为从 2 到 MAX_CLUSTER = 20。我们使用带有默认设置的 sci kit-learn(0.18 版)的 sklearn.cluster.KMeans 和 sklearn . cluster . aggregation 群集。

4.2.3 基线。在实验中，我们将需求侧管理的有效性与许多以前的规范挖掘工作进行了比较。克尔卡等人提出了许多分析执行轨迹的算法来推断 FSAs [24]。这些算法是 k-tails、CON-TRACTOR++、SEKT 和 TEMI。承包商++、TEMI 和塞科特利用戴康学到的不变量推断模型。另一方面，k-tails 仅根据执行跟踪中方法的顺序构建模型。尽管需求侧管理没有处理可能的不变量，但我们将 CONTRACTOR++、SEKT 和 TEMI 作为基线，比较深度学习和可能的不变量推理在规范挖掘中的适用性。对于 k-tails 和 SEKT，我们选择 k∑{ 1，2}跟随 Krka 等人的[24]和 Le 等人的[26 的配置。总的来说，我们有六种不同的基线:传统的 1 尾、传统的 2 尾、CONTRACTOR++、SEKT 1 尾。SEKT 2-tails 和 TEMI

我们使用 Krka 等人提供的实现[[24]。我们利用 Daikon [14]从 Randoop 生成的测试用例中收集执行跟踪，并从所有跟踪中推断出可能的不变量。最初，Krka 等人的实现使用 32 位版本的 ices 1.0 Java Lite 应用编程接口，该接口仅适用于 32 位的 Java 虚拟机，并且最多只能使用 4 GB 的堆内存。由于执行跟踪的数量很大，我们遵循两个实验方案来运行 Krka 等人的代码:

(1)方案一:Krka 等人的实现被更新以与 ices 1 SMT 求解器的 64 位库一起工作。然后，我们将所有生成的测试用例的执行轨迹以及由这些轨迹推断的 Daikon 不变量输入到所有基线。对于 Krka 等人代码的每个应用，我们将最大分配内存设置为 7 GB，时间限制为 12 小时。

https://github.com/hunkim/word-rnn-tensorflow·http://softarch.usc.edu/wiki/doku.php?id=inference:start

111

表 3:需求侧管理的有效性。“福”是福的量度。F 类(%)F 类(%)数组列表 22.21 签名 100.00 哈希表 86.71 套接字 54.24 哈希表 76.84 堆栈卡 74.38 哈希表 79.92 字符串化器 100.00 链接列表 30.98 输出流 88.82 NFST 77.52%平均 71.97

(2)方案二:我们使用原始的 Krka 等人的实现和一组对应于兰多普用一个特定种子生成的测试用例的执行轨迹。

4.3 研究问题

问题 1:需求侧管理有多有效？在这个研究问题中，我们计算了用我们的方法为 11 个目标库类推断的 FSA 的精度、召回率和 f 测度。

RQ2:需求侧管理与现有的规格最小化算法相比如何？在这个研究问题中，我们在各种实验方案中将需求侧管理与 Krka 等人提出的许多现有规范挖掘算法进行了比较。RQ3:哪种递归神经网络(RNN)最适合需求侧管理？默认情况下，DSM 使用来自执行跟踪的长短期内存(LSTM)网络来训练 RNNLMs。在这个研究问题中，我们首先调整需求侧管理以使用标准的 RNN 和门控循环单元(GRU) [9]网络来构建具有与 LSTM 相同学习配置的语言模型(见第 3.2 节)。然后，我们分析了需求侧管理对于每个神经网络体系结构(即标准 RNN、LSTM 和 GRU)的有效性。

4.4 调查结果

RQ1:需求侧管理的有效性。表 3 显示了 11 个目标库类的需求侧管理的 F 值(第 4.1 节)。从表中可以看出，我们的方法达到了 71.97%的平均离差。值得注意的是，对于字符串化和签名，需求侧管理推断出与基本事实模型完全匹配的模型(即 100%的离差度量)。在 11 个库类中，还有 6 个我们的方法达到了 70%或更高的 F 值。

通过仔细研究需求侧管理的中间结果，我们发现在 11 个类中，需求侧管理执行 26 - 1，072 个合并操作(平均 303.45 个操作)来从 PTa 构建最终的 FSa。此外，在聚类步骤中产生的金融服务协议的预测金融服务措施的差异在 22.05% - 68.32%之间(平均为 39.13%)。这些表明，需求侧管理部门需要做大量的工作才能使家长教师协会达到最终的家庭服务协议。

RQ2:需求侧管理与前期工作。

方案一:表 4 的第一部分强调了 Krka 等人根据方案一提出的 6 个基线的 F 度量。在这个实验方案中，我们使用从生成的测试用例中收集的所有执行轨迹和从这些轨迹推断的可能不变量作为基线的输入数据。根据该图，承包商++是该实验方案中平均测得的最有效基线(即 55.22%)；传统的单尾是性能最好的基线，它只使用执行轨迹来构建 FSA(即平均 F 度量为 33.42%)。设计标准手册

在平均预测、召回和离差度量方面比所有基线都更有效。就平均离差度量而言，我们的表现分别优于承包商++(即最佳绩效基线)和传统单尾(即仅使用执行轨迹的最佳基线)30.33%和 115.35%。此外，我们注意到许多基线没有精度、召回率和离差测量的可用结果。这是因为这些基线无法返回 FSa，主要是因为堆内存不足或时间限制超出了错误。

方案二。表 4 的第二部分显示了克尔卡等人[[24]根据方案二提出的 6 个基线的氟测量。从平均离差来看，最佳 TEMI 是表现最好的基线；传统的单尾是从执行轨迹构建模型的最佳基线，平均离差为 53.97%。与需求侧管理相比，我们注意到我们方法的平均召回率和离差度量高于所有六个基线。这表明我们推断的模型更一般化，也更不过度。就平均离差度量而言，我们的方法比最佳基线(即乐观 TEMI)高出

28.22%。需求侧管理也比传统的单尾(即仅从跟踪中的方法顺序推断 FSa 的最佳基线)更有效 31.57%。值得注意的是，与从执行轨迹构造自动机的基线(即传统的 1 尾和传统的 2 尾)相比，DSM 对所有目标库类的 F 度量都更高。对于使用可能不变量的其他基线，我们的方法对于除数组列表、链接列表、哈希表和套接字之外的 11 个目标库类中的 7 个更有效。RQ3:最佳 RNN 建筑。表 5 显示了用三种不同的 RNN 架构配置的需求侧管理的有效性。我们可以注意到，DSMLSTM(我们的默认配置)和 DSMGRU 在 F 度量方面的表现分别比 DSMRNN 好 7.92%和 6.88%。帝斯曼集团的表现几乎和帝斯曼一样。

5 用于检测 ANDROID 恶意行为的挖掘 FSA

如今，安卓是最受欢迎的移动平台，拥有数百万个应用和支持的设备。事实上，机器人用户很容易成为攻击者的目标。最近，已经提出了几种方法来保护用户免受恶意软件的潜在威胁。在最先进的方法中，Jamrozik 等人提出 Boxmate，通过探索目标良性应用的行为来挖掘构建安卓沙盒的规则，[21]。Boxmate 的关键思想是它阻止一个程序改变它的行为；它可以防止隐藏的攻击、后门和利用漏洞危及安卓应用的安全性。Boxmate 分两个阶段工作:监控和部署。在监控阶段，Boxmate 使用一个名为 Droidmate [22 的测试用例生成工具来创建一组丰富的图形用户界面测试用例。在这些测试用例的执行过程中，Boxmate 记录敏感的应用编程接口方法的调用(例如，访问摄像机、位置等的方法)。)，并使用它们创建沙箱规则。这些规则指定了在部署期间允许调用哪些敏感的应用编程接口方法。在部署期间，当应用程序访问上述规则中未记录的敏感应用编程接口方法时，沙箱会立即接受该操作，并向用户发出关于可疑活动的警告消息。

表 4:离差(%):方案一与方案二。“T1”是传统的 1-tails，“T2”是传统的 2-tails，“C+”是 CONTRACTOR++，“S1”是 SEKT 1-tails，“S2”是 SEKT 2-tails，“OT”是乐观的 TEMI，“NFST”是数字格式字符串化者，“-”表示结果不可用。

目标方案一方案二

图书馆等级 T1 T2·C+S1·S2·OT T1 T2·C+S1·S2·OT

数组列表 13.96 13.13 36.03 13.86 13.07 16.87 5.61 3.08 36.03 4.34 3.08 35.82

哈希表 25.41 8.71 68.94 37.35 21.93 68.94 37.35 21.93 68.94

HashSet 20.88 21.27 52.22 20.88 21.27 23.34 22.96 15.35 52.22 22.96 15.35 55.62

哈希表 42.39 33.58 92.78 78.42 73.37 92.78 81.69 65.47 92.78

链接列表 27.15 25.72 86.02 26.67 24.52 7.51 26.03 8.07 86.02 18.45 6.59 86.02

NFST 24.57 25.52 30.40 24.56 25.78 11.80 68.72 51.04 30.40 66.58 49.51 33.40

签名 61.54 64.25 66.88 62.05 63.98 39.06 100 95.41 66.88 95.41 89.78 66.88

插座 35.89 31.52 55.15 34.73 28.37 37.30 21.98 55.15 35.32 20.87 55.62

stack ar 16.54 16.54 34.91 16.54 16.54 42.57 42.57 34.91 42.57 42.57

stringTokenizer 52.88 52.97 21.30 52.15 100 89.69 21.30 92.21 84.77 0.00

zip outputStream 46.36 47.42 62.80 47.91 82.51 78.08 62.08 86.47 67.84 66.20

平均 33.42 30.97 55.22 33.26 27.65 19.72 54.70 45.50 55.22 53.03 42.52 56.13

表 5:离差(%):标准 RNN、LSTM 和 GRU 的需求侧管理。

目标库 DSMLSTM DSMRNN DSMGRU 类数组列表 22.21 4.95 9.39 哈希表 86.71 97.76 100.00 哈希表 76.84 68.86 73.88 哈希表 79.92 87.94 87.94 链接列表 30.98 32.29 34.86 NFST 77.52 70.09 73.31 签名 100.00 65

Boxmate 挖掘的沙箱规则构成了一个行为模型，它接受敏感的应用编程接口方法作为输入，并预测其调用是良性的还是恶意的。然而，我们发现攻击者可以绕过 Boxmate 的模型，如果他们可以通过调用应用程序在正常操作中使用的敏感 APIs 来执行恶意活动的话。因此，我们提出了一种技术来使用 DSM 推断的 FSAs 来构建更全面的沙箱，该沙箱考虑敏感的应用编程接口调用的上下文，即在敏感的应用编程接口调用之前调用的一系列方法，可以更好地保护安卓用户免受攻击者的攻击。对于每个敏感的应用编程接口方法，我们在创建训练跟踪之前选择先前执行的非敏感的应用编程接口方法。然后，我们将所有的训练轨迹分组到一个单独的集合中，并将其输入到需求侧管理。需求侧管理的输出是一个捕捉敏感和非敏感应用编程接口方法之间相互作用的前端服务协议。然后，FSA 被用作基于自动机的行为模型，以构建沙箱来实现更有效的保护。

默认情况下，我们设置 W = 3。

应用行为模型推断

良性应用

设计标准手册

监控阶段部署阶段

敏感和相关 APIs 的执行跟踪

有限状态自动机

APPS

应用程序

可疑的恶意行为

美国药品管理局

敏感和相关 APIs 的执行跟踪

拒绝

原木

原木

图 5:利用需求侧管理推断的行为模型的恶意软件检测框架

图 5 展示了我们恶意软件检测系统的框架。根据该图，我们的框架有两个阶段:

监控阶段:该阶段接受目标安卓应用程序的良性版本作为输入。我们首先利用图形用户界面测试用例生成工具(即猴子[1 号、开膛手[2 号、彪马[18 号、[机器人 22 号和[机器人 30 号)来创建一组多样化的测试用例。接下来，我们用生成的测试用例执行输入应用程序，并监控调用的 API 方法。特别是，每次应用程序调用一个敏感的应用编程接口方法时，我们都会选择一系列之前执行的应用编程接口方法，并将它们包含到训练跟踪中。然后，利用需求侧管理对收集到的轨迹的挖掘算法，构建了一个基于有限状态机的行为模型。构建的模型反映了调用敏感的应用程序接口方法时应用程序的行为。随后，我们使用 BM 来指导

恶意软件检测部署阶段基于自动机的沙箱。

部署阶段:在这个阶段，我们的框架利用推断模型 BM 来构建基于自动机的沙箱。沙箱用于管理和控制安卓应用的执行。每当一个应用程序调用一个敏感的 API X 时，沙箱会在 X 之前选择一系列先前执行的方法，并将它们输入到行为模型 BM 中，将对 X 的调用分为恶意调用和良性调用。如果模型 BM 预先断定 X 的执行是恶意的，沙箱通过发出关于可疑活动的警告消息来通知用户。否则，沙箱允许应用程序在不通知用户的情况下继续执行。

我们使用最初由李等人[收集的 102 对安卓应用的数据集来评估我们提出的恶意软件检测框架。每对应用程序包含一个良性应用程序及其相应的恶意版本。恶意应用程序是通过将恶意代码注入相应的未打包良性应用程序而产生的[29]。所有这些应用都是发布到不同应用市场的真实应用。最近，鲍等人[4]利用上述 102 对，用 5 种不同的测试用例生成工具(即猴子[1、开膛手[2、美洲狮[18、机器人[22 和机器人[30])来评估 Boxmate 挖掘规则的有效性。在我们的评估中，我们利用了包等人[4]收集的 102 个安卓应用对的执行轨迹。我们将 w(即在调用敏感的应用编程接口方法之前选择的方法的数量)设置为 3，并使用 Boxmate、DSM 以及 k-tails (k = 1)来使用这些轨迹来推断几个行为模型。我们包括 k-tails (k = 1)，因为根据表 4，这是从 API 调用的原始跟踪中推断 FSA 的最佳基线挖掘算法。我们让需求侧管理和基于不变量的挖掘器(即承包商++和 TEMI)之间的比较用于未来的工作，因为 Daikon 目前无法为安卓应用挖掘不变量。接下来，我们使用以下评估指标评估推断行为模型在检测恶意软件方面的有效性:

真实阳性率(TPR):被正确归类为恶意应用的百分比。总生产率计算如下

T P

T P + FN (2)

假阳性率(FPR):被直接归类为恶意的应用的百分比。FPR 的计算方法如下

FPR =

FP

FP +T N (3)

在上式中，TP(真阳性)指被归类为恶意的恶意应用的数量，TN(真阴性)指被归类为良性的良性应用的数量，FP(假阳性)指被归类为恶意的良性应用的数量，FN(假阴性)指被归类为良性的恶意应用的数量。主题方案审查和 FPR 都很重要；TPR 是重要的，因为否则恶意行为是允许的，而 FPR 是重要的，因为否则用户会对错误的警告感到恼火。这两个指标是众所周知的，并且被安卓恶意软件检测的最先进方法广泛采用(例如，[3，50])。

https://github.com/baolingfeng/SANER2018Sandboxes

表 6:需求侧管理与基线。“APR”代表“方法”，“TP”代表真阳性，“FN”代表假阴性，“TN”代表真阴性，“FP”代表假阳性，“TPR”代表“真阳性率”，“FPR”代表“假阳性率”。

FPR

需求侧管理 93 9 398 112 91.18% 21.97%

k-tails 94 8 350 160 92.16% 31.37%

Boxmate 77 25 421 89 75.49% 17.45%

为了计算真阳性和假阴性，对于每对良性和恶意应用，我们使用 DSM、k-tails 和 Boxmate 来构建行为模型，使用良性应用的训练轨迹，并将这些模型部署在相应恶意应用的轨迹上。根据等式 2，我们使用真阳性和假阴性的值来估计真阳性率(TPR)。TPR 值越高，检测到的恶意活动就越多。另一方面，为了计算真阴性和假阳性，对于每一个良性的安卓应用程序，我们在五个测试用例生成工具中执行交叉验证(即，猴子[1、桂利普[2、彪马[18、机器人[22 和机器人[30])。特别是，我们使用 4 个工具的执行轨迹作为训练数据，学习 DSM、k-tails 和 Boxmate 的行为模型。然后，我们在剩余工具的踪迹上部署推断的模型，以检查模型是否将良性应用检测为恶意应用(即假阳性)。我们总共分析了良性应用和测试用例生成工具之间的 5 × 102 = 510 个组合。然后，我们利用真阴性和假阳性的值来计算假阳性率(FPR)(见等式 3)。FPR 值越小，错误警报的数量就越少。

表 6 显示了 DSM、k-tails 和 Boxmate 的结果。Boxmate 可以检测到 75.49%的恶意应用程序，而误报率为 17.43%。我们注意到，DSM 在真阳性率方面的表现优于 Boxsmate 15.69%，而在假阳性率方面仅损失 4.52%。将需求侧管理与 k 尾进行比较，我们注意到它们具有相似的真阳性率(差异小于 1%)，但后者具有显著更高的假阳性率(差异接近 10%)。显然，需求侧管理在考虑真阳性率和假阳性率的情况下达到了最佳的折衷。

6 有效性威胁

对内部有效性的威胁。我们已经仔细检查了我们的实施情况，但是有些错误我们没有注意到。我们使用的由克尔卡等人([24)创建的基本真理模型的正确性也有潜在的威胁。为了减轻这种威胁，我们将它们的模型与从 Randoop 生成的测试用例以及由库类作者(例如 Javadocs)发布的文本文档中收集的执行跟踪进行了比较。我们相应地修改了基本事实模型。

有效性的另一个威胁与 tar-get API 方法的参数值有关。我们使用包等人[4]收集的轨迹，这些轨迹排除了所有参数值。这与 Jamrozik 等人的不同

这与博克斯马特的原始论文《[21》中报道的结果是一致的。他们报告说，在考虑的 18 个用例中，出现了两个错误警报(见他们论文的表 2)-这导致了 11.11%的错误阳性率。在我们的实验中，我们借助不同的测试用例生成工具来考虑更多的用例。Jamrozik 等人没有报告真正的阳性率，因为研究中没有考虑恶意应用。

工作[21]排除大多数(但不是全部)参数值。我们决定排除所有参数值，因为本文中考虑的所有规范挖掘算法(包括需求侧管理)都会产生对参数值没有约束的 FSA。作为未来的工作，我们计划扩展需求侧管理以生成包含参数值约束的模型。

对外部有效性的威胁。这些威胁对应于我们经验发现的普遍性。在这项工作中，我们分析了 11 个不同的图书馆类。这大于用于评估许多先前研究的目标类别的数量，例如[24、33、34]。作为未来的工作，我们计划通过分析更多的库类来推断它们基于自动机的规范，从而减少这种威胁。

对结构有效性的威胁。这些威胁对应于评估指标的使用。我们遵循了劳和霍的方法，该方法使用精确度、召回率和离差度量来测量自动机输出的精确度，该自动机输出是通过规范挖掘算法根据基本事实模型[32]进行的。此外，Lo 和 Khoo 的方法是众所周知的，并且已经被规范挖掘中的许多先前的搜索工作所采用，例如，[5、6、8、13、24、27、31、33]。此外，真阳性率和假阳性率是众所周知的指标，并被安卓恶意软件检测领域的最先进方法(例如[3，50)广泛采用。

7 相关工作

采矿规范。除了第 4 节中考虑的最先进的基线之外，还有从执行跟踪中挖掘基于 FSA 的规范的其他相关工作。Lo 等人提出了智能集成电路，该智能集成电路使用构造概率性有限状态机的 k-tails 算法的变体，从一组执行轨迹中挖掘有限状态机。马里亚尼等人提出了 k 行为[36]，它通过一次分析一条轨迹来构造一个自动机。沃金肖和博格丹诺夫提出了一种方法，允许用户输入时间属性，以支持规范挖掘器根据执行轨迹构建一个 FSA[45]。Lo 等人进一步扩展了 Walkinshaw 和 Bogdanov 的工作，从执行轨迹中自动推断时间属性，并使用这些属性来自动支持规范挖掘者[33]的模型推断过程。概要从执行轨迹中推断出三种时间不变量，并使用它们生成一个简明的 FSA [6]。SpecForge [26]是一种元方法，它分析由其他规范挖掘者推断的金融服务协议，并将它们组合在一起，以创建更准确的金融服务协议。上述方法都没有采用深度学习。

软件工程任务的深度学习。最近，提出了深度学习方法来学习具有多个抽象层次的数据表示[28]。研究人员一直在[17，25，4749]利用深度学习解决软件工程中的挑战性任务。例如，Gu 等人提出了 DeepAPI，它将自然语言中的查询作为输入，并输出开发人员应该遵循[17]的 API 方法序列。简而言之，DeepAPI 回复了一个基于 RNN 的模型，该模型可以将一种语言的句子翻译成另一种语言的新句子。与深度应用编程接口不同，需求侧管理将应用编程接口或库的输入方法序列作为输入，并输出表示该应用编程接口或库行为的有限状态自动机。在我们工作之前，深度学习模型还没有被用来有效地挖掘规格。

软件工程任务的语言模型。统计语言模型已经被用于许多软件工程任务。例如，欣德尔等人在代码库中使用 n-gram 模型来证明源代码语料库的高度局部重复性，并利用它来改进 Eclipse 的代码完成引擎[19]。其他几项工作扩展了 Hindle 等人的工作，以构建更强大的代码完成引擎；例如，雷-切夫等人利用 n-gram 和递归神经网络语言模型向具有漏洞的程序推荐可能的方法调用序列[41]，而阮氏等人利用隐马尔可夫模型从安卓应用程序字节码中学习应用程序接口用法以推荐应用程序接口[40]。除了代码完成之

外，王等人使用 n-gram 模型通过识别低概率令牌序列来检测错误[46]。我们的工作使用语言模型来完成不同的任务。

8 结论和未来工作

正式规范对许多软件过程都有帮助。在这项工作中，我们提出了需求侧管理，这是一种利用基于递归神经网络的语言模型来挖掘基于需求侧管理的规范的新方法。我们应用 Randoop，一种众所周知的测试用例生成方法，为训练 RNNLM 创建一组更丰富的执行跟踪。从一组采样的执行轨迹中，我们构造了一个前缀树接受器(PTA)，并利用学习的 RNNLM 提取了 PTA 状态的许多特征。然后，聚类算法利用这些特征来合并相似的自动机状态，以使用各种设置构建许多 FSa。然后，我们使用模型选择启发式算法来选择被估计为最精确的有限状态机，并将其作为最终模型输出。我们运行我们提出的方法来推断 11 个目标库类的规范。我们的结果显示，需求侧管理的平均离差为 71.97%，比最佳绩效基线高出 28.82%。此外，我们还提出了一种技术，该技术采用了由需求侧管理挖掘的前端服务协议来检测安卓应用程序中的恶意行为。特别是，我们的技术使用推断的 FSA 作为行为模型来构建一个更全面的沙箱，该沙箱考虑敏感的应用编程接口方法的执行上下文。我们的评估表明，该技术可将黄杨的真阳性率提高 15.69%，而假阳性率仅提高 4.52%。

作为未来的工作，我们计划通过将可能不变量的信息集成到我们基于深度学习的框架中来进一步提高需求侧管理的有效性。我们还计划使用 EvoSuite [16]和许多其他测试用例生成工具来生成一组更加全面的培训跟踪，以提高需求侧管理的有效性。此外，我们计划通过考虑除 k 均值和分层聚类之外的更多聚类算法来改进需求侧管理，尤其是那些不需要输入聚类数量的算法(例如，DBSCAN [15)，等等。)。最后，我们计划用更多的类和库来评估需求侧管理，以减少对外部有效性的威胁。

附加资源和确认。我们的数据集、电力需求侧管理的实施以及包含其他结果的技术报告可在以下网站上公开获取:https://github.com/电力需求侧管理。这项研究得到了新加坡国家研究基金会国家网络安全研究与发展计划的支持(获奖号码:NRF2016NCR-NCR001- 008)。

参考

[1]最后一次访问是在 2017 年 2 月 25 日。在 https://github.com/linkedin/camus.

[2]多梅尼科·阿马尔菲塔诺、安娜·丽塔·法索里诺、波菲里奥·特拉蒙塔纳、萨尔瓦托勒·德·卡明和阿蒂夫·梅蒙。2012.使用图形用户界面抓取来自动测试安卓应用程序。在 2012 年 9 月 3 日至 7 日于德国埃森举行的美国电气工程师协会/美国计算机学会自动化软件工程国际会议上。258-261。

[3]丹尼尔·阿尔普、迈克尔·斯普雷岑巴特、马尔特·胡布纳、雨果·加斯孔和康拉德·里克。2014.DREBIN:在你的口袋里有效和可解释的检测安卓恶意软件。第 21 届年度网络和分布式系统安全研讨会，2014 年 NDSS，美国加利福尼亚州圣地亚哥，2014 年 2 月 23 日至 26 日。

[4]灵峰宝、天都乐、卢大伟。2018.采矿沙箱:我们到了吗？。2018 年 3 月 20 日至 23 日在意大利坎波巴索举行的第 25 届软件分析、进化和再造国际会议上。445-455。

[5]伊万·贝沙斯蒂尼克、塞维多夫·布伦、珍妮·亚伯拉罕森、迈克尔·恩斯特和阿尔温德·克里希纳穆西。2015.使用声明性规范来改进模型推理算法的不足、可扩展性和比较。IEEE 传输。软件工程。41，4 (2015)，408-428。

[6]伊万·贝沙斯蒂尼克、塞维多夫·布伦、齐格鲁德·施耐德、迈克尔·斯隆和迈克尔·恩斯特。2011.利用现有仪器自动推断不变约束模型。在第 19 届 ACM SIGSOFT 研讨会和第 13 届欧洲软件工程基础会议记录中。ACM，267-277。

[7]艾伦·比尔曼和杰罗姆·费尔德曼。1972.从有限状态机的行为样本合成有限状态机。IEEE 传输。计算机。100，6 (1972)，592-597。

[8]曹哲瑞、田园、天都乐、卢大伟。[特区]。基于规则的指定挖掘利用学习排名。自动化软件工程([特区)，1-30。

[9]钟俊英、恰拉尔·居莱尔、赵庆云和约绍·本吉奥。2014.门控递归神经网络对序列建模的实证评价。CoRR abs/1412.3555 (2014)。

[10]小埃德蒙·克拉克、阿娜·格鲁伯格和多伦·佩莱德。1999.模型检查。麻省理工学院出版社，美国马萨诸塞州剑桥。

[11]瓦伦汀·达尔梅尔、尼古拉·克诺普、克里斯托夫·马龙、戈登·弗雷泽、赛巴斯·田哈克和安德列亚斯·泽勒。2012.为规范挖掘自动生成测试用例。IEEE 传输。软件工程。38，2 (2012)，243-257。

[12]瓦伦汀·达尔梅尔、尼古拉·克诺普、克里斯托夫·马龙、塞巴斯蒂安·哈克和安-德雷斯·泽勒。2010.为规范挖掘生成测试用例。2010 年 7 月 12 日至 16 日在意大利特伦托举行的国际软件测试与分析研讨会论文集。85-96。

[13]卢大伟和肖成浩。2006.SMArTIC:致力于构建一个精确、健壮和可扩展的规范挖掘器。2006 年 11 月 5 日至 11 日在美国俄勒冈州波特兰市举行的第 14 届美国计算机学会软件工程基础国际研讨会记录。265-275。

[14]迈克尔·恩斯特、杰夫·珀金斯、菲利普·郭、斯蒂芬·麦卡曼、卡洛斯·帕切科、马修·茨尚茨和陈晓。2007.动态检测可能不变量的 Daikon 系统。Sci。计算机。程序。69，1-3 (2007)，35-45。

[15]马丁·埃斯特、汉斯-彼得·克里格尔、约格·桑德和徐小薇。1996.一种基于密度的大型噪声空间数据库聚类发现算法。在美国俄勒冈州波特兰举行的第二届知识发现和数据挖掘国际会议(KDD-96)上。226-231。

[16]戈登·弗雷泽和安德里亚·阿库里。2011.EvoSuite:面向对象软件的自动测试套件生成。在 2011 年 9 月 5 日至 9 日在匈牙利塞格德举行的 SIGNSOFT/FSE 第 11 届第 19 届 ACM SIGSOFT 软件工程基础研讨会(FSE-19)和 ESEC 第 11 届第 13 届欧洲软件工程会议(ESEC-13)上。416-419。

[17]顾晓东、张洪宇、张冬梅和金成勋。2016.深度应用编程接口学习。2016 年 11 月 13 日至 18 日在美国华盛顿州西雅图市举行的第 24 届美国计算机学会软件工程基础国际研讨会记录。631-642。

[18]郝帅、柳斌、苏曼·纳特、威廉·哈尔丰德和拉梅什·戈文丹。2014.PUMA:可编程用户界面自动化，用于移动应用的大规模动态分析。2014 年 6 月 16 日至 19 日，在美国新罕布什尔州布雷顿森林举行的第 12 届移动系统、应用和服务国际年会上。204-217。

[19]亚伯兰·欣德尔、厄尔·巴尔、苏振东、马克·加贝尔和普雷姆库马·德·万布。2012.论软件的自然性。在 2012 年 6 月 2 日至 9 日于瑞士苏黎士举行的第 34 届软件工程国际会议上，2012 年，ICSE。837-847。

[20]塞普·霍克莱特和于尔根·施密德胡伯。1997.长期短期记忆。神经计算 9，8 (1997)，1735-1780。

[21]康拉德·贾姆罗齐克、菲利普·冯·斯蒂普-雷考斯基和安德烈亚斯·泽勒。2016.最小沙箱。2016 年 5 月 14 日至 22 日在美国德克萨斯州奥斯汀市 ICSE 举行的第 38 届国际软件工程会议记录。37-48。

[22]康拉德·贾姆罗齐克和安德烈亚斯·泽勒。2016.机器人(DroidMate):一个健壮且可扩展的安卓测试生成器。在 2016 年 5 月 14 日至 22 日在美国德克萨斯州奥斯汀举行的移动软件工程和系统国际会议记录中。293-294。

约翰·奈特、科琳·德容、马修·吉布和 luãŋs·中野 1997.为什么正式方法没有得到更广泛的应用？。美国宇航局第四次正式方法研讨会。1-12。

[24]伊沃·克尔卡、塞维多夫·布伦和内纳德·梅德维多维奇。2014.从调用跟踪和方法不变量中自动挖掘指定。2014 年 11 月 16 日至 22 日在中国香港举行的第 22 届美国计算机学会软件工程基础国际研讨会论文集(FSE-22)。178-189。

[25]安哥林、安团阮、和安阮和田恩阮。2015.将深度学习与信息检索相结合，为错误报告本地化错误文件。2015 年 11 月 9 日至 13 日在美国东北林肯举行的第 30 届美国电气工程师协会/美国计算机学会自动化软件工程国际会议上。476-481。

[26]蒂恩-杜伊·贝尔、轩-巴赫·戴尔、卢大伟和伊万·贝斯恰斯蒂尼克。2015.通过模型裂变和聚变给规范矿工提供能量。2015 年 11 月 9 日至 13 日在美国东北林肯举行的第 30 届美国电气工程师协会/美国计算机学会自动化软件工程国际会议上。115-125。

[27]蒂恩-杜伊·贝尔和卢大伟。2015.超越支持和信任:探索基于规则的规范挖掘的兴趣度度量。2015 年 3 月 2 日至 6 日在加拿大蒙特利尔举行的第 22 届美国电气工程师学会软件分析、进化和再设计国际会议上。331-340。

[28]扬·勒春、约舒·本吉奥和杰弗里·辛顿。2015.深度学习。自然 521，7553 (2015)，436-444。

[29]李莉、李道元、泰加文代夫·比斯安代、雅克·克莱因、伊夫·勒特龙、卢大伟和洛伦佐·卡瓦拉罗。2017.理解安卓应用搭载:恶意代码移植的系统研究。IEEE 传输。信息取证与安全 12，6 (2017)，1269-1284。

[30]李园春、杨子岳、郭尧和陈向群。2017.机器人(DroidBot):一个面向安卓的轻量级用户界面引导的测试输入生成器。在 2017 年 ICSE 第 39 届国际软件工程会议记录中，阿根廷布宜诺斯艾利斯，2017 年 5 月 20-28 日-配套卷。23-26。

[31]卢大伟、洪成、韩嘉伟、肖成浩、孙成年。2009.用于故障检测的软件行为分类:一种判别模式挖掘方法。在第 15 届 ACM 全球知识发现与数据挖掘国际会议记录中。ACM，557-566。

[32]卢大伟和肖成浩。2006.夸克:基于自动机的规范挖掘器的经验评估。第十三届逆向工程工作会议(2006 年，WCRE)，2006 年 10 月 23 日至 27 日，意大利贝内文托。51-60。

[33]卢大伟、莱昂纳多·马里亚尼和毛罗·佩泽。2009.行为模型推理的自动导向。2009 年 8 月 24 日至 28 日在荷兰阿姆斯特丹举行的欧洲软件工程会议和 ACM SIGSOFT 软件工程基础国际研讨会第七届联席会议记录。345-354。

[34]卢大伟、莱昂纳多·马里亚尼和毛罗·桑托罗。2012.从软件中学习扩展的 FSA:一个经验评估。系统和软件杂志 85，9 (2012)，2063-2076。

[35]詹姆斯·麦克奎恩等人，1967 年。多元观测数据分类和分析的一些方法。在第五届伯克利数理统计和概率研讨会记录，第 1 卷。美国加利福尼亚州奥克兰。，281-297。

[36]莱昂纳多·马里亚尼，法布里齐奥·帕斯托雷和毛罗·佩泽。2011.集成故障诊断的动态分析。IEEE 传输。软件工程。37，4 (2011)，486-508。

[37]魏凯苗和应劭刘。2012.一种基于规范的集成测试方法。在 SOFL。26-43。

[38]托马斯·米科洛夫、阿诺普·德奥拉斯、斯特凡·康布林克、卢克斯·勃艮第和扬·瑟-诺克。2011.高级语言建模技术的实证评估和组合。2011 年 8 月 27 日至 31 日在意大利佛罗伦萨举行的国际语言交流协会第十二届年会。605-608。

[39]托马什·米科洛夫、马丁·卡拉菲埃特、卢克·勃艮第、扬·恩诺基和桑吉夫·库-`丹普尔。2010.基于递归神经网络的语言模型。在国际言语交际协会第十一届年会上。

[40]阮潭、洪越、冯明武和阮东三。2016.从字节码中学习应用编程接口用法:一种统计方法。2016 年 5 月 14 日至 22 日在美国德克萨斯州奥斯汀市 ICSE 举行的第 38 届国际软件工程会议记录。416-427。

[41]维塞林·赖切夫、马丁·维切夫和埃兰·亚哈夫。2014.用统计语言模型完成代码。2014 年 6 月 9 日至 11 日，在英国爱丁堡 PLDI 14 日举行的 ACM SIGPLAN 编程语言设计与实现会议上。419-428。

[42]布赖恩·罗宾逊、迈克尔·恩斯特、杰夫·珀金斯、维奈·奥古斯丁和李诺。2011.扩展自动测试生成:为程序自动生成可维护的回归单元测试。ASE 2011:第 26 届自动化软件工程国际年会论文集。美国堪萨斯州劳伦斯市，23-32。

[43]利奥·洛克赫和奥德·梅蒙。2005.聚类方法。数据挖掘和知识发现手册。321-352。

[44]伊利亚·苏斯基弗，奥里奥尔·温耶尔斯和科克·维尔。2014.用神经网络进行序列到序列学习。《神经信息处理系统的进展》第 27 期:2014 年神经信息处理系统年会，2014 年 12 月 8 日至 13 日，加拿大魁北克蒙特利尔。3104-3112。

[45]尼尔·沃金肖和基里尔·博格达诺夫。2008.推断具有时间约束的有限状态模型。2008 年 9 月 15 日至 19 日在意大利拉奎拉举行的第 23 届 IEEE/ACM 自动化软件工程国际会议(ASE 2008)。248-257。

[46]王松、德文·乔拉克、达纳·莫夫谢维茨-阿提亚斯和林坦。2016.Bugram:用 n-gram 语言模型进行 bug 检测。2016 年 9 月 3 日至 7 日在新加坡举行的第 31 届美国电气工程师学会/美国计算机学会自动化软件工程国际会议记录。708-719。

[47]王松、刘太岳、林谭。2016.缺陷预测的语义特征自动学习。2016 年 5 月 14 日至 22 日在美国德克萨斯州奥斯汀市 ICSE 举行的第 38 届国际软件工程会议记录。297-308。

[48]马丁·怀特、米歇尔·图法诺、克里斯托弗·文多姆和丹尼斯·波希万尼克。2016.用于代码克隆检测的深度学习代码片段。在第 31 届美国电气工程师学会/美国计算机学会自动化软件工程国际会议记录中，

ASE 2016，新加坡，2016 年 9 月 3-7 日。87-98。

[49]马丁·怀特、克里斯托弗·文多姆、马里奥·利纳雷斯·巴斯克斯和丹尼斯·波希-万尼克。2015.走向深度学习软件仓库。2015 年 5 月 16 日至 17 日在意大利佛罗伦萨举行的第 12 届 IEEE/ACM 采矿软件库工作会议，MSR，2015 年。334-345。

[50]张穆、岳端、恒贤、赵志若。2014.使用加权上下文相关应用编程接口依赖图的语义感知安卓恶意软件分类。2014 年美国亚利桑那州斯科茨代尔 2014 年计算机与通信安全会议录，2014 年 11 月 3-7 日。1105-1116。

[51]钟浩和苏振东。2013.检测应用编程接口文档错误。2013 年美国印第安纳波利斯面向对象编程系统语言与应用国际会议录，面向对象编程语言与应用协会，2013 年，2013 年飞溅的一部分，2013 年 10 月 26-31 日。803-816。