软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.006968]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

Apache Flink 复杂事件处理语言的形式语义*

傅宣登[1,2], 吴志林[1,2]

[1](计算机科学国家重点实验室 (中国科学院 软件研究所), 北京 100190)

[2](中国科学院大学 计算机科学与技术学院, 北京 100049)

通信作者: 吴志林, E-mail: wuzl@ios.ac.cn

摘　要: Apache Flink 是目前最流行的流式计算平台之一, 已经在工业界得到了广泛应用. 复杂事件处理是流式计算的一种重要使用场景, Apache Flink 平台定义并实现了一种复杂事件处理语言 (简称 FlinkCEP). FlinkCEP 语法特性丰富, 不仅包括常见的过滤、连接、循环等操作, 还包括迭代条件、匹配筛选策略等高级特性. FlinkCEP 语义复杂, 尚缺乏语言规范对其语义进行准确描述, 只能通过实现细节来理解, 因此对其语义进行形式描述对于开发人员准确理解其语义非常必要. 针对 FlinkCEP 提出一种数据流转换器的自动机模型, 该模型包括用于刻画迭代条件的数据变量、存储输出结果的数据流变量、用于刻画匹配筛选策略的迁移优先级等特性. 使用数据流转换器对 FlinkCEP 的语义进行形式建模, 并且根据形式语义设计 FlinkCEP 的查询求值算法, 实现原型系统. 进一步, 生成能够较为全面覆盖 FlinkCEP 语法特性的测试用例集, 利用这些测试用例与 FlinkCEP 在 Flink 平台上的实际运行结果进行对比实验. 实验结果表明所提出的形式语义与 FlinkCEP 在 Flink 平台上的实际语义基本是一致的. 而且, 对实验结果不一致的情况进行分析, 指出 FlinkCEP 在 Flink 平台上的实现对于组模式的处理可能存在错误.

关键词: 流式计算; Flink; 复杂事件处理; 形式语义; 数据流转换器; 查询求值

中图法分类号: TP311

中文引用格式: 傅宣登, 吴志林. Apache Flink复杂事件处理语言的形式语义. 软件学报. http://www.jos.org.cn/1000-9825/6968.htm

英文引用格式: Fu XD, Wu ZL. Formal Semantics of Apache Flink Complex Event Processing Language. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/6968.htm

Formal Semantics of Apache Flink Complex Event Processing Language

FU Xuan-Deng[1,2], WU Zhi-Lin[1,2]

[1](State Key Laboratory of Computer Science (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[2](School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Apache Flink is one of the most popular stream computing platforms and has many applications in industry. Complex event processing (CEP) is one of the important usage scenarios of stream computation. Apache Flink defines and implements a language for complex event processing (referred to as FlinkCEP). FlinkCEP includes rich syntactic features, not only the usual features of filtering, connecting, and looping, but also the advanced features of iterative conditions and after-match skip strategies. The semantics of FlinkCEP is complex, no language specification of FlinkCEP defines its semantics precisely, so it can only be understood by checking the implementation details. This motivates the definition of formal semantics for FlinkCEP so that the developers could understand its semantics precisely. This study proposes an automaton model called data stream transducers (DST) for FlinkCEP, where the data variables are applied to capture the iterative conditions, the data stream variables are adopted to store the outputs, and transition priorities are introduced to capture the after-match skip strategies. DST is leveraged to define the formal semantics of FlinkCEP and design the query evaluation algorithms based on the formal semantics. Moreover, a prototype of the CEP engine is implemented. Finally, test case sets are generated, which cover the syntactic features of FlinkCEP more comprehensively. They are utilized to conduct comparison experiments against the actual results of FlinkCEP on the Flink platform. The experimental results show that the proposed formal semantics of

* 基金项目: 国家自然科学基金 (61872340)
收稿时间: 2022-12-08; 修改时间: 2023-02-21, 2023-04-15; 采用时间: 2023-05-16; jos 在线出版时间: 2023-11-22

FlinkCEP conforms to the actual semantics of FlinkCEP in the vast majority of the cases. Furthermore, the inconsistencies between the formal and the actual semantics are analyzed and it is discovered that the Flink implementation of FlinkCEP may not deal with the group patterns correctly.

Key words: stream computing; Flink; complex event processing; formal semantics; data stream transducer (DST); query evaluation

1 介 绍

随着计算机技术的飞速发展, 整个社会的信息化程度不断提高. 最近 5–10 年, 由于物联网、在线社交网络等新型应用需求的推动, 信息社会中的数据量每天都在急剧增长. 如何对这些超大规模的数据进行快速准确的处理对于计算机的学术界与产业界都提出了很大的挑战. 为了应对这种挑战, 计算机的从业人员已经开发了各种大数据处理平台. 这些大数据处理平台按照计算模式来划分, 可以分为批处理平台和流处理平台.

批处理一般用于对大规模的历史数据进行集中处理. 目前主流的大数据批处理平台基本上都遵循 MapReduce 框架[1]. 开源批处理平台包括 Apache Hadoop (http://hadoop.apache.org/)、Apache Spark (http://spark.apache.org/) 等. 商业批处理平台包括亚马逊的 EC2 (https://aws.amazon.com/ec2)、微软的 SCOPE[2]、Yahoo 的 Pig Latin[3]、Facebook 的 Hive[4]、腾讯的大数据处理套件 TBDS (https://www.qcloud.com/product/tbds)、阿里的大数据计算服务 MaxCompute (https://data.aliyun.com/) 等.

批处理平台虽然具有吞吐量大的优点, 但是无法对实时数据流, 比如传感器或在线社交网络产生的数据流, 进行快速处理. 一般来讲, 数据流是指由若干数据源产生的连续无限数据记录序列. 计算机学术界和产业界已经开发了各种流处理平台, 以对连续无限数据序列进行实时处理. 流处理与批处理相比, 除了具有实时性的优点之外, 数据处理也更加灵活, 更反映大数据处理的本质. 开源流处理平台包括 Apache SPARK Streaming[5]、Apache Apex (https://apex.apache.org/)、Apache Beam (https://beam.apache.org/)、Apache Flink (http://flink. apache.org/)、Apache Storm(http://storm.apache.org/) 等. 商业流处理平台包括 Google Cloud Dataflow (https://cloud.google.com/dataflow/)、亚马逊的 Kinesis (https://aws.amazon.com/kinesis/)、微软的 StreamScope[6]、Twitter 的 Heron[7]、IBM Streams (https://ibmstreams.github.io/) 等.

Apache Flink 是一个具有代表性的开源流处理平台, Flink 的核心是一个用 Java 和 Scala 写的分布式的数据流引擎, Flink 以数据并行和流水线的方式来执行数据流程序, Flink 的流水线式的运行时系统使得可以同时支持批处理和流处理程序的执行. Apache Flink 目前已经在工业界得到了广泛应用, 包括阿里巴巴、亚马逊、滴滴、eBay、爱立信、UBER、华为、腾讯、小米等.

复杂事件处理 (complex event processing) 是流处理平台的典型应用场景之一, 其对事件流进行实时处理, 从中提取出有用的信息, 其目标是及时发现有意义的事件模式 (比如网络攻击) 并做出快速响应. FlinkCEP (https://nightlies.apache.org/flink/flink-docs-release-1.16/docs/libs/cep/) 是在 Apache Flink 平台上实现的一个复杂事件处理的 API 库, 使得可以在输入事件流中检测事件模式. FlinkCEP 提供了一组模式 API 使得可以对用户想从输入数据流 (事件序列) 提取的复杂模式序列进行描述. 每一个复杂模式序列包括多个简单模式 (即描述具有某种共性的输入事件). 复杂模式序列的一次匹配是一个能够访问复杂模式序列中的所有简单模式的由某些输入事件构成的序列. FlinkCEP 已经在实时股票预测、故障检测、网络欺诈检测、电子商务用户行为实时分析、网络恶意攻击检测、交通、物流、公共安全等领域得到了广泛应用.

FlinkCEP 的简单模式包括两种, 单一模式和循环模式, 单一模式描述单一输入事件的性质, 而循环模式则描述多个具有某种共性的单一事件组成的序列. 简单模式可以通过具有不同连续性 (contiguity) 的连接操作组成一个模式序列. FlinkCEP 提供了具有以下 3 种不同连续性的连接操作: 严格连续 (strict contiguity)、放松连续 (relaxed contiguity)、非确定放松连续 (non-deterministic relaxed contiguity). 除了上述简单模式序列之外, FlinkCEP 还包括两种高级特性: 迭代条件与匹配筛选策略. 迭代条件类似于函数式编程语言中的折叠函数 (fold function), 可以对已经匹配的部分事件序列的信息进行聚合, 在此基础上对其需要满足的性质进行描述. FlinkCEP 按照某种顺序对模式序列的多个匹配对应的输入事件序列进行输出. 而且, FlinkCEP 还包括匹配筛选策略, 即对模式序列的多个匹配进行筛选, 在输出中忽略某些匹配对应的事件序列. FlinkCEP 提供了多种匹配筛选策略, 包括 NoSkip、

SkipToNext、SkipPastLastEvent 等.

FlinkCEP 的这些特性和机制结合起来使得 FlinkCEP 查询的语义变得非常复杂和不容易理解, 尤其是对于不同连续性与匹配筛选策略的语义. 目前 FlinkCEP 的官方文档对于这些机制的语义描述也是模糊甚至是不准确的, 这使得开发人员很难准确理解和把握 FlinkCEP 的语义, 因此对其语义进行准确的形式建模是非常必要.

虽然目前已经有一些针对复杂事件处理语言的形式建模的工作, 比如文献 [8–10], 但这些工作中提出的复杂事件自动机模型表达能力受限, 无法对 FlinkCEP 的迭代条件和匹配筛选策略这两种高级特性进行建模, 因此需要建立新的理论框架对 FlinkCEP 的语义进行形式建模.

针对 FlinkCEP 的语义形式建模问题, 本文做出了如下贡献.

● 首先, 我们提出了一个称为流转换器的自动机模型来刻画 FlinkCEP 的各种特性. 具体来讲, 流转换器的输入为数据流, 它通过数据变量来存储迭代条件中用到的聚合操作的结果, 通过数据流变量来存储匹配的输出, 通过迁移的优先级来刻画匹配输出数据流的顺序和匹配筛选策略.

● 然后, 我们通过提供从 FlinkCEP 查询到数据流转换器的翻译来得到了 FlinkCEP 的形式语义.

● 而且, 基于形式语义, 我们提出了 FlinkCEP 的在线查询求值算法, 并实现了原型系统.

● 最后, 我们生成了能够涵盖 FlinkCEP 的各种语法特性的测试用例集, 并在测试用例集上与 FlinkCEP 查询在 Flink 平台上的实际运行结果进行了对比实验. 实验结果证实了我们提出的形式语义与 FlinkCEP 实际语义的一致性.

我们需要指出, 复杂事件处理语言 FlinkCEP 与关系数据库查询语言 SQL 类似, 是一种描述性的语言, 其语义形式化与命令式编程语言 (比如 C 语言等) 的语义形式化 (比如操作语义和指称语义) 差别较大. 粗略来讲, FlinkCEP 可以看成正则表达式的扩展, 而正则表达式的语义可以通过有限自动机来刻画, 因此我们使用数据流转换器的自动机模型来刻画 FlinkCEP 的形式语义.

本文第 2 节讨论相关工作. 第 3 节介绍基础知识. 第 4 节定义 FlinkCEP 的语法. 第 5 节给出数据流转换器的定义. 第 6 节提供 FlinkCEP 查询到数据流转换器的翻译. 第 7 节提出 FlinkCEP 查询求值算法. 第 8 节描述实现和对比试验. 第 9 节进行总结与展望.

2 相关工作

在本节我们讨论另外一些相关工作, 我们主要聚焦在复杂事件处理语言和平台方面.

FlinkCEP 的实现可以看成 SASE 语言实现文献 [11,12] 的扩展. SASE 语言允许使用过滤操作、连接操作、循环、窗口操作等对复杂事件模式进行描述, 但 SASE 不包含迭代条件和匹配筛选策略. SASE 语言实现基于非确定自动机的一种简单扩展, 而 FlinkCEP 的实现直接使用了 SASE 中的自动机模型, 该自动机模型无法对迭代条件和匹配筛选策略进行语义建模, 这两者的语义只是通过系统实现来确定. 数据流转换器通过在有限自动机模型中引入迁移的优先级、数据变量和数据流变量来实现对迭代条件和匹配筛选策略的准确语义建模.

Esper 复杂事件处理平台使用 EPL 语言 (https://www.espertech.com/esper/esper-documentation/). EPL 语言允许使用过滤、窗口、循环、连接等操作来描述查询, 但 EPL 只包含严格连续的连接操作, 不允许放松连续和非确定放松连续的连接操作, 也不包含迭代条件和匹配筛选策略.

OpenCEP 语言和平台 (https://research.redhat.com/blog/research_project/complex-event-processing-2/) 允许使用过滤操作、事件布尔操作、连接操作、循环操作来描述复杂事件模式, 但不允许非严格连续的连接操作、也不包含迭代条件和匹配筛选策略.

另外, 带有优先级的流式字符串转换器也被提出用于编程语言中的正则表达式的语义刻画 [13]. 本文提出的数据流转换器可以看成流式字符串转换器在数据流的扩展, 其区别在于字符串上字母表是有限的, 而数据流转换器的字母表包括事件名和属性值, 因此是无穷的.

3 基础知识

对于自然数 n, 我们用 $[n]$ 表示集合 $\{1,\ldots,n\}$.

在本文中, 令A为一个属性名字的集合, V为属性值的集合, E是事件名字的有限集合. 为了简单起见, 我们假设所有的属性的取值来自于同一个集合V, 而且V为整数集合. 本文后面提出的对 FlinkCEP 通过数据流转换器来进行形式语义建模以及相应的查询求值算法与所使用的数据类型相对独立, 这些结果可以很自然的扩展到其他数据类型上, 比如字符串数据类型.

给定一个变量集合X, 我们用E_X表示使用X中的变量的整数线性算术表达式, 即使用以下语法定义的表达式, $t := v \mid x \mid t+t \mid t-t \mid vt$, 其中$v$是整数常量, x是整数变量. 而且, 我们用L_X表示仅使用X中的变量的无量词的整数线性算术公式, 即使用如下语法定义的公式:

$$\varphi := t_1 \ op \ t_2 \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi,$$

其中, $t_1, t_2 \in E_X$, 且$op \in \{=, \neq, <, \leqslant, >, \geqslant\}$.

变量来自于集合X的整数线性算术表达式t和整数线性算术公式ψ的语义解释在一个赋值函数$\eta : X \to V$上, 记为$[\![t]\!]_\eta$和$[\![\psi]\!]_\eta$, 其定义如下.

- $[\![v]\!]_\eta = v$, $[\![x]\!]_\eta = \eta(x)$, $[\![t_1 + t_2]\!]_\eta = [\![t_1]\!]_\eta + [\![t_2]\!]_\eta$, $[\![t_1 - t_2]\!]_\eta = [\![t_1]\!]_\eta - [\![t_2]\!]_\eta$, $[\![vt]\!]_\eta = v \times [\![t]\!]_\eta$.
- $[\![t_1 = t_2]\!]_\eta$为真当且仅当$[\![t_1]\!]_\eta$等于$[\![t_2]\!]_\eta$, $[\![t_1 \neq t_2]\!]_\eta$等的解释类似.
- $[\![\varphi_1 \vee \varphi_2]\!]_\eta$为真当且仅当$[\![\varphi_1]\!]_\eta$为真或者$[\![\varphi_2]\!]_\eta$为真.
- $[\![\varphi_1 \wedge \varphi_2]\!]_\eta$为真当且仅当$[\![\varphi_1]\!]_\eta$为真且$[\![\varphi_2]\!]_\eta$为真.
- $[\![\neg\varphi_1]\!]_\eta$为真当且仅当$[\![\varphi_1]\!]_\eta$为假.

给定一个集合X, 我们用$\vec{X}$表示由X上的元素构成的 (有限) 向量的集合. 令$\vec{x} \in \vec{X}$, 我们用$x' \in \vec{x}$表示x'在$\vec{x}$中出现. 我们用$X \to Y$表示X到Y的函数的集合, 而用$X \rightharpoonup Y$表示X到Y的偏函数的集合. 对于$f \in X \rightharpoonup Y$, 我们用$dom(f)$表示$f$的定义域, 即$f(x)$有定义的所有元素$x$的集合.

每个事件名字$e \in E$关联着一个事件模式, 即一个属性名字的有限元组, 记为$sch(e)$ (即存在$l > 0$使得$sch(e) \in A^l$).

一个事件是一个二元组$(e, (a_1 : v_1, \ldots, a_l : v_l))$, 这里$e \in E$, $sch(e) = (a_1, \ldots, a_l)$, 且$(v_1, \ldots, v_l) \in V^l$. 而且, 直观起见, 我们一般可以省略属性名字, 将$(e, (a_1 : v_1, \ldots, a_l : v_l))$简写为$e(v_1, \ldots, v_l)$.

一个数据流是事件的无穷序列, 一个有限数据流是事件的有穷序列. 我们用ε表示空数据流, 用$\mathcal{S}$表示所有数据流的集合, 而$\mathcal{S}_f$表示所有有限数据流的集合. 我们说数据流$s_1 \in \mathcal{S}_f$是$s_2 \in \mathcal{S}_f$的子数据流, 如果存在数据流$u, v \in \mathcal{S}_f$使得$s_2 = us_1v$.

4 FlinkCEP 的语法

FlinkCEP 的语法由如下规则来定义.

$$\begin{cases}
flinkcep & ::= & (patseq)_{NoSkip} \mid (patseq)_{SkipToNext} \mid (patseq)_{SkipPastLastEvent} \\
patseq & ::= & ipat \mid gpat \mid patseq\ \theta\ ipat \mid patseq \cdot gpat,\ \theta \in \{\cdot, \circ, \odot\} \\
ipat & ::= & spat \mid lpat \\
spat & ::= & p : e : [cndt] \\
lpat & ::= & p : e : [cndt]_\theta\{n, m\} \mid p : e : [cndt]_\theta\{n, \infty\} \mid p : e : [cndt]_\theta\{n, \infty\} U [cndt],\ \theta \in \{\cdot, \circ, \odot\} \\
gpat & ::= & (patseq) \mid (patseq)\{n, m\} \mid (patseq)\{n, \infty\} \mid (patseq)\{n, \infty\} U [cndt], \\
cndt & ::= & scndt \mid icndt \mid cndt \wedge cndt \mid cndt \vee cndt \mid \neg cndt \\
scndt & ::= & \psi\left(\overrightarrow{cur}\right),\ \psi\text{不含有折叠项} \\
icndt & ::= & \psi\left(\overrightarrow{cur}\right),\ \psi\text{含有折叠项} \\
\psi & ::= & t\ op\ t \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi,\ op \in \{=, \neq, <, \leqslant, >, \geqslant\} \\
t & ::= & v \mid x \mid t+t \mid t-t \mid vt \mid \left(fold_{p:e:\vec{z}=\overrightarrow{fexp\left(\vec{z}, \overrightarrow{cur}\right)}}\left(\overrightarrow{v_0}\right)\right)[i] \\
fexp & ::= & v \mid x \mid fexp + fexp \mid fexp - fexp \mid v \cdot fexp \mid ite(\psi, fexp, fexp)
\end{cases},$$

其中,

• p和e分别代表简单模式和事件的名字.

• n,m是自然数, 且$n \leqslant m$.

• 变量向量$\overrightarrow{cur} = (cur_1,\ldots,cur_l)$表示当前事件$e$的属性值组成的元组.

• $\left(fold_{p:e:\vec{z}=\overrightarrow{fexp(\vec{z},\overrightarrow{cur})}}(\overrightarrow{v_0})\right)[i]$称为折叠项, 其表示折叠函数$fold_{p:e:\vec{z}=\overrightarrow{fexp(\vec{z},\overrightarrow{cur})}}(\overrightarrow{v_0})$计算完成后, 状态变量$z_i$最后的值, 其中$p$表示折叠函数所关注的简单模式的名字, e表示简单模式对应的事件的名字, $\overrightarrow{v_0}$表示初始值, $\vec{z}$表示折叠函数的状态变量组成的向量, $fexp(\vec{z},\overrightarrow{cur})$表示用于状态变量向量更新的表达式向量 (这些表达式里面只含有$\vec{z}$和$\overrightarrow{cur}$中的变量), $\overrightarrow{v_0}$表示V中的常数值组成的向量. 状态变量更新表达式$fexp$含有$ite(scndt,fexp,fexp)$, 使得可以在更新状态变量表达式的时候根据不同的情况使用不同的表达式进行更新. 注意折叠项中的变量$\vec{z}$不是自由变量, 而是看成受限变量. 而且, $\overrightarrow{fexp(\vec{z},\overrightarrow{cur})}$中的所有变量来自于$\vec{z}$和$\overrightarrow{cur}$.

直观来讲, 一个 FlinkCEP 查询形如$(patseq)_{SkipStrategy}$, 其中$patseq$是模式序列, $SkipStrategy$称为匹配筛选策略, 用于匹配之后输出结果时用来过滤掉一些匹配的输出. 模式序列$patseq$是由个体模式$ipat$构成的序列, 个体模式分为单例模式$ipat$和循环模式$lpat$.

● 单例模式$ipat$是一个三元组$p:e:[cndt]$, 其中p表示模式的名字, e表示模式对应的事件, 而$cndt$表示事件中的属性值需要满足的条件.

● 循环模式$lpat$有 3 种形式:

■ $p:e:[cndt]_\theta\{n,m\}$: 满足$cndt$的事件出现至少n次, 至多m次, 且事件的相邻两次出现的相对位置与连续性标识θ相符合: 如果$\theta = \cdot$, 则满足条件的事件的下一次出现跟上一次的出现接壤, 即要在上一次出现位置之后的第一个位置; 如果$\theta = \circ$则满足条件的事件的下一次出现可以不跟上一次出现接壤, 但必须是上一次出现最后位置之后的第 1 次出现; 如果$\theta = \odot$, 则满足条件的事件的下一次出现可以不跟上一次出现接壤, 且可以是上一次出现最后位置之后的任意出现.

■ $p:e:[cndt]_\theta\{n,\infty\}$: 满足$cndt$的事件出现至少$n$次, 且满足条件的事件的相邻两次出现的相对位置与$\theta$相符合.

■ $p:e:[cndt]_\theta\{n,\infty\}U[cndt']$: 满足$cndt$的事件出现至少$n$次, 直至$cndt'$满足为止 (或者$cndt'$一直不满足, 满足$cndt$的事件一直出现). 而且, 满足条件的事件的相邻两次出现的相对位置与θ相符合. 这里我们称$U[cndt']$为终止条件.

● 组模式$gpat$与$lpat$类似, 但要求$patseq$的相邻两次匹配要接壤, 比如$(patseq)\{n,m\}$表示$patseq$匹配至少n次, 至多m次, 且相邻两次匹配须满足后一次匹配的开始位置刚好是上一次匹配的结束位置的下一个位置.

而且, 在 FlinkCEP 的语法定义中, 我们考虑 3 种具有不同连续性 (continguity) 的连接操作.

● 严格连续连接操作$patseq \cdot ipat$: 个体模式$ipat$的匹配在$patseq$的匹配之后, 而且需要和$patseq$的匹配接壤, 即$ipat$匹配的第一个位置是$patseq$的匹配的最后一个位置的下一个位置.

● 非严格连续连接操作$patseq \circ ipat$: 个体模式$ipat$的匹配在$patseq$的匹配之后, 但不一定和$patseq$的匹配接壤, 即$ipat$的匹配的第 1 个位置可以是$patseq$匹配的最后一个位置的下一个位置之后的位置, 但要求$ipat$匹配必须是$patseq$的匹配之后的最左匹配.

● 非确定非严格连续操作$patseq \odot ipat$: $ipat$的匹配不一定和个体模式$patseq$的匹配接壤, 而且可以非确定地忽略$ipat$的任意匹配.

FlinkCEP 中的条件分为简单条件$scndt$和迭代条件$icndt$, $scndt$和$icndt$都是线性算术公式, 它们的区别在于前者不含有折叠项, 而后者含有折叠项.

我们要求 FlinkCEP 的$patseq$满足如下条件.

● 所有折叠项中出现的模式名字和事件名字二元组$p:e$必须在非折叠项中出现.

● 对于所有的模式名字p, 如果忽略p在折叠项中的出现, 其在$patseq$只出现一次.

匹配筛选策略包括以下 3 种策略.

● *NoSkip*: 所有匹配的结果均被输出.

● *SkipToNext*: 当一个匹配被发现且输出结果之后, 所有从同一位置开始的部分匹配将被忽略.

● *SkipPastLastEvent*: 当一个匹配被发现且输出结果之后, 所有从该匹配结束位置之前位置 (包括结束位置本身) 开始的所有部分匹配将被忽略.

例 1: 我们考虑电子商务中的商品购买事件 e, 它具有 3 个属性, 即 $(id, name, price)$, 其中 *id* 表示商品的标识, *name* 表示商品的名字, *price* 表示商品的价格. 下面的查询匹配数据流中 *name* 为 1 且 *price* 之和不超过 10 的位置可以不连续的序列, 而且该查询的匹配结束之前不允许 *name* 为 2 的事件出现.

$$\left(p_1 : e : \left[cur_{name} = 1 \wedge \left(fold_{p_1:e:z=z+cur_{price}}(0)\right)[1] + cur_{price} \leqslant 10\right]_{\circ} \{1, \infty\} U[cur_{name} = 2]\right)_{NoSkip}.$$

该查询在输入数据流 $e(1,1,2)$, $e(2,2,5)$, $e(3,1,6)$, $e(4,3,2)$, $e(5,1,5)$上存在以下 4 次匹配: $e(1,1,2)$、 $e(3,1,6)$、$e(3,1,6)$、 $e(5,1,5)$. 注意, $e(1,1,2)$, $e(3,1,6)$不是该查询的匹配, 因为在 $e(3,1,6)$之前存在 *name* 为 2 的事件 $e(2,2,5)$, 而且 $e(3,1,6)$, $e(5,1,5)$ 也不是该查询的匹配, 因为其 *price* 之和大于 10.

为了定义 FlinkCEP 的形式语义, 在第 5 节中我们将引入一种称为数据流转换器的自动机模型.

5 数据流转换器

给定一个数据流变量的有限集合 Y, 我们用 S_Y 表示由事件常量和 Y 中的数据流变量通过连接操作构造的数据流表达式的集合, 即通过如下语法构造的表达式的集合, $s := e(\vec{n})|y|s_1 \cdot s_2$, 这里这里 $e(\vec{n})$ 表示事件常量 (e 表示事件的名字, $\vec{n}$ 表示事件属性值组成的元组), $y \in Y$, 而 $s_1 \cdot s_2$ 表示数据流表达式 s_1 和 s_2 的连接. 数据流表达式 s 解释在一个赋值函数 $\theta : Y \to \mathcal{S}_f$ 上, 记为 $[\![s]\!]_\theta$, 即从 $[\![s]\!]_\theta$ 为从 s 中把每个数据流变量 y 替换为 $\theta(y)$ 之后得到的数据流.

定义 1. 一个数据流转换器 $\mathcal{T}$ 是一个元组 $(\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 其中,

● Σ 表示事件种类的有限集合.

● X 是整数变量的有限集合.

● Y 是数据流变量的有限集合.

● Q 是状态的有限集合.

● $\delta \in (Q \times \Sigma \times L_{X\cup\{Cur\}}) \rightharpoonup \overrightarrow{Q \times (E_{X\cup\{Cur\}})^X \times (S_{Y\cup\{curEvent\}})^Y}$ 定义从 (q, e, ψ) 出发的非空迁移, 而且确定了这些迁移的优先级, 其中 $Cur = \overrightarrow{cur}$, 而且, δ 需要满足以下约束.

■ $dom(\delta)$, δ 的定义域, 是有限的, 即 $dom(\delta)$ 为 $Q \times \Sigma \times L_{X\cup\{Cur\}}$ 的一个有限子集, 其中 $L_{X\cup\{Cur\}}$ 为仅使用 $X \cup \{Cur\}$ 中的变量的无量词的整数线性算术公式 (具体定义参见第 3 节).

■ 对于所有迁移 $(q, e, \psi_1), (q, e, \psi_2) \in dom(\delta)$, 如果 $\psi_1 \neq \psi_2$, 则 $\psi_1 \wedge \psi_2$ 不可满足.

■ 对于所有 $(q, e, \psi) \in dom(\delta)$ 和 $(q', \alpha, \beta) \in \delta(q, e, \psi)$, 我们有对于所有的 $y \in Y$, $\beta(y) = y \cdot curEvent$, 或 $\beta(y) = y$, 或 $\beta(y) = curEvent \cdot y$.

● $\xi \in Q \rightharpoonup Q$ 定义了从给定 q 出发的空迁移, 而且确定了这些迁移的优先级.

● $q_0 \in Q$ 是初始状态.

● $\eta_0 : V^X$ 给 X 中的变量赋初始值.

● $F \subseteq Q$ 表示接受状态的有限集合. 在接受状态可以产生输出, 其输出形式为 $(y_1 : s_1), \ldots, (y_k : s_k)$, 其中 $Y = \{y_1, \ldots, y_k\}$, 而 $s_1, \ldots, s_k$ 为存储在 $y_1, \ldots, y_k$ 中的数据流.

直观来讲, $\mathcal{T}$ 中的迁移分为如下两种.

● 非空迁移: 如果 $(q_2, \alpha, \beta) \in \delta(q_1, e, \psi)$, 则 $(q_1, e, \psi, q_2, \alpha, \beta)$ 被称为非空迁移.

● 空迁移: 如果 $q' \in \xi(q)$, 则 (q, ε, q') 称为空迁移.

我们假设非空迁移优先级高于空迁移. 具体来讲, 我们对于从状态 q 读取事件 e 且满足条件 ψ 的所有非空迁移, 以及从 q 出发的所有空迁移按照优先级进行排序, 即对于 $(q, e, \psi) \in dom(\delta)$, 令 $\delta(q, e, \psi) = (q_1, \alpha_1, \beta_1) \ldots (q_k, \alpha_k, \beta_k)$, $\xi(q) = (q'_1, \ldots, q'_l)$, 则这些迁移按照优先级从高到低排列应该是:

$$(q,e,\psi,q_1,\alpha_1,\beta_1),\ldots,(q,e,\psi,q_k,\alpha_k,\beta_k),(q,\varepsilon,q_1'),\ldots,(q,\varepsilon,q_l').$$

另外, 我们说非空迁移 $(q_1,e,\psi,q_2,\alpha,\beta)$ 是一个存储迁移, 如果存在 $y\in Y$ 使得 *curEvent* 在 $\beta(y)$ 中出现.

下面我们定义数据流转换器的语义.

令 $\mathcal{T}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$ 为一个数据流转换器. $\mathcal{T}$ 的一个运行格局为三元组 (q,η,ctx), 其中 $q\in Q$, $\eta\in X\to V$ 为一个赋值函数, $ctx\in Y\to\mathcal{S}_f$ 是一个上下文.

我们定义初始格局为 $(q_0,\eta_0,ctx_\varepsilon)$, 这里对于所有的 $y\in Y$, 我们有 $ctx_\varepsilon(y)=\varepsilon$.

对于两个格局 (q_1,η_1,ctx_1)、(q_2,η_2,ctx_2), 事件 $e(\vec{v})$, 迁移 $\tau=(q_1,e,\psi,q_2,\alpha,\beta)$, 我们说 $(q_1,\eta_1,ctx_1)\xrightarrow[\mathcal{T}]{e(\vec{v}),\tau}(q_2,\eta_2,ctx_2)$, 如果 $[\![\psi]\!]_{\eta_1[\vec{v}/\overrightarrow{cur}]}$ 为真, 对于每一个 $x\in X$, $\eta_2(x)=[\![\alpha(x)]\!]_{\eta_1[\vec{v}/\overrightarrow{cur}]}$, 且对每一个 $y\in Y$, $ctx_2(y)=[\![\beta(y)]\!]_{ctx_1[e(\vec{v})/curEvent]}$. 这里迁移条件 ψ 是一个无量词的整数线性算术公式, 其语义解释 $[\![\psi]\!]_{\eta_1[\vec{v}/\overrightarrow{cur}]}$ 见第 3 节. 而且, 对于两个格局 (q_1,η_1,ctx_1)、(q_2,η_2,ctx_2), 与迁移 $\tau=(q_1,\varepsilon,q_2)$, 我们说 $(q_1,\eta_1,ctx_1)\xrightarrow[\mathcal{T}]{\varepsilon,\tau}(q_2,\eta_2,ctx_2)$, 如果 $\eta_2=\eta_1$ 且 $ctx_2(y)=[\![\beta(y)]\!](ctx_1(y))$.

对于数据流 $s=e_1(\overrightarrow{v_1})\ldots e_n(\overrightarrow{v_n})$, $\mathcal{T}$ 在 s 上的一个运行是一个序列 $\rho=c_0\tau_1c_1\tau_2c_2\ldots c_{m-1}\tau_mc_m$, 其须满足如下条件.

- $c_0=(q_0,\eta_0,ctx_\varepsilon)$.
- 对于所有的 $1\leqslant i\leqslant m$, 设 $c_i=(q_i,\eta_i,ctx_i)$、$\tau_i=(q_{i-1},e_i',\psi_{i-1},q_i,\alpha_{i-1},\beta_{i-1})$ 或 $\tau_i=(q_{i-1},\varepsilon,q_i)$ (这里令 $e_i'=\varepsilon$), 则 $c_{i-1}\xrightarrow[\mathcal{T}]{\tau_i}c_i$.
- $e_1\ldots e_n=e_1'e_2'\ldots e_m'$.
- 在 ρ 的所有空转移子序列上, 状态不会重复, 即对于任意 $\rho[i,j]=c_i\tau_{i+1}c_{i+1}\ldots c_{j-1}\tau_jc_j$, 如果 $\tau_{i+1},\ldots,\tau_j$ 均为空转移, 则我们有 $q_i,\ldots,q_j$ 互不相同.
- ρ 至少包括一次非空迁移, 而且 ρ 的第 1 个和最后一个非空迁移是存储迁移.

上述倒数第 2 个条件保证了对于给定数据流 $\sigma=e_1(\overrightarrow{v_1})\ldots e_n(\overrightarrow{v_n})$, $\mathcal{T}$ 在 S 上的运行数目是有限的. 倒数第 1 个条件则保证了 S 的第 1 个和最后一个事件一定被存储在输出中 (如果最终输出的话).

对于 $\mathcal{T}$ 在 σ 上的运行 $\rho=c_0\tau_1c_1\tau_2c_2\ldots c_{m-1}\tau_mc_m$, $1\leqslant i\leqslant m$, 令 $c_i=(q_i,\eta_i,ctx_i)$, 如果 $q_m\in F(Q)$, 则我们称 ρ 是 $\mathcal{T}$ 在 S 上的一个可接受运行, 且 $\mathcal{T}$ 接受 S, 并产生输出 $Out(\rho)=(y:ctx_m(y))_{y\in Y}$. 另外, 为了后面算法描述的方便, 对于空数据流 ε, 我们说 $\mathcal{T}$ 接受它, 如果 $q_0\in F$. 而且, 我们认为 $\mathcal{T}$ 接受 ε 只有唯一的运行 c_0, 也就是说, 我们不考虑全部由空迁移组成的可接受运行. 对于给定数据流 S, 我们用 $\mathcal{R}_\mathcal{T}(S)$ 表示 $\mathcal{T}$ 在 S 上的所有可接受运行.

接下来我们利用迁移的优先级来定义同一个数据流上的可接受运行之间的优先级.

设 $s=e_1(\overrightarrow{v_1})\ldots e_n(\overrightarrow{v_n})$ 为数据流, $\mathcal{T}$ 在 S 上的两个不同的运行为 $\rho_1=c_{1,0}\tau_{1,1}c_{1,1}\tau_{1,2}c_{1,2}\ldots c_{1,m_1-1}\tau_{1,m_1}c_{1,m_1}$ 和 $\rho_2=c_{2,0}\tau_{2,1}c_{2,1}\tau_{2,2}c_{2,2}\ldots c_{2,m_2-1}\tau_{2,m_2}c_{2,m_2}$. 我们说 ρ_2 要比 ρ_1 优先级低, 记为 $\rho_2\prec\rho_1$, 如果存在 r 使得 $\tau_{1,1}\ldots\tau_{1,r-1}=\tau_{2,1}\ldots\tau_{2,r-1}$, 且 $\tau_{1,r}$ 优先级要比 $\tau_{2,r}$ 高.

从上面的定义可以看出, $\prec$ 是为 $\mathcal{T}$ 在 S 的数据流上的可接受运行的一个全序关系, 即对于任何两个不同的可接受运行 ρ_1 和 ρ_2, 要么 $\rho_1\prec\rho_2$, 要么 $\rho_2\prec\rho_1$.

例 2: 对应于例 1 中查询的数据流转换器为 $\mathcal{T}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 其中 $\Sigma=\{e\}$, $X=\{z\}$, $Y=\{y_{p_1}\}$, $Q=\{q_0,q_1\}$, $\eta_0(z)=0$, $F=\{q_1\}$, ξ 定义域为空, 而 δ 中的迁移如图 1 所示, 其中如果变量的值在迁移中没有改变, 则其赋值操作在迁移中进行了省略, 而且 $F=\{q_1\}$.

$\mathcal{T}$ 在数据流 $e(1,1,2)$ 上存在一个可接受运行 $\rho=c_0\tau_1c_1$, 这里:

- $c_0=(q_0,\eta_0,ctx_\varepsilon)$, 其中 $\eta_0(z)=0$, $ctx_\varepsilon(y_{p_1})=\varepsilon$.
- τ 为从 q_0 到 q_1 的迁移.
- $c_1=(q_1,\eta_1,ctx_1)$, 其中 $\eta_1(z)=2$, $ctx_1(y_{p_1})=e(1,1,2)$.

从而, $\mathcal{T}$ 接受 $e(1,1,2)$, 且产生输出 $(y:e(1,1,2))$.

$\mathcal{T}$ 在数据流 $e(1,1,2)$, $e(2,2,5)$, $e(3,1,6)$ 上不存在一个可接受运行, 其原因如下: $\mathcal{T}$ 在读事件 $e(1,1,2)$ 的时候必定走从 q_0 到 q_1 的迁移, 而在状态 q_1 下 $\mathcal{T}$ 无法处理 $e(2,2,5)$, 原因在于从 q_1 出发的两个迁移的条件要求 $cur_{name}\neq 2$ 和 $cur_{name}=1$, 而事件 $e(2,2,5)$ 中 *name* 属性值为 2. 所以, $\mathcal{T}$ 在数据流 $e(1,1,2)$ $e(2,2,5)$ $e(3,1,6)$ 上无法产生输出.

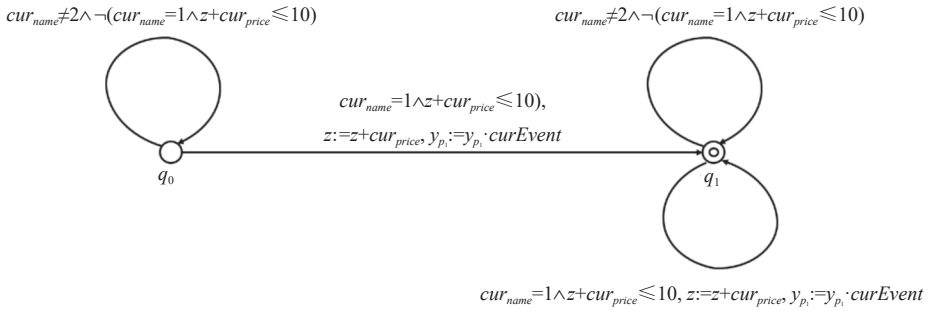


图 1 例 1 中查询对应的数据流转换器

6 FlinkCEP 形式语义

在本节, 给定一个 FlinkCEP 查询 $(patseq)_{SS}$, 我们首先从 $patseq$ 出发构造数据流转换器 $\mathcal{T}_{patseq}$, 并基于 $\mathcal{T}_{patseq}$ 来定义 $(patseq)_{SS}$ 的语义.

6.1 数据流转换器 $\mathcal{T}_{patseq}$ 的构造

下面我们给出 $\mathcal{T}_{patseq}$ 的具体构造过程, 其主要思想是针对 $patseq$ 的子查询 $patseq'$, 递归构造数据流转换器 $\mathcal{T}_{patseq'}$.

为了方便构造, 我们假设 $patseq$ 出现的折叠项中的变量 $\vec{z}$ 没有重复出现, 即对于两个不同的折叠项 $fold_{p_1:e_1:\vec{z}_1=\overrightarrow{fexp_1}}\left(\overrightarrow{v_{0,1}}\right)[i_1]$ 和 $fold_{p_2:e_2:\vec{z}_2=\overrightarrow{fexp_2}}\left(\overrightarrow{v_{0,2}}\right)[i_2]$, 我们有 $\vec{z}_1\cap\vec{z}_2=\varnothing$.

我们用 PN_{patseq} 表示出现在 $patseq$ 中的模式名字的集合, 而用 FV_{patseq} 表示出现在折叠项中的变量 $\vec{z}$ 的有限集合. 而且, 我们通过定义函数 $FEXP_{patseq}$、$INIT_{patseq}$、和 PEV_{patseq} 把 FV_{patseq} 中的变量和它在折叠项中对应的表达式、初始值和模式与事件名字联系起来, 具体来讲, 对于折叠项 $fold_{p:e:\vec{z}=\overrightarrow{fexp}}\left(\overrightarrow{v_0}\right)[i]$, 我们有 $FEXP_{patseq}(z_j)=fexp_j$、$INIT_{patseq}(z_j)=v_{0,j}$、$PEV_{patseq}(z_j)=p:e$, 对于所有的 $j\in[|\vec{z}|]$.

另外, 我们用 Id_X 和 Id_Y 表示 X 和 Y 上的恒等函数, 即对于所有的 $x\in X$ 和 $y\in Y$, $Id_X(x)=x$ 和 $Id_Y(y)=y$.

在递归构造 $\mathcal{T}_{patseq'}$ 过程中, 我们假定所有的数据流转换器的事件种类的集合都是 Σ, 整数变量集合 X 总是等于 FV_{patseq}, 数据流变量集合 Y 总是等于 $\{y_p\mid p\in PN_{patseq}\}$, 且变量的初始赋值 $\eta_0=INIT_{patseq}$, 而且在构造中为了避免重复, 我们省略掉它们的具体定义. 而且, 在构造过程中, 对于条件 $cndt$, 我们使用 ψ_{cndt} 表示从 $cndt$ 中将所有折叠项 $fold_{p':e':\vec{z}=\overrightarrow{fexp}}\left(\overrightarrow{v_0}\right)[i]$ 替换为 z_i 而得到的公式. 同时, 我们使用 α_p 表示以下函数: 对于所有 $x\in X$, 如果 $PEV_{patseq}(x)=p:e$, 则 $\alpha_p(x)=FEXP_{patseq}(x)$, 否则 $\alpha_p(x)=x$, 用 β_p 表示以下函数: $\beta_p(y_p)=y_p\cdot curEvent$, 且对于所有满足 $p'\neq p$ 的 $p'\in PN_{patseq}$, $\beta_p(y_{p'})=y_{p'}$.

由于 $\mathcal{T}_{patseq'}$ 的完整构造过程比较冗长, 下面我们以 $patseq'\equiv p:e:[cndt]_\theta\{n,\infty\}$、$patseq'\equiv p:e:[cndt]_\theta(n,\infty)U[cndt']$、$patseq'\equiv(patseq'_1)\{n,\infty\}$、$patseq'\equiv(patseq'_1)\{n,\infty\}\ U\ cndt'$ 这 4 种典型情况为例来说明 $\mathcal{T}_{patseq'}$ 的构造, 而将完整的构造过程放在附录 A 中.

- 如果 $patseq'\equiv p:e:[cndt]_\theta\{n,\infty\}$, 则 $\mathcal{T}_{patseq'}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 这里:
  - $Q=\{q_i\mid 0\leqslant i\leqslant n\}\cup\{q'_n\}$.
  - δ 定义如下.
    - 如果 $\theta=\cdot$, 则对于所有 $0\leqslant i<n$, 我们有 $\delta(q_i,e,\psi_{cndt})=((q_{i+1},\alpha_p,\beta_p))$, 且 $\delta(q_n,e,\psi_{cndt})=((q_n,\alpha_p,\beta_p))$.
    - 如果 $\theta=\circ$, 则:
      - 如果 $n=0$, 则:
        - $\delta(q_0,e,\psi_{cndt})=((q_0,\alpha_p,\beta_p))$.
        - $\delta(q_0,e,\neg\psi_{cndt})=((q'_0,Id_X,Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_0, e', true) = ((q_0', Id_X, Id_Y))$.

■ $\delta(q_0', e, \neg\psi_{cndt}) = ((q_0', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_0', e', true) = ((q_0', Id_X, Id_Y))$.

■ $\delta(q_0', e, \psi_{cndt}) = ((q_0, \alpha_p, \beta_p))$.

● 如果 $n > 0$, 则 $\delta(q_0, e, \psi_{cndt}) = ((q_1, \alpha_p, \beta_p))$, 且对于所有 $1 \leqslant i < n$, 我们有 $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p))$, $\delta(q_i, e, \neg\psi_{cndt}) = ((q_i, Id_X, Id_Y))$, 对于所有 $e' \neq e$, $\delta(q_i, e', true) = ((q_i, Id_X, Id_Y))$, 另外:

■ $\delta(q_n, e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p))$.

■ $\delta(q_n, e, \neg\psi_{cndt}) = ((q_n', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_n, e', true) = ((q_n', Id_X, Id_Y))$.

■ $\delta(q_n', e, \neg\psi_{cndt}) = ((q_n', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_n', e', true) = ((q_n', Id_X, Id_Y))$.

■ $\delta(q_n', e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p))$.

◆ 如果 $\theta = \odot$, 则:

● 如果 $n = 0$, 则:

■ $\delta(q_0, e, \psi_{cndt}) = ((q_0, \alpha_p, \beta_p), (q_0', Id_X, Id_Y))$.

■ $\delta(q_0, e, \neg\psi_{cndt}) = ((q_0', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_0, e', true) = ((q_0', Id_X, Id_Y))$.

■ $\delta(q_0', e, \neg\psi_{cndt}) = ((q_0', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_0', e', true) = ((q_0', Id_X, Id_Y))$.

■ $\delta(q_0', e, \psi_{cndt}) = ((q_0, \alpha_p, \beta_p), (q_0', Id_X, Id_Y))$.

● 如果 $n > 0$, 则 $\delta(q_0, e, \psi_{cndt}) = ((q_0, \alpha_p, \beta_p))$, 对于所有 $1 \leqslant i < n$, $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p), (q_i, Id_X, Id_Y))$, $\delta(q_i, e, \neg\psi_{cndt}) = ((q_i, Id_X, Id_Y))$, 而且对于所有 $e' \neq e$, $\delta(q_i, e', true) = ((q_i, Id_X, Id_Y))$, 另外:

■ $\delta(q_n, e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p), (q_n', Id_X, Id_Y))$.

■ $\delta(q_n, e, \neg\psi_{cndt}) = ((q_n', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_n, e', true) = ((q_n', Id_X, Id_Y))$.

■ $\delta(q_n', e, \neg\psi_{cndt}) = ((q_n', Id_X, Id_Y))$,

■ 对于所有 $e' \neq e$, $\delta(q_n', e', true) = ((q_n', Id_X, Id_Y))$.

■ $\delta(q_n', e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p), (q_n', Id_X, Id_Y))$.

■ ξ 的定义域为空.

■ $F = \{q_n\}$.

● 如果 $patseq' \equiv p : e : [cndt]_\theta(n, \infty)\ U[cndt']$, 令 $\mathcal{T}_{patseq''} = (\Sigma, X, Y, Q', \delta', \xi', q_0', \eta_0, F')$, 这里 $patseq'' = p : e : [cndt]_\theta(n, \infty)$, 则 $\mathcal{T}_{patseq'}$ 从 $\mathcal{T}_{patseq''}$ 中通过将 δ' 替换为如下的 δ 而得到:

对于所有的 $(q, e', \psi) \in dom(\delta')$, 设 $\delta'(q, e', \psi) = ((q_1, \alpha_1, \beta_1), \ldots, (q_k, \alpha_k, \beta_k))$, 则:

$$\delta(q, e', \psi \wedge \neg\psi_{cndt'}) = ((q_1, \alpha_1, \beta_1), \ldots, (q_k, \alpha_k, \beta_k)).$$

● 如果 $patseq' \equiv (patseq_1')\{n, \infty\}$, 令 $\mathcal{T}_{patseq'_1} = (\Sigma, X, Y, Q_1, \delta_1, \xi_1, q_{1,0}, \eta_0, F_1)$, 则 $\mathcal{T}_{patseq'} = (\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 这里:

■ $Q = Q_1 \times [n]$.

■ δ 定义如下.

对于所有的 $(q, e, \psi) \in dom(\delta_1)$, 令 $\delta_1(q, e, \psi) = ((q_1, \alpha_1, \beta_1), \ldots, (q_k, \alpha_k, \beta_k))$, 则对于所有 $i \in [n]$, $\delta((q, i), e, \psi) = (((q_1, i), \alpha_1, \beta_1), \ldots, ((q_k, i), \alpha_k, \beta_k))$.

■ ξ 定义如下.

◆ 对于所有 $q \in Q_1 \setminus F_1$ 和 $i \in [n]$, $\xi((q,i))$ 从 $\xi_1(q)$ 中通过将所有的状态 q' 替换为 (q',i) 而得到.

◆ 对于所有 $q \in F_1$ 和 $i \in [n-1]$.

- 如果 $q \notin dom(\xi_1)$, 则 $\xi((q,i)) = ((q_{1,0}, i+1))$.
- 否则, 令 $\xi_1(q) = (q_1, \ldots, q_k)$, 则 $\xi((q,i)) = ((q_1,i), \ldots, (q_k,i), (q_{1,0}, i+1))$.

◆ 对于所有 $q \in F_1$.

- 如果 $q \notin dom(\xi_1)$, 则 $\xi((q,n)) = ((q_{1,0}, n))$.
- 否则, 令 $\xi_1(q) = (q_1, \ldots, q_k)$, 则 $\xi((q,n)) = ((q_1,n), \ldots, (q_k,n), (q_{1,0}, n))$.

■ $q_0 = (q_{1,0}, 1)$.

■ $F = F_1 \times \{n\}$.

● 如果 $patseq' \equiv (patseq'_1)\{n,\infty\}\ U\ cndt'$, 令 $\mathcal{T}_{patseq''} = (\Sigma, X, Y, Q', \delta', \xi', q'_0, \eta_0, F')$, 这里 $patseq'' \equiv (patseq'_1)\{n,\infty\}$, 则 $\mathcal{T}_{patseq'}$ 从 $\mathcal{T}_{patseq''}$ 中通过将 δ' 替换为如下的 δ 而得到. 对于所有的 $(q,e',\psi) \in dom(\delta')$:

设 $\delta'(q,e',\psi) = ((q_1,\alpha_1,\beta_1), \ldots, (q_k,\alpha_k,\beta_k))$, 则 $\delta(q,e',\psi \wedge \neg cndt') = ((q_1,\alpha_1,\beta_1), \ldots, (q_k,\alpha_k,\beta_k))$.

例 3: FlinkCEP 查询.

$$\left(p_1 : e : \left[cur_{name} = 1 \wedge \left(fold_{p_1:e:z=z+cur_{price}}(0)\right)[1] + cur_{price} \leqslant 10\right]_{\circ} \{1,\infty\}\, U [cur_{name} = 2]\right)_{NoSkip}.$$

对应的数据流转换器 $\mathcal{T}_{patseq}$ 如图 2 所示. 图 2 中的 $\mathcal{T}_{patseq}$ 根据从 FlinkCEP 到数据流转换器的转换算法得到, 其和图 1 中的数据流转换器语义是等价的.

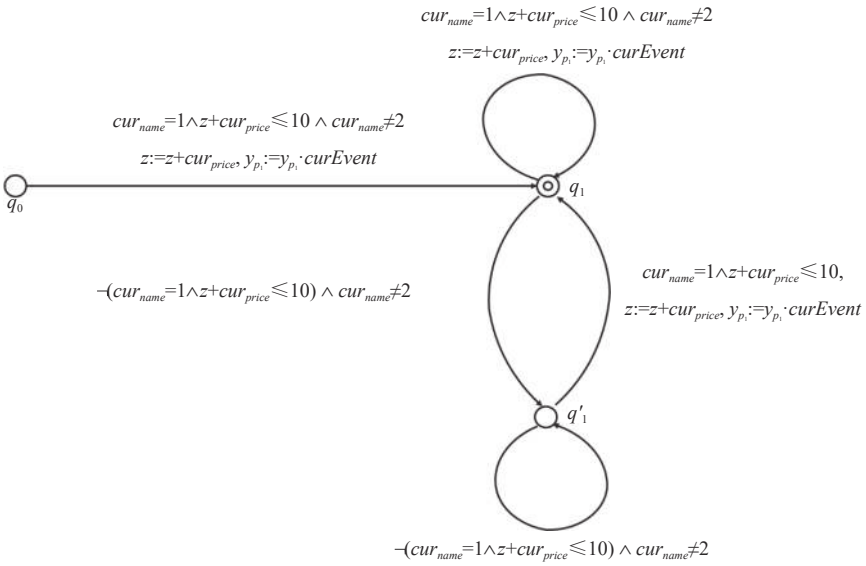


图 2 $\mathcal{T}_{patseq}$

6.2 FlinkCEP 查询的语义

给定一个有限事件数据流 $s = e_1(\vec{v_1}) \ldots e_n(\vec{v_n})\ (n \geqslant 1)$ 和 $1 \leqslant i \leqslant j \leqslant n$, 令 $s[i,j] = e_i(\vec{v_i}) \ldots e_j(\vec{v_j})$.

我们首先考虑 $SS = NoSkip$ 的情况. 给定一个数据流 $S = e_1(\vec{v_1}) \ldots e_n(\vec{v_n})\ (n \geqslant 1)$ 和模式序列 $(patseq)_{NoSkip}$, $(patseq)_{NoSkip}$ 在 S 上的查询结果, 记为 $[\![(patseq)_{NoSkip}]\!]_s$, 定义为通过执行如下算法而得到的 OUT.

(1) 初始, 令 $OUT := \varepsilon$, $j := 1$.

(2) 执行如下循环, 直至 $j = n+1$.

1) 令 $i := 1$, 执行如下循环, 直至 $i = j+1$.

i) 如果$\mathcal{R}_{\mathcal{T}_{patseq}}(s[i,j])\neq\varnothing$, 将$\mathcal{R}_{\mathcal{T}_{patseq}}(s[i,j])$中的运行按照优先级从高到低排列, 记为$\rho_{i,j}^{1},\ldots,\rho_{i,j}^{m_{i,j}}$, 则:

● 如果$OUT=\varepsilon$, 则$OUT:=Out(\rho_{i,j}^{1});\ldots;Out(\rho_{i,j}^{m_{i,j}})$.

● 否则, $OUT:=OUT;Out(\rho_{i,j}^{1});\ldots;Out(\rho_{i,j}^{m_{i,j}})$.

ii) $i:=i+1$.

2) $j:=j+1$.

直观来讲, $\left[\!\left[(patseq)_{NoSkip}\right]\!\right]_s$通过把$S$的子数据流上的所有可接受运行的输出按照如下顺序连接而得到: 首先子数据流按照结束位置从小到大进行排列, 而且, 结束位置一样的子数据流按照开始位置从小到大进行排列, 最后, 同一个子数据流上的可接受运行按照运行的优先级从高到低进行排列.

然后, 我们考虑$SS=SkipToNext$. 给定一个数据流$S=e_1\left(\overrightarrow{v_1}\right)\ldots e_n\left(\overrightarrow{v_n}\right)$ ($n\geqslant 1$) 和模式序列$(patseq)_{SkipToNext}$, $(patseq)_{SkipToNext}$在S上的查询结果, 记为$\left[\!\left[(patseq)_{SkipToNext}\right]\!\right]_s$, 定义为通过执行如下算法而得到的$OUT$.

(1) 初始, 令$OUT:=\varepsilon$, $j:=1$, $skipIdx:=\varnothing$.

(2) 执行如下循环, 直至$j=n+1$.

1) 令$i:=1$, 执行如下循环, 直至$i=j+1$.

i. 如果$i\notin skipIdx$且$\mathcal{R}_{\mathcal{T}_{patseq}}(s[i,j])\neq\varnothing$, 则令$\rho_{i,j}$为$\mathcal{R}_{\mathcal{T}_{patseq}}(s[i,j])$中的优先级最高的运行.

● 如果$OUT=\varepsilon$, 则$OUT:=Out(\rho_{i,j})$.

● 否则, $OUT:=OUT;Out(\rho_{i,j})$.

而且, 令$skipIdx:=skipIdx\cup\{i\}$.

ii. $i:=i+1$.

2) $j:=j+1$.

直观来讲, $SS=SkipToNext$时的匹配输出顺序与$SS=NoSkip$一样, 只是对于每一个开始位置i, 最多输出一个匹配, 而变量$skipIdx$用于存储已经产生过匹配的子数据流开始位置集合.

最后,我们考虑$SS=SkipPastLastEvent$.给定一个数据流$S=e_1\left(\overrightarrow{v_1}\right)\ldots e_n\left(\overrightarrow{v_n}\right)(n\geqslant 1)$和模式序列$(patseq)_{SkipPastLastEvent}$, $(patseq)_{SkipPastLastEvent}$在$S$上的查询结果, 记为$\left[\!\left[(patseq)_{SkipPastLastEvent}\right]\!\right]_s$, 定义为通过执行如下算法而得到的$OUT$.

(1) 初始, 令$OUT:=\varepsilon$, $j:=1$, $lastMatchIdx:=0$.

(2) 执行如下循环, 直至$j=n+1$.

1) 初始, 令$i:=lastMatchIdx+1$.

2) 如果$i\leqslant j$, 则执行如下循环, 直至$i=j+1$.

● 如果$\mathcal{R}_{\mathcal{T}_{patseq}}(s[i,j])\neq\varnothing$, 则令$\rho_{i,j}$为$\mathcal{R}_{\mathcal{T}_{patseq}}(s[i,j])$中的优先级最高的运行.

■ 如果$OUT=\varepsilon$, 则$OUT:=Out(\rho_{i,j})$.

■ 否则$OUT:=OUT;Out(\rho_{i,j})$.

而且, 令$i:=j+1$、$lastMatchIdx:=j$.

● 否则, 令$i:=i+1$.

3) $j:=j+1$.

直观来讲, $SS=SkipPastLastEvent$时的匹配输出顺序与$SS=NoSkip$一样, 只是对于相邻两次匹配输出, 后一个匹配输出的开始位置须在前一个匹配输出的最后一个位置之后, 而变量$lastMatchIdx$用于存储最近一次匹配的结束位置.

7 基于 FlinkCEP 形式语义的数据流查询求值算法

基于 FlinkCEP 的形式语义, 我们知道对于给定查询$(patseq)_{SS}$在给定数据流$s=e_1\left(\overrightarrow{v_1}\right)\ldots e_n\left(\overrightarrow{v_n}\right)$上的最终输出, 但是从 FlinkCEP 的形式语义并不能直接得出在数据流上的在线求值算法, 即求值所需的计算应该在从前往后逐一读取事件中逐步完成, 而且数据流上的每个事件只能读取一次.

下面, 我们基于形式语义设计 FlinkCEP 的数据流求值算法, 算法主要思想是在从前往后逐一读取 s 中的事件的过程中, 在一个队列 C 中记录 $\mathcal{T}_{patseq}$ 从当前位置之前的所有位置开始的所有部分运行的当前格局 (以及开始位置), 而且这些运行按照开始位置从小到大、优先级从高到低进行排列, 求值算法在读取事件时按照 $\mathcal{T}_{patseq}$ 的迁移更新队列中的所有运行的当前格局, 同时根据筛选策略决定是否删除某些运行.

给定 FlinkCEP 查询 $(patseq)_{SS}$, 数据流 $S=e_1\left(\overrightarrow{v_1}\right)\ldots e_n\left(\overrightarrow{v_n}\right)$, 我们执行如下求值算法来在线计算 $[\![(patseq)_{SS}]\!]_s$. 设 $\mathcal{T}_{patseq}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$. 我们将用队列 C 来存储 $\mathcal{T}_{patseq}$ 的所有运行的当前格局, 我们用假设存在如下过程对队列进行操作.

- $getFirst(C)$: 如果 C 非空, 则返回并删除 C 的第 1 个元素, 否则返回 $\perp$.
- $getLast(C)$: 如果 C 非空, 则返回并删除 C 的最后一个元素, 否则返回 $\perp$.
- $putFirst(C,(i,(q,\eta,ctx)))$: 将元素 $(i,(q,\eta,ctx))$ 插入到队列 C 的最前面.
- $putLast(C,(i,(q,\eta,ctx)))$: 将元素 $(i,(q,\eta,ctx))$ 插入到队列 C 的最后面.
- $Remove(C,C)$: 从 C 中将 C 中所有元素去掉.
- $Reset(C)$: 将 C 清空.

而且, 在求值算法中, 对于 $q\in Q$, 我们需要计算即从 q 仅使用 ε-迁移能够到达的状态序列 (按照运行的优先级来排列), 记为 $Closure_{\varepsilon}(q)$. 我们使用如下递归算法 $EPSCLS(\vec{q},P)$来计算 $Closure_{\varepsilon}(q)$, 即 $Closure_{\varepsilon}(q):=EPSCLS((q),\varnothing)$.

在算法 $EPSCLS(\vec{q},P)$ 中, 我们将用到状态序列的去重操作 $RMRPT$, 即对于 $(q'_1,\ldots,q'_k)\in Q^{\oplus}$, 我们使用如下算法计算 $RMRPT(q'_1,\ldots,q'_k)$.

(1) 令 $i:=1$, $S=()$, 执行如下循环直至 $i=k+1$: 如果 $q'_i\in S$, 则 $i:=i+1$, 否则 $S:=S\cdot(q'_i)$ (这里 $\cdot$ 表示序列连接操作) 且 $i:=i+1$.

(2) 令 $RMRPT(q'_1,\ldots,q'_k):=S$.

而且算法 $EPSCLS(\vec{q},P)$ 用到了状态序列的状态删除操作 $RMST((q'_1,\ldots,q'_k),P)$. 我们使用如下算法计算 $RMST((q'_1,\ldots,q'_k),P)$.

1) 令 $i:=1$, $S=()$, 执行如下循环直至 $i=k+1$: 如果 $q'_i\in P$, 则 $i:=i+1$, 否则 $S:=S\cdot(q'_i)$ (这里 $\cdot$ 表示序列连接操作) 且 $i:=i+1$.

2) 令 $RMST((q'_1,\ldots,q'_k),P)=S$.

$EPSCLS(\vec{q},P)$ 的伪代码如算法 1.

算法 1. $EPSCLS(\vec{q},P)$ 的伪代码.

输入: 状态序列 $\vec{q}$ 和状态子集 P, 满足 $\vec{q}$ 中的不同位置的状态互不相同, 且 $P\cap\vec{q}=\varnothing$.

1. 如果 $|\vec{q}|=0$, 则 $EPSCLS((q_1,\ldots,q_n),P)=()$. 否则, 令 $\vec{q}=(q_1,\ldots,q_n)$, 进入步骤 2.
2. 如果 $q_1\notin dom(\xi)$ 且 $n=1$, 则 $EPSCLS(\vec{q},P)=(q_1)$.
3. 否则, 如果 $q_1\notin dom(\xi)$ 且 $n>1$, 则:

$$EPSCLS(\vec{q},P)=(q_1)\cdot EPSCLS((q_2,\ldots,q_n),P\cup\{q_1\}).$$

4. 否则, 我们有 $q_1\in dom(\xi)$ 且 $n\geqslant 1$. 令 $\xi(q_1)=(q'_1,\ldots,q'_m)$, 则:

$$EPSCLS(\vec{q},P)=(q_1)\cdot EPSCLS(RMRPT(RMST(q'_1,\ldots,q'_m,P\cup\{q_1\})\cdot(q_2,\ldots,q_n)),P\cup\{q_1\}).$$

注意, 算法 $EPSCLS(q_1,\ldots,q_n)$ 一定会终止, 因为每次递归调用所需要排除的状态子集在增长.

下面给出求值算法的具体描述, 如算法 2.

算法 2. 求值算法.

输入: $(patseq)_{SS}$、 $S=e_1\left(\overrightarrow{v_1}\right)\ldots e_n\left(\overrightarrow{v_n}\right)$;

输出: $[\![(patseq)_{SS}]\!]_s$.

1. 令$C:=\varepsilon$、 $j:=1$.

2. 执行下面的循环, 直至$j=n+1$.

1) 令$C:=putLast(C,\ (j,(q_0,\eta_0,ctx_\varepsilon)))$, $C':=\varepsilon$. //初始化

2) 执行以下循环, 直至$C=\varepsilon$. //读取$e_j(\overrightarrow{v_j})$, 根据迁移更新$C$中的格局, 存入$C'$中.

a) 令$(i,(q,\eta,ctx))=getFirst(C)$.

b) 根据$\mathcal{T}_{patseq}$的假设, 存在最多一个ψ使得$[\![\psi]\!]_{\eta[\overrightarrow{v_j}/\overrightarrow{cur}]}$为真, 且$(q,e_j,\psi)\in dom(\delta)$. 如果存在满足上述条件的$\psi$, 则设$\delta(q,e_j,\psi)=((q_1,\alpha_1,\beta_1),\ldots,(q_k,\alpha_k,\beta_k))$, 令$C'$为执行以下操作序列而得到的队列:

$$putLast(C',(i,(q_1,\eta_1,ctx_1))),\ldots,putLast(C',(i,(q_k,\eta_k,ctx_k))),$$

这里对于任意$r\in[k]$, 对于每一个$x\in X$, $\eta_r(x)=[\![\alpha_r(x)]\!]_{\eta[\overrightarrow{v_j}/\overrightarrow{cur}]}$, 对每一个$y\in Y$, $ctx_r(y)=[\![\beta_r(y)]\!]_{ctx[e_j(\overrightarrow{v_j})/curEvent]}$.

c) 设$Closure_\varepsilon(q)=(q'_1,\ldots,q'_r)$, 则令$r':=1$, 执行如下循环直至$r'=r+1$:

如果$q_{r'}\neq q$且$q_{r'}\in F$, 则输出$(y:ctx(y))_{y\in Y}$. 如果存在 (唯一)$\psi_{r'}$使得$[\![\psi_{r'}]\!]_{\eta[\overrightarrow{v_j}/\overrightarrow{cur}]}$为真, 且$(q'_{r'},e_j,\psi_{r'})\in dom(\delta)$, 则设:

$$\delta(q'_{r'},e_j,\psi_{r'})=((q''_1,\alpha_1,\beta_1),\ldots,(q''_k,\alpha_k,\beta_k)).$$

令C'为执行以下操作序列而得到的队列:

$$putLast(C',(i,(q''_1,\eta''_1,ctx''_1))),\ldots,putLast(C',(i,(q''_k,\eta''_k,ctx''_k))),$$

这里对于任意$r''\in[k]$, 对于每一个$x\in X$, $\eta''_{r''}(x)=[\![\alpha_{r''}(x)]\!]_{\eta[\overrightarrow{v_j}/\overrightarrow{cur}]}$, 对每一个$y\in Y$, $ctx''_{r''}(y)=[\![\beta_{r''}(y)]\!]_{ctx[e_j(\overrightarrow{v_j})/curEvent]}$.

3) 从前往后遍历队列C', 进行如下操作: 设$(i',(q',\eta',ctx'))$为遍历的当前元素, 如果$q'\in F$, 则产生输出$(y:ctx'(y))_{y\in Y}$, 而且:

● 如果匹配筛选策略$SS=SkipToNext$, 则令:

$$C':=Remove(C',\{(i,(q,\eta,ctx))\in C'\mid i=i'\}).$$

● 如果匹配筛选策略$SS=SkipPastLastEvent$, 则令$C':=\epsilon$.

4) 令$C:=C'$, $j:=j+1$.

例 4: 考虑查询$(patseq)_{NoSkip}$, 这里$patseq$来自例 3. 则$(patseq)_{NoSkip}$在数据流$e(1,1,2)$, $e(2,2,5)$, $e(3,1,6)$, $e(4,3,2)$, $e(5,1,5)$上的求值过程如下: 在下面, 我们用C_j表示第j轮循环开始时的队列内容C.

● 首先, $C_1=\varepsilon$, 令$C'_1:=\epsilon$, 然后我们将$(1,(q_0,\eta_0,ctx_\varepsilon))$加入$C_1$. 接着, 将$(1,(q_0,\eta_0,ctx_\varepsilon))$从$C_1$中取出, 执行从$q_0$到$q_1$的迁移, 得到$C'_1=(1,(q_1,z=2,y_{p_1}=(e(1,1,2))))$. 由于$q_1\in F$, 我们输出$p_1:e(1,1,2)$.

令$C_2:=C'_1$, $j:=2$.

● 令$C'_2:=\epsilon$, 将$(2,(q_0,\eta_0,ctx_\varepsilon))$加入$C_2$的后面. 接着, 将$(1,(q_1,z=2,y_{p_1}=(e(1,1,2))))$从$C_2$中取出, 由于第 2 个位置的事件为$e(2,2,5)$, 不满足从$q_1$出发的所有迁移须满足的$cur_{name}\neq 2$的条件, 因此, 这时不会往$C'_2$中加入格局. 然后将$(2,(q_0,\eta_0,ctx_\varepsilon))$从$C_2$中取出, 由于第 2 个位置的事件为$e(2,2,5)$, 从$q_0$出发没有与之匹配的迁移, 因此这时不会往$C'_2$中加入格局. 所以最终$C'_2=\epsilon$. 令$C_3:=C'_2$, $j:=3$.

● 令$C'_3:=\epsilon$, 将$(3,(q_0,\eta_0,ctx_\varepsilon))$加入$C_3$的后面. 接着, 将$(3,(q_0,\eta_0,ctx_\varepsilon))$从$C_3$中取出, 执行从$q_0$到$q_1$的迁移, 得到$(1,(q_1,z=6,y_{p_1}=(e(3,1,6))))$, 将其加入$C'_3$. 由于$q_1\in F$, 我们输出$p_1:e(3,1,6)$. 最后, 我们有$C'_3=(1,(q_1,z=6,y_{p_1}=(e(3,1,6))))$. 令$C_4:=C'_3$, $j:=4$.

● 令$C'_4:=\epsilon$, 将$(4,(q_0,\eta_0,ctx_\varepsilon))$加入$C_4$的后面. 接着, 将$(1,(q_1,z=6,y_{p_1}=(e(3,1,6))))$从$C_4$中取出, 由于第 4 个位置的事件为$e(4,3,2)$, 执行从$q_1$到$q'_1$的迁移, 得到$(1,(q'_1,z=6,y_{p_1}=(e(3,1,6))))$, 将其加入$C'_4$. 然后将$(4,(q_0,\eta_0,ctx_\varepsilon))$从$C_4$中取出, 由于第 4 个位置的事件为$e(4,3,2)$, 从$q_0$到$q_1$的迁移无法执行, 因此这时不会往$C'_4$中添加格局. 令$C_5:=C'_4$, $j:=5$.

● 令$C'_5:=\epsilon$, 将$(5,(q_0,\eta_0,ctx_\varepsilon))$加入$C_5$的后面. 接着, 将$(1,(q'_1,z=6,y_{p_1}=(e(3,1,6))))$从$C_5$中取出, 由于第 5 个位置的事件为$e(5,1,5)$, 条件$z+cur_{price}=6+5\leqslant 10$为假, 因此, 这时从$q'_1$到$q'_1$的迁移可以执行, 得到$(1,(q'_1,z=6,y_{p_1}=(e(3,1,6))))$, 将其加入$C'_5$. 然后将$(5,(q_0,\eta_0,ctx_\varepsilon))$从$C_5$中取出, 由于第 5 个位置的事件为$e(5,1,5)$, 从$q_0$到$q_1$

的迁移可以执行, 得到$(1,(q_1,z=5,y_{p_1}=(e(5,1,5))))$, 将其加入$C'_5$的后面. 由于$q_1\in F$, 我们输出$p_1:e(5,1,5)$. 令$C_6:=C'_5$, $j:=6$. 求值过程终止.

8 FlinkCEP 形式语义与 FlinkCEP 平台实际语义对比实验

为了检验本文提出的形式语义是否和 FlinkCEP 平台是否一致, 我们使用 Python 语言实现了上述 FlinkCEP 查询求值算法, 得到了 FlinkCEP 查询求值系统, 并通过全面覆盖 FlinkCEP 各种语法特性的 4 410 个查询进行了实验. 我们的实现完全开源, 读者可以访问 https://github.com/Abreto/reflinkcep 来了解更多细节.

下面, 我们首先对求值算法的实现进行介绍, 然后对对比实验的测试用例描述, 最后对对比实验结果进行分析总结.

8.1 查询求值算法实现

FlinkCEP 查询求值系统由 6 个模块组成, 分别为: 公共定义、查询抽象语法树定义、查询编译器、数据流转换器、CEP 单步执行器、CEP 流处理算子. 整个系统的整体运行流程为: 给定查询后, 查询编译器会翻译出对应的数据流转换器, 并提取出对应的匹配筛选策略, 生成 CEP 单步执行器, 然后生成 CEP 流处理算子, 最后由该算子接受输入事件流并产生输出结果.

公共定义部分定义了事件、数据流、匹配、匹配输出结果的数据结构.

为了便于实现, 该系统以抽象语法树的方式输入查询. 抽象语法树的顶层是 query 节点, 包含 patseq 和 context 两个成员. context 成员是一个 dict, 记录了全局事件类型、匹配筛选策略等信息. 为了简化实现代码, patseq 成员也是一个特定结构的 dict, 一共有 7 种节点, 分别对应单例模式、循环模式、无限循环模式、组合模式、组模式、组循环模式、组无限循环模式.

给定一个查询后, 系统会生成一个 CEP 流处理算子 (CEPOperator). 该算子的构造过程如下.

(1) 首先, 将查询根据上文定义的规则翻译为数据流转换器并提取出匹配筛选策略.

(2) 其次, 用数据流转换器和匹配筛选策略构造 CEP 单步执行器.

(3) 最后, 使用 CEP 单步执行器来构造 CEP 查询求值算子.

CEP 查询求值算子接受一串输入数据流后, 会将事件逐个喂给 CEP 单步执行器, 同时收集 CEP 单步执行器产生的匹配并输出.

CEP 单步执行器实现了第 7 节中查询求值算法的内层循环. 首先, CEP 单步执行器初始化当前部分匹配列表 (实际上是当前存在的运行的最后一个格局) 为空, 并置已处理事件计数器为 0. 随后, 当 CEP 单步执行器每收到一个事件时, 它会遵循上文执行算法内层循环的逻辑更新部分匹配列表和事件计数器. 更新完毕后按照优先级顺序遍历所有部分匹配, 在遇到接受的匹配后, 应用匹配筛选策略并生成匹配, 最后输出所有匹配结果组成的匹配数据流.

DST 类是数据流转换器的实现, 它通过传入类似上文 数据流转换器的定义中的成员来构造, 并提供了若干接口方便 DST 执行器的执行. 如 initial_configuration() 可以返回该 dst 的初始格局, start_from(q) 可以遍历以 q 为起点的所有转移, find_accepted(conf) 可以寻找从 conf 出发有没有仅通过空转移能到达的接受格局并返回. 在理论定义部分, 数据流转换器使用两个偏函数δ和ξ表示转移和空转移; 在实际实现中, DST 类维护一个 Delta: list[Transition] 成员表示所有转移. Transition 是一个五元组 (q1, pred, q2, alpha, beta), 分别是起始状态、转移条件、目标状态、数据变量更新函数、数据流变量更新函数. pred 由理论定义中事件类型和公式合并而成, 对于空转移, 其事件类型约定为 None. 我们保证同一个状态出发优先级更高的转移对应的 Transition 总是在 Delta 的更前面. 同时所有非空转移对应的 Transition 在空转移前面, 故通过按顺序遍历 Delta 即可得到按优先级考虑转移的效果. 除此之外, 理论定义中的 F 被实现为每个状态有一个输出函数, 如果该函数不为 None, 则表示其属于 F. 所有接受状态使用同一个输出函数的实现, 其实现细节和理论定义部分输出形式对应.

编译器主体是根据 CEP 查询的 patseq AST 构造对应的 Sigma, X, Y, Q, q0, eta, Delta 等成员并传入 DST 类构

建数据流转换器实例的过程. 如上文翻译算法所述, 编译器会将大的 AST 拆分成小的 AST 递归地翻译小数据流转换器并合成整体的数据流转换器.

8.2 对比实验测试用例

为了验证我们定义的形式语义是否与 FlinkCEP 官方实现的语义是否一致, 我们设计并大规模生成了 4 组能够较为全面覆盖 FlinkCEP 的不同语法特性的 FlinkCEP 查询作为测试用例集. 第 1 组测试用例集主要考虑单例模式和循环模式通过不同连接操作的组合, 而且在单例模式中同时考虑了不含折叠项和含有折叠项的条件, 第 2 组测试用例集在第 1 组的基础上添加了量词, 第 3 组测试用例集考虑循环模式和量词的嵌套, 第 4 组测试用例集考虑嵌套的组模式. 下面对这四组测试用例集的生成进行详细说明.

在生成的查询中, 我们考虑电子商务中的购买事件 e, 它具有 3 个属性, 即 $(id, name, price)$, 其中 id 表示商品的标识, $name$ 表示商品的名字, $price$ 表示商品的价格.

在生成的查询中, 我们考虑形如 $p:e:[name=n]$ 和 $\left(fold_{p:e:z=z+cur_{price}}(0)\right)[1]\leqslant n$ 的单例模式, 而且我们考虑形如 $U\,[name=n]$ 的终止条件.

第 1 组 FlinkCEP 查询 (记为 NOGP) 为 $spat\ \theta\ lpat$ 的形式, 其中 $spat$ 的条件固定为一个简单条件, $\theta\in\{\cdot,\circ,\odot\}$, $lpat$ 的条件在一个简单条件或迭代条件中选取, 其循环的修饰量词在 $\{0,3\},\{1,3\},\{3,3\},\{0,\infty\},\{1,\infty\}$ 中选取, 同时 $\{0,\infty\},\{1,\infty\}$ 有添加和不添加 $U[cndt]$ 两个版本. 最后, 在此基础之上, 针对 3 个匹配筛选策略分别生成对应的查询. 第 1 组共生成了 378 个查询.

第 2 组 FlinkCEP 查询 (记为 SLGP) 在第 1 组查询的基础上, 添加了分组模式, 其形式为 $(spat\ \theta\ lpat)quan$. 分组模式的子模式 $spat\ \theta\ lpat$ 的生成规则与第 1 组相同. $quan$ 有 8 种选择, 分别是空 (对应于形如 $(patseq)$ 的分组模式), 以及 $\{0,3\},\{1,3\},\{3,3\}$ 中的一种, 以及 $\{0,\infty\},\{1,\infty\}$ 中的一种再添加或不添加 $U[cndt]$. 第 2 组共生成了 3 024 个查询.

第 3 组 FlinkCEP 查询 (记为 LGP) 形如 $(lpat)quan$, 其中 $lpat$ 的生成与第 1 组查询对应部分相同, $quan$ 的选取与第 2 组查询对应部分相同, 共生成了 1 008 个查询.

第 4 组 FlinkCEP 查询 (记为 SLGGP) 形如 $((spat\ \theta\ lpat)quan_1)quan_2$, 其中 $(spat\ \theta\ lpat)quan_1$ 的生成与第 2 组查询对应部分相同. $quan_2$ 有 3 种选择, 分别是空或者 $\{0,3\},\{1,\infty\}$ 中的一种. 第 4 组共生成了 9 072 个查询.

由于 FlinkCEP 官方实现中对分组查询的子模式序列的第 1 个模式存在特殊的处理使得语义和直观认识不一致, 故我们分别构造了第 2 组和第 3 组两组包含分组模式的查询, 用以验证我们语义的一致性和说明其与 FlinkCEP 的区别. 第 2 组查询中分组模式的子模式序列以 $spat$ 起始, FlinkCEP 官方实现引入的差异对 $spat$ 没有影响, 我们通过这组测试用例说明我们定义的分组模式以及分组模式中的 $lpat$ 和 FlinkCEP 官方实现一致. 第 3 组查询中分组模式的子模式序列仅包含一个 $lpat$, 用以和第 2 组实验对比说明 FlinkCEP 的官方实现在分组模式子模式序列的第 1 个模式的处理上有一些反直觉的结果. 第 4 组用以和第 2 组实验对比说明 FlinkCEP 的官方实现在嵌套组模式的处理上亦存在问题.

最后我们对于所有的 FlinkCEP 查询, 生成了对应的用于在 FlinkCEP 平台上运行的 Java 代码. $\{0,\infty\}$ 形式的修饰量词用 $\{1,\infty\}$.optional() 对应的代码实现.

我们固定了以下有限事件数据流:

$$e(1,1,0),\ e(2,2,5),\ e(3,1,0),\ e(4,2,2),\ e(5,1,0),\ e(6,3,2),\ e(7,1,0),\ e(8,2,5),\ e(9,1,8)$$

作为所有测试用例的输入数据流. 将输入数据流和生成的抽象语法树交由 FlinkCEP 查询求值系统进行求值, 得到形式语义查询的求值结果. 同时我们在 FlinkCEP 平台中, 编译并运行生成的 FlinkCEP 查询 (包含输入数据流), 得到 FlinkCEP 官方实现的查询输出.

8.3 对比实验结果

实验结果如表 1 所示.

表 1 FlinkCEP 查询求值系统与 FlinkCEP 官方平台查询输出结果比较

分组	查询数量	查询结果比较		单个查询平均时间 (ms)	
		结果相同	结果不同	FlinkCEP 查询求值系统	FlinkCEP 官方实现
NOGP($spat\ \theta\ lpat$)	378	378	0	1.94	60.69
SLGP(($spat\ \theta\ lpat$)$quan$)	3 024	3 024	0	4.94	147.49
LGP(($lpat$)$quan$)	1 008	563	445	14.02	77.15
SLGGP((($spat\ \theta\ lpat$)q_1)q_2)	9 072	7 483	1 589	11.12	306.51
总计	13 482	11 448	2 034	9.56	241.32

从表 1 可以看出, 对于 NOGP 分组中的 378 个查询和 SLGP 分组中的 3 024 个查询, FlinkCEP 查询求值系统的结果与 FlinkCEP 官方平台查询输出的结果完全一样. 然而, 对于 LGP 分组中的 1 008 个查询, 只有 563 个查询结果是一样的, 而 445 个查询结果是不一样的; 对于 SLGGP 分组中的 9 072 个查询, 有 1 589 个查询结果不一致. 比如对于查询 $(p:e:[cur_{name}=2].\{0,3\})$, FlinkCEP 查询求值系统在输入数据流 $e(1,1,0)$, $e(2,2,5)$, $e(3,1,0)$, $e(4,2,2)$, $e(5,1,0)$, $e(6,3,2)$, $e(7,1,0)$, $e(8,2,5)$, $e(9,1,8)$ 上的结果是 $p:e(2,2,5), p:e(4,2,2), p:e(8,2,5)$, 即 $(p:e:[cur_{name}=2].\{0,3\})$ 有 3 次匹配, 每次都匹配单个事件. 而 FlinkCEP 官方平台查询输出结果为 $p:e(2,2,5), e(4,2,2), e(8,2,5)$.

即 $(p:e:[cur_{name}=2].\{0,3\})$ 仅有一次匹配, 一次匹配了 3 个事件.

与之对比, 不加 () 的循环查询 $p:e:[cur_{name}=2].\{0,3\}$ 在 FlinkCEP 官方平台上的查询结果为:

$$p:e(2,2,5), p:e(4,2,2), p:e(8,2,5).$$

我们认为, $(p:e:[cur_{name}=2].\{0,3\})$ 与 $p:e:[cur_{name}=2].\{0,3\}$ 的查询结果一致是比较合理的, FlinkCEP 这两个查询结果不一致很可能是实现存在错误. 为此, 我们阅读了 FlinkCEP 官方平台的源代码, 发现其对于组模式 ($patseq$)$quan$ 中 $patseq$ 模式序列的第 1 个模式 (设为 ps_0) 进行了特殊处理. 首先, FlinkCEP 官方实现在将 ps_0 翻译为 (有限) 自动机时, 其内部连续性标识不由 ps_0 自己的定义决定, 而是由外层分组模式 ($patseq$)$quan$ 的内部连续性标识决定. 虽然我们将分组模式 $(p:e:[cur_{name}=2].\{0,3\})$ 的内部连续性标识定义为 $\theta=\cdot$, 但 FlinkCEP 的官方实现中默认为 $\theta=\circ$, 即使是不加量词的情况, 即 ($patseq$), 其内部连续性标识也是 $\theta=\circ$ (这很可能是个实现的 bug), 这就导致了上述情况中 FlinkCEP 的官方实现给出了违反连续性标识的结果. 其次, 在 ps_0 含有可选的匹配情况时, FlinkCEP 官方实现并不会翻译这个可选匹配引入的空转移 (也许是因为他认为这个空转移会在外层分组模式的翻译中体现出来, 但是他忽略了诸如 $\{1,3\}$ 的翻译也复用了这一套逻辑), 导致 $\{0,3\}$ 实际被翻译为了 $\{3,3\}$, 于是产生了上述一个匹配输出 3 个事件的结果, 而没有包括其他 3 个匹配, 其中每个匹配分别包含一个事件. 其他使查询结果不一致的原因包括因 FlinkCEP 官方实现对嵌套组模式的处理不佳导致的输出重复结果或死循环崩溃而无结果等.

此外, 我们可以发现 FlinkCEP 官方实现的运行时间远大于 FlinkCEP 求值系统. 这是因为 FlinkCEP 官方实现是作为 Flink 的一个算子实现的, 而 Flink 本身作为一个流处理基础平台, 在集群、通用性等角度比 FlinkCEP 求值系统更加复杂, 引入了额外的开销. 这两者的运行时间没有太多的可比性, 在此列出仅作参考. 图 3 展示了更为详细的效率比较结果, 图中每个点为一个测试用例, 其横坐标为 FlinkCEP 求值系统针对该测试用例的用时, 纵坐标为 FlinkCEP 官方实现针对该测试用例的用时. 图 3 中虚线为直线 $y=x$, 在此直线之上的点表示对于该测试用例 FlinkCEP 求值系统用时少于 FlinkCEP 官方实现用时. 位于横轴下方 (视觉上靠近横轴) 的点表示在该测试用例上 FlinkCEP 官方实现出现了崩溃无输出结果. 可以看出大部分点落于虚线之上, 即针对大部分测试用例, FlinkCEP 求值系统效率比 FlinkCEP 官方实现高.

9 总结与展望

本文针对 Flink 平台上的复杂事件处理语言 FlinkCEP 语义复杂、难以准确理解的问题, 提出了数据流转换器的自动机模型, 以该自动机为基础定义了 FlinkCEP 的形式语义, 获得了对 FlinkCEP 语义的准确理解. 而且, 我们

基于数据流转换器实现了 FlinkCEP 的查询求值算法, 并且通过全面覆盖 FlinkCEP 语法特性的 4 410 个测试用例验证了我们提出的形式语义与 FlinkCEP 在 Flink 平台上的实际语义基本是一致的, 而且对于不一致的情况进行了分析, 指出 FlinkCEP 在 Flink 平台上的实现对于组模式的处理可能存在错误. 而且, 实验结果表明, 数据流转换器不仅有助于我们对于 FlinkCEP 的语义理解, 其还可以作为一种基本的自动机模型用于复杂事件处理语言的高效查询求值.

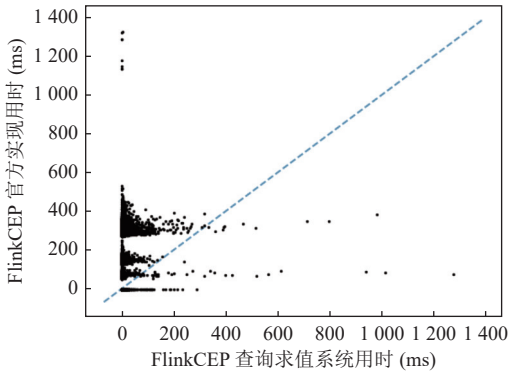


图 3　FlinkCEP 求值系统与 FlinkCEP 官方实现效率的比较

未来工作包括对数据流转换器自动机模型的理论探讨, 比如分析其在查询求值过程中的计算复杂度, 探讨何种查询可以在常数空间 (与数据流长度无关的空间) 内进行查询求值. 另外, 我们目前基于形式语义实现了 FlinkCEP 查询求值算法的原型系统, 其目的只是进行语义比较, 后面可以考虑在 Flink 平台中实现我们提出的查询求值算法, 并与 FlinkCEP 在 Flink 平台中的官方实现进行更准确的查询求值效率的比较.

References:

[1] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the 6th Symp. on Operating System Design and Implementation. San Francisco: USENIX Association, 2004. 137–150.

[2] Chaiken R, Jenkins B, Larson PÅ, Ramsey B, Shakib D, Weaver S, Zhou JR. SCOPE: Easy and efficient parallel processing of massive data sets. Proc. of the VLDB Endowment, 2008, 1(2): 1265–1276. [doi: 10.14778/1454159.1454166]

[3] Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig Latin: A not-so-foreign language for data processing. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. Vancouver: ACM, 2008. 1099–1110. [doi: 10.1145/1376616.1376726]

[4] Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R. Hive: A warehousing solution over a MapReduce framework. Proc. of the VLDB Endowment, 2009, 2(2): 1626–1629. [doi: 10.14778/1687553.1687609]

[5] Zaharia M, Das T, Li HY, Hunter T, Shenker S, Stoica I. Discretized streams: Fault-tolerant streaming computation at scale. In: Proc. of the 24th ACM Symp. on Operating Systems Principles. Farminton: ACM, 2013. 423–438. [doi: 10.1145/2517349.2522737]

[6] Lin W, Fan HC, Qian ZP, Xu JW, Yang S, Zhou JR, Zhou LD. StreamScope: Continuous reliable distributed processing of big data streams. In: Proc. of the 13th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2016. 439–453.

[7] Kulkarni S, Bhagat N, Fu MS, Kedigehalli V, Kellogg C, Mittal S, Patel JM, Ramasamy K, Taneja S. Twitter heron: Stream processing at scale. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. Melbourne: ACM, 2015. 239–250. [doi: 10.1145/2723372.2742788]

[8] Grez A, Riveros C, Ugarte M, Vansummeren S. A formal framework for complex event recognition. ACM Trans. on Database Systems, 2021, 46(4): 16. [doi: 10.1145/3485463]

[9] Grez A, Riveros C. Towards streaming evaluation of queries with correlation in complex event processing. In: Proc. of the 23rd Int'l Conf. on Database Theory. Copenhagen: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. 14:1–14:17.

[10] Grez A, Riveros C, Ugarte M, Vansummeren S. On the expressiveness of languages for complex event recognition. In: Proc. of the 23rd Int'l Conf. on Database Theory. Copenhagen: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. 15:1–15:17.

[11] Agrawal J, Diao YL, Gyllstrom D, Immerman N. Efficient pattern matching over event streams. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. Vancouver: ACM, 2008. 147–160. [doi: 10.1145/1376616.1376634]

[12] Wu E, Diao YL, Rizvi S. High-performance complex event processing over streams. In: Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of Data. Chicago: ACM, 2006. 407–418. [doi: 10.1145/1142473.1142520]

[13] Chen TL, Flores-Lamas A, Hague M, Han ZL, Hu DH, Kan SL, Lin AW, Rümmer P, Wu ZL. Solving string constraints with Regex-dependent functions through transducers with priorities and variables. Proc. of the ACM on Programming Languages, 2022, 6(POPL): 45. [doi: 10.1145/3498707]

附录 A. 数据流转换器$\mathcal{T}_{patseq}$的完整构造过程

下面给出$\mathcal{T}_{patseq}$的具体构造过程, 其主要思想是针对$patseq$的子查询$patseq'$, 递归构造数据流转换器$\mathcal{T}_{patseq'}$.

为了方便构造, 我们假设$patseq$出现的折叠项中的变量$\vec{z}$没有重复出现, 即对于两个不同的折叠项$fold_{p_1:e_1:\vec{z}_1=\overrightarrow{fexp_1}}\left(\overrightarrow{v_{0,1}}\right)[i_1]$和$fold_{p_2:e_2:\vec{z}_2=\overrightarrow{fexp_2}}\left(\overrightarrow{v_{0,2}}\right)[i_2]$, 我们有$\vec{z}_1 \cap \vec{z}_2 = \varnothing$.

我们用PN_{patseq}表示出现在$patseq$中的模式名字的集合, 而用FV_{patseq}表示出现在折叠项中的变量$\vec{z}$的有限集合. 而且, 我们通过定义函数$FEXP_{patseq}$、$INIT_{patseq}$、和PEV_{patseq}把FV_{patseq}中的变量和它在折叠项中对应的表达式、初始值、和模式与事件名字联系起来, 具体来讲, 对于折叠项$fold_{p:e:\vec{z}=\overrightarrow{fexp}}\left(\overrightarrow{v_0}\right)[i]$, 我们有$FEXP_{patseq}(z_j) = fexp_j$、$INIT_{patseq}(z_j) = v_{0,j}$、$PEV_{patseq}(z_j) = p:e$, 对于所有的$j \in [|\vec{z}|]$.

另外, 我们用Id_X和Id_Y表示X和Y上的恒等函数, 即对于所有的$x \in X$和$y \in Y$, $Id_X(x) = x$和$Id_Y(y) = y$.

在递归构造$\mathcal{T}_{patseq'}$过程中, 我们假定所有的数据流转换器的事件种类的集合都是Σ, 整数变量集合X总是等于FV_{patseq}, 数据流变量集合Y总是等于$\{y_p \mid p \in PN_{patseq}\}$, 且变量的初始赋值$\eta_0 = INIT_{patseq}$, 而且在构造中为了避免重复, 我们省略掉它们的具体定义. 而且, 在构造过程中, 对于条件$cndt$, 我们使用ψ_{cndt}表示从$cndt$中将所有折叠项$fold_{p':e':\vec{z}=\overrightarrow{fexp}}\left(\overrightarrow{v_0}\right)[i]$替换为$z_i$而得到的公式. 同时, 我们使用$\alpha_p$表示以下函数: 对于所有$x \in X$, 如果$PEV_{patseq}(x) = p:e$, 则$\alpha_p(x) = FEXP_{patseq}(x)$, 否则$\alpha_p(x) = x$, 用$\beta_p$表示以下函数: $\beta_p(y_p) = y_p \cdot curEvent$, 且对于所有满足$p' \neq p$的$p' \in PN_{patseq}$, $\beta_p(y_{p'}) = y_{p'}$.

- 如果$patseq' \equiv p:e:[cndt]$, 则$\mathcal{T}_{patseq'} = (\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 这里$Q = \{q_0, q_f\}$, $\delta(q_0, e, \psi_{cndt}) = ((q_f, \alpha_p, \beta_p))$, ξ定义域为空, $F = \{q_f\}$.
- 如果$patseq' \equiv p:e:[cndt]_\theta\{n,m\}$, 则$\mathcal{T}_{patseq'} = (\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 这里:
  - $Q = \{q_i \mid 0 \leqslant i \leqslant m\} \cup \{q'_i \mid n \leqslant i < m\}$.
  - δ定义如下.
    - 如果$\theta = \cdot$, 则对于所有$0 \leqslant i < m$, $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p))$.
    - 如果$\theta = \circ$, 则对于所有$0 \leqslant i < m$, $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p))$, 这里:
      - 如果$n = 0$, 则对于所有$1 \leqslant i < m$:
        - $\delta(q_i, e, \neg\psi_{cndt}) = ((q'_i, Id_X, Id_Y))$.
        - 对于所有$e' \neq e$, $\delta(q_i, e', true) = ((q'_i, Id_X, Id_Y))$.
        - $\delta(q'_i, e, \neg\psi_{cndt}) = ((q'_i, Id_X, Id_Y))$.
        - 对于所有$e' \neq e$, $\delta(q'_i, e', true) = ((q'_i, Id_X, Id_Y))$.
        - $\delta(q'_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha, \beta))$.
      - 如果$n > 0$, 则对于所有$1 \leqslant i < n$, $\delta(q_i, e, \neg\psi_{cndt}) = ((q_i, Id_X, Id_Y))$, 对于所有$e' \neq e$, $\delta(q_i, e', true) = ((q_i, Id_X, Id_Y))$, 而且, 对于所有$n \leqslant i < m$, 我们有:
        - $\delta(q_i, e, \neg\psi_{cndt}) = ((q'_i, Id_X, Id_Y))$.
        - 对于所有$e' \neq e$, $\delta(q_i, e', true) = ((q'_i, Id_X, Id_Y))$.
        - $\delta(q'_i, e, \neg\psi_{cndt}) = ((q'_i, Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_i', e', true) = ((q_i', Id_X, Id_Y))$.

■ $\delta(q_i', e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p))$.

◆ 如果 $\theta = \odot$, 则$\delta(q_0, e, \psi_{cndt}) = ((q_1, \alpha_p, \beta_p))$, 且对于所有$1 \leqslant i < m$, $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p), (q_i', Id_X, Id_Y))$, 而且:

● 如果$n = 0$, 则对于所有$1 \leqslant i < m$:

■ $\delta(q_i, e, \neg\psi_{cndt}) = ((q_i', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_i, e', true) = ((q_i', Id_X, Id_Y))$.

■ $\delta(q_i', e, \neg\psi_{cndt}) = ((q_i', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_i', e', true) = ((q_i', Id_X, Id_Y))$.

■ $\delta(q_i', e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p), (q_i', Id_X, Id_Y))$.

● 如果$n > 0$, 则对于所有$1 \leqslant i < n$, 则$\delta(q_i, e, \neg\psi_{cndt}) = ((q_i, Id_X, Id_Y))$, 对于所有 $e' \neq e$, $\delta(q_i, e', true) = ((q_i, Id_X, Id_Y))$, 而且对于所有$n \leqslant i < m$:

■ $\delta(q_i, e, \neg\psi_{cndt}) = ((q_i', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_i, e', true) = ((q_i', Id_X, Id_Y))$.

■ $\delta(q_i', e, \neg\psi_{cndt}) = ((q_i', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_i', e', true) = ((q_i', Id_X, Id_Y))$.

■ $\delta(q_i', e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p), (q_i', Id_X, Id_Y))$.

■ ξ 定义域为空.

■ $F = \{q_n, \ldots, q_m\}$.

● 如果 $patseq' \equiv p : e : [cndt]_\theta\{n, \infty\}$, 则$\mathcal{T}_{patseq'} = (\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 这里:

■ $Q = \{q_i \mid 0 \leqslant i \leqslant n\} \cup \{q_n'\}$.

■ δ 定义如下.

◆ 如果$\theta = \cdot$, 则对于所有$0 \leqslant i < n$, 我们有 $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p))$, 且$\delta(q_n, e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p))$.

◆ 如果$\theta = \circ$, 则:

● 如果$n = 0$, 则:

■ $\delta(q_0, e, \psi_{cndt}) = ((q_0, \alpha_p, \beta_p))$.

■ $\delta(q_0, e, \neg\psi_{cndt}) = ((q_0', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_0, e', true) = ((q_0', Id_X, Id_Y))$.

■ $\delta(q_0', e, \neg\psi_{cndt}) = ((q_0', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_0', e', true) = ((q_0', Id_X, Id_Y))$.

■ $\delta(q_0', e, \psi_{cndt}) = ((q_0, \alpha_p, \beta_p))$.

● 如果$n > 0$, 则 $\delta(q_0, e, \psi_{cndt}) = ((q_1, \alpha_p, \beta_p))$, 且对于所有 $1 \leqslant i < n$, 我们有 $\delta(q_i, e, \psi_{cndt}) = ((q_{i+1}, \alpha_p, \beta_p))$, $\delta(q_i, e, \neg\psi_{cndt}) = ((q_i, Id_X, Id_Y))$, 对于所有 $e' \neq e$, $\delta(q_i, e', true) = ((q_i, Id_X, Id_Y))$, 另外,

■ $\delta(q_n, e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p))$.

■ $\delta(q_n, e, \neg\psi_{cndt}) = ((q_n', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_n, e', true) = ((q_n', Id_X, Id_Y))$.

■ $\delta(q_n', e, \neg\psi_{cndt}) = ((q_n', Id_X, Id_Y))$.

■ 对于所有 $e' \neq e$, $\delta(q_n', e', true) = ((q_n', Id_X, Id_Y))$.

■ $\delta(q_n', e, \psi_{cndt}) = ((q_n, \alpha_p, \beta_p))$.

◆ 如果$\theta = \odot$, 则:

● 如果$n = 0$, 则:

■ $\delta(q_0,e,\psi_{cndt})=((q_0,\alpha_p,\beta_p),(q_0',Id_X,Id_Y))$.

■ $\delta(q_0,e,\neg\psi_{cndt})=((q_0',Id_X,Id_Y))$.

■ 对于所有 $e'\neq e$, $\delta(q_0,e',true)=((q_0',Id_X,Id_Y))$.

■ $\delta(q_0',e,\neg\psi_{cndt})=((q_0',Id_X,Id_Y))$.

■ 对于所有 $e'\neq e$, $\delta(q_0',e',true)=((q_0',Id_X,Id_Y))$.

■ $\delta(q_0',e,\psi_{cndt})=((q_0,\alpha_p,\beta_p),(q_0',Id_X,Id_Y))$.

● 如果 $n>0$, 则 $\delta(q_0,e,\psi_{cndt})=((q_0,\alpha_p,\beta_p))$, 对于所有 $1\leqslant i<n$, $\delta(q_i,e,\psi_{cndt})=((q_{i+1},\alpha_p,\beta_p),(q_i,Id_X,Id_Y))$, $\delta(q_i,e,\neg\psi_{cndt})=((q_i,Id_X,Id_Y))$, 而且对于所有 $e'\neq e$, $\delta(q_i,e',true)=((q_i,Id_X,Id_Y))$, 另外,

■ $\delta(q_n,e,\psi_{cndt})=((q_n,\alpha_p,\beta_p),(q_n',Id_X,Id_Y))$.

■ $\delta(q_n,e,\neg\psi_{cndt})=((q_n',Id_X,Id_Y))$.

■ 对于所有 $e'\neq e$, $\delta(q_n,e',true)=((q_n',Id_X,Id_Y))$.

■ $\delta(q_n',e,\neg\psi_{cndt})=((q_n',Id_X,Id_Y))$.

■ 对于所有 $e'\neq e$, $\delta(q_n',e',true)=((q_n',Id_X,Id_Y))$.

■ $\delta(q_n',e,\psi_{cndt})=((q_n,\alpha_p,\beta_p),(q_n',Id_X,Id_Y))$.

■ ξ 的定义域为空.

■ $F=\{q_n\}$.

● 如果 $patseq'\equiv p:e:[cndt]_\theta(n,\infty)\ U[cndt']$, 令 $\mathcal{T}_{patseq''}=(\Sigma,X,Y,Q',\delta',\xi',q_0',\eta_0,F')$, 这里:

$patseq''=p:e:[cndt]_\theta(n,\infty)$, 则 $\mathcal{T}_{patseq'}$ 从 $\mathcal{T}_{patseq''}$ 中通过将 δ' 替换为如下的 δ 而得到.

对于所有的 $(q,e',\psi)\in dom(\delta')$, 设 $\delta'(q,e',\psi)=((q_1,\alpha_1,\beta_1),\ldots,(q_k,\alpha_k,\beta_k))$, 则:

$$\delta(q,e',\psi\wedge\neg\psi_{cndt'})=((q_1,\alpha_1,\beta_1),\ldots,(q_k,\alpha_k,\beta_k)).$$

● 如果 $patseq'\equiv patseq_1'\ \theta\ spat$, 且 $spat=p:e:[cndt]$, 令:

$\mathcal{T}_{patseq'_1}=(\Sigma,X,Y,Q_1,\delta_1,\xi_1,q_{1,0},\eta_0,F_1)$, $\mathcal{T}_{spat}=(\Sigma,X,Y,Q_2,\delta_2,\xi_2,q_{2,0},\eta_0,F_2)$, 且:

$Q_1\cap Q_2=\varnothing$, 则 $\mathcal{T}_{patseq'}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 这里:

■ $Q=Q_1\cup Q_2$.

■ δ 从 $\delta_1\cup\delta_2$ 通过如下操作得到: 设 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,f},\alpha_p,\beta_p))$,

◆ 如果 $\theta=\circ$, 则添加如下迁移 $\delta(q_{2,0},e,\neg\psi_{cndt})=((q_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$:

$$\delta(q_{2,0},e',true)=((q_{2,0},Id_X,Id_Y)).$$

◆ 如果 $\theta=\odot$, 去掉迁移 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,f},\alpha_p,\beta_p))$, 且添加如下迁移:

$$\delta(q_{2,0},e,\psi_{cndt})=((q_{2,f},\alpha_p,\beta_p),(q_{2,0},Id_X,Id_Y)),\delta(q_{2,0},e,\neg\psi_{cndt})=((q_{2,0},Id_X,Id_Y)).$$

对于所有 $e'\neq e$, $\delta(q_{2,0},e',true)=((q_{2,0},Id_X,Id_Y))$.

■ ξ 从 $\xi_1\cup\xi_2$ 中通过如下操作得到: 对于所有 $q_{f,1}\in F_1$,

◆ 如果 $q_{f,1}\notin dom(\xi_1)$, 则 $\xi(q_{f,1})=(q_{2,0})$.

◆ 否则, 令 $\xi_1(q_{f,1})=(q_1,\ldots,q_k)$, 则 $\xi(q_{f,1})=(q_1,\ldots,q_k,q_{2,0})$.

■ $q_0=q_{1,0}$.

■ $F=F_2$.

● 如果 $patseq'\equiv patseq_1'\ \theta\ lpat$, 且 $lpat=p:e:[cndt]_{\theta'}\{n,m\}$, 令:

$\mathcal{T}_{patseq'_1}=(\Sigma,X,Y,Q_1,\delta_1,\xi_1,q_{1,0},\eta_0,F_1)$, $\mathcal{T}_{lpat}=(\Sigma,X,Y,Q_2,\delta_2,\xi_2,q_{2,0},\eta_0,F_2)$, 且:

$Q_1\cap Q_2=\varnothing$,则如果 $n=m=0$, $\mathcal{T}_{patseq'}=\mathcal{T}_{patseq'_1}$,否则,令 $Q_2=\{q_{2,0},\ldots,q_{2,m}\}\cup\{q'_{2,n},\ldots,q'_{2,m}\}$,则 $\mathcal{T}_{patseq'}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 这里 Q,ξ,q_0,F 与 $patseq'\equiv patseq_1'\ \theta\ spat$ 时类似, 但是 δ 的定义有所改变: δ 从 $\delta_1\cup\delta_2$ 通过如下操作得到.

◆ 如果 $\theta=\circ$ 且 $\theta'=\cdot$, 则:

● 如果 $n>0$, 则添加如下迁移 $\delta(q_{2,0},e,\neg\psi_{cndt})=((q_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$,

$$\delta(q_{2,0},e',true)=((q_{2,0},Id_X,Id_Y)).$$

● 如果 $n=0$(这时我们有 $n=0<m$), 令 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$, 则添加迁移 $\delta(q_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$, $\delta(q_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, $\delta(q'_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 对于所有 $e'\neq e$, $\delta(q'_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, 且 $\delta(q'_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$.

◆ 如果 $\theta=\odot$ 且 $\theta'=\cdot$, 则令 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$.

● 如果 $n>0$, 则去掉迁移 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$, 添加如下迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q_{2,0},Id_X,Id_Y))$, $\delta(q_{2,0},e,\neg\psi_{cndt})=((q_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$, $\delta(q_{2,0},e',true)=((q_{2,0},Id_X,Id_Y))$.

● 如果 $n=0$(这时我们有 $n=0<m$), 则去掉迁移 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$, 添加迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q'_{2,0},Id_X,Id_Y))$, $\delta(q_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$, $\delta(q_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, $\delta(q'_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 对于所有 $e'\neq e$, $\delta(q'_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, 且 $\delta(q'_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q'_{2,0},Id_X,Id_Y))$.

◆ 如果 $\theta=\odot$ 且 $\theta'=\circ$, 则:

● 如果 $n>0$, 则将迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$ 替换为:

$$\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q_{2,0},Id_X,Id_Y)).$$

● 如果 $n=0$, 则将迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$ 替换为 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q'_{2,0},Id_X,Id_Y))$.

● 如果 $patseq'\equiv patseq'_1\ \theta\ lpat$, 且 $lpat=p:e:[cndt]_{\theta'}\{n,\infty\}$, 令:

$\mathcal{T}_{patseq'_1}=(\Sigma,X,Y,Q_1,\delta_1,\xi_1,q_{1,0},\eta_0,F_1)$, $\mathcal{T}_{lpat}=(\Sigma,X,Y,Q_2,\delta_2,\xi_2,q_{2,0},\eta_0,F_2)$, 且:

$Q_1\cap Q_2=\varnothing$, $Q_2=\{q_{2,0},...,q_{2,n}\}$, 则 $\mathcal{T}_{patseq'}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 这里 Q,ξ,q_0,F 与 $patseq'\equiv patseq'_1\ \theta\ spat$ 时类似, 但是 δ 的定义有所改变. δ 从 $\delta_1\cup\delta_2$ 通过如下操作得到.

■ 如果 $\theta=\circ$ 且 $\theta'=\cdot$, 则:

◆ 如果 $n>0$, 则添加如下迁移 $\delta(q_{2,0},e,\neg\psi_{cndt})=((q_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$,

$$\delta(q_{2,0},e',true)=((q_{2,0},Id_X,Id_Y)).$$

◆ 如果 $n=0$, 令 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p))$, 则添加迁移 $\delta(q_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$, $\delta(q_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, $\delta(q'_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 且对于所有 $e'\neq e$, $\delta(q'_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, $\delta(q'_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p))$.

■ 如果 $\theta=\odot$ 且 $\theta'=\cdot$, 则:

◆ 如果 $n>0$, 则去掉迁移 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$, 然后添加如下迁移:

$$\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q_{2,0},Id_X,Id_Y)),\delta(q_{2,0},e,\neg\psi_{cndt})=((q_{2,0},Id_X,Id_Y)).$$

对于所有 $e'\neq e$, $\delta(q_{2,0},e',true)=((q_{2,0},Id_X,Id_Y))$.

◆ 如果 $n=0$, 则去掉迁移 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p))$, 然后添加如下迁移:

$\delta(q_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p),(q'_{2,0},Id_X,Id_Y))$, $\delta(q'_{2,0},e,\neg\psi_{cndt})=((q'_{2,0},Id_X,Id_Y))$, 对于所有 $e'\neq e$, $\delta(q'_{2,0},e',true)=((q'_{2,0},Id_X,Id_Y))$, $\delta(q'_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p),(q'_{2,0},Id_X,Id_Y))$.

■ 如果 $\theta=\odot$ 且 $\theta'=\circ$, 则:

◆ 如果 $n>0$, 则去掉迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p))$, 然后添加迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,1},\alpha_p,\beta_p),(q_{2,0},Id_X,Id_Y))$.

◆ 如果 $n=0$, 则去掉迁移 $\delta_2(q_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p))$, 然后添加迁移 $\delta(q_{2,0},e,\psi_{cndt})=((q_{2,0},\alpha_p,\beta_p),(q'_{2,0},Id_X,Id_Y))$.

● 如果 $patseq'\equiv patseq'_1\ \theta\ lpat$, 且 $lpat=p:e:[cndt]_{\theta'}\{n,\infty\}\ U\ [cndt']$, 令:

$\mathcal{T}_{patseq'_1}=(\Sigma,X,Y,Q_1,\delta_1,\xi_1,q_{1,0},\eta_0,F_1)$, $\mathcal{T}_{lpat}=(\Sigma,X,Y,Q_2,\delta_2,\xi_2,q_{2,0},\eta_0,F_2)$, 且:

$Q_1\cap Q_2=\varnothing$, $Q_2=\{q_{2,0},\ldots,q_{2,n}\}$, 则 $\mathcal{T}_{patseq'}=(\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 这里 Q,ξ,q_0,F 与 $patseq'\equiv patseq'_1\ \theta\ spat$

时类似, 但是δ的定义有所改变: δ从$\delta_1 \cup \delta_2$通过如下操作得到.

■ 如果$\theta = \circ$且$\theta' = \cdot$, 则添加如下迁移$\delta(q_{2,0}, e, \neg\psi_{cndt}) = ((q_{2,0}, Id_X, Id_Y))$, 而且, 对于所有$e' \neq e$, $\delta(q_{2,0}, e', \neg\psi_{cndt'}) = ((q_{2,0}, Id_X, Id_Y))$.

■ 如果$\theta = \odot$且$\theta' = \cdot$, 则:

◆ 如果$n > 0$, 则去掉迁移$\delta_2(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,1}, \alpha_p, \beta_p))$, 然后添加如下迁移:

$\delta(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,1}, \alpha_p, \beta_p), (q_{2,0}, Id_X, Id_Y))$, $\delta(q_{2,0}, e, \neg\psi_{cndt}) = ((q_{2,0}, Id_X, Id_Y))$, 对于所有$e' \neq e$, $\delta(q_{2,0}, e', \neg\psi_{cndt'}) = ((q_{2,0}, Id_X, Id_Y))$.

◆ 如果$n = 0$, 则去掉迁移$\delta_2(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, \alpha_p, \beta_p))$, 然后添加如下迁移:

$\delta(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, \alpha_p, \beta_p), (q'_{2,0}, Id_X, Id_Y))$, $\delta(q_{2,0}, e, \neg\psi_{cndt}) = ((q'_{2,0}, Id_X, Id_Y))$, 对于所有$e' \neq e$, $\delta(q_{2,0}, e', \neg\psi_{cndt'}) = ((q'_{2,0}, Id_X, Id_Y))$, $\delta(q'_{2,0}, e, \neg\psi_{cndt}) = ((q'_{2,0}, Id_X, Id_Y))$, 对于所有$e' \neq e$, $\delta(q'_{2,0}, e', \neg\psi_{cndt'}) = ((q'_{2,0}, Id_X, Id_Y))$, 且:

$$\delta(q'_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, \alpha_p, \beta_p), (q'_{2,0}, Id_X, Id_Y)).$$

■ 如果$\theta = \odot$且$\theta' = \circ$, 则:

◆ 如果$n > 0$, 则将迁移$\delta(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,1}, \alpha_p, \beta_p))$替换为迁移$\delta(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,1}, \alpha, \beta), (q_{2,0}, Id_X, Id_Y))$, 且将迁移$\delta(q_{2,0}, e, \neg\psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, Id_X, Id_Y))$替换为$\delta(q_{2,0}, e, \neg\psi_{cndt}) = ((q_{2,0}, Id_X, Id_Y))$.

◆ 如果$n = 0$, 则去掉迁移$\delta_2(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, \alpha_p, \beta_p))$, 添加迁移$\delta(q_{2,0}, e, \psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, \alpha_p, \beta_p), (q'_{2,0}, Id_X, Id_Y))$, 且将迁移$\delta(q_{2,0}, e, \neg\psi_{cndt} \wedge \neg\psi_{cndt'}) = ((q_{2,0}, Id_X, Id_Y))$替换为$\delta(q_{2,0}, e, \neg\psi_{cndt}) = ((q_{2,0}, Id_X, Id_Y))$.

● 如果$patseq' \equiv (patseq'_1)$, 则$\mathcal{T}_{patseq'} = \mathcal{T}_{patseq'_1}$.

● 如果$patseq' \equiv (patseq'_1)\{n, m\}$, 令$\mathcal{T}_{patseq'_1} = (\Sigma, X, Y, Q_1, \delta_1, \xi_1, q_{1,0}, \eta_0, F_1)$, 则$\mathcal{T}_{patseq'} = (\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 这里:

■ $Q = (Q_1 \times [m]) \cup \{q_f\}$.

■ δ定义如下: 对于所有的$(q, e, \psi) \in dom(\delta_1)$, 令$\delta_1(q, e, \psi) = ((q_1, \alpha_1, \beta_1), \ldots, (q_k, \alpha_k, \beta_k))$, 则对于所有$i \in [m]$, $\delta((q, i), e, \psi) = (((q_1, i), \alpha_1, \beta_1), \ldots, ((q_k, i), \alpha_k, \beta_k))$.

■ ξ定义如下.

◆ 对于所有$q \in Q_1 \setminus F_1$和$i \in [m]$, $\xi((q, i))$从$\xi_1(q)$中通过将所有的状态q'替换为(q', i)而得到.

◆ 对于所有$q \in F_1$和$i \in [n-1]$.

● 如果$q \notin dom(\xi_1)$, 则$\xi((q, i)) = ((q_{1,0}, i+1))$.

● 否则, 令$\xi_1(q) = (q_1, \ldots, q_k)$, 则$\xi((q, i)) = ((q_1, i), \ldots, (q_k, i), (q_{1,0}, i+1))$.

◆ 对于所有$q \in F_1$和$n \leqslant i < m$,

● 如果$q \notin dom(\xi_1)$, 则:

$$\xi((q, i)) = ((q_{1,0}, i+1), q_f).$$

● 否则, 令$\xi_1(q) = (q_1, ..., q_k)$, 则:

$$\xi((q, i)) = ((q_1, i), \ldots, (q_k, i), (q_{1,0}, i+1), q_f).$$

◆ 对于所有$q \in F_1$,

● 如果$q \notin dom(\xi_1)$, 则:

$$\xi((q, m)) = (q_f).$$

● 否则, 令$\xi_1(q) = (q_1, \ldots, q_k)$, 则:

$$\xi((q, m)) = ((q_1, m), \ldots, (q_k, m), q_f).$$

■ $q_0 = (q_{1,0}, 1)$.

■ $F = \{q_f\}$.

● 如果$patseq' \equiv (patseq'_1)\{n, \infty\}$, 令$\mathcal{T}_{patseq'_1} = (\Sigma, X, Y, Q_1, \delta_1, \xi_1, q_{1,0}, \eta_0, F_1)$.

则$\mathcal{T}_{patseq'} = (\Sigma, X, Y, Q, \delta, \xi, q_0, \eta_0, F)$, 这里:

■ $Q = Q_1 \times [n]$.

■ δ 定义如下.

对于所有的 $(q,e,\psi) \in dom(\delta_1)$, 令 $\delta_1(q,e,\psi) = ((q_1,\alpha_1,\beta_1),\ldots,(q_k,\alpha_k,\beta_k))$, 则对于所有 $i \in [n]$, $\delta((q,i),e,\psi) = (((q_1,i),\alpha_1,\beta_1),\ldots,((q_k,i),\alpha_k,\beta_k))$.

■ ξ 定义如下.

◆ 对于所有 $q \in Q_1 \setminus F_1$ 和 $i \in [n]$, $\xi((q,i))$ 从 $\xi_1(q)$ 中通过将所有的状态 q' 替换为 (q',i) 而得到.

◆ 对于所有 $q \in F_1$ 和 $i \in [n-1]$.

● 如果 $q \notin dom(\xi_1)$, 则 $\xi((q,i)) = ((q_{1,0},i+1))$.

● 否则, 令 $\xi_1(q) = (q_1,\ldots,q_k)$, 则 $\xi((q,i)) = ((q_1,i),\ldots,(q_k,i),(q_{1,0},i+1))$.

◆ 对于所有 $q \in F_1$,

● 如果 $q \notin dom(\xi_1)$, 则 $\xi((q,n)) = ((q_{1,0},n))$.

● 否则, 令 $\xi_1(q) = (q_1,\ldots,q_k)$, 则 $\xi((q,n)) = ((q_1,n),\ldots,(q_k,n),(q_{1,0},n))$.

■ $q_0 = (q_{1,0},1)$.

■ $F = F_1 \times \{n\}$.

● 如果 $patseq' \equiv (patseq'_1)\{n,\infty\}\ U\ cndt'$, 令 $\mathcal{T}_{patseq''} = (\Sigma,X,Y,Q',\delta',\xi',q'_0,\eta_0,F')$, 这里 $patseq'' \equiv (patseq'_1)\{n,\infty\}$, 则 $\mathcal{T}_{patseq'}$ 从 $\mathcal{T}_{patseq''}$ 中通过将 δ' 替换为如下的 δ 而得到. 对于所有的 $(q,e',\psi) \in dom(\delta')$,

设 $\delta'(q,e',\psi) = ((q_1,\alpha_1,\beta_1),\ldots,(q_k,\alpha_k,\beta_k))$, 则 $\delta(q,e',\psi \wedge \neg cndt') = ((q_1,\alpha_1,\beta_1),\ldots,(q_k,\alpha_k,\beta_k))$.

● 如果 $patseq' \equiv patseq'_1 \cdot gpat$, 设 $\mathcal{T}_{patseq_1} = (\Sigma,X,Y,Q_1,\delta_1,\xi_1,q_{1,0},\eta_0,F_1)$.

$\mathcal{T}_{gpat} = (\Sigma,X,Y,Q_2,\delta_2,\xi_2,q_{2,0},\eta_0,F_2)$, 且 $Q_1 \cap Q_2 = \varnothing$, 则 $\mathcal{T}_{patseq} = (\Sigma,X,Y,Q,\delta,\xi,q_0,\eta_0,F)$, 这里:

■ $Q = Q_1 \cup Q_2$.

■ $\delta = \delta_1 \cup \delta_2$.

■ ξ 从 $\xi_1 \cup \xi_2$ 中通过如下操作得到.

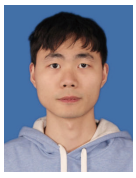
◆ 对于所有 $q_{f,1} \in F_1$,

● 如果 $q_{f,1} \notin dom(\xi_1)$, 则 $\xi(q_{f,1}) = (q_{2,0})$.

● 否则, 令 $\xi_1(q_{f,1}) = (q_1,\ldots,q_k)$, 则 $\xi(q_{f,1}) = (q_1,\ldots,q_k,q_{2,0})$.

■ $q_0 = q_{1,0}$.

■ $F = F_2$.



傅宣登(1998－), 男, 硕士, 主要研究领域为复杂事件处理语言.

吴志林(1980－), 男, 博士, 研究员, CCF 专业会员, 主要研究领域为计算逻辑, 自动机理论, 程序验证.