CEP In Flink (1) - CEP规则解析

前言

随着数据分析精细化程度的提升，在企业级应用上，对事件的分析不再仅仅局限于普通的离线计算或者是简单的数字统计，而是转向更实时、更精准、更复杂的数据分析。复杂事件处理的复杂在于事件之间具有关联性，且这种联结在时间上是多种多样的。在之前的Oracle中CEP的语法介绍过一个复杂事件的案例，在此就不做过多阐述。

现状

- EsperTech已经开发出了自己的一套DSL来描述复杂事件，同时运用在了自身的2B产品上。
- siddhi是一个同时支持流式数据处理和复杂事件分析的开源框架，Uber和Apache Eagle都曾使用过，非常轻量，可以部署在用户终端使用。
- Oracle和Calcite目前都有一套相对完整的SQL来描述复杂事件，详见Oracle中CEP的语法。

由于事件之间的关系复杂性，导致描述这种关系的语言也显得比较复杂。上述三种CEP相关服务中使用的描述语言和API都不一样，且学习成本也相对较大。不过Calcite和Oracle都可以使用SQL来描述CEP中事件之间的关系，Flink的CEP也在基于Calcite去做SQL的实现，个人认为这种标准的SQL规范方式会比较方便理解和广泛推广。

理论支持

Apache Flink在实现CEP时借鉴了Efficient Pattern Matching over Event Streams中NFA的模型，在这篇论文中，还提到了一些优化，我们在这里先跳过，专注于CEP规则解析的部分。

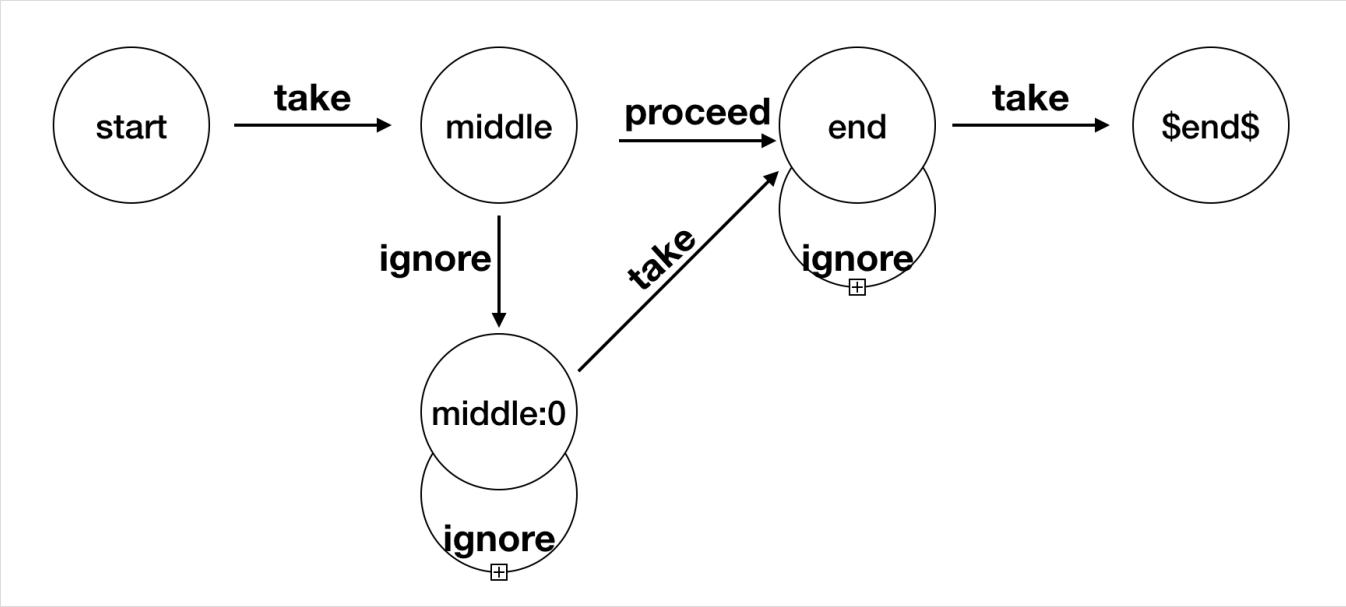
在这篇论文中，提到了NFA，也就是Non-determined Finite Automaton，叫做不确定的有限状态机，指的是状态有限，但是每个状态可能被转换成多个状态（不确定）。

我们以一个简单的CEP规则为例，看看在NFA中，这些事件之间是什么样的关系。

```
Pattern<Event, ?> pattern = Pattern.<Event>begin("start").where(new SimpleCondition<Ev
ent>() {
    @Override
    public boolean filter(Event value) throws Exception {
        return value.getName().equals("a");
    }
}).followedBy("middle").optional().where(new SimpleCondition<Event>() {
    @Override
    public boolean filter(Event value) throws Exception {
        return value.getName().equals("b");
```

```
    }
}).followedBy("end").where(new SimpleCondition<Event>() {
    @Override
    public boolean filter(Event value) throws Exception {
        return value.getName().equals("c");
    }
});
```

上述代码描述的是start/middle/end之间的关系且每个事件满足的条件，其中，middle要在start之后，end要在middle之后；三者之间并不需要严格邻近，其中middle是可有可无的（optional），用NFA的结构来描述他们就是下面这张图：



先介绍两个概念：

- **状态**：start/middle/middle:0/end/\$end\$都是状态的名称。
- **转换**：take/ignore/proceed都是转换的名称。

在这NFA匹配规则里，本质上是一个状态转换的过程。三种转换的含义如下所示：

- **Take**: 满足条件，获取当前元素，进入下一状态。
- **Proceed**：不论是否满足条件，不获取当前元素，直接进入下一状态（如optional）并进行判断是否满足条件。
- **Ignore**：不满足条件，忽略，进入下一状态。

现在让我们假设一条只有四个元素的数据流：

> start –> xx –> middle –> end

1. 收到start，满足条件，进行Take转换，当前状态转为middle。
2. 收到xx，不满足条件，当前状态转为middle:0；因为有Proceed存在，当前状态转为end，但也不满足条件，所以丢弃这条转换。
3. 收到middle，对于middle:0满足条件(Take)转换为end；对于end，不满足条件(Ignore)，转换为自身。

4. 收到end，满足条件(Take)，转换为\$end\$，结束匹配。

综述

这个是Flink-CEP系列的第一篇文章，描述了一些背景知识，利用一个简单的例子来解释NFA对于CEP规则的解析。下一篇文章将会深入到CEP具体的匹配过程中，敬请期待。