Otto Friedrich University Bamberg

Data Streams and Complex Event Processing

Assignment - 04

Supervisor:
Prof. Dr. Daniela Nicklas
Mobile Software Systems

Submitted By :
Manasa Reddy Bandari - 1942692
Akshay Sharma - 1943984
Kummari Daniel Raj - 1973448

Jan 13, 2019

Contents

1 Exercise 1 :

We want to perform some pattern matching on data streams using the bids stream provided by the Nexmark data generator. Write queries for Odysseus using SASE+ language to detect the following patterns:

1.1 A bid for auction id = 0 is followed by another bid for auction id = 1 within 10 seconds.

```
PATTERN SEQ(bid auction1, bid auction2)
WHERE skip_till_next_match(auction1,auction2)
{ auction1.auction =0 and auction2.auction = 1 }
WITHIN 10 seconds
RETURN auction1.auction,auction2.auction
```

1.2 A bid for any auction is followed by a higher bid of the same auction (also within 10 seconds).

```
PATTERN SEQ(bid auction1, bid auction2)
WHERE skip_till_next_match(auction1,auction2)
{ auction1.auction = auction2.auction and auction1.price >
   auction2.price }
WITHIN 10 seconds
RETURN auction1.auction, auction1.price, auction2.auction,
   auction2.price
```

1.3 A bid for any auction is followed by a doubled offer bid of the same auction within 60 seconds.

```
PATTERN SEQ(bid auct1,bid+ a[])
WHERE skip_till_next_match (auct1,a[])
{[auction] and auct1.price = 0.5 * a[a.LEN].price}
WITHIN 60 seconds
RETURN auct1.auction, auct1.price, a[a.LEN].auction, a[a.LEN].
   price
```

1.4 A bid for any auction is followed by a bid, with a price 80]% more than the previous bid (within 10 seconds).

```
PATTERN SEQ(bid auct1,bid+ a[]) WHERE
skip_till_next_match (auct1,a[])
{[auction] and auct1.price + (auct1.price * 0.8)<a[a.LEN].
   price}
```

```
WITHIN 10 seconds
RETURN auct1.auction, auct1.price, a[a.LEN].auction, a[a.LEN].
    price
```

1.5 Draw the automaton and state transitions for the queries 2, 3 and 4.

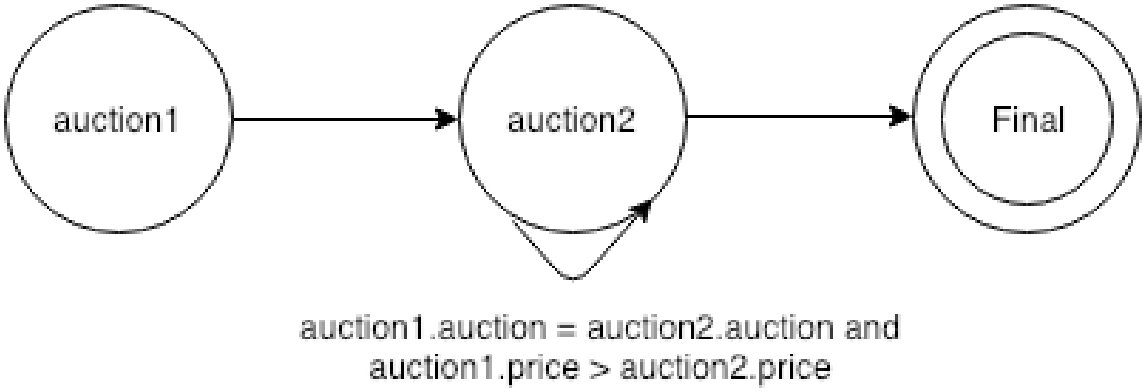

Figure 1: Query 2

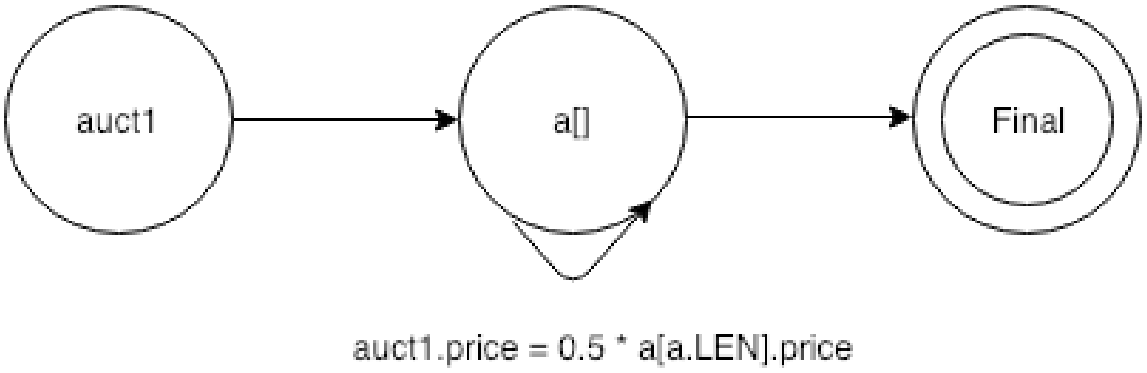

Figure 2: Query 3

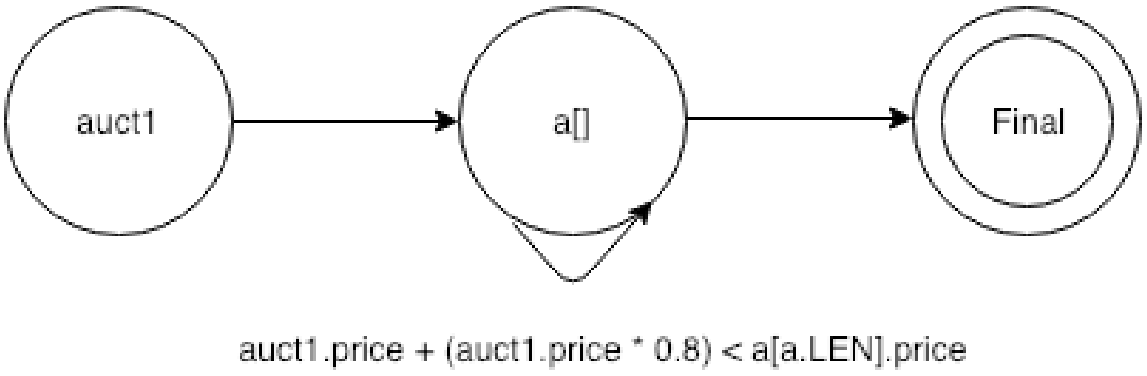

Figure 3: Query 4

2 Answer the following questions:

2.1 What are the pattern policies offered by SASE+?

There are four policies offered by SASE+:

–> Strict contiguity
–> Skip till next match
–> Skip till any match
–> Partition Contiguity

2.2 Give a use case for each pattern policy (use case means an example where the policy is applied).

2.2.1 Strict contiguity

It's generally used for expression matching for example checking your location through google maps

2.2.2 Skip till next match

In this you have to skip till the next match is found. Many dependant scenarios can be implemented in this. For example, calculating mileage of the car

2.2.3 Skip till any match

It's more likely the same as the previous one but will do it for any match found

2.2.4 Partition Contiguity

This policy can be used in which sorting requirements are there.

2.3 Explain briefly what each of the clauses PATTERN, WHERE, and WITHIN is responsible for.

PATTERN is used to provide the pattern to match with the available requirements.

WHERE provides constraints and conditions for the previous pattern clauses.

WITHIN provides a time frame for the pattern and where clause to get the output within given time window.

2.4 What are the termination criteria for a Kleene closure (might need to look further than the slides content for this, you can take a look at the paper: On Supporting Kleene Closure over Event Streams; Daniel Gyllstrom, Jagrati Agrawal, Yanlei Diao and Neil Immerman).

There are two termination criteria for Kleene closure:
–> Termination criteria for contiquity requirement
–> Termination criteria for relaxed pattern policies

2.5 Which additional general pattern policy is "hidden" in the termination criteria of Kleene closure in SASE+?

Skip until next match is the hidden pattern policy in the termination criteria in which all the events will appear until the match is finally there.