Pattern operator

To use the pattern operator, you will need to install the **de.uniol.inf.is.odysseus.cep.feature (e.g. with "#REQUIRED de.uniol.inf.is.odysseus.cep.feature.feature.group")**

This generic operator allows the definition of different kinds of pattern (e.g. all, any). For sequence based patterns see SASE operator. In the following the implemented pattern types are described.

Parameter

- `type`: The type of pattern that should be detected. See below for a list of all supported types and examples.
- `eventTypes`: Describes the types of the input ports
- `time`: If there should be a temporal context (window) this states the time interval size.
- `timeUnit`: If there should be a temporal context, this states the time unit.
- `size`: For element based windows, this is the count of elements that are treated together, size and time can be used together
- `assertions`: Predicate over the input data that must be fullfilled to create an output
- `outputmode`: states, what the operator should deliver:
  - EXPRESSION: use the `return` parameter to create the output
  - INPUT: Deliver events from input port 0, can be changed with parameter `inputPort`
  - TUPLE_CONTAINER: Deliver all events that are related to this matching
  - SIMPLE
- `return`: see `outputmode`/EXPRESSION
- `inputPort`: see `outputmode`/INPUT

Logical Pattern

ALL

The ALL type selects all events that match a given pattern.

```
PATTERN({type = 'ALL', eventTypes = ['person', 'bid'],
time = 10, timeUnit = 'MINUTES',
assertions = ['person.id = bid.bidder && bid.price > 200'],
outputmode = 'EXPRESSIONS',
return = ['person.id', 'person.name', 'bid.price']},
person, bid)
```

In the example, the query selects a bid, with a value higher than 200, and the person who won the bid, th e ID and name of the person and the price of the bid. Be taken into account only persons and commandm ents, which are not older than 10 min. In summary, one could say that the request selects the persons wit hin 10 min after its release already bid with a value over 200 exits. The time and size parameter determine how long and how many events are cached.

ANY

The ANY type selects one event that matches the given pattern.

```
PATTERN({type = 'ANY', eventTypes = ['bid'],
assertions = ['bid.price > 220'],
outputMode = 'INPUT'}, bid)
```

In the example, the query selects the request of each bid with a value higher than 200 from. The i-th asse rtion applies only to events of the type that is the i-th position of the relevant Event Type list.

ABSENCE

The ABSENCE type detects the absence of a particular event in the stream.

```
PATTERN({type = 'ABSENCE', eventTypes = ['bid'],
outputMode = 'SIMPLE', time = 400}, bid)
```

In the example, the query recognizes the request if 400 milliseconds was no bid. Success and accuracy depends on the heartbeats. As output mode only SIMPLE is possible.

Threshold Pattern

COUNT

The COUNT type triggers as soon as the given number of events are detected.

```
PATTERN({type = 'FUNCTOR', eventTypes = ['aggr'],
assertions = ['count_price > 20']},
AGGREGATE({aggregations = [['COUNT', 'price',
'count_price', 'double']]}, bid))
```

The request is fulfilled once more than 20 bids were submitted. Any currently based on the pattern, the ag gregation gets as input from the outside.

VALUE-MAX

The VALUE-MAX type triggers as soon as the given value is exceeded.

```
PATTERN({type = 'FUNCTOR', eventTypes = ['bid'],
assertions = ['max_price > 300']},
AGGREGATE({aggregations=[['MAX', 'price', 'max_price',
'double']]}, bid))
```

The request is fulfilled as soon as the maximum value of the bid exceeds 300. Any currently based on the pattern, the aggregation gets as input from the outside.

VALUE-MIN

The VALUE-MIN type triggers as soon as the minimum value fall below the given threshold.

```
PATTERN({type = 'FUNCTOR', eventTypes = ['bid'],
assertions = ['min_price > 50 && min_price < 100']},
AGGREGATE({aggregations=[['MIN', 'price', 'min_price',
'double']]}, bid))
```

The request is fulfilled as long as the minimum value of a bid is greater than 50 and less than 100. Any cu rrently based on the pattern, the aggregation gets as input from the outside.

VALUE-AVERAGE

The VALUE-AVERAGE type triggers as soon as th earithmetic mean fall below the given threshold.

```
PATTERN({type = 'FUNCTOR', eventTypes = ['bid'],
assertions = ['avg_price < 140']},
AGGREGATE({aggregations=[['AVG', 'price', 'avg_price',
'double']]}, bid))
```

The request is fulfilled if the arithmetic mean of a bid is less than 140. Any currently based on the pattern, the aggregation gets as input from the outside.

Subset Selection Pattern

RELATIVE-N-HIGHEST

The RELATIVE-N-HIGHEST type selects the N highest matching events.

```
PATTERN({type = 'RELATIVE_N_HIGHEST', eventTypes = ['bid'],
attribute = 'price', count = 3,
time = 6, timeUnit = 'SECONDS',
outputmode = 'expressions',
return = ['bid.timestamp', 'bid.bidder',
'bid.price']}, bid)
```

The query selects every six seconds from the three highest bids. The output mode SIMPLE is possible, but does not make much sense.

RELATIVE-N-LOWEST

The RELATIVE-N-LOWEST type selects the N lowest matching events.

```
PATTERN({type = 'RELATIVE_N_LOWEST', eventTypes = ['bid'],
assertions = ['price > 80'],
attribute = 'price', count = 3,
time = 10, timeUnit = 'SECONDS',
outputmode = 'TUPLE_CONTAINER'}, bid)
```

The query selects every six seconds from the commandments which are higher than 80, the three lowest bids. The output mode SIMPLE is possible, but in this context usually makes no sense.

Modale Pattern

ALWAYS

The ALWAYS type triggers iff all events match the given pattern.

```
PATTERN({type = 'ALWAYS', eventTypes = ['bid'],
time = 3, timeUnit = 'SECONDS',
assertions = ['bid.price > 140'],
outputMode = 'INPUT'}, bid)
```

If all bids are greater than 140 within the fixed interval of three seconds, it will be issued by the pattern. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

SOMETIMES

The SOMETIMES type triggers if at least one event match the given pattern.

```
PATTERN({type = 'SOMETIMES', eventTypes = ['bid'],
time = 10, timeUnit = 'SECONDS',
assertions = ['bid.price > 280']}, bid)
```

The pattern is satisfied if at least one bid is greater than 280 within the fixed interval of ten seconds. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

Temporal Order Pattern

SEQUENCE

The SEQUENCE type triggers iff the input events match the given pattern. The request is based on the SASE operator. The query parameter expects a query formulated in the SASE query language. See. SASE

```
SASE({query = 'PATTERN SEQ(person p, bid b)
WHERE skip_till_next_match(p,b)
{p.id = b.bidder, b.price > 200}
RETURN p.id, p.name, b.price', schema=[['id','Integer'],['name','String'],
['price','double']], type='PersonEvent1'} , person, bid)
```

The query selects attributes of people and the commandments of the respective persons for whom the person appears before the commandment and his commandment is greater than 200.

FIRST-N

The FIRST-N type returns the first N events.

```
PATTERN({type = 'FIRST_N', eventTypes = ['bid'],
time = 10, timeUnit = 'SECONDS',
count = 3,
assertions = ['bid.price > 100'],
outputmode = 'EXPRESSIONS',
return = ['bid.timestamp', 'bid.bidder',
'bid.price']}, bid)
```

The query selects every ten seconds, the first three bids from greater than 100, and outputs the specified attributes from. The output mode SIMPLE is possible, but in this context usually makes no sense.

LAST-N

The LAST-N type returns the last N events.

```
PATTERN({type = 'LAST_N', eventTypes = ['person',
'auction'],
time = 10, timeUnit = 'SECONDS',
count = 3,
outputmode = 'TUPLE_CONTAINER'}, auction, person)
```

The query selects every ten seconds from the last three relevant events. This can be bids and auctions. T he output mode SIMPLE is possible, but in this context usually makes no sense. Contains the output vari ous types of events makes the output mode TUPLE_CONTAINER sense, since there the schema of the d ata does not matter. For other output modes may arise null values or the like.

Trend Pattern

INCREASING

The INCREASING type triggers if the value of the event expression rise monotonically.

```
PATTERN({type = 'INCREASING', eventTypes = ['bid'],
attribute = 'price',
time = 2, timeUnit = 'SECONDS'}, bid)
```

The pattern is met when the values of the bids rise monotonically within the fixed time interval of 2 second s. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

DECREASING

The DECREASING type triggers if the value of the event expression fall monotonically.

```
PATTERN({type = 'DECREASING', eventTypes = ['bid'],
attribute = 'price',
time = 2, timeUnit = 'SECONDS'}, bid)
```

The pattern is met when the values of the bids within the fixed time interval 2 seconds fall monotonically. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

STABLE

The STABLE type triggers if the value of the event expression does not change.

```
PATTERN({type = 'STABLE', eventTypes = ['bid'],
attribute = 'price',
time = 2, timeUnit = 'SECONDS'}, bid)
```

The pattern is met if not change the values of the bids within the fixed time interval of 2 seconds. Is not th e output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the a ssertions.

NON-INCREASING

```
PATTERN({type = 'NON_INCREASING', eventTypes = ['bid'],
attribute = 'price',
time = 2, timeUnit = 'SECONDS'}, bid)
```

The pattern is met when the values of the bids within the fixed time interval 2 seconds fall monotonically. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

NON-DECREASING

```
PATTERN({type = 'NON_DECREASING', eventTypes = ['bid'],
attribute = 'price',
time = 2, timeUnit = 'SECONDS'}, bid)
```

The pattern is met when the values of the bids rise monotonically within the fixed time interval of 2 second s. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

NON-STABLE

```
PATTERN({type = 'NON_STABLE', eventTypes = ['bid'],
attribute = 'price', size = 3}, bid)
```

The pattern is the counterpart to the stable pattern. It is true if the value changes of three consecutive bids . Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

MIXED

```
PATTERN({type = 'MIXED', eventTypes = ['bid'],
attribute = 'price',
time = 2, timeUnit = 'SECONDS'}, bid)
```

The pattern is met when the values of the bids within the fixed time interval 2 seconds to rise at least once a strictly monotonic and strictly monotonic fall at least once. Is not the output mode SIMPLE be spent in the performance of the pattern all the relevant events that satisfy the assertions.

Spatial Pattern

MIN-DISTANCE

MAX-DISTANCE

AVERAGE-DISTANCE

RELATIVE-MIN-DISTANCE

RELATIVE-MAX-DISTANCE

RELATIVE-AVERAGE-DISTANCE

Spatial Temporal Pattern

MOVING-IN-A-CONSTANT-DIRECTION

MOVING-IN-A-MIXED-DIRECTION

STATIONARY

MOVING-TOWARD