

SASE

The following text is from <http://avid.cs.umass.edu/sase/index.php?page=language> (slightly modified to the capabilities of Odysseus)

The SASE+ language has a high-level structure similar to SQL for ease of use. The design of the language, however, is centered on temporal event patterns that have not been sufficiently addressed in relational data processing. In particular, SASE+ has the following features:

- **Sequencing:** a sequence of events occur in specified order.
- **Negation:** an event pattern can address the non-occurrences of events.
- **Kleene closure:** a pattern can also be used to extract a finite yet unbounded number of events with a particular property.
- **Parameterized predicates:** different events are compared via value-based constraints as well as temporal constraints.
- **Sliding windows:** the entire pattern needs to occur within a specified period of time.

The overall structure of the SASE+ language is:

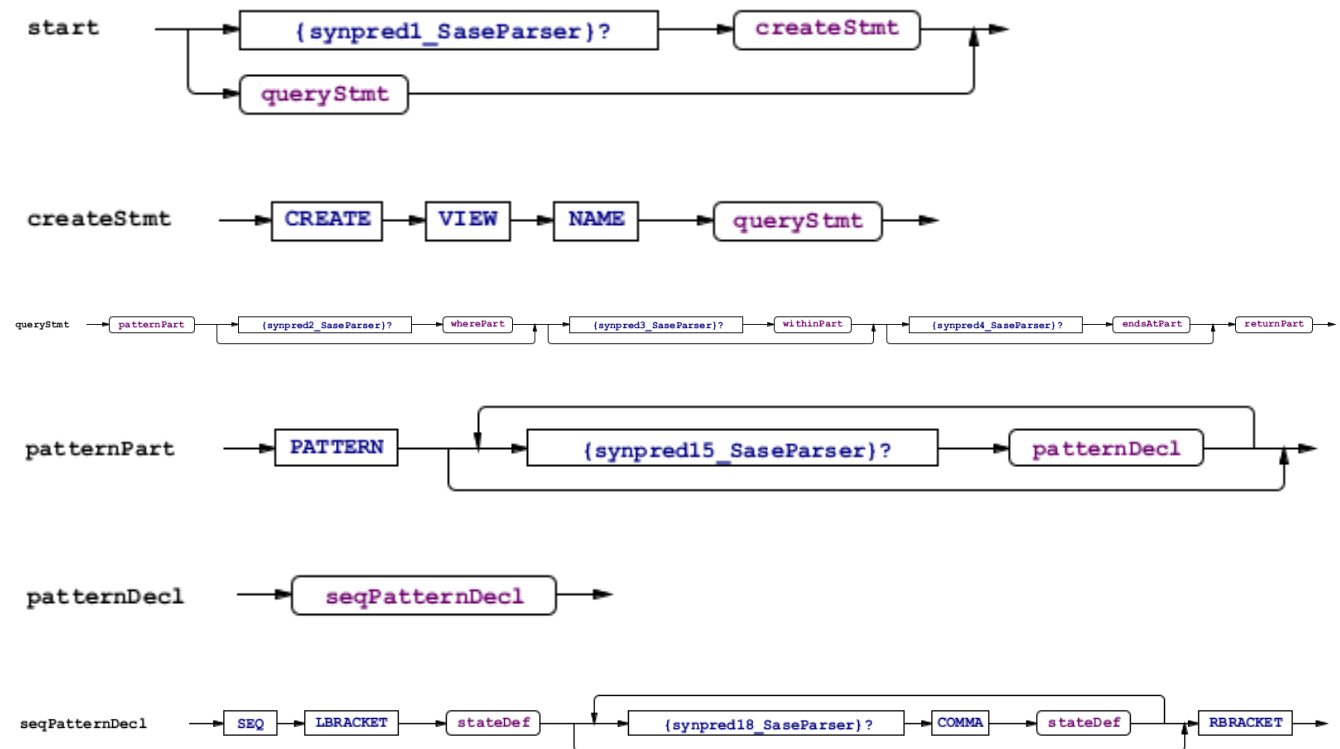
```
PATTERN < pattern structure >  
[WHERE < pattern matching condition >]  
[WITHIN < sliding window >]  
[ENDS AT] < attribute with temporal condition >  
[RETURN < output specification >]
```

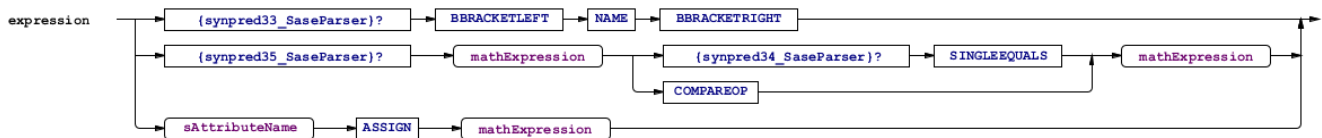
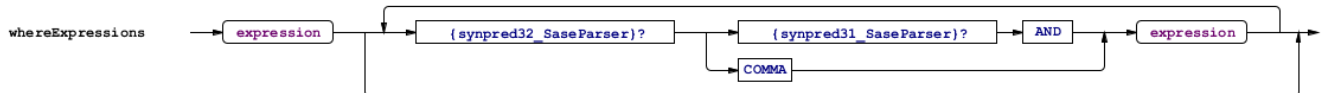
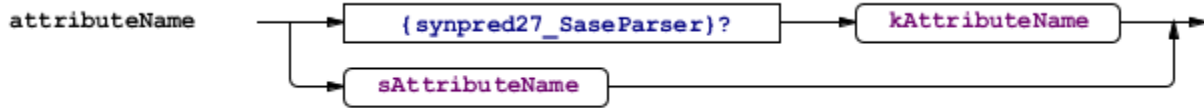
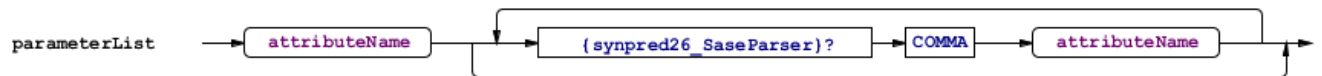
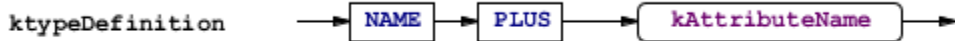
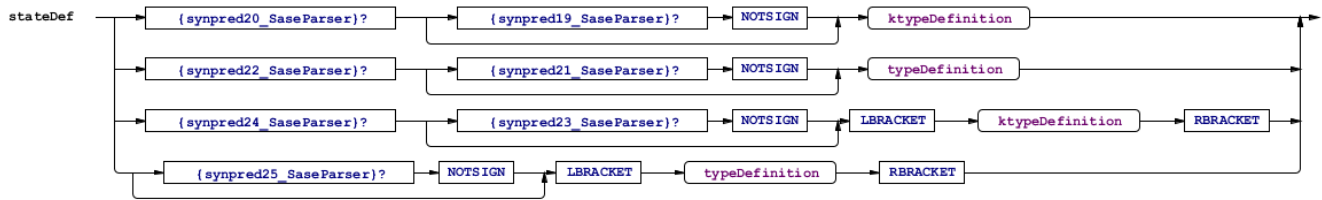
The PATTERN, WHERE and WITHIN clauses form the pattern matching block. The PATTERN clause specifies the structure of a pattern to be matched against the input stream. The WHERE clause, if present, imposes value-based constraints on the events addressed by the pattern. The WITHIN clause further specifies a sliding window over the entire pattern.

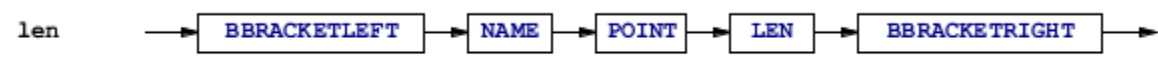
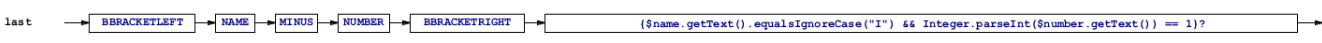
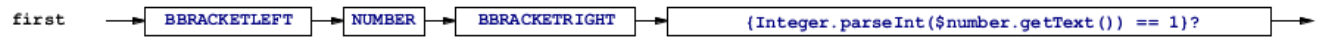
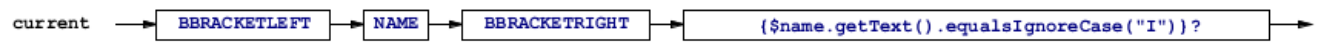
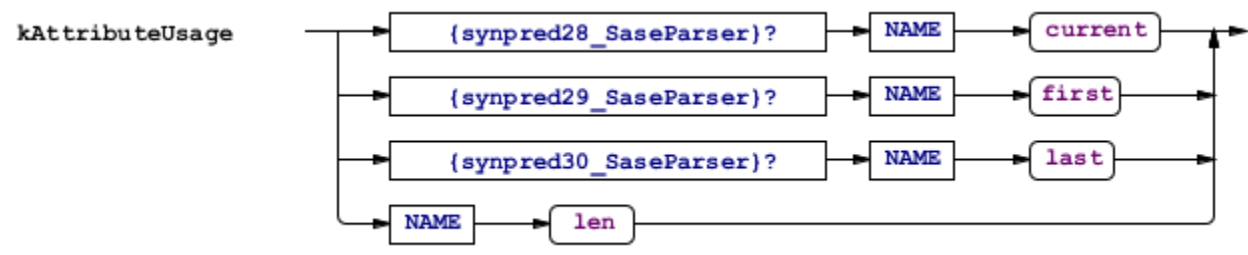
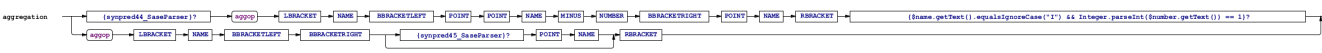
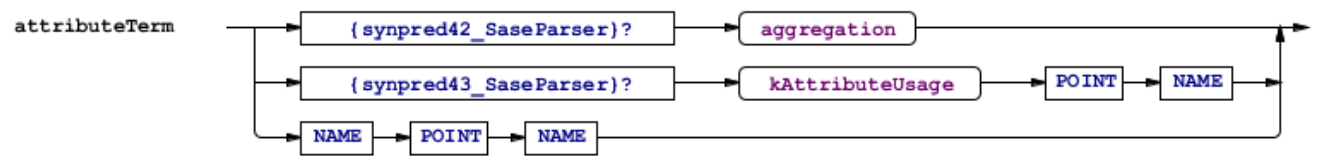
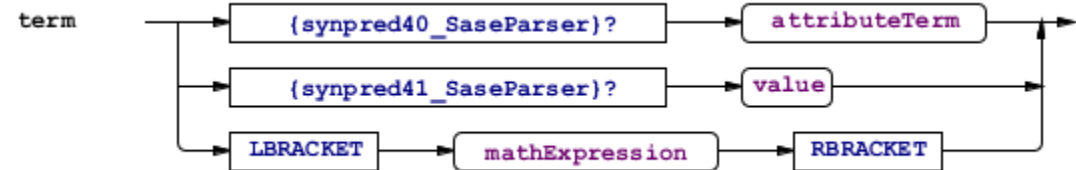
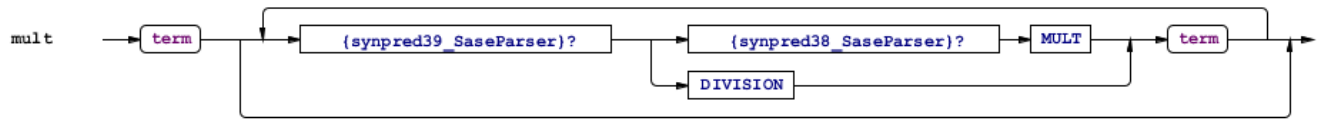
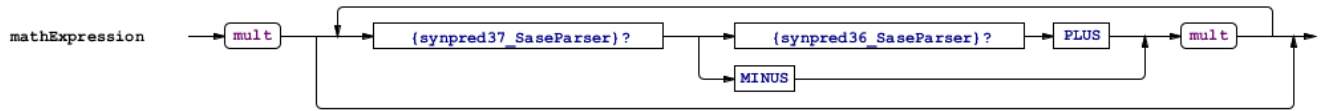
The evaluation of PATTERN, WHERE, and WITHIN clauses results in a stream of pattern matches; each consists of a unique sequence of events used to match the pattern. The HAVING clause further filters each pattern match by applying predicates on the constituent events.

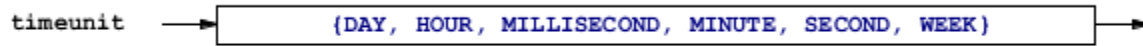
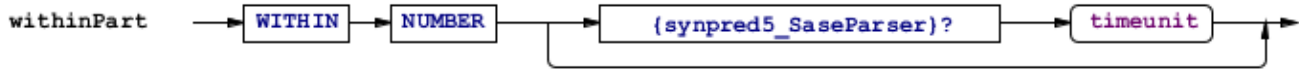
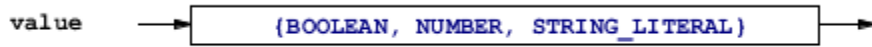
Finally, the RETURN clause transforms each pattern match into a result event for output. Although expressions are possible, in Odysseus only attributes should be used in the RETURN statement

Syntax of our Implementation









Examples

The following examples to not work all in Odysseus, currently. There is no ANY.

Examples from <http://avid.cs.umass.edu/sase/>

```
Query 1:
PATTERN      SEQ(ShelfReading x, ~ CounterReading y, ExitReading z)
WHERE        x.id = y. id AND x.id = z.id      /* equivalently, [id] */
WITHIN 16 hours
```

```
Query 2:
PATTERN      SEQ(ShelfReading x, ShelfReading y, ~ (ANY(CounterReading, ShelfReading) z) )
WHERE        [id] AND x.shelf_id y.shelf_id AND x.shelf_id = z.shelf_id
WITHIN      1 hour
```

```
Query 3:
PATTERN      (StartLoading a, RfidReading+ b[ ], EndLoading c)
WHERE        [loading_dock]
AND          a.session_id = c.session_id
AND          b[i].packaging_level = 'pallet'
RETURN      a.session_id, count(b[ ]), b[ ].tag_id
```

```
PATTERN SEQ(MedicineTaken x, MedicineTaken y)
WHERE      [name='John']
AND        [medicine='Antibiotics']
AND        (x.amount + y.amount) > 1000
WITHIN    4 hours
```

```
PATTERN SEQ(News a, Stock+ b[ ])
WHERE      [symbol]
AND        a.type = 'bad'
AND        b[i].symbol = 'GOOG'
AND        b[b.LEN].volume < 80%*b[1].volume
WITHIN    4 hours
RETURN    sum(b[ ].volume)
```

```
PATTERN      SEQ(Stock+ a[])
WHERE        [symbol]
AND          a[1].price = 10
AND          a[i].price > a[i-1].price
AND          a[a.LEN].price = 20
AND          avg(a[.].volume) a[1].volume
WITHIN      1 hour
RETURN       a[1].symbol, a[.].price
```

```
PATTERN SEQ(Stock+ a[], Stock b)
WHERE        [symbol]
AND          a[1].volume > 1000
AND          a[i].price > avg(a[...i-1].price)
AND          b.volume < 80% * a[a.LEN].volume
WITHIN      1 hour
RETURN       a[1].symbol, a[.](price,volume), b.(price,volume)
```

```
PATTERN SEQ(Scan a, ~(Scan+ b[]), Scan c)
WHERE        [object_id]
AND          a.location = "New York"
AND          c.location = "Amherst"
AND          b[1].location = a.next
AND          b[i].location = b[i-1].next
AND          c.location = b[b.LEN].next
AND          b.LEN = 3
RETURN       c.object_id, c.courier_id
```

```
PATTERN SEQ(Alert a, Shipment+ b[ ])
WHERE        a.type = 'contaminated'
AND          b[1].from = a.site
AND          b[i].from = b[i-1].to
WITHIN      3 hours
RETURN       a.type, a.site, b[ ].to
```