Incremental Graph Queries with openCypher

Gábor Szárnyas

GraphDevroom @ FOSDEM 2017

Budapest University of Technology and Economics
McGill University, Montréal

Incremental Graph Queries with openCypher

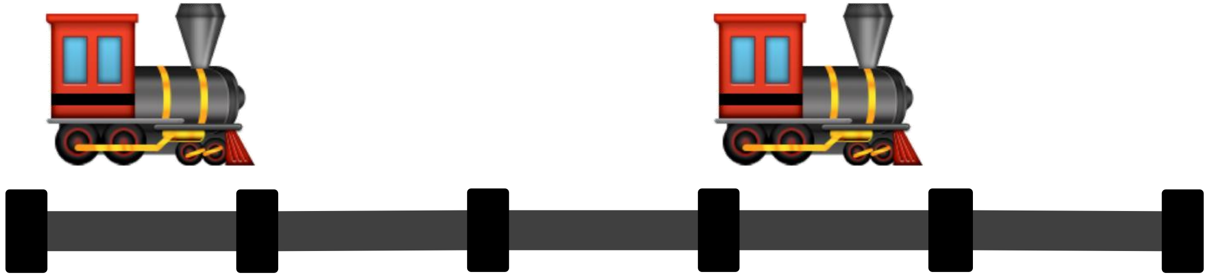
Incremental Graph Queries

with openCypher

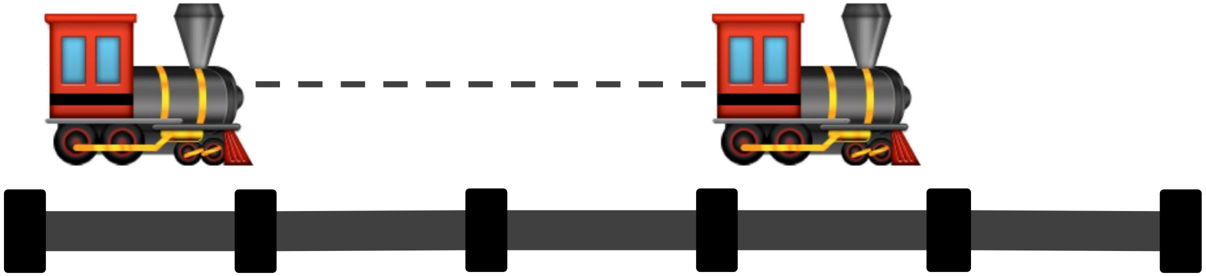
Live railway model

Live railway model

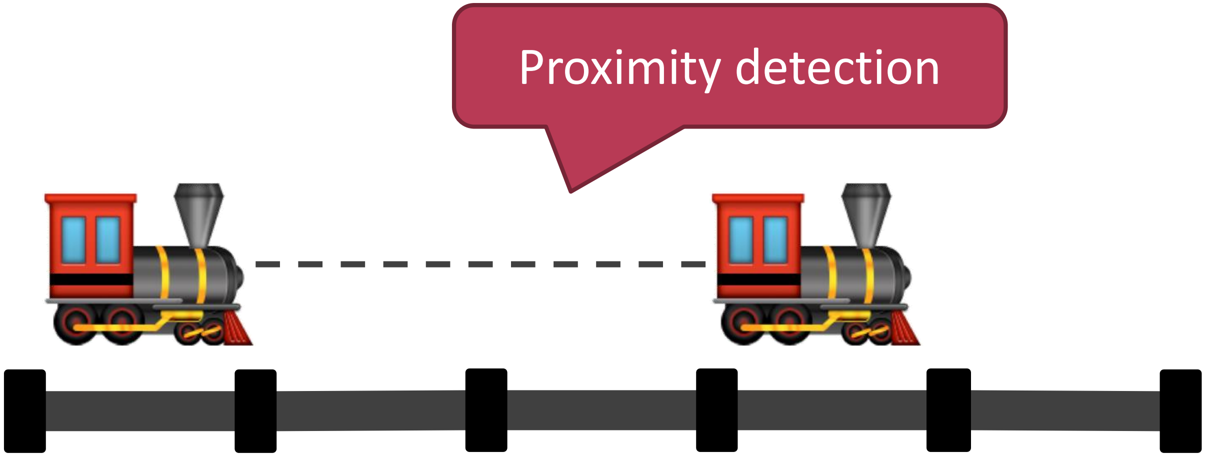
Live railway model



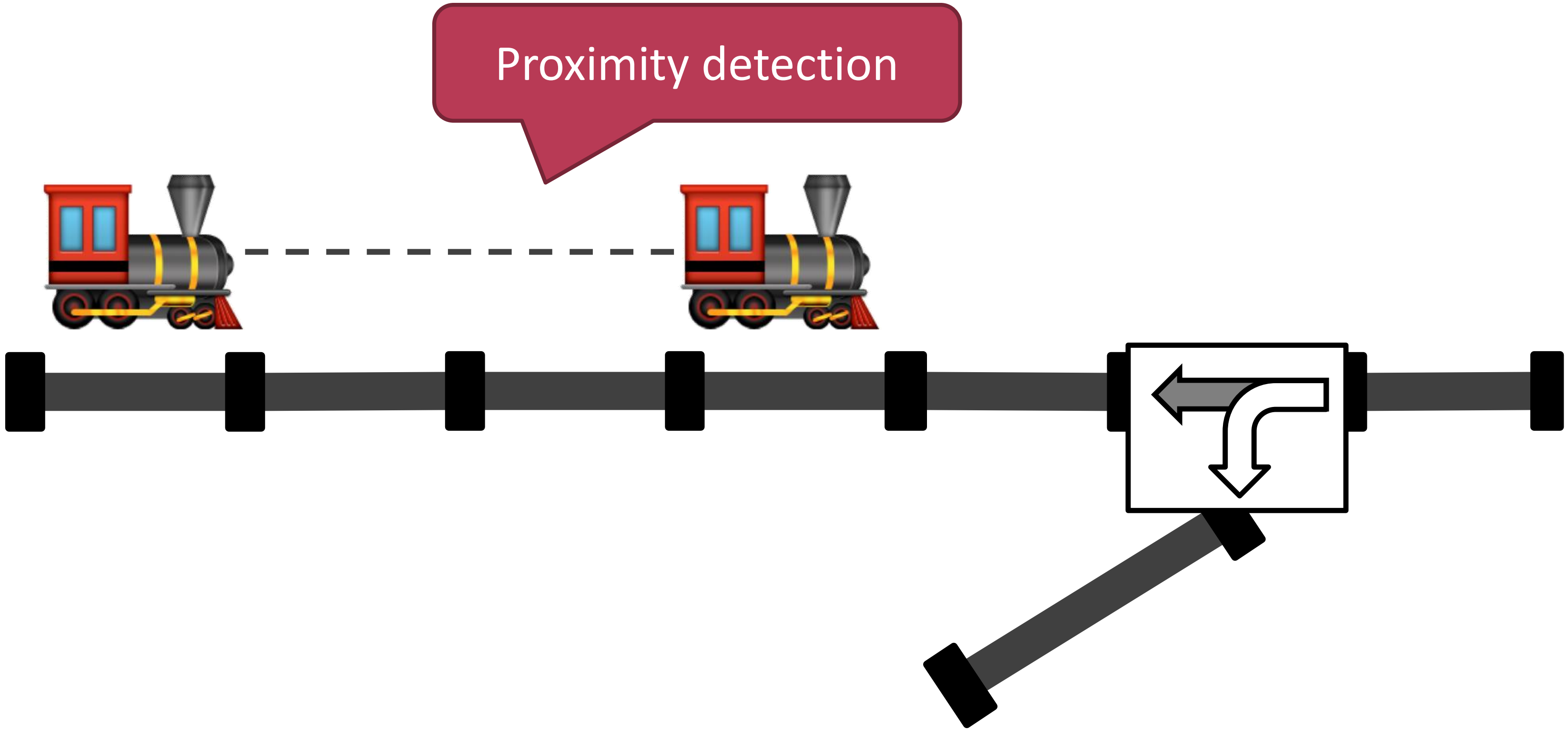
Live railway model



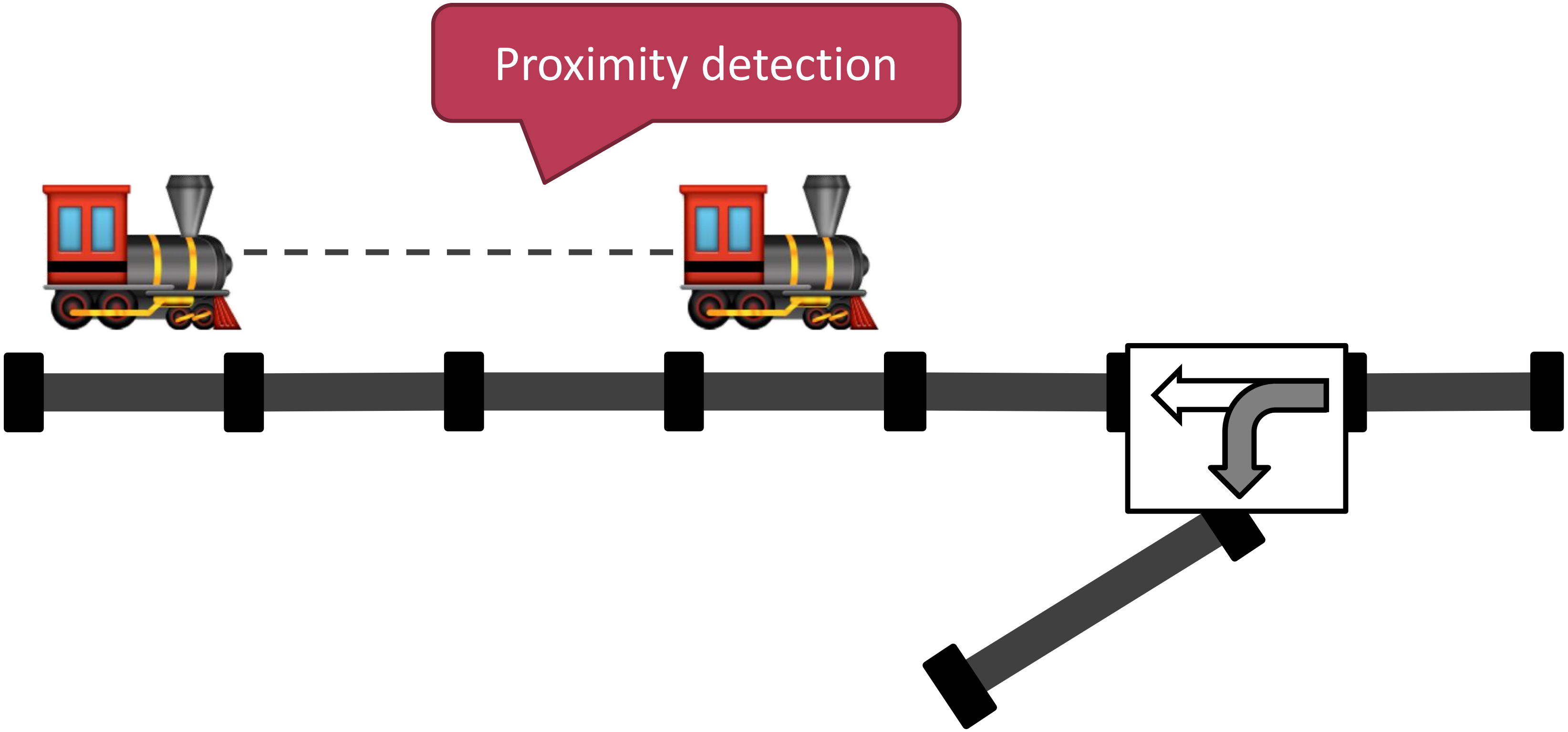
Live railway model



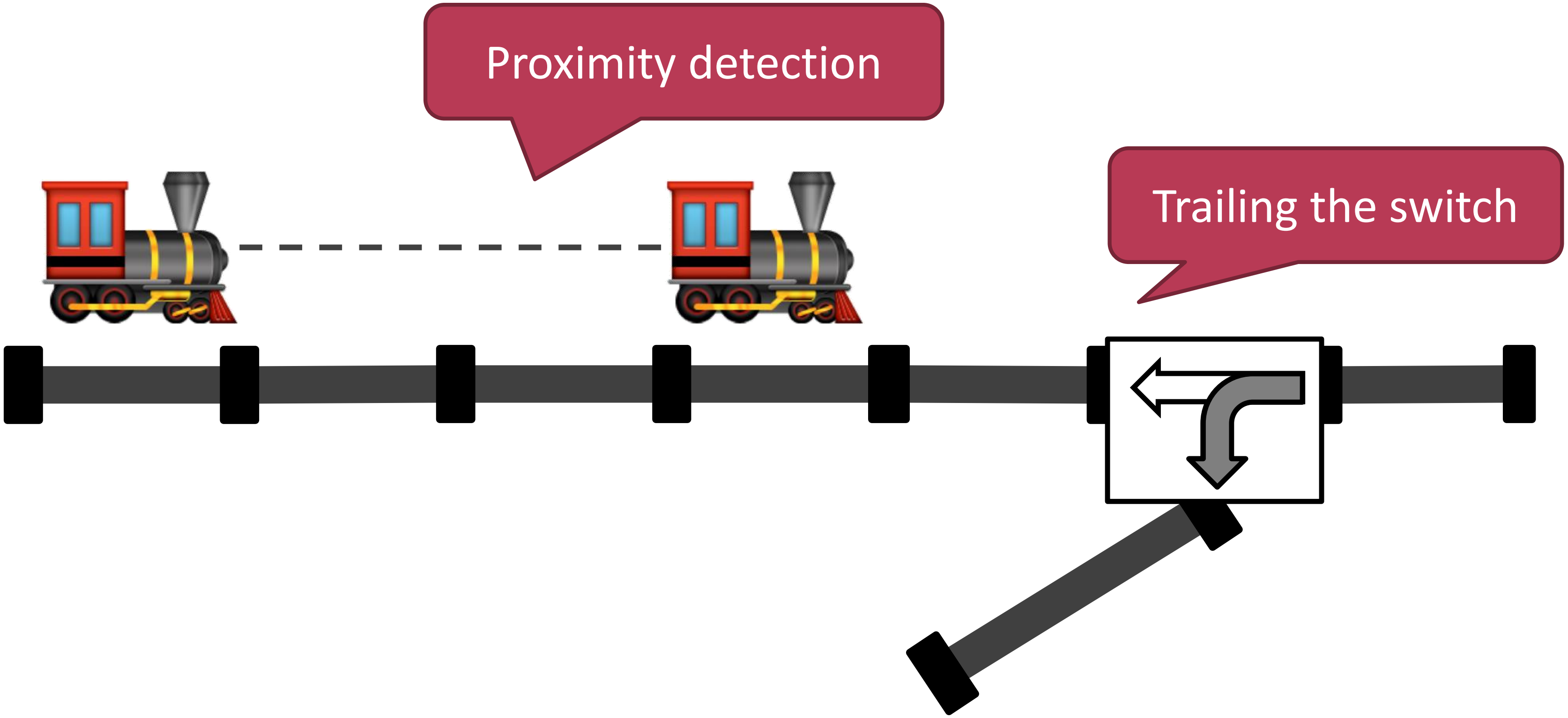
Live railway model



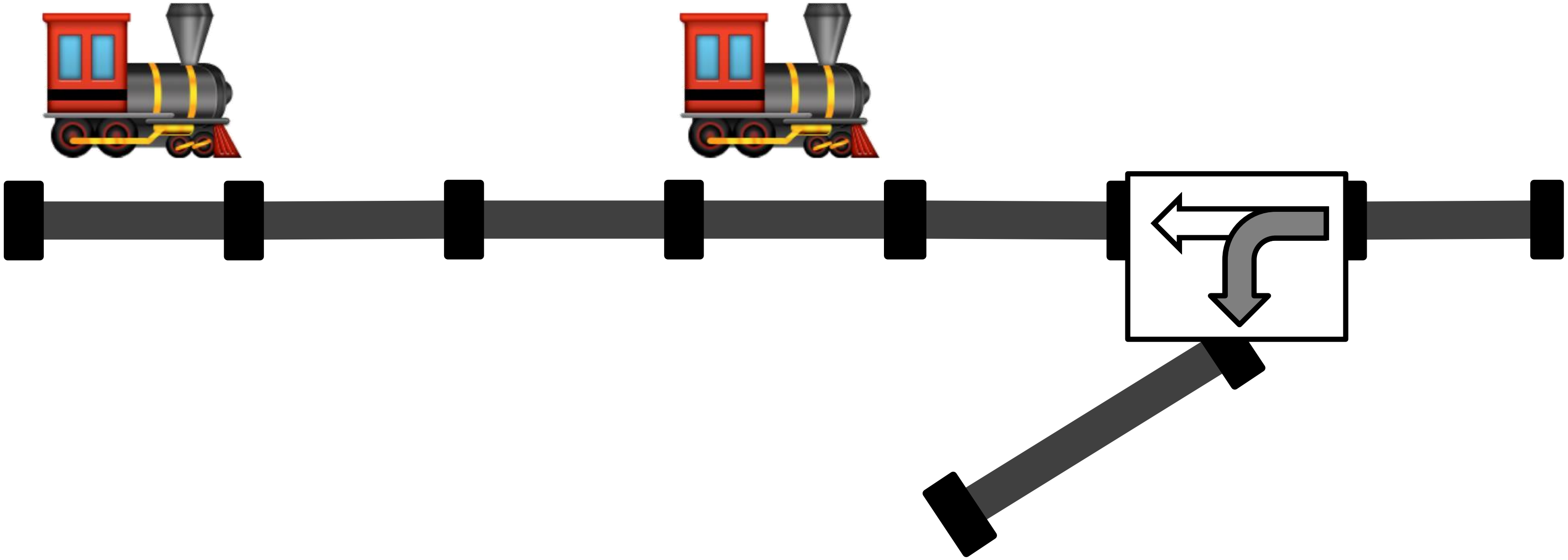
Live railway model



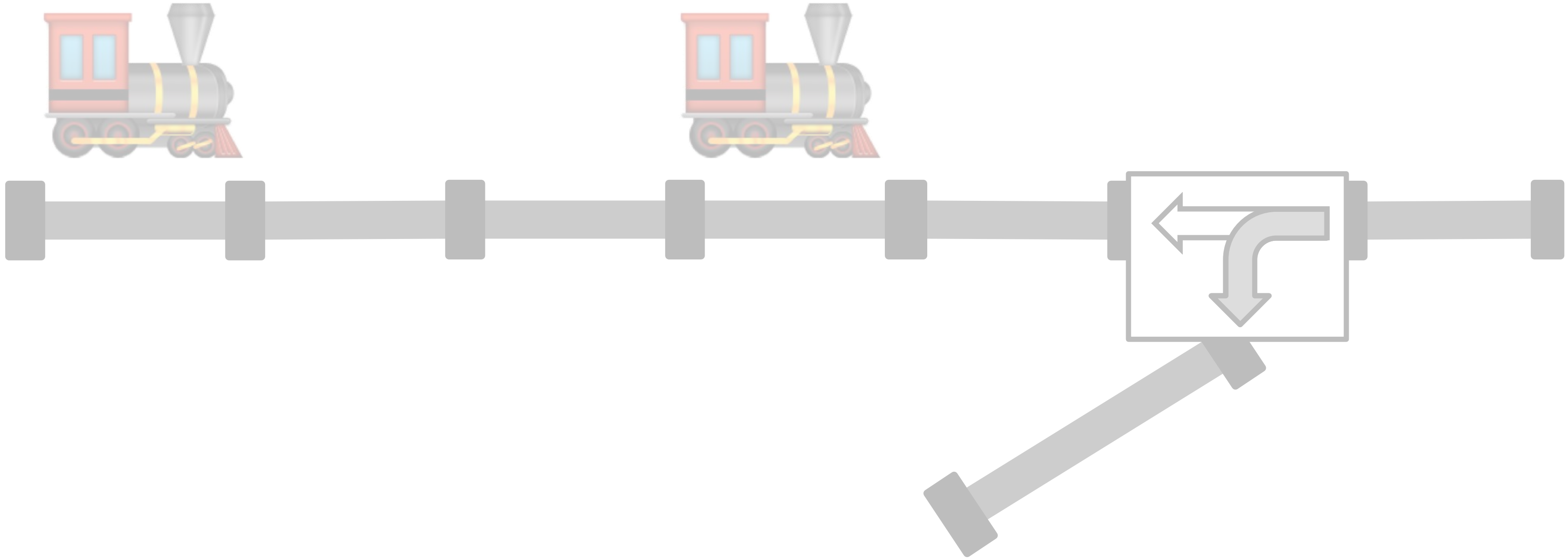
Live railway model



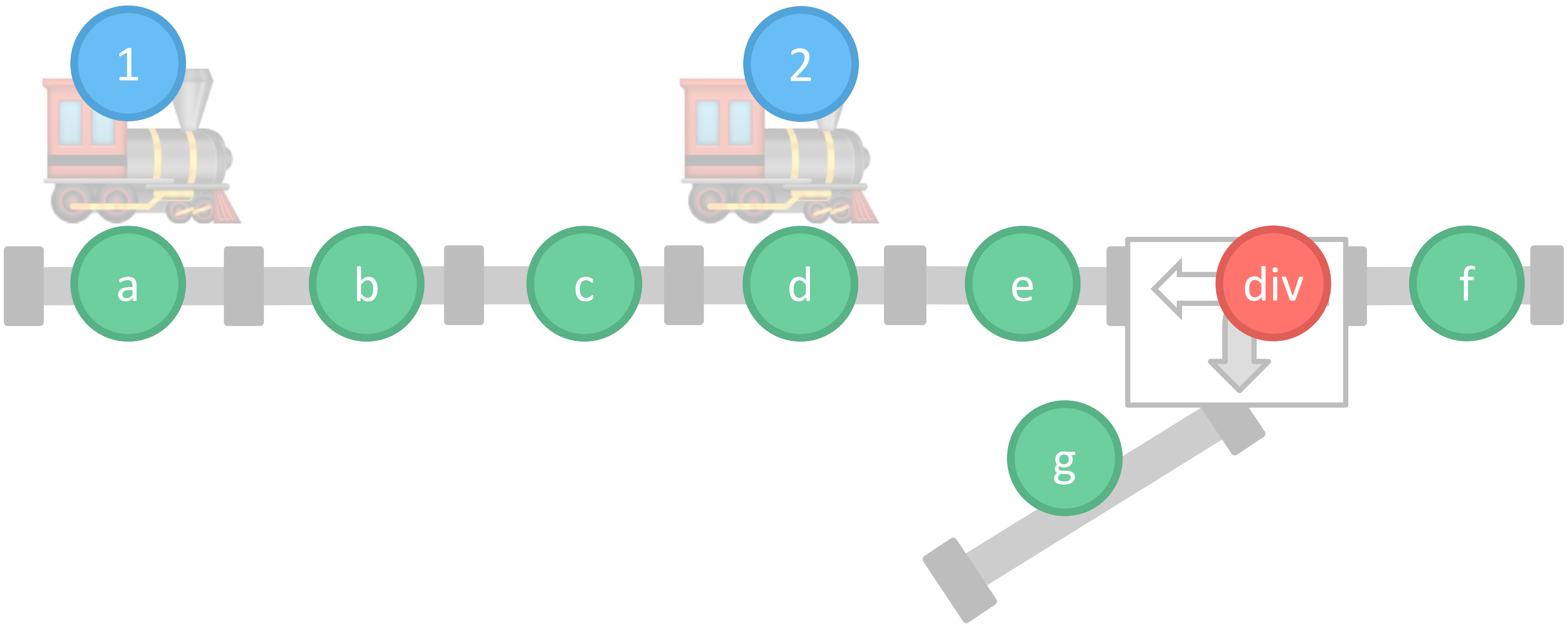
Live railway model



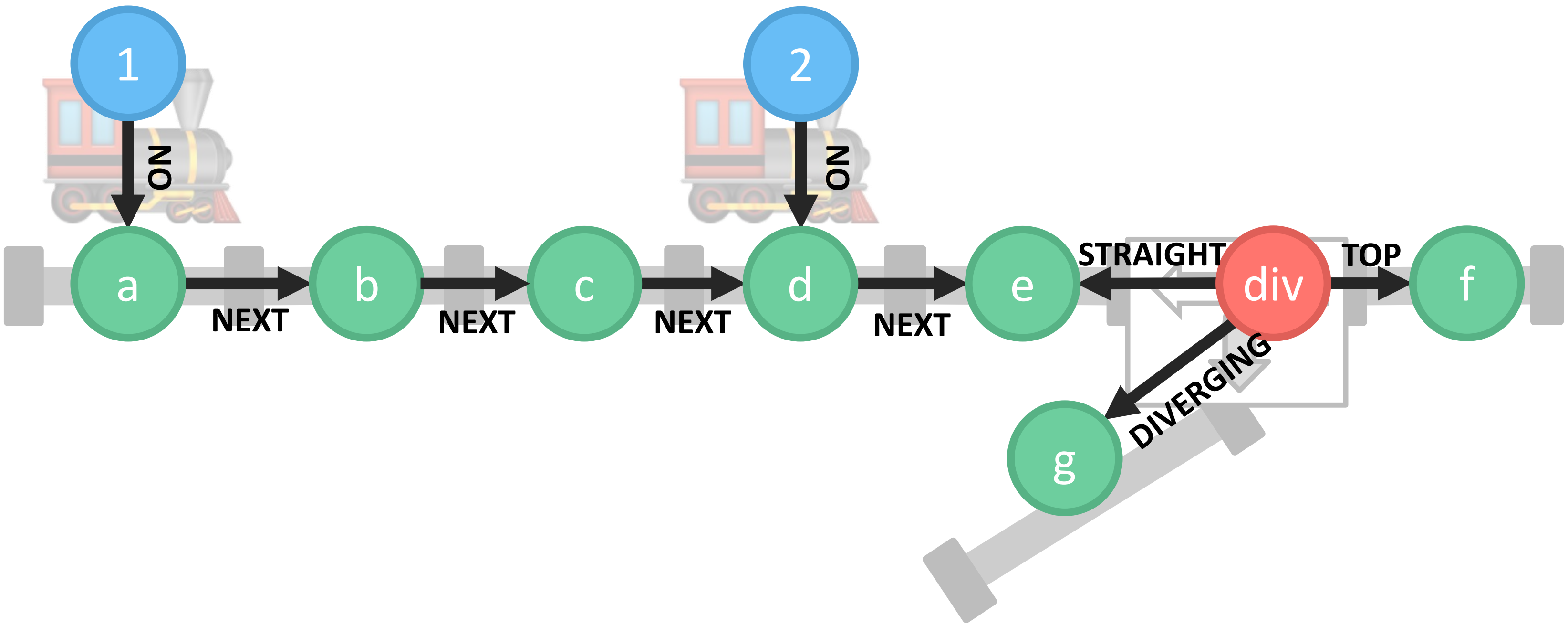
Live railway model



Live railway model

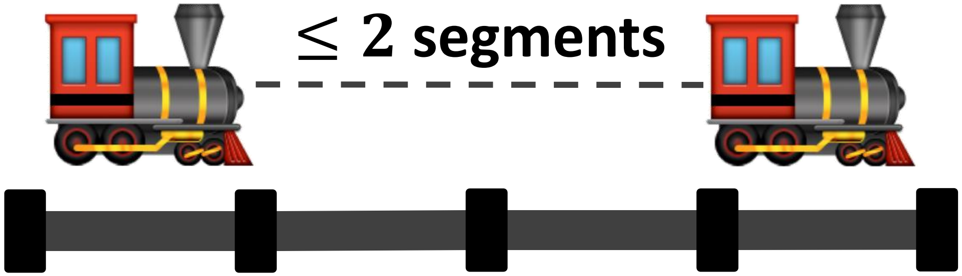


Live railway model



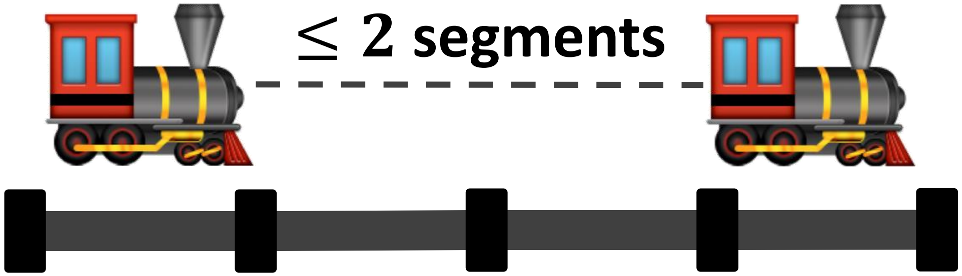
Proximity detection

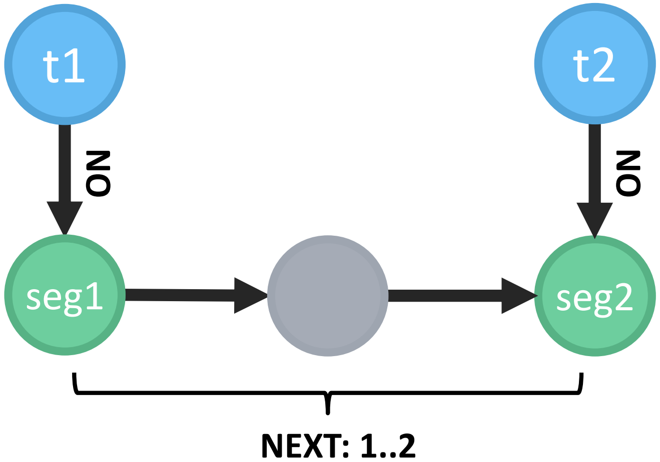
Proximity detection



Proximity detection

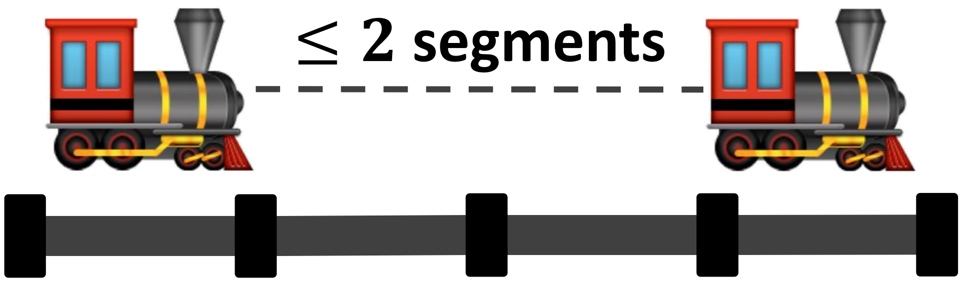
Proximity detection

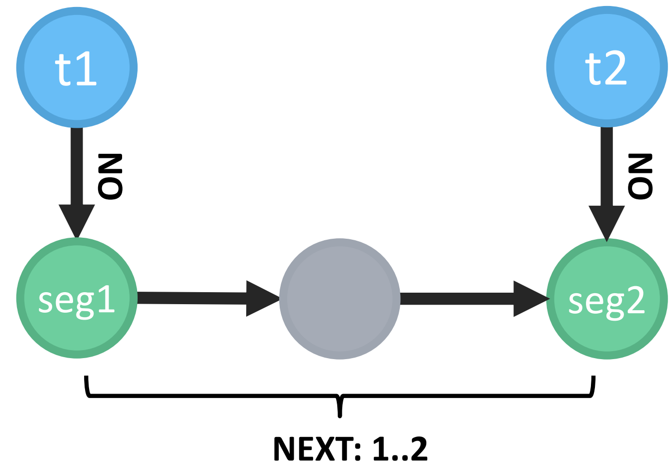




Proximity detection

Proximity detection

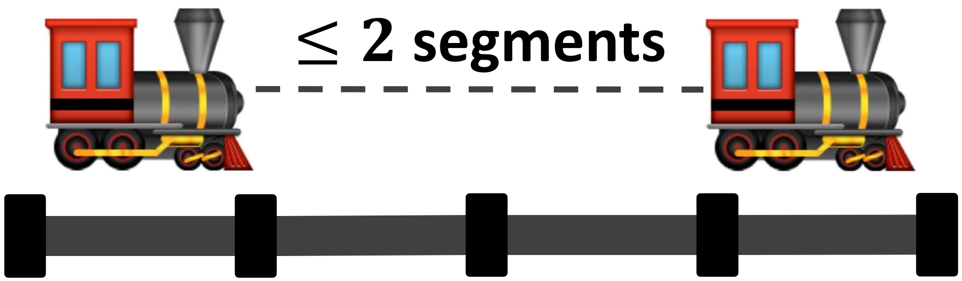


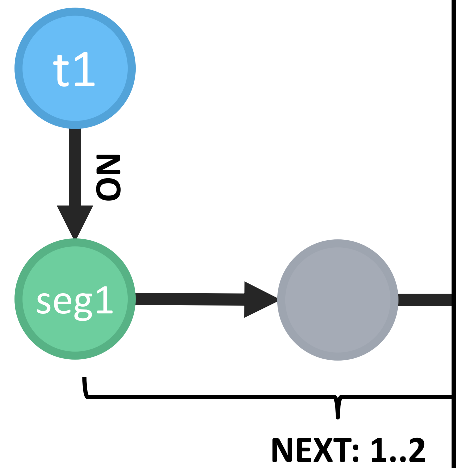


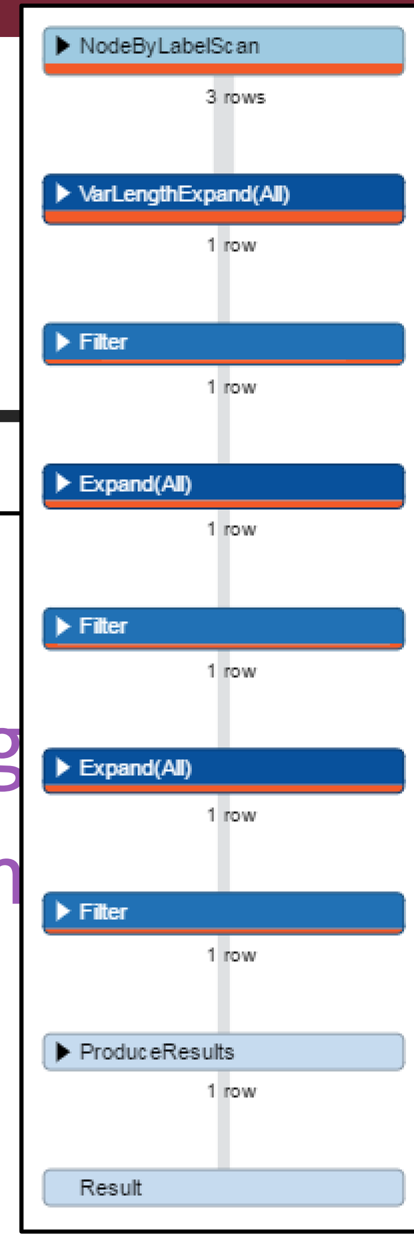

```
MATCH
   (t1:Train)-[:ON]->(seg1:Segment)
   -[:NEXT*1..2]->(seg2:Segment)
   <-[:ON]-(t2:Train)
RETURN t1, t2, seg1, seg2
```

Proximity detection

Proximity detection

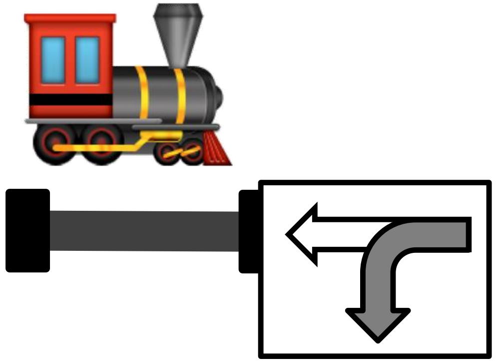




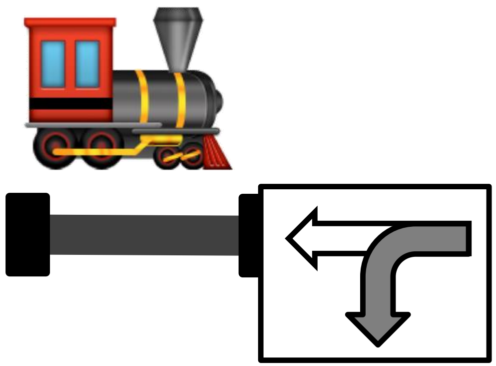


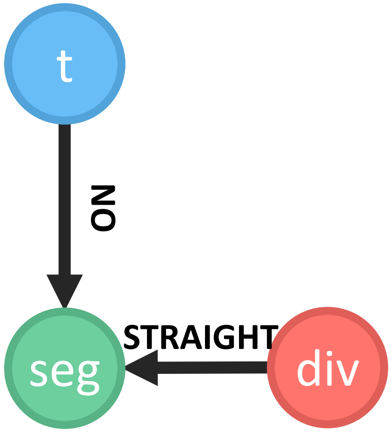

```
MATCH
   (t1:Train)-[:ON]->(seg1:Seg
   -[:NEXT*1..2]->(seg2:Segmen
   <-[:ON]-(t2:Train)
RETURN t1, t2, seg1, seg2
```

Trailing the switch

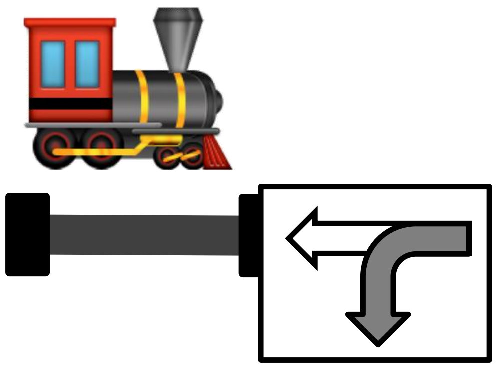


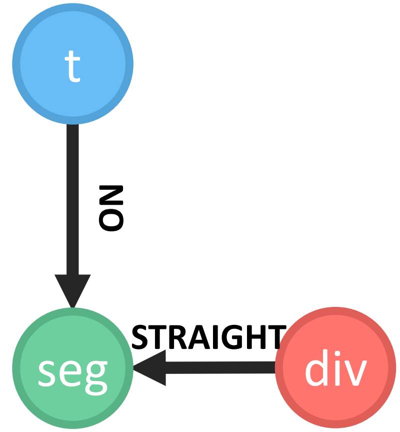
Trailing the switch





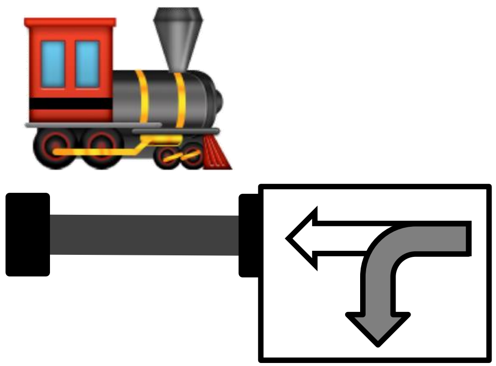
Trailing the switch

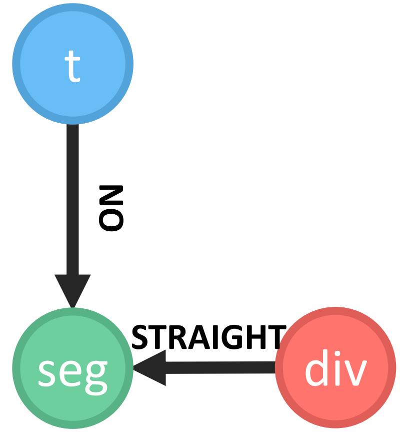




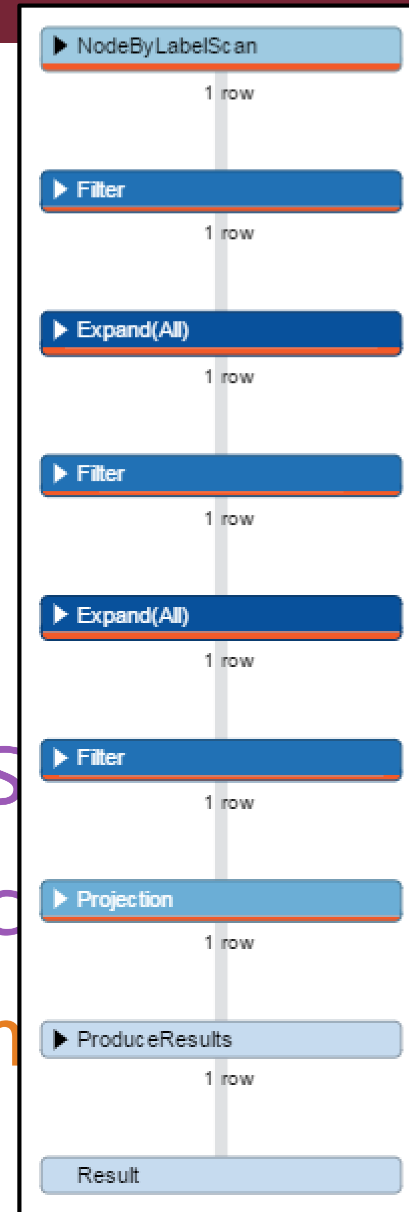

```
MATCH (t:Train)-[:ON]->(seg:Segment)
      <-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw
```

Trailing the switch

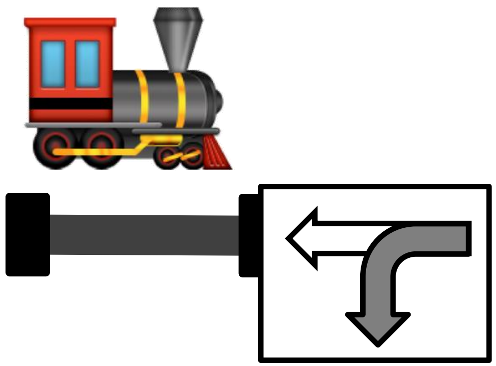


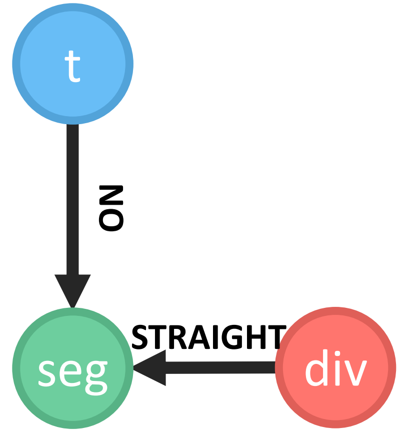


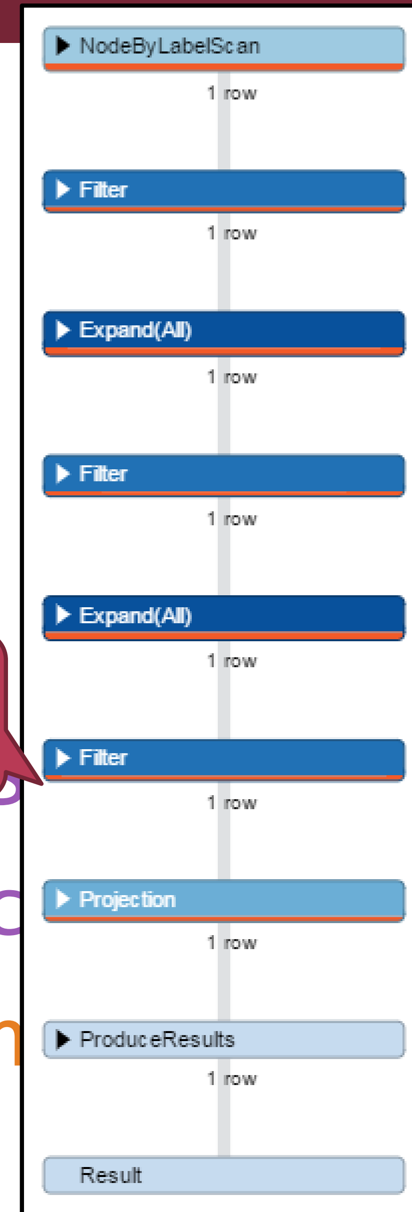

```
MATCH (t:Train)-[:ON]->(seg:S
      <-[:STRAIGHT]-(sw:Switc
WHERE sw.position = 'divergin
RETURN t.number, sw
```



Trailing the switch







Evaluate continuously

```
MATCH (t:Train)-[:ON]
      <-[:STRAIGHT]-(sw:Switc
WHERE sw.position = 'divergin
RETURN t.number, sw
```

Incremental queries

Incremental queries

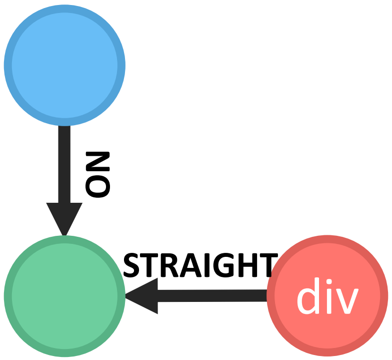
- Register a set of **standing queries**
- **Continuously evaluate** queries on changes

Incremental queries

- Register a set of **standing queries**
- **Continuously evaluate** queries on changes

- The **Rete algorithm** (1974)
  - Originally for rule-based expert systems
  - Indexes the graph and caches interim query results

Trailing the switch



$\pi_{t.number,\ sw}$

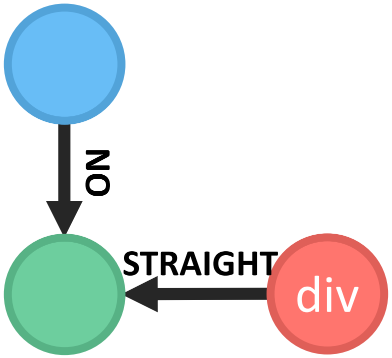
$\sigma_{sw.position\ =\ 'diverging'}$

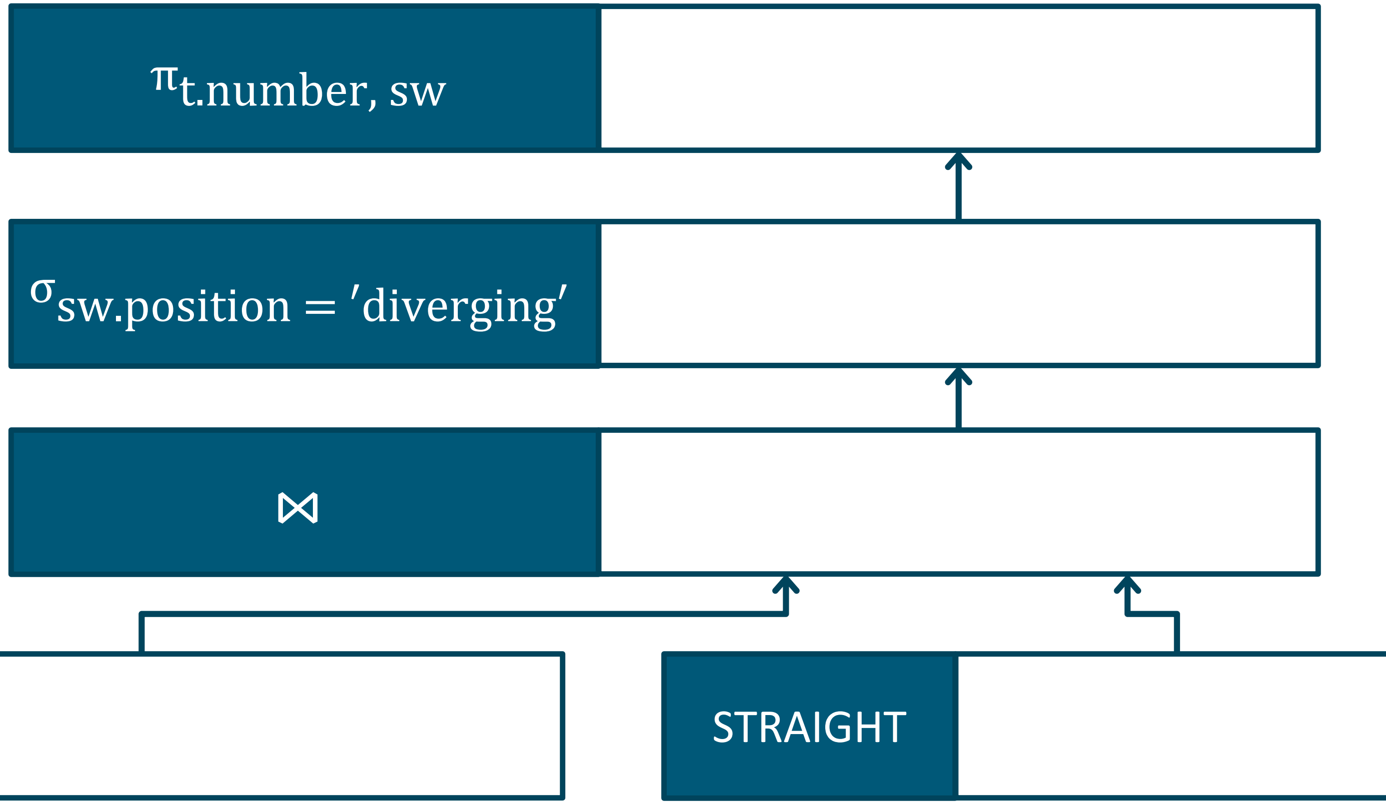
$\bowtie$

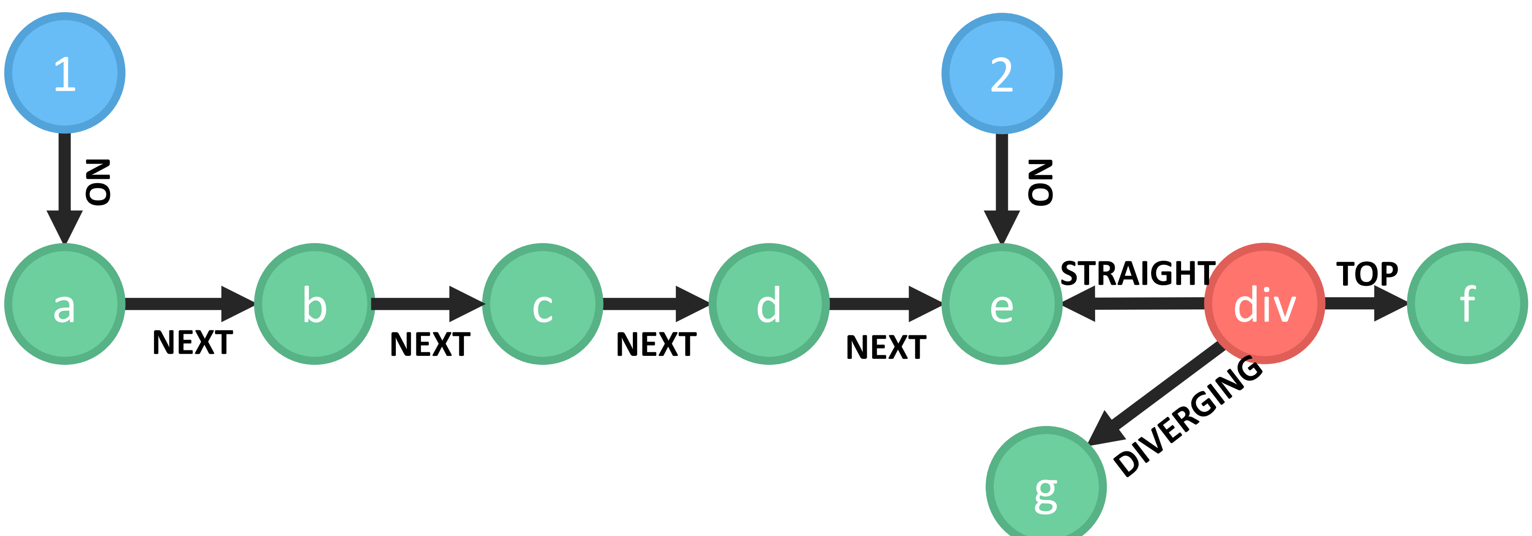
ON

STRAIGHT

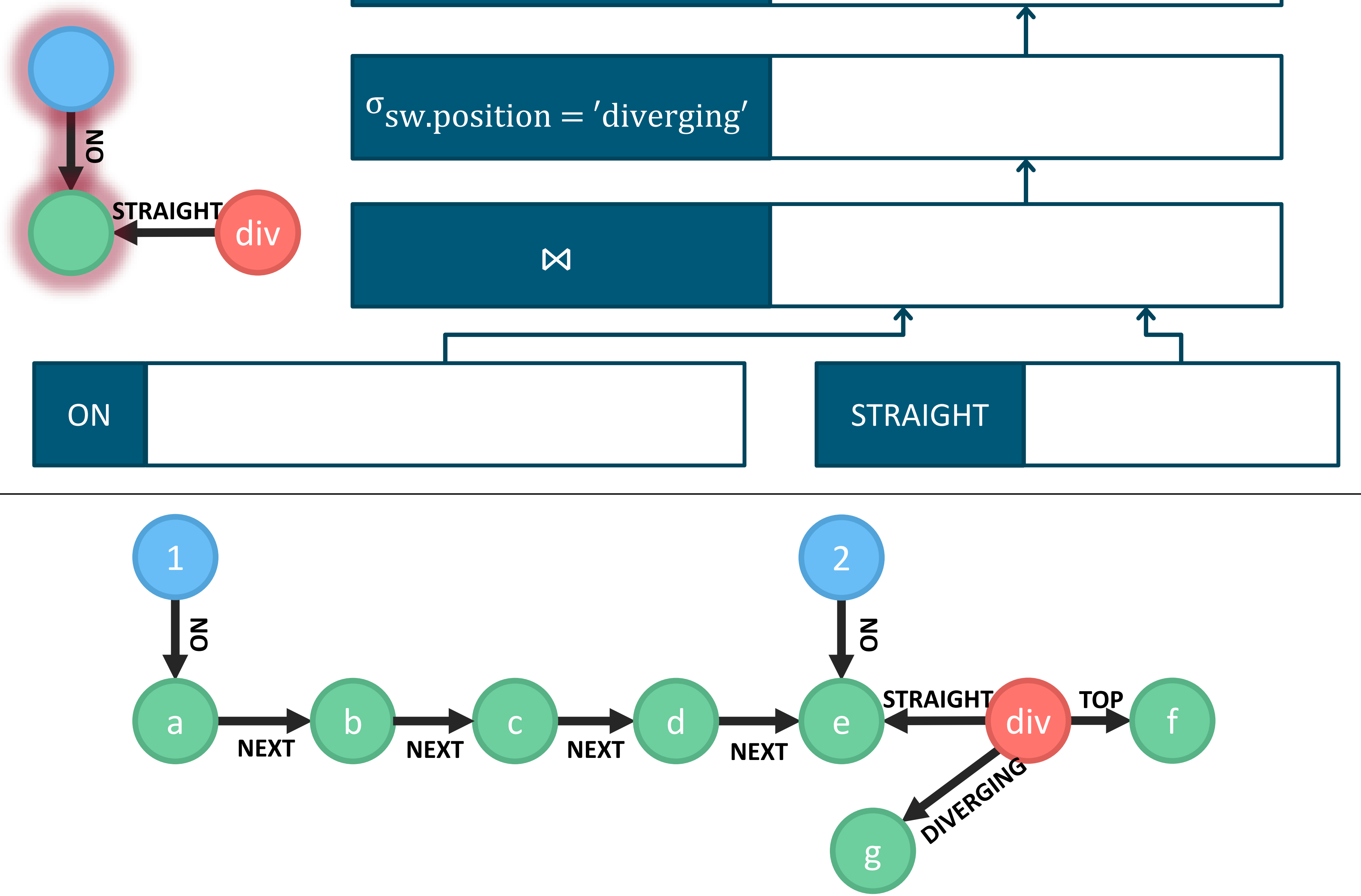
Trailing the switch

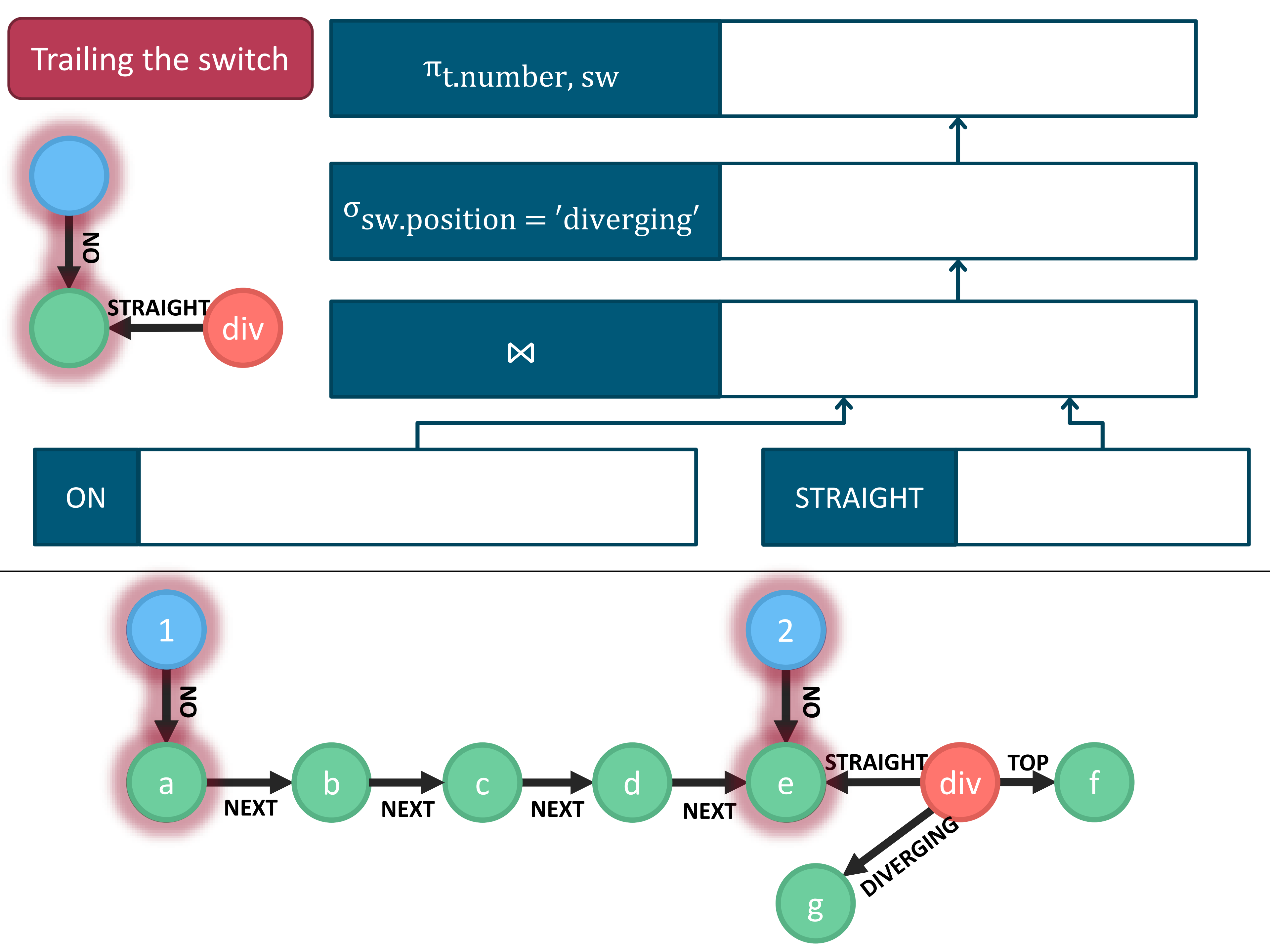




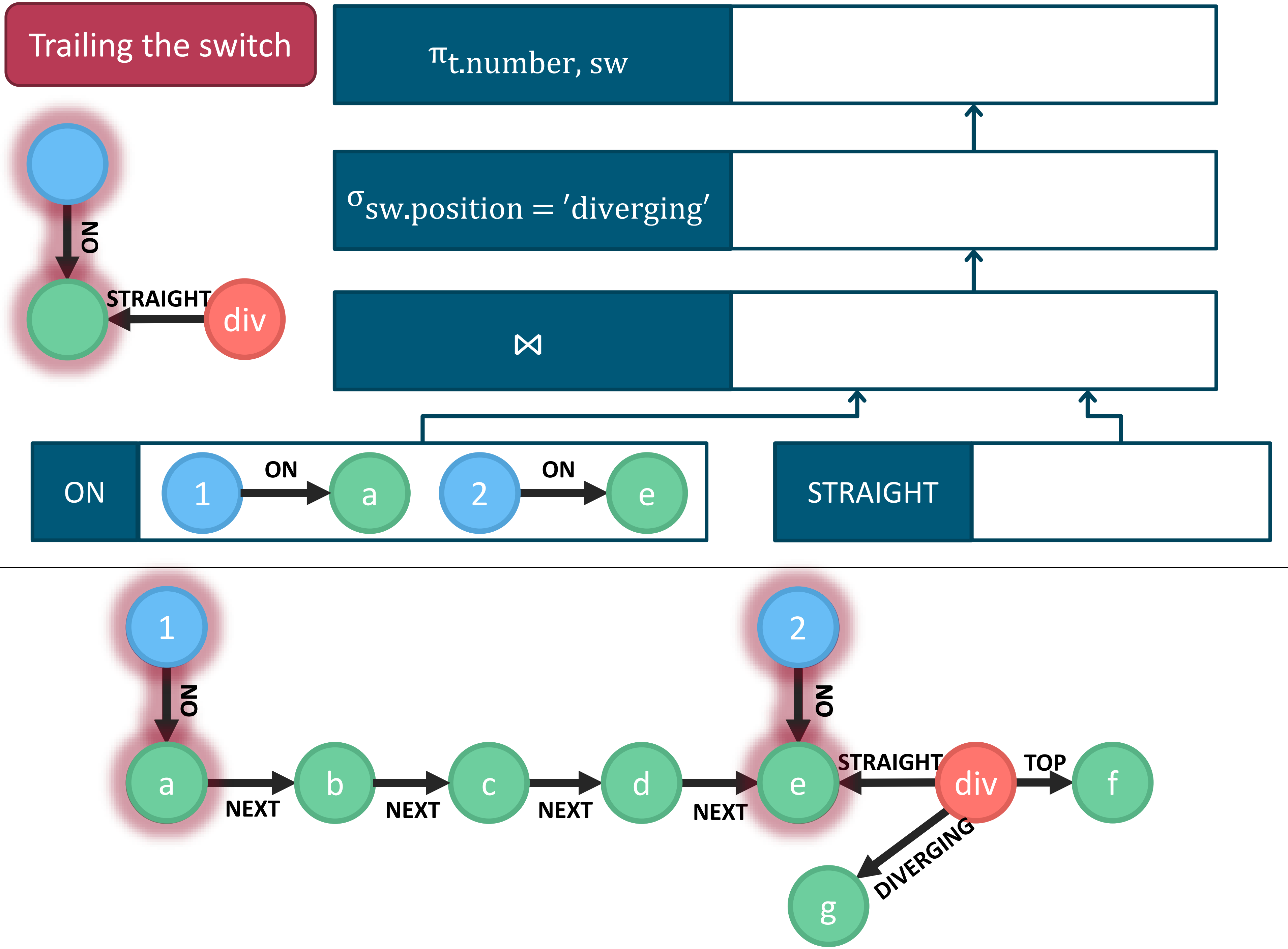


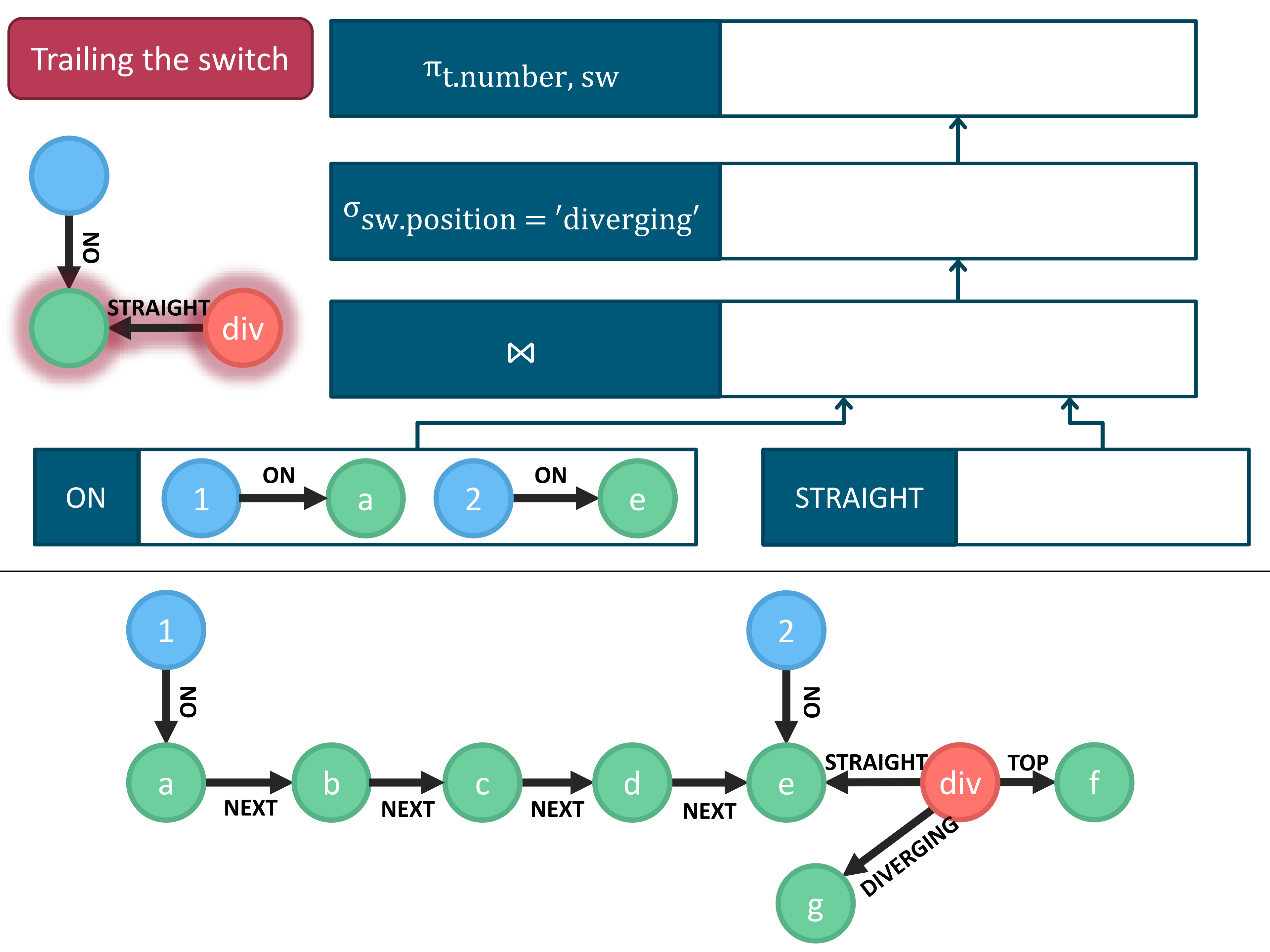
Trailing the switch



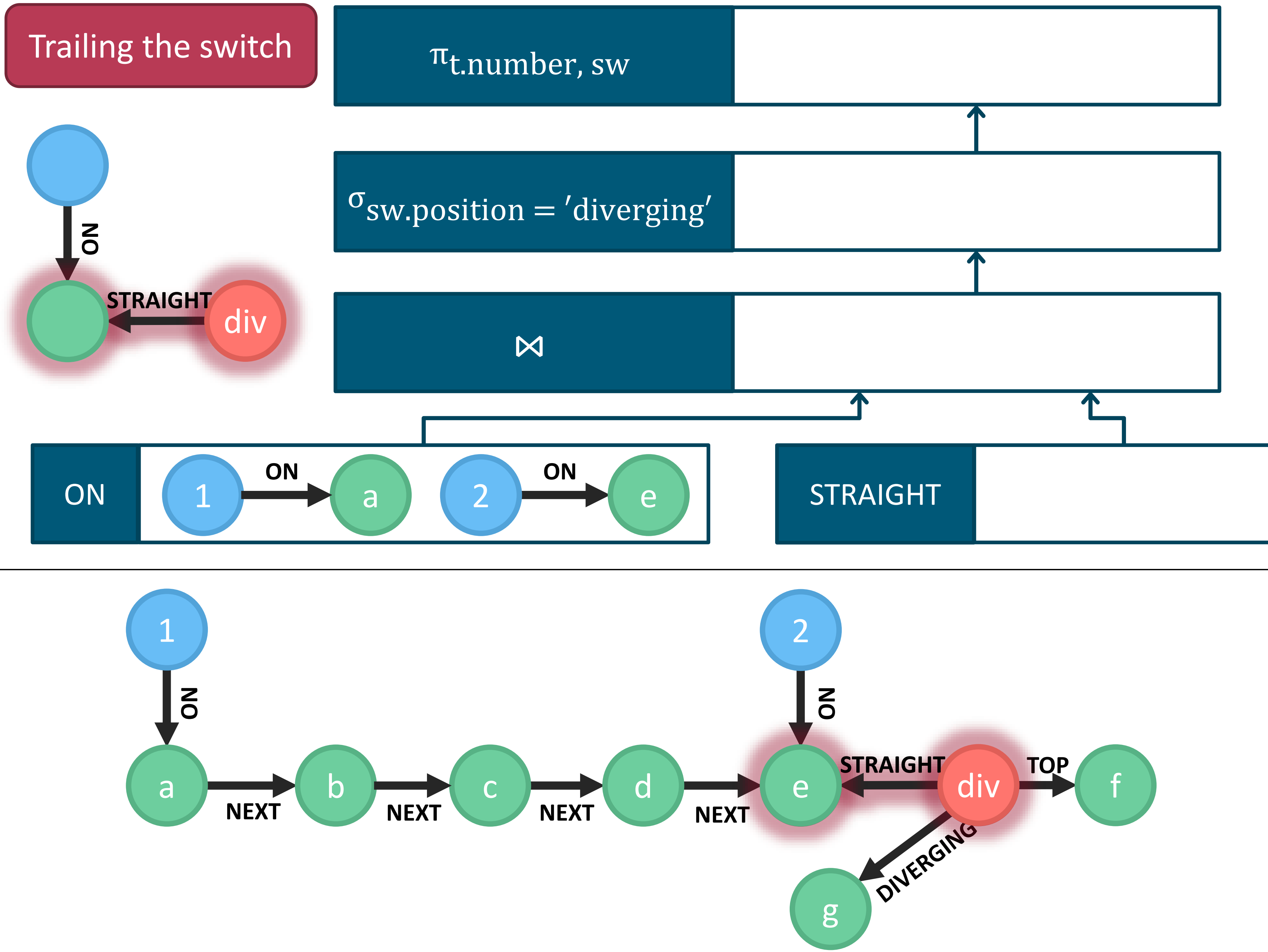

Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position\ =\ 'diverging'}$
$\bowtie$
ON
STRAIGHT
ON
STRAIGHT
div
1
2
ON
ON
a
b
c
d
e
div
f
g
NEXT
NEXT
NEXT
NEXT
STRAIGHT
TOP
DIVERGING

Trailing the switch

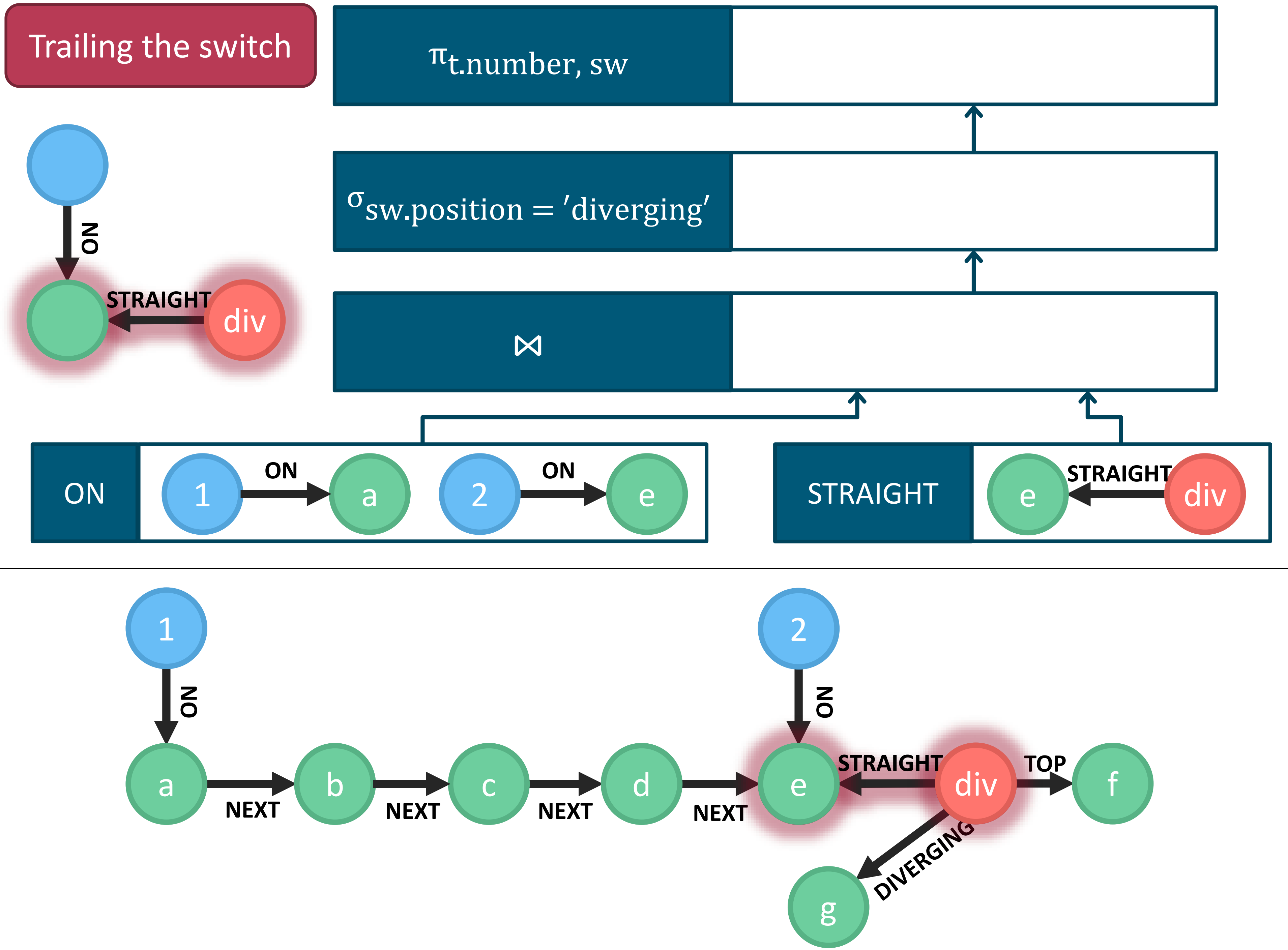




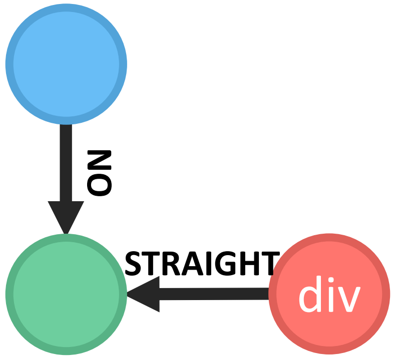
Trailing the switch
$\pi_{t.number,\, sw}$
$\sigma_{sw.position\, =\, 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
ON
1
ON
a
2
ON
e
STRAIGHT
1
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
2
ON
e
STRAIGHT
div
TOP
f
DIVERGING
g

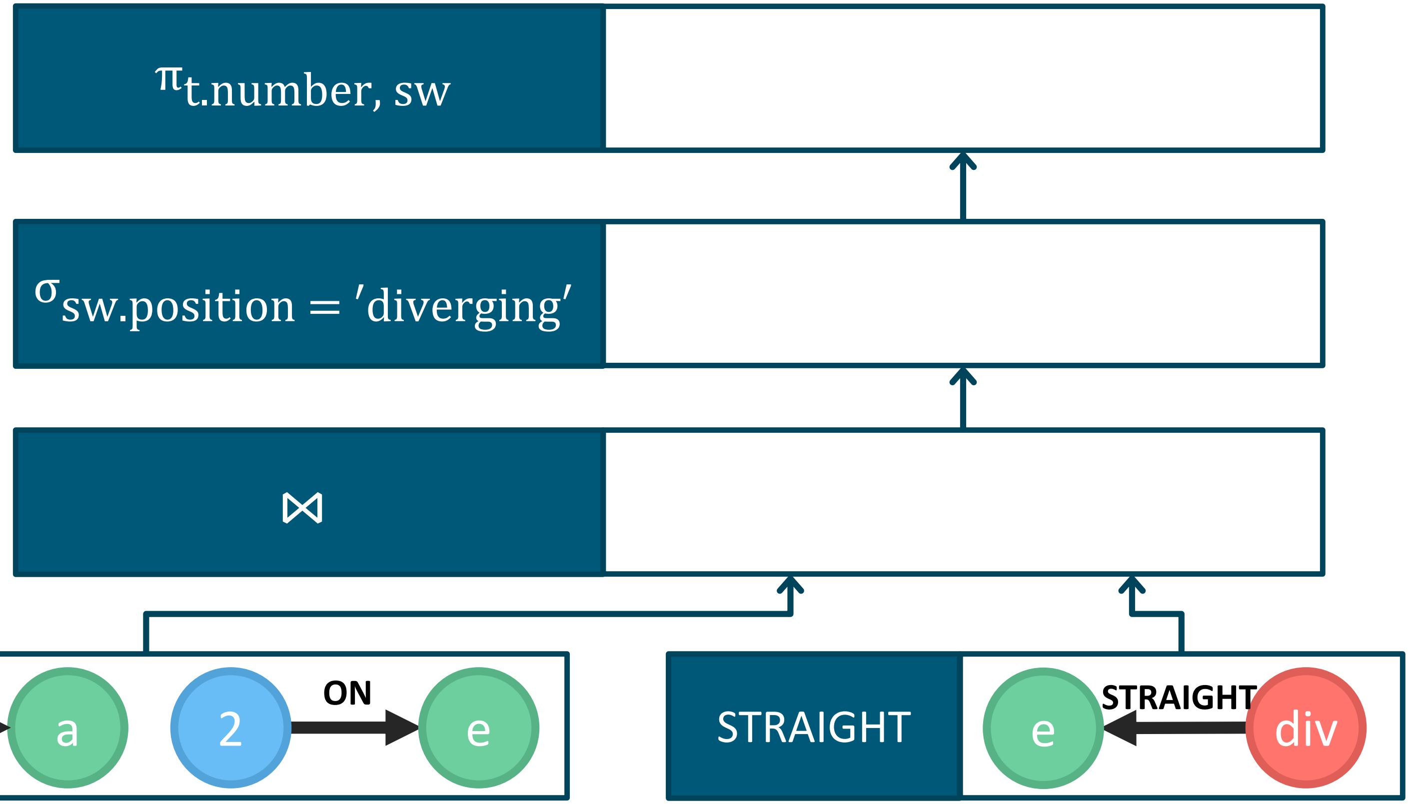


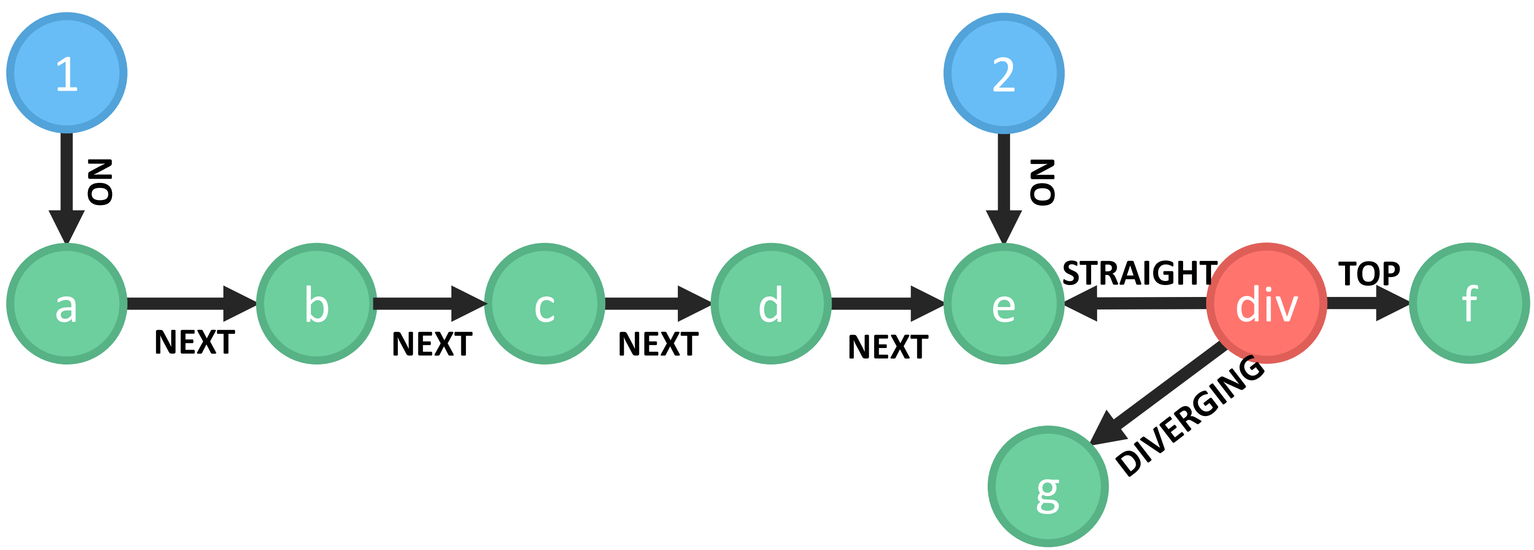
Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position\ =\ 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
ON
1
ON
a
2
ON
e
STRAIGHT
1
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
2
ON
e
STRAIGHT
div
TOP
f
DIVERGING
g

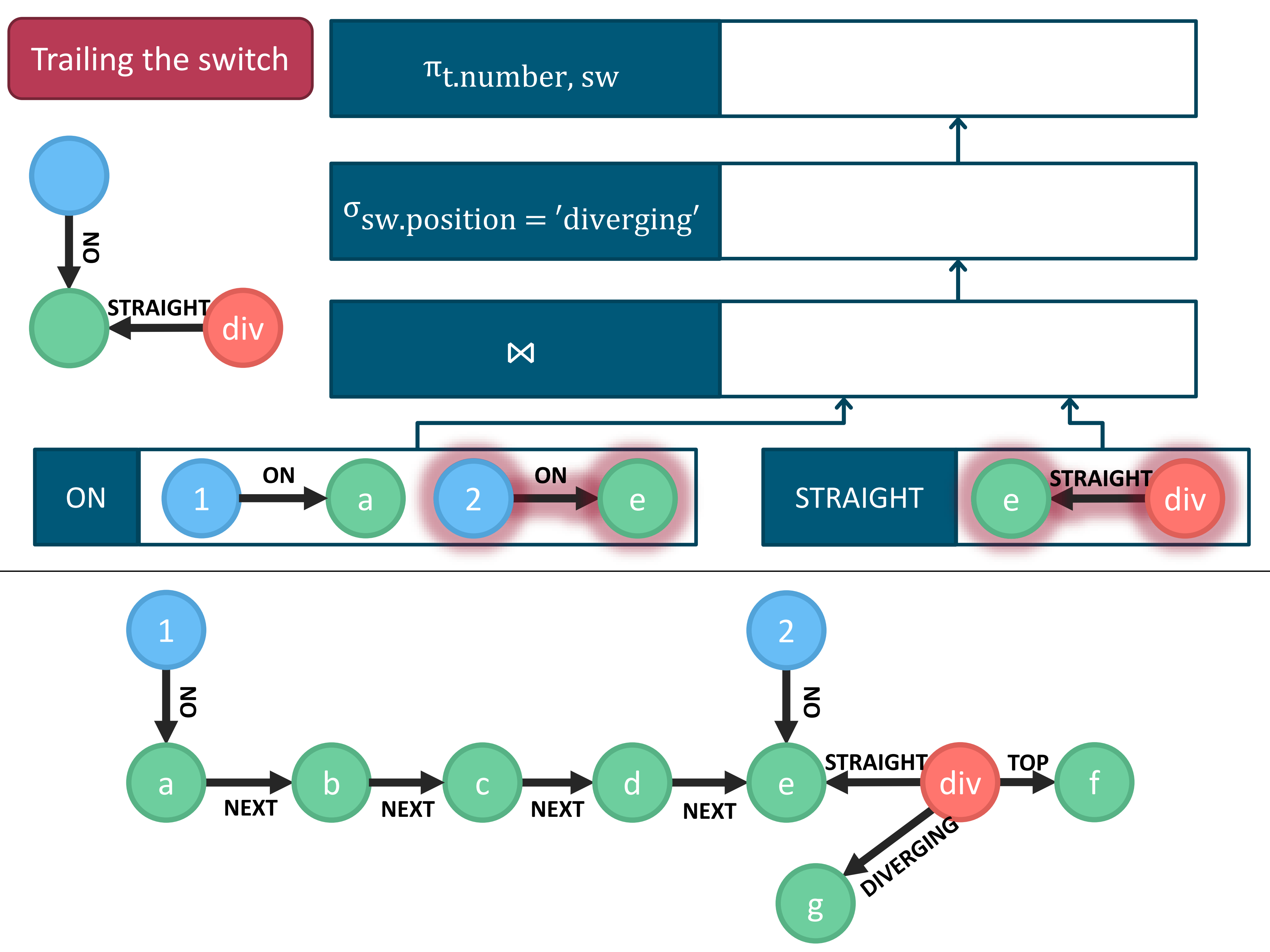

Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position\ =\ 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
ON
1
a
2
e
STRAIGHT
1
a
NEXT
b
c
d
2
e
div
TOP
f
DIVERGING
g

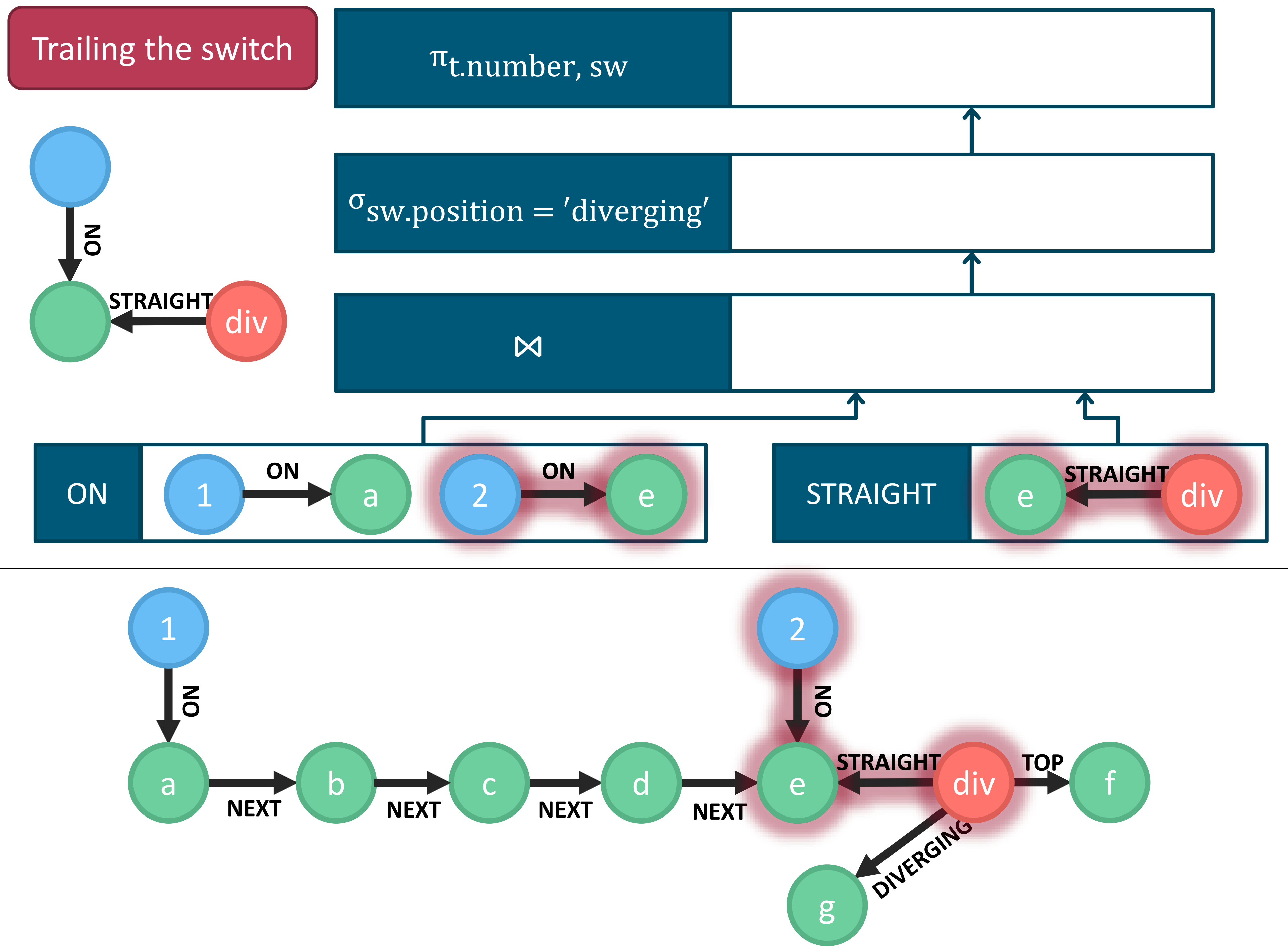
Trailing the switch





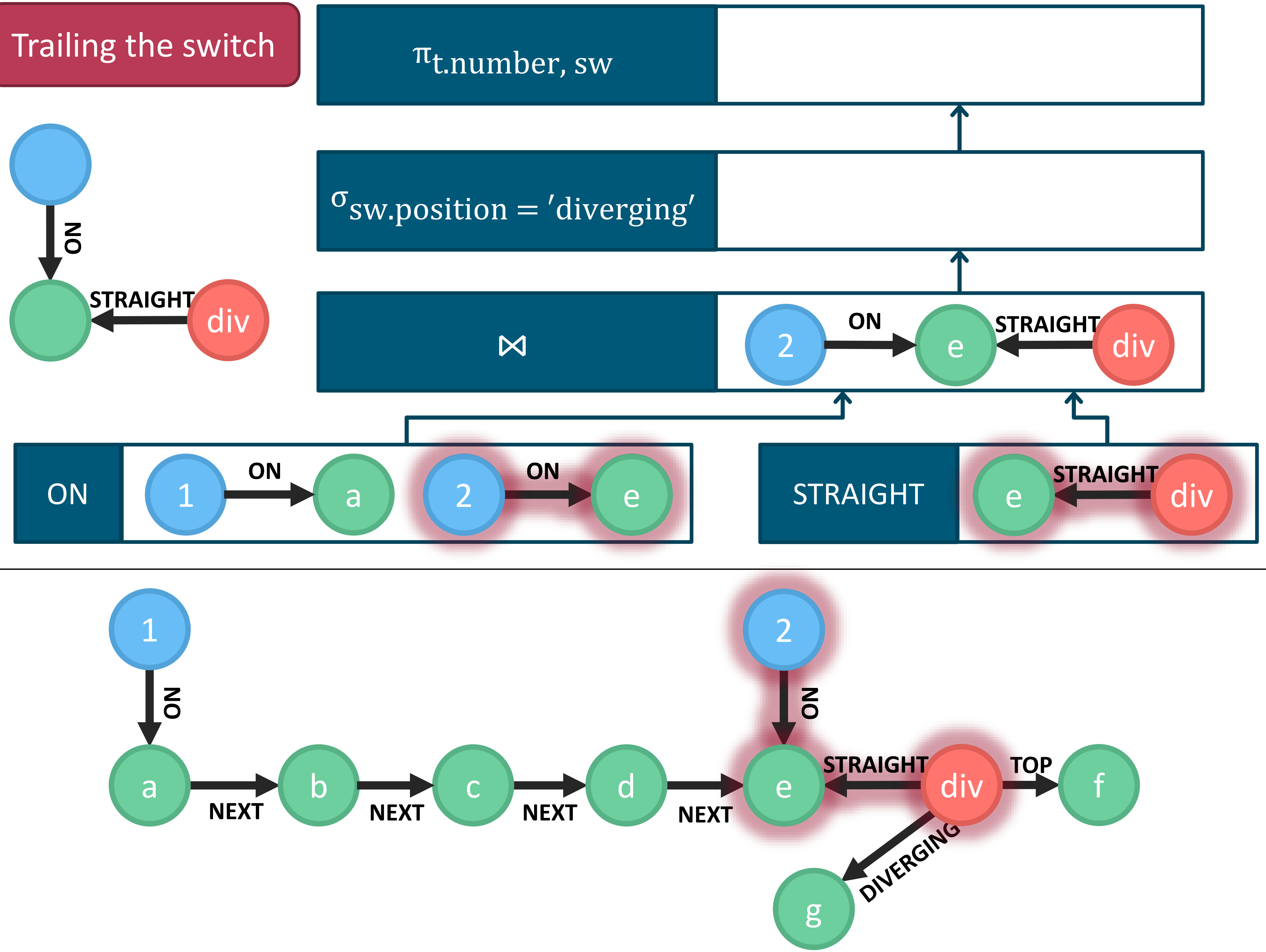


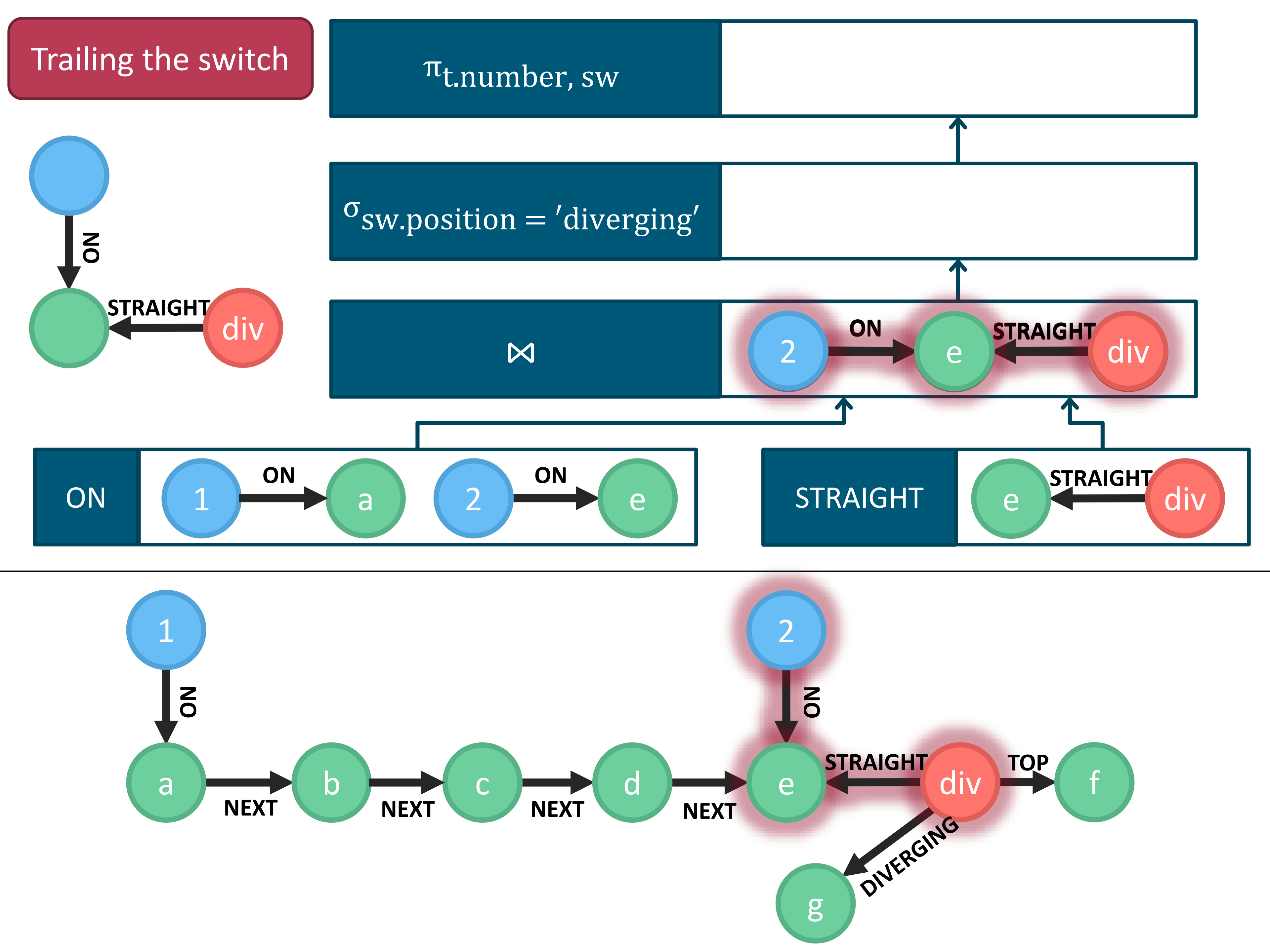

Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position\ =\ 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
ON
1
ON
a
2
ON
e
STRAIGHT
e
STRAIGHT
div
1
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
2
ON
e
STRAIGHT
div
TOP
f
DIVERGING
g



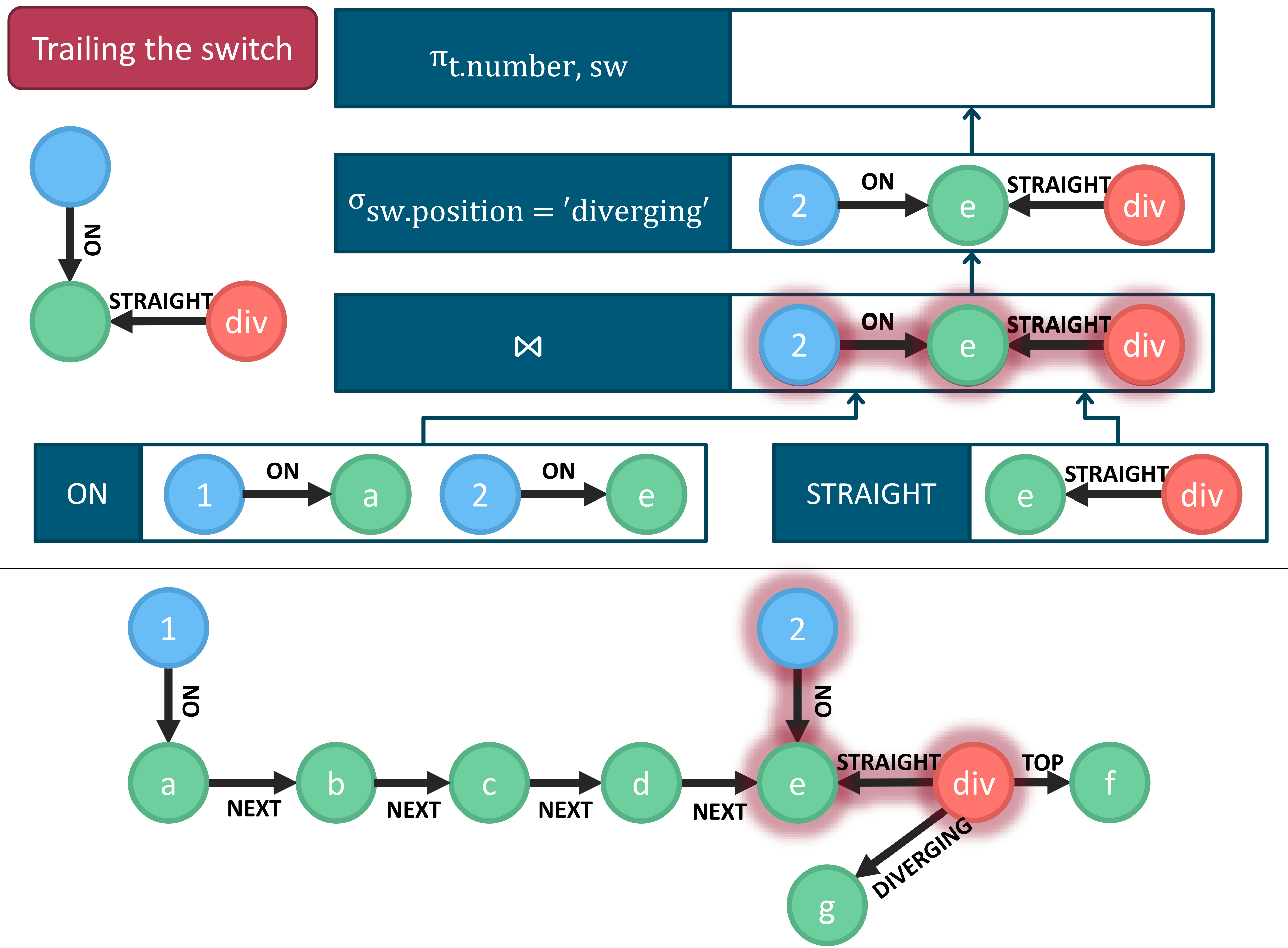
Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position\ =\ 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
ON
1
ON
a
2
ON
e
STRAIGHT
e
STRAIGHT
div
1
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
e
2
ON
STRAIGHT
div
TOP
f
DIVERGING
g

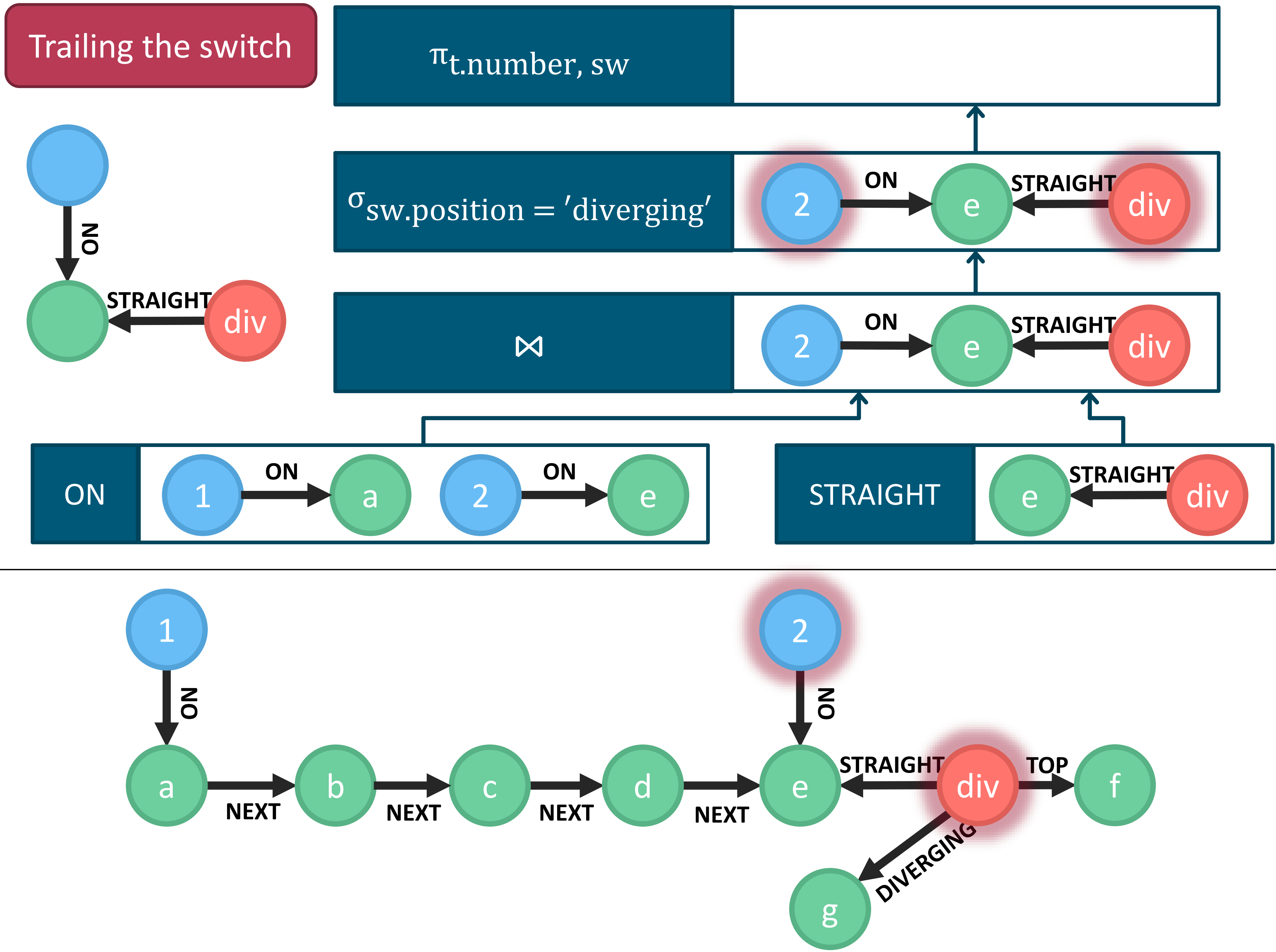
Trailing the switch

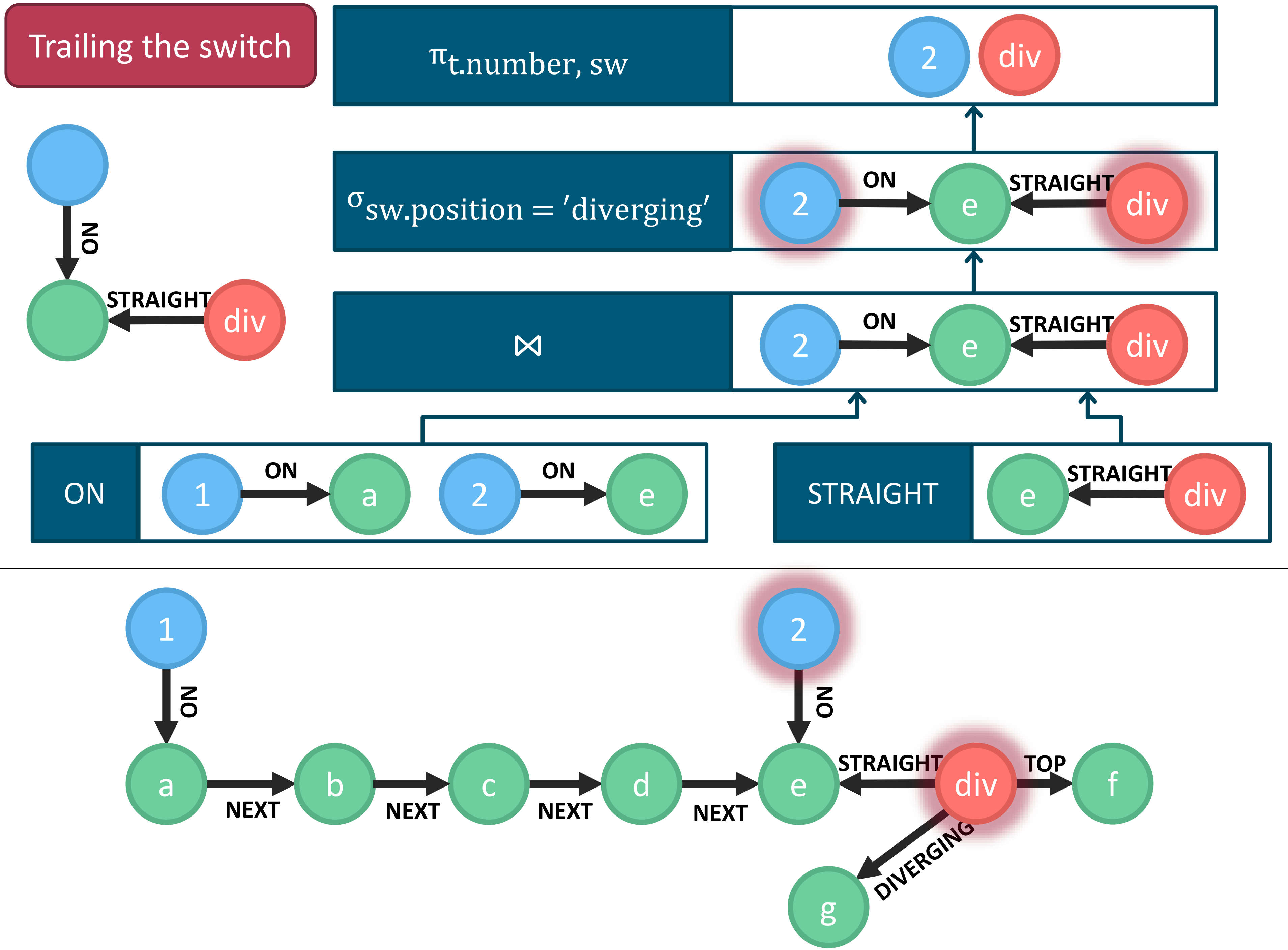




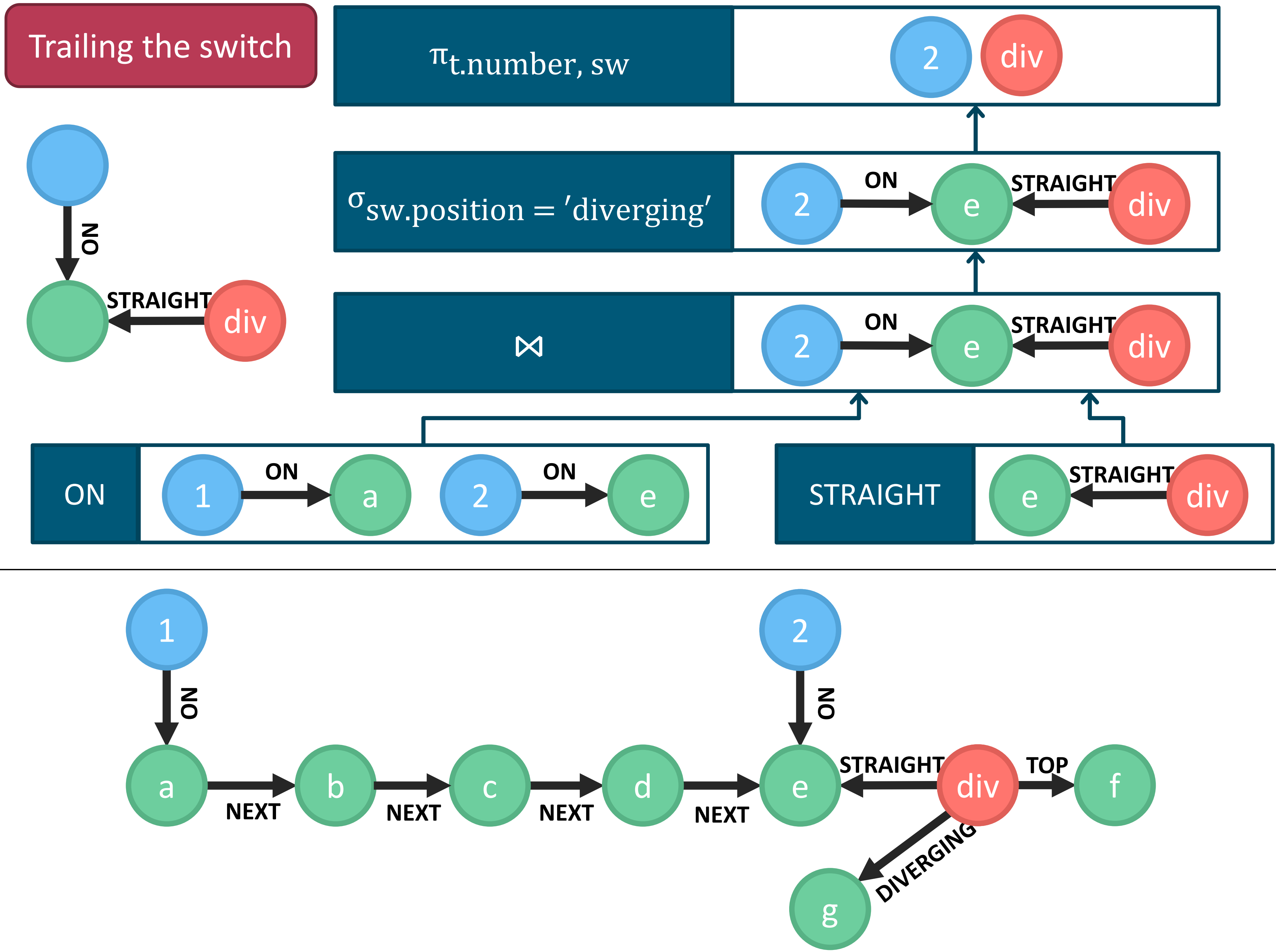
Trailing the switch
πt.number, sw
σsw.position = 'diverging'
⋈
ON
STRAIGHT
div
2
e
ON
1
a
STRAIGHT
e
div
1
2
a
b
c
d
e
f
g
NEXT
NEXT
NEXT
NEXT
STRAIGHT
TOP
DIVERGING


Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position\ =\ 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
2
e
1
a
b
c
d
f
g
NEXT
TOP
DIVERGING

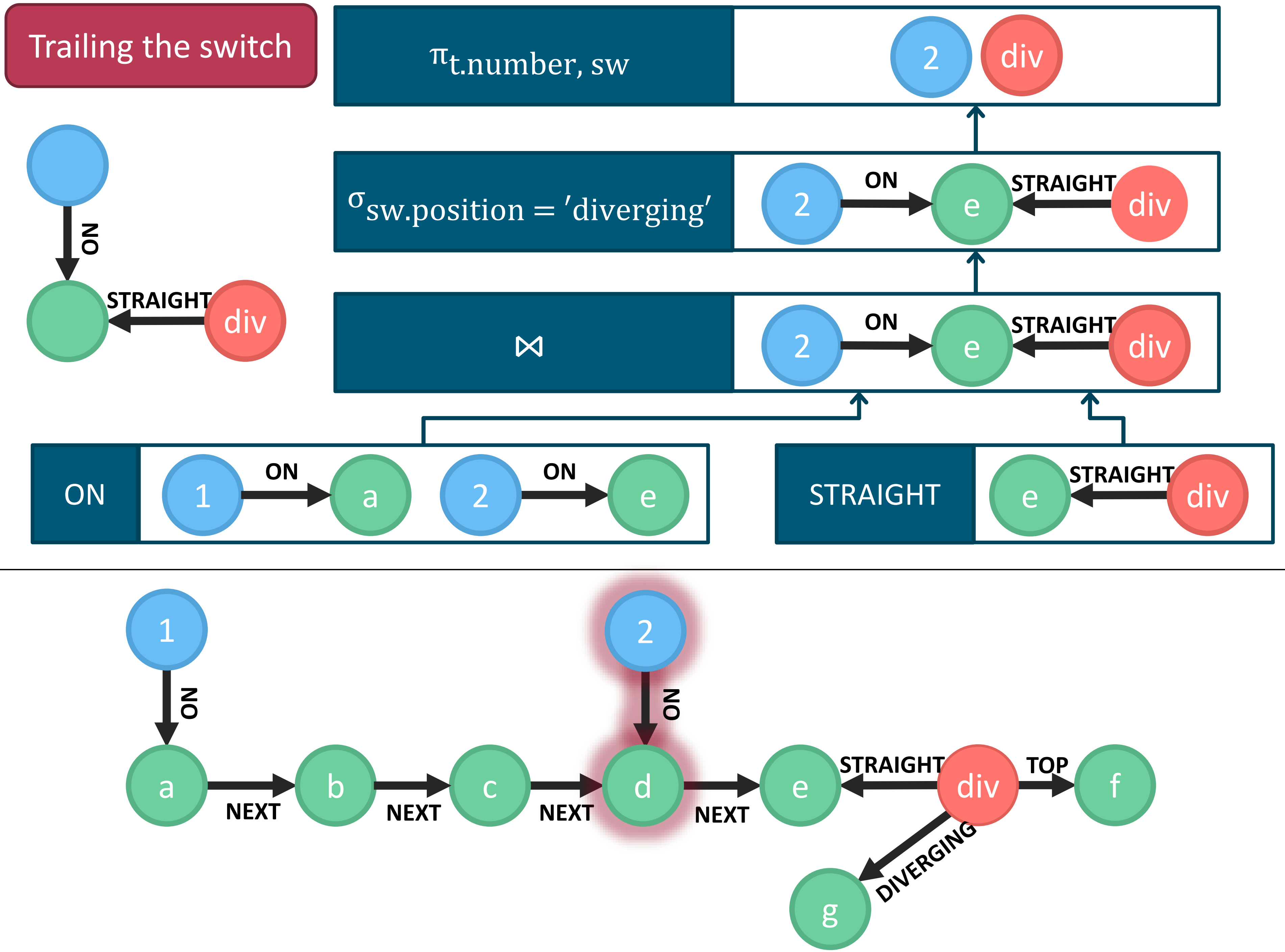

Trailing the switch
π t.number, sw
σ sw.position = 'diverging'
⋈
ON
STRAIGHT
div
2
e
1
a
b
c
d
f
g
NEXT
TOP
DIVERGING



Trailing the switch
$\pi_{t.number,\ sw}$
2
div
$\sigma_{sw.position\ =\ 'diverging'}$
2
ON
e
STRAIGHT
div
$\bowtie$
2
ON
e
STRAIGHT
div
ON
1
ON
a
2
ON
e
STRAIGHT
e
STRAIGHT
div
ON
STRAIGHT
div
1
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
e
2
ON
STRAIGHT
div
TOP
f
DIVERGING
g

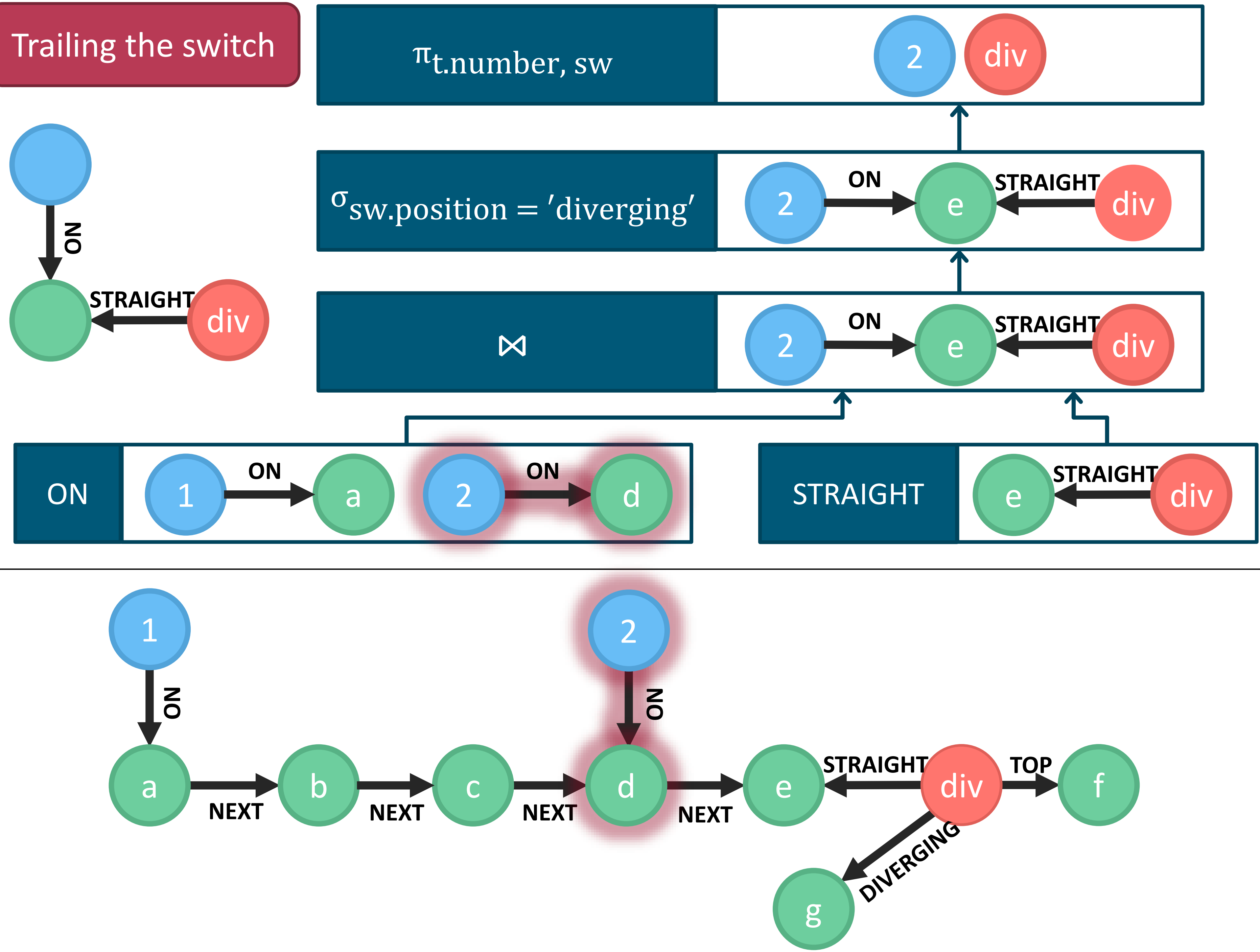


Trailing the switch
$\pi_{t.number,\ sw}$
2
div
$\sigma_{sw.position\ =\ 'diverging'}$
ON
STRAIGHT
2
e
div
$\bowtie$
ON
STRAIGHT
2
e
div
ON
div
STRAIGHT
ON
1
a
2
e
ON
ON
STRAIGHT
e
div
STRAIGHT
1
2
ON
ON
a
b
c
d
e
div
f
g
NEXT
NEXT
NEXT
NEXT
STRAIGHT
TOP
DIVERGING

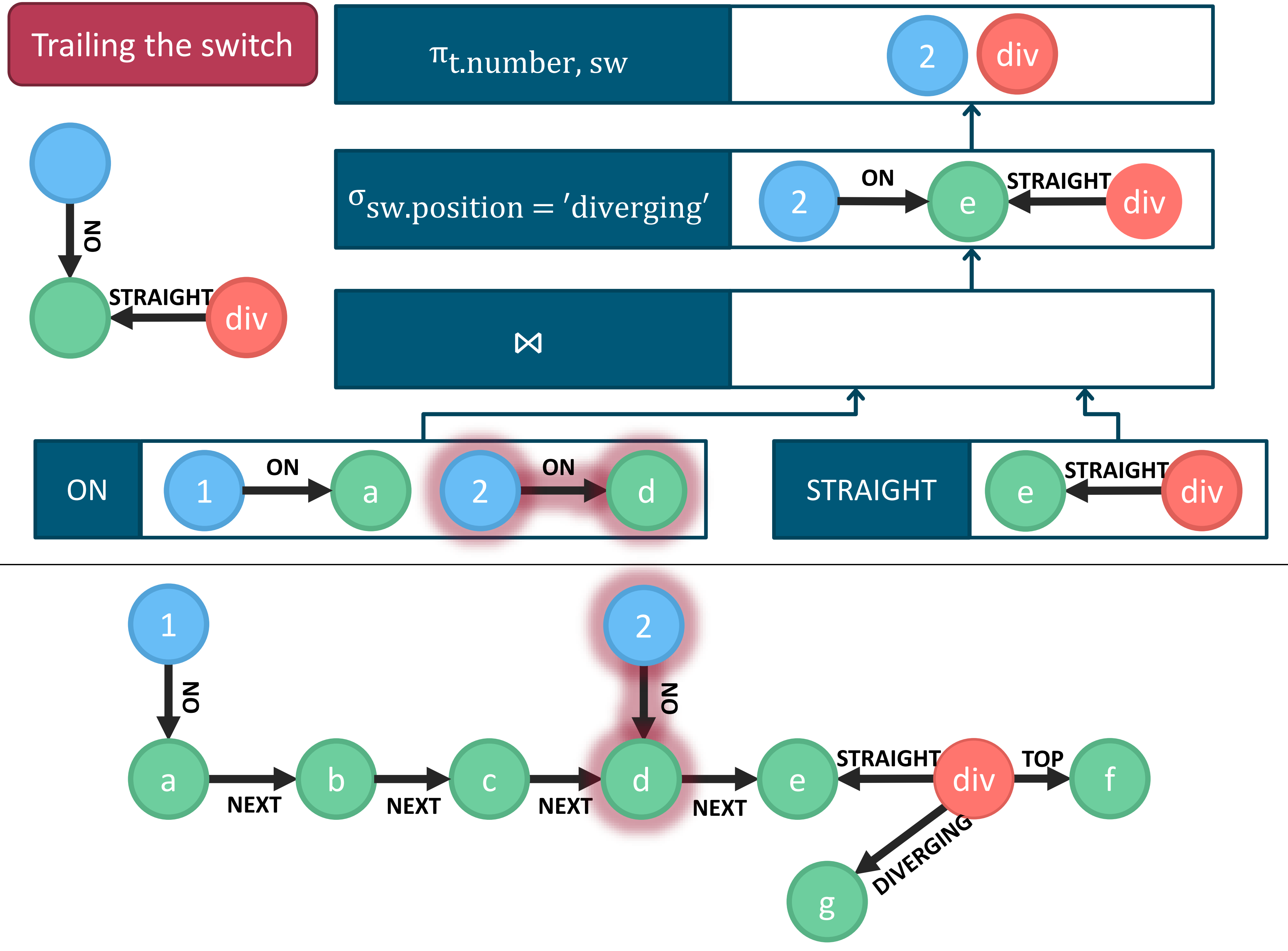


Trailing the switch
$\pi_{t.number,\ sw}$
$\sigma_{sw.position = 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
2
e
1
a
b
c
d
f
g
NEXT
TOP
DIVERGING

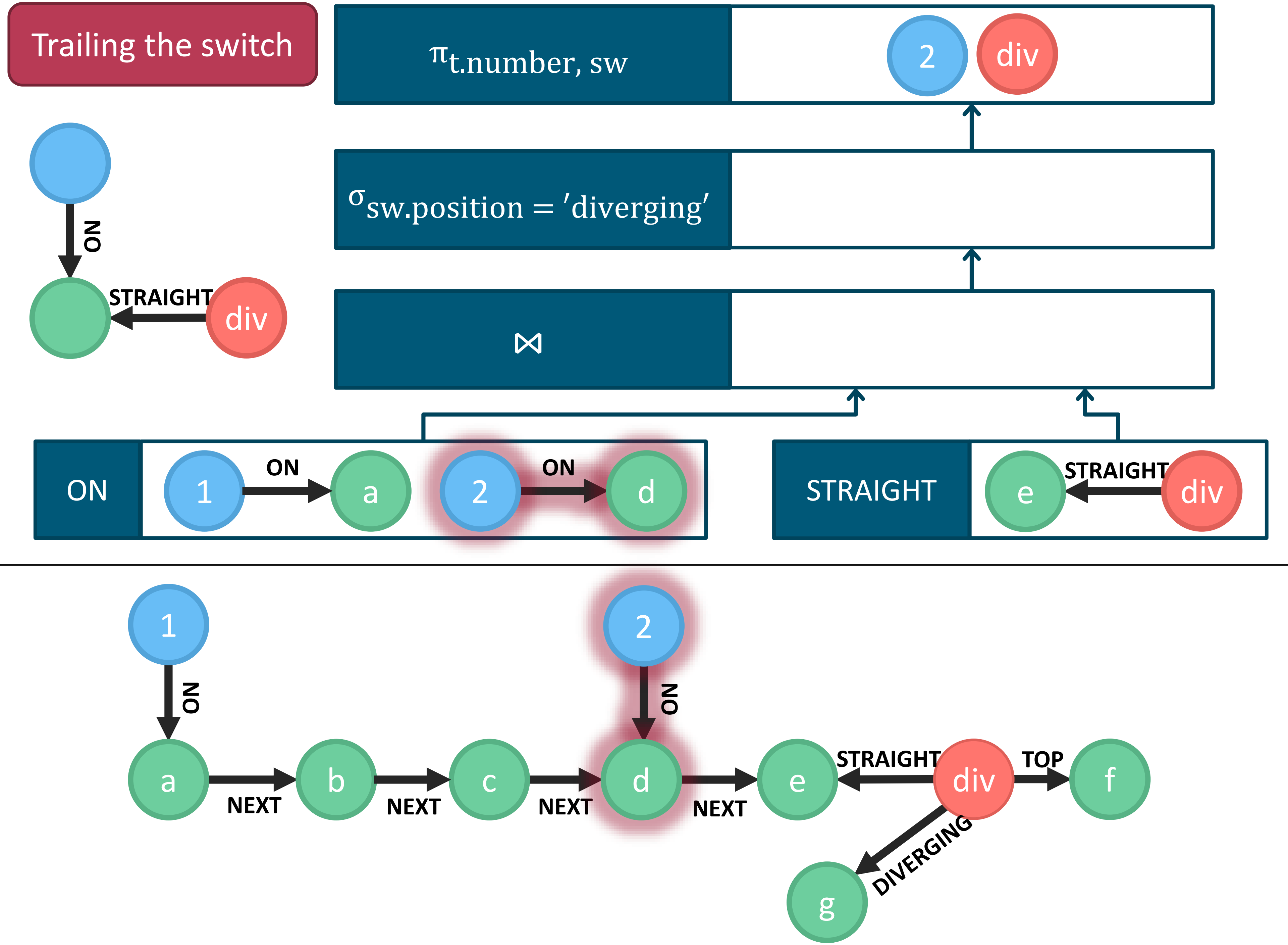
Trailing the switch

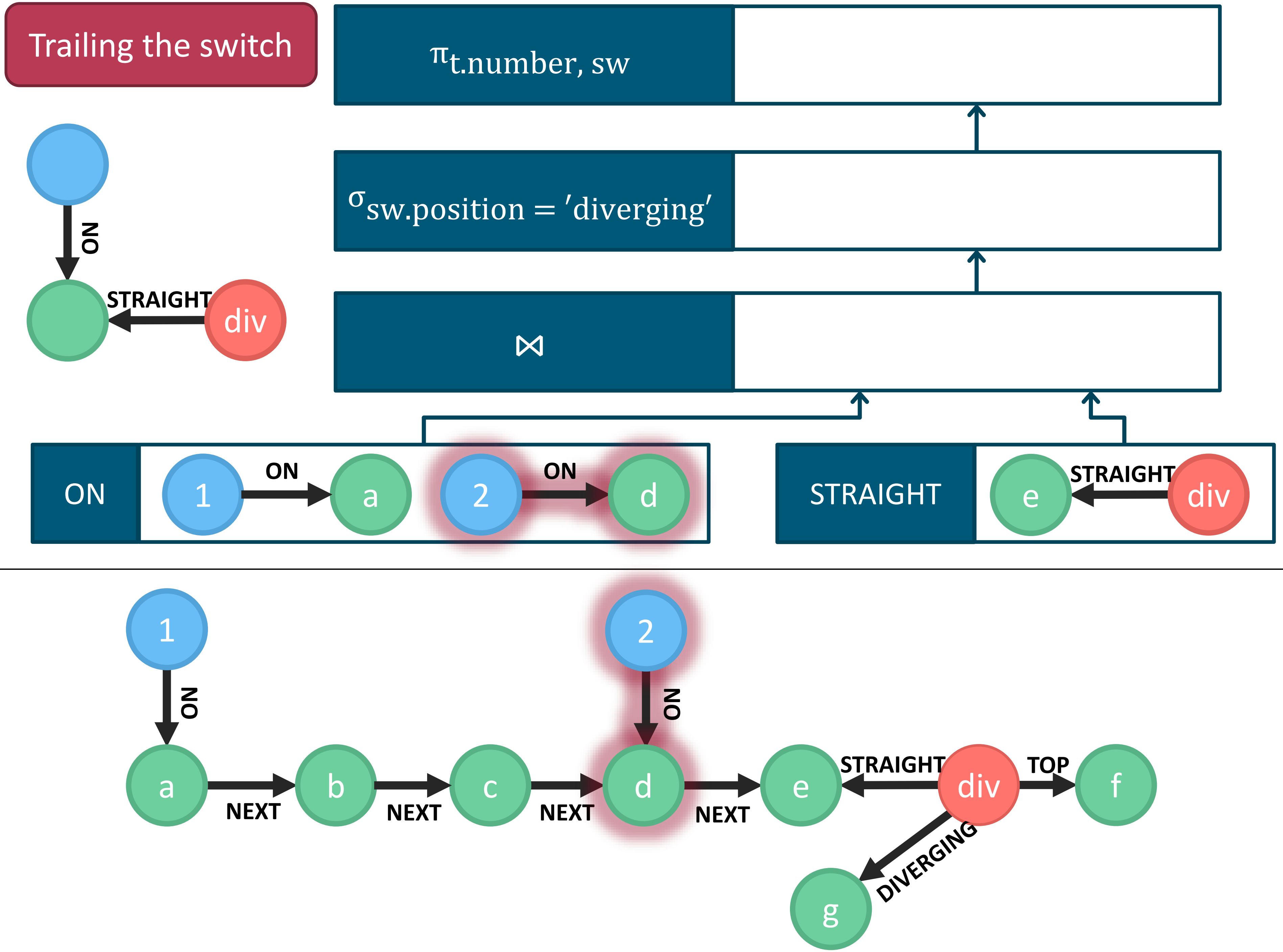


Trailing the switch



Trailing the switch





Trailing the switch
$\pi_{t.number,\, sw}$
$\sigma_{sw.position = 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
ON
1
ON
a
2
ON
d
STRAIGHT
e
STRAIGHT
div
1
ON
a
NEXT
b
NEXT
c
NEXT
2
ON
d
NEXT
e
STRAIGHT
div
TOP
f
DIVERGING
g

Batch vs. incremental queries

- **Batch queries**
  (pull / request-driven):
  1. Client selects a query
  2. Results are calculated
- Query results
  obtained on demand

Batch vs. incremental queries

- **Batch queries** (pull / request-driven):
  1. Client selects a query
  2. Results are calculated
- Query results obtained on demand

- **Incremental queries** (push / event-driven):
  1. Client registers queries
  2. Graph is changed
  3. Results are maintained
  4. Goto 2
- Query results are always available

Incremental query engines

CLIPS	C structures	NASA
Drools	POJO	Red Hat
VIATRA	EMF	BME / IncQuery Labs.

Incremental query engines

- CLIPS C structures NASA
- Drools POJO Red Hat
- VIATRA EMF BME / IncQuery Labs.

- INSTANS RDF Aalto University
- i3QL POJO TU Darmstadt
- IncQuery-D RDF BME

Incremental query engines

CLIPS	C structures	NASA
Drools	POJO	Red Hat
VIATRA	EMF	BME / IncQuery Labs.
INSTANS	RDF	Aalto University
i3QL	POJO	TU Darmstadt
IncQuery-D	RDF	BME

- **No implementations for property graphs yet**

Incremental Graph Queries with openCypher

Incremental Graph Queries **with openCypher**

openCypher

„The openCypher project aims to deliver a full and open specification of the industry’s most widely adopted graph database query language: Cypher.”

- Grammar specification
- Technology Compatibility Kit (TCK)
- Reference implementation

openCypher constructs

- Standard constructs
  - pattern matching
  - filtering
  - lists, maps
  - data manipulation
  - variable length paths

openCypher constructs

- Standard constructs
  - pattern matching
  - filtering
  - lists, maps
  - data manipulation
  - variable length paths
- Legacy constructs
  - indexing, constraints
  - regular expressions
  - some list functions, including `reduce`
  - most predicate functions
  - shortest path functions
  - CASE expressions
  - `id()`

openCypher constructs

- Standard constructs
  - pattern matching
  - filtering
  - lists, maps
  - data manipulation
  - variable length paths
- Legacy constructs
  - indexing, constraints
  - regular expressions
  - some list functions, including `reduce`
  - most predicate functions
  - shortest path functions
  - `CASE` expressions
  - `id()`

Difficult to handle incrementally

Mapping openCypher to relational algebra

Combining and filtering pattern matches	
`MATCH «p»`	$\not\equiv_{\text{edges of p}}(\mathsf{p})$
`MATCH «p1», «p2»`	$\not\equiv_{\text{edges of p1 and p2}}(\mathsf{p}_1 \bowtie \mathsf{p}_2)$
`MATCH «p1»` `MATCH «p2»`	$\not\equiv_{\text{edges of p1}}(\mathsf{p}_1) \bowtie \not\equiv_{\text{edges of p2}}(\mathsf{p}_2)$
`MATCH «p1»` `OPTIONAL MATCH «p2»`	$\not\equiv_{\text{edges of p1}}(\mathsf{p}_1) \mathbin{⟕} \not\equiv_{\text{edges of p2}}(\mathsf{p}_2)$
`MATCH «p»` `WHERE «condition»`	$\sigma_{\text{condition}}(r)$, where condition may specify patterns and arithmetic constraints on existing variables
Result and sub-result operations. Rules for `RETURN` also apply to `WITH`.	
`RETURN «variables»`	$\pi_{\text{variables}}(r)$
`RETURN «v1» AS «alias1» ...`	$\pi_{\text{v1}\rightarrow\text{alias1},\ldots}(r)$
`RETURN DISTINCT «variables»`	$\delta\left(\pi_{\text{variables}}(r)\right)$
`RETURN «variables», «aggregates»`	$\gamma_{\text{variables,aggregates}}(r)$
List operations	
`ORDER BY «v1» [ASC|DESC] ...`	$\tau_{\Uparrow/\Downarrow\text{v1},\ldots}(r)$
`LIMIT «l»`	$\lambda_l(r)$

Mapping openCypher to relational algebra

Combining and filtering pattern matches	
`MATCH «p»`	$\dot{\neq}_{\text{edges of p}}(\mathsf{p})$
`MATCH «p1», «p2»`	$\dot{\neq}_{\text{edges of p1 and p2}}(\mathsf{p}_1 \bowtie \mathsf{p}_2)$
`MATCH «p1»` `MATCH «p2»`	$\dot{\neq}_{\text{edges of p1}}(\mathsf{p}_1) \bowtie \dot{\neq}_{\text{edges of p2}}(\mathsf{p}_2)$
`MATCH «p1»` `OPTIONAL MATCH «p2»`	$\dot{\neq}_{\text{edges of p1}}(\mathsf{p}_1)$ ⟕ $\dot{\neq}_{\text{edges of p2}}(\mathsf{p}_2)$
`MATCH «p»` `WHERE «condition»`	$\sigma_{\text{condition}}(r)$, where condition may specify patterns and arithmetic constraints on existing variables
Result and sub-result operations. Rules for `RETURN` also apply to `WITH`.	
`RETURN «variables»`	$\pi_{\text{variables}}(r)$
`RETURN «v1» AS «alias1» ...`	$\pi_{\text{v1}\rightarrow\text{alias1},\ldots}(r)$
`RETURN DISTINCT «variables»`	$\delta\left(\pi_{\text{variables}}(r)\right)$
`RETURN «variables», «aggregates»`	$\gamma_{\text{variables,aggregates}}(r)$
List operations	
`ORDER BY «v1» [ASC|DESC] ...`	$\tau_{\Uparrow/\Downarrow\text{v1}}$
`LIMIT «l»`	$\lambda_l(r)$

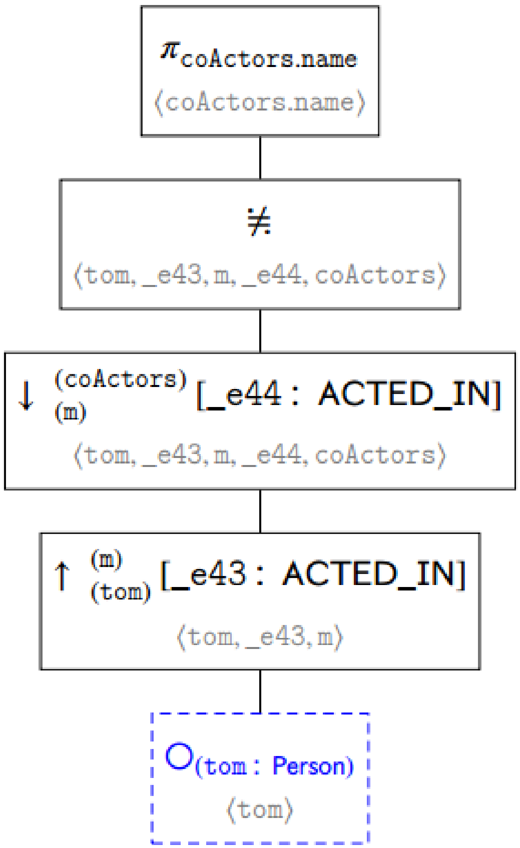
Technical report

Query specification (query-18)

```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors)
RETURN coActors.name
```

Query specification (query-18)

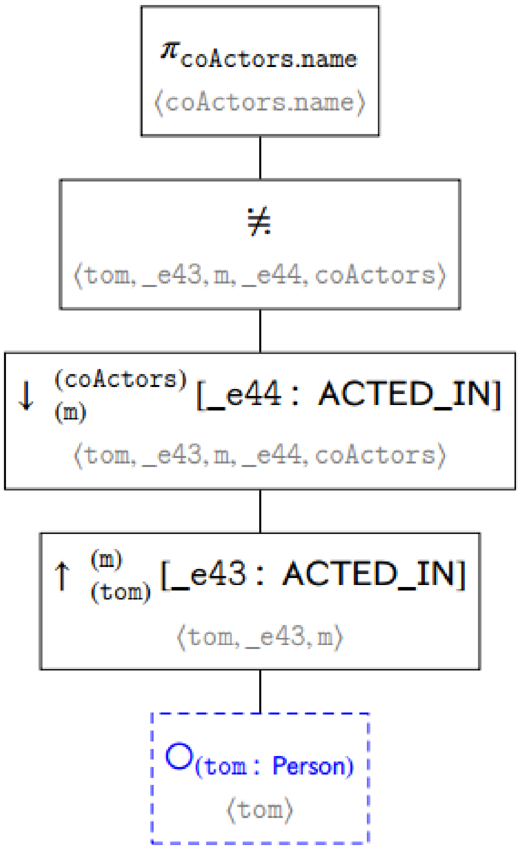
```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors)
RETURN coActors.name
```



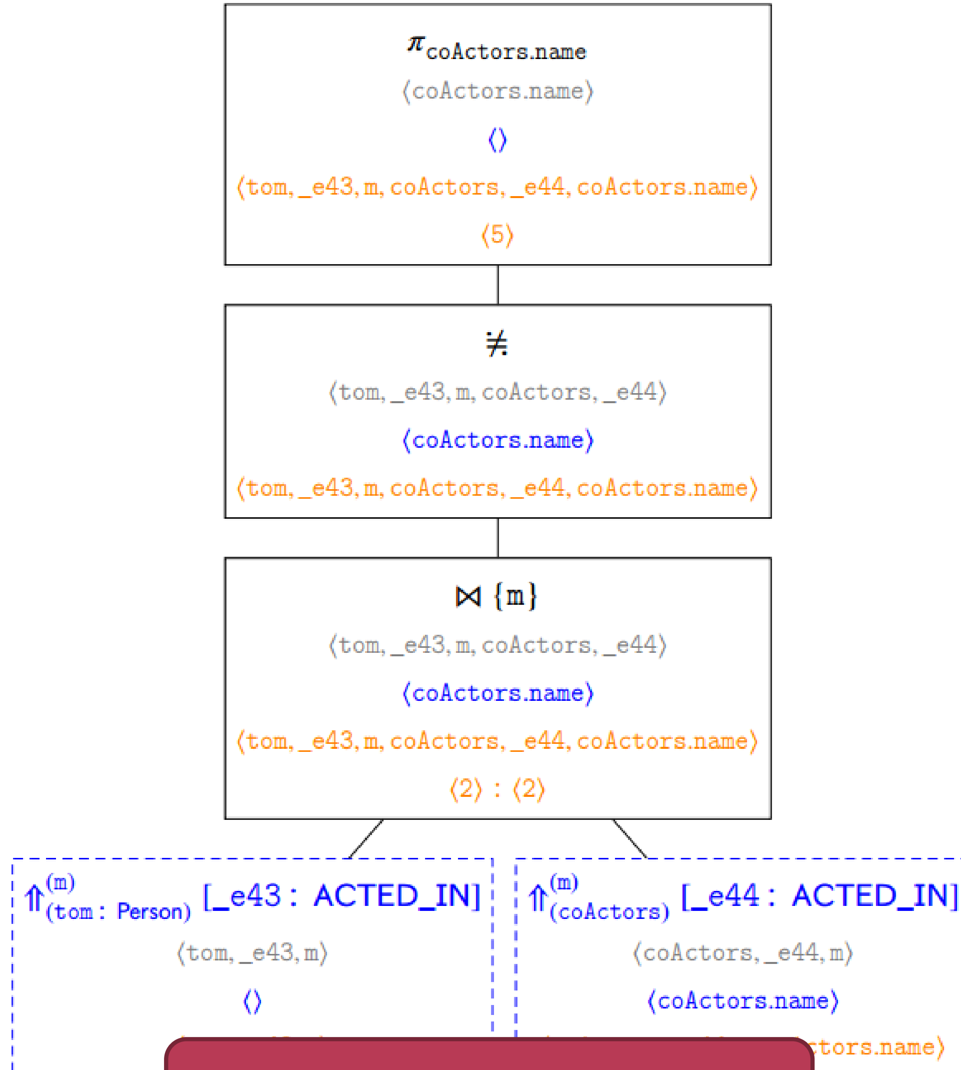
Batch search plan

Query specification (query-18)

```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors)
RETURN coActors.name
```



Batch search plan



Incremental search plan

Incremental openCypher challenges

- Lists
  - `['a', 1, 2, true]`
  - `['a', [1, [2]], true]`
  - `UNWIND`
  - `collect()`
- Bag semantics, `ORDER BY`, `SKIP` and `LIMIT`
- Legacy constructs
  - `shortestPath()`, `allShortestPaths()`
  - `reduce()`

Incremental openCypher challenges

- Lists
  - `['a', 1, 2, true]`
  - `['a', [1, [2]], true]`
  - `UNWIND`
  - `collect()`
- Bag semantics, `ORDER BY`, `SKIP` and `LIMIT`
- Legacy constructs
  - `shortestPath()`, `allShortestPaths()`
  - `reduce()`

These are not handled by traditional Rete implementations

ingraph

- An incremental, in-memory graph query engine

ingraph

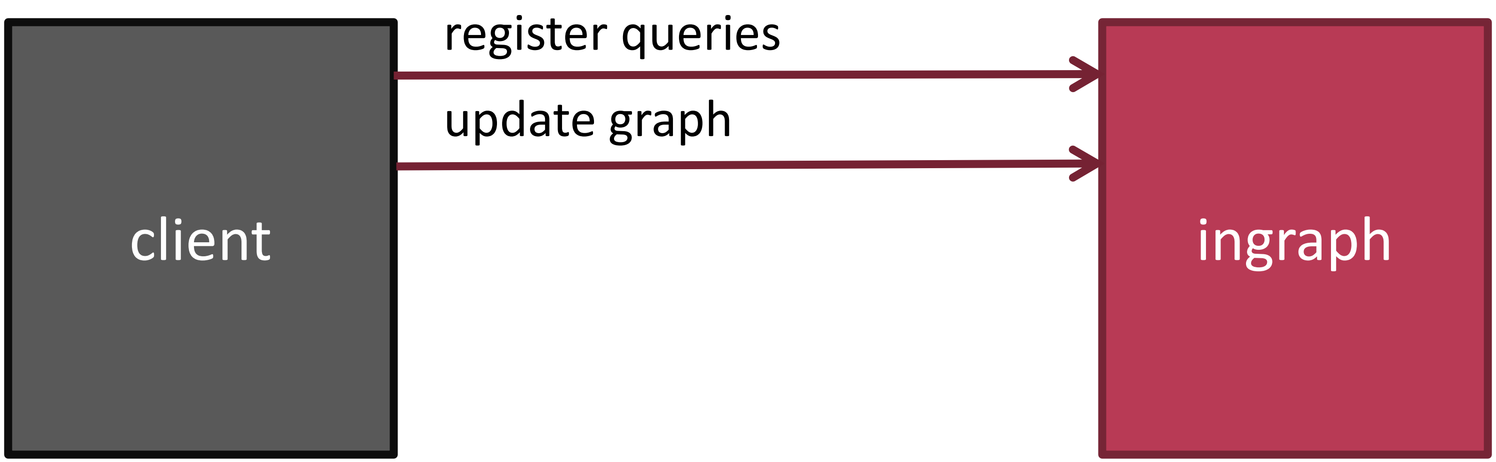
- An incremental, in-memory graph query engine

ingraph

- An incremental, in-memory graph query engine

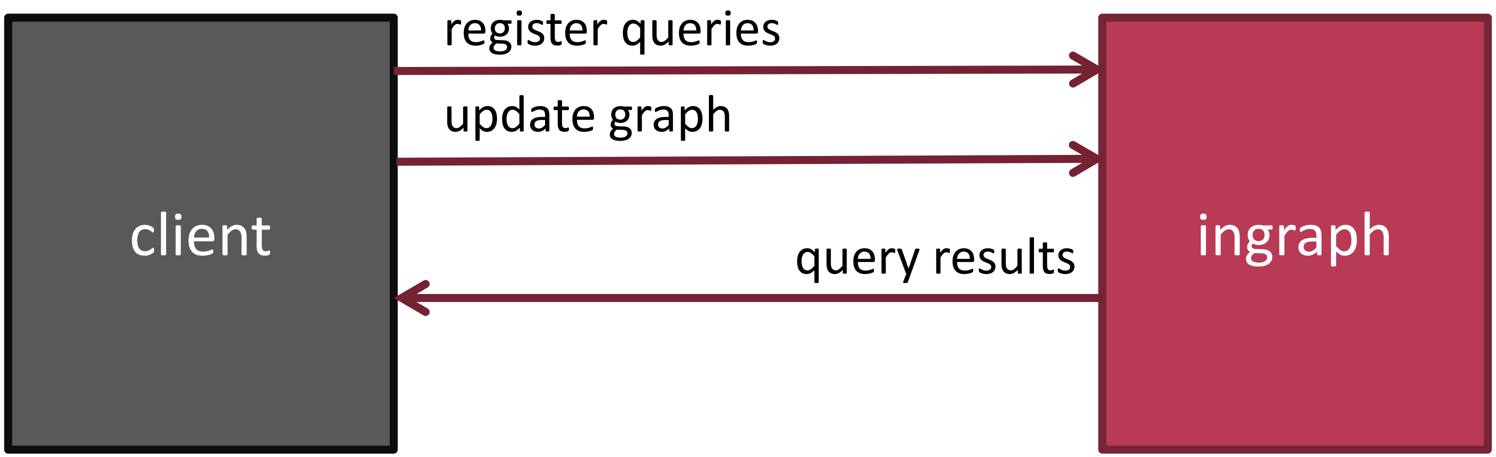
ingraph

- An incremental, in-memory graph query engine



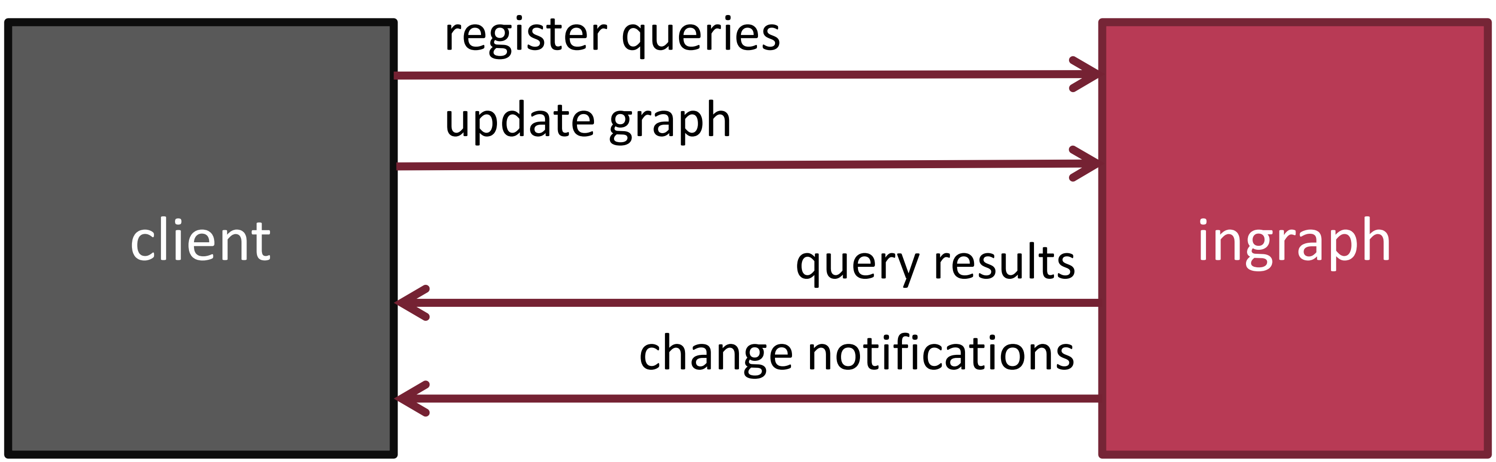
ingraph

- An incremental, in-memory graph query engine



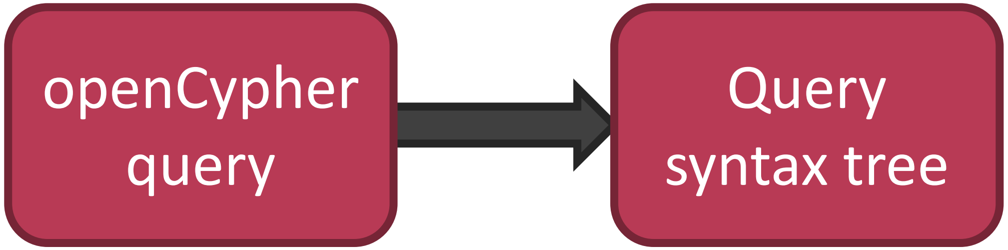
ingraph

- An incremental, in-memory graph query engine

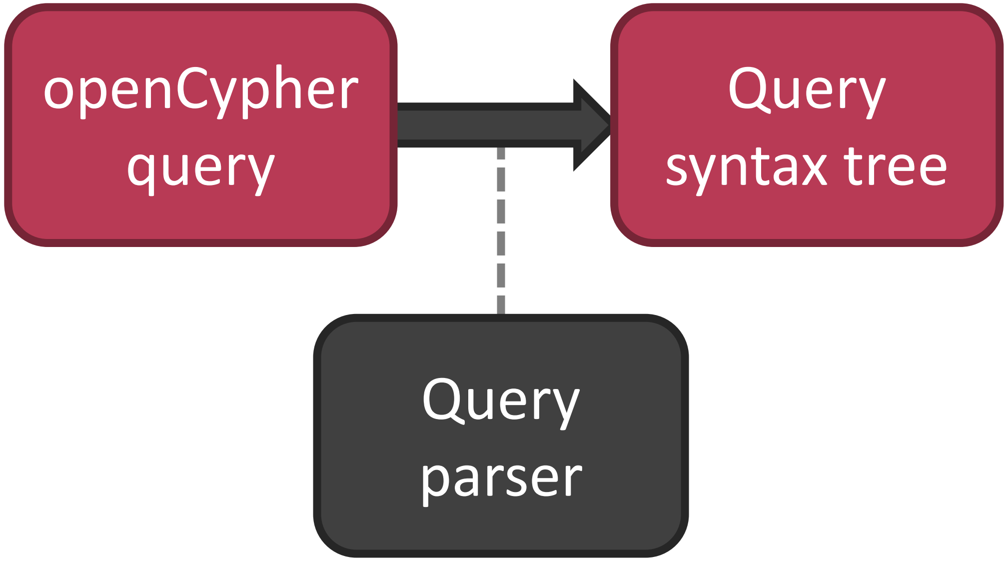


openCypher query

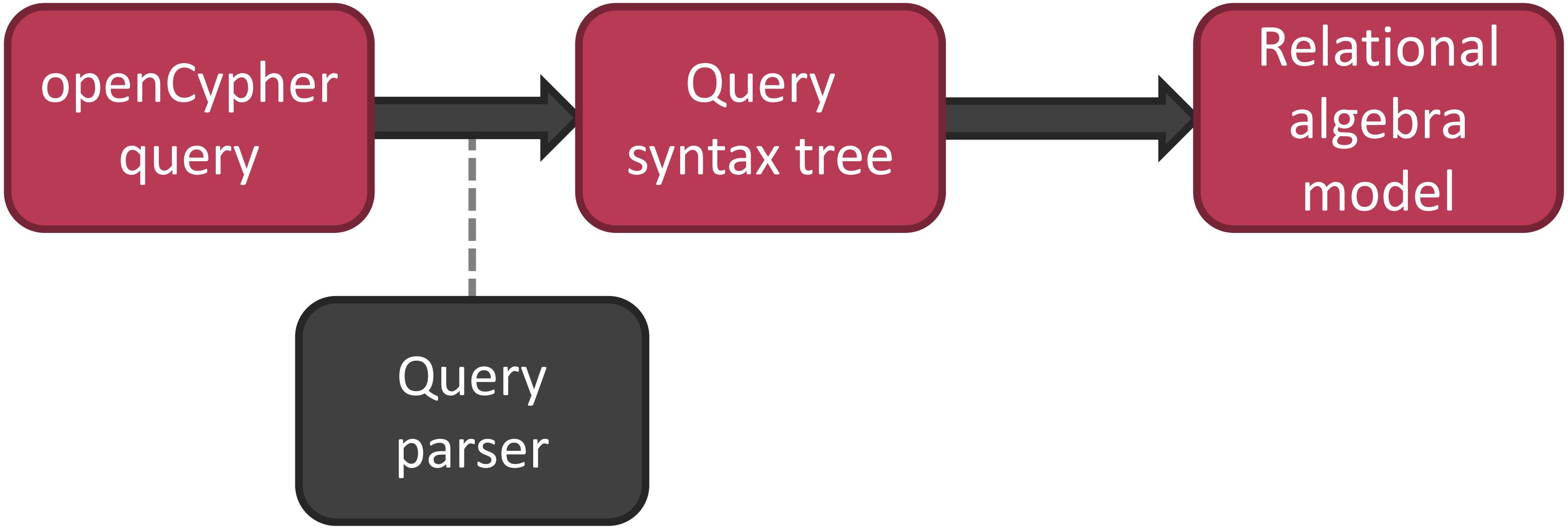
```
MATCH (t:Train)-[:ON]->(seg:Segment)
      <-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw
```



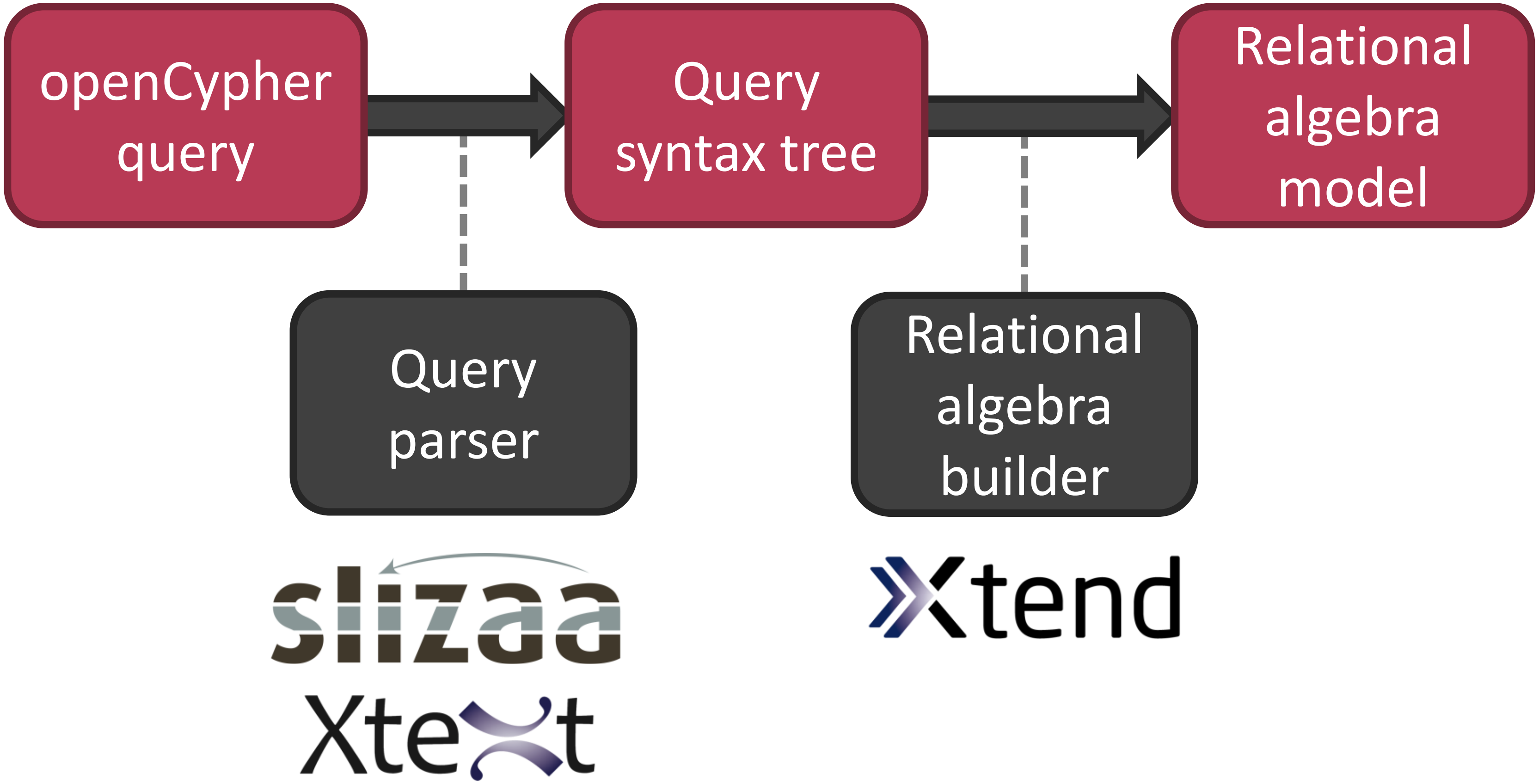

```
MATCH (t:Train)-[:ON]->(seg:Segment)
      <-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw
```



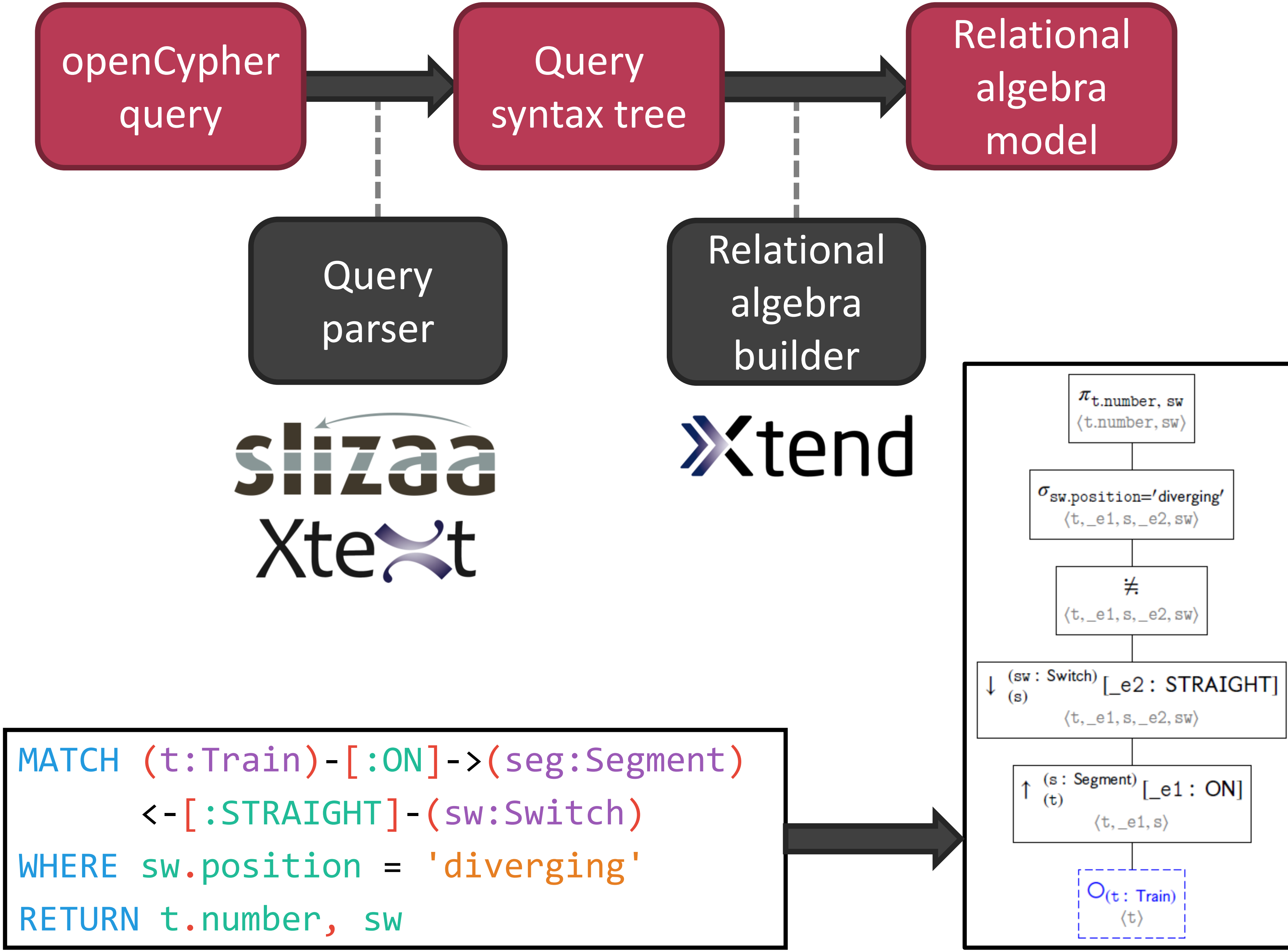

```
MATCH (t:Train)-[:ON]->(seg:Segment)
      <-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw
```



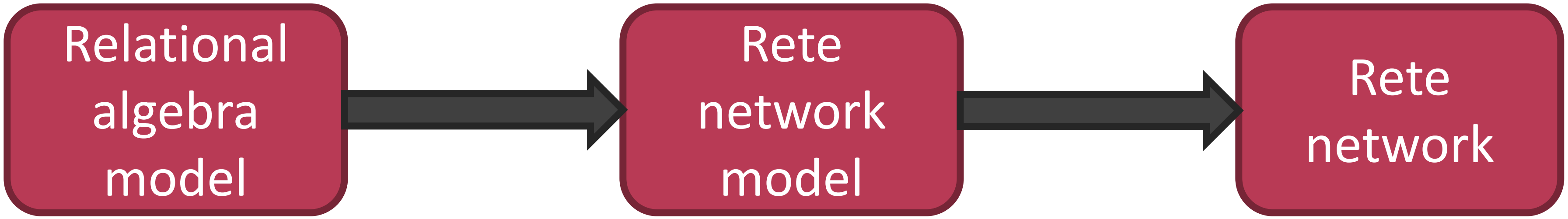

```
MATCH (t:Train)-[:ON]->(seg:Segment)
      <-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw
```

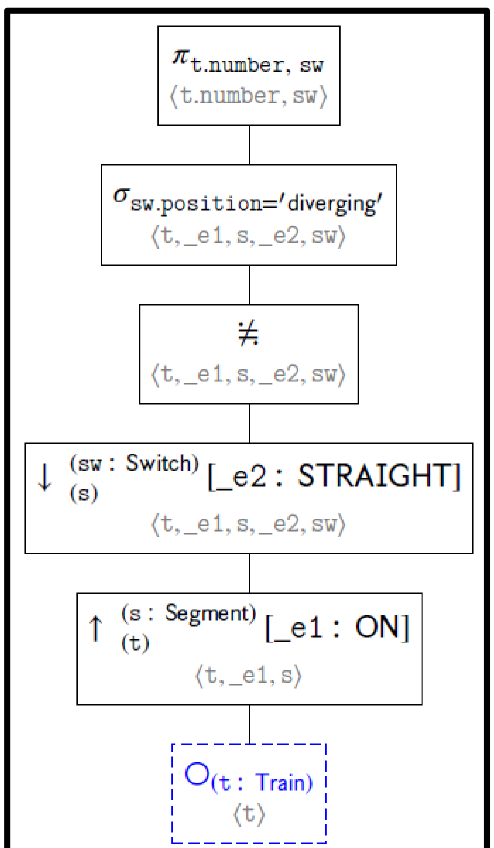


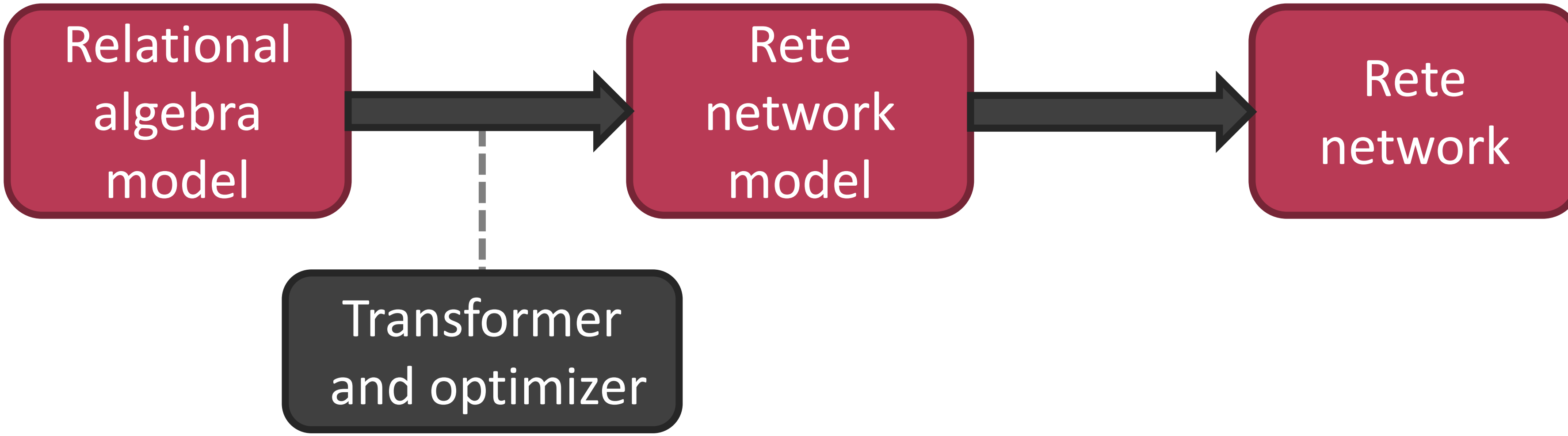

```
MATCH (t:Train)-[:ON]->(seg:Segment)
      <-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw
```



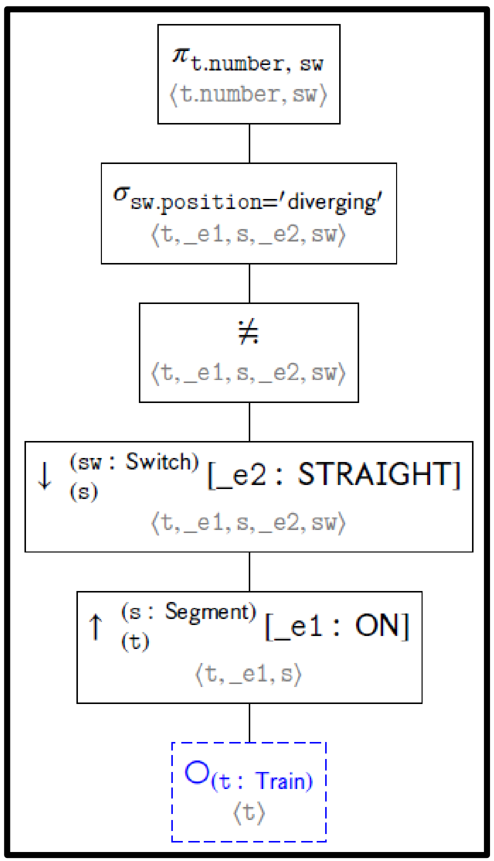
openCypher query
Query syntax tree
Relational algebra model
Query parser
Relational algebra builder
slizaa
Xtext
Xtend
$\pi_{\text{t.number, sw}}$
⟨t.number, sw⟩
$\sigma_{\text{sw.position='diverging'}}$
⟨t, _e1, s, _e2, sw⟩
$\not\equiv$
⟨t, _e1, s, _e2, sw⟩
$\downarrow^{(\text{sw : Switch})}_{(\text{s})}$ [_e2 : STRAIGHT]
⟨t, _e1, s, _e2, sw⟩
$\uparrow^{(\text{s : Segment})}_{(\text{t})}$ [_e1 : ON]
⟨t, _e1, s⟩
$\bigcirc_{(\text{t : Train})}$
⟨t⟩
MATCH (t:Train)-[:ON]->(seg:Segment)
<-[:STRAIGHT]-(sw:Switch)
WHERE sw.position = 'diverging'
RETURN t.number, sw


Relational algebra model
Rete network model
Rete network


$\pi_{\mathtt{t.number,\ sw}}$
$\langle \mathtt{t.number, sw} \rangle$
$\sigma_{\mathtt{sw.position}='\mathsf{diverging}'}$
$\langle \mathtt{t, \_e1, s, \_e2, sw} \rangle$
$\not\asymp$
$\langle \mathtt{t, \_e1, s, \_e2, sw} \rangle$
$\downarrow^{(\mathtt{sw}:\ \mathsf{Switch})}_{(\mathtt{s})} [\mathsf{\_e2} : \mathsf{STRAIGHT}]$
$\langle \mathtt{t, \_e1, s, \_e2, sw} \rangle$
$\uparrow^{(\mathtt{s}:\ \mathsf{Segment})}_{(\mathtt{t})} [\mathsf{\_e1} : \mathsf{ON}]$
$\langle \mathtt{t, \_e1, s} \rangle$
$\bigcirc_{(\mathtt{t}:\ \mathsf{Train})}$
$\langle \mathtt{t} \rangle$

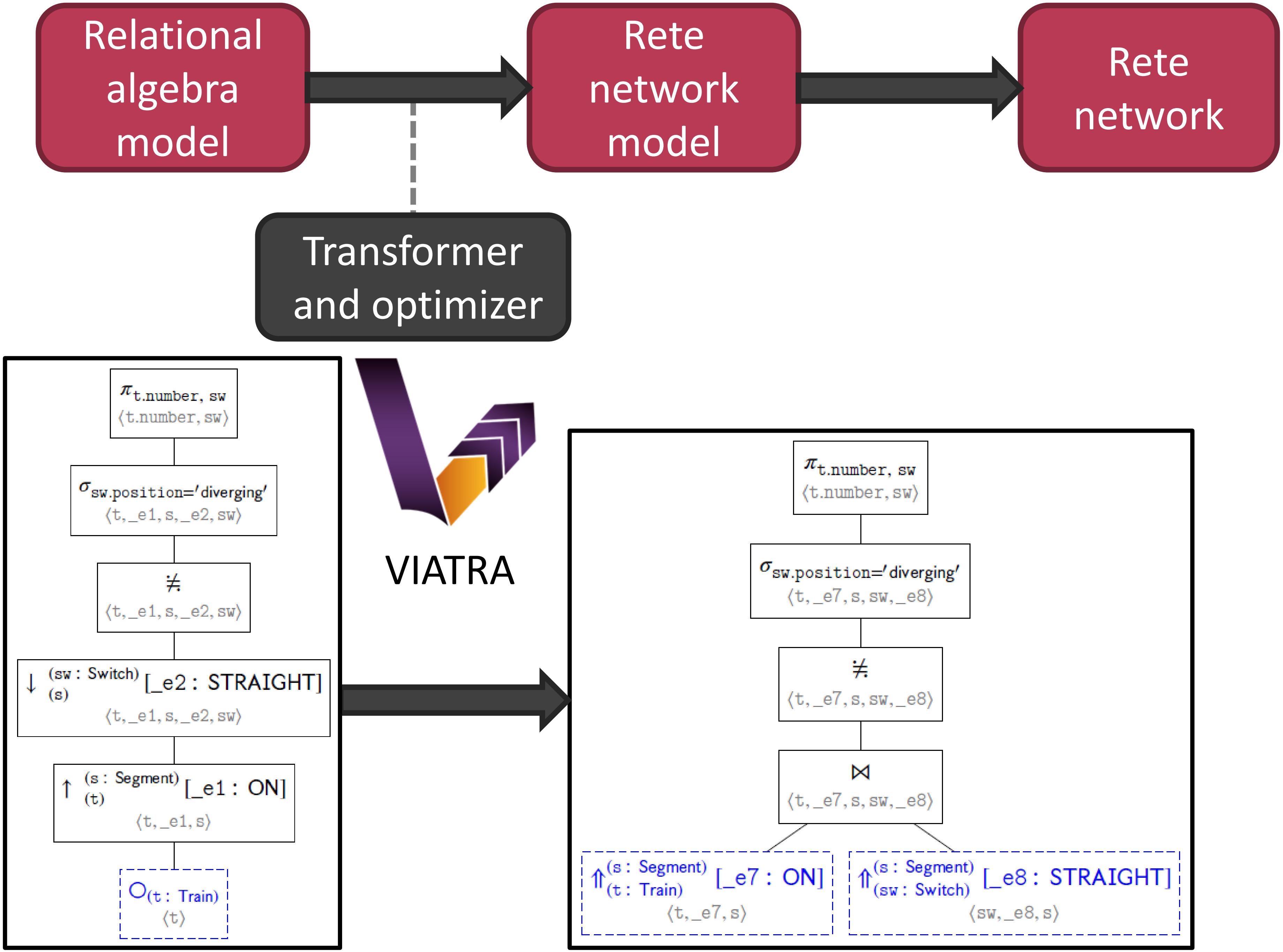


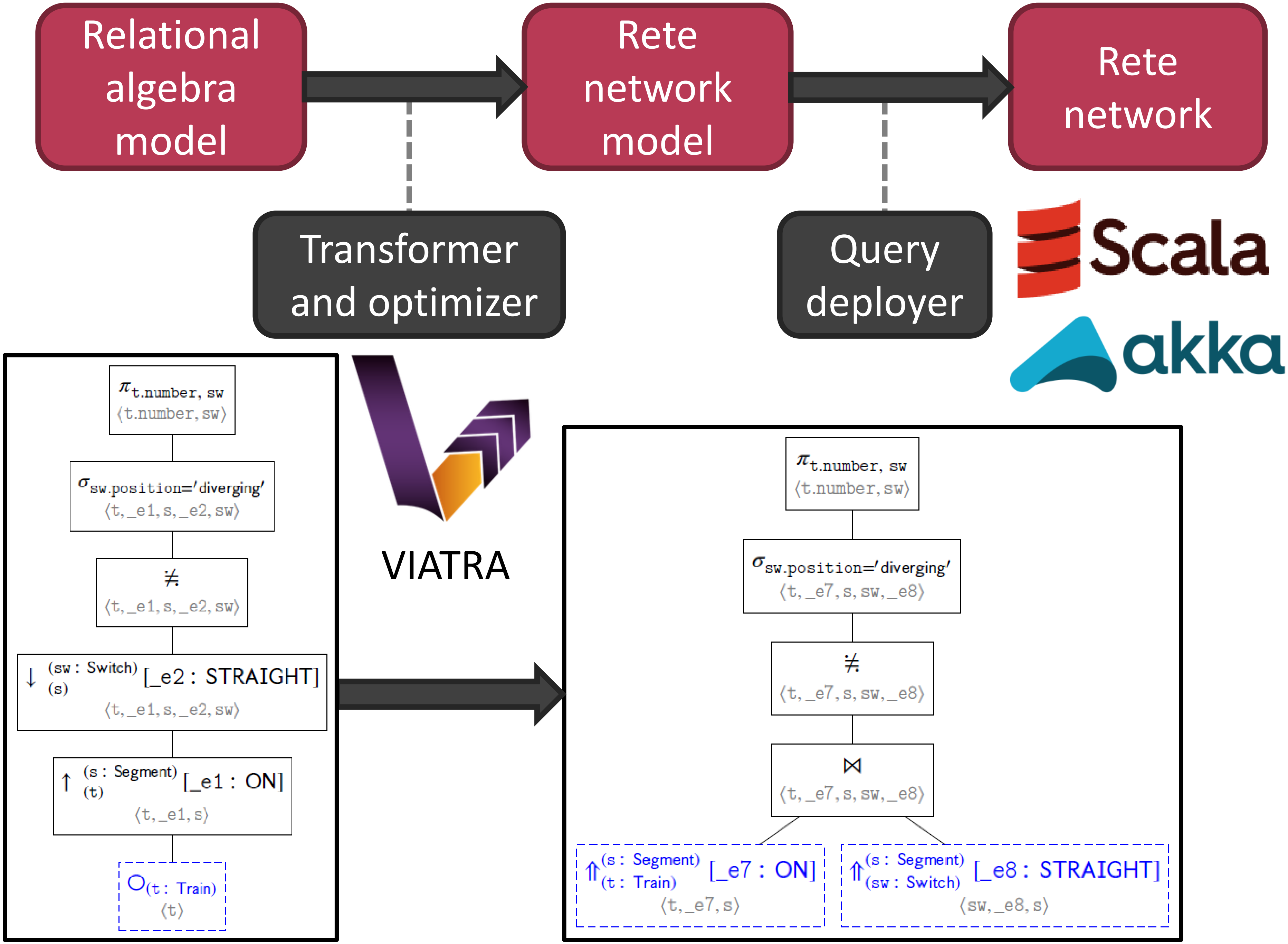
Relational algebra model
Rete network model
Rete network
Transformer and optimizer



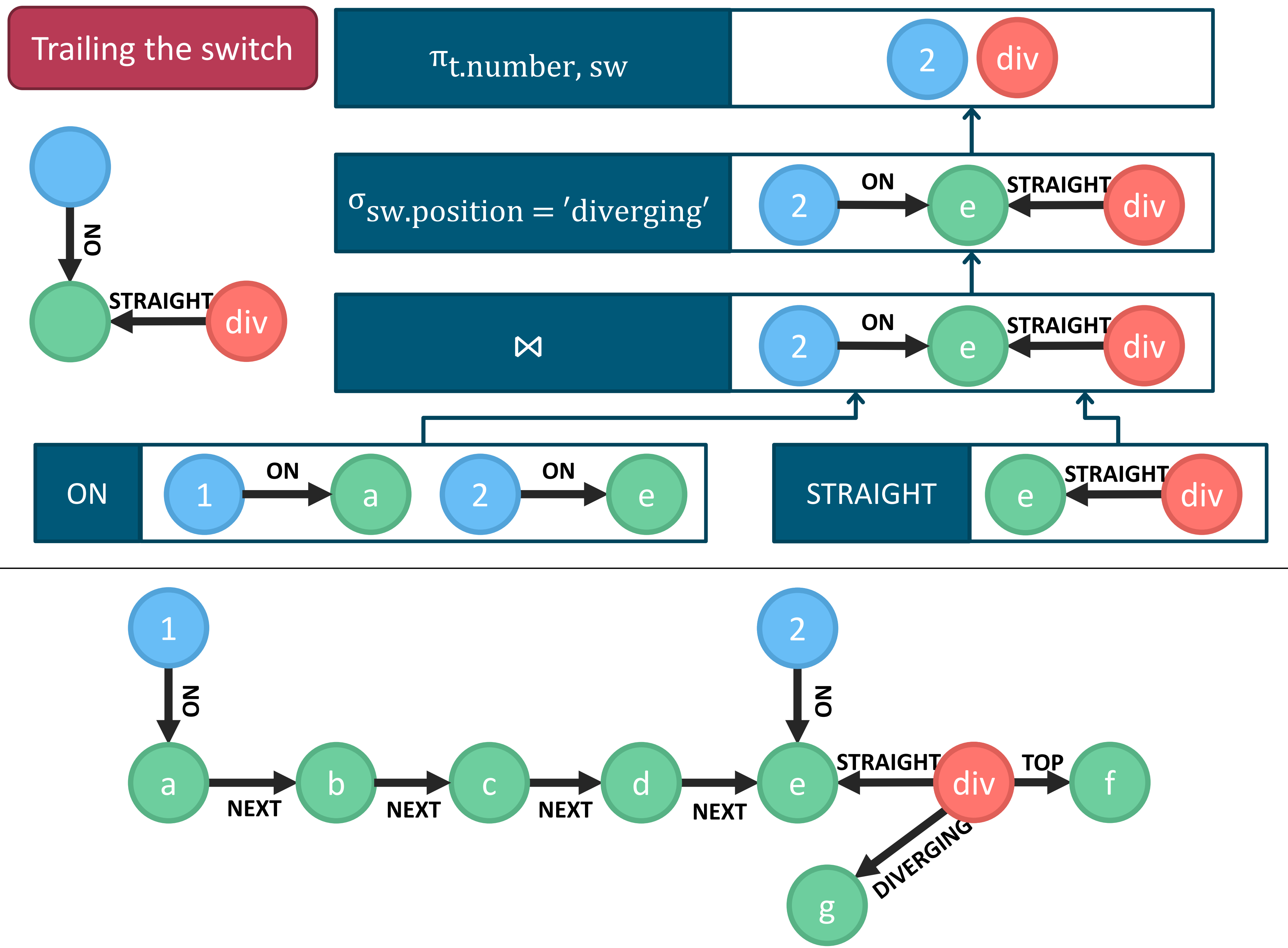
$\pi_{\mathtt{t.number,\ sw}}$
$\langle \mathtt{t.number, sw} \rangle$
$\sigma_{\mathtt{sw.position}='\mathsf{diverging}'}$
$\langle \mathtt{t, \_e1, s, \_e2, sw} \rangle$
$\not\equiv$
$\langle \mathtt{t, \_e1, s, \_e2, sw} \rangle$
$\downarrow^{(\mathtt{sw}:\ \mathsf{Switch})}_{(\mathtt{s})} [\_\mathsf{e2}:\ \mathsf{STRAIGHT}]$
$\langle \mathtt{t, \_e1, s, \_e2, sw} \rangle$
$\uparrow^{(\mathtt{s}:\ \mathsf{Segment})}_{(\mathtt{t})} [\_\mathsf{e1}:\ \mathsf{ON}]$
$\langle \mathtt{t, \_e1, s} \rangle$
$\bigcirc_{(\mathtt{t}:\ \mathsf{Train})}$
$\langle \mathtt{t} \rangle$

VIATRA

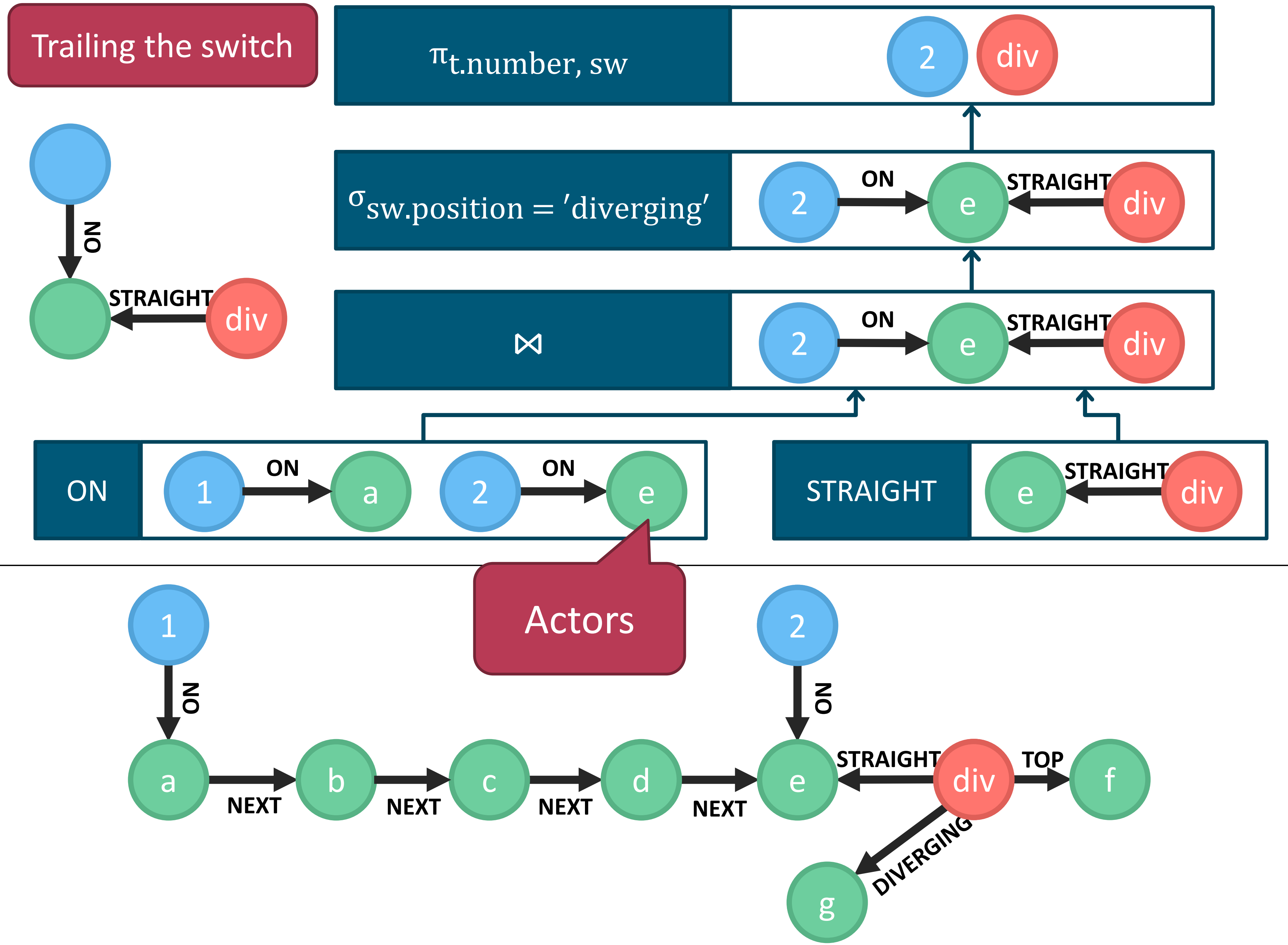

Relational algebra model
Rete network model
Rete network
Transformer and optimizer
VIATRA
π t.number, sw
⟨t.number, sw⟩
σ sw.position='diverging'
⟨t, _e1, s, _e2, sw⟩
≉
⟨t, _e1, s, _e2, sw⟩
↓ (sw : Switch) (s) [_e2 : STRAIGHT]
⟨t, _e1, s, _e2, sw⟩
↑ (s : Segment) (t) [_e1 : ON]
⟨t, _e1, s⟩
○ (t : Train)
⟨t⟩
π t.number, sw
⟨t.number, sw⟩
σ sw.position='diverging'
⟨t, _e7, s, sw, _e8⟩
≉
⟨t, _e7, s, sw, _e8⟩
⋈
⟨t, _e7, s, sw, _e8⟩
⇑ (s : Segment) (t : Train) [_e7 : ON]
⟨t, _e7, s⟩
⇑ (s : Segment) (sw : Switch) [_e8 : STRAIGHT]
⟨sw, _e8, s⟩



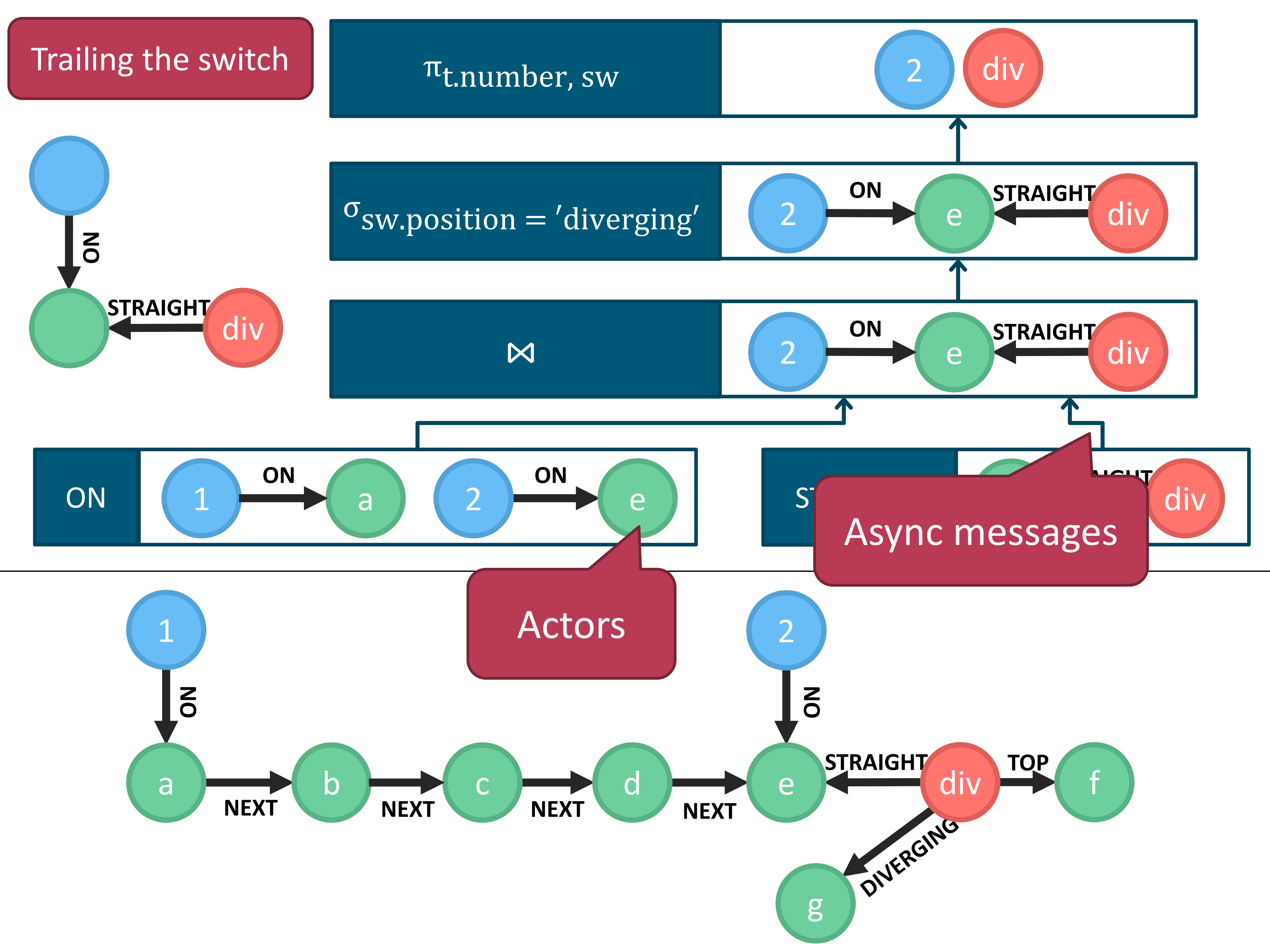
Relational algebra model
Rete network model
Rete network
Transformer and optimizer
Query deployer
VIATRA
$\pi_{\mathtt{t.number,\ sw}}$ ⟨t.number, sw⟩
$\sigma_{\mathtt{sw.position='diverging'}}$ ⟨t, _e1, s, _e2, sw⟩
$\neq$ ⟨t, _e1, s, _e2, sw⟩
$\downarrow^{\mathsf{(sw:\ Switch)}}_{\mathsf{(s)}}$ [_e2 : STRAIGHT] ⟨t, _e1, s, _e2, sw⟩
$\uparrow^{\mathsf{(s:\ Segment)}}_{\mathsf{(t)}}$ [_e1 : ON] ⟨t, _e1, s⟩
$\bigcirc_{\mathsf{(t:\ Train)}}$ ⟨t⟩
$\pi_{\mathtt{t.number,\ sw}}$ ⟨t.number, sw⟩
$\sigma_{\mathtt{sw.position='diverging'}}$ ⟨t, _e7, s, sw, _e8⟩
$\neq$ ⟨t, _e7, s, sw, _e8⟩
$\bowtie$ ⟨t, _e7, s, sw, _e8⟩
$\Uparrow^{\mathsf{(s:\ Segment)}}_{\mathsf{(t:\ Train)}}$ [_e7 : ON] ⟨t, _e7, s⟩
$\Uparrow^{\mathsf{(s:\ Segment)}}_{\mathsf{(sw:\ Switch)}}$ [_e8 : STRAIGHT] ⟨sw, _e8, s⟩



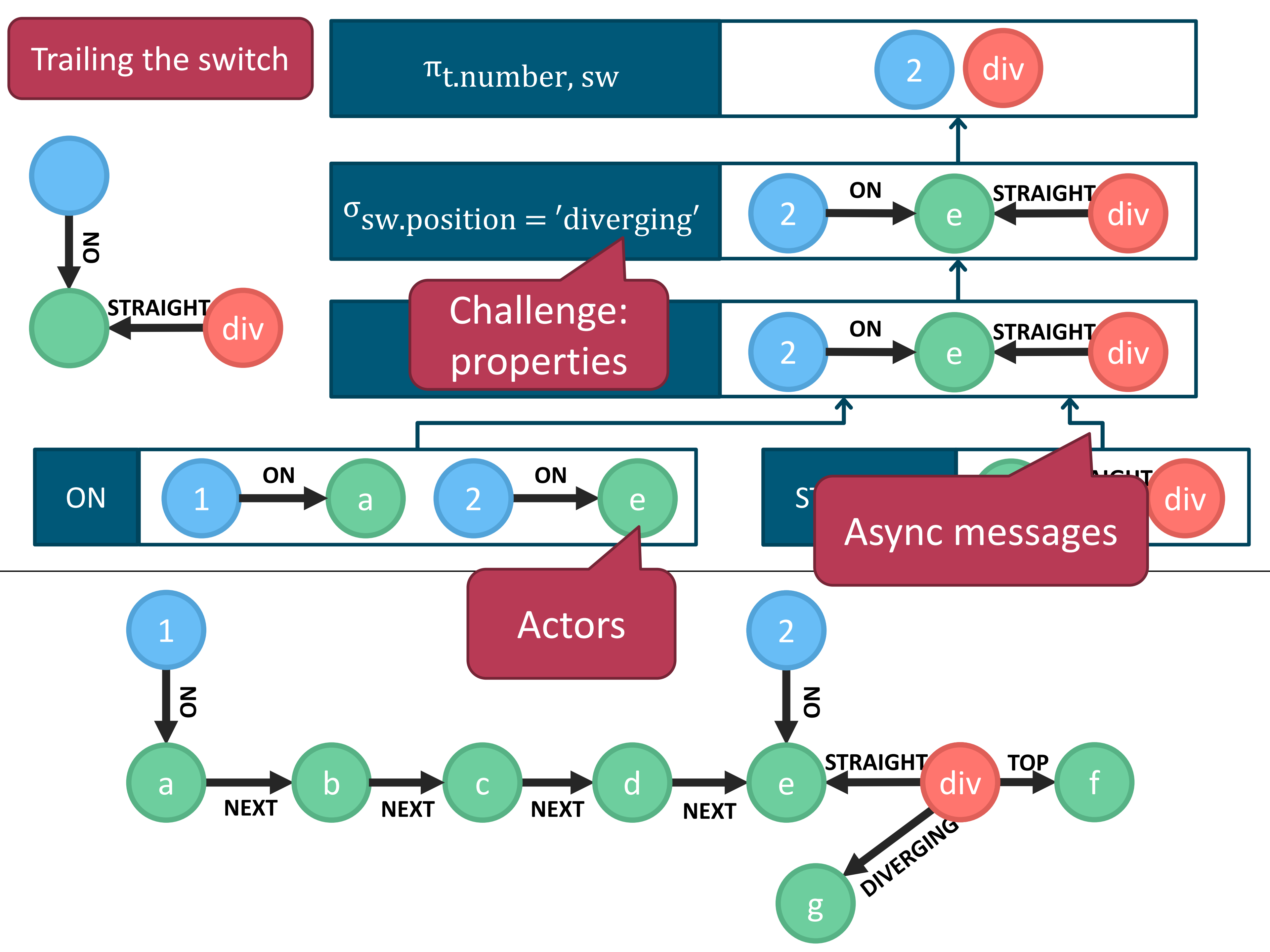
Trailing the switch
ON
STRAIGHT
div
$\pi_{t.number, sw}$
2
div
$\sigma_{sw.position = 'diverging'}$
2
ON
e
STRAIGHT
div
$\bowtie$
2
ON
e
STRAIGHT
div
ON
1
ON
a
2
ON
e
STRAIGHT
e
STRAIGHT
div
1
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
2
ON
e
STRAIGHT
div
TOP
f
DIVERGING
g



Trailing the switch
$\pi_{t.number, sw}$
$\sigma_{sw.position = 'diverging'}$
$\bowtie$
ON
STRAIGHT
div
2
e
1
a
Actors
b
c
d
f
g
NEXT
TOP
DIVERGING

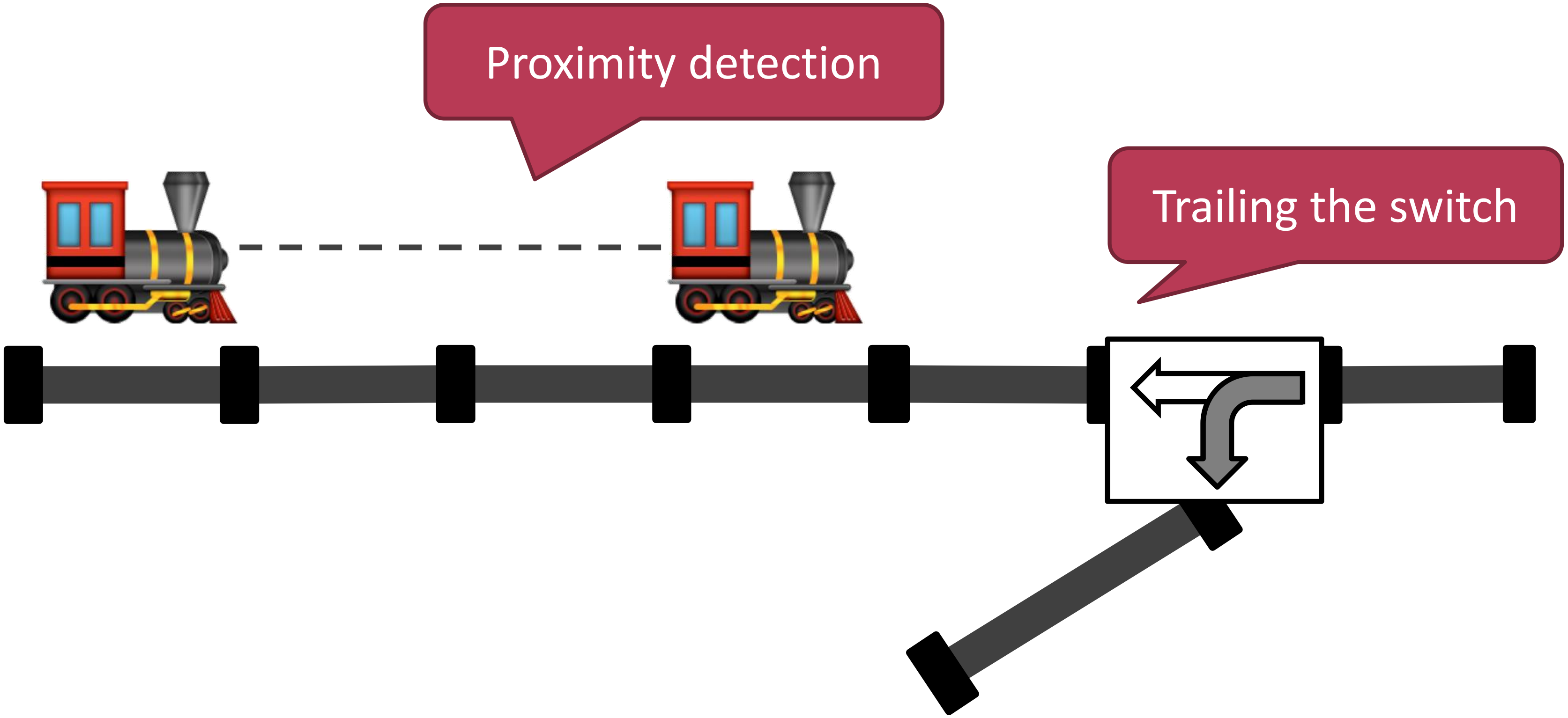


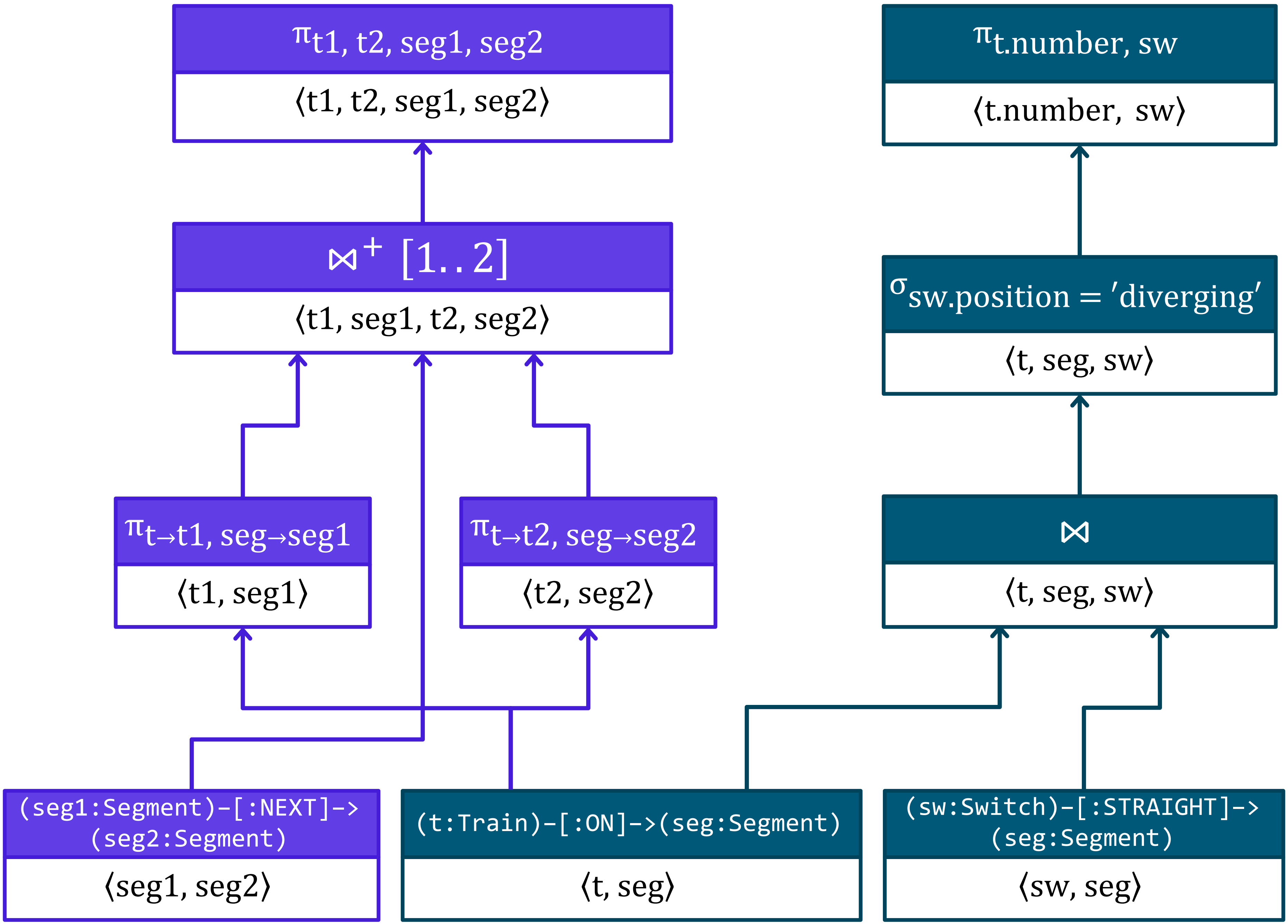
Trailing the switch
$\pi_{t.number,\ sw}$
2
div
$\sigma_{sw.position\ =\ 'diverging'}$
ON
STRAIGHT
e
⋈
ON
1
a
Async messages
Actors
NEXT
b
c
d
TOP
f
DIVERGING
g



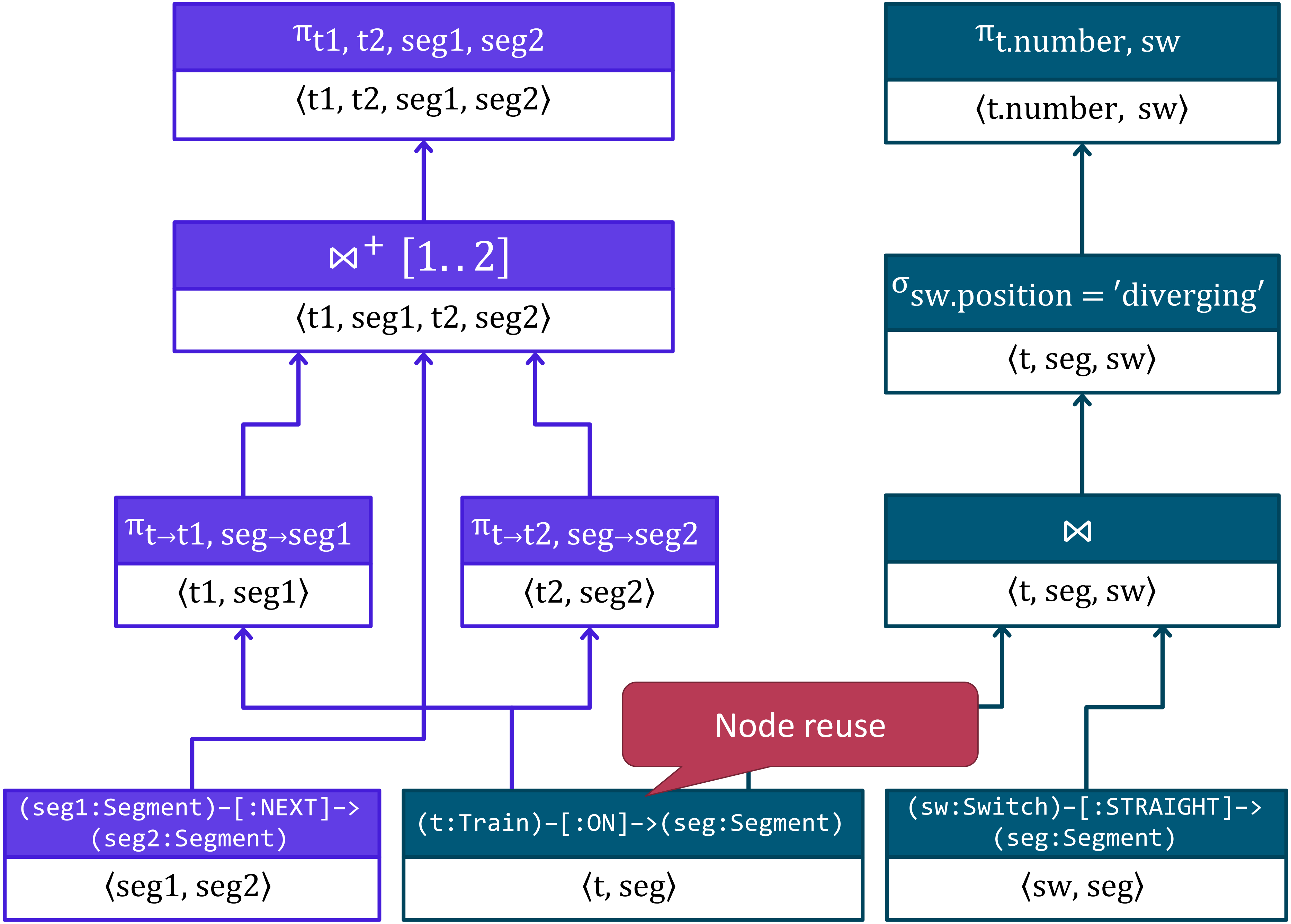
Trailing the switch
$\pi_{t.number,\ sw}$
2
div
$\sigma_{sw.position\ =\ 'diverging'}$
ON
e
STRAIGHT
Challenge: properties
ON
1
a
Async messages
Actors
NEXT
b
c
d
TOP
f
DIVERGING
g

Live railway model (revisited)

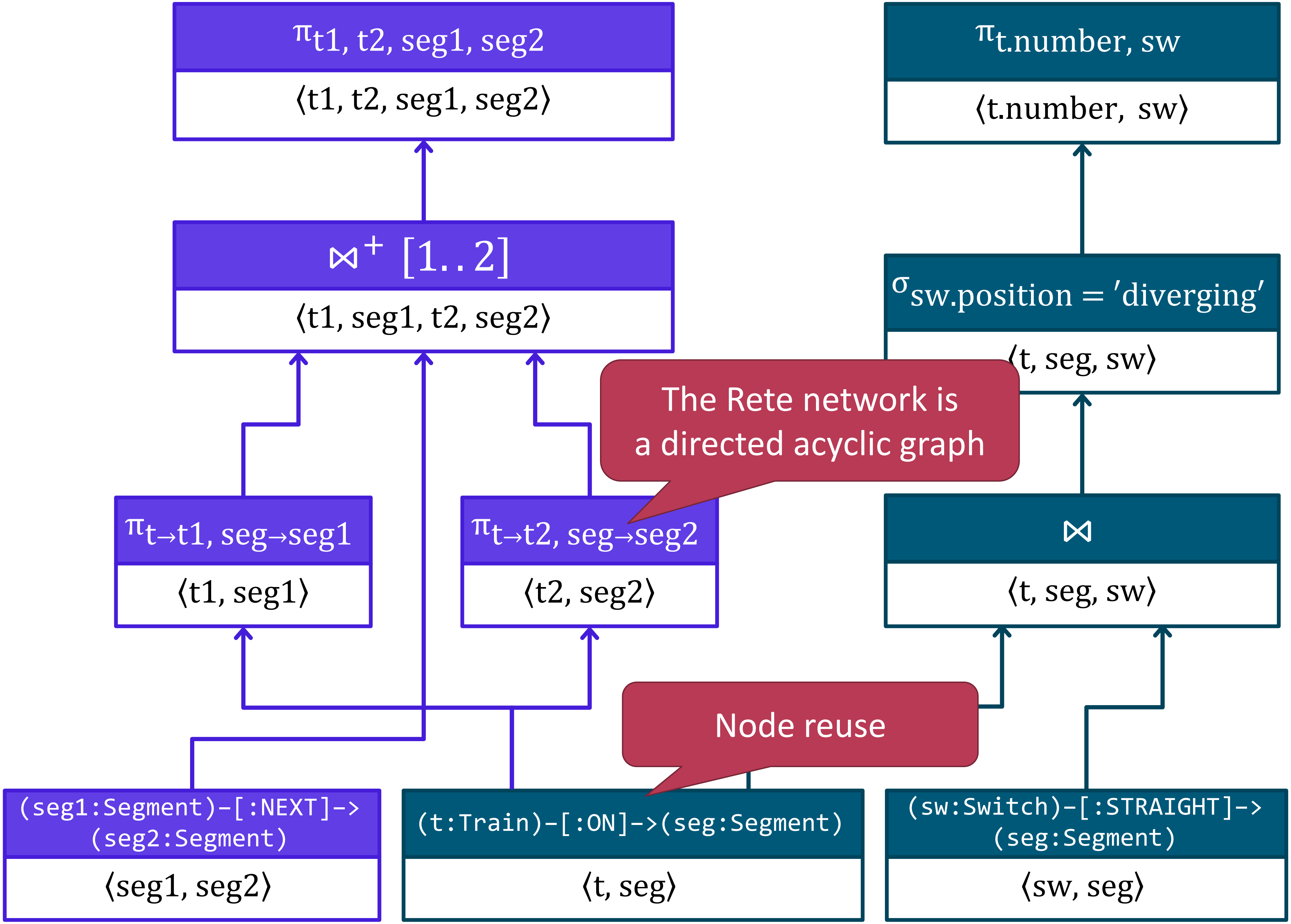




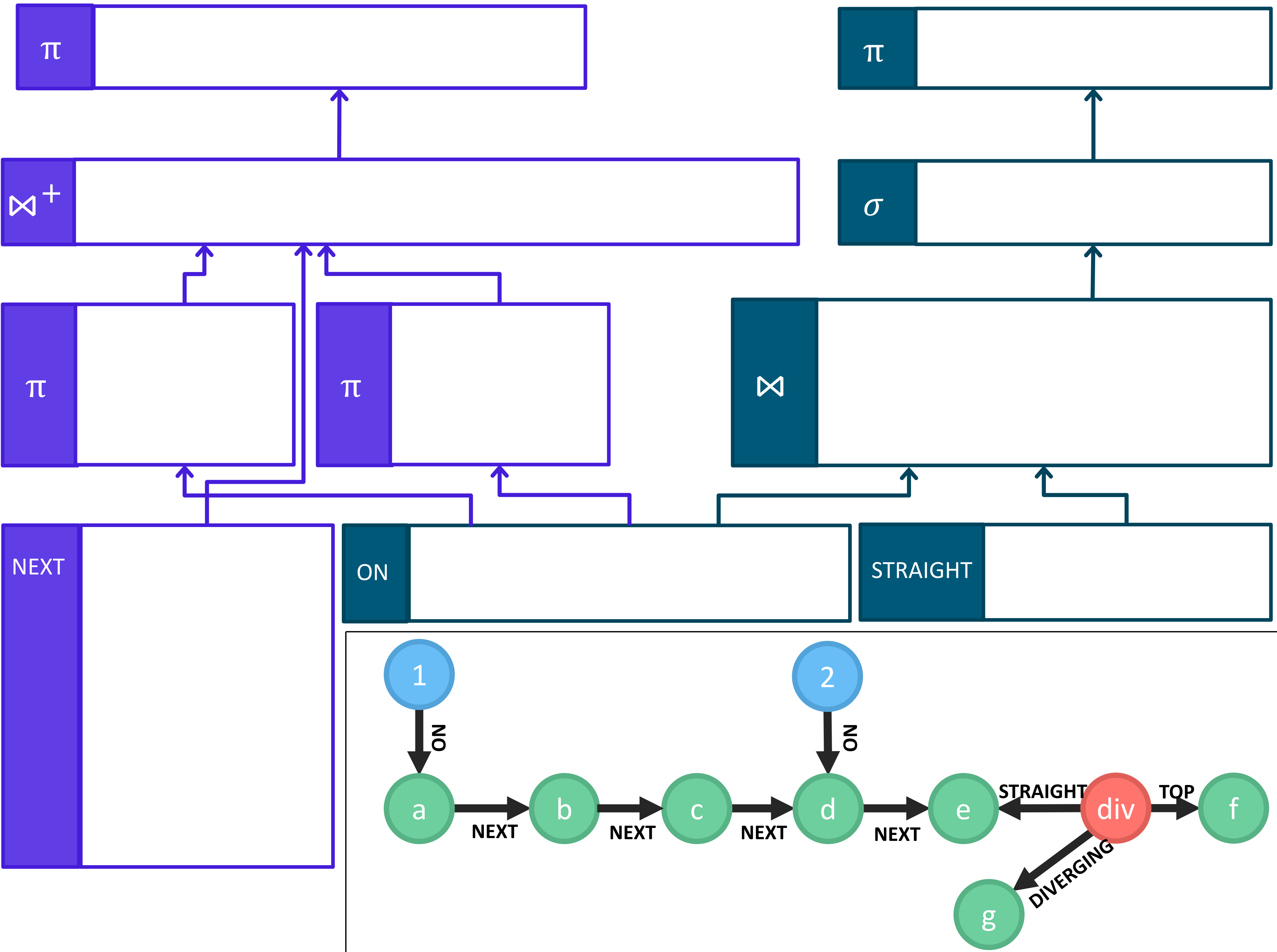
$\pi_{t1,\, t2,\, seg1,\, seg2}$
⟨t1, t2, seg1, seg2⟩
$\bowtie^{+}$ [1..2]
⟨t1, seg1, t2, seg2⟩
$\pi_{t \to t1,\, seg \to seg1}$
⟨t1, seg1⟩
$\pi_{t \to t2,\, seg \to seg2}$
⟨t2, seg2⟩
(seg1:Segment)-[:NEXT]->(seg2:Segment)
⟨seg1, seg2⟩
(t:Train)-[:ON]->(seg:Segment)
⟨t, seg⟩
(sw:Switch)-[:STRAIGHT]->(seg:Segment)
⟨sw, seg⟩
$\bowtie$
⟨t, seg, sw⟩
$\sigma_{sw.position\, =\, 'diverging'}$
⟨t, seg, sw⟩
$\pi_{t.number,\, sw}$
⟨t.number, sw⟩



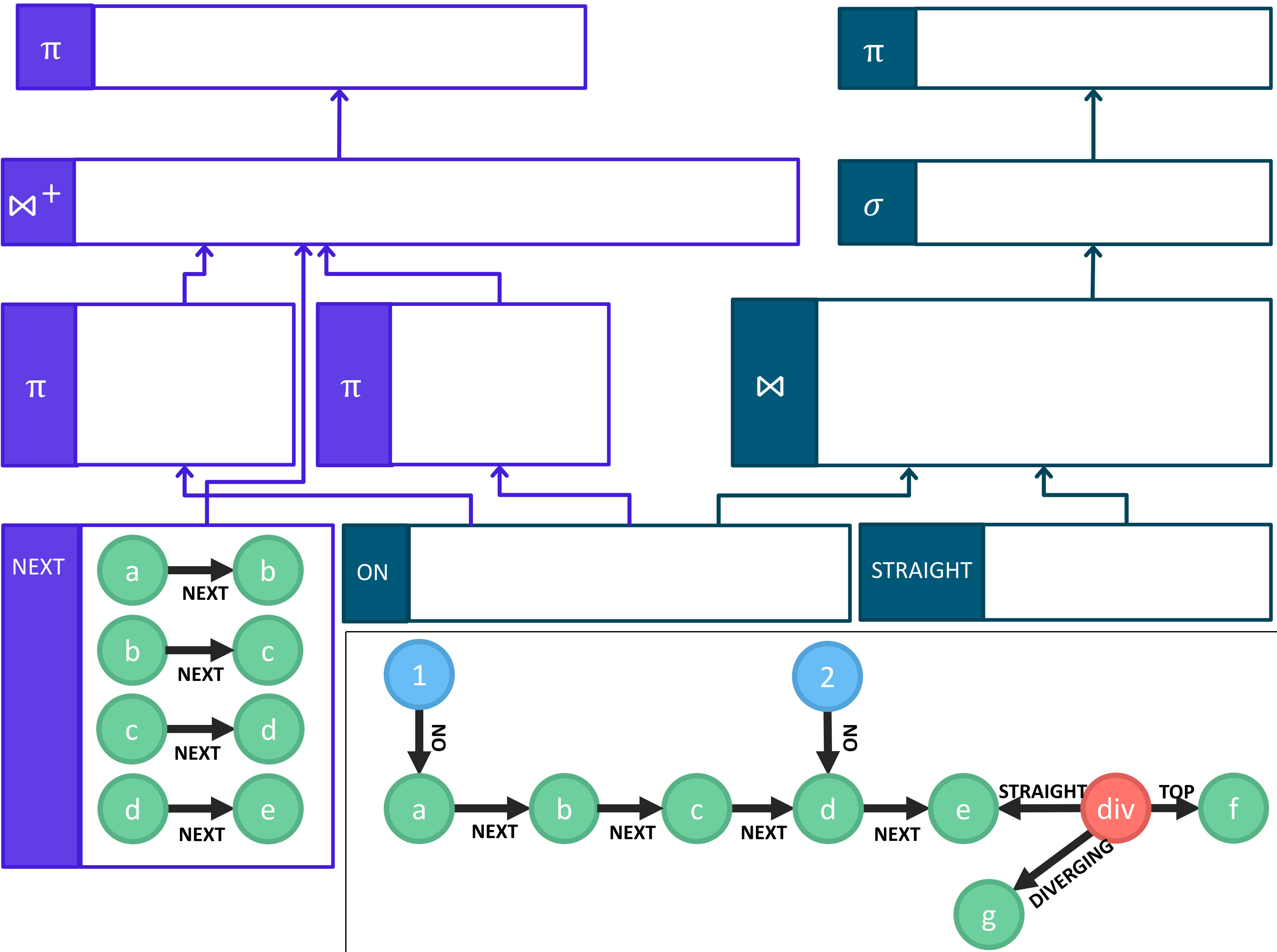
$\pi_{t1, t2, seg1, seg2}$
⟨t1, t2, seg1, seg2⟩
$\bowtie^{+}$ [1..2]
⟨t1, seg1, t2, seg2⟩
$\pi_{t \to t1, seg \to seg1}$
⟨t1, seg1⟩
$\pi_{t \to t2, seg \to seg2}$
⟨t2, seg2⟩
(seg1:Segment)-[:NEXT]-> (seg2:Segment)
⟨seg1, seg2⟩
(t:Train)-[:ON]->(seg:Segment)
⟨t, seg⟩
Node reuse
$\pi_{t.number, sw}$
⟨t.number, sw⟩
$\sigma_{sw.position = 'diverging'}$
⟨t, seg, sw⟩
$\bowtie$
⟨t, seg, sw⟩
(sw:Switch)-[:STRAIGHT]-> (seg:Segment)
⟨sw, seg⟩



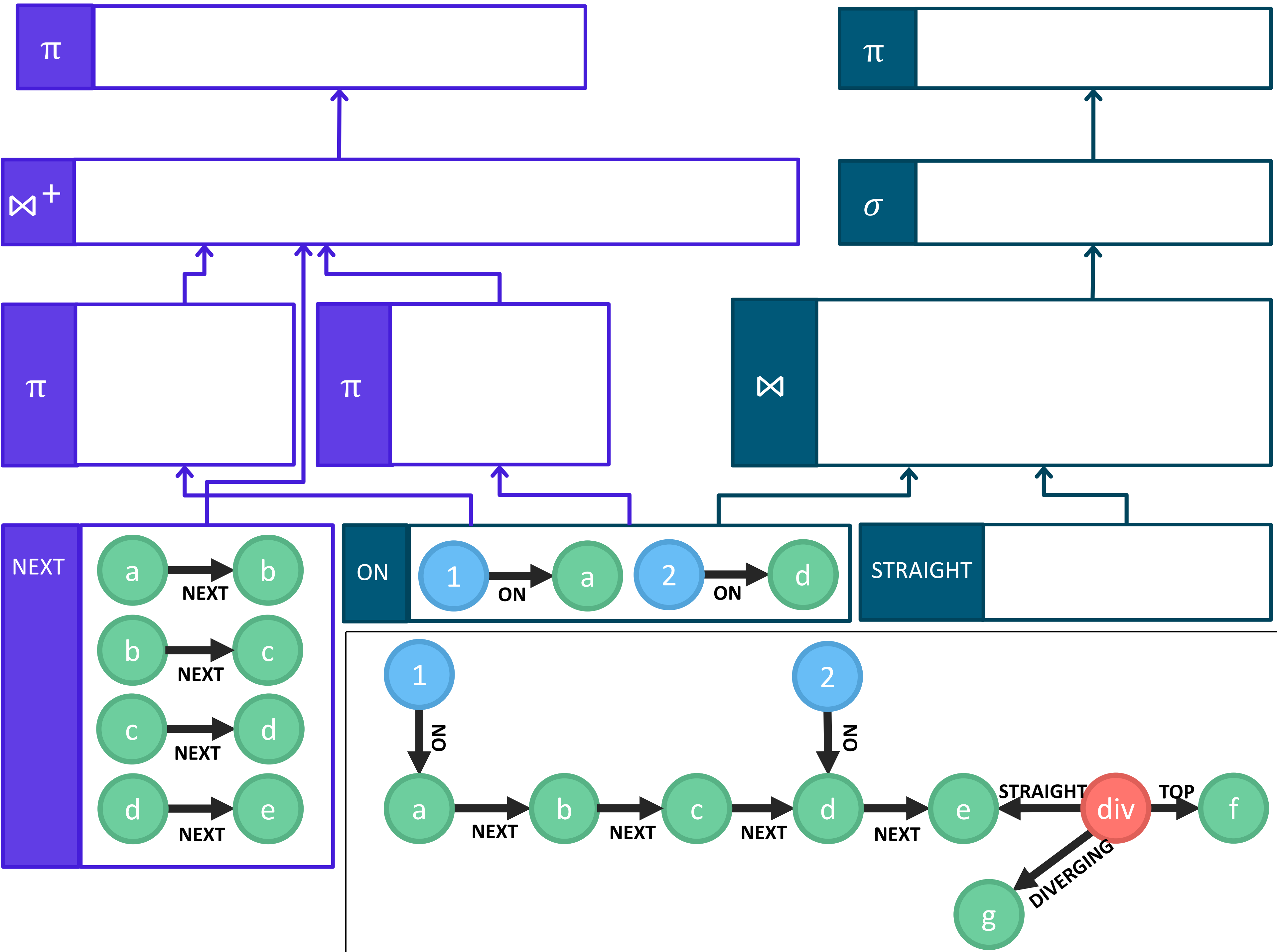
$\pi_{t1,\ t2,\ seg1,\ seg2}$
⟨t1, t2, seg1, seg2⟩
$\bowtie^{+}\ [1..2]$
⟨t1, seg1, t2, seg2⟩
$\pi_{t \to t1,\ seg \to seg1}$
⟨t1, seg1⟩
$\pi_{t \to t2,\ seg \to seg2}$
⟨t2, seg2⟩
(seg1:Segment)-[:NEXT]-> (seg2:Segment)
⟨seg1, seg2⟩
(t:Train)-[:ON]->(seg:Segment)
⟨t, seg⟩
(sw:Switch)-[:STRAIGHT]-> (seg:Segment)
⟨sw, seg⟩
$\bowtie$
⟨t, seg, sw⟩
$\sigma_{sw.position\ =\ 'diverging'}$
⟨t, seg, sw⟩
$\pi_{t.number,\ sw}$
⟨t.number, sw⟩
The Rete network is a directed acyclic graph
Node reuse



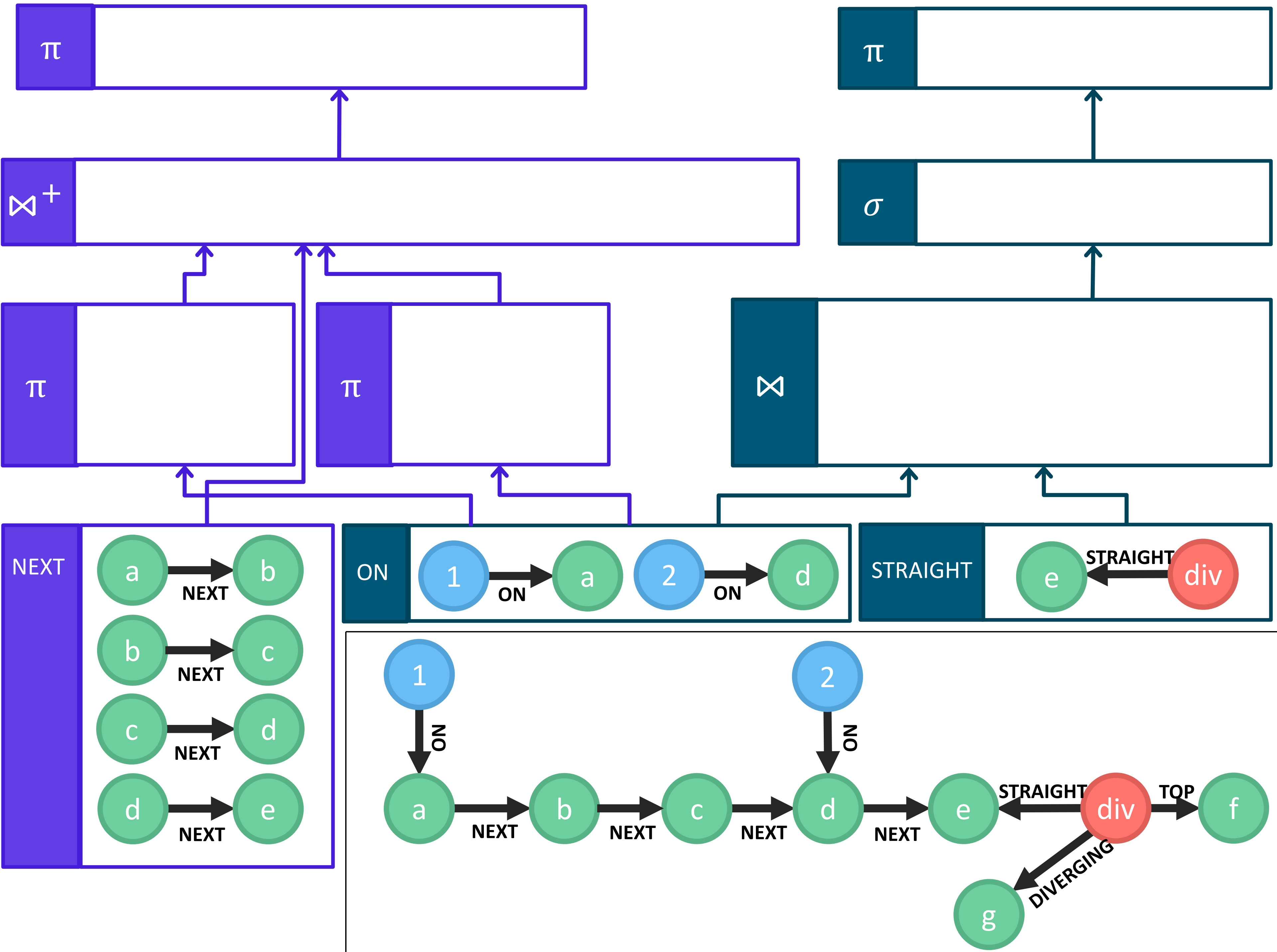
π
⋈+
π
π
NEXT
ON
π
σ
⋈
STRAIGHT
1
2
ON
ON
a
b
c
d
e
div
f
g
NEXT
NEXT
NEXT
NEXT
STRAIGHT
TOP
DIVERGING



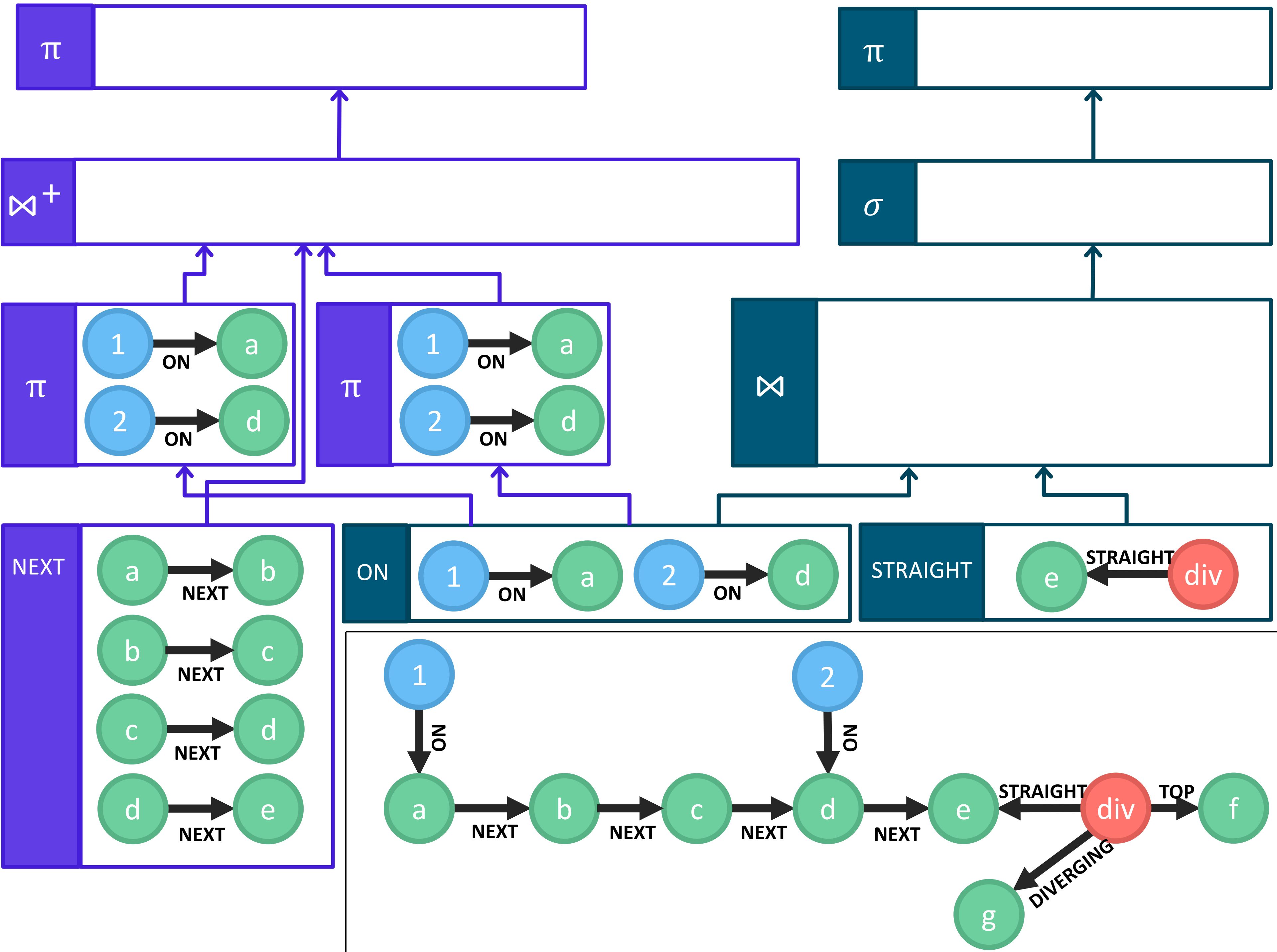
π
⋈+
π
π
NEXT
a NEXT b
b NEXT c
c NEXT d
d NEXT e
π
σ
⋈
ON
STRAIGHT
1
2
ON
ON
a NEXT b NEXT c NEXT d NEXT e
STRAIGHT
div
TOP
f
DIVERGING
g

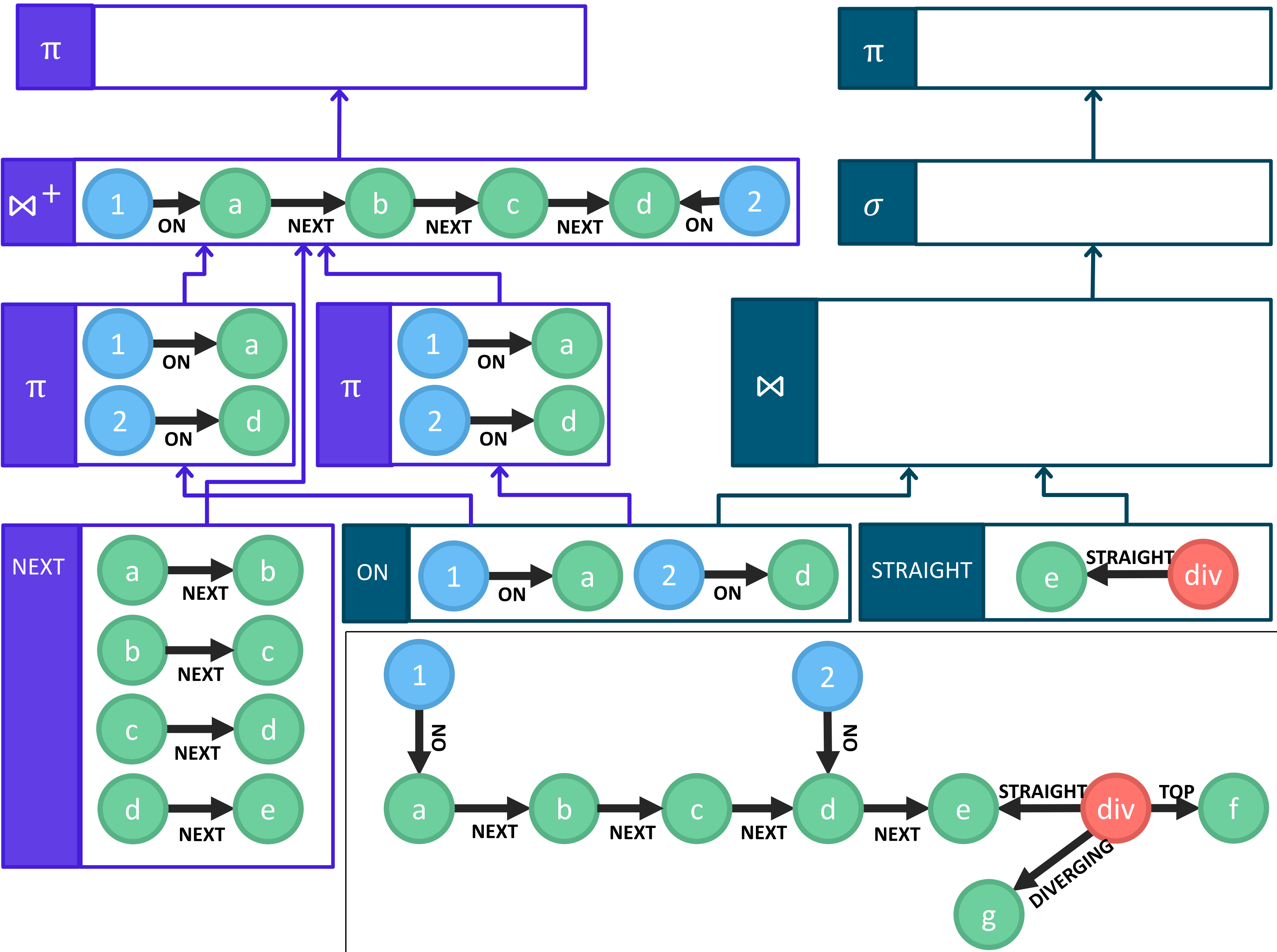


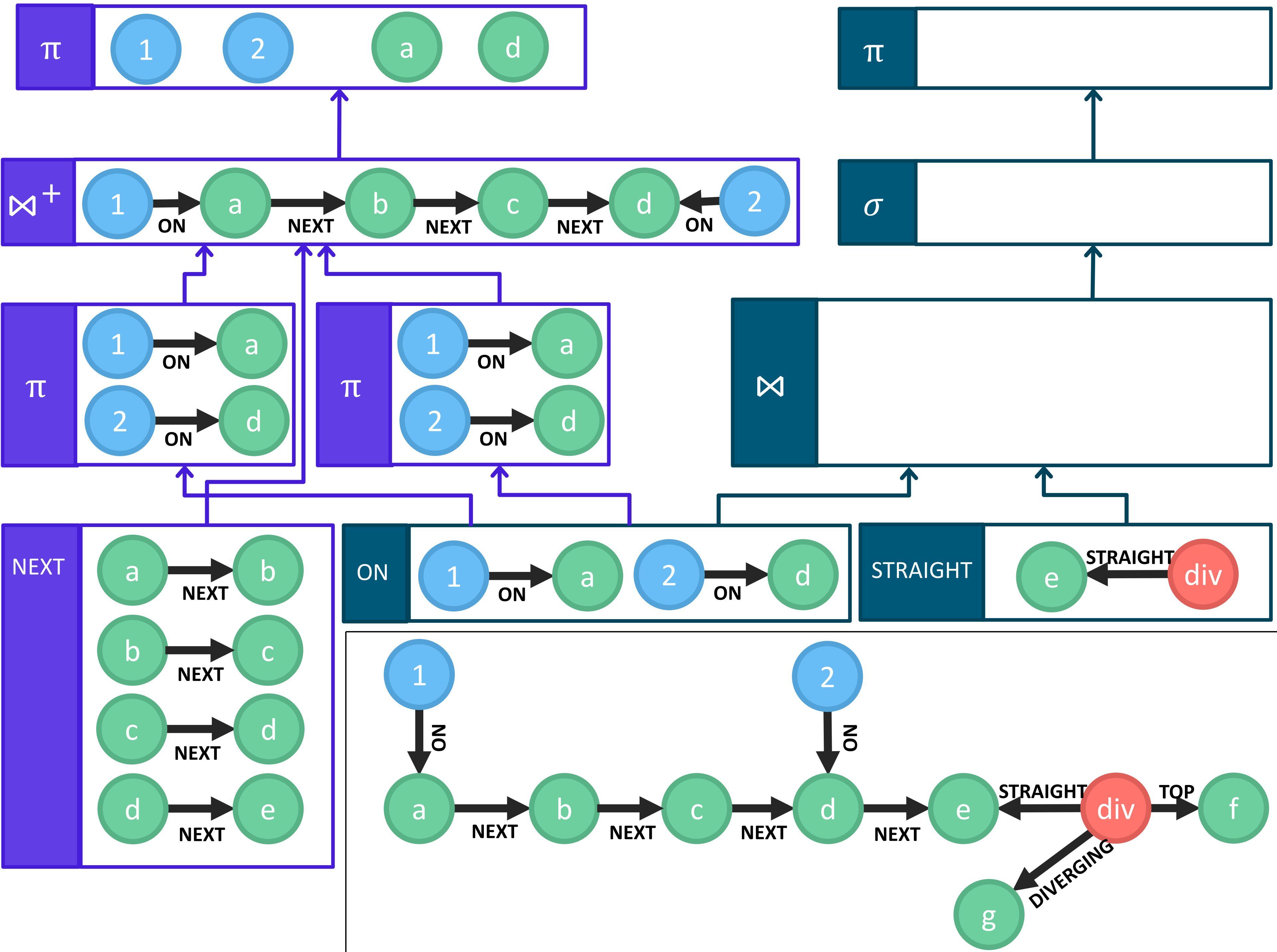
π
⋈+
π
π
NEXT
a NEXT b
b NEXT c
c NEXT d
d NEXT e
π
σ
⋈
ON
1 ON a
2 ON d
STRAIGHT
1
2
ON
ON
a NEXT b NEXT c NEXT d NEXT e
STRAIGHT
div
TOP
f
DIVERGING
g



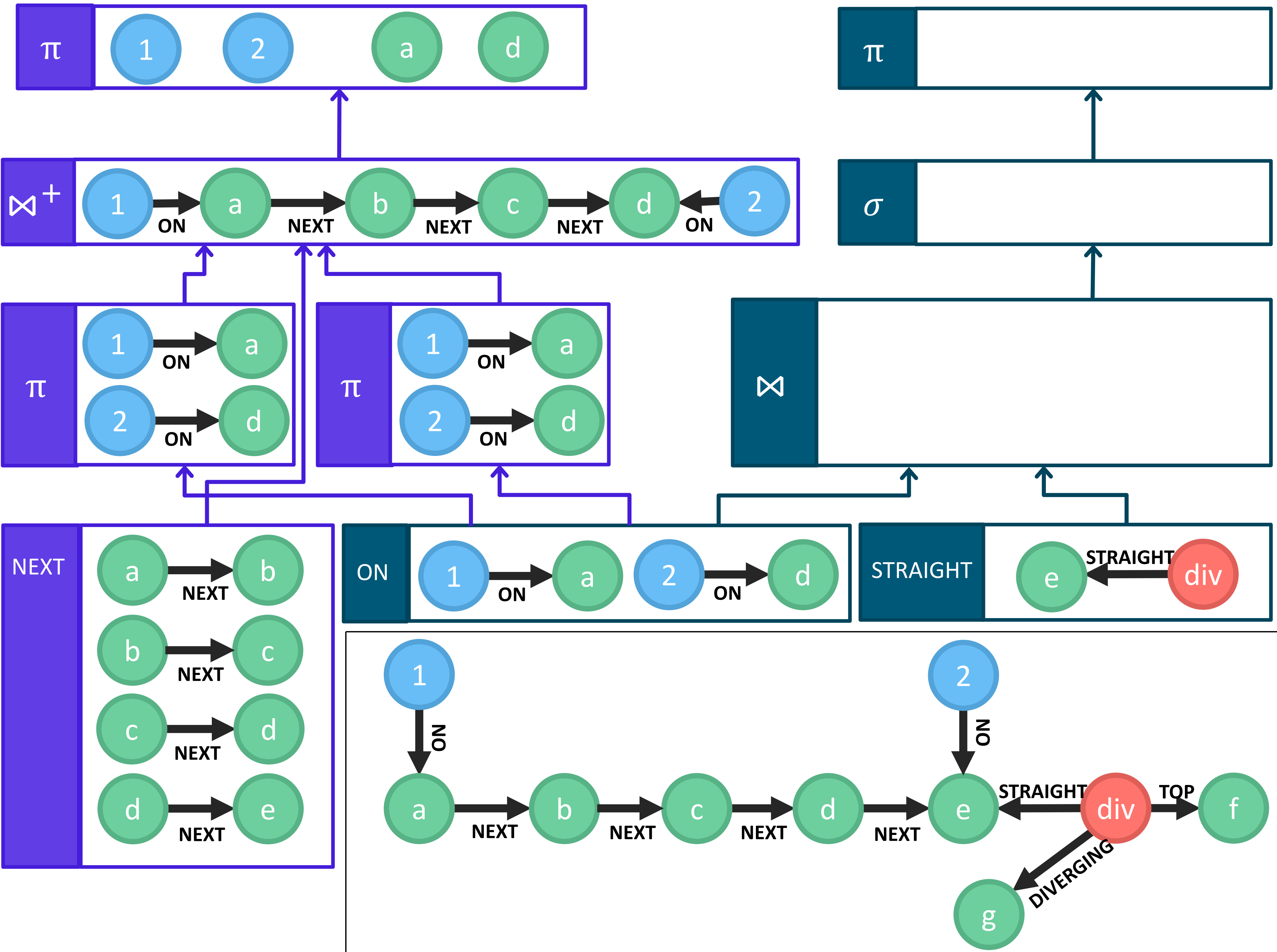
π
⋈+
π
π
NEXT
a
NEXT
b
b
NEXT
c
c
NEXT
d
d
NEXT
e
π
σ
⋈
ON
1
ON
a
2
ON
d
STRAIGHT
e
STRAIGHT
div
1
ON
2
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
e
STRAIGHT
div
TOP
f
DIVERGING
g


π
⋈+
π
1
ON
a
2
ON
d
π
1
ON
a
2
ON
d
σ
⋈
NEXT
a
NEXT
b
b
NEXT
c
c
NEXT
d
d
NEXT
e
ON
1
ON
a
2
ON
d
STRAIGHT
e
STRAIGHT
div
1
2
ON
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
e
STRAIGHT
div
TOP
f
DIVERGING
g

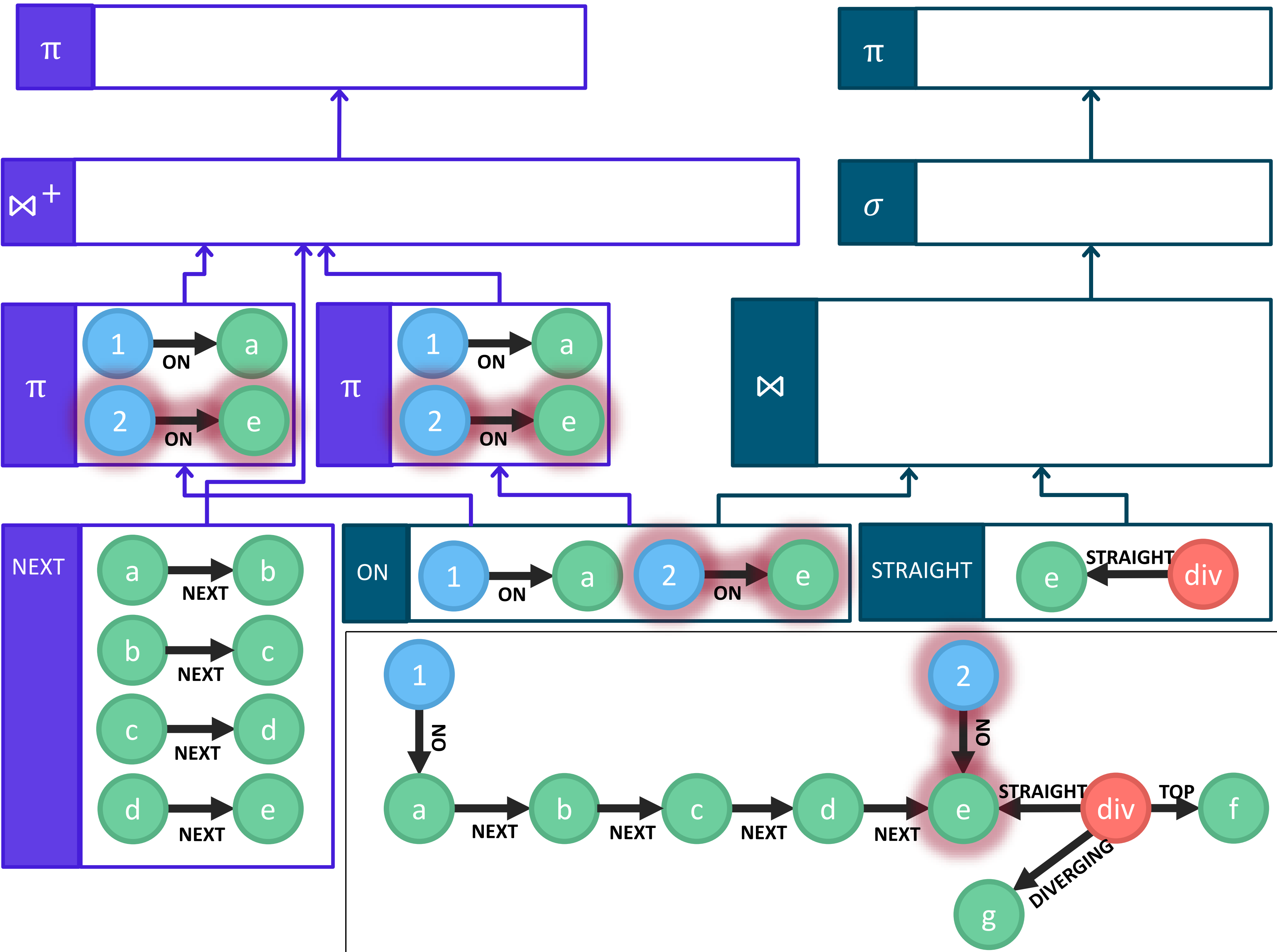

π
⋈+
1
ON
a
NEXT
b
NEXT
c
NEXT
d
ON
2
π
σ
⋈
NEXT
ON
STRAIGHT
e
div
TOP
f
DIVERGING
g



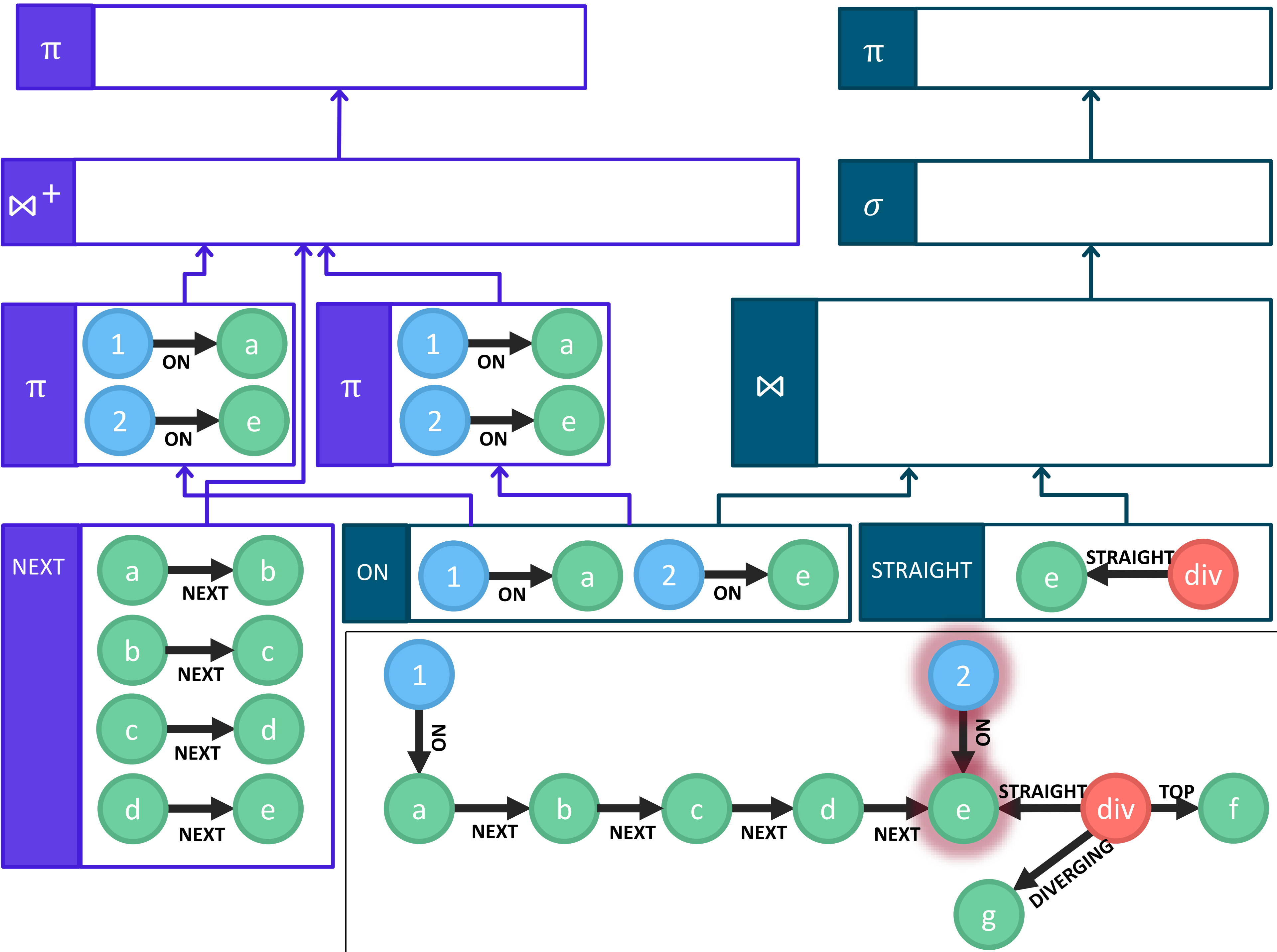
π
1
2
a
d
π
⋈+
1
ON
a
NEXT
b
NEXT
c
NEXT
d
ON
2
σ
π
1
ON
a
2
ON
d
π
1
ON
a
2
ON
d
⋈
NEXT
a
NEXT
b
b
NEXT
c
c
NEXT
d
d
NEXT
e
ON
1
ON
a
2
ON
d
STRAIGHT
e
STRAIGHT
div
1
ON
2
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
e
STRAIGHT
div
TOP
f
DIVERGING
g

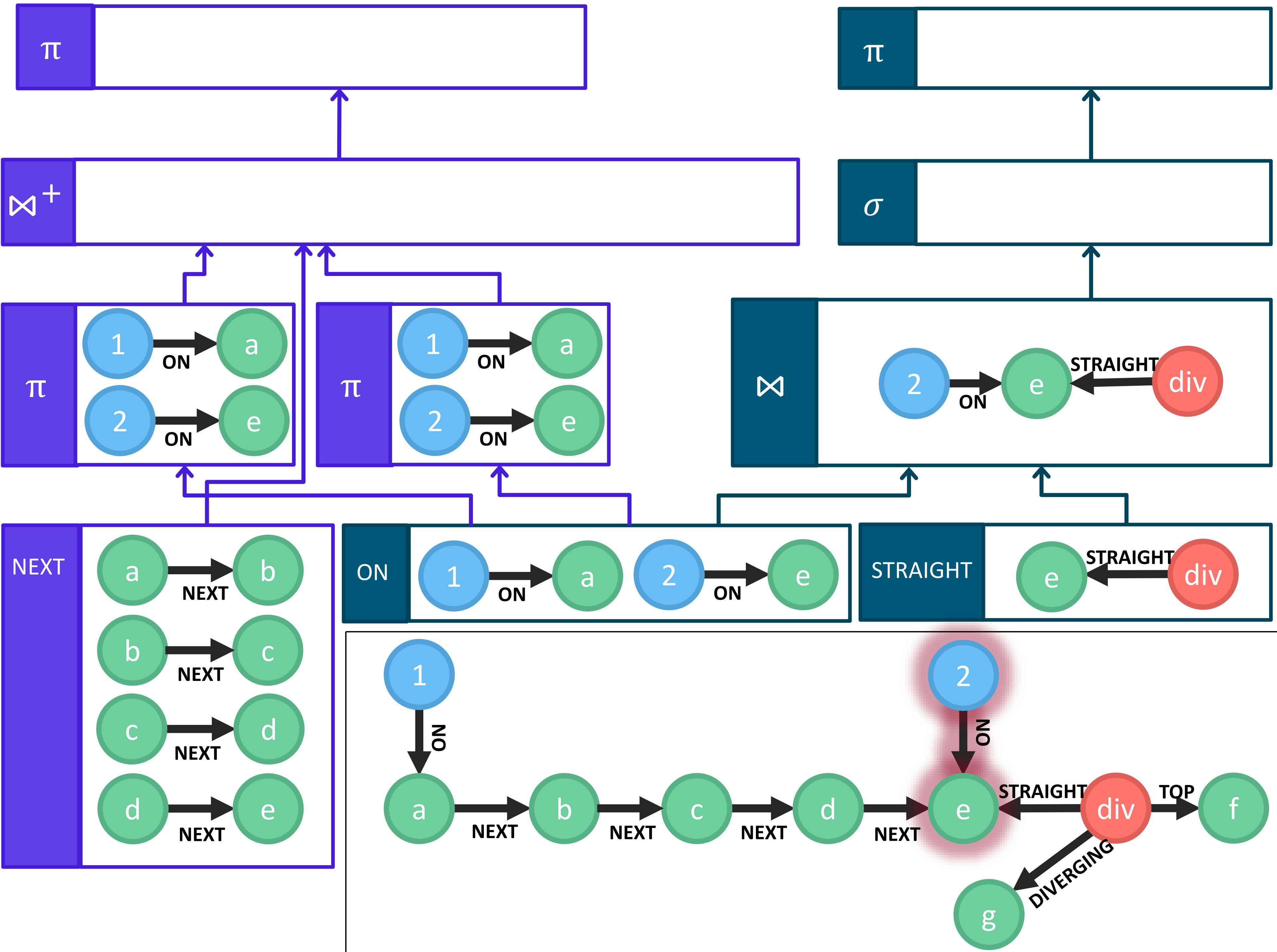


π
1
2
a
d
π
⋈+
ON
NEXT
NEXT
NEXT
ON
σ
⋈
NEXT
ON
STRAIGHT
e
div
STRAIGHT
TOP
DIVERGING
f
g
b
c

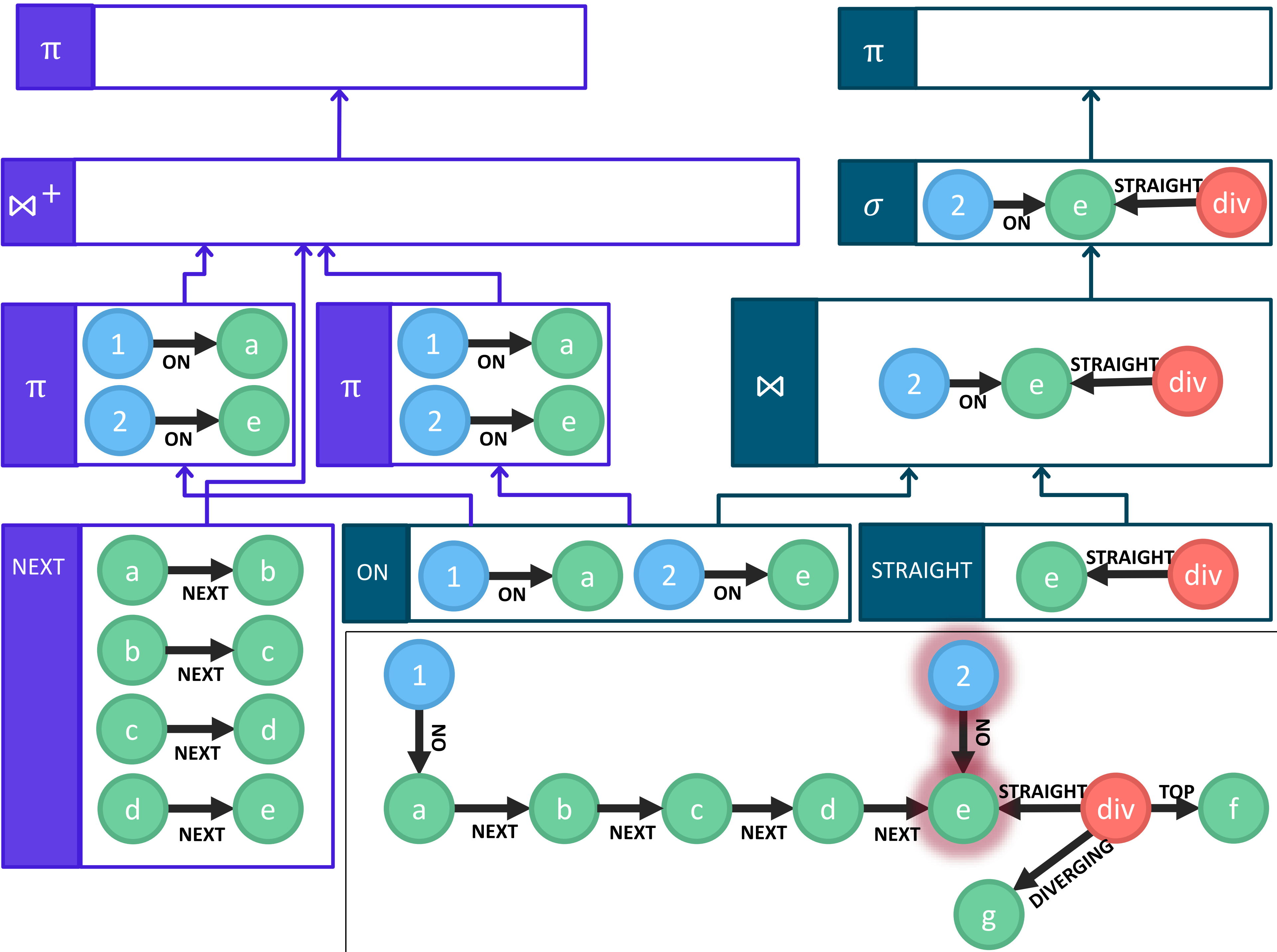


π
⋈+
π
1 ON a
2 ON e
π
1 ON a
2 ON e
NEXT
a NEXT b
b NEXT c
c NEXT d
d NEXT e
ON
1 ON a
2 ON e
π
σ
⋈
STRAIGHT
e STRAIGHT div
1 ON a
2 ON e
a NEXT b NEXT c NEXT d NEXT e
div STRAIGHT e
div TOP f
div DIVERGING g

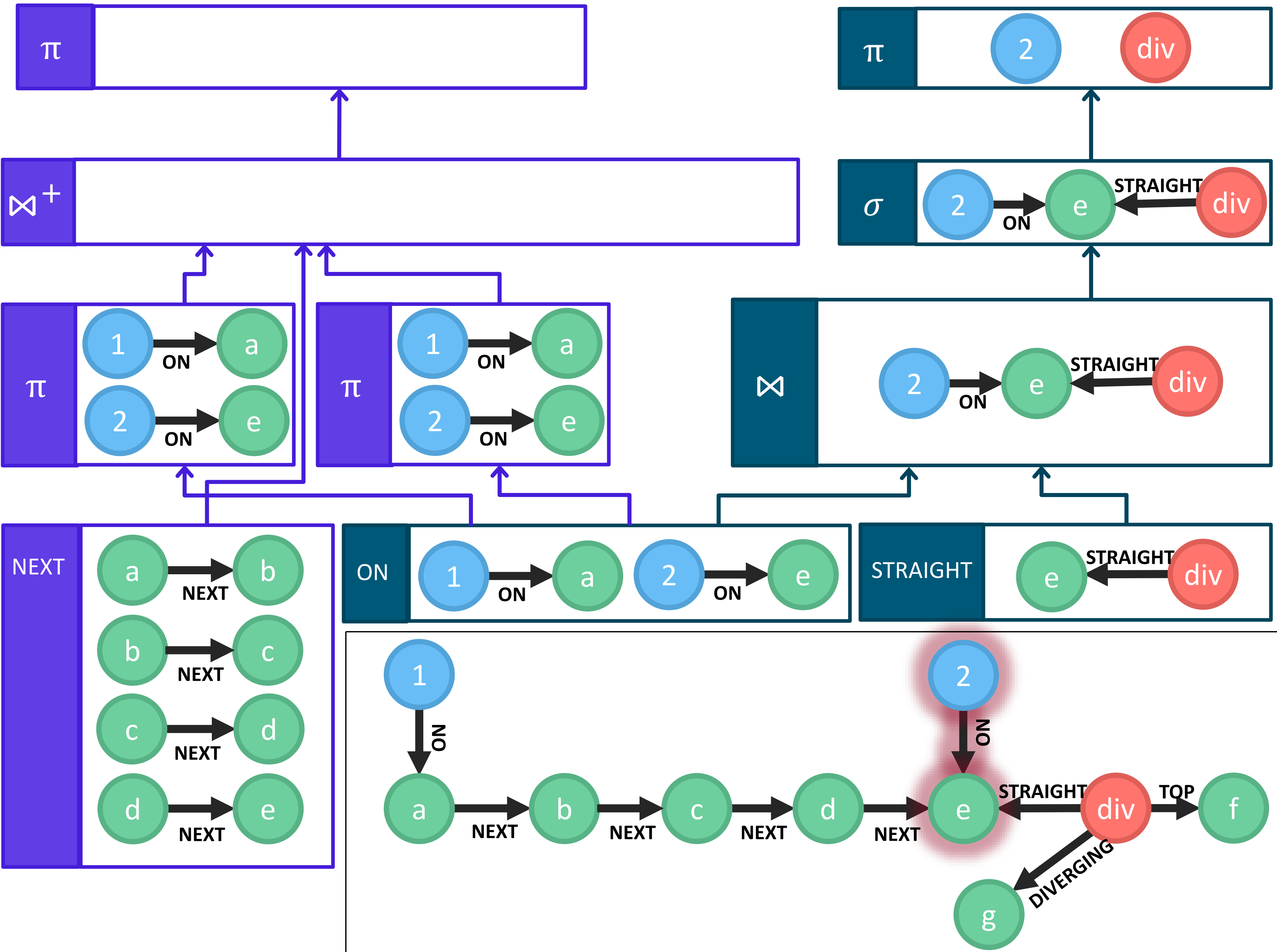

π
π
⋈+
σ
π
1 ON a
2 ON e
π
1 ON a
2 ON e
⋈
NEXT
a NEXT b
b NEXT c
c NEXT d
d NEXT e
ON
1 ON a
2 ON e
STRAIGHT
e STRAIGHT div
1 ON a
2 ON e
a NEXT b NEXT c NEXT d NEXT e
div STRAIGHT e
div TOP f
div DIVERGING g



π
⋈+
π
π
NEXT
ON
STRAIGHT
⋈
σ
π
1 ON a
2 ON e
NEXT
STRAIGHT
ON
DIVERGING
TOP
div



π
⋈+
π
1
ON
a
2
ON
e
π
1
ON
a
2
ON
e
NEXT
a
NEXT
b
b
NEXT
c
c
NEXT
d
d
NEXT
e
ON
1
ON
a
2
ON
e
π
σ
2
ON
e
STRAIGHT
div
⋈
2
ON
e
STRAIGHT
div
STRAIGHT
e
STRAIGHT
div
1
ON
2
ON
a
NEXT
b
NEXT
c
NEXT
d
NEXT
e
STRAIGHT
div
TOP
f
DIVERGING
g



π
⋈+
π
1 ON a
2 ON e
π
1 ON a
2 ON e
⋈
2 ON e STRAIGHT div
σ
2 ON e STRAIGHT div
π
2 div
NEXT
a NEXT b
b NEXT c
c NEXT d
d NEXT e
ON
1 ON a 2 ON e
STRAIGHT
e STRAIGHT div
1 ON a
2 ON e
a NEXT b NEXT c NEXT d NEXT e
div STRAIGHT e
div TOP f
div DIVERGING g

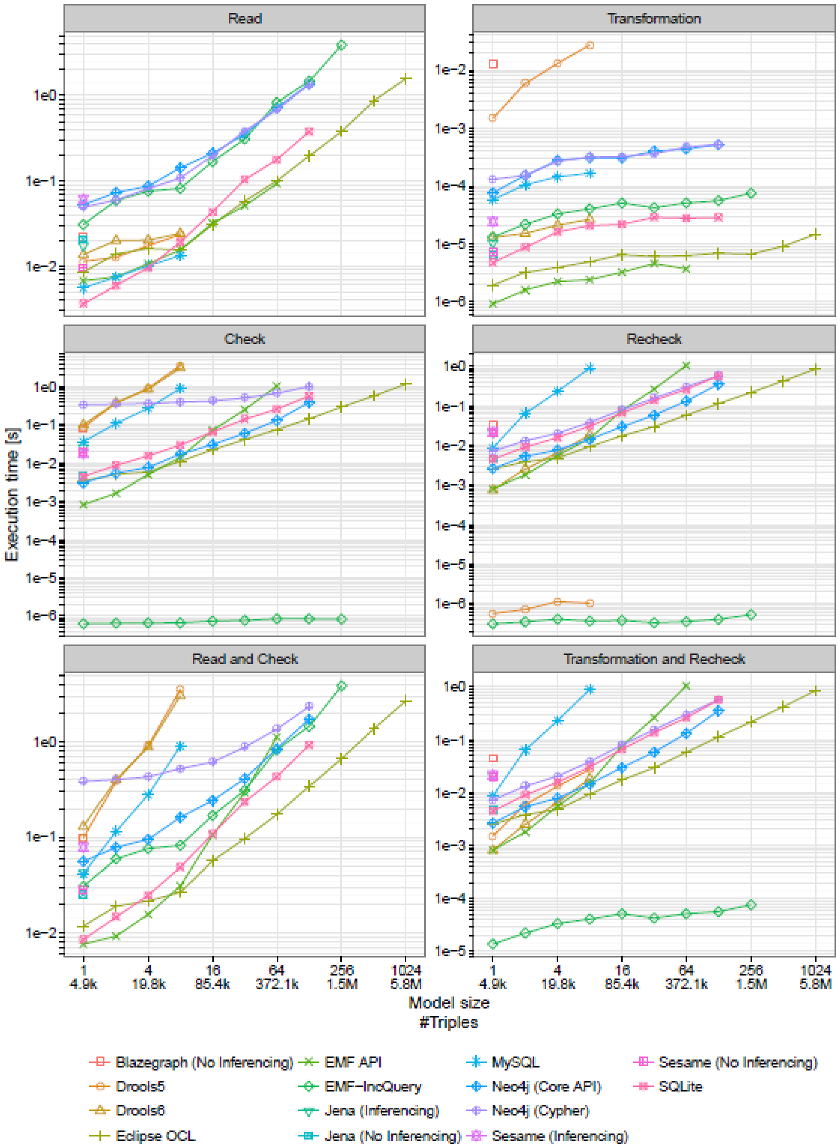
ingraph use cases

Standing queries on large & quickly changing graphs

- Model validation: The Train Benchmark
- Static analysis of JavaScript source code
- Fraud detection
- IT infrastructure monitoring

Model validation: the Train Benchmark

- Scalable graph generator
  - EMF
  - Property graph
  - RDF
  - SQL
- Realistic workload
  - 6 validation queries
  - 12 transformations
  - Implemented for 12+ tools
- Visualization and reporting

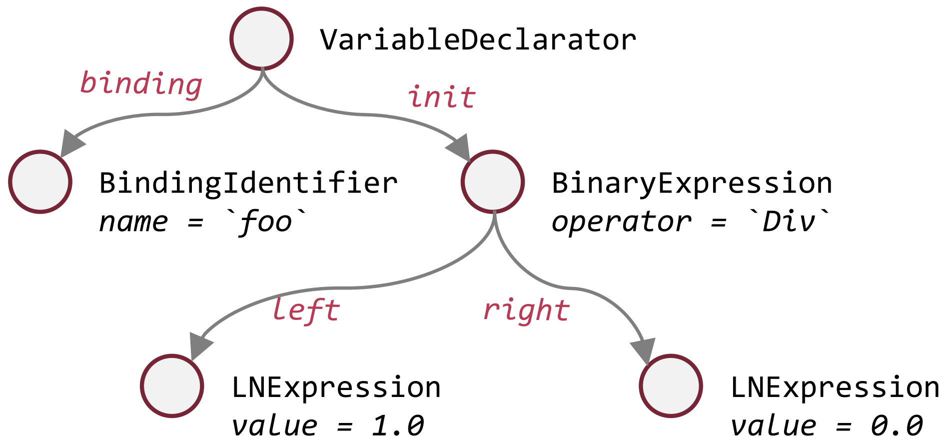


Static analysis of JavaScript

```
var foo = 1 / 0;
```

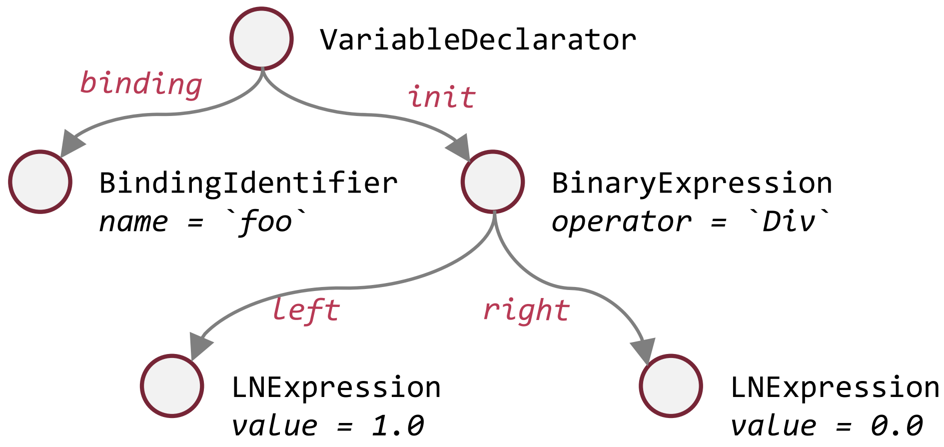
Static analysis of JavaScript

```
var foo = 1 / 0;
```



Static analysis of JavaScript

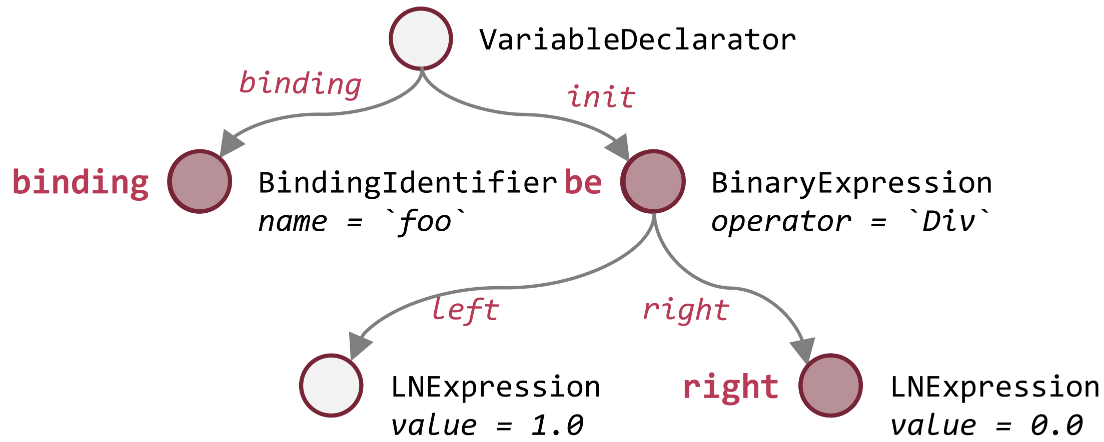
```
var foo = 1 / 0;
```




```
MATCH
  (binding:BindingIdentifier)<-[:binding]-()-->
  (be:BinaryExpression)-[:right]->
  (right:LNExpression)
WHERE be.operator = 'Div'
  AND right.value = 0.0
RETURN binding
```

Static analysis of JavaScript

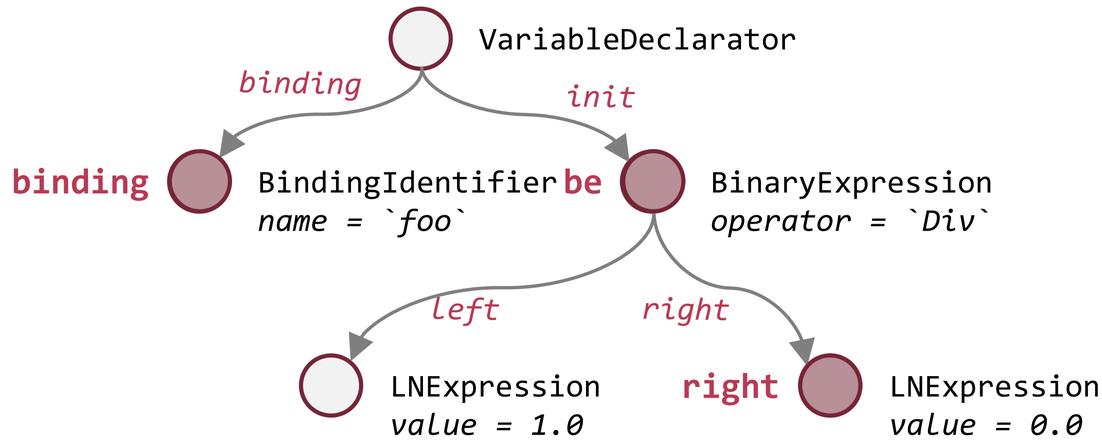
```
var foo = 1 / 0;
```




```
MATCH
  (binding:BindingIdentifier)<-[:binding]-()-->
  (be:BinaryExpression)-[:right]->
  (right:LNExpression)
WHERE be.operator = 'Div'
  AND right.value = 0.0
RETURN binding
```

Static analysis of JavaScript

```
var foo = 1 / 0;
```




```
MATCH
  (binding:BindingIdentifier)<-[:binding]-()-->
  (be:BinaryExpression)-[:right]->
  (right:LNExpression)
WHERE be.operator = 'Div'
  AND right.value = 0.0
RETURN binding
```

- Dead code detection
- Type inferencing
- Test generation

Project roadmap

Project roadmap

- Until now
  - formalization for most standard openCypher constructs [technical report]
  - research prototype for model validation queries

Project roadmap

- Until now
  - formalization for most standard openCypher constructs [technical report]
  - research prototype for model validation queries
- 2017 Q1
  - Use TCK tests for feature coverage & debugging

Project roadmap

- Until now
  - formalization for most standard openCypher constructs [technical report]
  - research prototype for model validation queries
- 2017 Q1
  - Use TCK tests for feature coverage & debugging
- 2017 Q2
  - prototype for core openCypher constructs
  - implement optimizer
  - publish benchmark results [conference paper]

Project roadmap

- Until now
  - formalization for most standard openCypher constructs [technical report]
  - research prototype for model validation queries
- 2017 Q1
  - Use TCK tests for feature coverage & debugging
- 2017 Q2
  - prototype for core op
  - implement optimizer
  - publish benchmark results [conference paper]

Spark Catalyst or Apache Calcite

Open-Source Projects

Incremental Graph Engine:

https://github.com/ftsrg/ingraph

Train Benchmark:

https://github.com/ftsrg/trainbenchmark

BME-MODES3:

https://github.com/ftsrg/bme-modes3

Available under EPL v1.0.

Ω