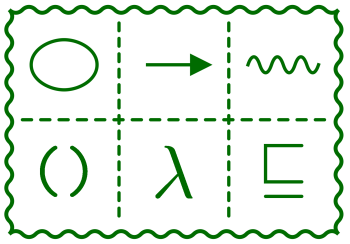


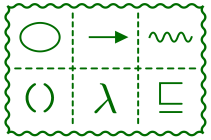
SyncStitch 3

Hisabumi HATSUGAI

hatsugai@principia-m.com

PRINCIPIA Limited

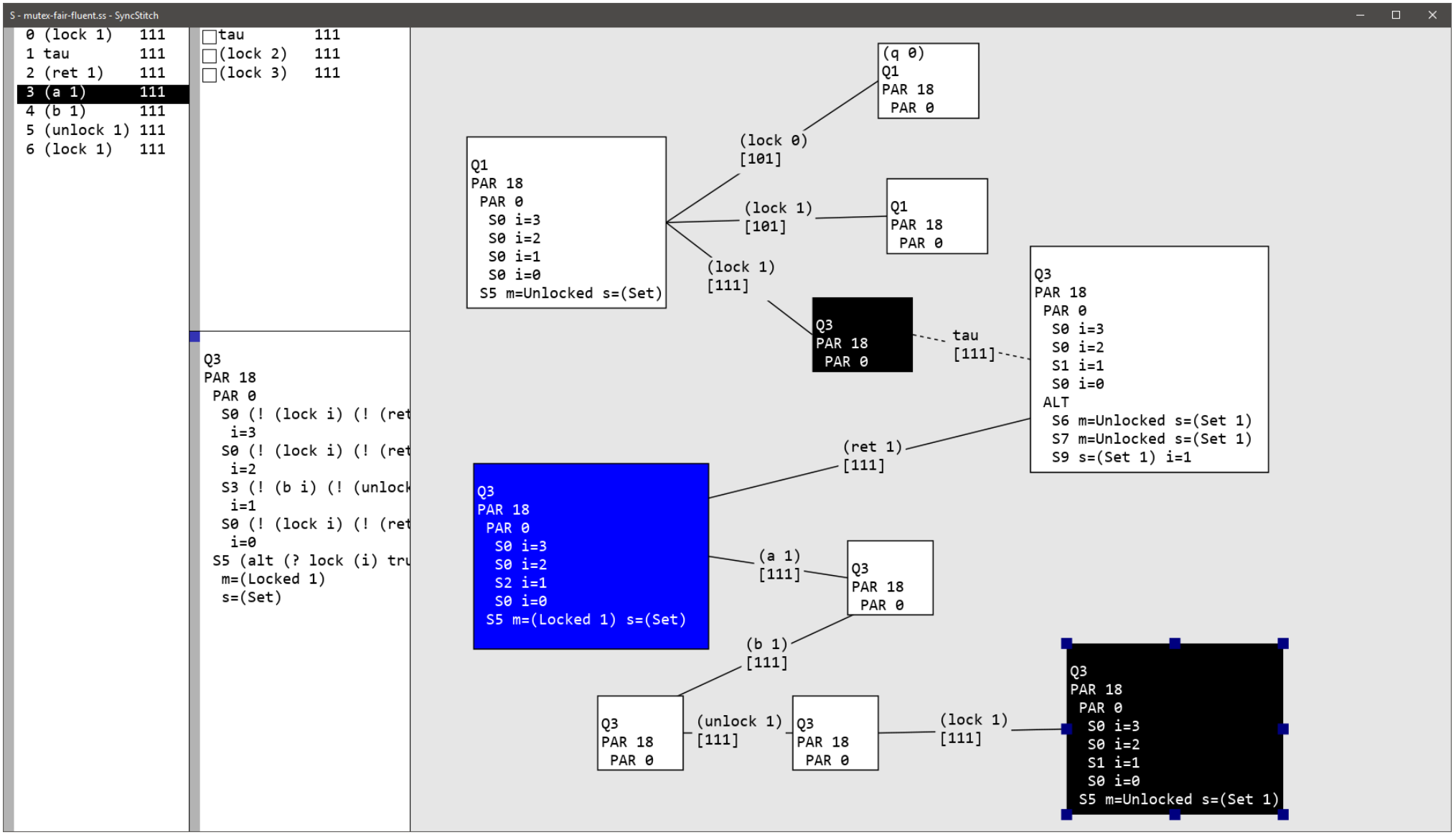
http://www.principia-m.com/

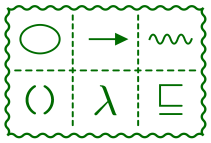


SyncStitch 3

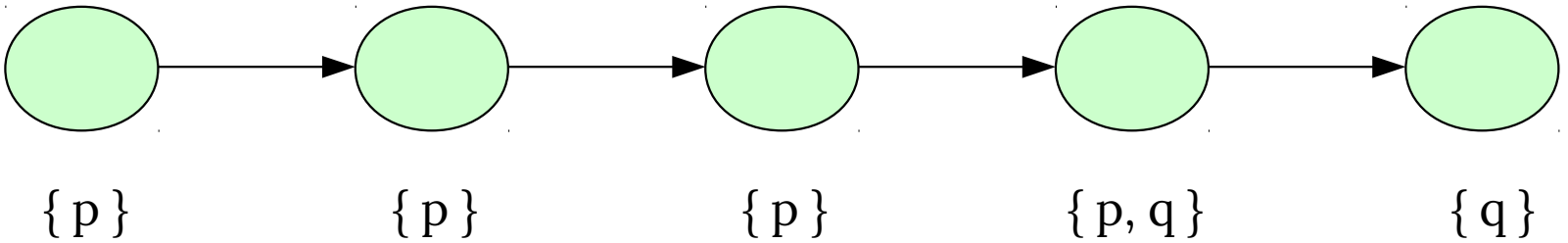
- Redesigned Process Explorer
- Model checking with temporal logics
  - Temporal Logics
    - Linear Temporal Logic (LTL)
    - Computation Tree Logic (CTL)
  - Atomic Propositions
    - Observants
    - Fluents

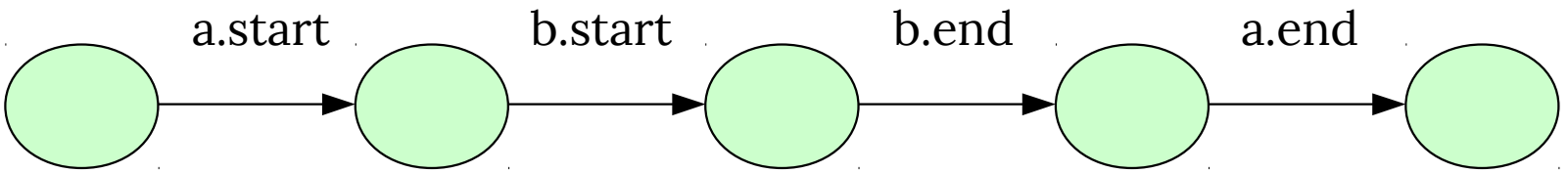
Process Explorer

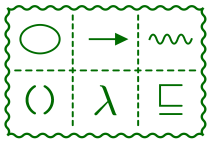




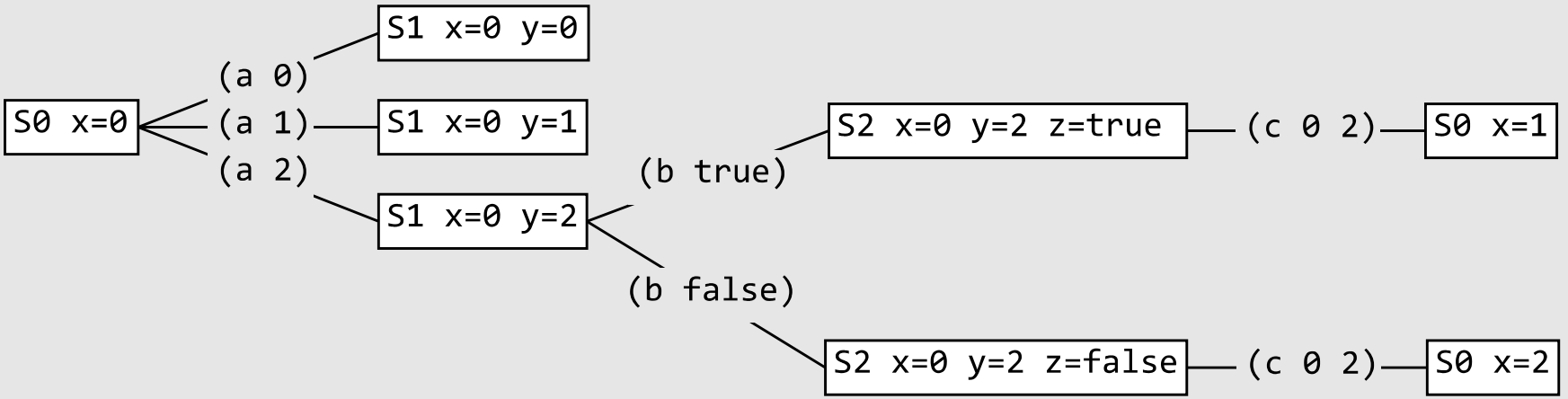
State vs Action (transition, event)



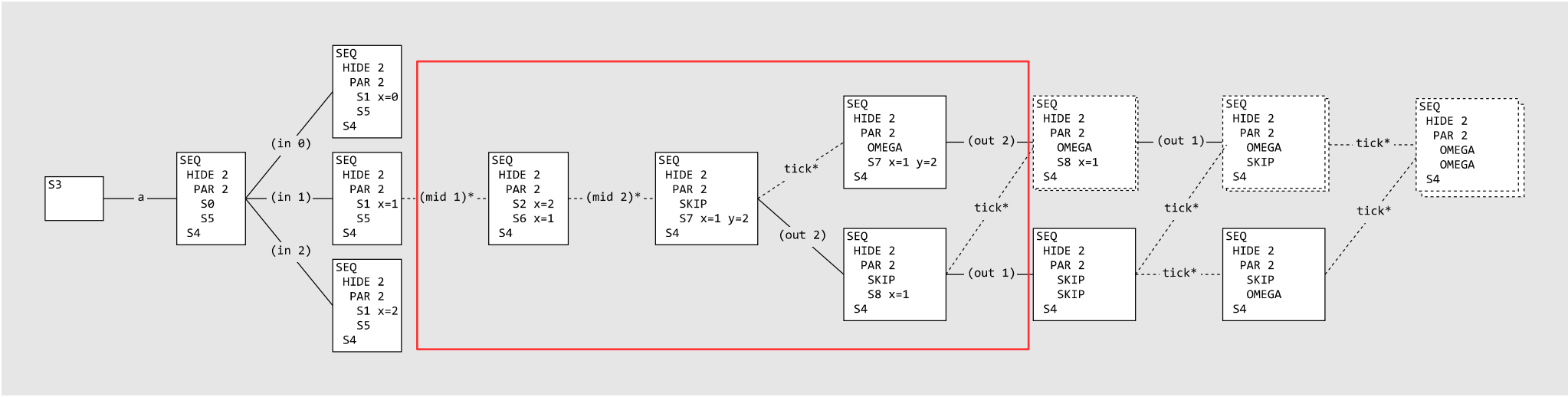


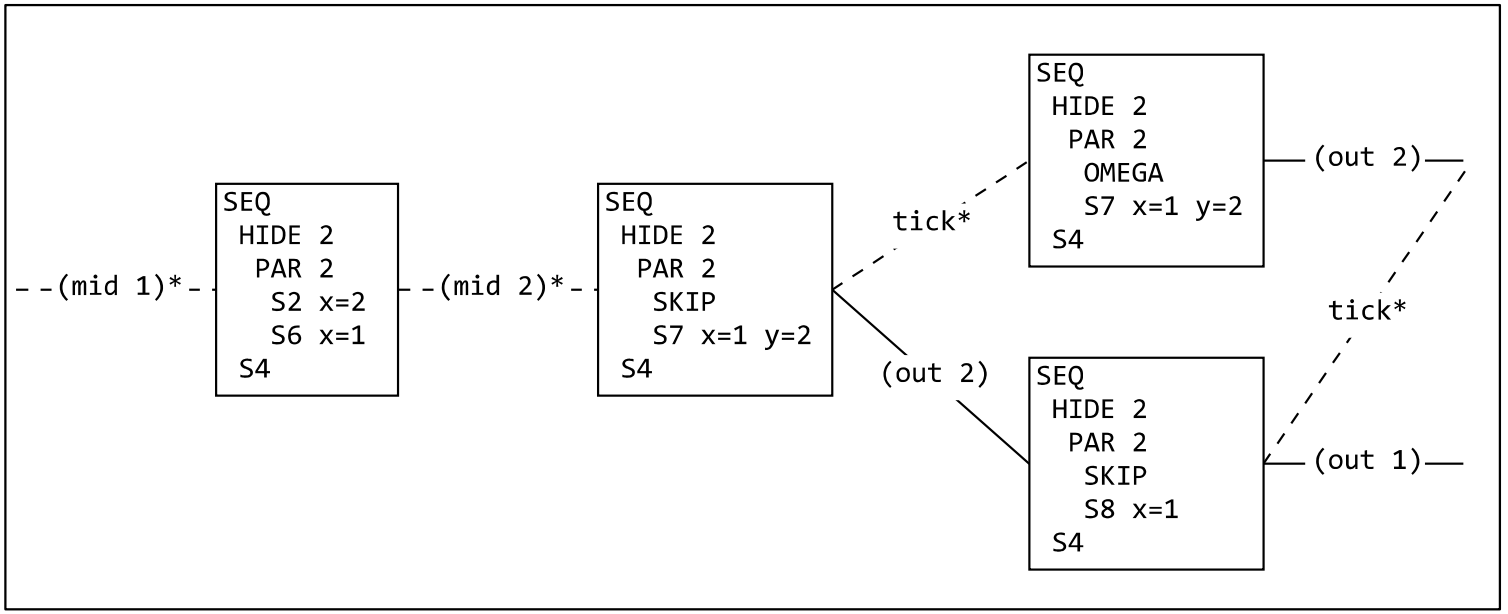


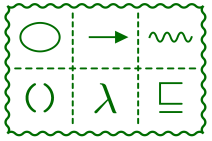
Propositions on state variables?



Propositions on state variables?







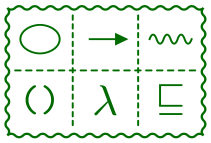
Model Checking with Temporal Logics

- Two types of propositions:
  - Observants
    - enabled(α) `:`α `(enabled` α`)`
    - executed(α) `@`α `(executed` α`)`
    - executed-just(α) `^`α `(executed-just` α`)`
  - Fluents
    - Defined with initiating set I_F and terminaring set T_F

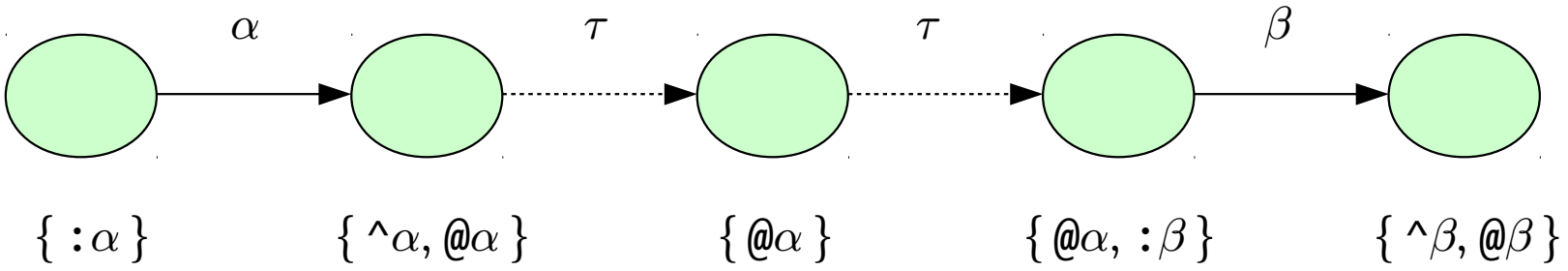
      ```
      (fluent p (chset lock) (chset unlock abort) false)
      ```

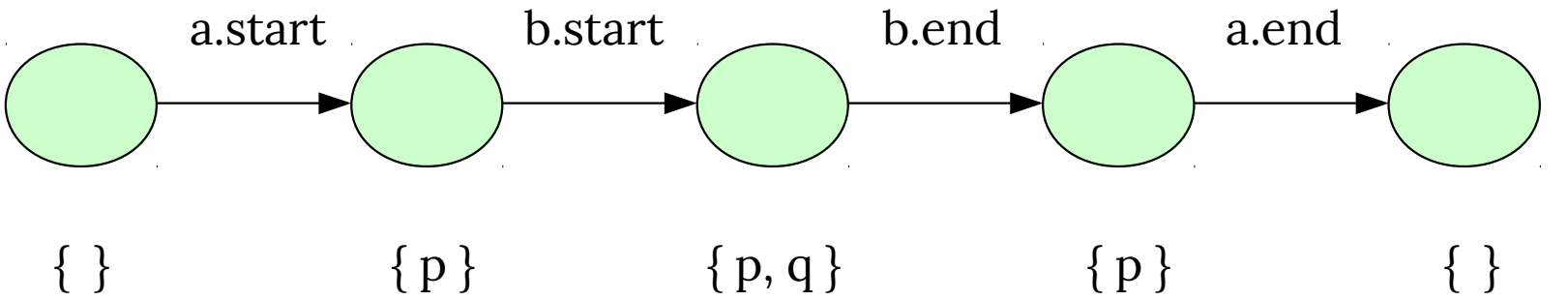
    - Can be indexed

      ```
      (fluent (p (i I)) (chset (a i)) (chset (b i)) true)
      ```

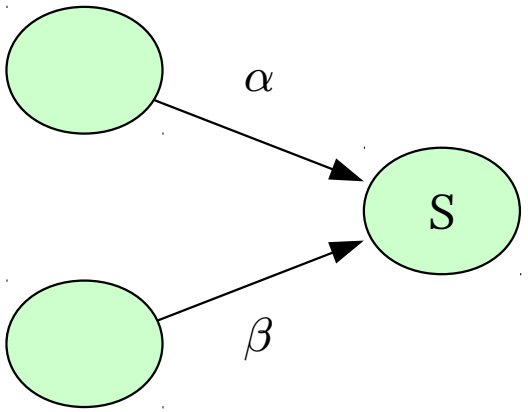


Observants and Fluents

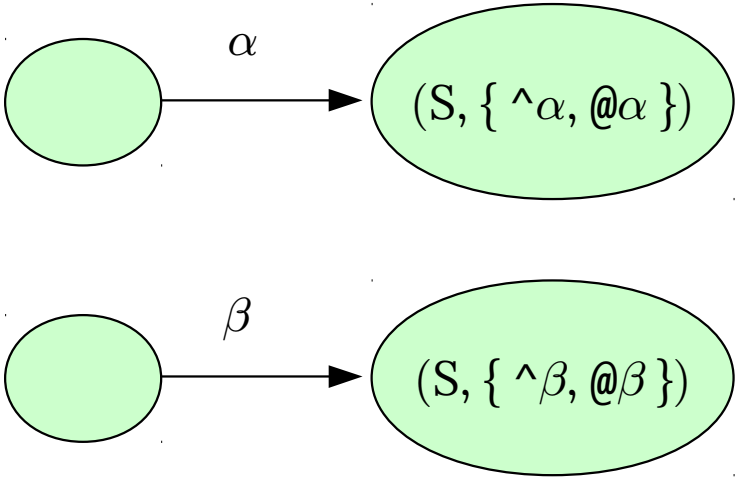




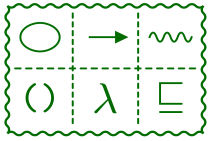
LTS × Fluents



LTS

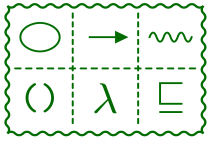


LTS × Fluents



Model Checking with Temporal Logics

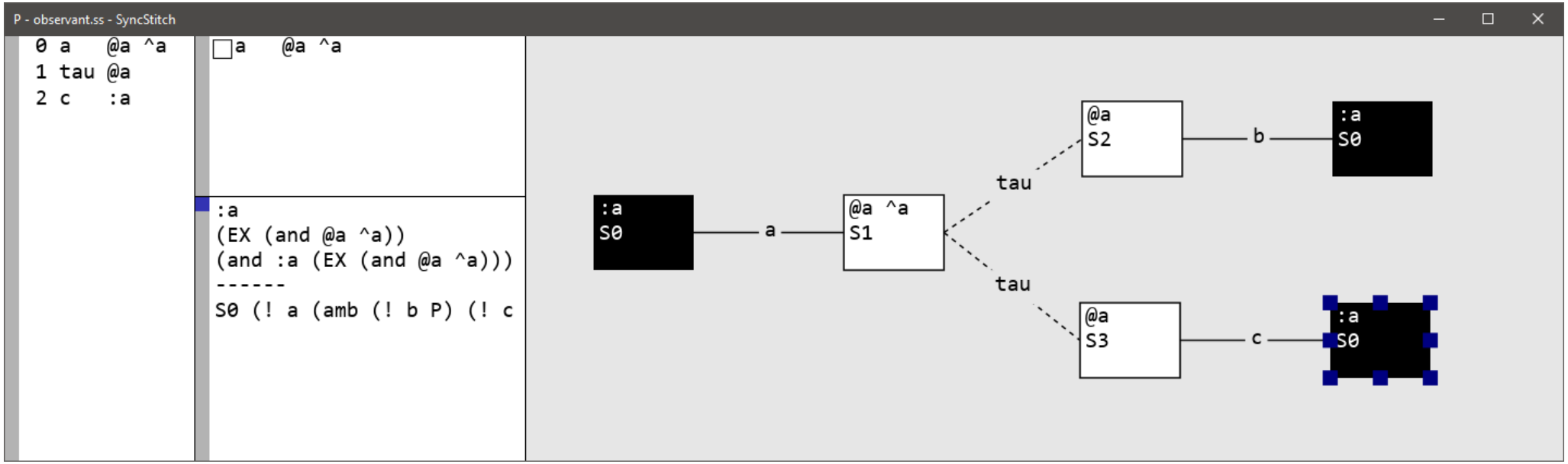
- Linear Temporal Logic (LTL)
  - X G F U W R M
- Computation Tree Logic (CTL)
  - EX EG EF EU AX AG AF AU
  - Fairness assumptions
    - Unconditional `(unconditional` ϕ`)`
      Indexed `(xunconditional` i I ϕ_i`)`
    - Strong `(strong` ϕ ψ`)`
      Indexed `(xstrong` i I ϕ_i ψ_i`)`
    - Weak `(weak` ϕ ψ`)`
      Indexed `(xweak` i I ϕ_i ψ_i`)`

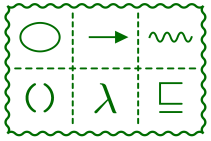


Observants

```
(def P
  (! a
     (amb
       (! b P)
       (! c P))))

(check (CTL P (and :a (EX (and @a ^a)))))
```





Fluents

```
(fluent (p (i I)) (set (lock i)) (set (unlock i)))
```

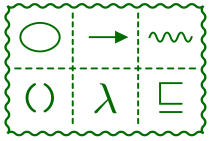
```
(def N 4)
(deftypename I (int 0 N))

(defch lock I)
(defch unlock I)

(def MUTEX
  (? lock (i) (! (unlock i) MUTEX)))

(def (P (i I))
  (! (lock i) (unlock i) (P i)))

(def S
  (par (chset lock unlock)
    (xpar i I (set) (P i))
    MUTEX))
```



CTL model checking with fluents

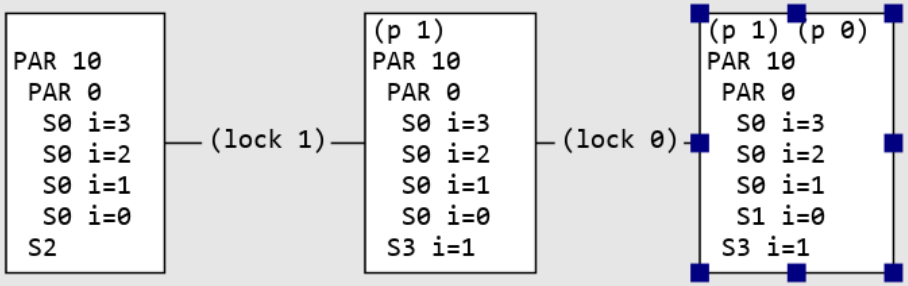
```
(check (CTL S (EF (and (p 0) (p 1)))))
```

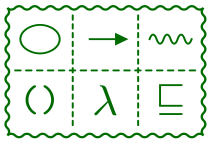
S - mutex-fluent.ss - SyncStitch

```
0 (lock 1)    (p 1)
1 (lock 0)    (p 1) (p 0)
```

```
☐(lock 1)    (p 1) (p 0)
☐(lock 2)    (p 1) (p 0)
☐(lock 3)    (p 1) (p 0)
```

```
(p 1)
(p 0)
(and (p 0) (p 1))
(EU true (and (p 0) (p 1)))
------
PAR 10
 PAR 0
  S0 (! (lock i) (! (unlock
   i=3
  S0 (! (lock i) (! (unlock
   i=2
  S0 (! (lock i) (! (unlock
   i=1
  S1 (! (unlock i) (P i))
   i=0
 S3 (! (unlock i) MUTEX)
  i=1
```





Quantification

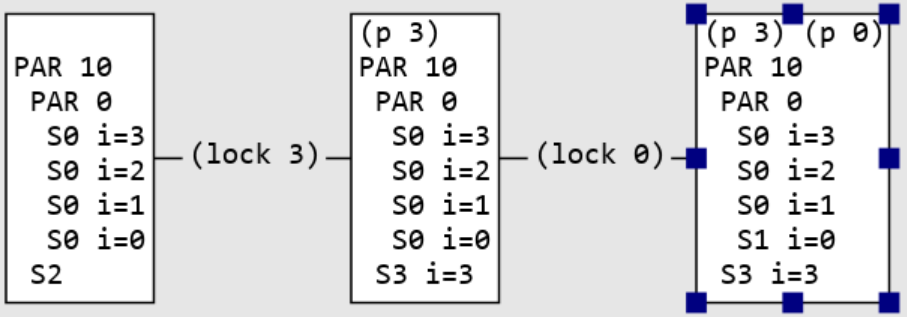
```
(check (CTL S
            (EF (exists i I
                  (exists j (remove I i)
                    (and (p i) (p j)))))))
```

S - mutex-fluent.ss - SyncStitch

```
0 (lock 3)    (p 3)
1 (lock 0)    (p 3) (p 0)
```

```
(lock 1)    (p 3) (p 1) (p
(lock 2)    (p 3) (p 2) (p
(lock 3)    (p 3) (p 0)
```

```
(p 3)
(p 0)
(and (p 0) (p 3))
(or (and (p 0) (p 1)) (and (
(or (and (p 0) (p 1)) (and (
(or (and (p 0) (p 1)) (and (
(and (p 3) (p 0))
(or (and (p 3) (p 0)) (and (
(or (and (p 3) (p 0)) (and (
(or (and (p 0) (p 1)) (and (
(EU true (or (and (p 0) (p 1
------
PAR 10
 PAR 0
  S0 (! (lock i) (! (unlock
   i=3
  S0 (! (lock i) (! (unlock
   i=2
```

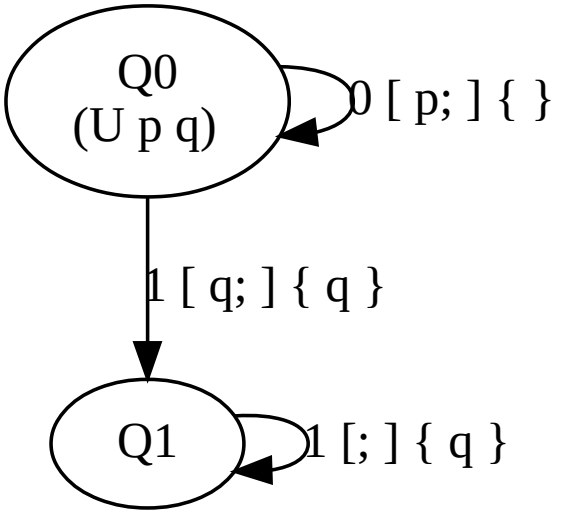


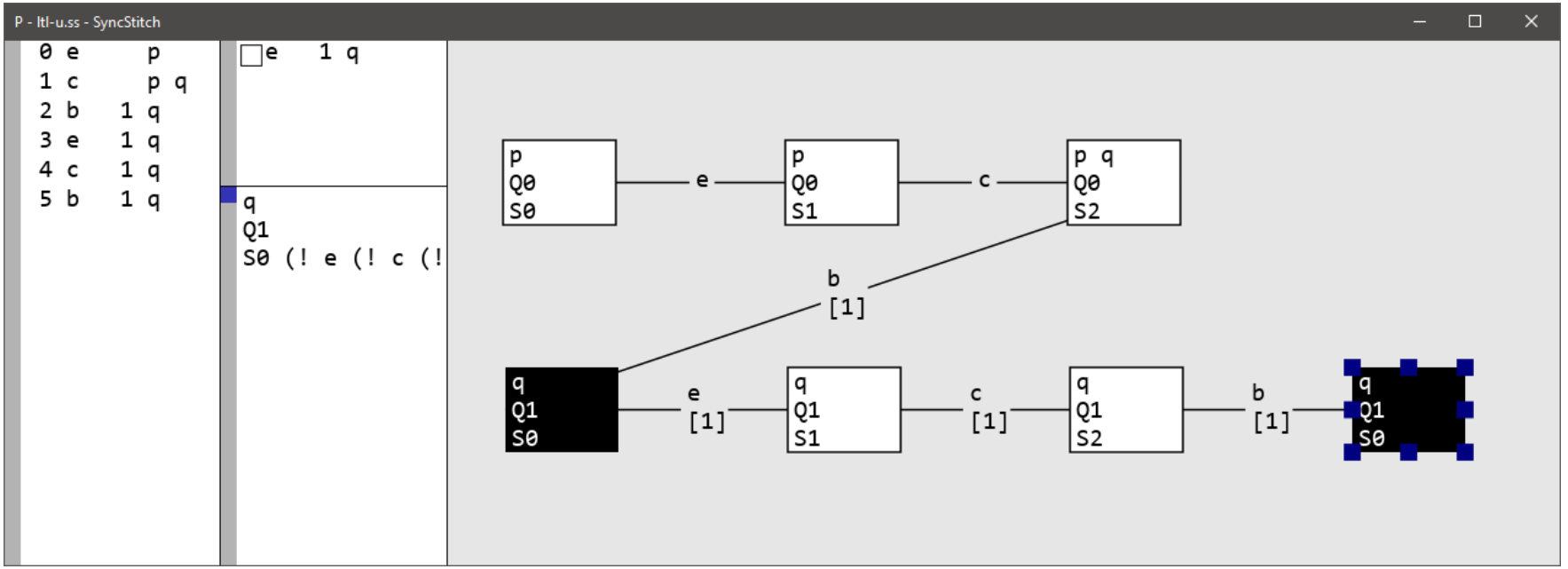
LTL model checking

```
(fluent p (chset a) (chset b) true)
(fluent q (chset c) (chset d) false)

(def P (! e c b P))

(check (LTL P (U p q)))
```



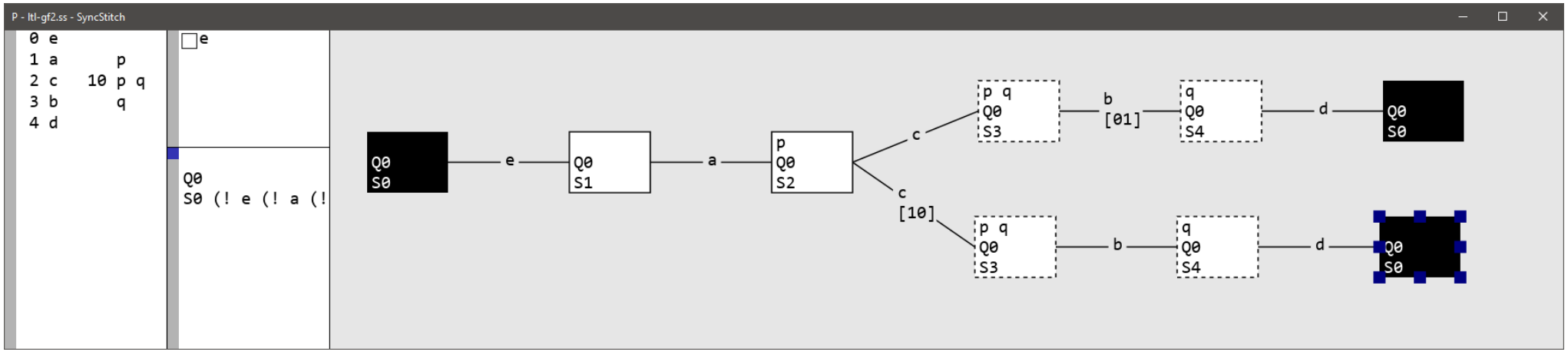


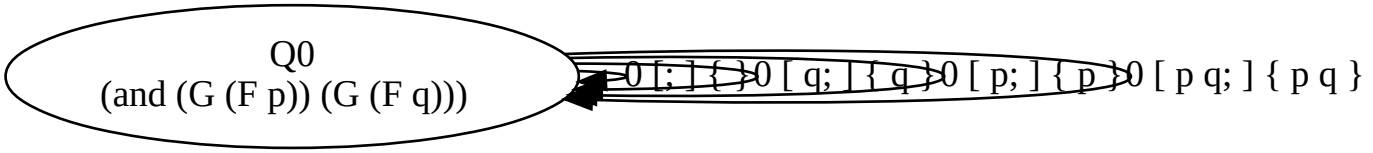
Acceptance cycles of Büchi automaton

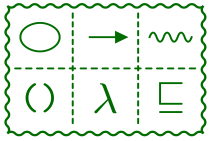
```
(fluent p (chset a) (chset b) false)
(fluent q (chset c) (chset d) false)

(def P (! e a c b d P))

(check (LTL P (and (G (F p)) (G (F q)))))
```







Witness (counterexample) report

```
LTL acceptance: P (and (G (F p)) (G (F q)))
---  initial path  ----------------
  0 tau 00 {     } Q0 S0 : (! e (! a (! c (! b (! d P)))))
---  acceptance paths : 2  ----------------
[0. accept marks: 10]
  0 tau 00 {     } Q0 S0 : (! e (! a (! c (! b (! d P)))))
  1 e   00 {     } Q0 S1 : (! a (! c (! b (! d P))))
  2 a   00 { p   } Q0 S2 : (! c (! b (! d P)))
  3 c   10 { p q } Q0 S3 : (! b (! d P))
  4 b   00 {   q } Q0 S4 : (! d P)
  5 d   00 {     } Q0 S0 : (! e (! a (! c (! b (! d P)))))
[1. accept marks: 01]
  0 tau 00 {     } Q0 S0 : (! e (! a (! c (! b (! d P)))))
  1 e   00 {     } Q0 S1 : (! a (! c (! b (! d P))))
  2 a   00 { p   } Q0 S2 : (! c (! b (! d P)))
  3 c   00 { p q } Q0 S3 : (! b (! d P))
  4 b   01 {   q } Q0 S4 : (! d P)
  5 d   00 {     } Q0 S0 : (! e (! a (! c (! b (! d P)))))
```