

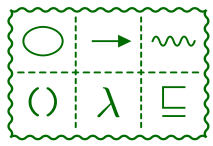
SyncStitch 3

Hisabumi HATSUGAI

hatsugai@principia-m.com

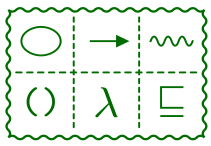
PRINCIPIA Limited

<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

S - mutex-fair-fluent.ss - SyncStitch

0 (lock 1)	111
1 tau	111
2 (ret 1)	111
3 (a 1)	111
4 (b 1)	111
5 (unlock 1)	111
6 (lock 1)	111

tau 111

(lock 2) 111

(lock 3) 111

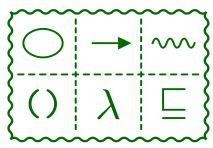
```

graph TD
    Q3_1["Q3  
PAR 18  
PAR 0  
S0 i=3  
S0 i=2  
S0 i=1  
S0 i=0  
S5 m=Unlocked s=(Set)"]
    Q3_2["Q3  
PAR 18  
PAR 0"]
    Q3_3["Q3  
PAR 18  
PAR 0  
S0 i=3  
S0 i=2  
S2 i=1  
S0 i=0  
S5 m=(Locked 1) s=(Set)"]
    Q3_4["Q3  
PAR 18  
PAR 0  
S0 i=3  
S0 i=2  
S1 i=1  
S0 i=0  
ALT  
S6 m=Unlocked s=(Set 1)  
S7 m=Unlocked s=(Set 1)  
S9 s=(Set 1) i=1"]
    Q3_5["Q3  
PAR 18  
PAR 0  
S0 i=3  
S0 i=2  
S1 i=1  
S0 i=0  
S5 m=Unlocked s=(Set 1)"]
    Q3_6["Q3  
PAR 18  
PAR 0"]

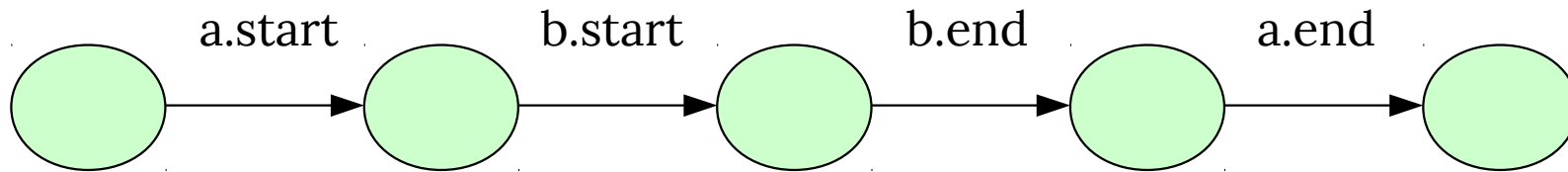
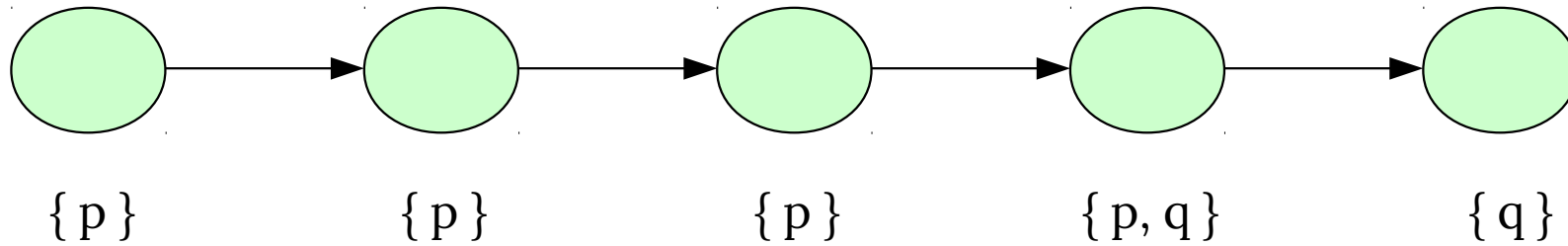
    Q3_1 -- "(lock 0) [101]" --> Q3_2
    Q3_1 -- "(lock 1) [101]" --> Q3_2
    Q3_1 -- "(lock 1) [111]" --> Q3_3
    Q3_3 -- "(a 1) [111]" --> Q3_4
    Q3_3 -- "(b 1) [111]" --> Q3_5
    Q3_5 -- "(unlock 1) [111]" --> Q3_6
    Q3_6 -- "(lock 1) [111]" --> Q3_5
    Q3_4 -- "(ret 1) [111]" --> Q3_1
    Q3_4 -.- "tau [111]" -.- Q3_3
  
```

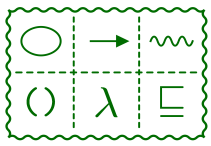
```

Q3
PAR 18
PAR 0
S0 (! (lock i) (! (ret
i=3
S0 (! (lock i) (! (ret
i=2
S3 (! (b i) (! (unlock
i=1
S0 (! (lock i) (! (ret
i=0
S5 (alt (? lock (i) tru
m=(Locked 1)
s=(Set)
  
```

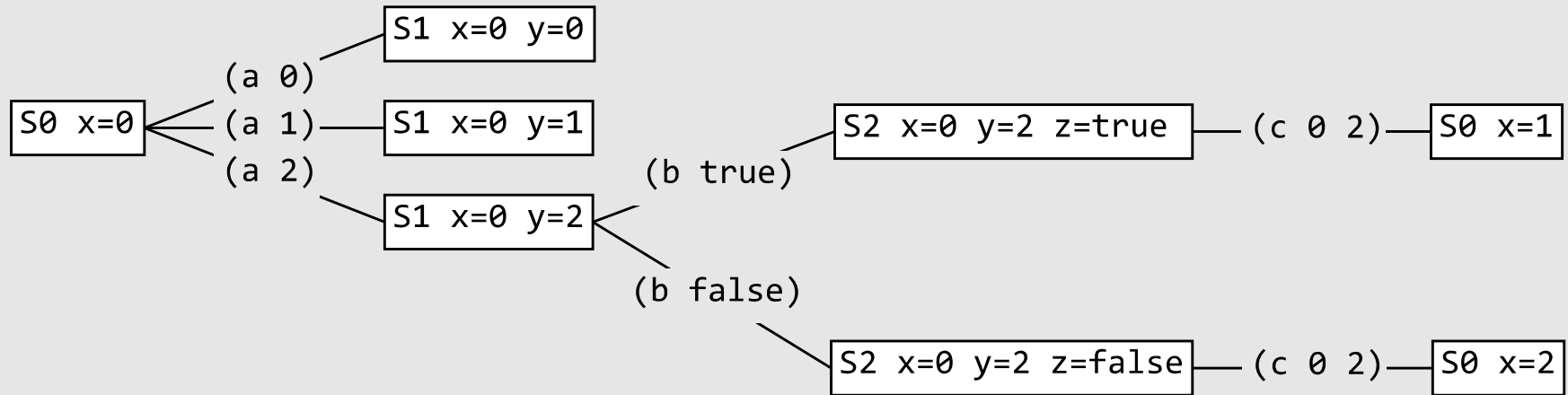


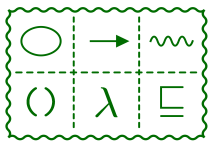
State vs Action (transition, event)



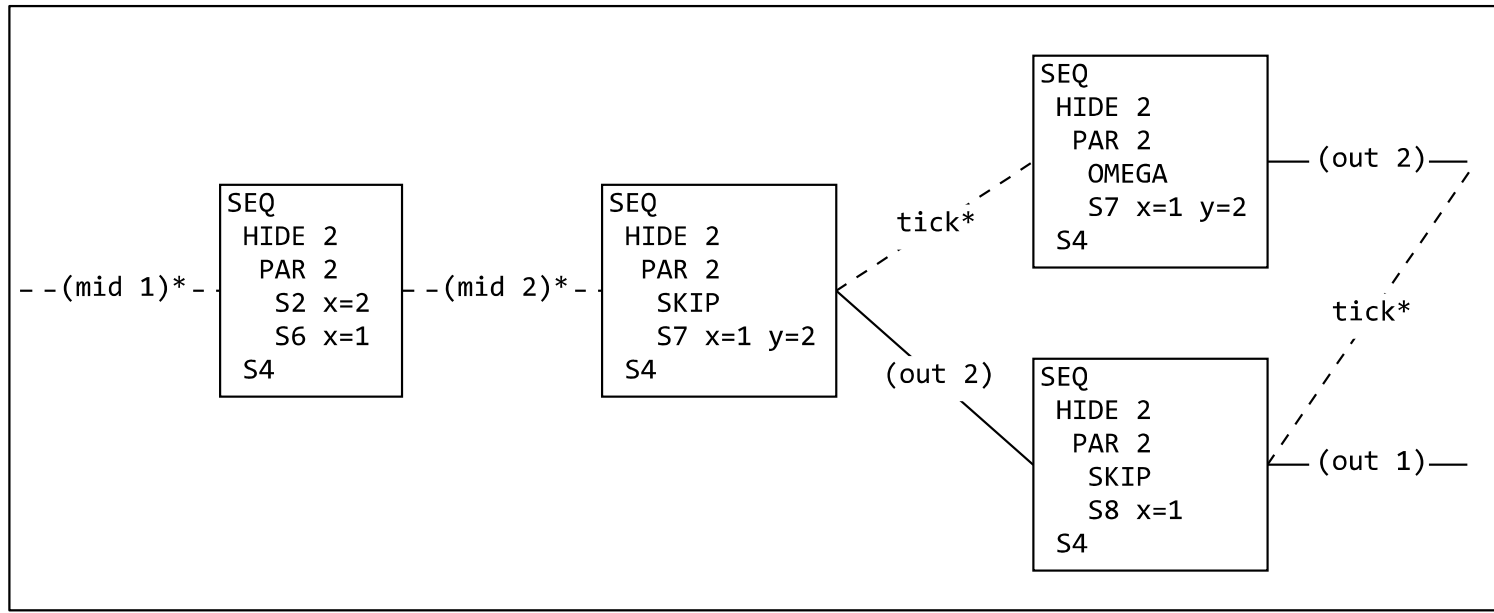
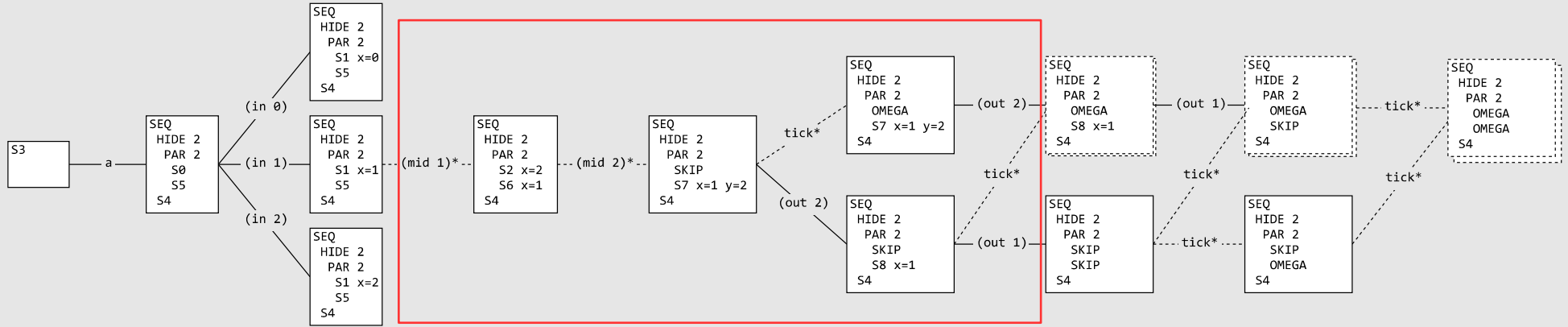


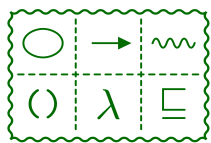
Propositions on state variables?





Propositions on state variables?





Model Checking with Temporal Logics

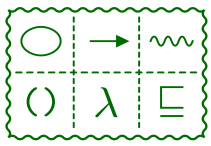
- Two types of propositions:

- Observants

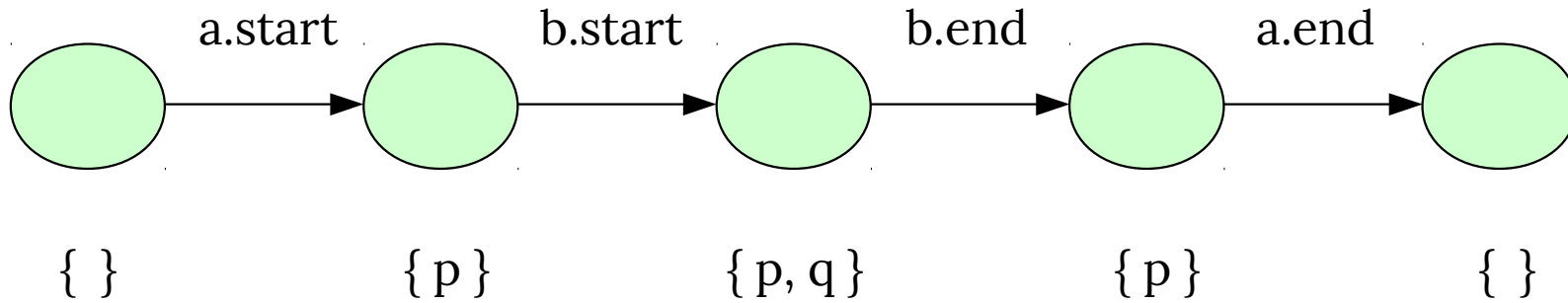
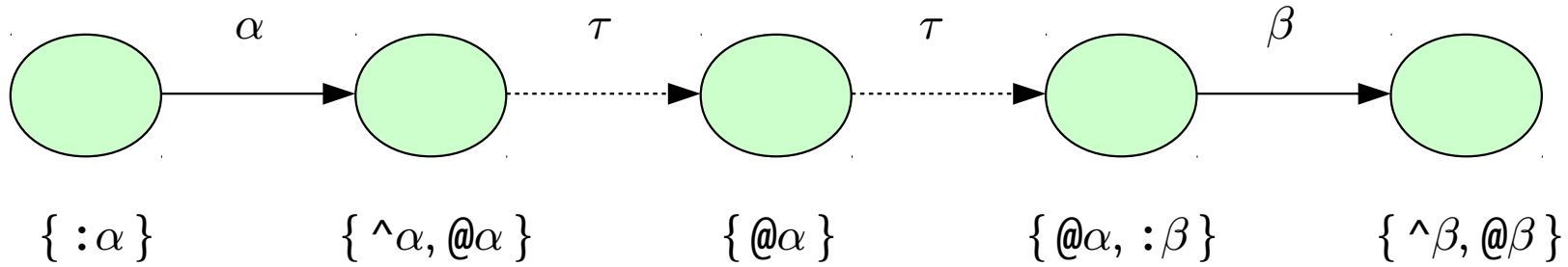
- `enabled(α)` `: α` `(enabled α)`
- `executed(α)` `@ α` `(executed α)`
- `executed-just(α)` `^ α` `(executed-just α)`

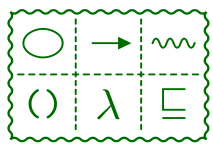
- Fluents

- 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)`

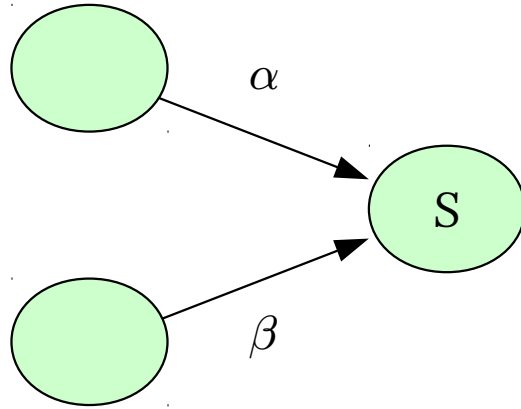


Observants and Fluents

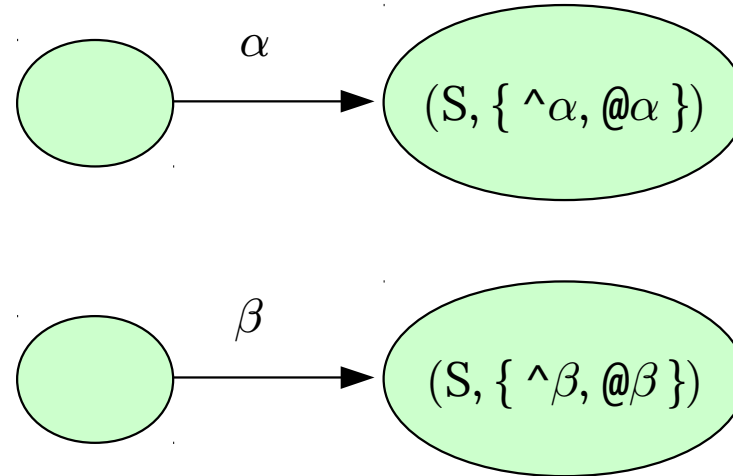




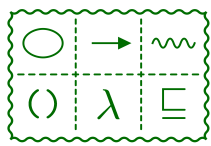
LTS \times Fluents



LTS

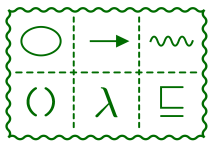


LTS \times Fluents



Model Checking with Temporal Logics

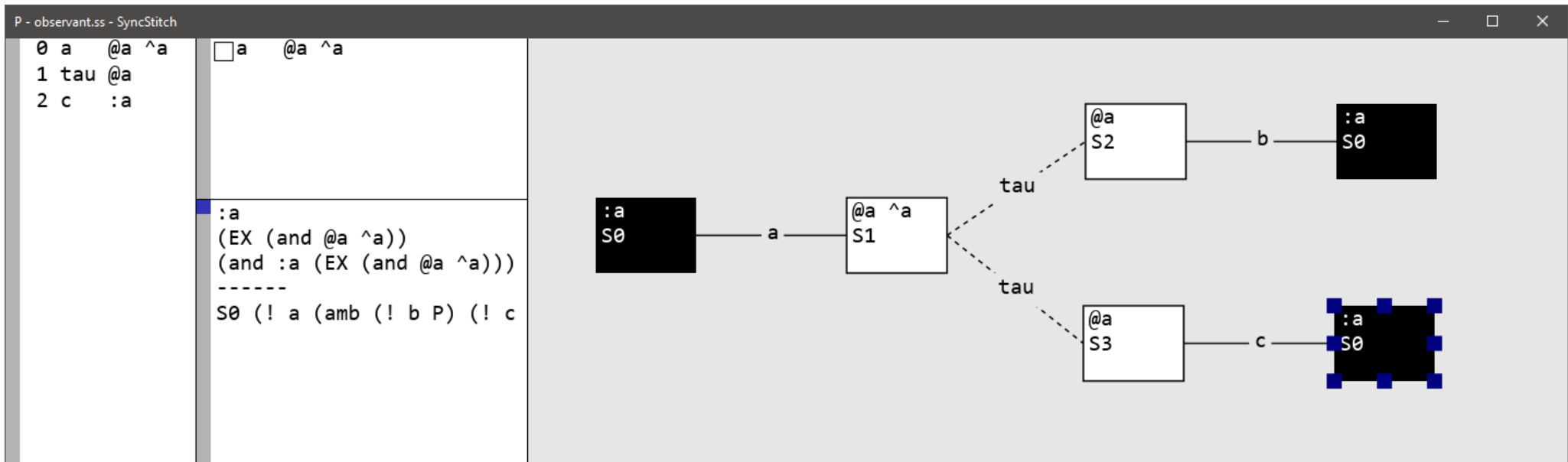
- Linear Temporal Logic (LTL)
 - X G F U W R M
- Computation Tree Logic (CTL)
 - EX EG EF EU AX AG AF AU
 - Fairness assumptions
 - Unconditional (unconditional ϕ)
 - Indexed (xunconditional $i I \phi_i$)
 - Strong (strong $\phi \psi$)
 - Indexed (xstrong $i I \phi_i \psi_i$)
 - Weak (weak $\phi \psi$)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

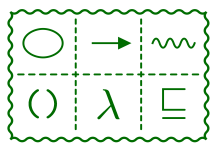


Observants

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

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





Fluent

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

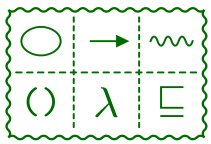
```
(def N 4)
(deftypename I (int 0 N))

(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

```

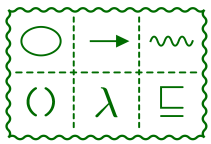
```

State 1:
PAR 10
PAR 0
S0 i=3
S0 i=2
S0 i=1
S0 i=0
S2

State 2:
(p 1)
PAR 10
PAR 0
S0 i=3
S0 i=2
S0 i=1
S0 i=0
S3 i=1

State 3:
(p 1) (p 0)
PAR 10
PAR 0
S0 i=3
S0 i=2
S0 i=1
S1 i=0
S3 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 0)
 (lock 2) (p 3) (p 2) (p 0)
 (lock 3) (p 3) (p 0)

```

```

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

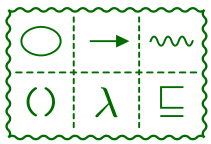
```

The diagram illustrates the state of three processes (p 0, p 1, p 3) in a parallel execution:

- Process p 3 (left):** PAR 10, PAR 0, S0 i=3, S0 i=2, S0 i=1, S0 i=0, S2.
- Process p 3 (middle):** (p 3), PAR 10, PAR 0, S0 i=3, S0 i=2, S0 i=1, S0 i=0, S3 i=3.
- Process p 0 (right):** (p 3) (p 0), PAR 10, PAR 0, S0 i=3, S0 i=2, S0 i=1, S1 i=0, S3 i=3.

Connections between processes:

- Process p 3 (left) is connected to Process p 3 (middle) via edge (lock 3).
- Process p 3 (middle) is connected to Process p 0 (right) via edge (lock 0).

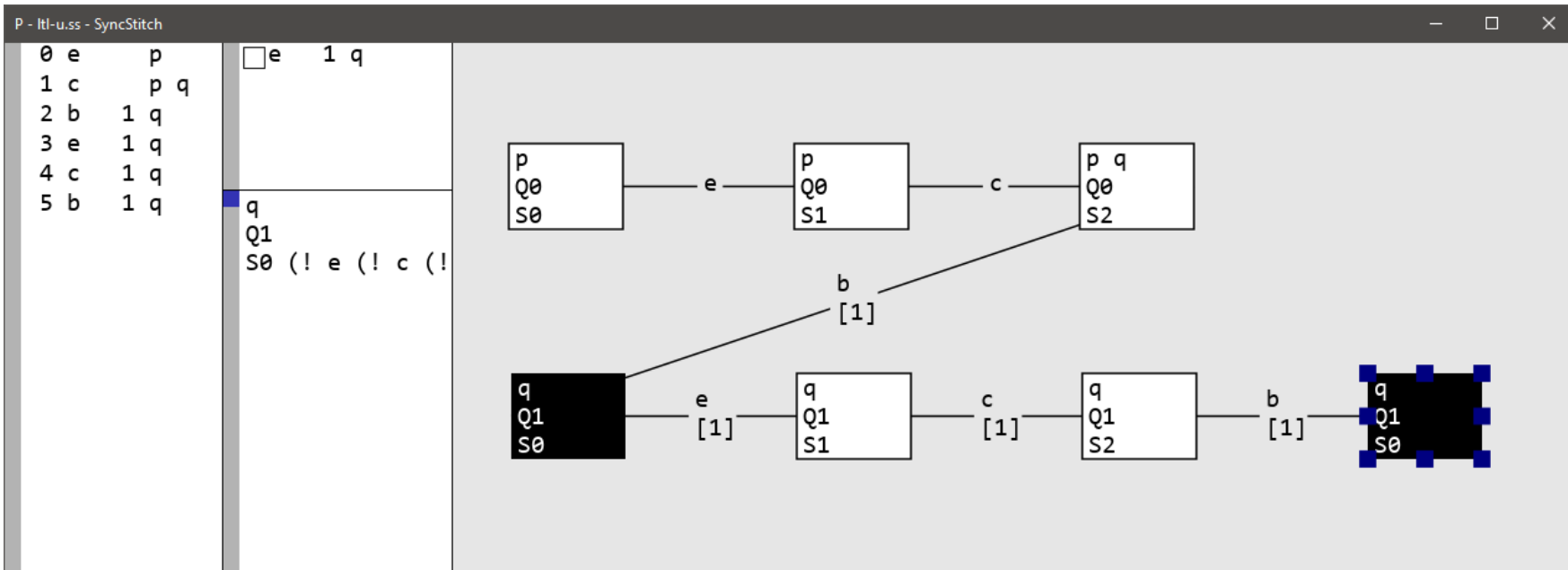
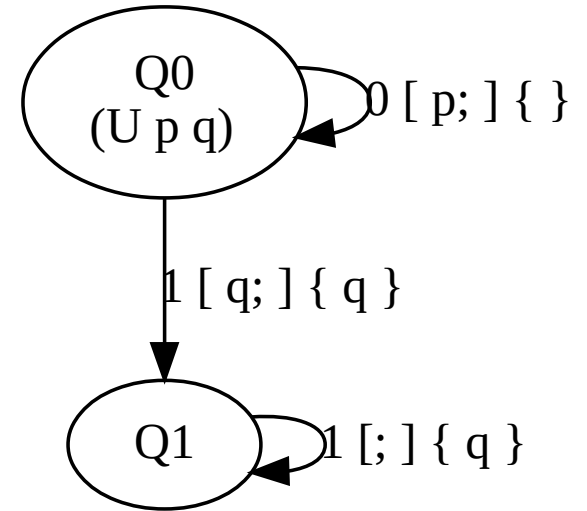


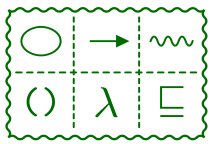
LTL model checking

```
(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)))
```



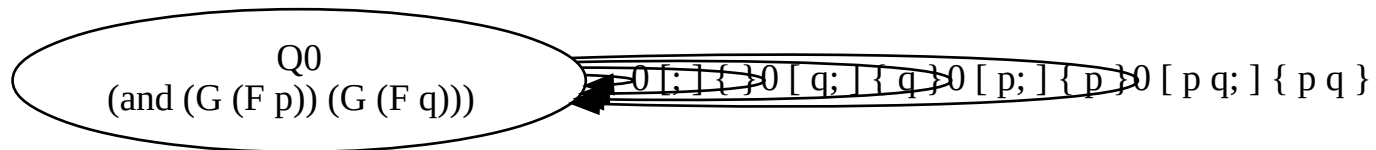
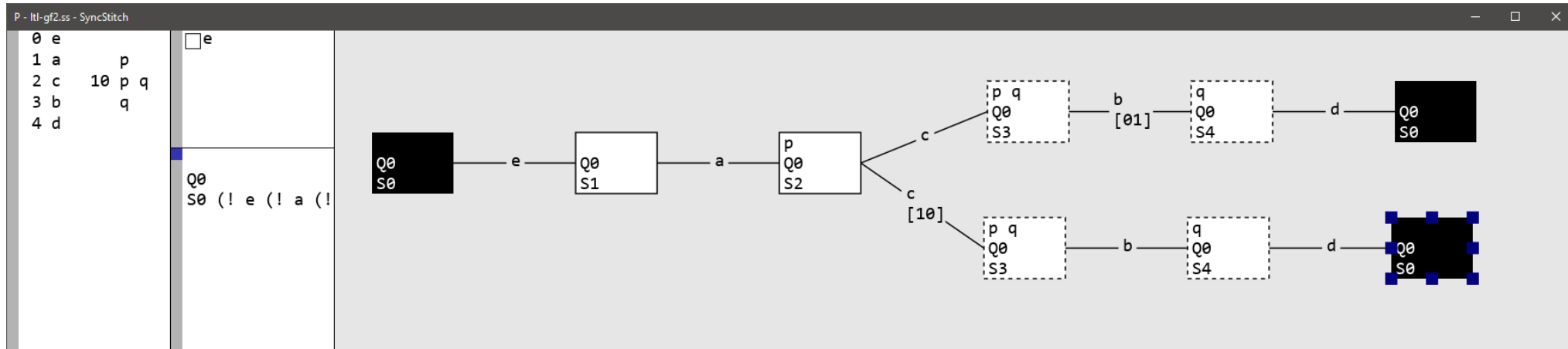


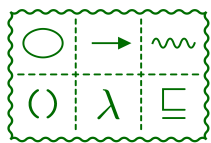
Acceptance cycles of Büchi automaton

```
(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))))))
```