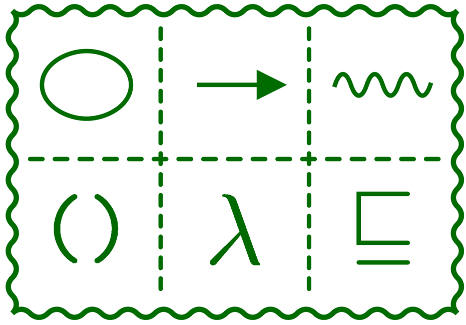
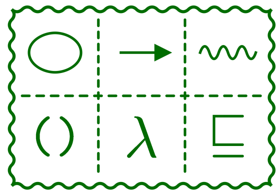


並行システムの検証と実装

第6章 プロセスの動的生成と終了

PRINCIPIA Limited
初谷 久史

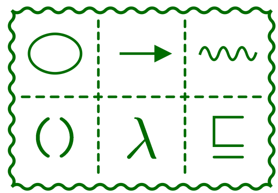
©2015 PRINCIPIA Limited



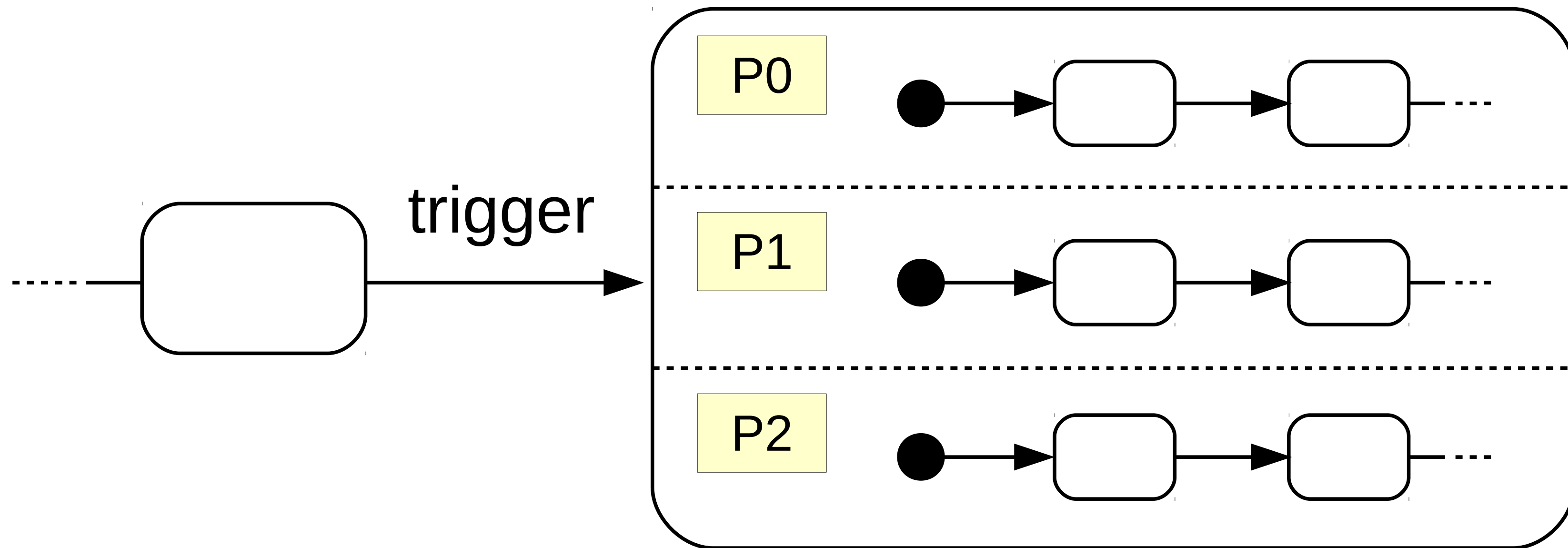
プロセスの動的生成と終了

- 並行合成 par によるプロセスの動的生成
- 並行プロセスの終了
- 親プロセスの継続
- 従属プロセスの終了

©2015 PRINCIPIA Limited

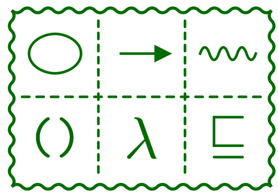


並行合成 par によるプロセスの動的生成



パターン

```
(! trigger
  (par sync-list
    P0
    P1
    P2))
```



動的生成の例

```
(define-process P
  (! trigger
    (par (list mid) P0 P1)))
```

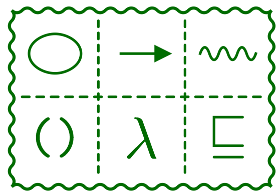
P0 と P1 を動的に生成する

```
(define-process P0
  (? in (x) (! mid (x) P0)))
```

```
(define-process P1
  (? mid (x) (! out (x) P1)))
```

チャネル通信の例

©2015 PRINCIPIA Limited



動的生成の計算木

example-6-0-dyn-par.ss

Process Explorer - P

Trans | Path | Tree | Expr | Env | Hide all

P —trigger→ (par * P0 P1)

(par * P0 P1) —in.0→ (par * (! mi..

(par * P0 P1) —in.1→ (par * (! mi..

(par * P0 P1) —in.2→ (par * (! mi..

(par * (! mi.. —mid.0→ (par * P0 (! ..

(par * (! mi.. —mid.1→ (par * P0 (! ..

(par * (! mi.. —mid.2→ (par * P0 (! ..

(par * P0 (! .. —in.0→ (par * (! mi..

(par * P0 (! .. —in.1→ (par * (! mi..

(par * P0 (! .. —in.2→ (par * (! mi..

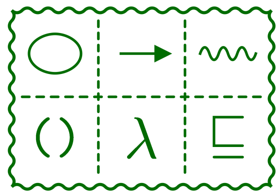
(par * P0 (! .. —out.0→ (par * P0 P1)

Process Tree

(! trigger ..

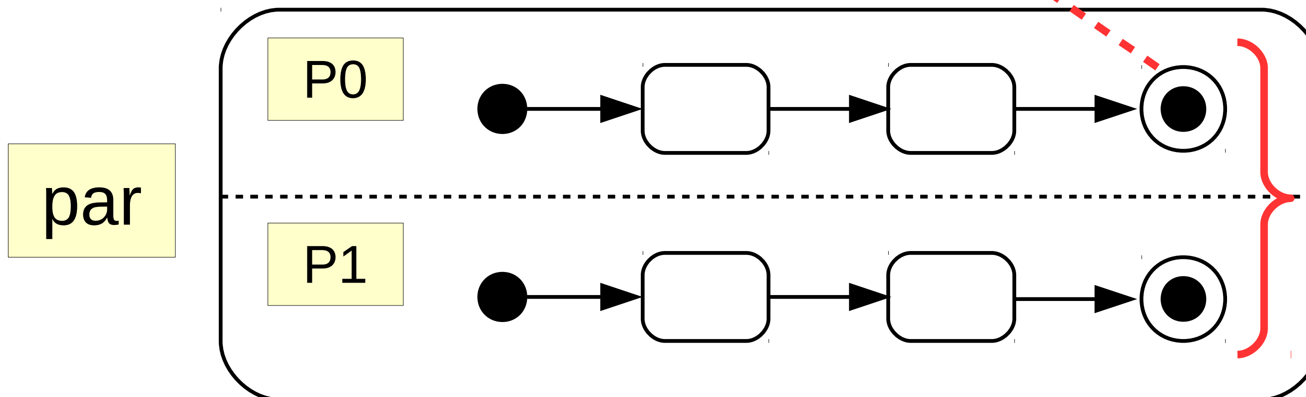
Process Tree

par
- P0
- P1



並行プロセスの終了

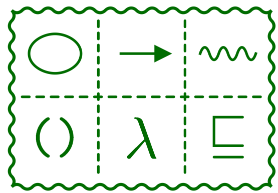
並行プロセスはそれぞれ独立に終了できる



合成系を外から観た場合
終了は1度だけ

すべてのサブプロセスが終了すると
合成系が終了する

©2015 PRINCIPIA Limited



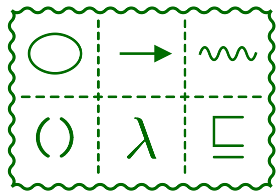
並行プロセスの終了: 例

```
(define-process P
  (! c
    (par '() P0 P1)))
```

```
(define-process P0
  (! a SKIP))
```

```
(define-process P1
  (! b SKIP))
```

©2015 PRINCIPIA Limited



並行プロセスの終了: 計算木

`example-6-1-dist-term.ss`

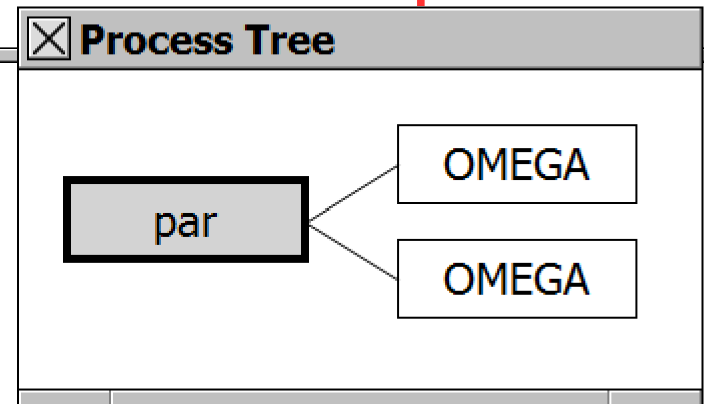
P0, P1 それぞれ独立に終了

すべてのサブプロセスが終了すると合成系が終了する．このとき1度だけ tick が発生する

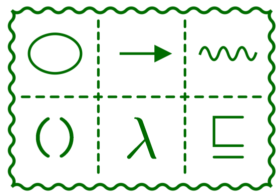
Process Explorer - P

Trans | Path | Tree | Expr | Env | Hide all

P
c
(par * P0 P1)
a
b
(par * SKIP P1)
(par * P0 SKIP)
(par * OMEGA P1)
(par * SKIP SKIP)
(par * P0 OMEGA)
(par * OMEGA SKIP)
(par * SKIP OMEGA)
(par * OMEGA OMEGA)
tick
OMEGA



par 内部の終了イベント tick は内部イベント tau に変換される（seq 同様）

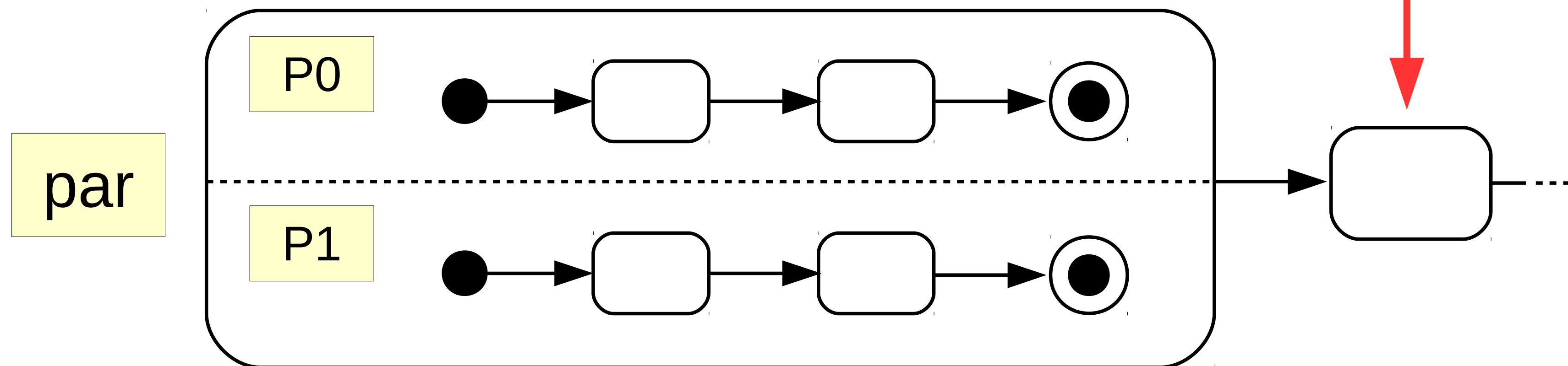


親プロセスの継続

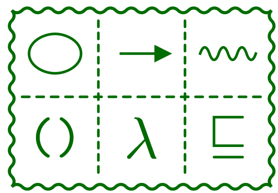
par が終了するとプロセス P も終了する

```
(define-process P
  (! trigger
    (par (list mid) P0 P1)))
```

par 終了後もプロセス P を継続したい



©2015 PRINCIPIA Limited



親プロセスの継続

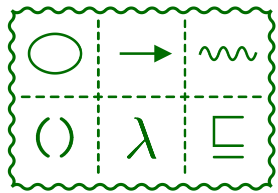
seq を使うと par 終了後も処理を続けられる

```
(define-process P
  (! trigger
    (seq
      (par sync-list P0 P1)
      継続処理)))
```

例

```
(define-process P
  (! c
    (seq
      (par '() P0 P1)
      (! d SKIP))))
```

©2015 PRINCIPIA Limited



親プロセスの継続: 例

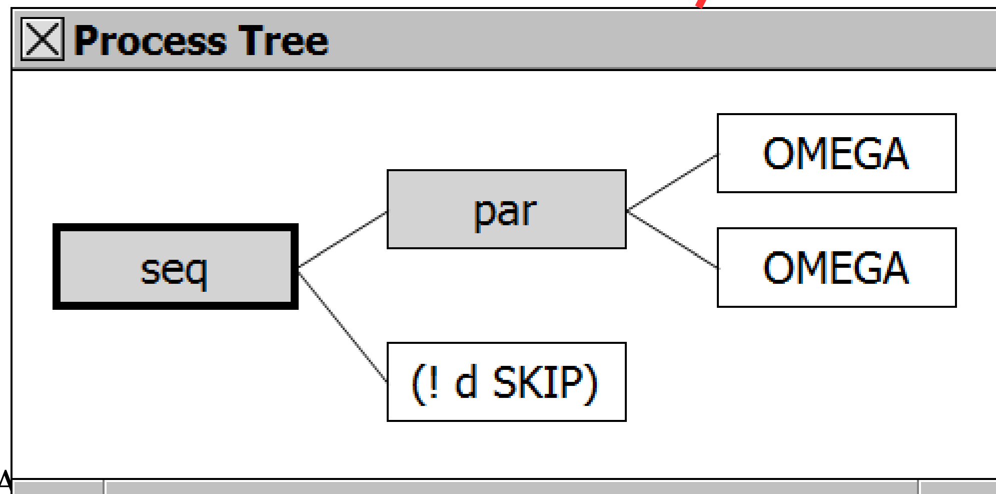
example-6-2-dist-term-cont.ss

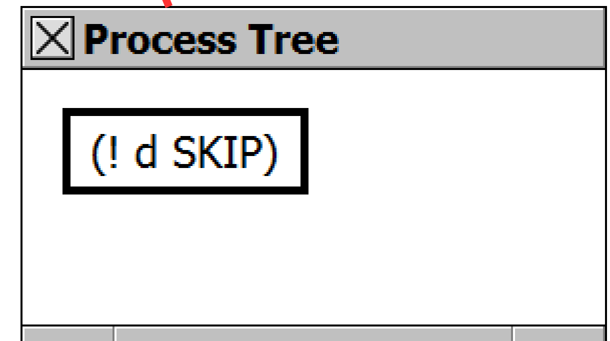
Process Explorer - P

Trans Path Tree Expr Env Hide all

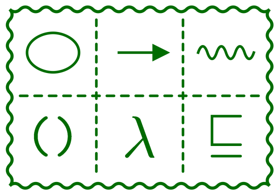
P c (seq.. a (seq.. b (seq.. b (seq.. (seq.. (! d .. d SKIP tick OMEGA

b (seq.. a (seq.. a (seq.. (seq..





©2015 PRINCIPIA



従属プロセスの終了

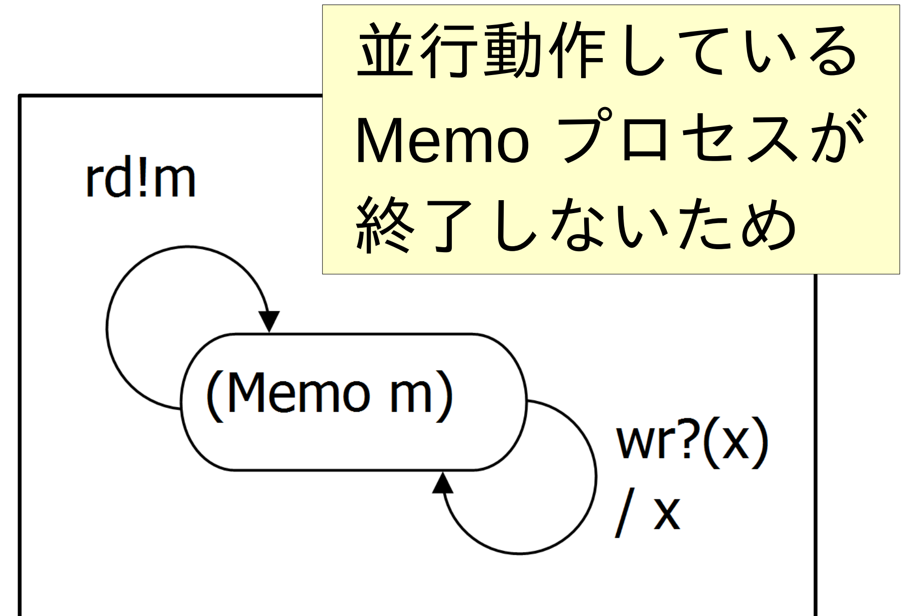
再掲: メモプロセスの使用例

```
(define-process Main
 (seq
  (Doubler 1)
  (? rd (x)
   (seq
    (Doubler x)
    (? rd (y)
     (! out (y) SKIP)))))))
```

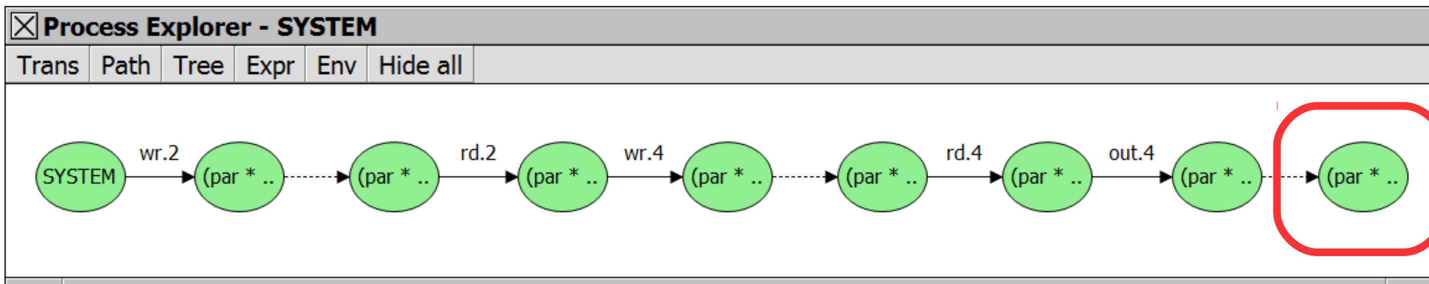
終了を意図している

```
(define-process (Doubler x)
  (! wr ((* x 2)) SKIP))
```

並行動作している Memo プロセスが終了しないため

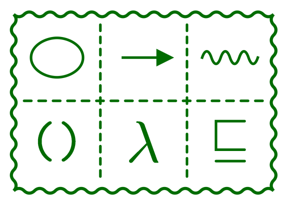



```
(define-process SYSTEM
  (par (list rd wr)
    Main (Memo 0)))
```



終了していない

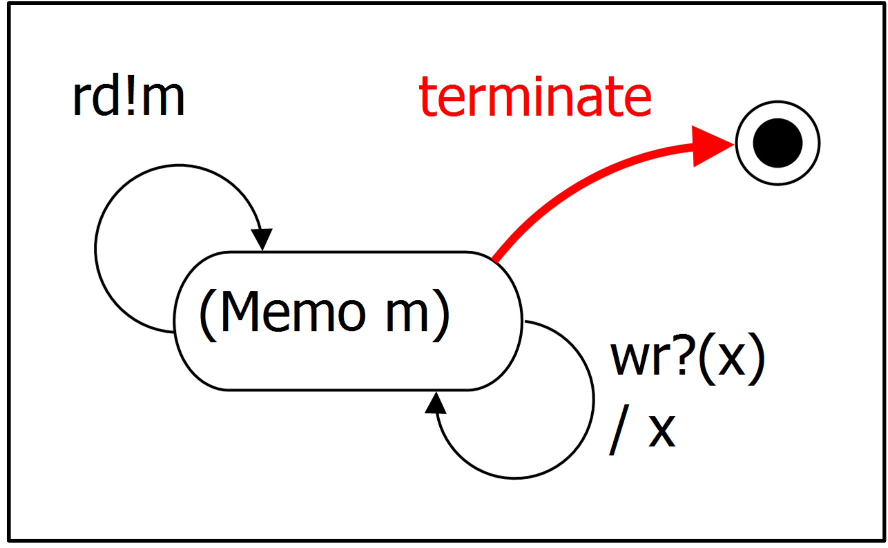
example-6-3-memo-term.ss



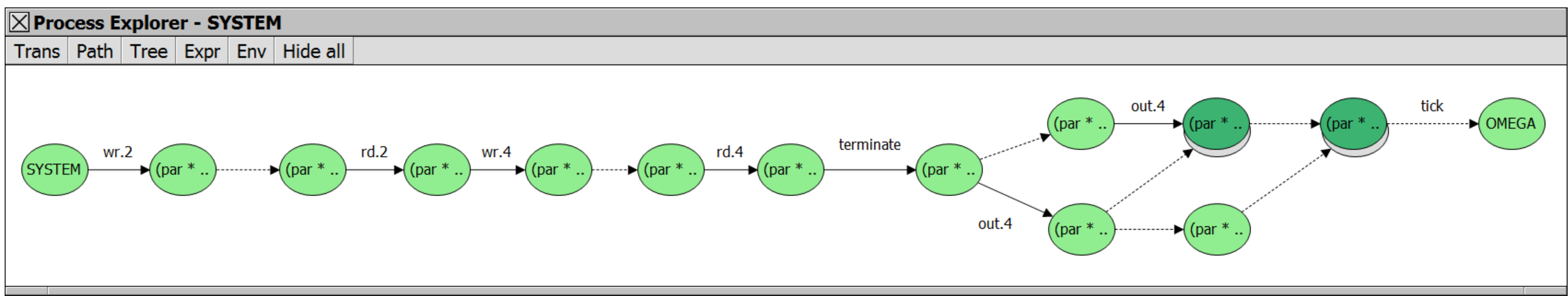
従属プロセスの終了

```
(define-process Main
  (seq
    (Doubler 1)
    (? rd (x)
      (seq
        (Doubler x)
        (? rd (y)
          (! terminate
            (! out (y) SKIP)))))))
```

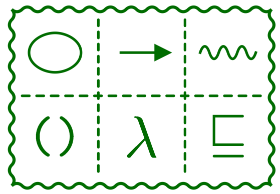
Memo に
終了を指示




```
(define-process SYSTEM
  (par (list rd wr terminate)
    Main (Memo 0)))
```



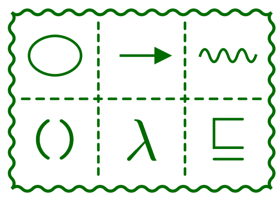
©2015 PRINCIPIA Limited



プロセスの動的生成と終了まとめ

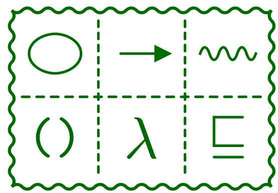
- 並行合成 par によってプロセスを実行時に動的生成することができる
- 並行プロセスの終了
  - 各サブプロセスは独立に終了できる
  - すべてのサブプロセスが終了すると合成系が終了する
  - 合成系の外部から観測したとき各サブプロセスの終了は見えない: 内部イベント tau になる

©2015 PRINCIPIA Limited

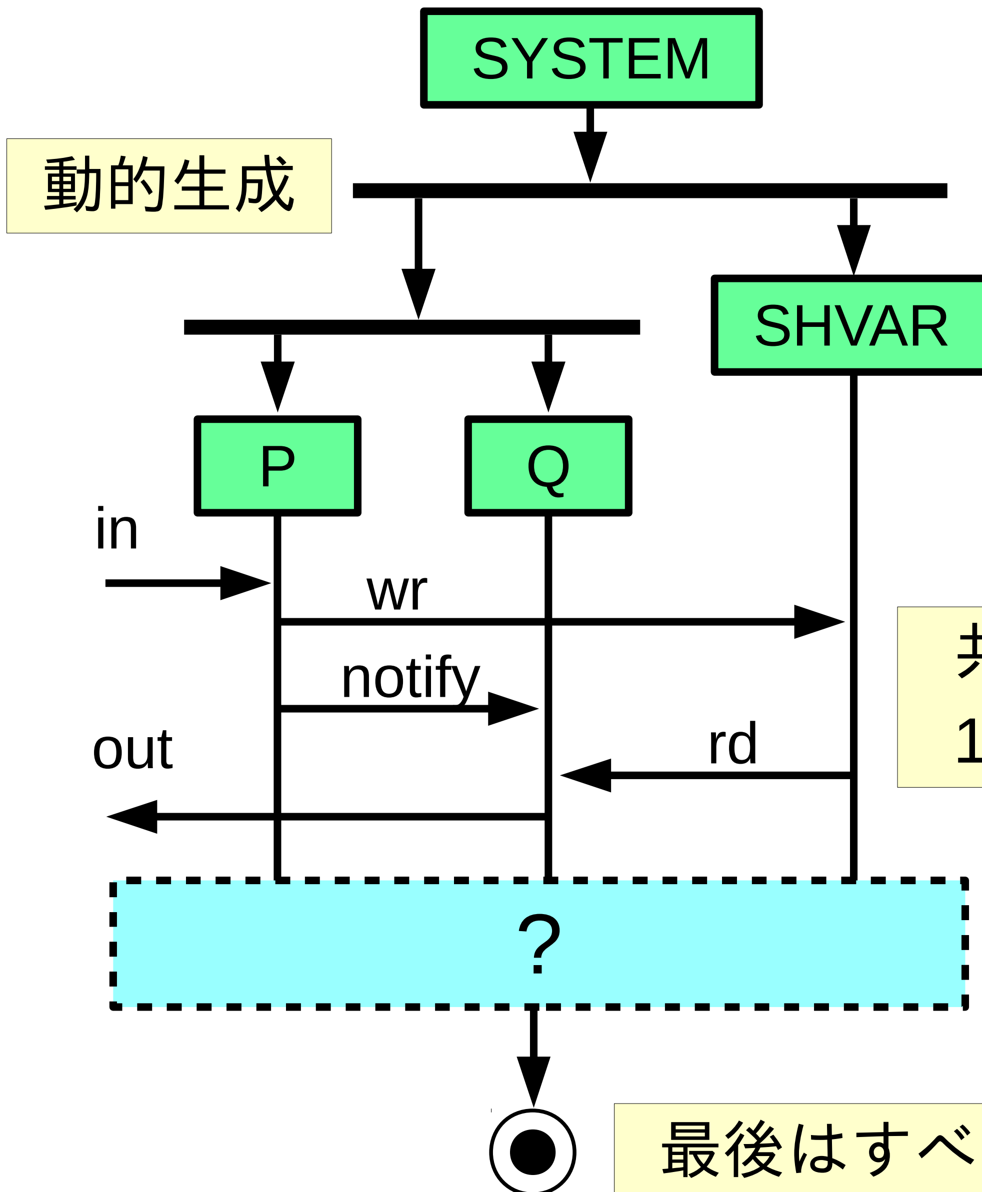


プロセスの動的生成と終了まとめ: テクニック

- 親プロセスの継続
  - プロセスを動的に生成する par を seq に入れると，par 終了後も処理を継続できる
- 従属プロセスの終了
  - 合成系を終了させるために，従属プロセスに終了指示を出す



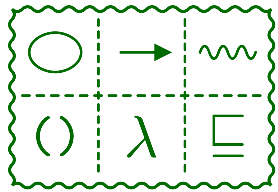
練習問題



左図のようなシーケンスを実行するシステムのモデルを作成してください

共有変数を通じて
1回だけデータ交換

最後はすべて終了



イベント定義

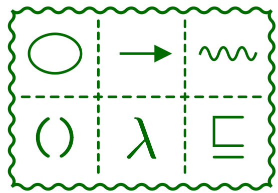
exercise-6-0.ss

```
(define M 2)
(define I (interval 0 M))
(define D (map list I))

(define-channel in (x) D)
(define-channel out (x) D)

(define-channel rd (x) D)
(define-channel wr (x) D)

(define-event notify)
(define-event terminate)
```



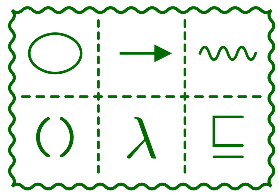
プロセス定義

```
(define-process (SHVAR m)
  (alt
    (! rd (m) (SHVAR m))
    (? wr (x) (SHVAR x))
    (! terminate SKIP)))
```

```
(define-process P
  (? in (x)
    (! wr (x)
        (! notify SKIP))))
```

```
(define-process Q
  (! notify
    (? rd (x)
      (! out (x) SKIP))))
```

©2015 PRINCIPIA Limited



システムプロセス定義

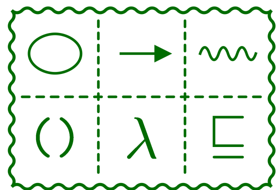
```
(define-process SYSTEM
  (par (list rd wr terminate)
    (SHVAR 0)
    (seq
      (par (list notify) P Q)
      (! terminate SKIP))))
```

終了通知イベント
terminate も同期

P-Q 間は notify で同期

逐次合成 seq により P, Q がともに
終了した後で terminate を発行

©2015 PRINCIPIA Limited



SYSTEM プロセスの計算木

SYSTEM
in.0
in.1
(par * (..
wr.0
(par * (..
notify
(par * (..
rd.0
(par * (..
(par * (..
rd.0
(par * (..
out.0
(par * (..
out.0
(par * (..
(par * (..
(par * (..
(par * (..
terminate
(par * S..
(par * O..
(par * S..
(par * O..
tick
OMEGA