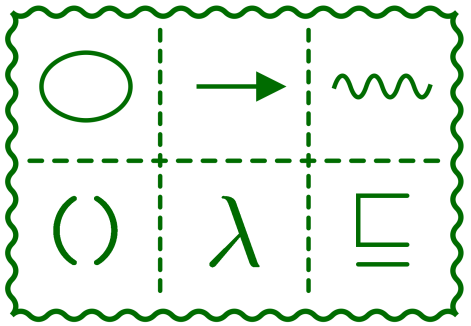


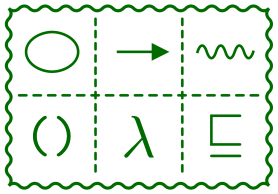
MCCSP

Libraries for Concurrent Pragramming in C based on CSP

PRINCIPIA Limited

Hisabumi HATSUGAI

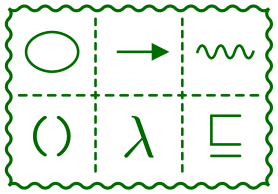
©2015 PRINCIPIA Limited



MCCSP: Why Another Library?

- Stackless Process
- Process implemented by Thread Pool
- Interoperability
  - C language
  - Threads created outside the lib can be attached

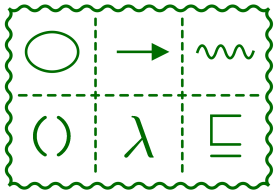
©2015 PRINCIPIA Limited



MCCSP Family

MCCSP Type	NT	TP
Stackless		○
Thread Pool		○
Thread Attachability	○	

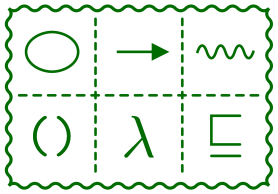
©2015 PRINCIPIA Limited



MCCSP NT API

- Processes
  - mccsp_create_process
  - mccsp_delete_process
- Channels
  - mccsp_create_channel
  - mccsp_delete_channel
- Synchronization (Communication)
  - mccsp_sync

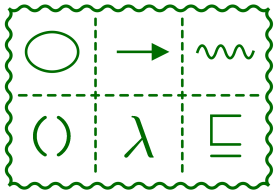
©2015 PRINCIPIA Limited



Sample Code: setup

```
int main()
{
   mccsp_channel_t ch = mccsp_create_channel();
   mccsp_process_t p = mccsp_create_process(1);
   p->v[0] = (intptr_t)ch;
   pthread_create(&p->thread, NULL, &producer, p);
   ...
}
```

©2015 PRINCIPIA Limited



Sample Code: mccsp_sync

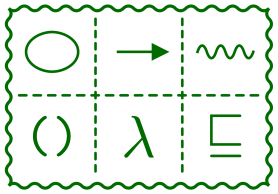
```
mccsp_sync_t syncv[3];
syncv[0].ch = ch_in0;
syncv[1].ch = ch_in1;
syncv[2].ch = ch_out;
syncv[2].slot = data;
k = mccsp_sync(self, syncv, 3, 0x03);
```

syncv length

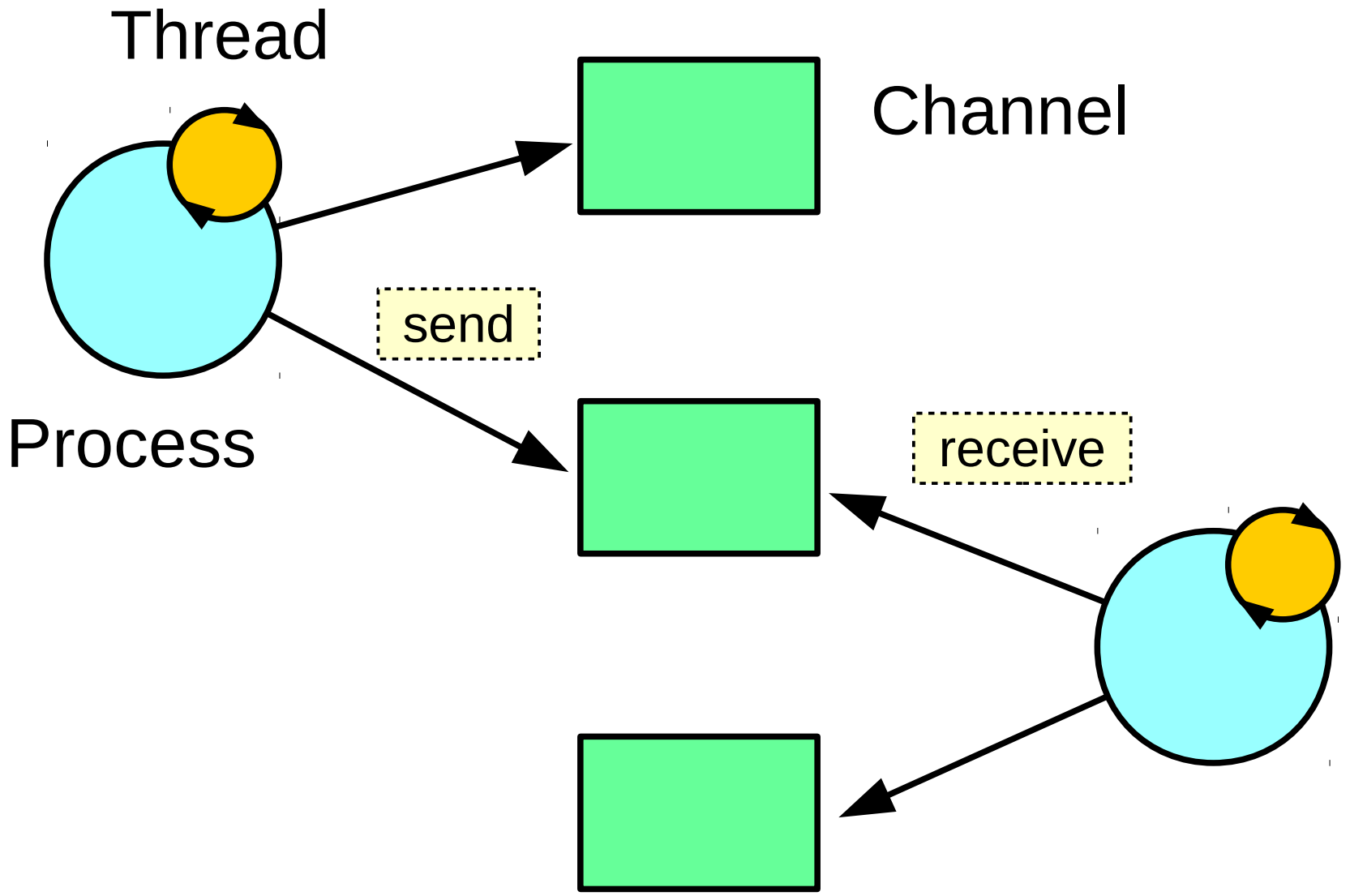
mccsp_process_t

bitmap
0: send
1: receive

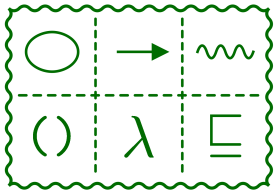
©2015 PRINCIPIA Limited



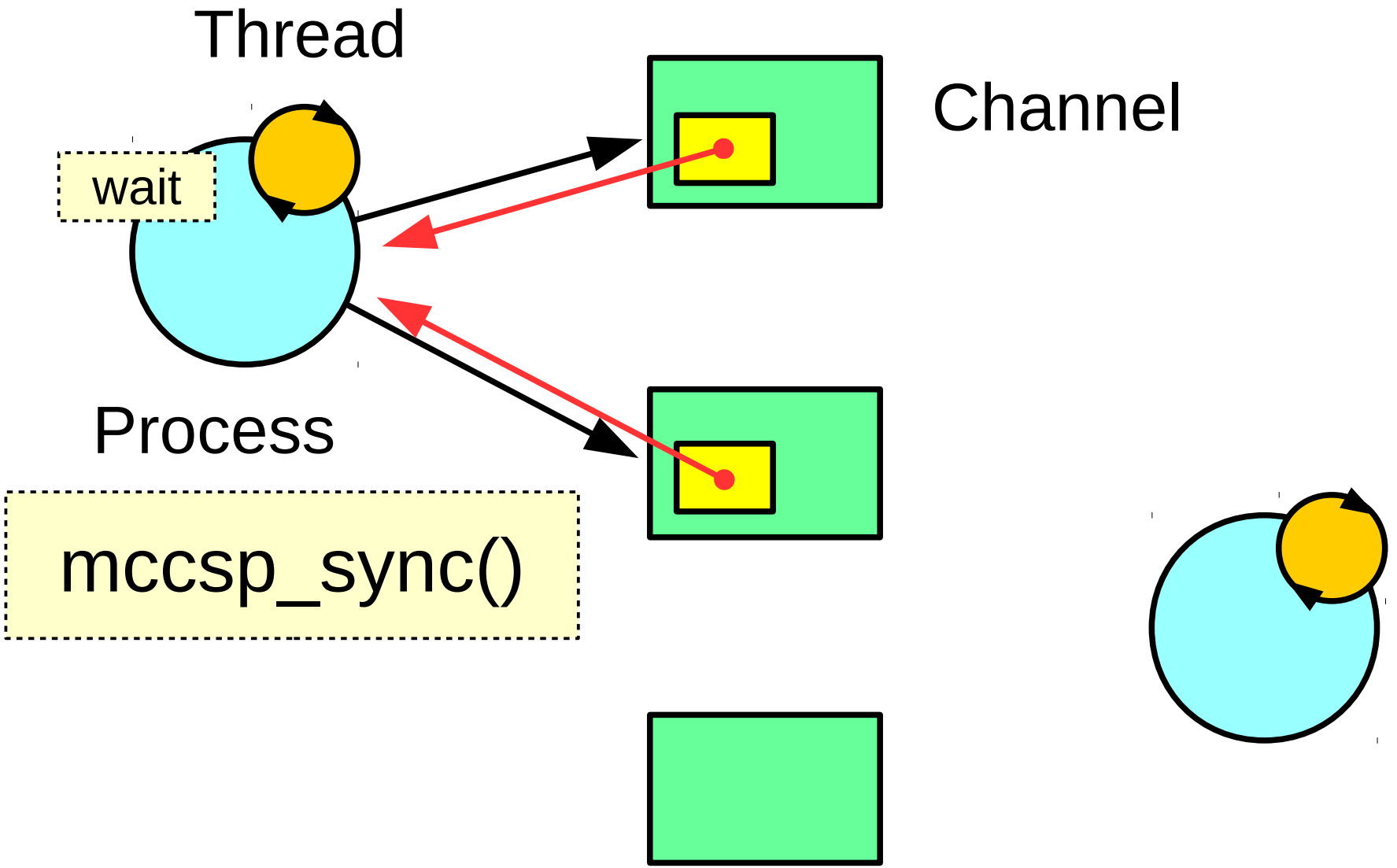
MCCSP NT



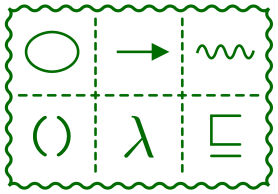
©2015 PRINCIPIA Limited



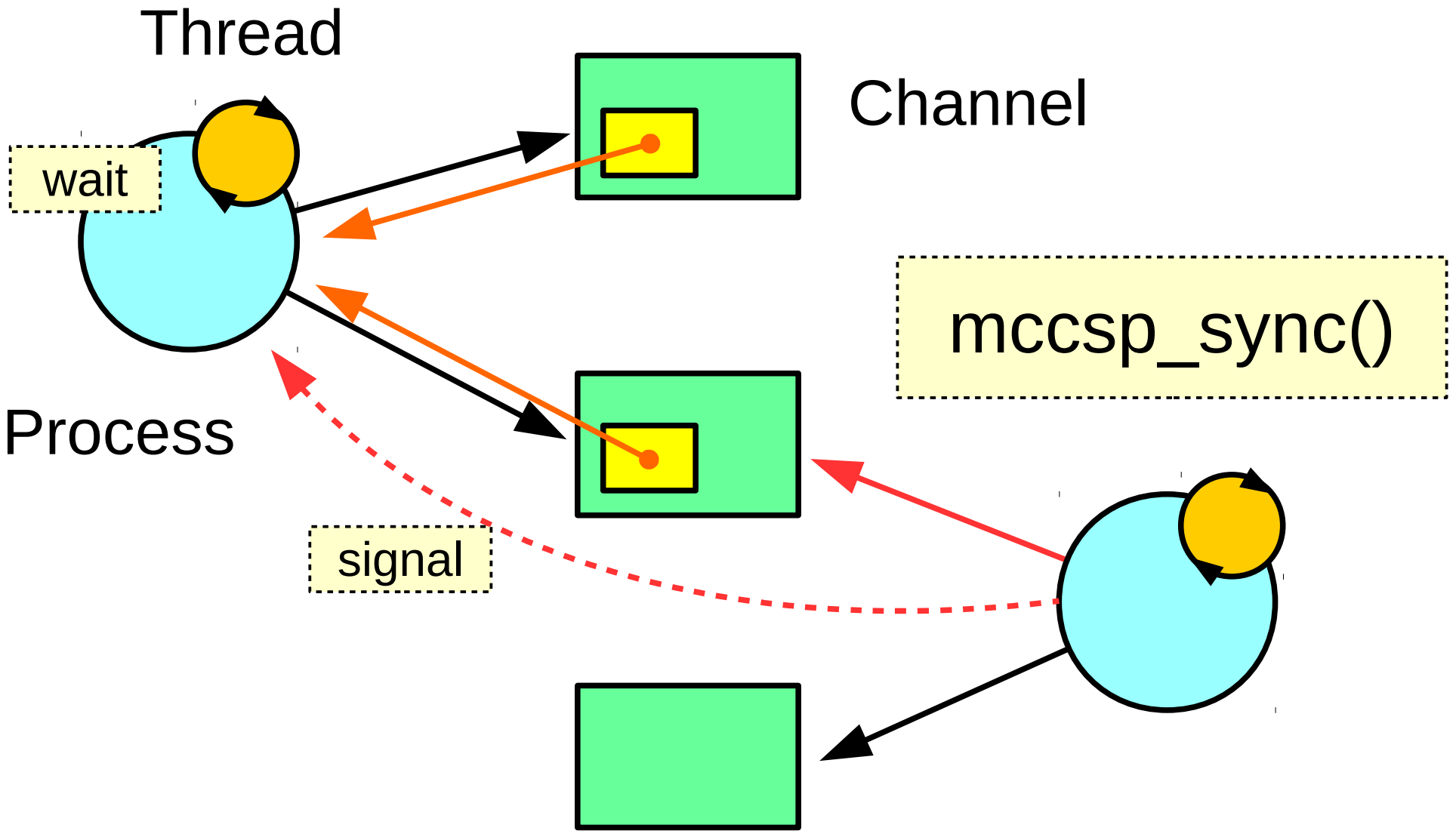
MCCSP NT



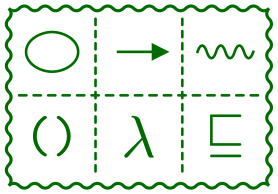
©2015 PRINCIPIA Limited



MCCSP NT



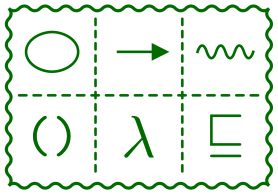
©2015 PRINCIPIA Limited



MCCSP TP API

- Channels
  - mccsp_create_channel
  - mccsp_delete_channel
- Initialization
  - mccsp_init(pf_process)

©2015 PRINCIPIA Limited



MCCSP TP

Channel

Process

Thread

sync

par

skip

Ω

call

return

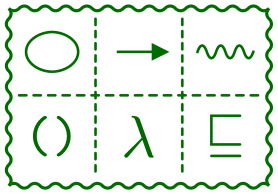
Ready Queue

Commands

```
MCCSP_SYNC
MCCSP_PAR
MCCSP_SKIP
```

```
int process_func(
  mccsp_process_t self) {
  ...
  return MCCSP_SYNC;
}
```

©2015 PRINCIPIA Limited



Model Check

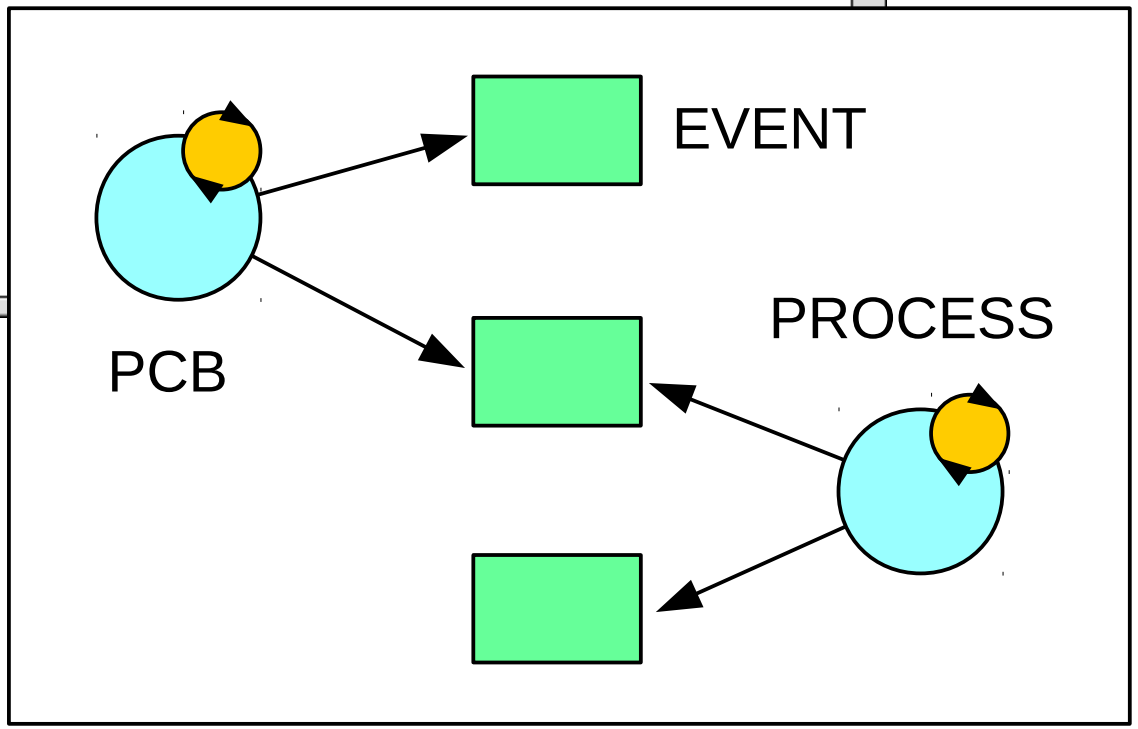
Process list - mccsp

Model | Definitions | Assertions | Status | Compile | Workspace

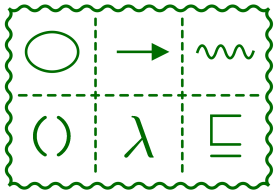
- PROCESS
- EVENT
- PCB
- SYSTEM
- HSYS
- SAFETY

Assertions - mccsp

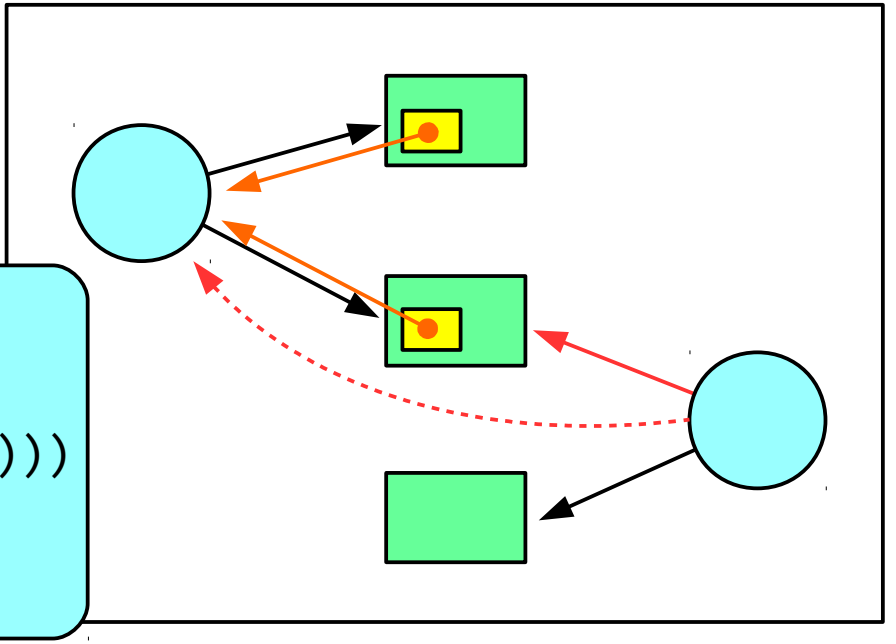
- (deadlock HSYS)
- (trace (SAFETY '()) HSYS)
- (trace HSYS (SAFETY '()))



©2015 PRINCIPIA Limited



Sync Spec



```
(define-process (PROCESS p)
  (xndc es
      (remove '() (powerset (interval 0 NE)))
    (! req (p es)
      (PROCESS-REG-LOOP p es '()))))
```

```
(define-process (PROCESS-REG-LOOP j cs rs)
  (if (null? cs)
      (PROCESS-WAIT j rs)
      (let ((c (car cs)))
        (alt
          (! (cst-free->reg c) (j)
             (PROCESS-REG-LOOP j (cdr cs) (cons
          (? (cst-reg c) (q)
             (alt
               (! (pst-wait->run q) (c j)
                  (! wait-event (p q)
                     (PROCESS-SYNC j rs)))
               (! (pst-notwait q) (c)
                  (PROCESS-REG-LOOP j cs rs)))))))))
```

```
(define-process (PROCESS-WAIT p es)
  (! pst-run->wait (p es)
     (? wait-event (q pp)
        (and (eq? p pp)
             (not (eq? p q)))
        (? pst-event (pp e q) (eq? p pp)
           (seq
             (PROCESS-SUB-RELEASE es)
             (PROCESS p))))))
```

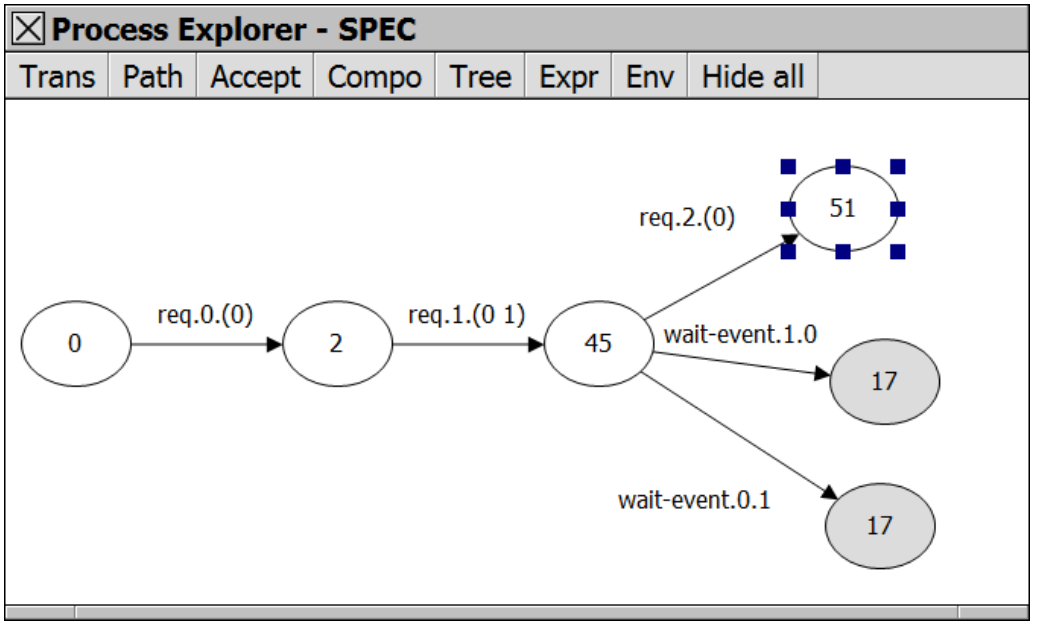
Sync Spec $\sqsubseteq_F$ System

Transitions

event	target
wait-event.0.1	5
wait-event.0.2	7
wait-event.1.0	5
wait-event.1.2	**2**
wait-event.2.0	7
wa	

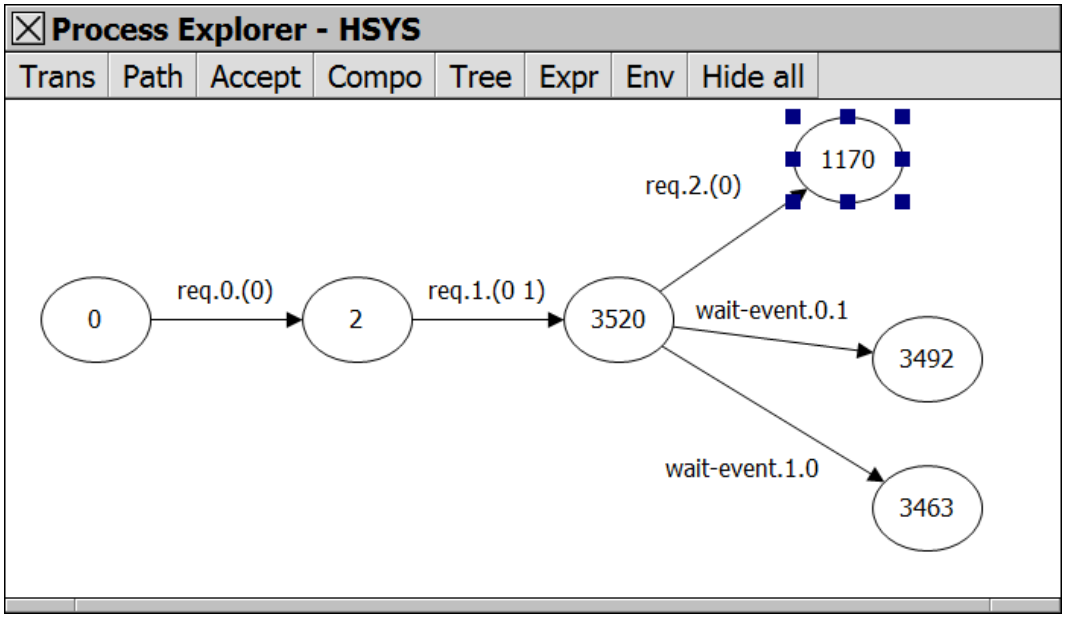
Acceptances

(wait-event.2.0)	wait-event.1
(wait-event.0.2)	
(wait-event.2.1)	
(wait-event.1.2)	
(wait-event.1.0)	

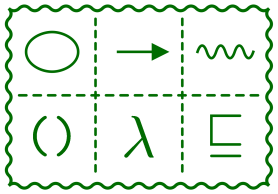


Transitions

event	target
wait-event.0.1	655
wait-event.0.2	654
wait-event.1.0	280
wait-event.1.2	284
wait-event.2.0	59
wait-event.2.1	734



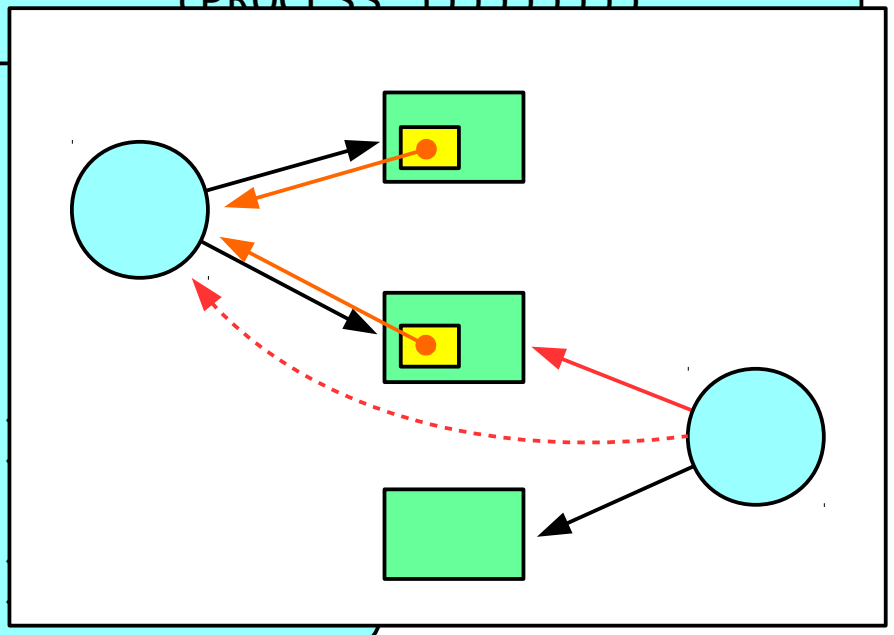
©2015 PRINCIPIA Limited



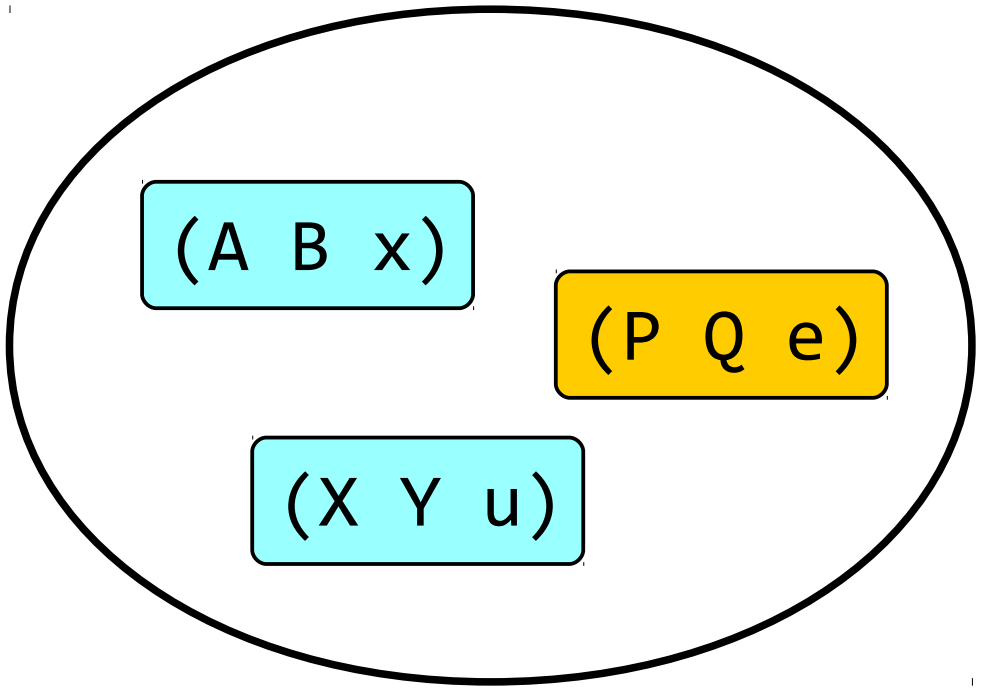
Model Check: Safety Spec

```
(define-process (PROCESS-WAIT j cs)
  (! (pst-reg->wait j) ((list-sort < cs))
     (xalt e (wait-events j)
       (! e
          (? (pst-event j) (c q)
             (! (sync j q c)
                (seq
                  (PROCESS-SUB-RELEASE cs)
                  (PROCESS j)))))))))
```

```
(define-process (PROCESS-REG-LO
  (if (null? cs)
      (PROCESS-WAIT j rs)
      (let ((c (car cs)))
        (alt
          (! (cst-free->reg c)
             (PROCESS-REG-LOOP
          (? (cst-reg c) (q)
             (alt
               (! (pst-wait->run q) (c j)
                  (! (wakeup j q)
                     (! (sync j q c)
                        (PROCESS-SYNC j rs)
               (! (pst-notwait q) (c)
                  (PROCESS-REG-LOOP j cs rs
```



Model Check: Safety Spec



Q's next event

(Q P e) → remove (P Q e)

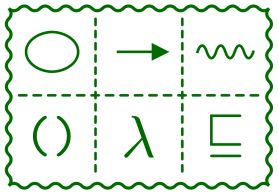
P's next event

(P Ψ ζ) → add

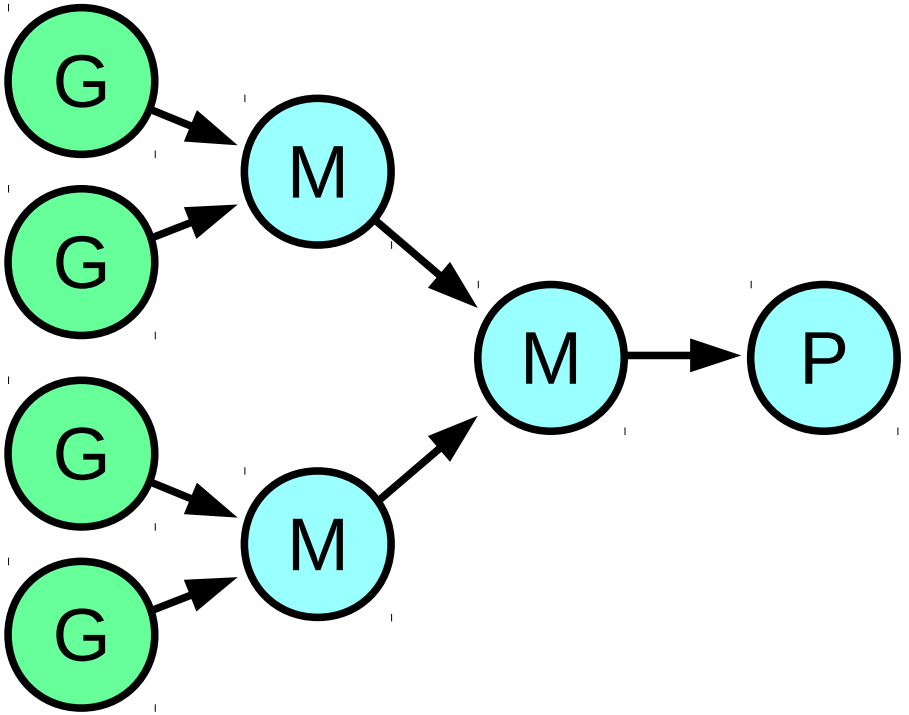
(Ψ ≠ Q)

SAFETY $=_T$ SYSTEM

©2015 PRINCIPIA Limited

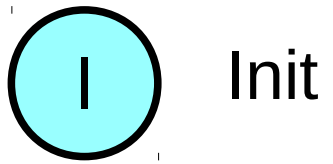


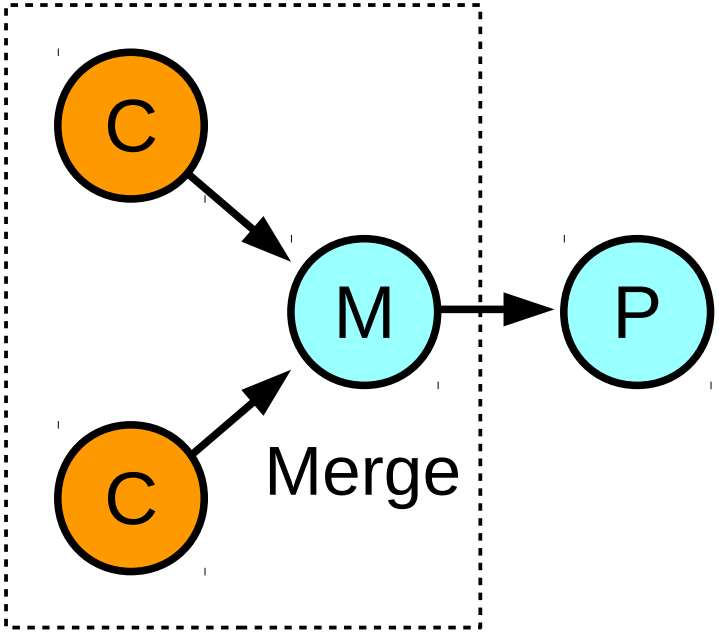
Benchmark A



Generator

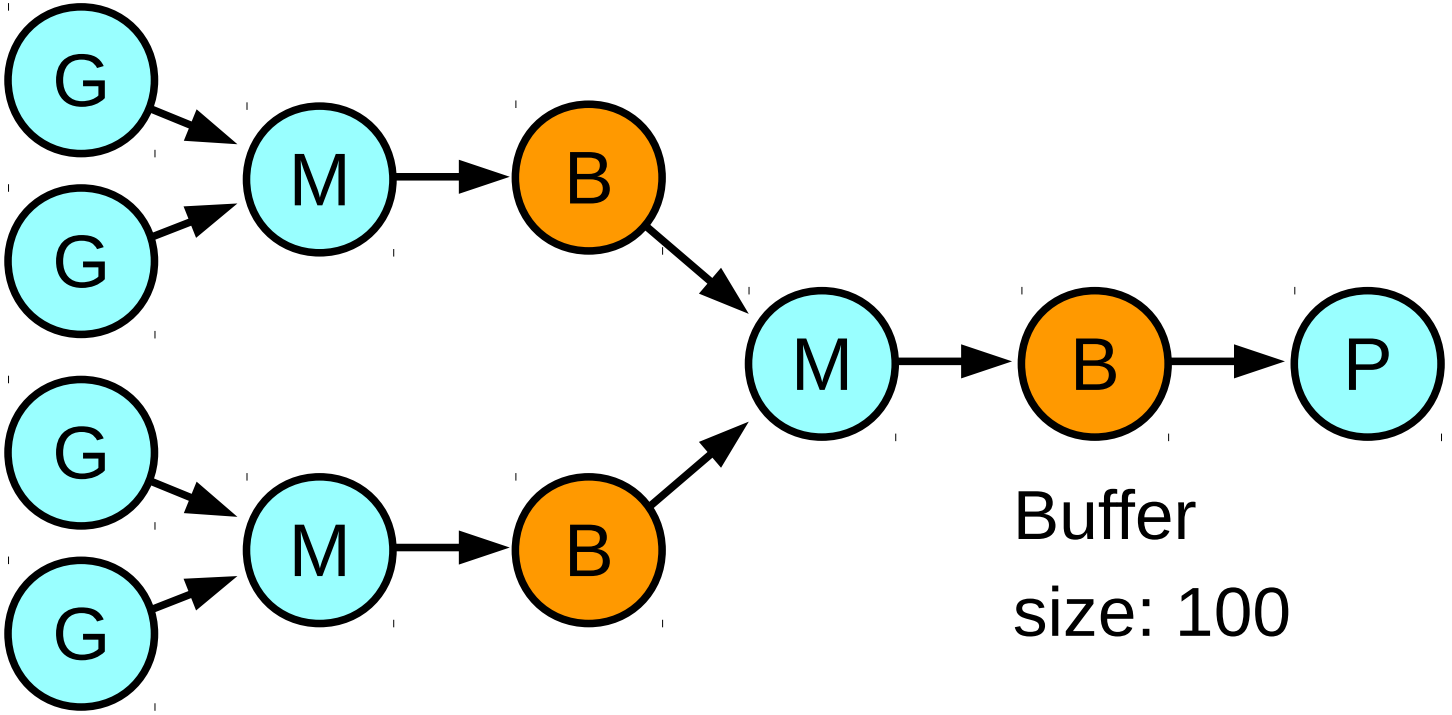
each sends 1,000 msgs



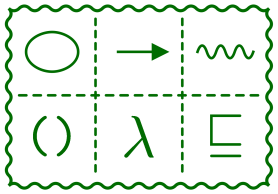


©2015 PRINCIPIA Limited

Benchmark B



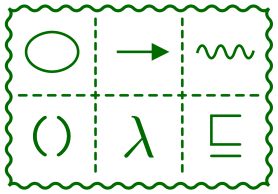
©2015 PRINCIPIA Limited



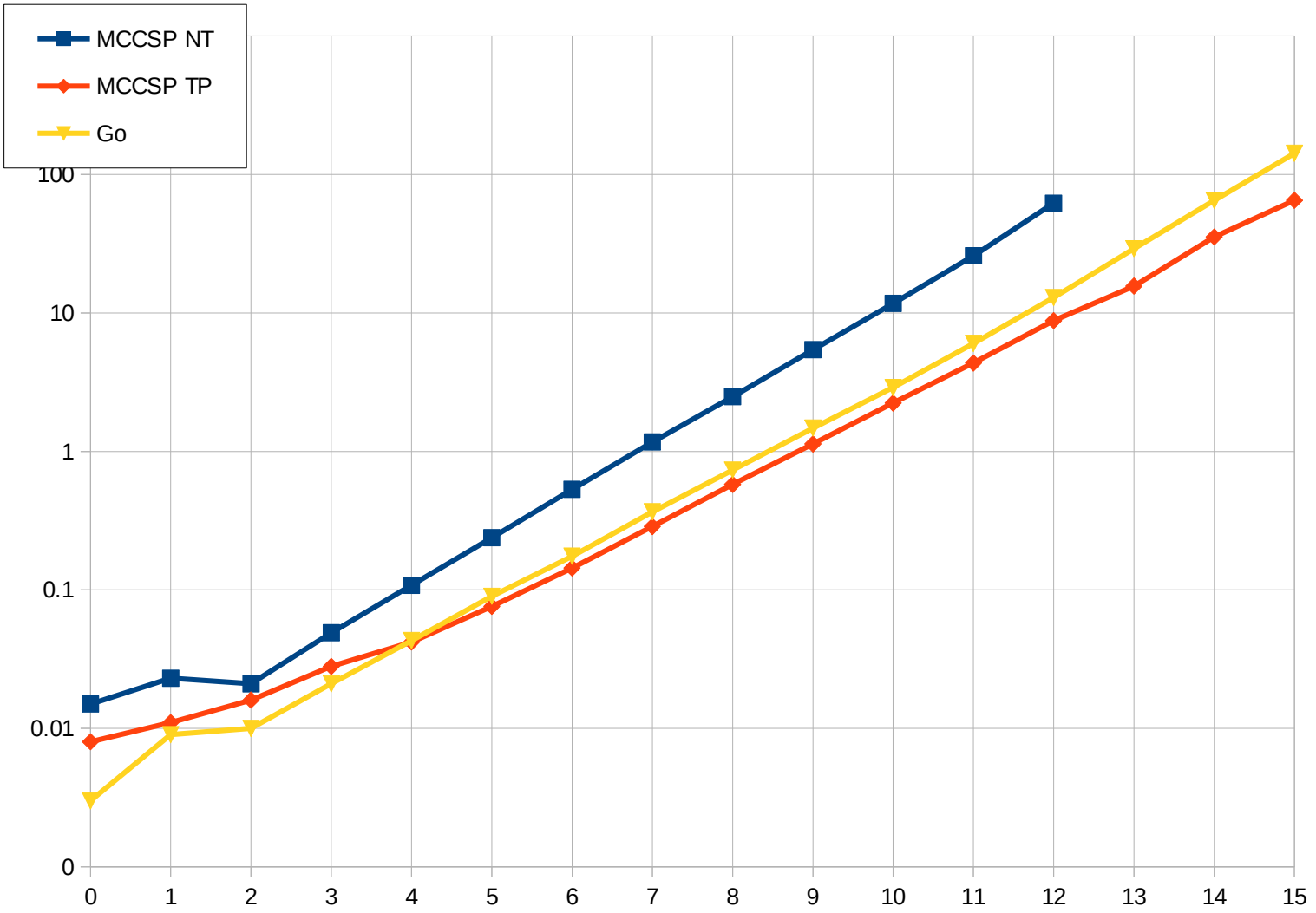
Benchmark Environment

- Hardware
  - CPU: Core i7 4770 3.7GHz
  - RAM: 32GB
- Software
  - OS: Linux 3.16
  - GCC 4.9
  - Go 1.4.2

©2015 PRINCIPIA Limited



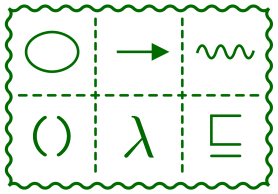
Benchmark A: level-time



Time (sec)

level ($N_{process} = 2^{level+2}$)

©2015 PRINCIPIA Limited



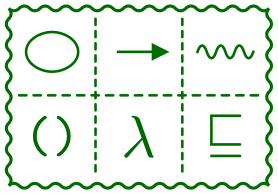
Benchmark A: level-time

level	MCCSP NT	MCCSP TP	Go
0	0.015	0.008	0.003
1	0.023	0.011	0.009
2	0.021	0.016	0.010
3	0.049	0.028	0.021
4	0.108	0.042	0.043
5	0.238	0.076	0.090
6	0.532	0.144	0.175
7	1.170	0.287	0.366
8	2.487	0.578	0.734
9	5.427	1.131	1.474
10	11.697	2.235	2.895
11	25.816	4.350	6.019
12	61.964	8.792	12.891
13		15.626	29.067
14		35.380	64.847
15		64.959	141.903

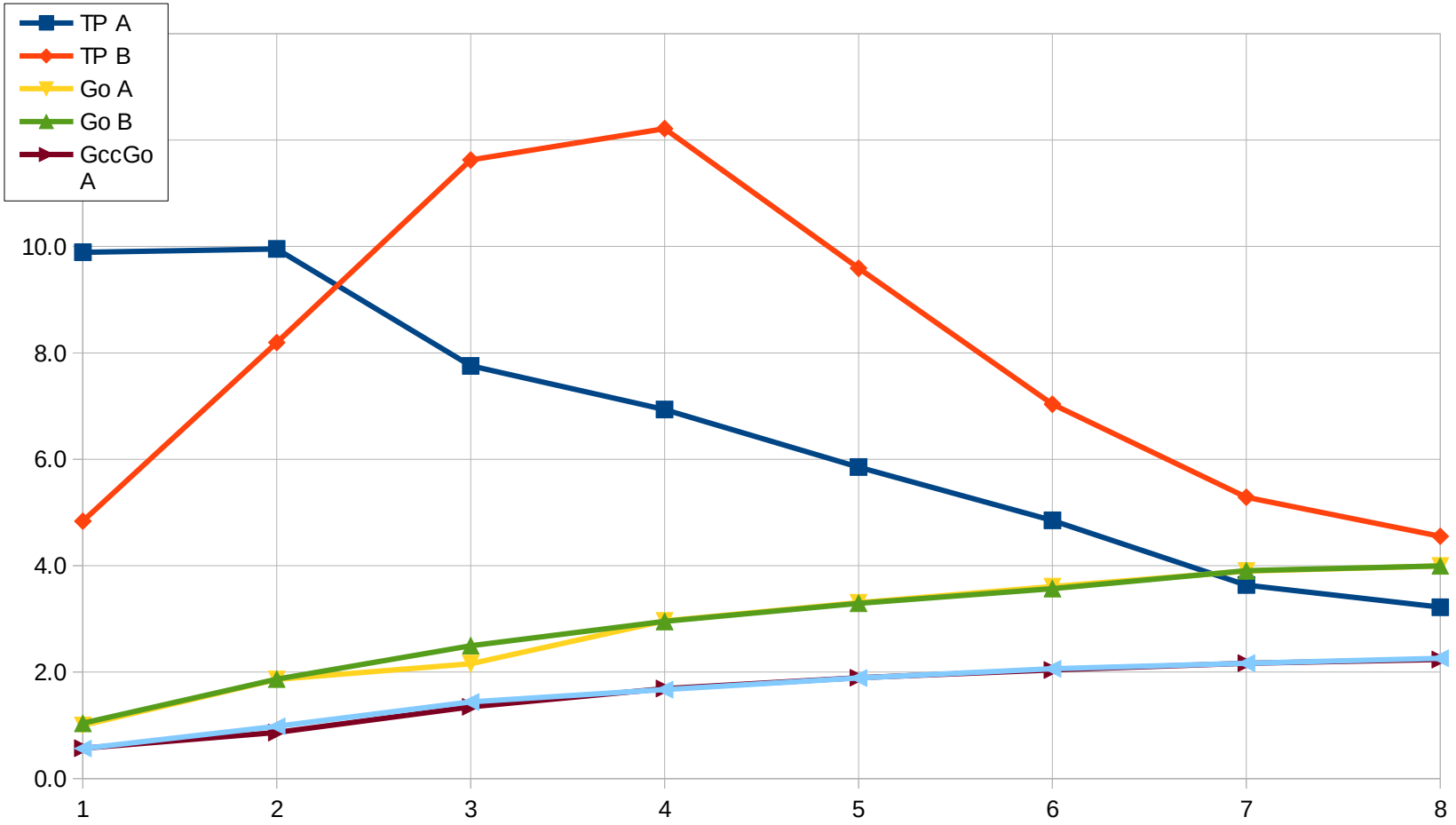
Time (sec)

N.B.
NT: 4 thread
Go: 8 thread

©2015 PRINCIPIA Limited



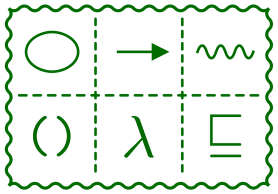
Becnhamrk: N_{thread} - Speed



Speed

N_{thread}

©2015 PRINCIPIA Limited



Bechmark: N_{thread} - Time

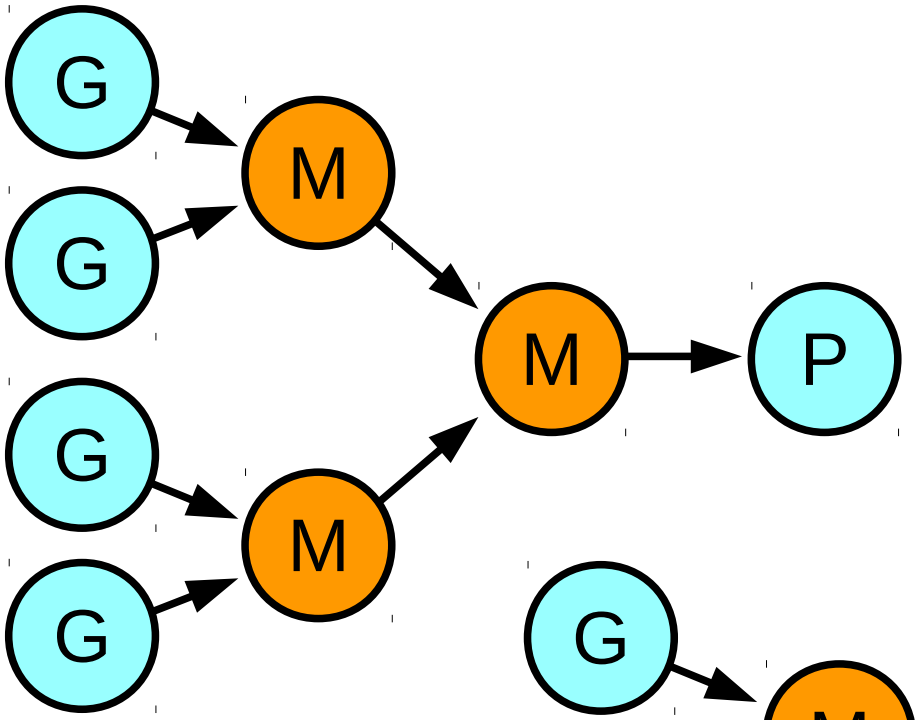
Time (sec)

N_{thread}	TP A	TP B	Go A	Go B	GccGo A	GccGo B
1	5.167	10.562	51.105	49.207	89.617	90.770
2	5.134	6.237	27.439	27.350	59.127	52.002
3	6.589	4.396	23.677	20.461	37.944	35.508
4	7.369	4.184	17.264	17.312	30.217	30.601
5	8.730	5.329	15.466	15.513	26.973	27.043
6	10.534	7.263	14.171	14.320	25.046	24.777
7	14.052	9.665	13.105	13.085	23.578	23.590
8	15.870	11.225	12.796	12.793	22.860	22.602

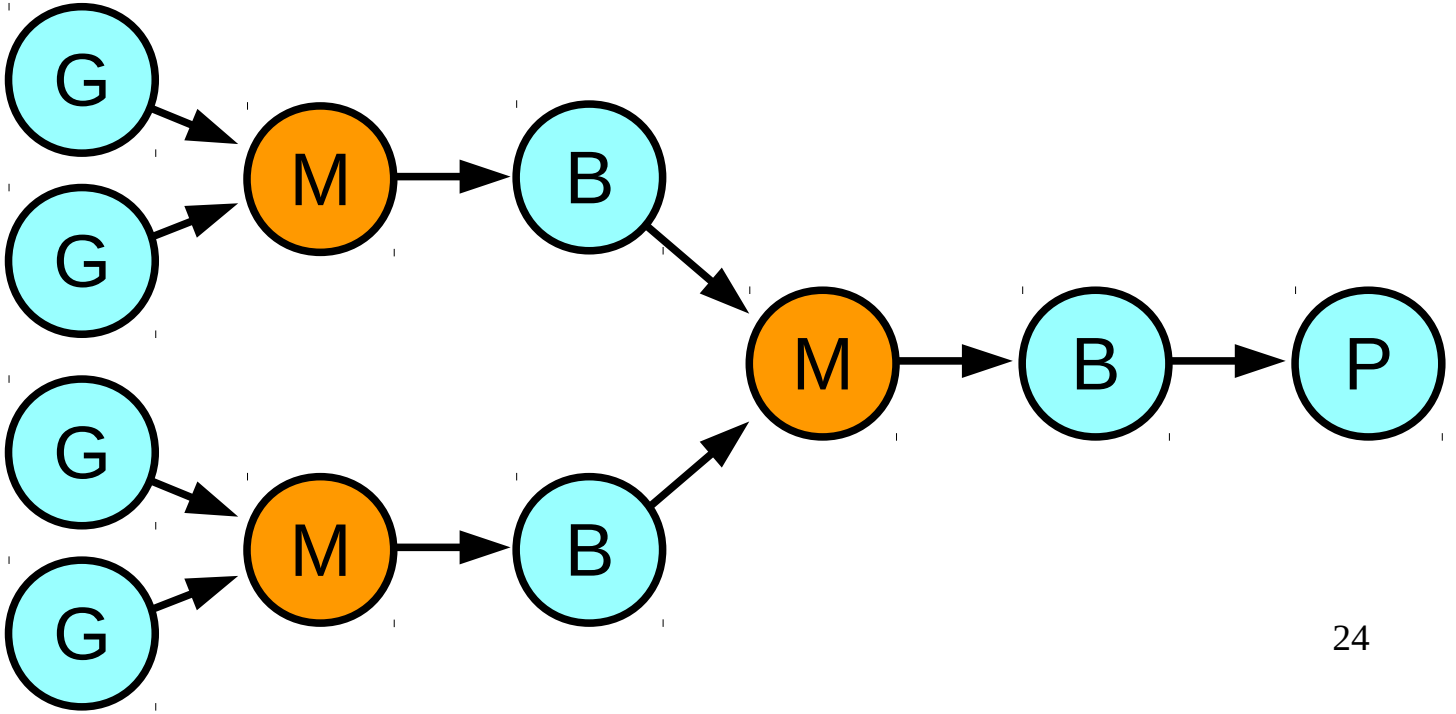
level = 12 (A:16,384 processes, B:32,768 processes)

©2015 PRINCIPIA Limited

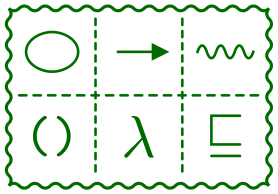
Internal Calculation as a Load



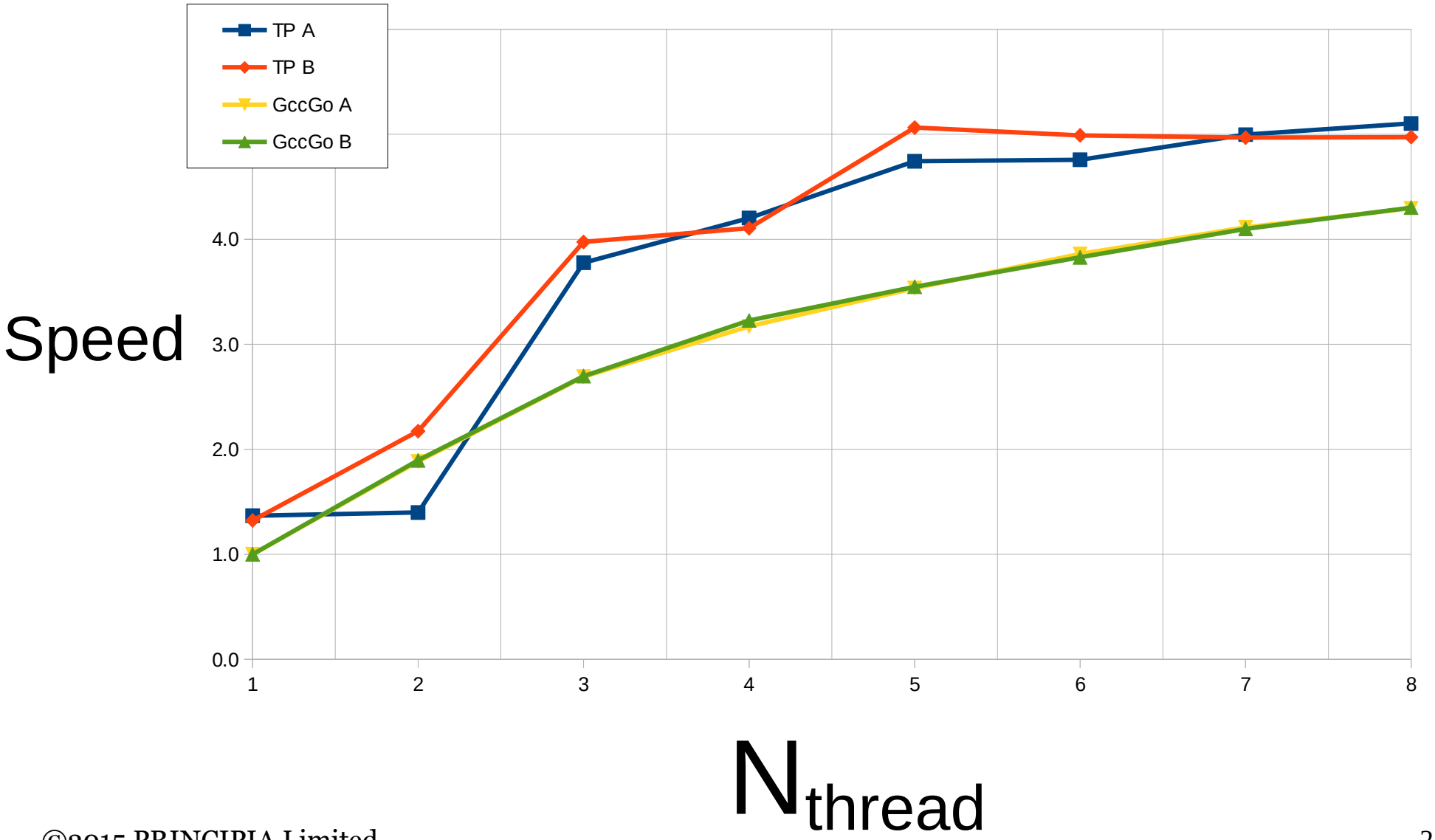
Added heavy calculation with no memory access to each Merge



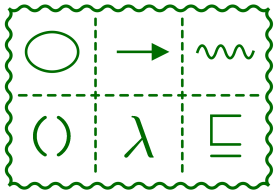
©2015 PRINCIPIA Limited



Benchmark w/Load: N_{thread} - Speed



©2015 PRINCIPIA Limited



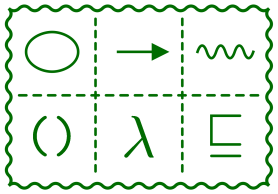
Benchmark w/Load: N_{thread} - Time

Time (sec)

N_{thread}	TP A	TP B	GccGo A	GccGo B
1	144.577	149.500	197.655	197.624
2	141.397	90.974	104.912	104.270
3	52.333	49.737	73.447	73.283
4	47.042	48.142	62.348	61.255
5	41.677	39.040	55.914	55.715
6	41.551	39.618	51.202	51.628
7	39.561	39.790	48.048	48.219
8	38.737	39.748	46.020	45.956

level = 12 (A:16,384 processes, B:32,768 processes)

©2015 PRINCIPIA Limited



Conclusion

- Developed two CSP libraries in C
  - NT: Thread Attachability
  - TP: Stackless and Tread Pool
- Checked the Algorithm by Model Checker
- Good Performance
  - Comparable to Go
  - Not scale well

©2015 PRINCIPIA Limited