Buffer Sharing in CSP-like Programs

Nalini Vasudevan   Stephen A. Edwards

Columbia University

Motivation

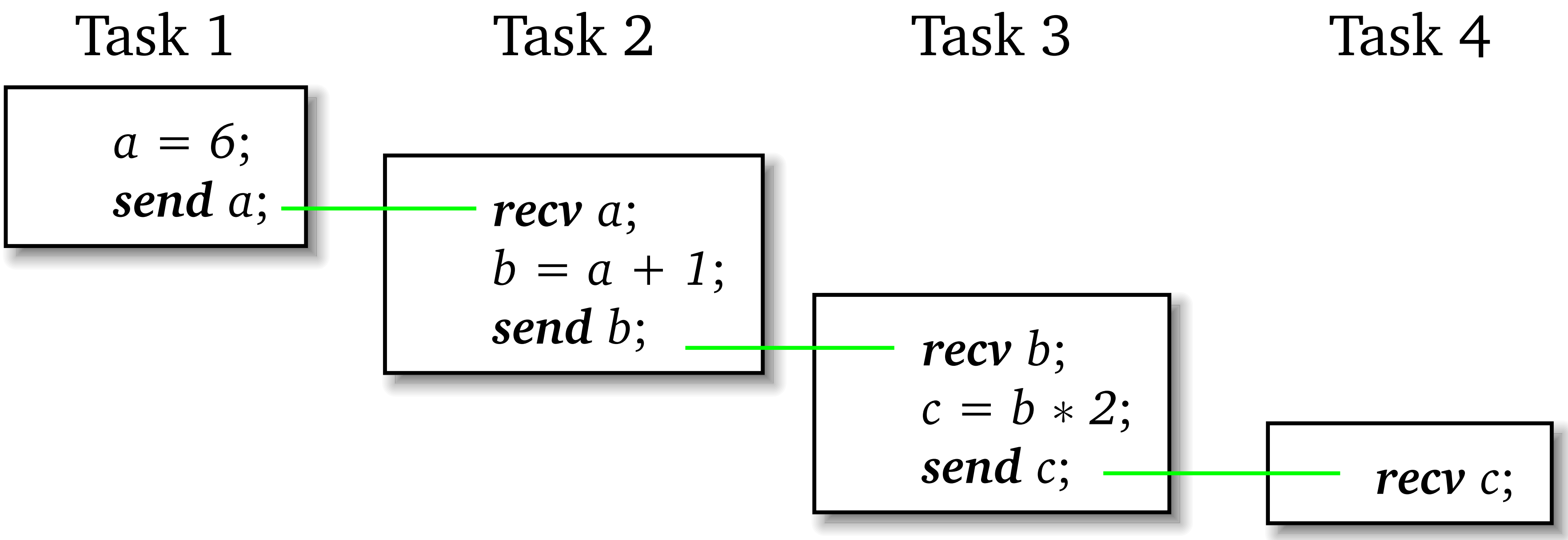
Task 1	Task 2	Task 3	Task 4
$a = 6;$ **send** $a;$	**recv** $a;$ $b = a + 1;$ **send** $b;$	**recv** $b;$ $c = b * 2;$ **send** $c;$	**recv** $c;$

Motivation

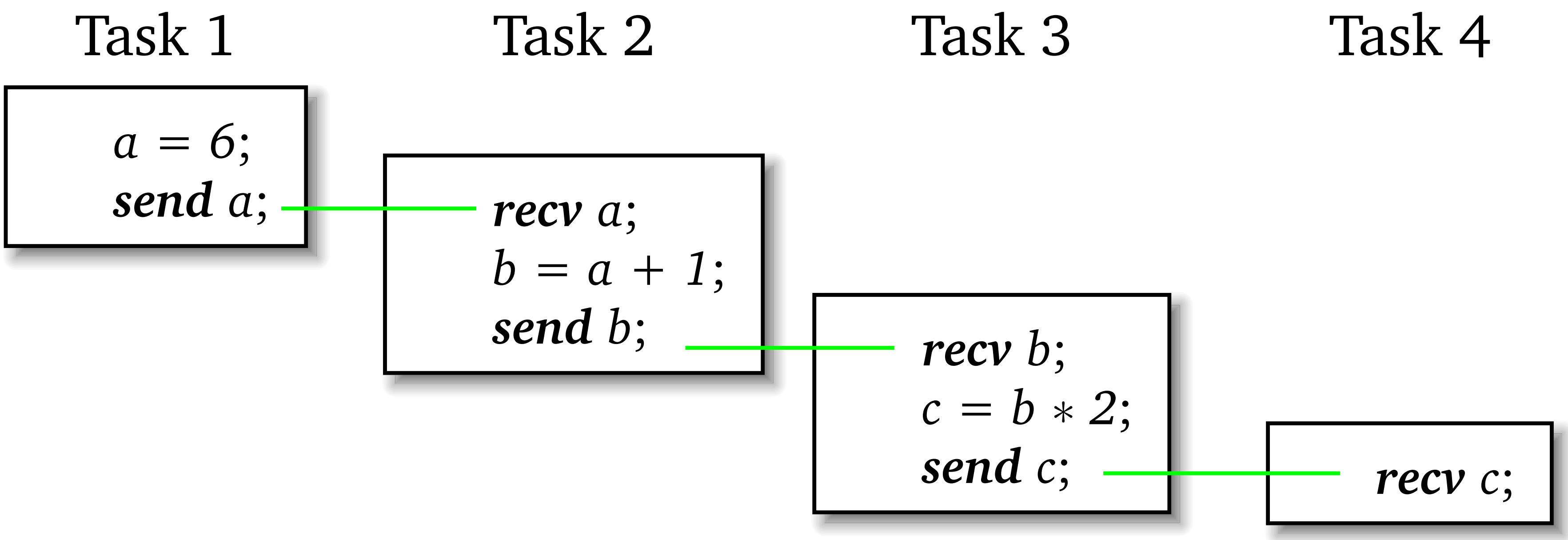
Task 1	Task 2	Task 3	Task 4
$a = 6$; **send** a;	**recv** a; $b = a + 1$; **send** b;	**recv** b; $c = b * 2$; **send** c;	**recv** c;

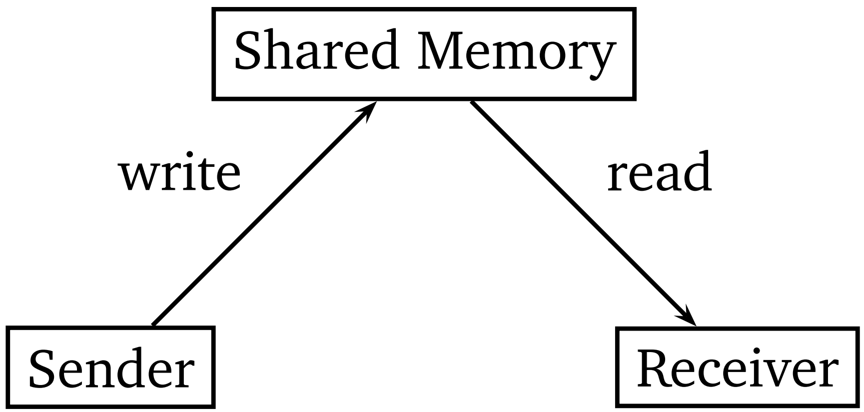
- Use rendezvous model of communication

Motivation

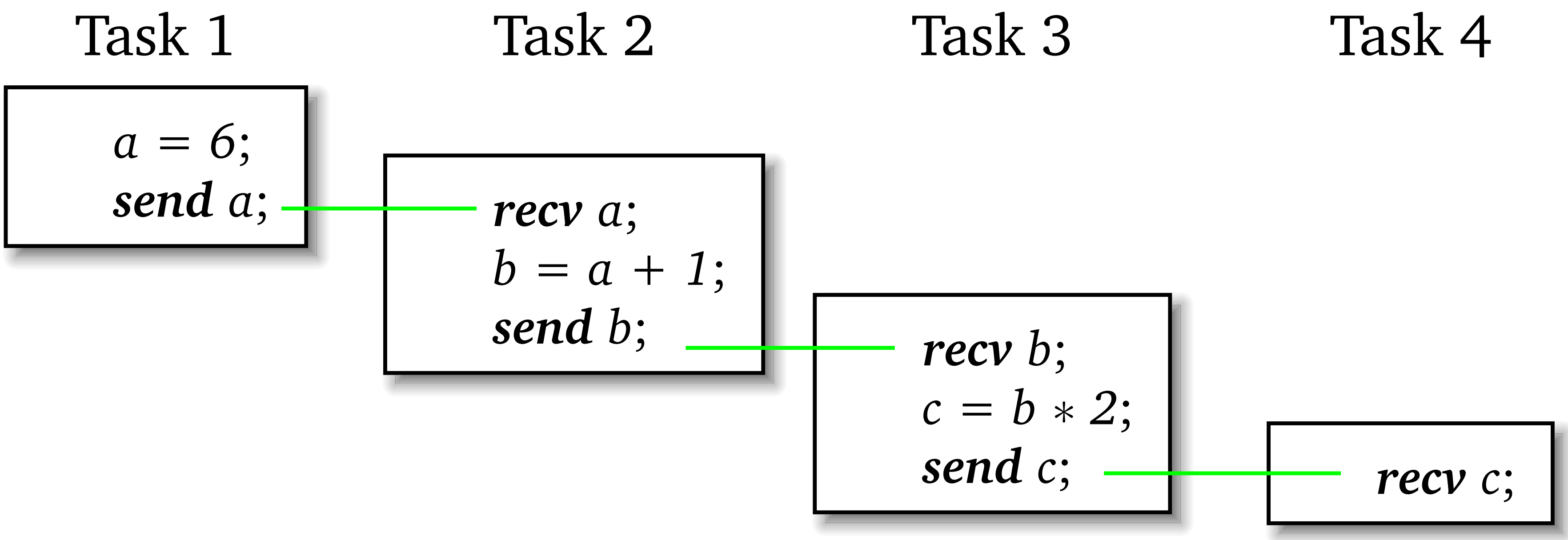

Task 1
Task 2
Task 3
Task 4
a = 6;
send a;
recv a;
b = a + 1;
send b;
recv b;
c = b * 2;
send c;
recv c;

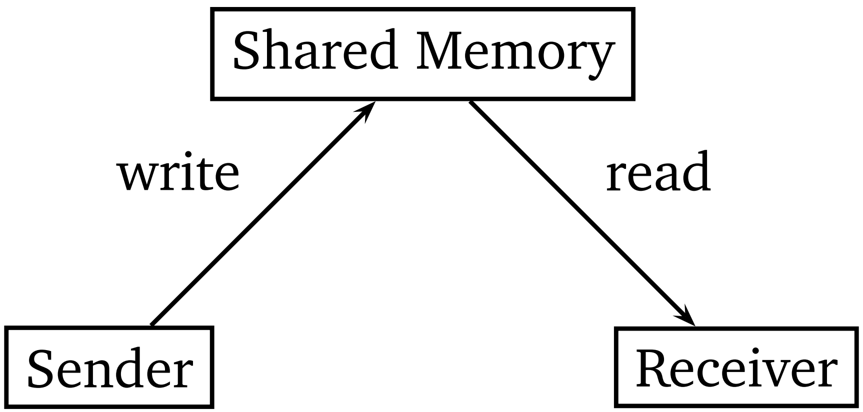
Motivation


Task 1
Task 2
Task 3
Task 4
a = 6;
send a;
recv a;
b = a + 1;
send b;
recv b;
c = b * 2;
send c;
recv c;


Shared Memory
write
read
Sender
Receiver

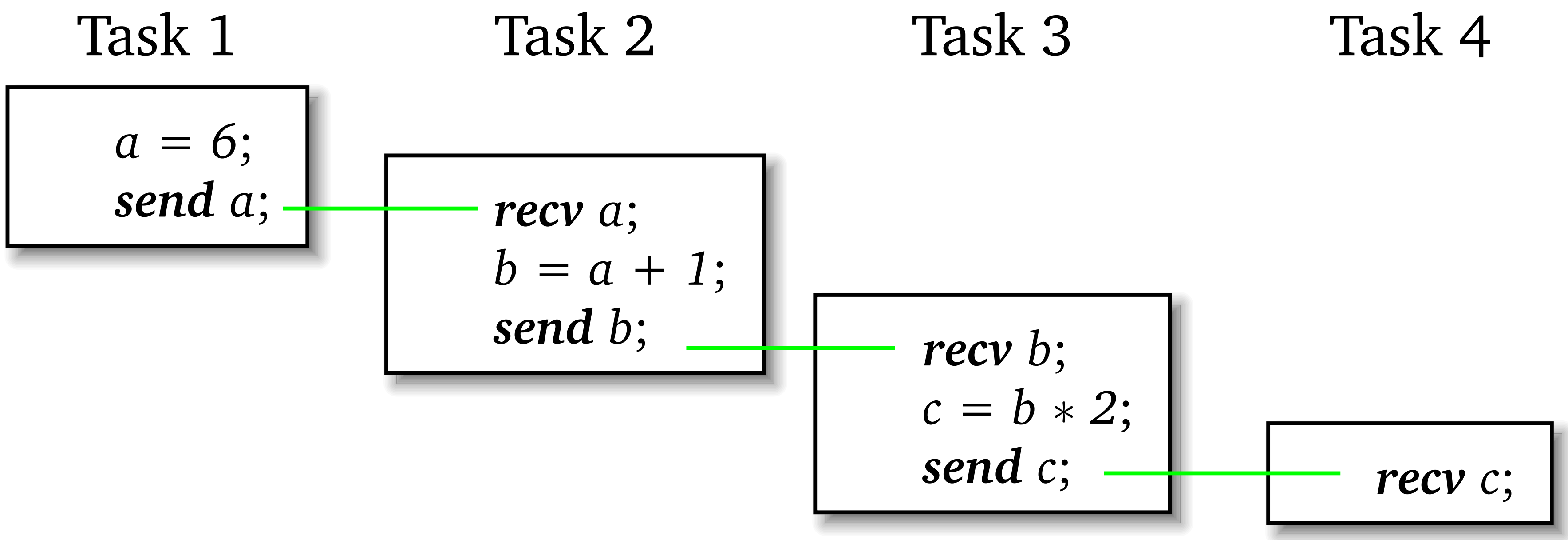
Motivation





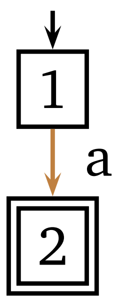
- a, b and c can share buffers

Buffer Sharing



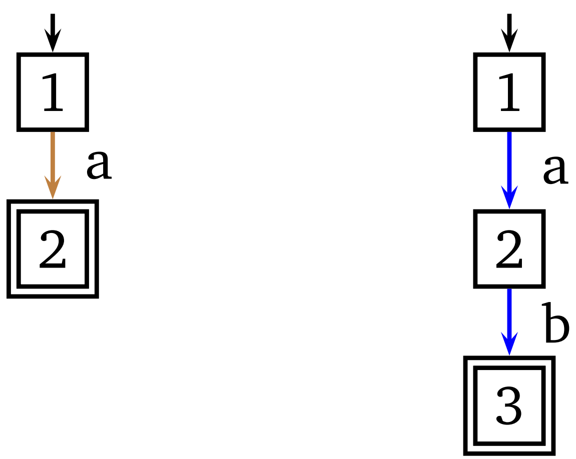
Buffer Sharing

Task 1	Task 2	Task 3	Task 4
$a = 6;$ **send** $a;$	**recv** $a;$ $b = a + 1;$ **send** $b;$	**recv** $b;$ $c = b * 2;$ **send** $c;$	**recv** $c;$

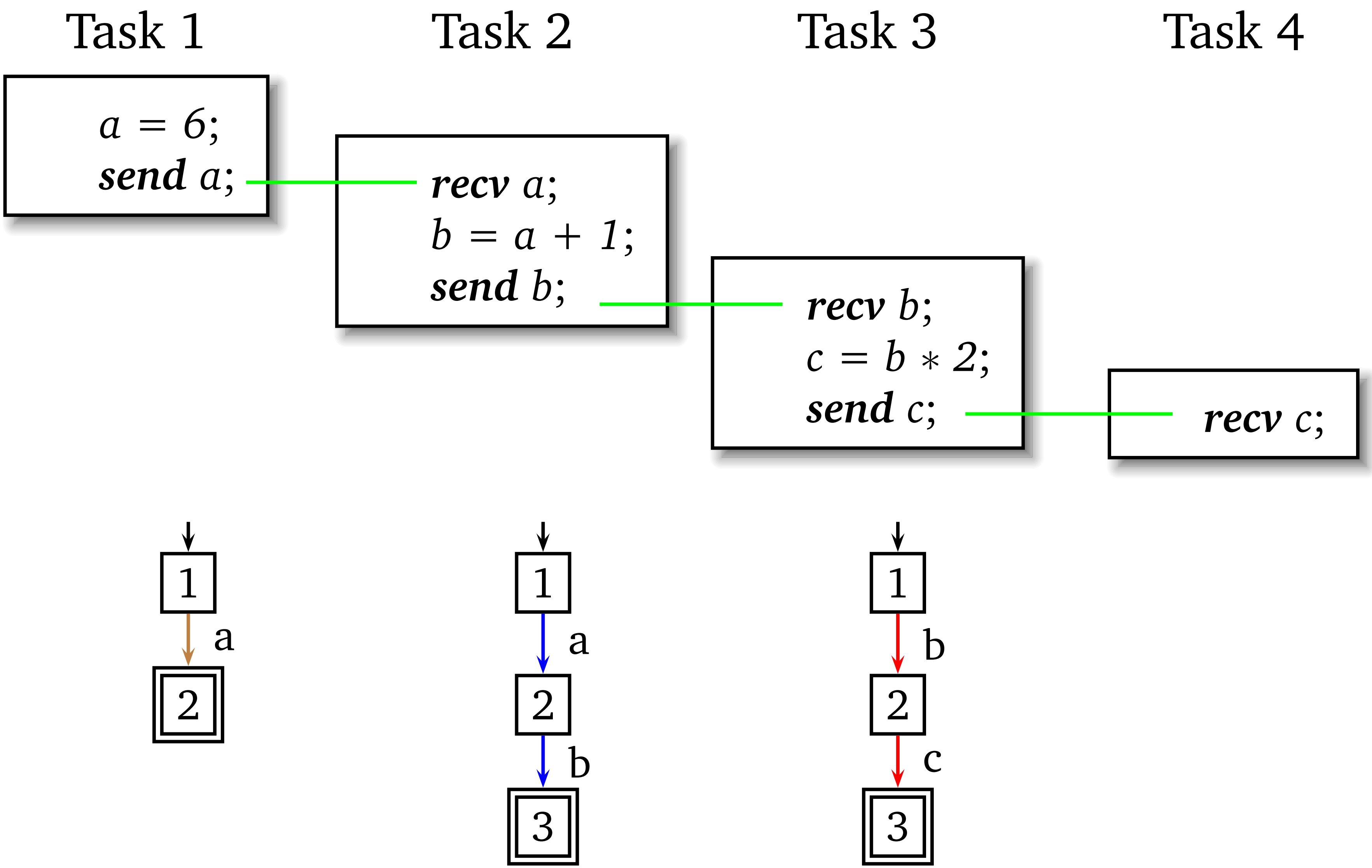


Buffer Sharing

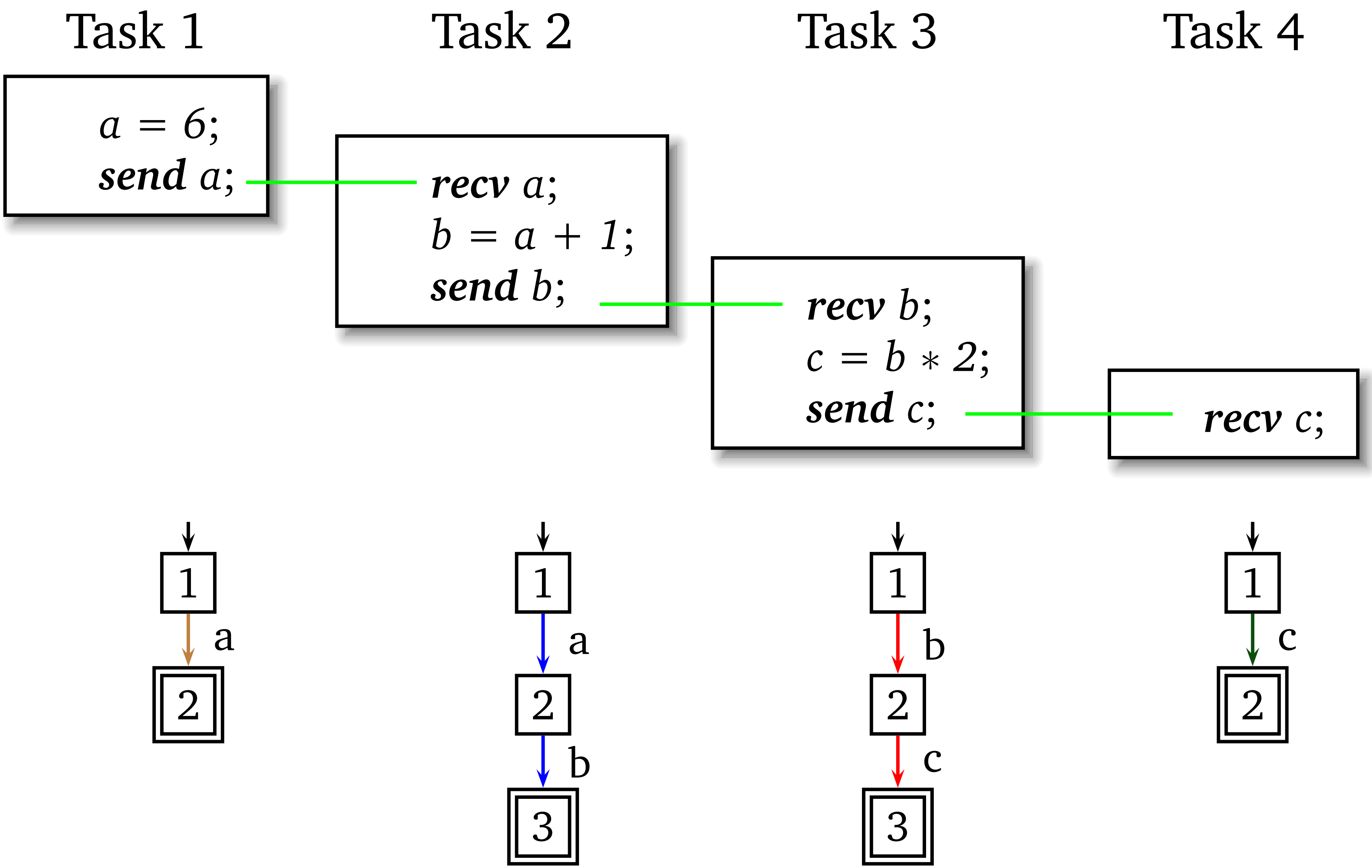
Task 1	Task 2	Task 3	Task 4
$a = 6$; **send** a;	**recv** a; $b = a + 1$; **send** b;	**recv** b; $c = b * 2$; **send** c;	**recv** c;



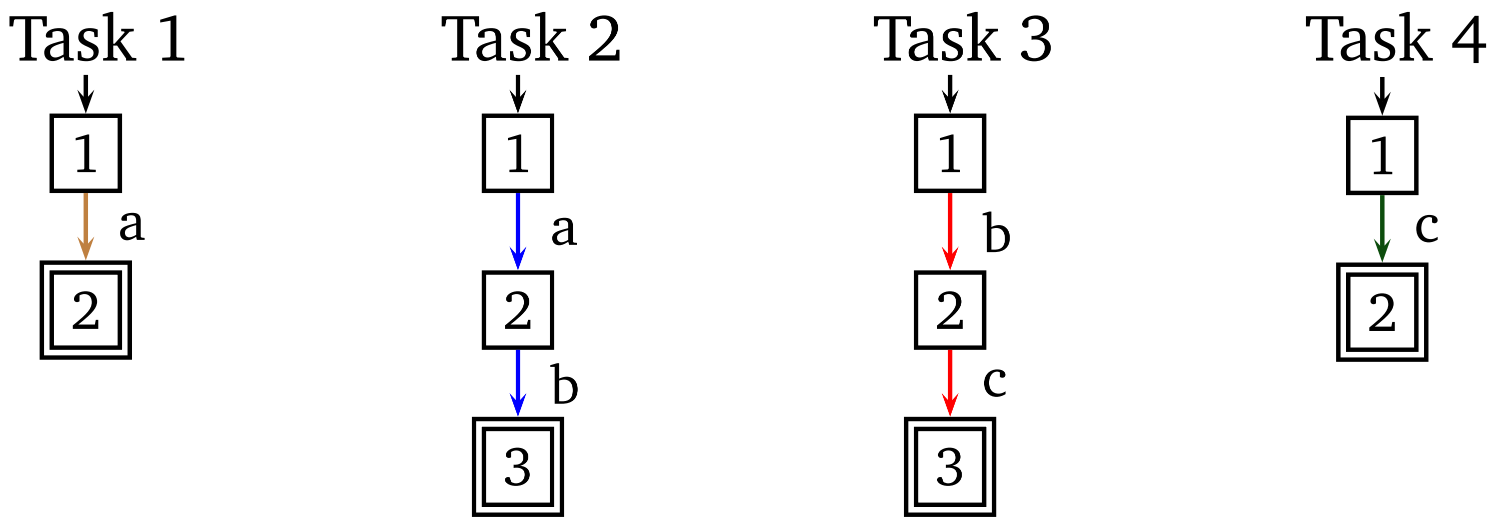
Buffer Sharing



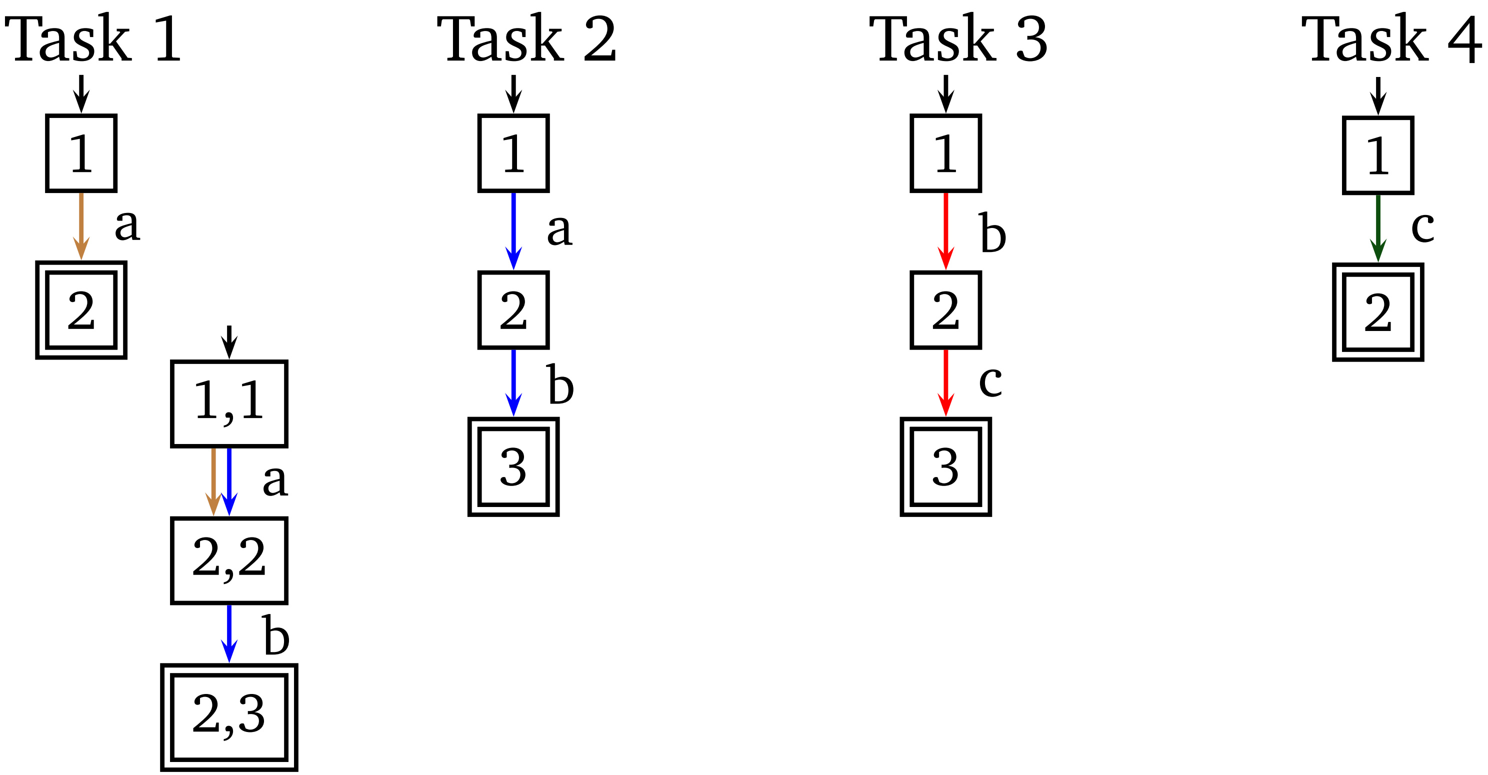
Buffer Sharing



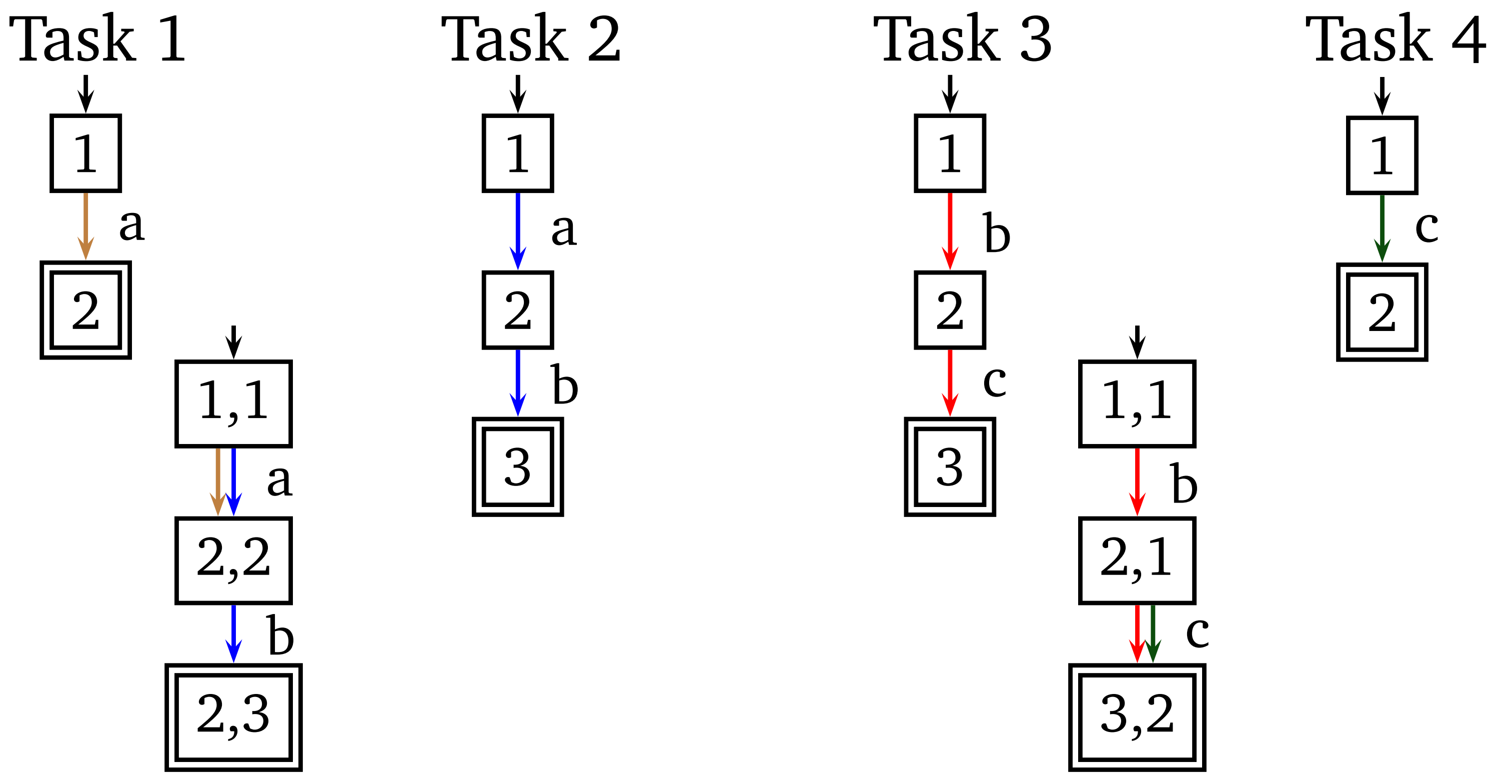
Automata Composition



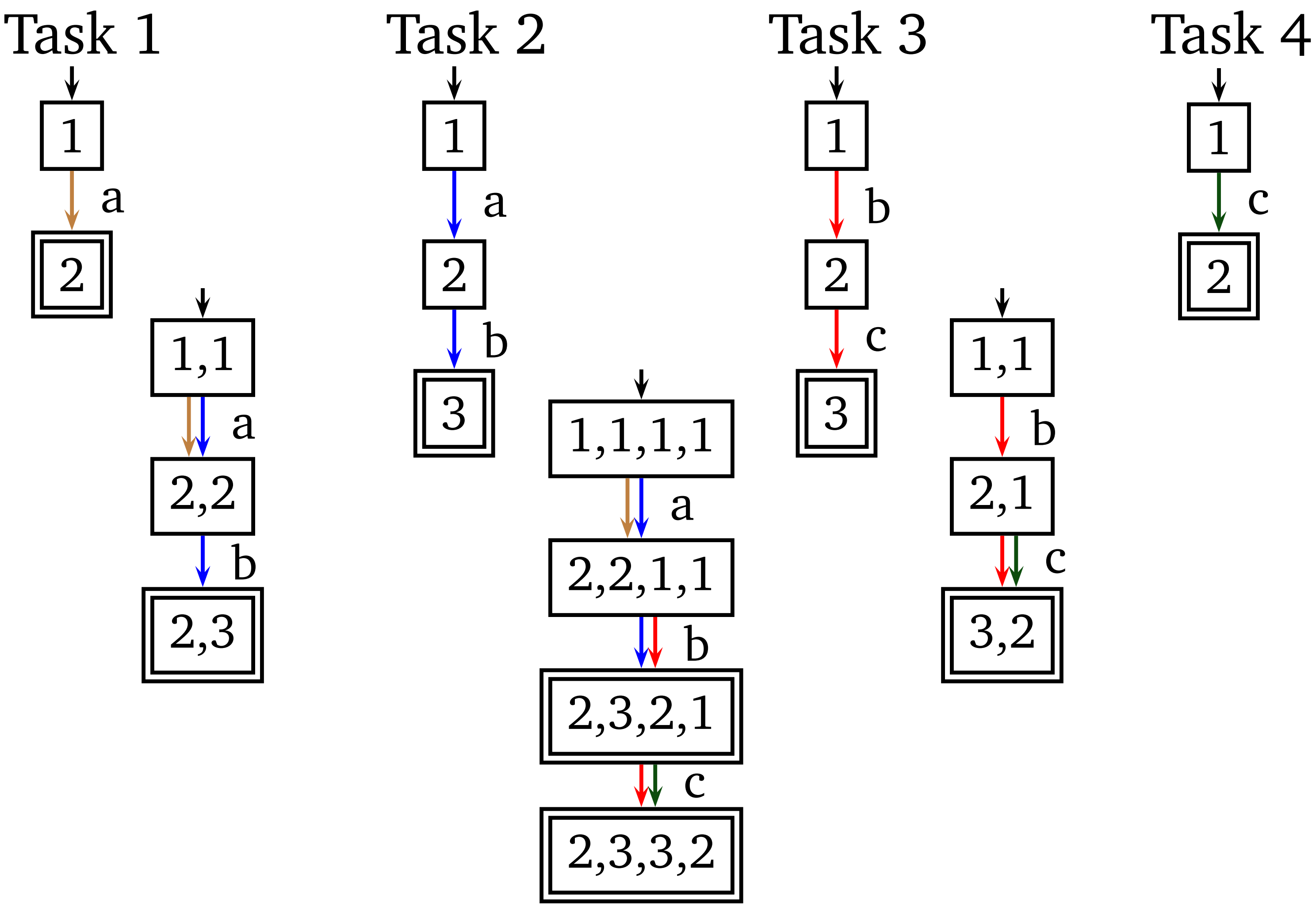
Automata Composition



Automata Composition



Automata Composition



Another Example

Task 1

```
for (int i = 0; i < 15; i++)
    recv a;

send b = 10;
send d = 8;
```

Task 2

```
recv b;
recv c;
```

Another Example

Task 1

```
for (int i = 0; i < 15; i++)
    recv a;

send b = 10;
send d = 8;
```

Task 2

```
recv b;
recv c;
```

Another Example

Task 1

for (*int* $i = 0$; $i < 15$; $i{+}{+}$)
  recv a;

send $b = 10$;
send $d = 8$;

Task 2

recv b;
recv c;

- a, b and d can share buffers
- b and c can share buffers
- a and c can share buffers
- **c and d cannot share buffers**

Another Example

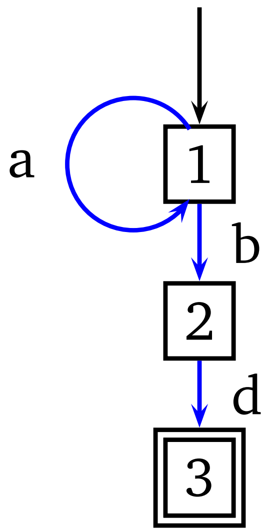
Task 1

```
for (int i = 0; i < 15; i++)
    recv a;

send b = 10;
send d = 8;
```

Task 2

```
recv b;
recv c;
```

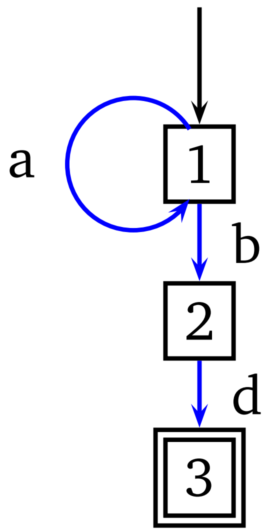


Another Example

Task 1

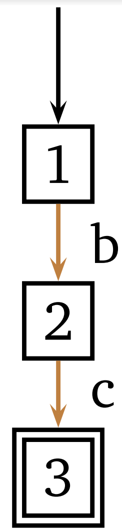
for (*int* i = *0*; i < *15*; i++)
    recv a;

send b = *10*;
send d = *8*;

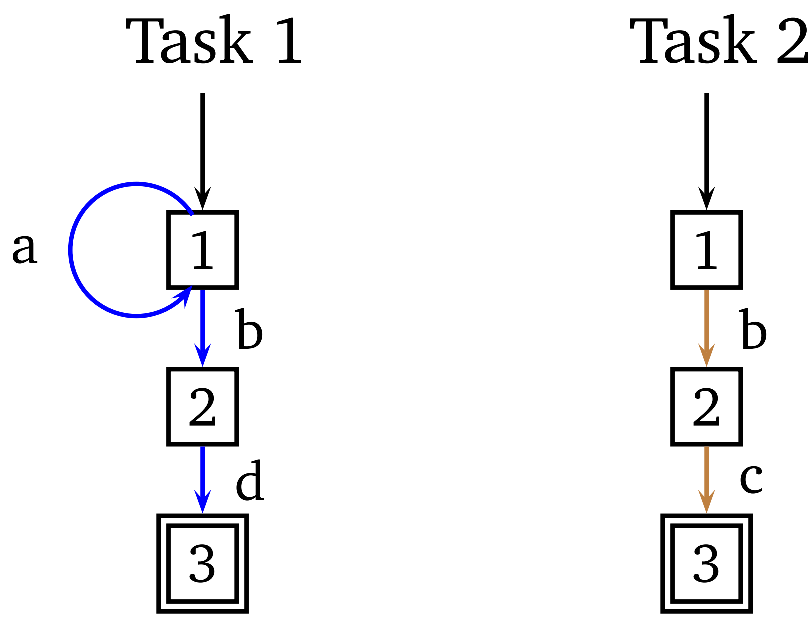


Task 2

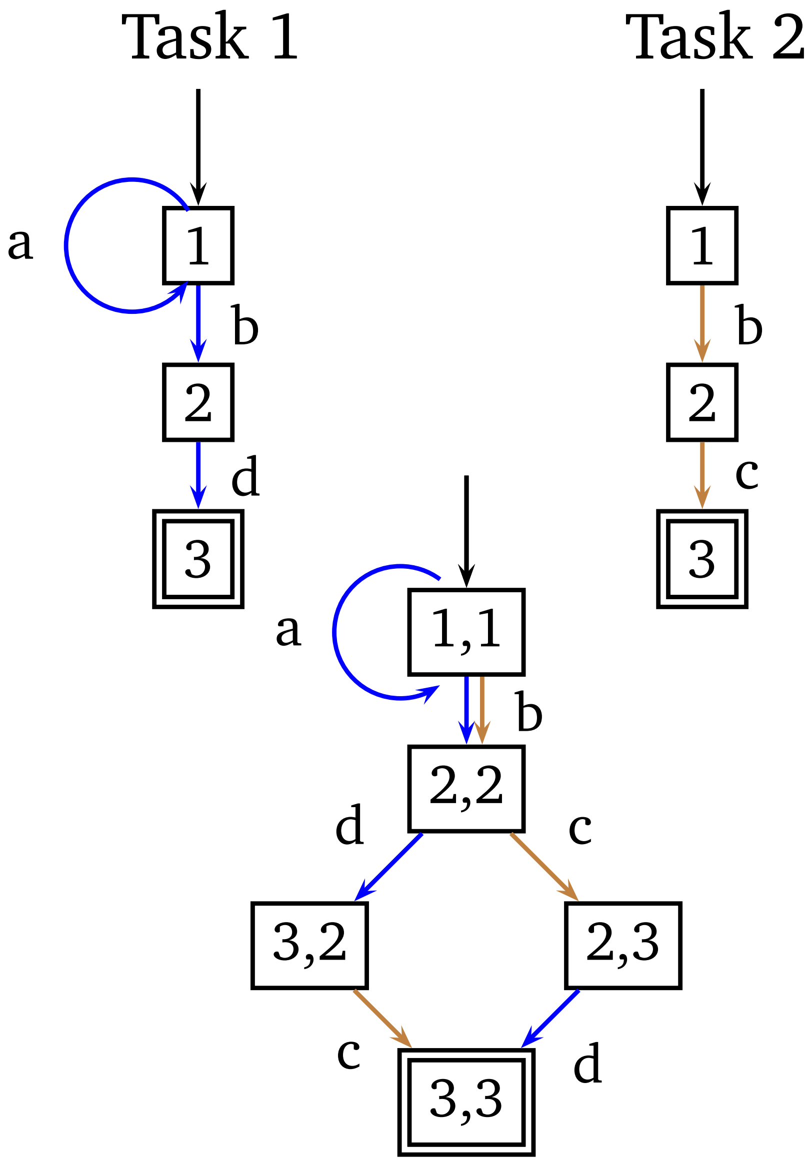
recv b;
recv c;



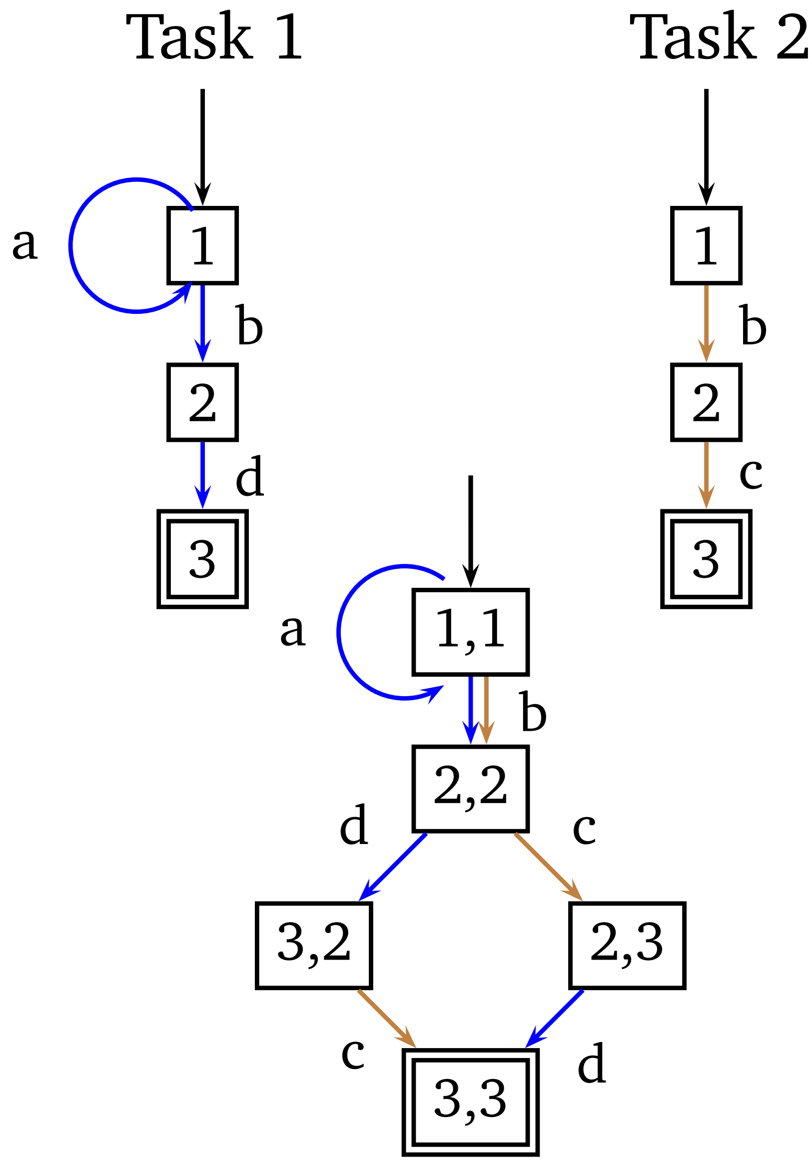
Automata Composition



Automata Composition



Automata Composition



- c and d cannot share buffers
- False positive: a and b cannot share buffers

Grouping Channels

- a and b can share buffers
- b and c can share buffers
- a and c cannot share buffers

Two possibilities

- {a,b} {c}
- {b,c} {a}

Grouping Channels

- a and b can share buffers
- b and c can share buffers
- a and c cannot share buffers

Two possibilities

- {a,b} {c}
- {b,c} {a}

Suppose

- a: 2MB
- b: 8MB
- c: 8MB

Grouping Channels

Greedy, first-fit method

- b: 8MB
- c: 8MB
- a: 2MB

Grouping Channels

Greedy, first-fit method

- b: 8MB
- c: 8MB
- a: 2MB

{b}

Grouping Channels

Greedy, first-fit method

- b: 8MB
- c: 8MB
- a: 2MB

{b}
{b,c}

Grouping Channels

Greedy, first-fit method

- b: 8MB
- c: 8MB
- a: 2MB

{b}
{b,c}
{b,c} {a}

Results

Example	Lines	Channels	Tasks	Bytes Saved	Buffer Reduction	Runtime
Source-Sink	35	2	11	4	50 %	0.1 s
Pipeline	35	5	9	16388	25	0.1
Bitonic Sort	35	5	13	12	60	0.1
Prime Sieve	40	5	16	12	60	0.5
Berkeley	40	3	11	4	33.33	0.6
FIR Filter	110	28	28	52	46.43	13.8
Framebuffer	185	11	16	28	0.002	1.3
FFT	230	14	15	344068	50	0.6
JPEG Decoder	1020	7	15	772	50.13	1.8

Related Work

- Significant work in sequential programs
  [Greef et al., ASAP ’97]
- Synchronous data flow
  [Murthy et al., ACM TODAES ’04]
- Constrain the schedule to save memory
  [Chrobak et al., ICALP ’01]

Conclusions

- Reduces memory without affecting the run-time schedule
- Can be applied to the Cell compiler
  - Can save 344 kB of PPE's memory for FFT
- Future work
  - More modular techniques
  - Reduce memory in k-place buffered models