Yale ENGINEERING

Syntax-directed translation

Rajit Manohar

AVLSI

A direct path from CHP to gates

- Goal: to provide a direct path from CHP to gates
- “Syntax directed”
  - Translation uses the syntax of the CHP program to generate the circuit
  - Uses structural induction
    - Induction on the *structure* of the program
    - Translations for
      - Base case: assignment, communication, skip, expression evaluation
      - Induction: selections, loops, sequential composition, parallel composition
- History
  - 1980s : multiple approaches developed
  - 1991 : Tangram language / Haste @ Handshake Solutions (Philips Research)
  - 1998 : Balsa, based on Tangram with extensions (U. Manchester)

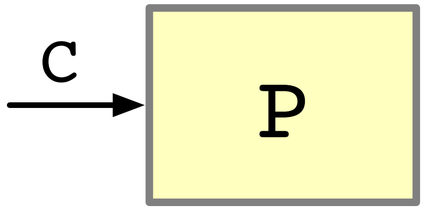
Key idea

- Use a communication *channel* to select a program for execution
- Given a program “P”, we will implement the following

```
*[          // infinite loop
  [#C];     // wait for pending
  P;        // execute P
  C?        // finish C
 ]
```

- We execute “P” by simply executing

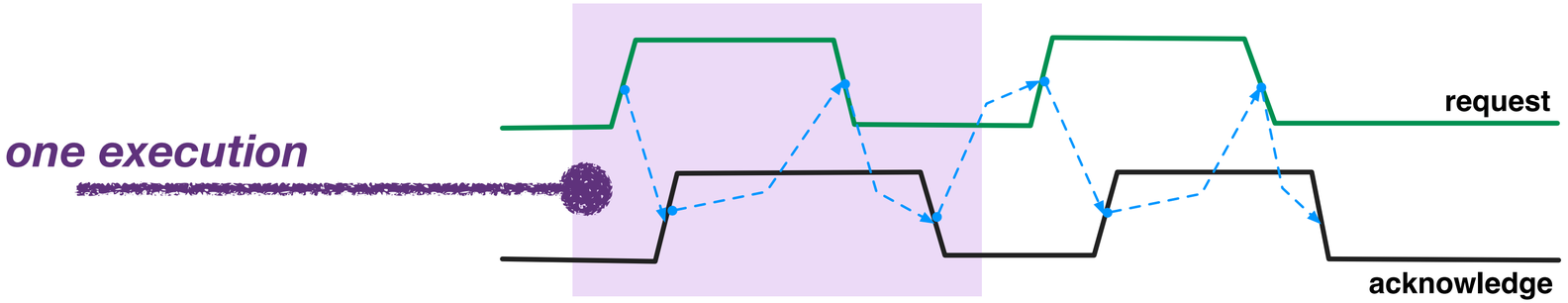
```
C!
```

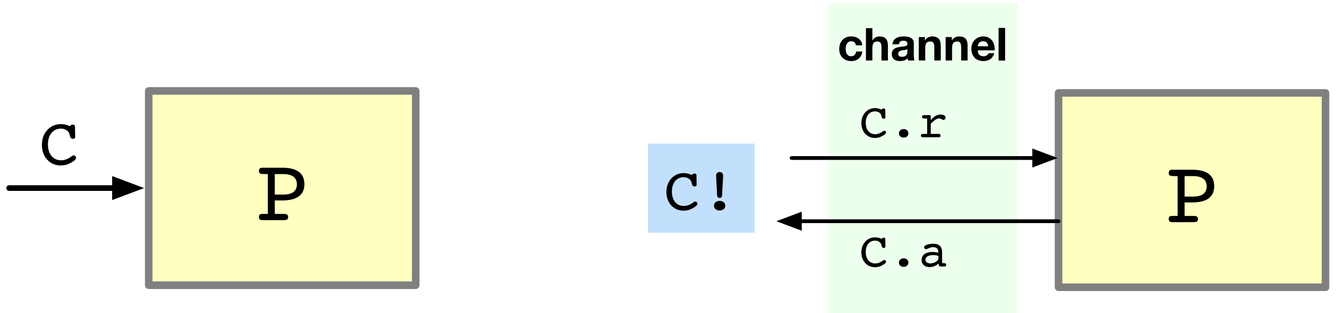


- This is sometimes called “process decomposition” or “process call”

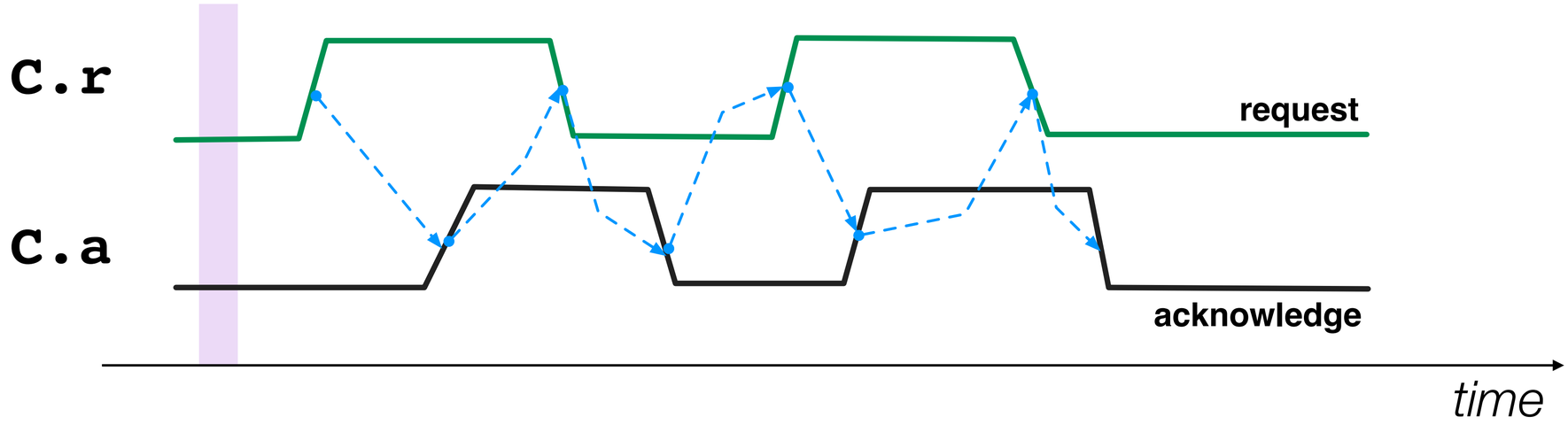
Wire implementation of channels

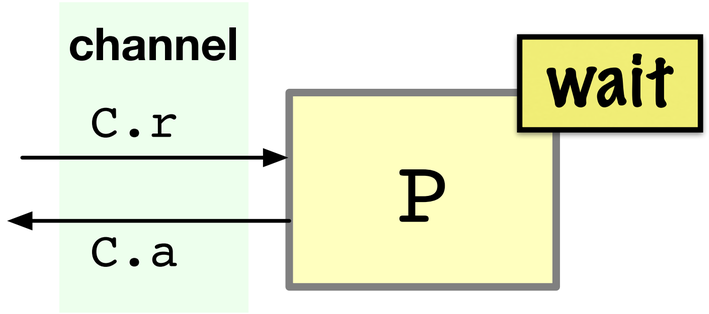
- Channel “C” that *controls* the execution of a program



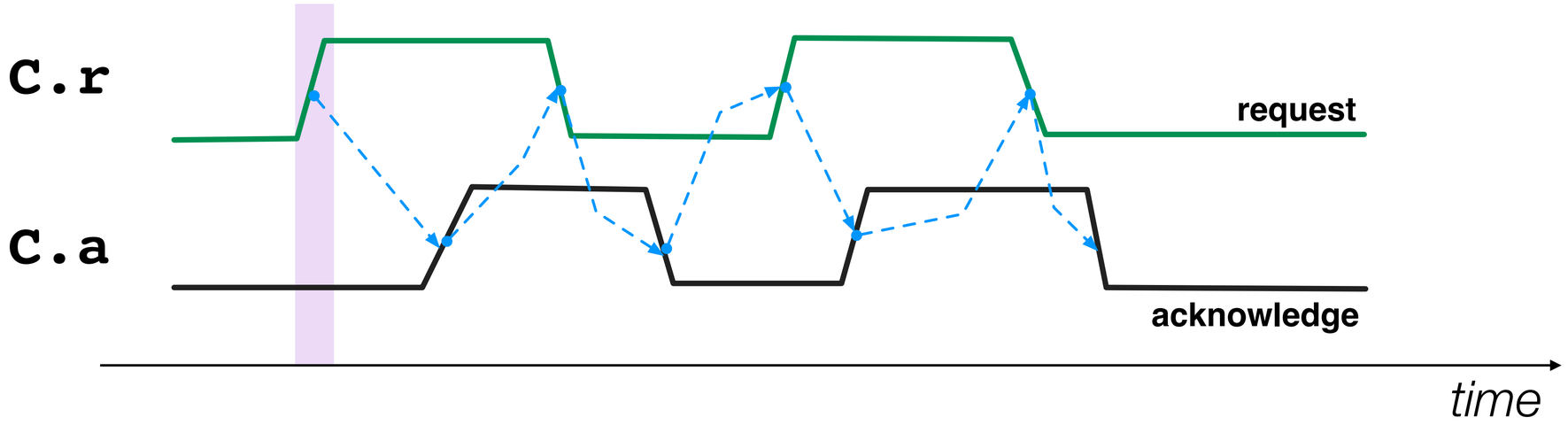


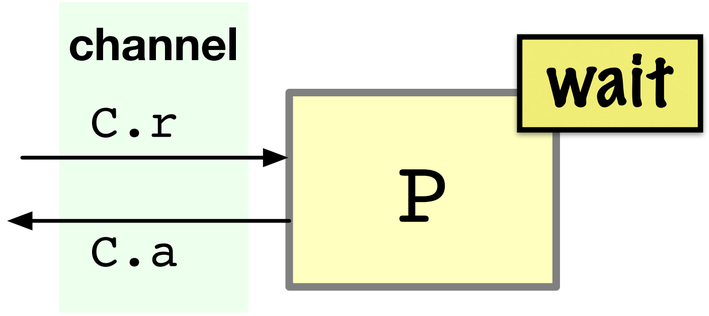
One execution: idle (waiting) state



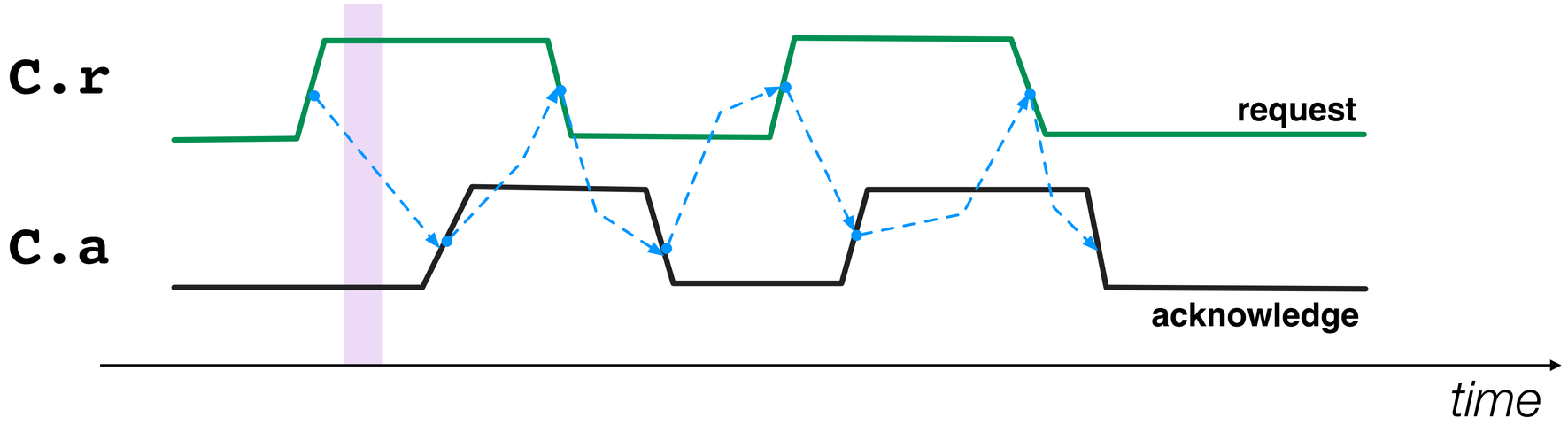


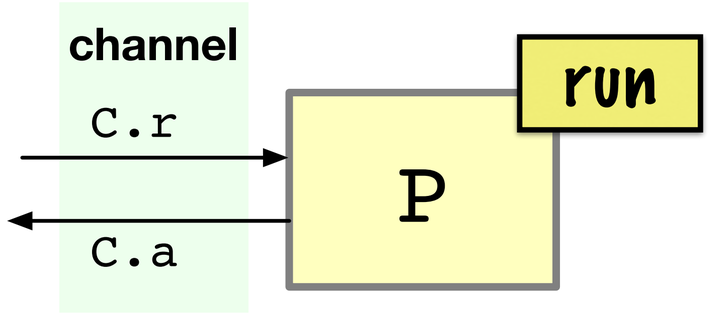
One execution: request execution



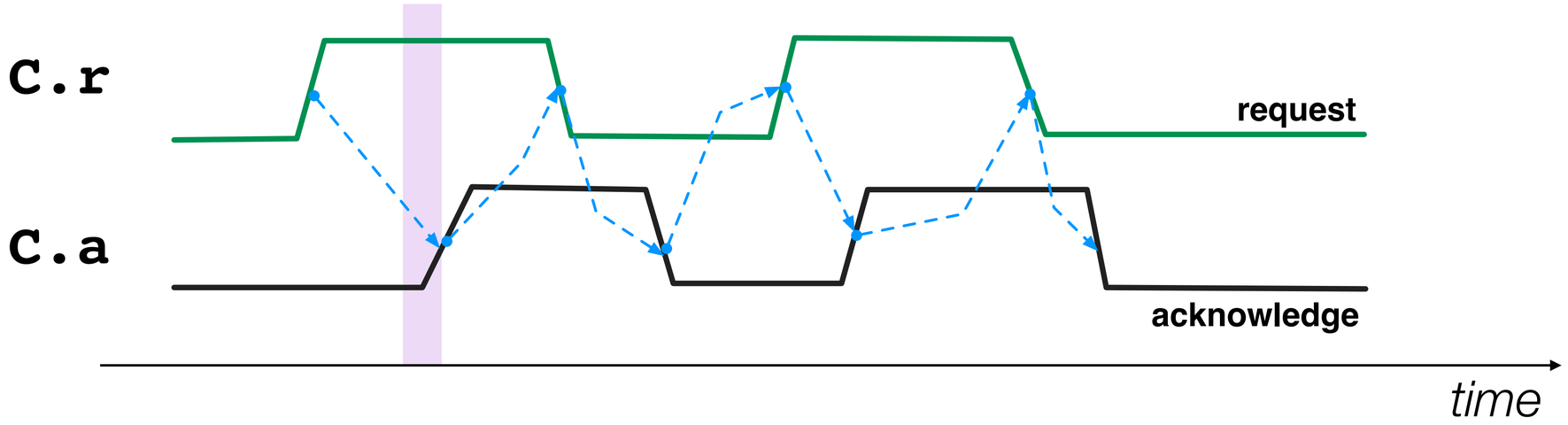


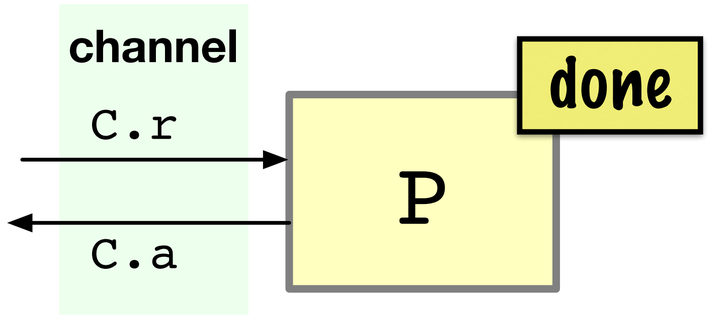
One execution: running



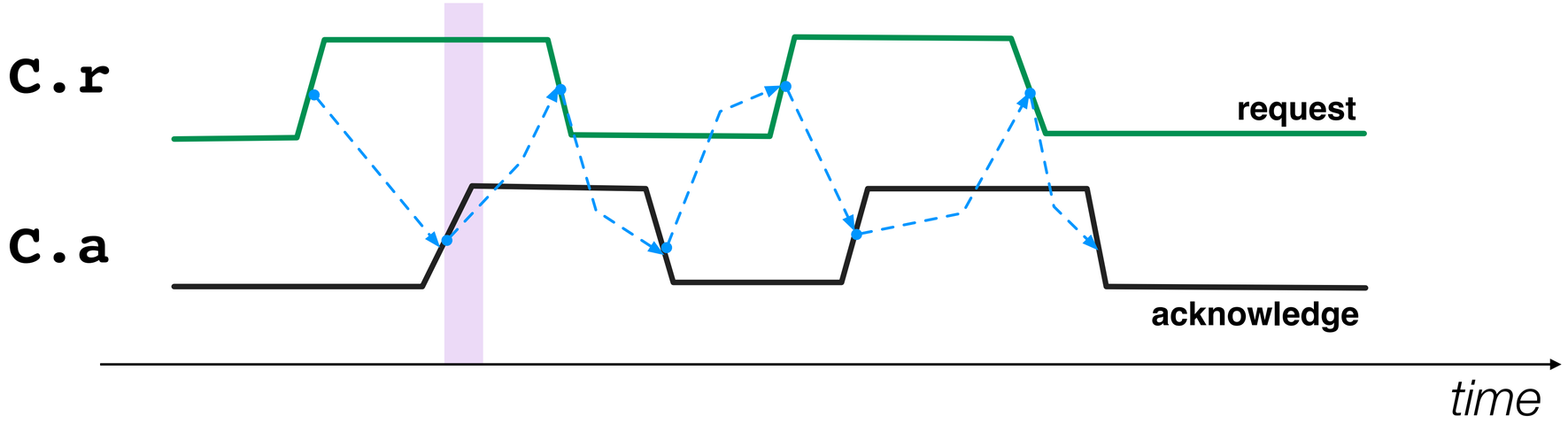


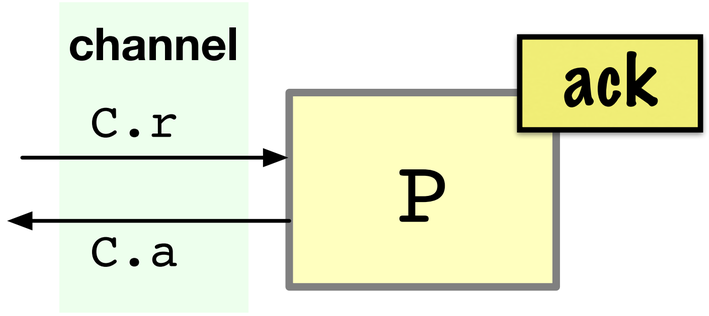
One execution: done



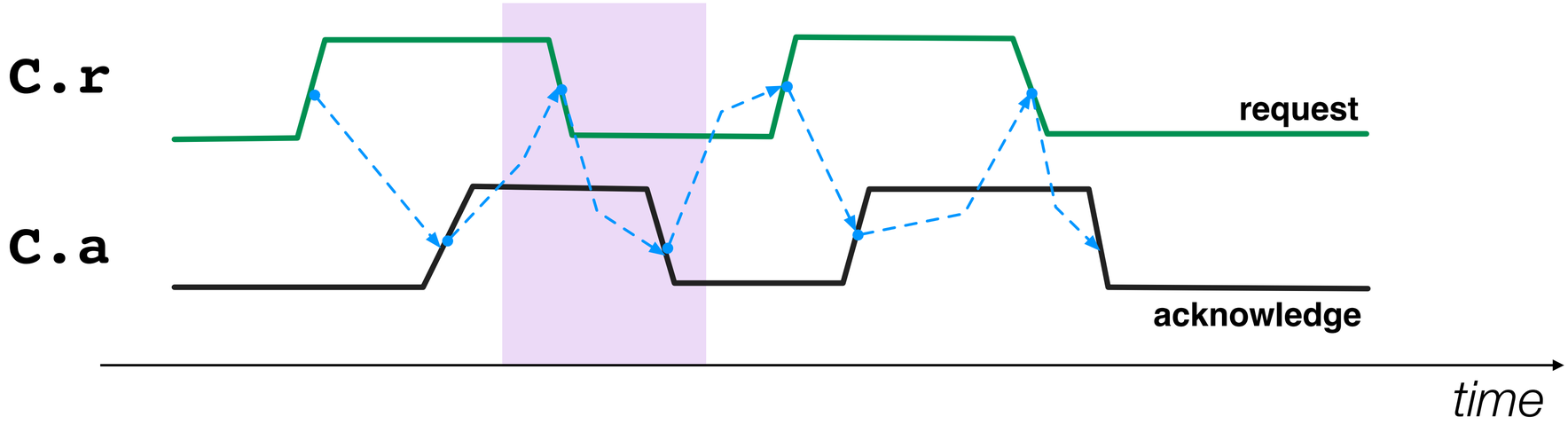


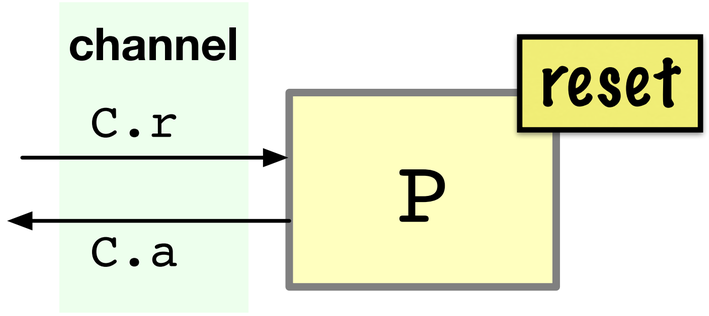
One execution: respond to requester



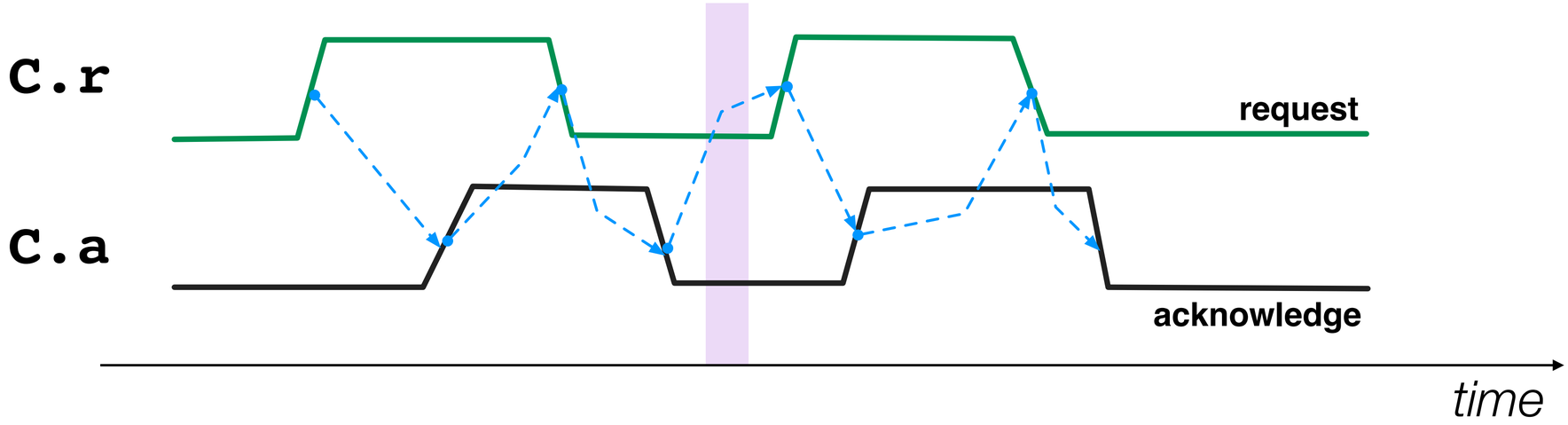


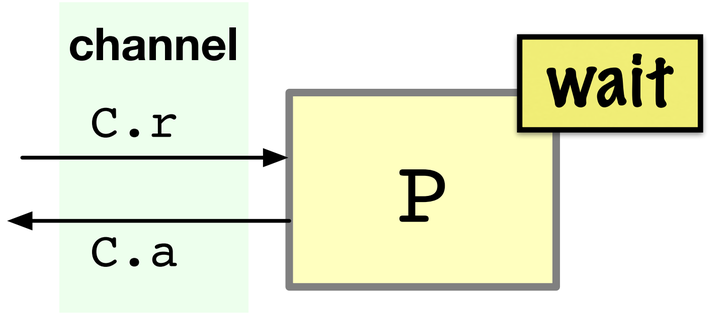
One execution: reset phase





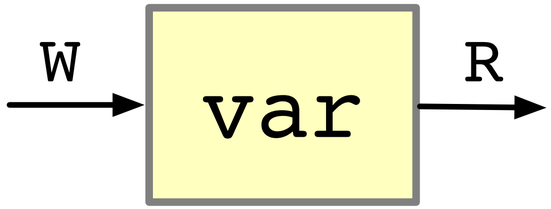
One execution: reset phase





Variables

- Two operations
  - Write a value to the variable



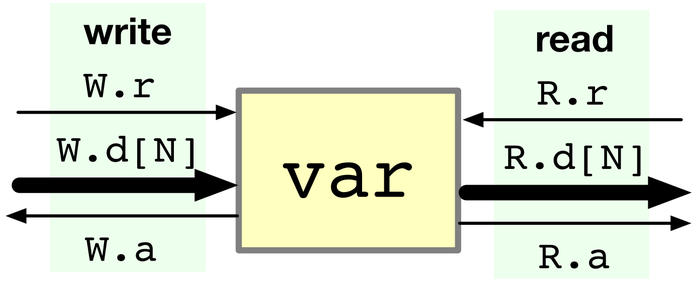

```
W!value
```

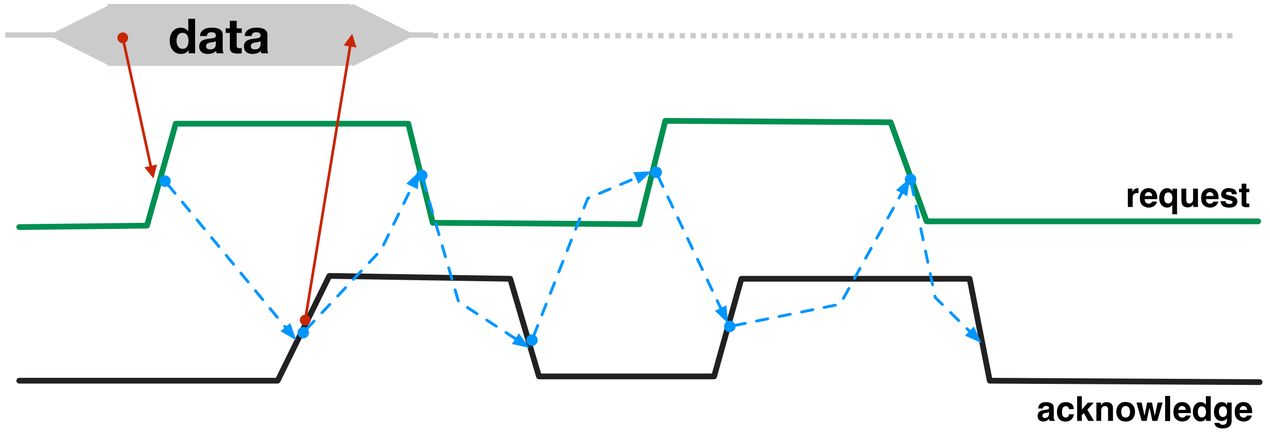
  - Read the current value of the variable

```
R?x
```

- The variable itself is “passive”
  - It waits for the environment to either write or read its value

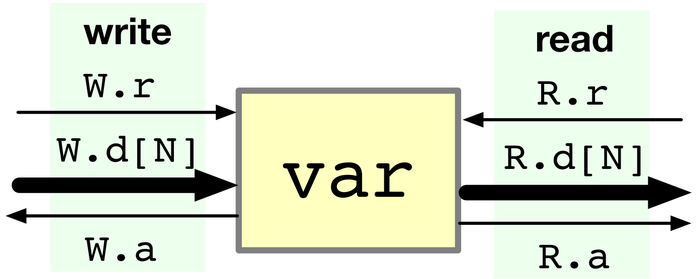
Writing and reading a variable

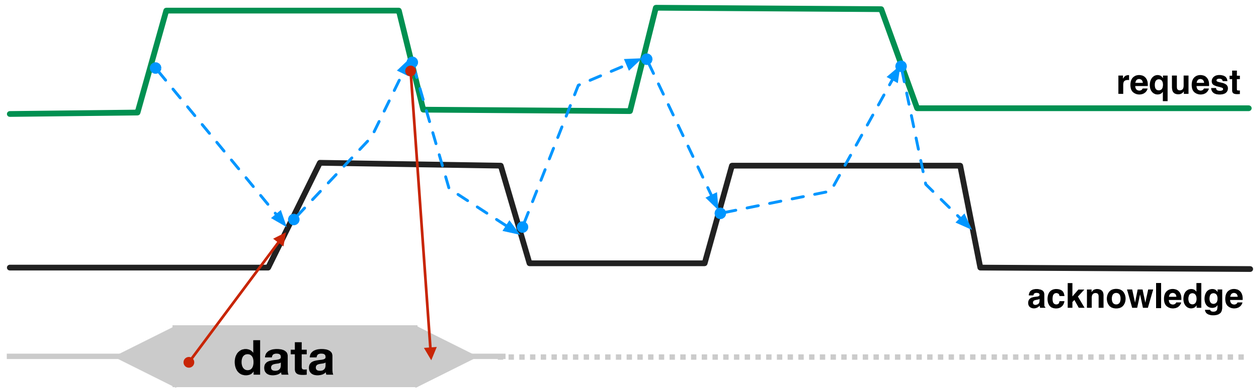




Write

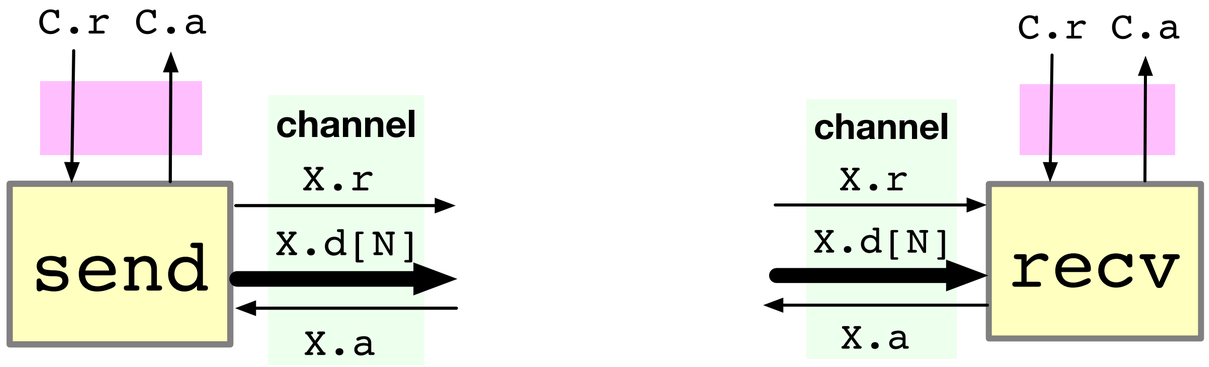
Writing and reading a variable

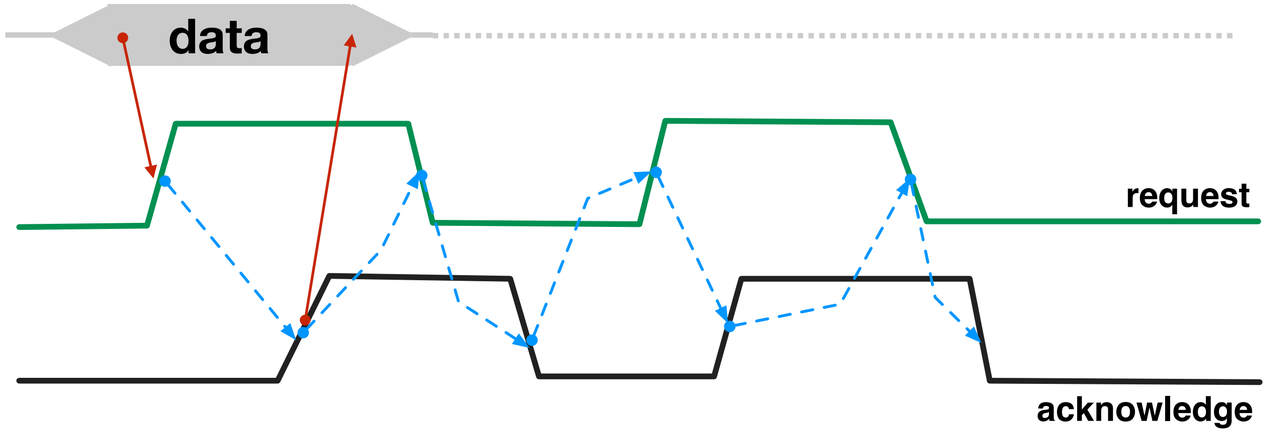




Read

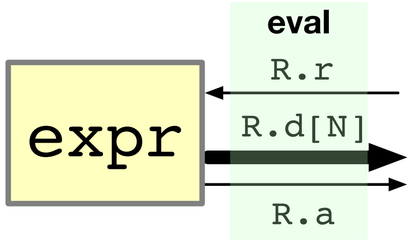
Sending and receiving on a channel



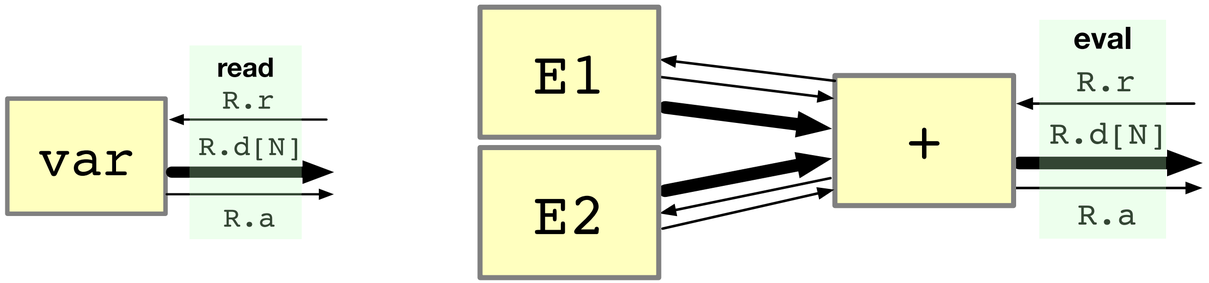


Send/recv

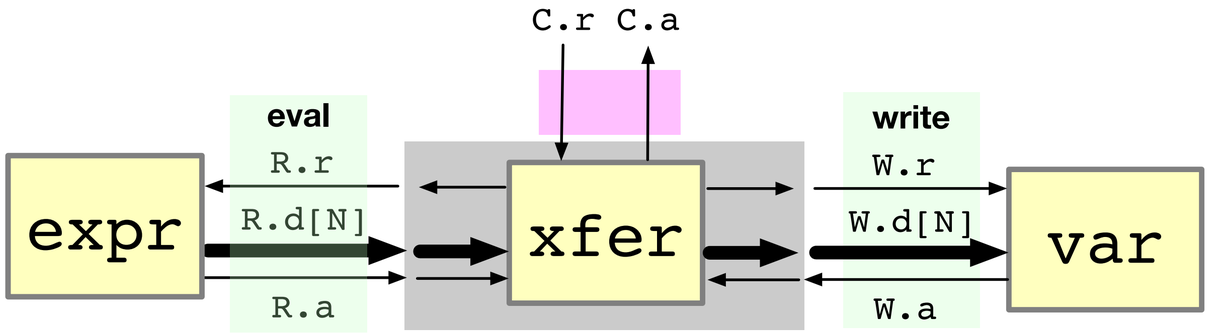
Expression evaluation



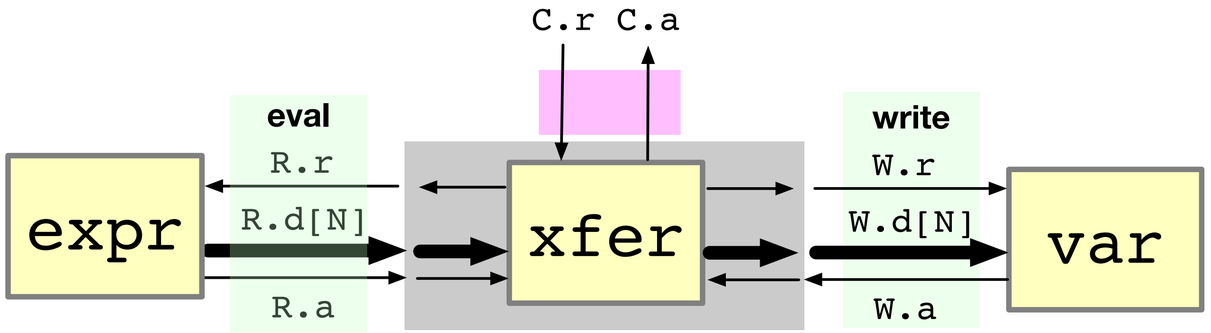
- Example of expression de-composition

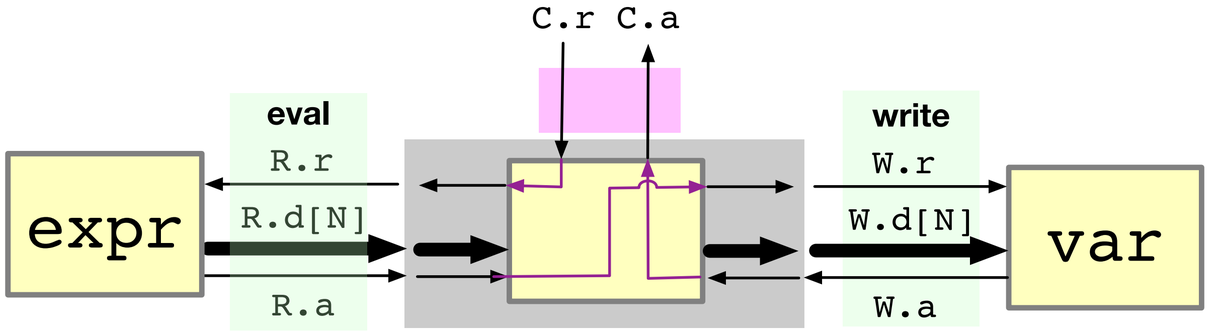


Assignment

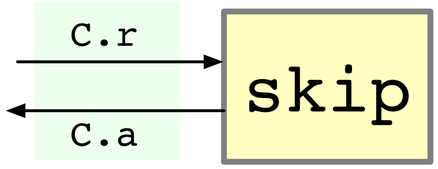


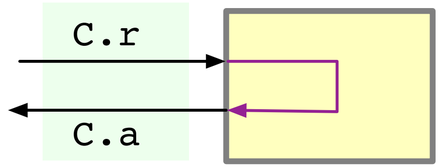
Assignment


C.r C.a
eval
R.r
R.d[N]
R.a
expr
xfer
write
W.r
W.d[N]
W.a
var

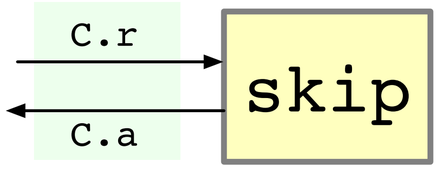

C.r C.a
eval
R.r
R.d[N]
R.a
expr
write
W.r
W.d[N]
W.a
var

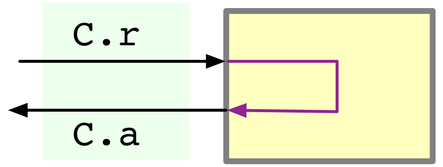
Building blocks

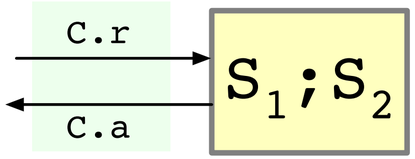


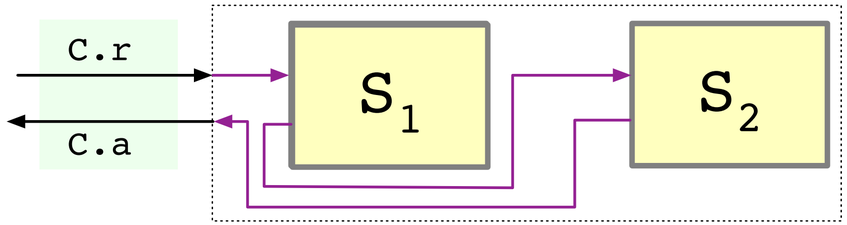


Building blocks

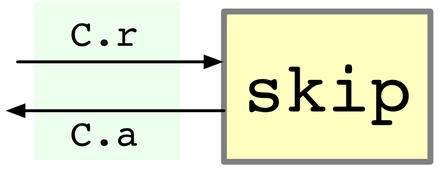


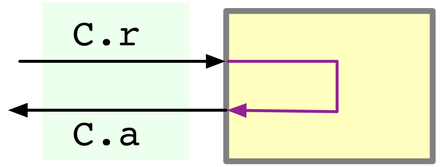


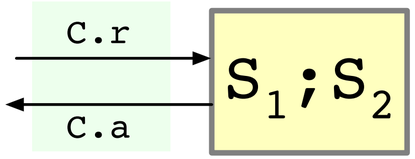


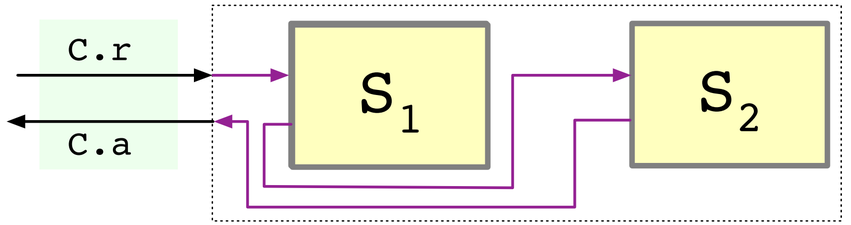


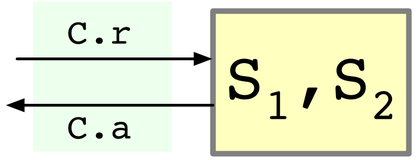
Building blocks

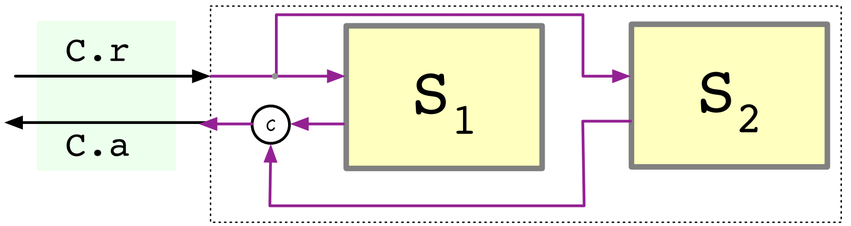




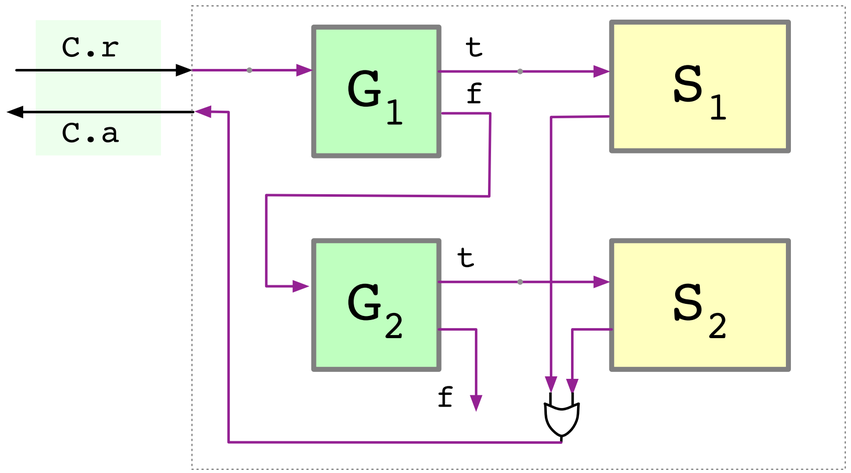






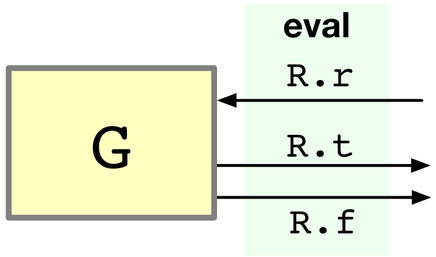


Selections and loops

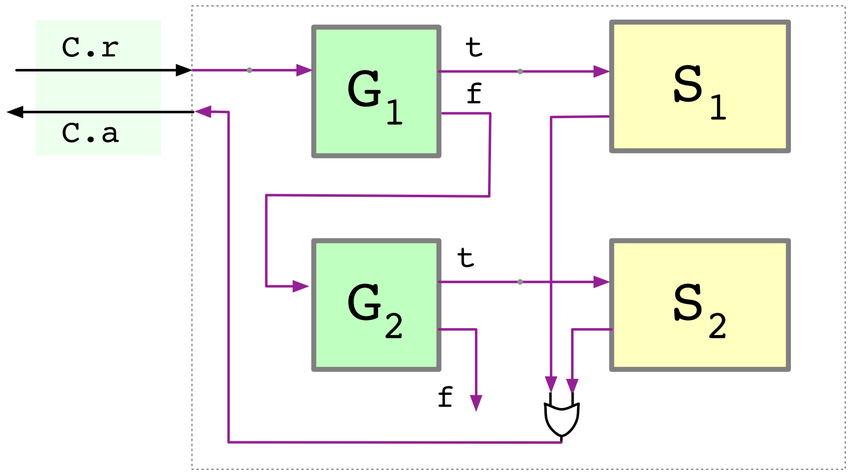



```
[   G1 -> S1
[ ] G2 -> S2
]
```

Selection

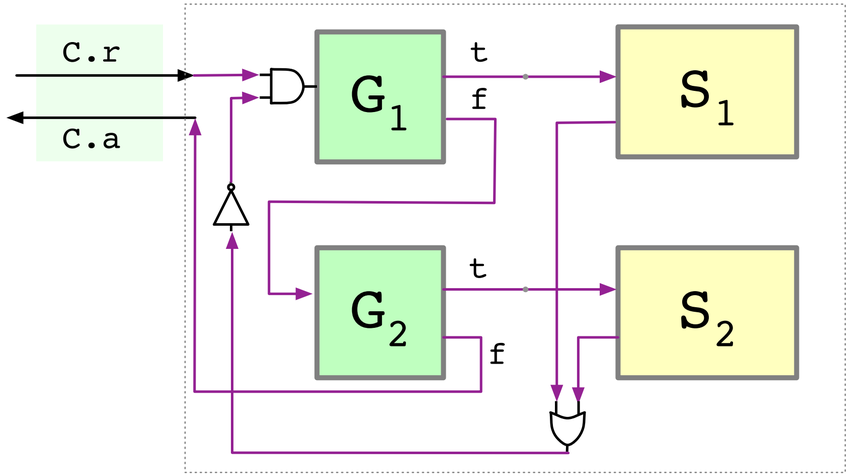


Selections and loops




```
[   G1 -> S1
[] G2 -> S2
]
```

Selection




```
*[   G1 -> S1
 [] G2 -> S2
 ]
```

Loop