Syntax-directed translation

Rajit Manohar

Asynchronous VLSI and Architecture (AVLSI) Group
Computer Systems Lab, Yale University

`https://csl.yale.edu/~rajit/`
`https://avlsi.csl.yale.edu/act`

Yale

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
  - 1988 : paper at Conference on Advanced Research in VLSI (Caltech)
  - 1991 : Tangram language / Haste @ Handshake Solutions (Philips Research)
  - 1998 : Balsa, based on Tangram with extensions (U. Manchester)

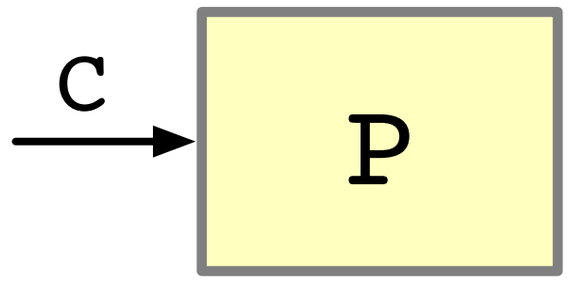
Key idea

- 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 C
  P;        // execute P
  C?        // finish C
]
```

- We execute “P” by simply executing

```
C!
```



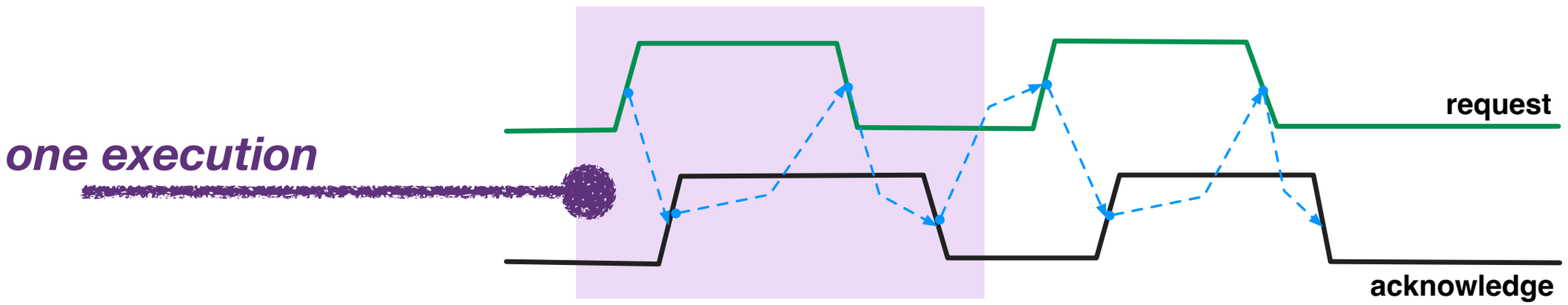
- This is sometimes called “process decomposition” or “process call”

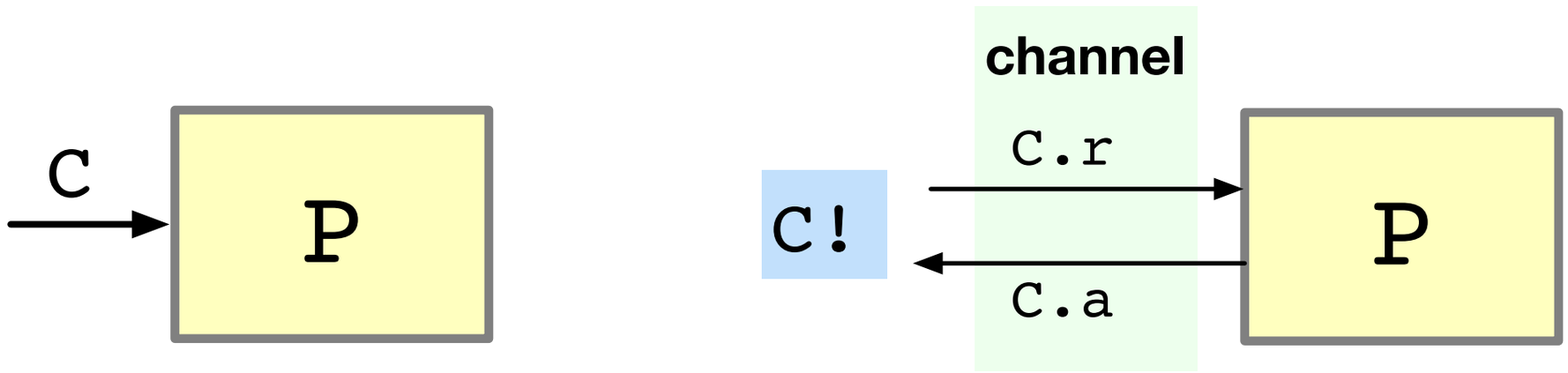
Yale

AVLSI

Wire implementation of channels

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

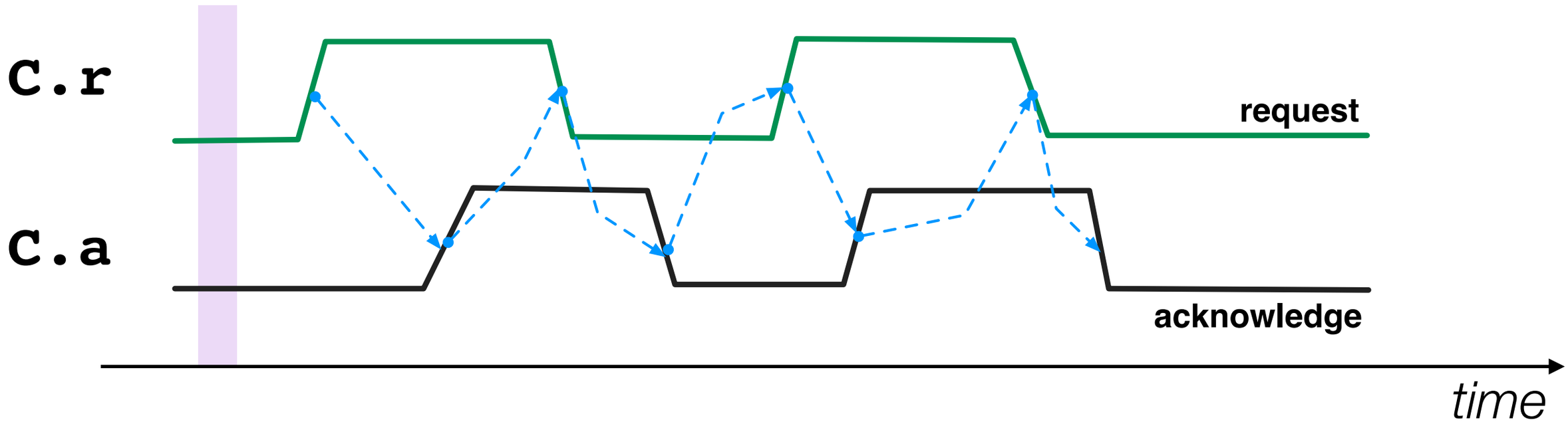


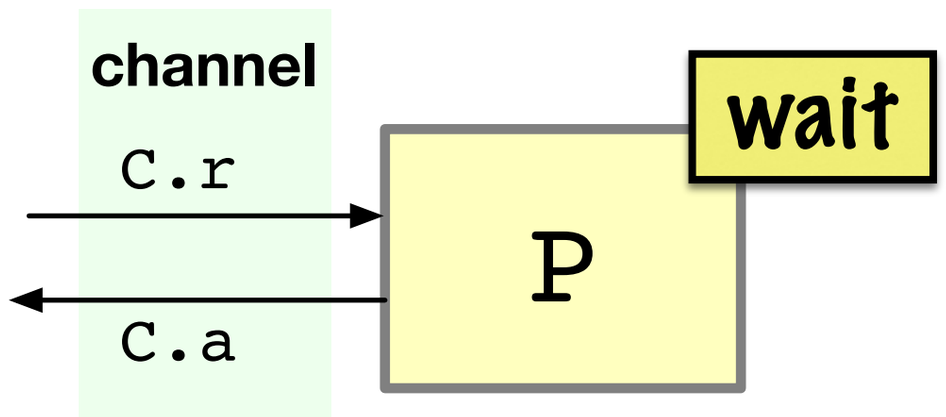


Yale

AVLSI

One execution: idle (waiting) state

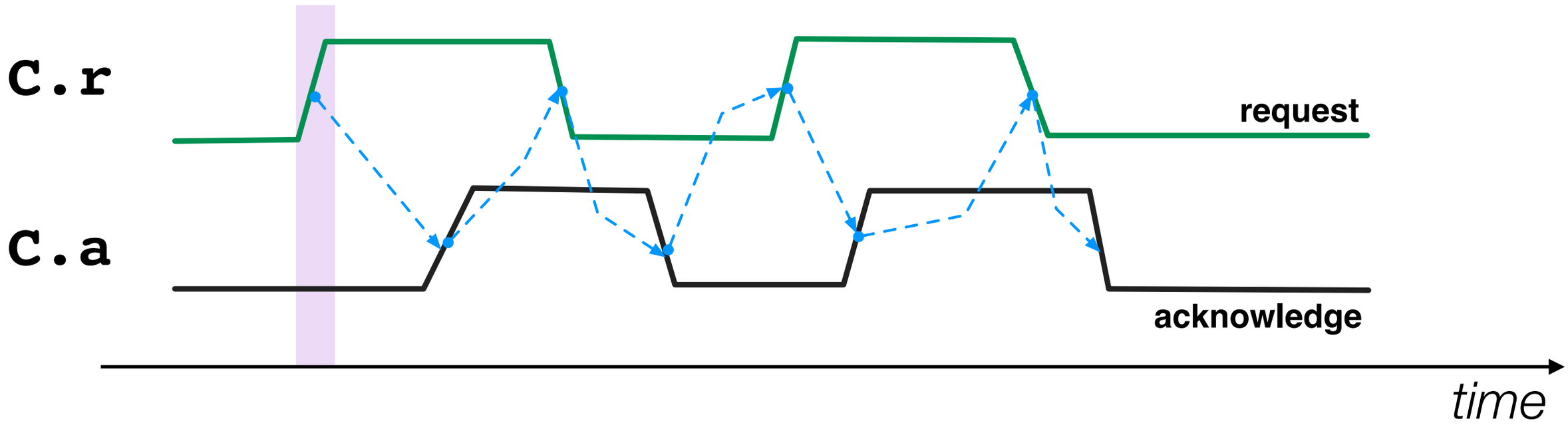


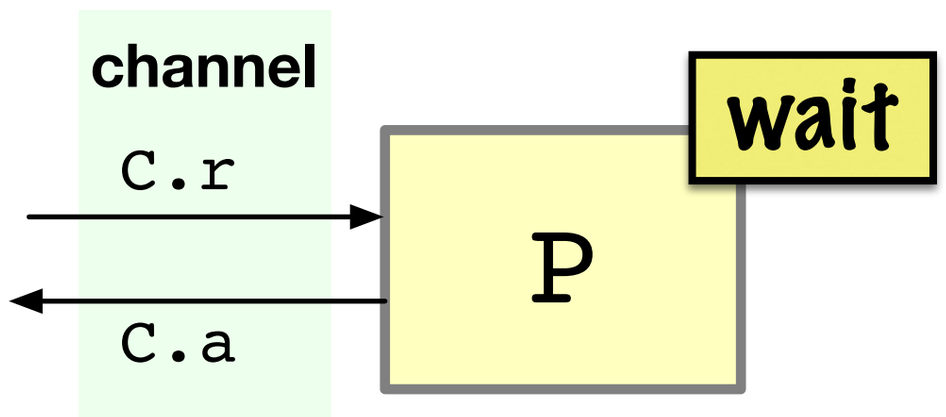


Yale

AVLSI

One execution: request execution

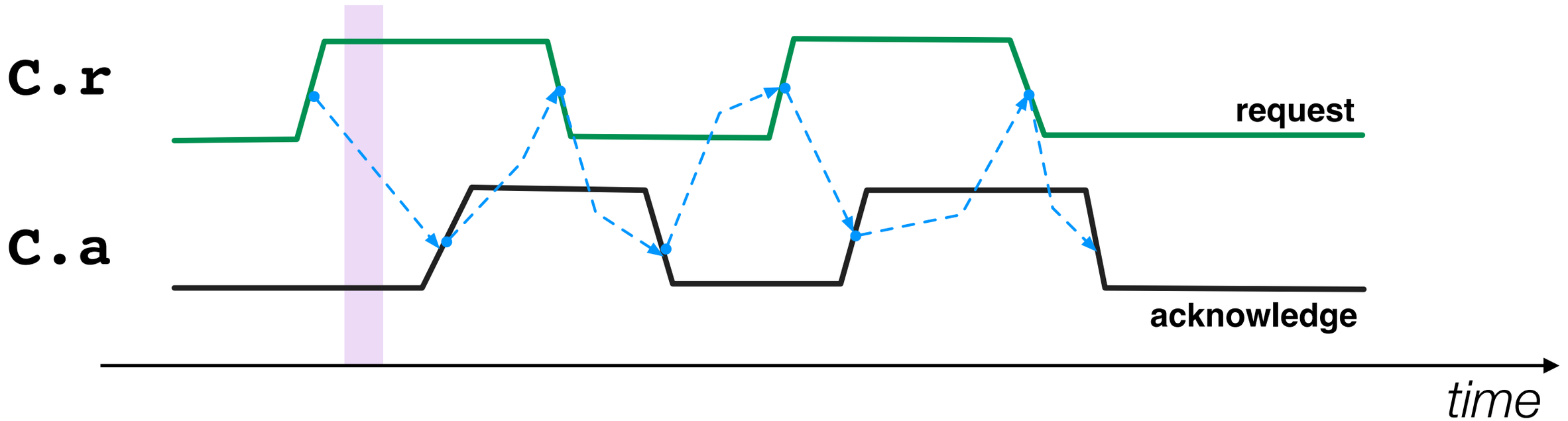


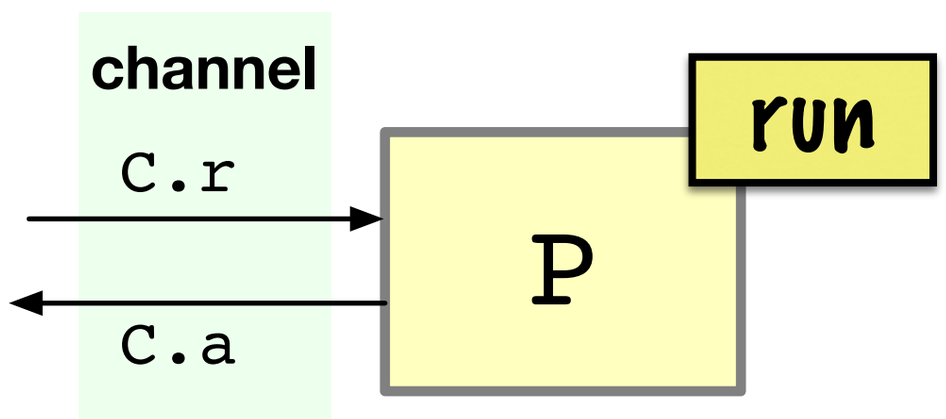


Yale

AVLSI

One execution: running

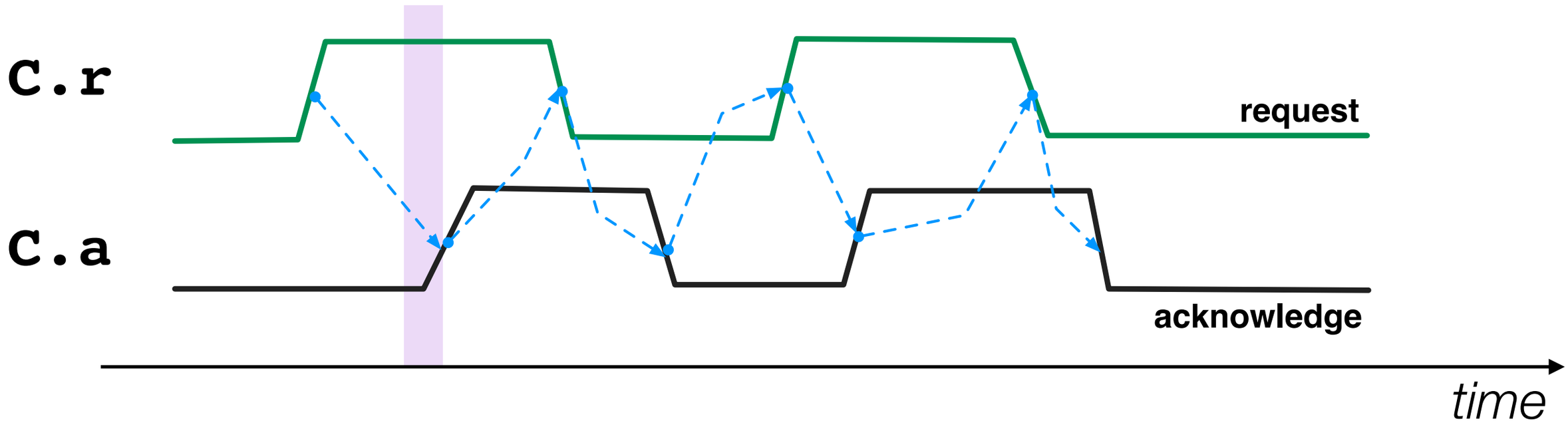


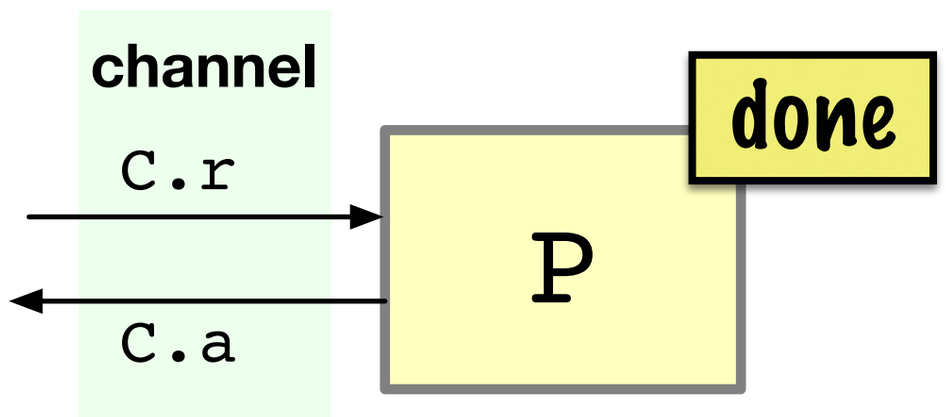


Yale

AVLSI

One execution: done

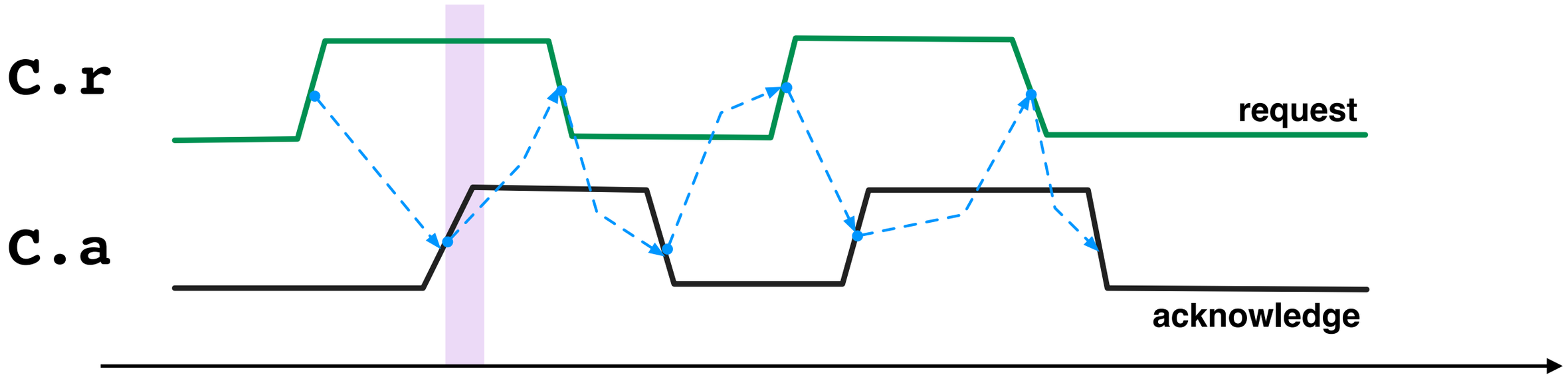


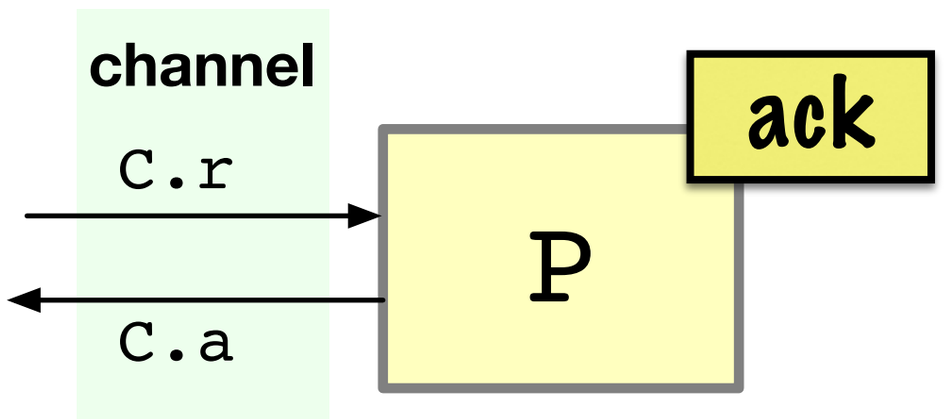


Yale

AVLSI

One execution: respond to requester

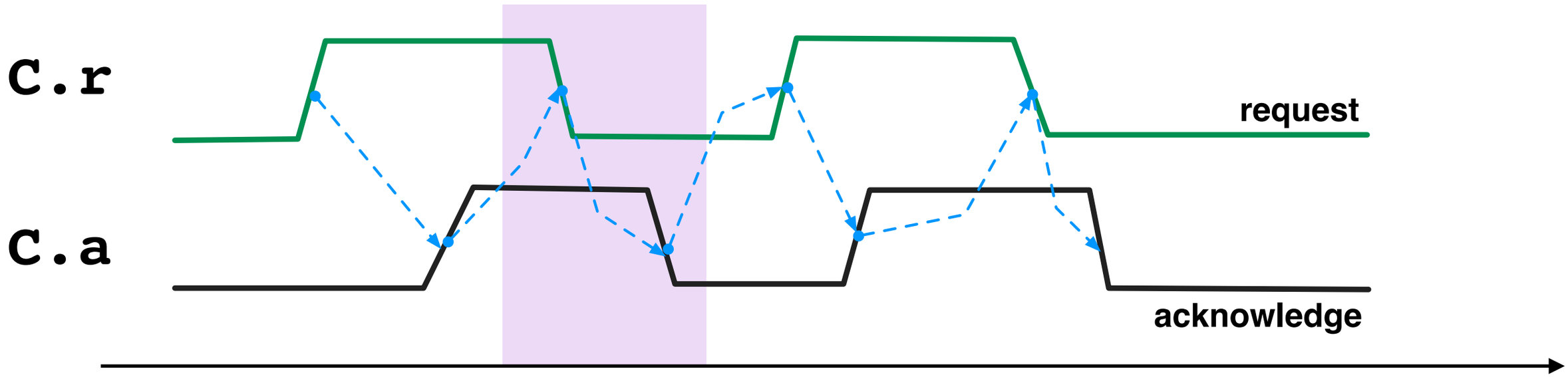


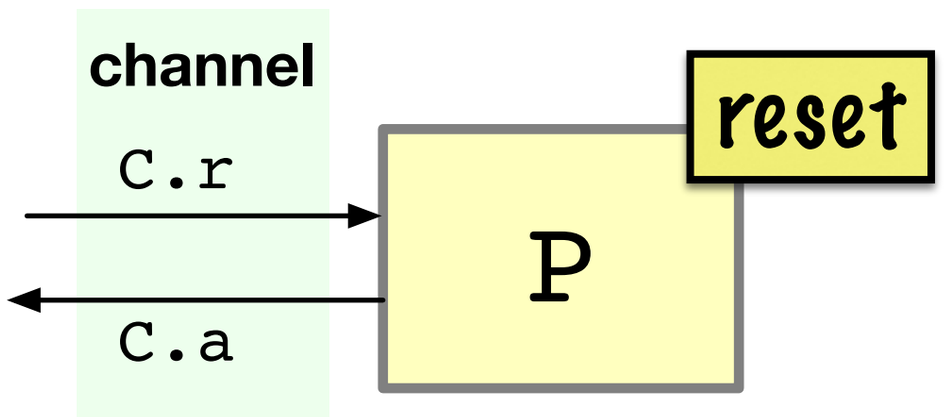


Yale

AVLSI

One execution: reset phase

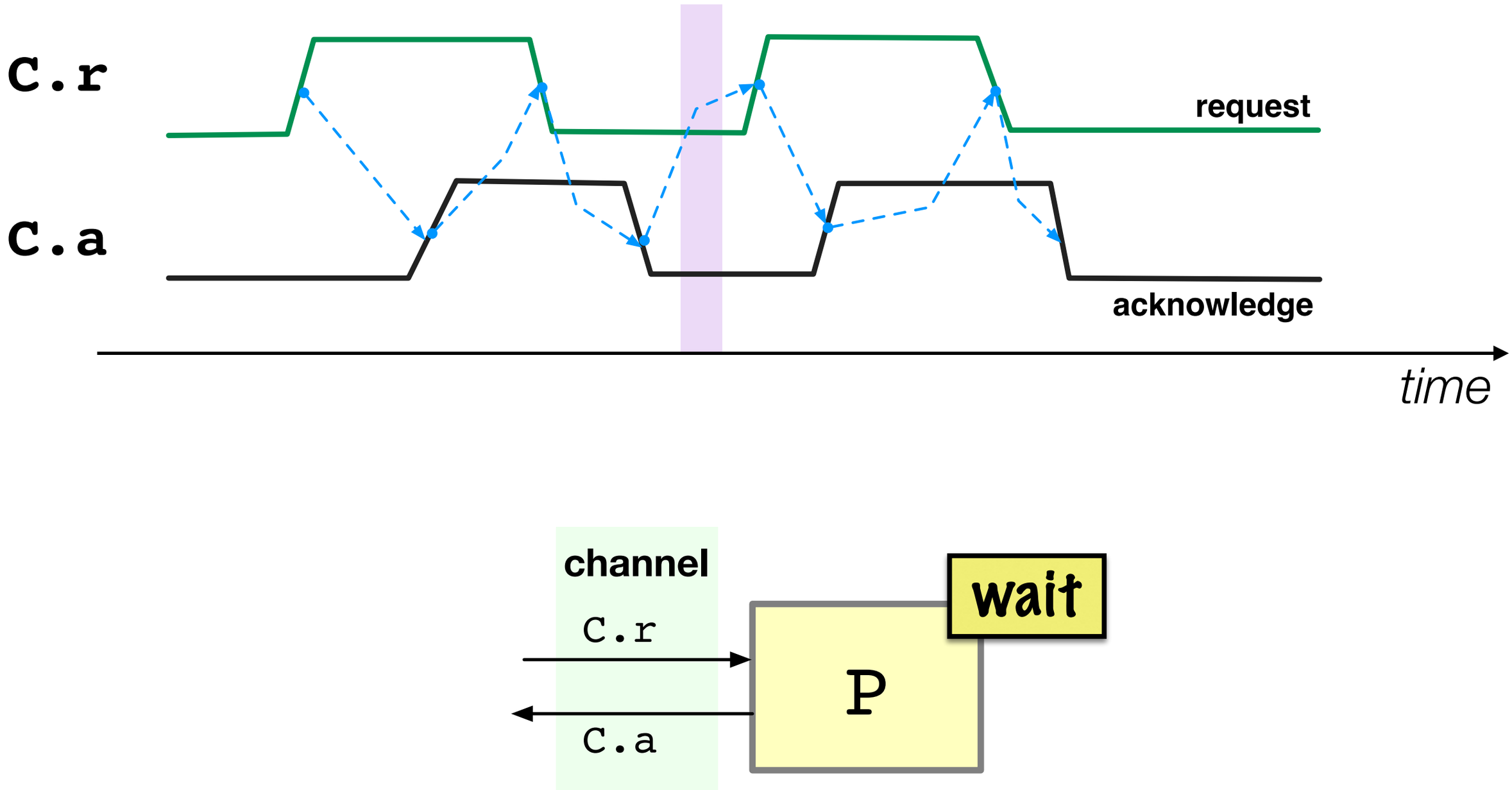




Yale

AVLSI

One execution: reset phase



Variables

- Two operations
  - ❖ Write a value to the variable

```
W!value
```

  - ❖ Read the current value of the variable

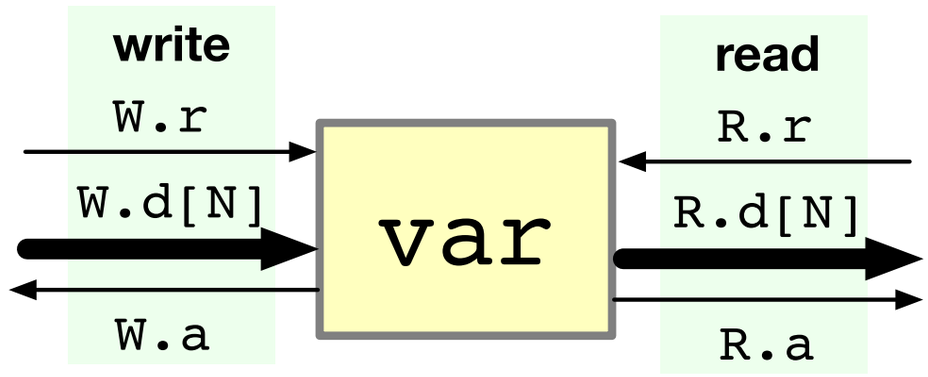
```
R?x
```

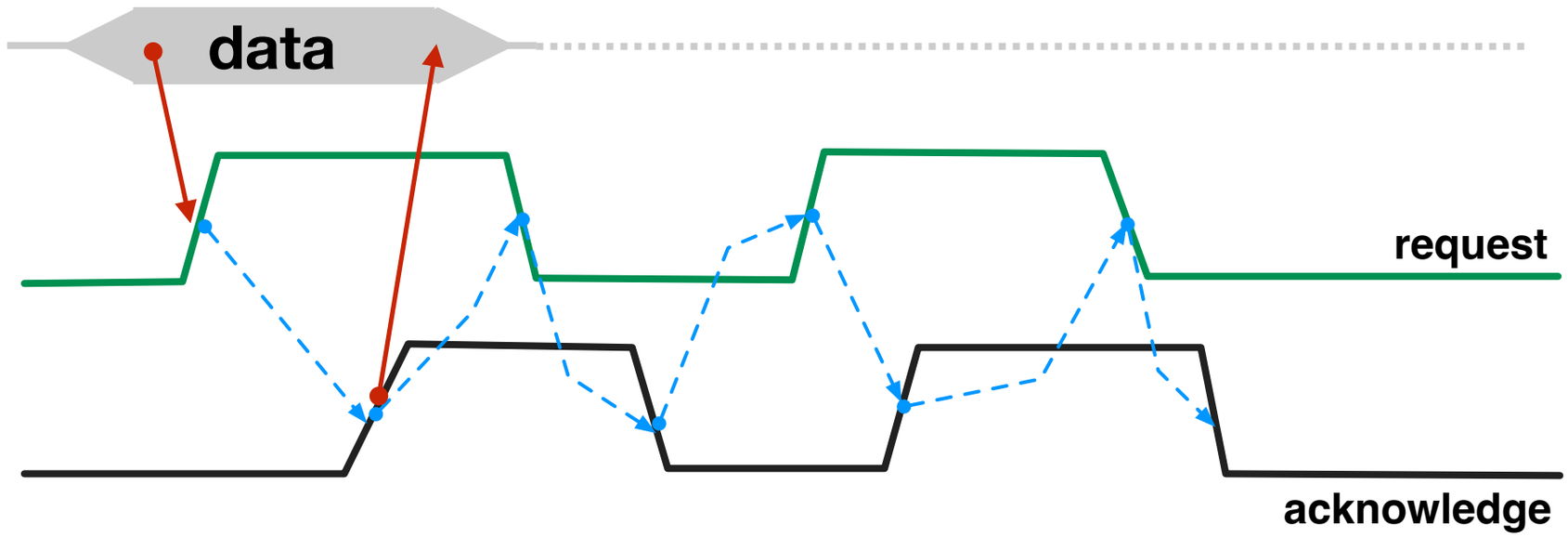
- The variable itself is “passive”
  - ❖ It waits for the environment to either write or read its value

Yale

AVLSI

Writing and reading a variable



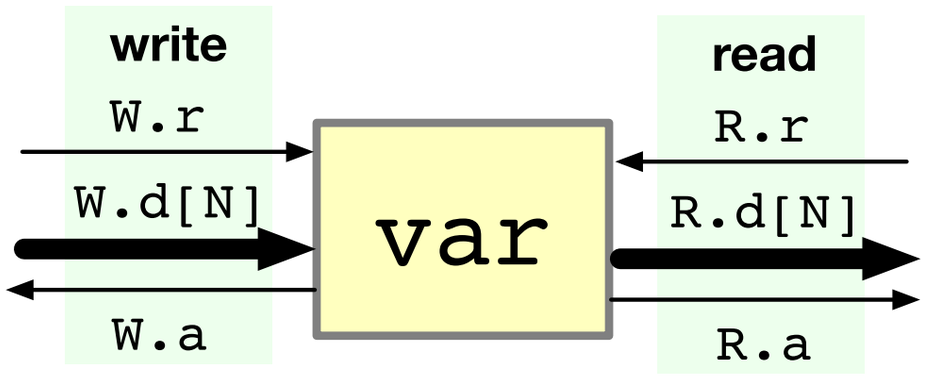


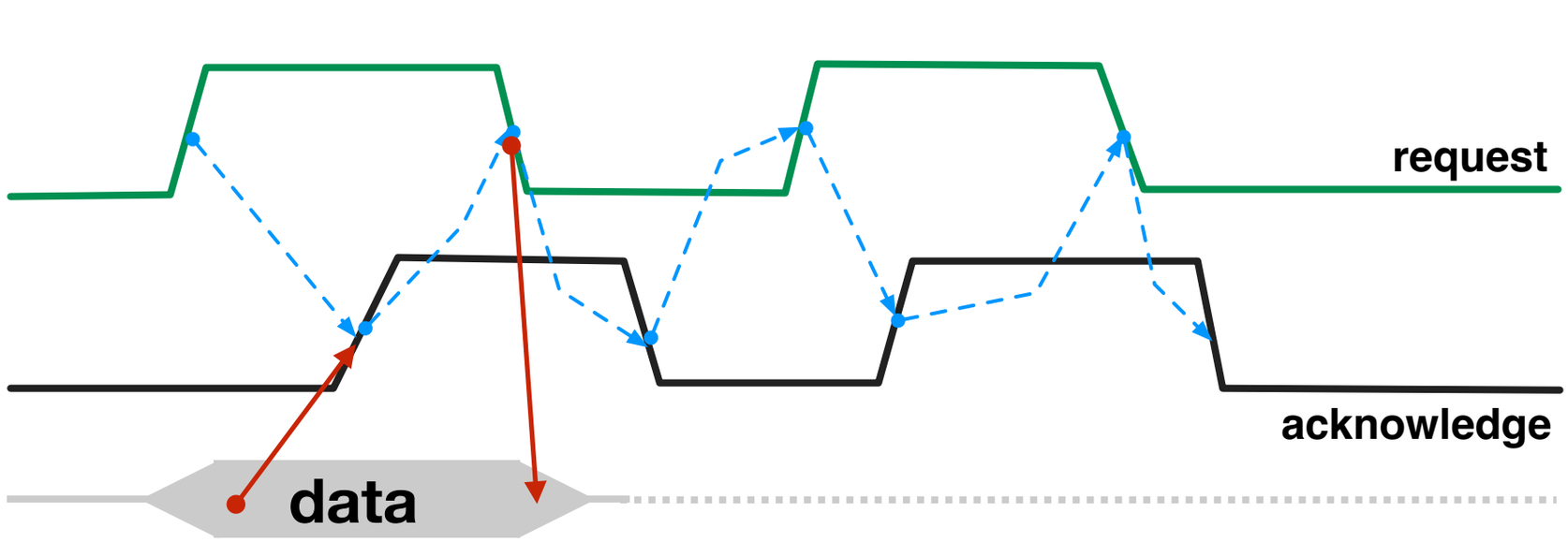
Write

Yale

AVLSI

Writing and reading a variable



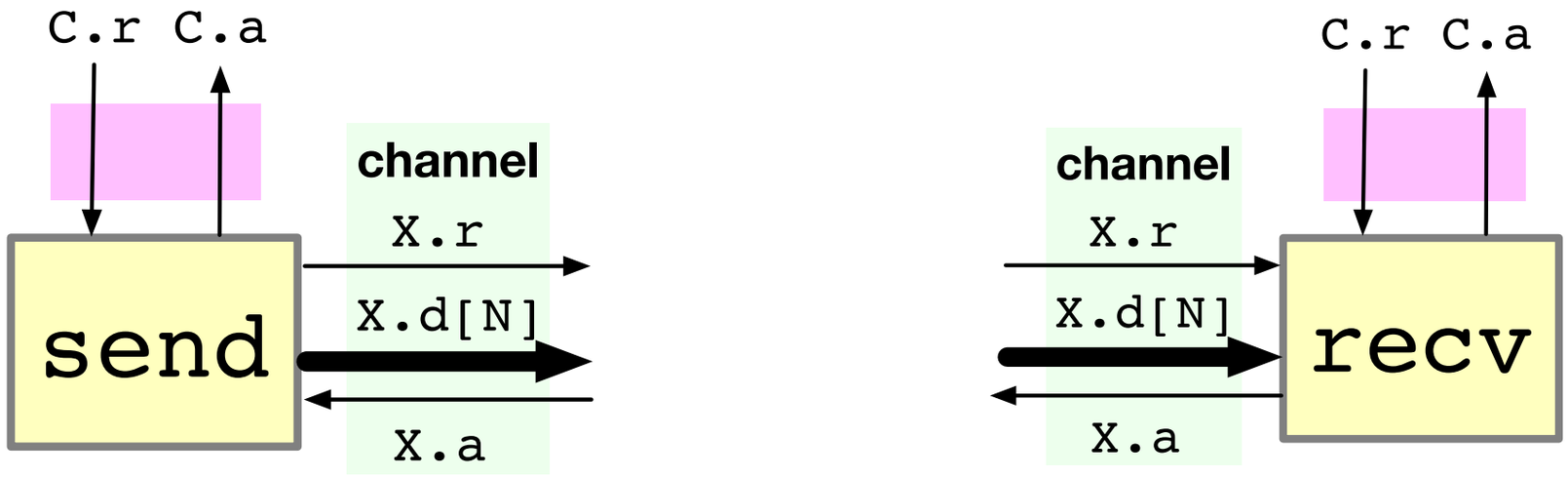


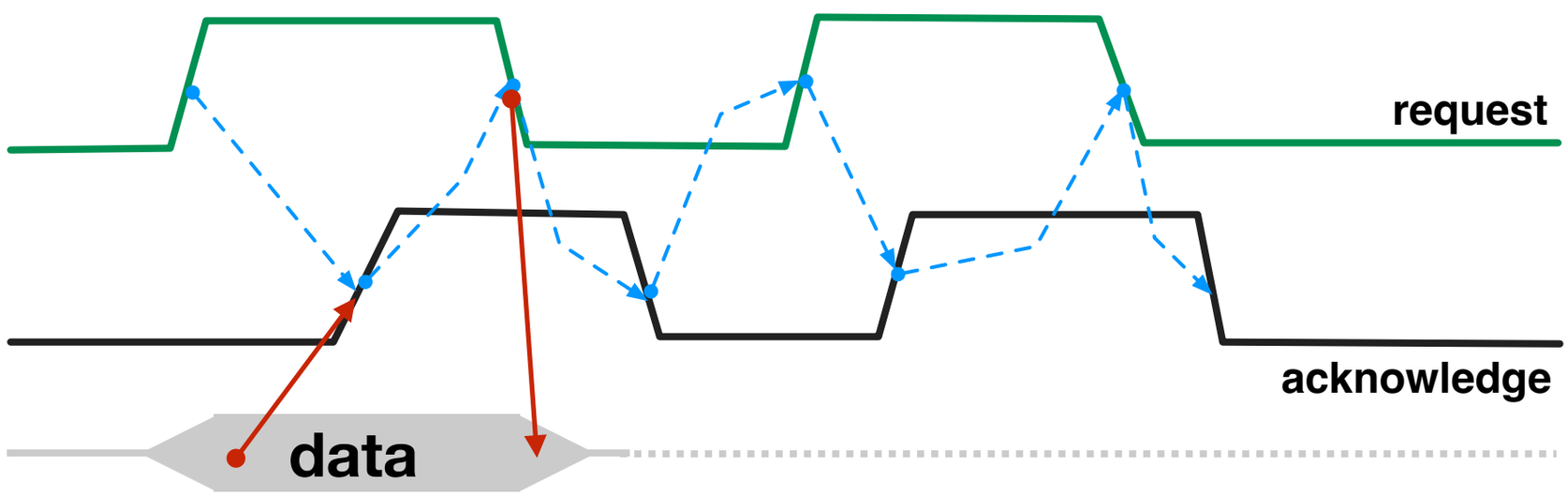
Read

Yale

AVLSI

Sending and receiving on a channel



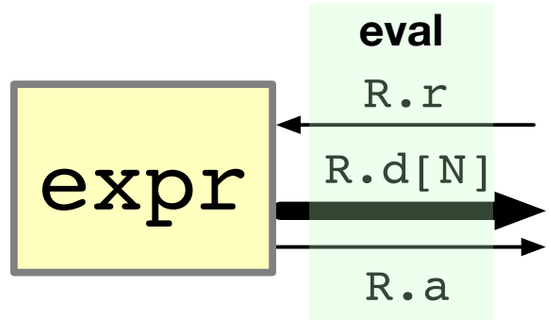


Send / receive

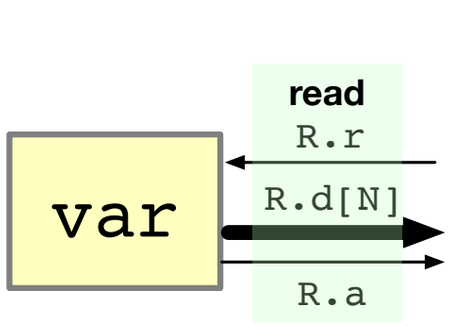
Yale

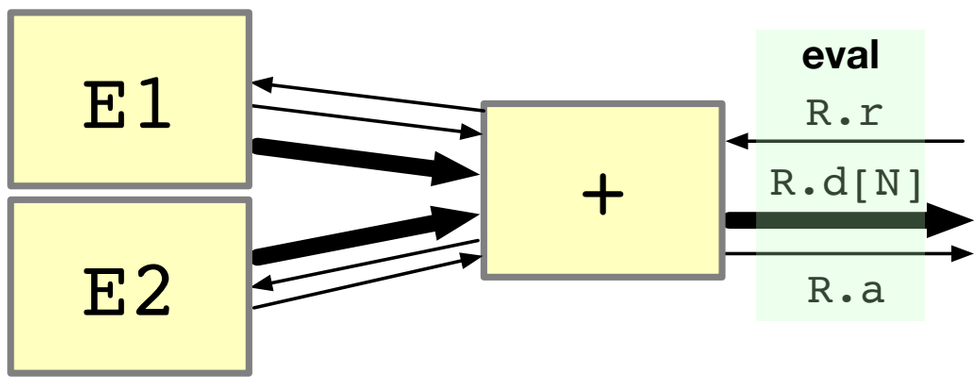
AVLSI

Expression evaluation



- Example of expression de-composition

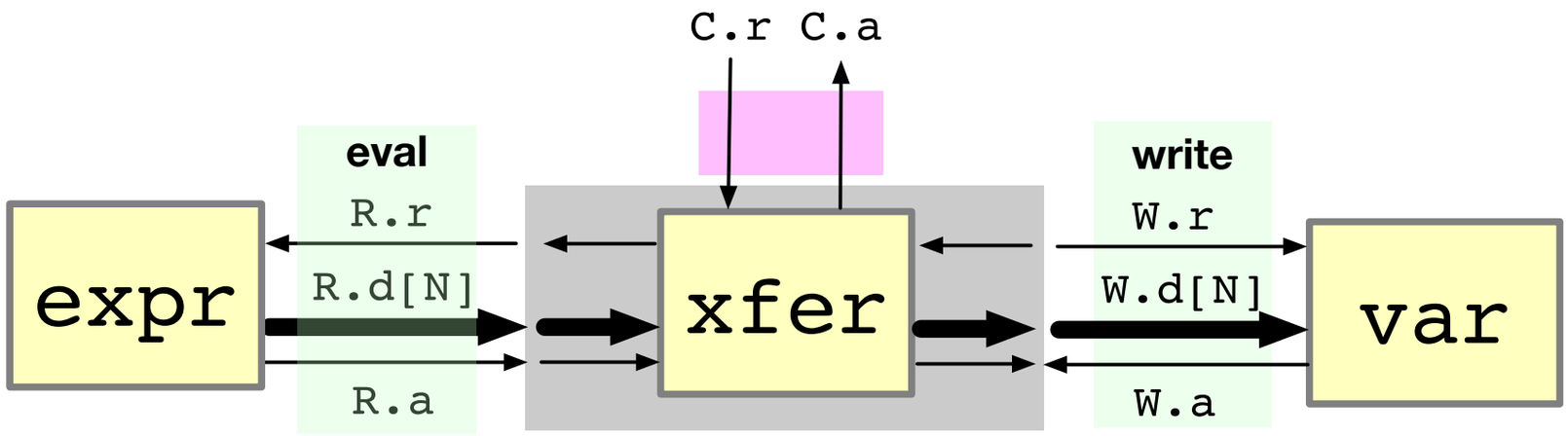




Yale

AVLSI

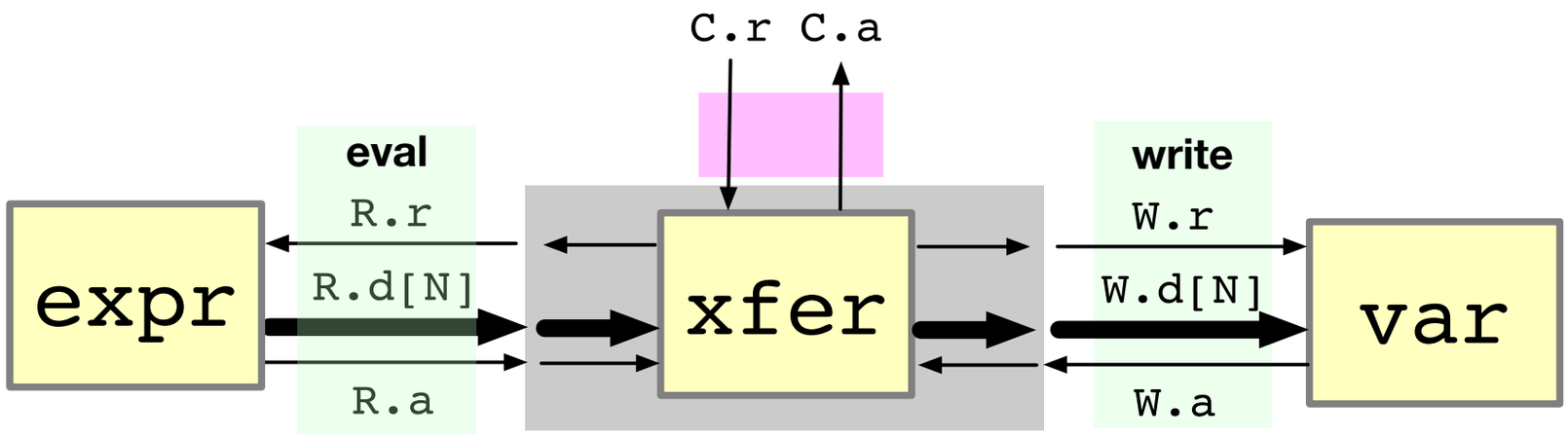
Assignment

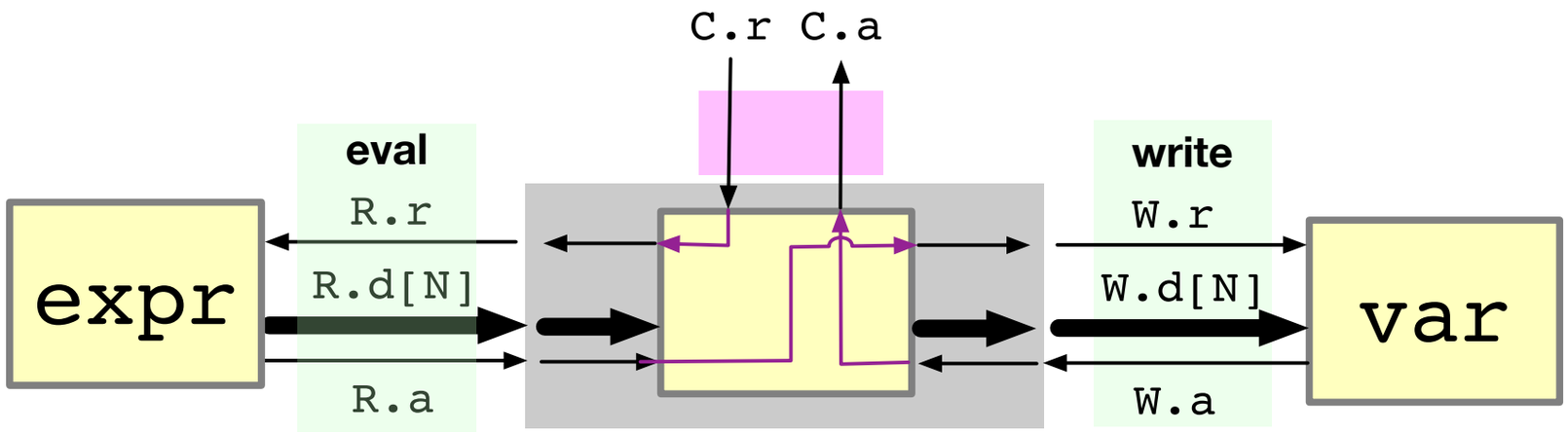


Yale

AVLSI

Assignment

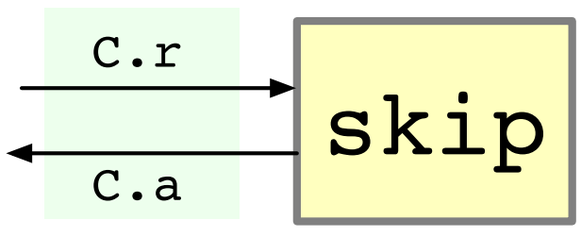


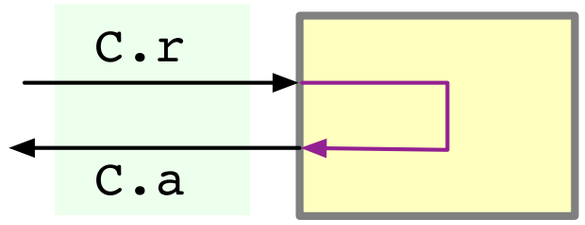


Yale

AVLSI

Building blocks

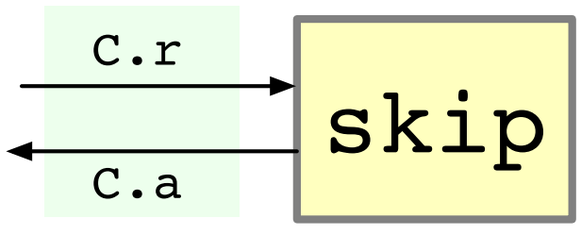


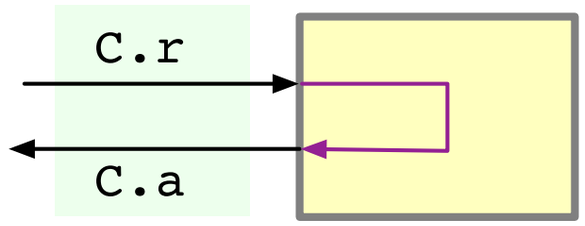


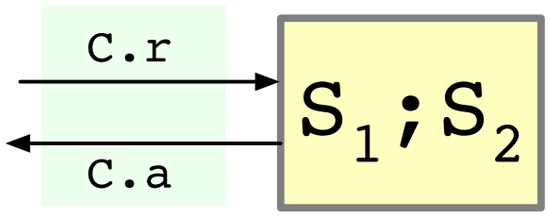
Yale

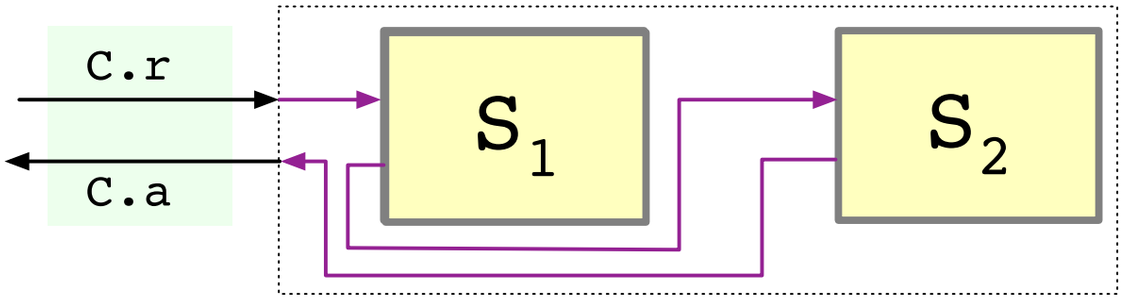
AVLSI

Building blocks





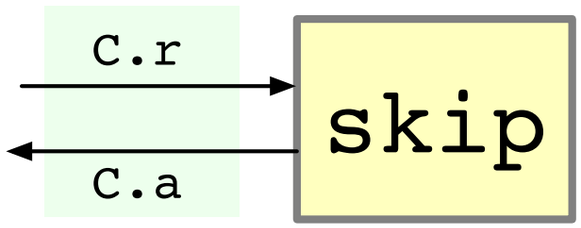


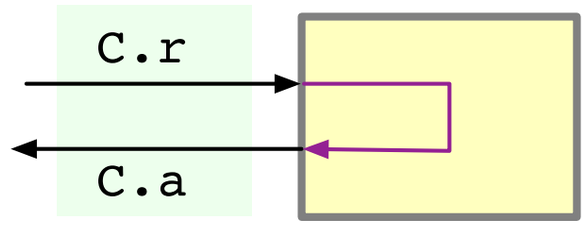


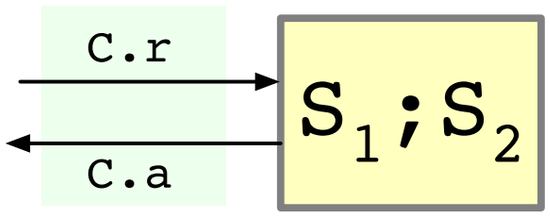
Yale

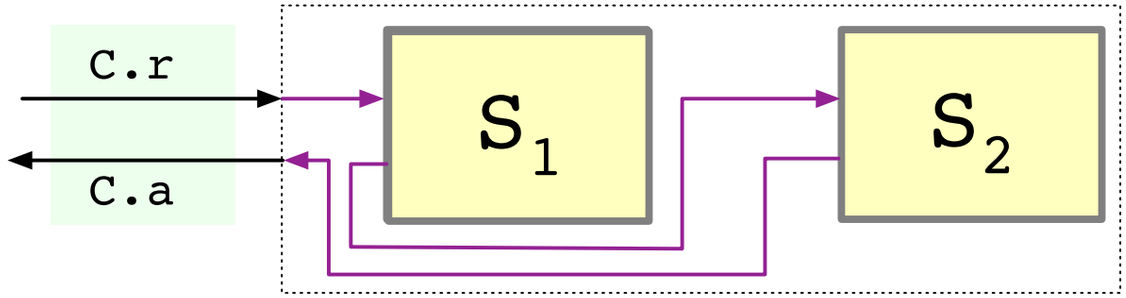
AVLSI

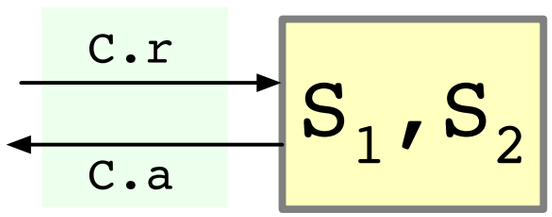
Building blocks

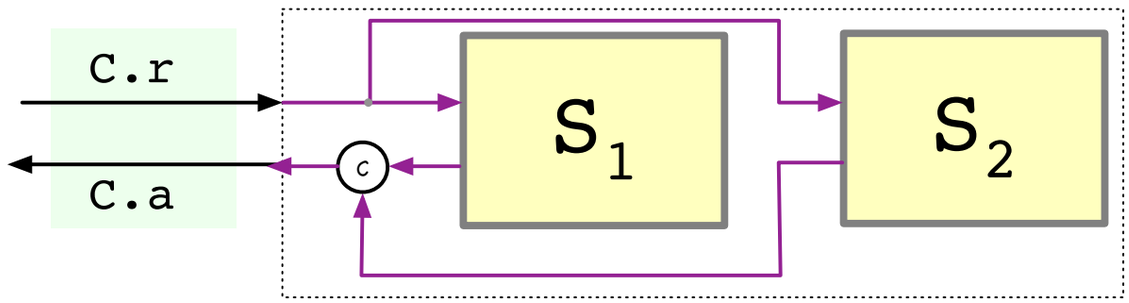








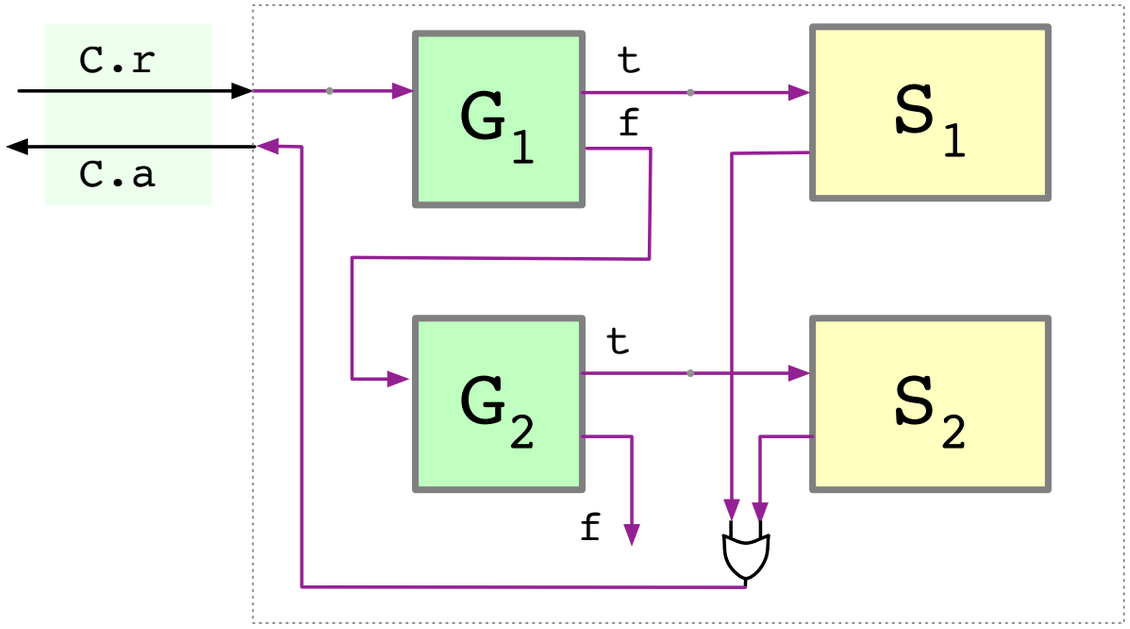




Yale

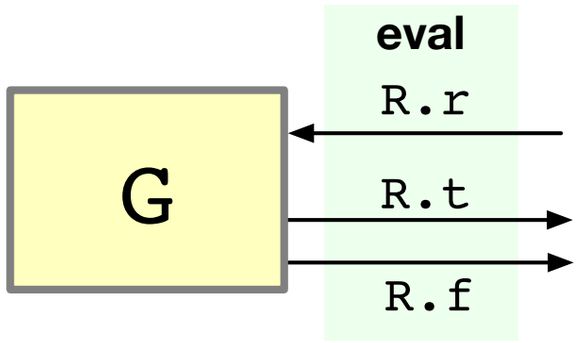
AVLSI

Selections and loops




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

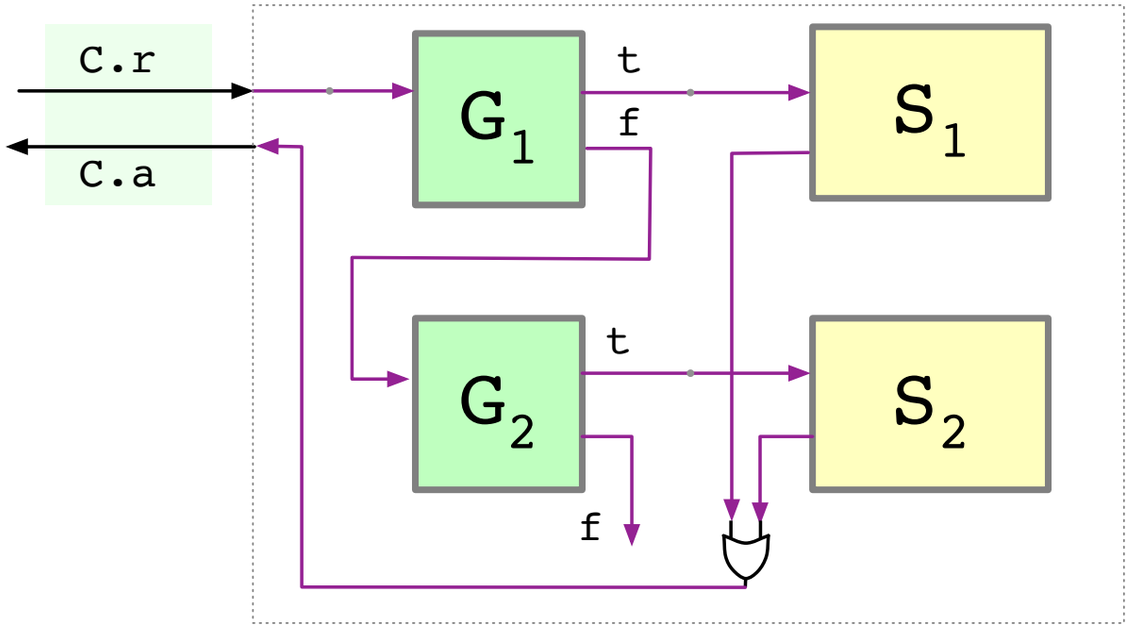
Selection



Yale

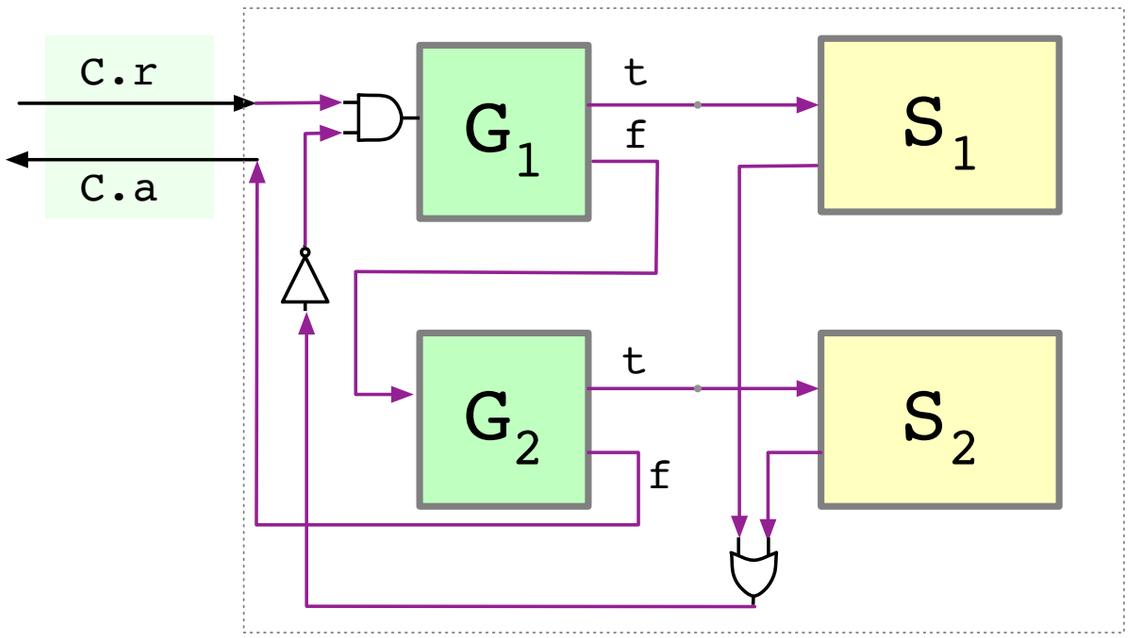
AVLSI

Selections and loops




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

Selection




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

Loop

Yale

AVLSI