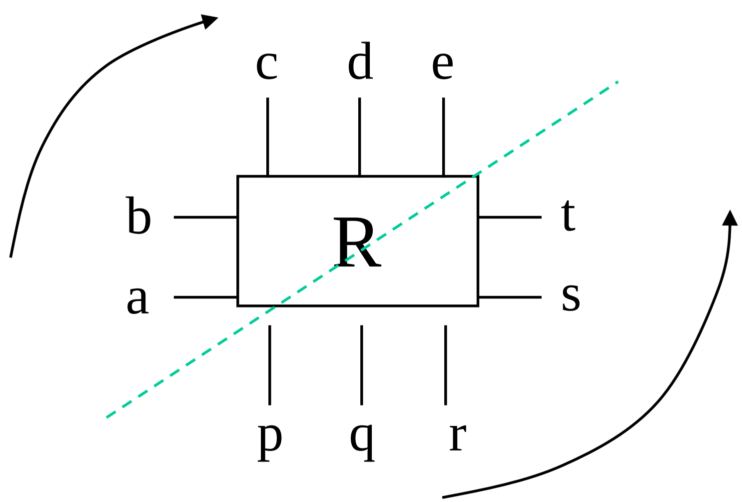
Where we are

- last lecture: types
  - primitive and composite types
  - types for primitive and composite blocks
- last lecture: triangular-shaped architectures
  - tree-shaped array
  - triangular array
- today: grid components and combinators
  - beside and below
  - transposed conjugate
  - row and column

Grid components

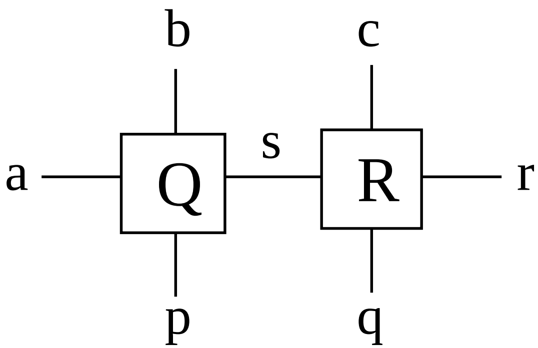
- connections on four sides, type always <X,Y> ~ <U,V>
- convention: from bottom left,
  - count domain connections clockwise
  - count range connections anticlockwise
- example



<<a, b>, <c, d, e>> R <<p, q, r>, <s, t>>

R: <<A,B>,<C,D,E>> ~ <<P,Q,R>,<S,T>>

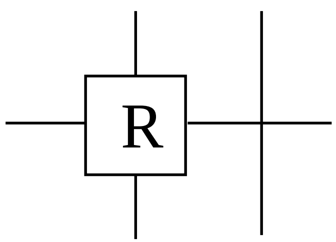
Beside and below

- 

  $\langle a, \langle b, c\rangle\rangle\ (Q \leftrightarrow R)\ \langle\langle p, q\rangle, r\rangle$

  $\Leftrightarrow\ \exists\, s\, .\ \langle a, b\rangle\ Q\ \langle p, s\rangle\ \wedge\ \langle s, c\rangle\ R\ \langle q, r\rangle$

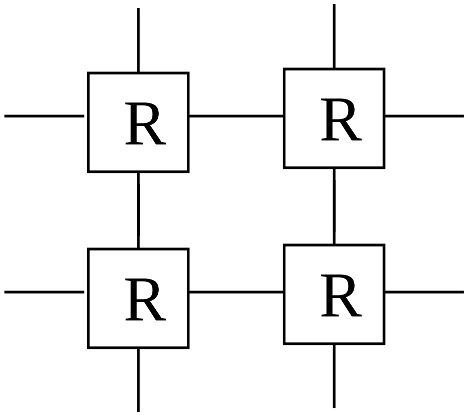
- below: $Q \updownarrow R = (Q^{-1} \leftrightarrow R^{-1})^{-1}$

- 

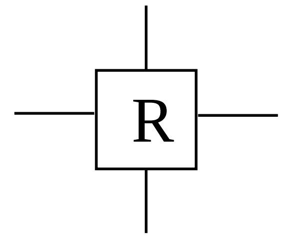
  fsth R = R $\leftrightarrow$ swap

- also fstv R, sndh R, sndv R
- Rebecca: **`Q <-> R, Q <|> R`**

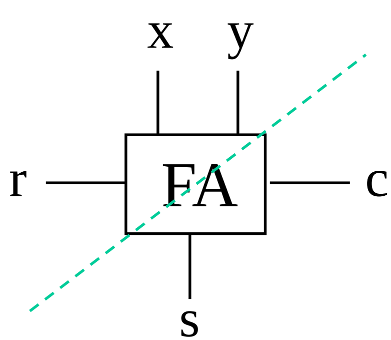
Grid structures

- 

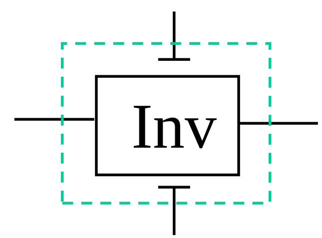
$\mathrm{grid}_{2\ 2}$ R



R \ [id, id]

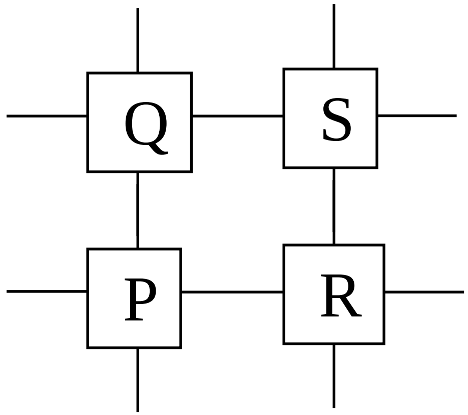
- 

<r, <x, y>> FA <s, c>

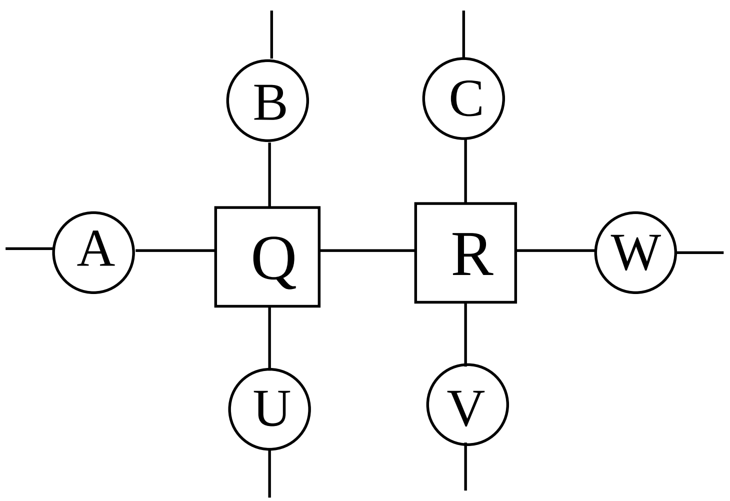


π_1 ; inv ; π_2^{-1}

Laws

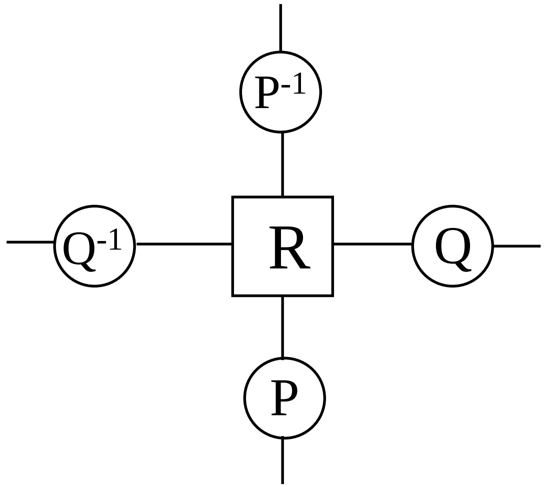
- 

$$(P \updownarrow Q) \;\leftrightarrow\; (R \updownarrow S)$$
$$= \;(P \leftrightarrow R) \;\updownarrow\; (Q \leftrightarrow S)$$

- 

$$[A, [B, C]] \;;\; Q \leftrightarrow R \;;\; [[U, V], W]$$
$$= \;([A, B] \;;\; Q \;;\; \mathrm{fst}\ U)$$
$$\leftrightarrow\; (\mathrm{snd}\ C \;;\; R \;;\; [V, W])$$

Transposed conjugate

- 

$$R \backslash\backslash [P, Q] \stackrel{\text{def}}{=} [Q, P]^{-1} ; R ; [P, Q]$$

- $$\begin{aligned} R \backslash\backslash (\text{fst } Q) &= R \backslash\backslash [Q, \text{id}] \\ &= [\text{id}, Q^{-1}] ; R ; [Q, \text{id}] \\ &= \text{snd } Q^{-1} ; R ; \text{fst } Q \end{aligned}$$

- $$\begin{aligned} &(\text{fstv } R) \leftrightarrow (\text{sndv } S) \\ &\quad = ((\text{sndv } S) \leftrightarrow (\text{fstv } R)) \backslash\backslash (\text{fst swap}) \\ &\quad = \text{snd swap}^{-1} ; (\text{sndv } s) \leftrightarrow (\text{fstv } R) ; \text{fst swap} \end{aligned}$$

Useful definitions

- create list with one element

  x [-] <x>

- infix append

  <a,b,c>^<d,e,f,g> = <a,b,c,d,e,f,g>

- append left

  <a,<b,c,d>> apl_3 <a,b,c,d>

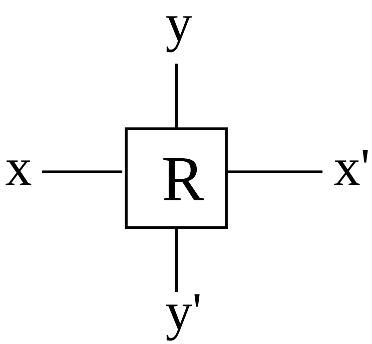
  <x, xs> apl_n <x>^xs, #xs = n (there are n elements in xs)

- append right

  <<a,b,c,d>, e> apr_4 <a,b,c,d,e>

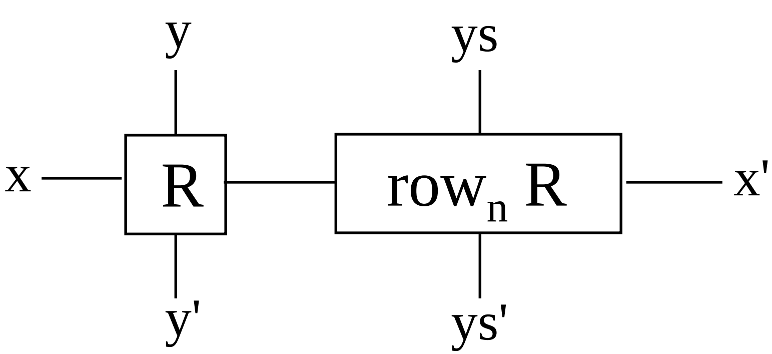
  <xs, x> apr_n xs^<x>, #xs = n

Row: repeated beside

- 

<x, <y>>   <x, y>   <y' x'>   <<y'>, x'>

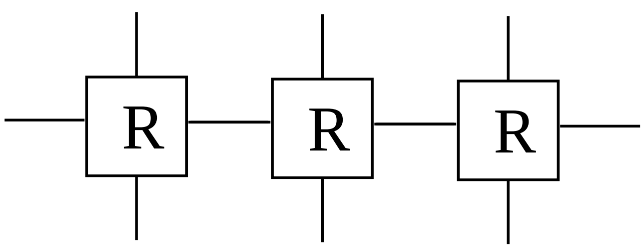
$$\mathrm{row}_1\ R \overset{\mathrm{def}}{=} \mathrm{snd}\ [-]^{-1}\ ;\ R\ ;\ \mathrm{fst}\ [-]$$
$$= R \setminus\!\setminus \mathrm{fst}\ [-]$$

- 

sequence concatenation

<x, <y> ^ ys>   <x, <y, ys>>

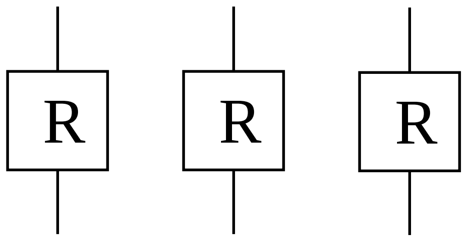
$$\mathrm{row}_{n+1}\ R \overset{\mathrm{def}}{=} \mathrm{snd}\ \mathrm{apl}_n^{-1}\ ;$$
$$R \leftrightarrow (\mathrm{row}_n\ R)\ ;$$
$$\mathrm{fst}\ \mathrm{apl}_n$$
$$= (R \leftrightarrow \mathrm{row}_n\ R) \setminus\!\setminus \mathrm{fst}\ \mathrm{apl}_n$$

- 

$\mathrm{row}_3\ R$

$$\mathrm{col}_n\ R = ?$$
$$\mathrm{grid}_{m\ n}\ R = ?$$

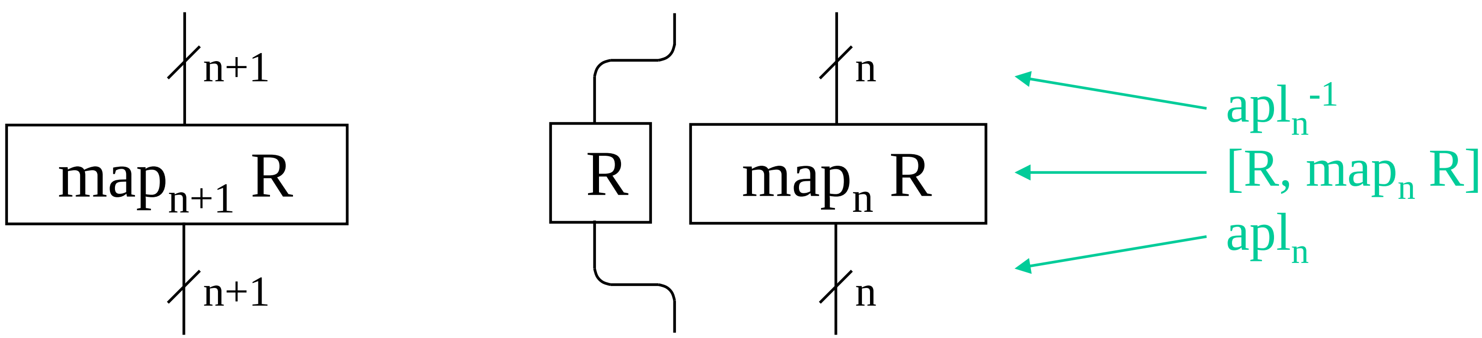
Comparison: repeated parallel composition

- 

$$\mathrm{map}_3\ R \ = \ [R, R, R]$$

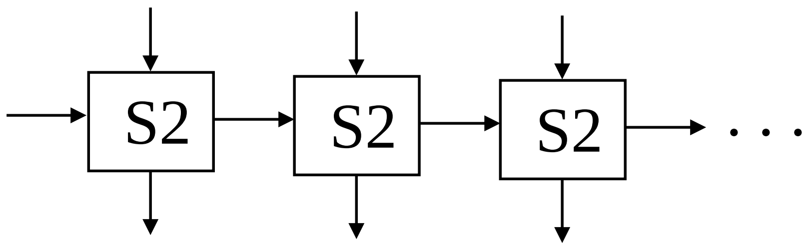
- recursive description:
  - base case: $\mathrm{map}_0\ R = [\,]$

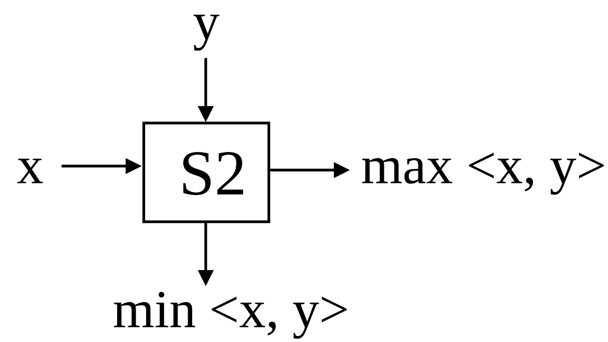
- induction

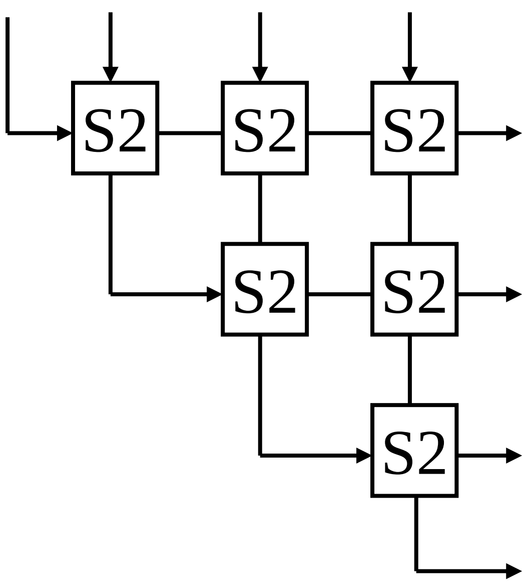


$$\begin{aligned}\mathrm{map}_{n+1}\ R \quad &= \quad \mathrm{apl}_n^{-1}\ ;\ [R, \mathrm{map}_n\ R]\ ;\ \mathrm{apl}_n \\ &= \quad [R, \mathrm{map}_n\ R] \setminus \mathrm{apl}_n \\ & \quad\ \ \text{where} \quad Q \setminus P \ = \ P^{-1}\ ;\ Q\ ;\ P\end{aligned}$$

What do they do?

- 



- 

Answer to Unassessed Coursework 2

```
1. (a) mac     = fst mult ; add.
   (b) twoadd  = rsh ; fst add ; add.

2. R:X~X, Q:X~Y, (R\Q):Y~Y
   append m n: <<X>_m, <X>_n> ~ <X>_(m+n)
   R:X~X, /\ n R: <X>_n ~ <x>_n
```

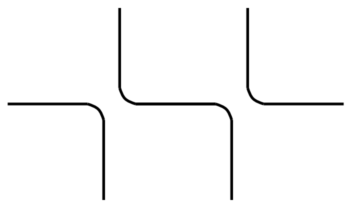
3. For polynomial evaluation $y = \sum_{i<n} a_i \times x^i$

```
 wpe n  = fst (mfork n) ; zip n.

 pecell = fst fork ; lsh ; snd mult.

  pe n  = wpe n; /\ n pecell ;
          map n pi2 ; btree (n $log 2) add.

btree n R = IF (n $eq 1) THEN R
            ELSE (half (2 $exp (n-1)) ;
            [btree (n-1) R, btree (n-1) R] ; R).
```

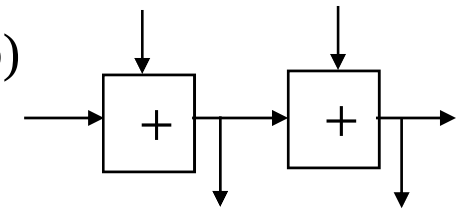
Unassessed Coursework 3

1. Describe the following without $rel. What are the types?

(a)



(b)



2. Describe and simulate the following design: