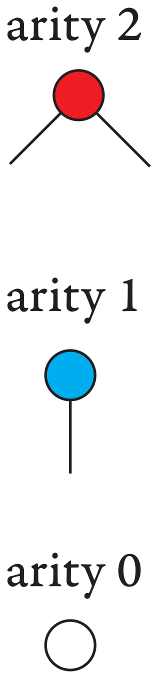
a ranked alphabet

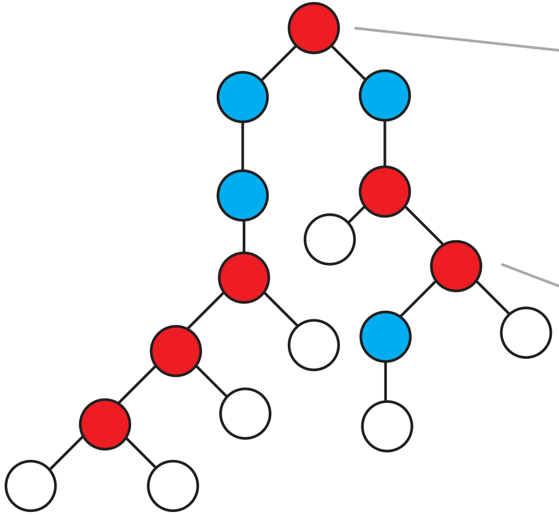


arity 2

arity 1

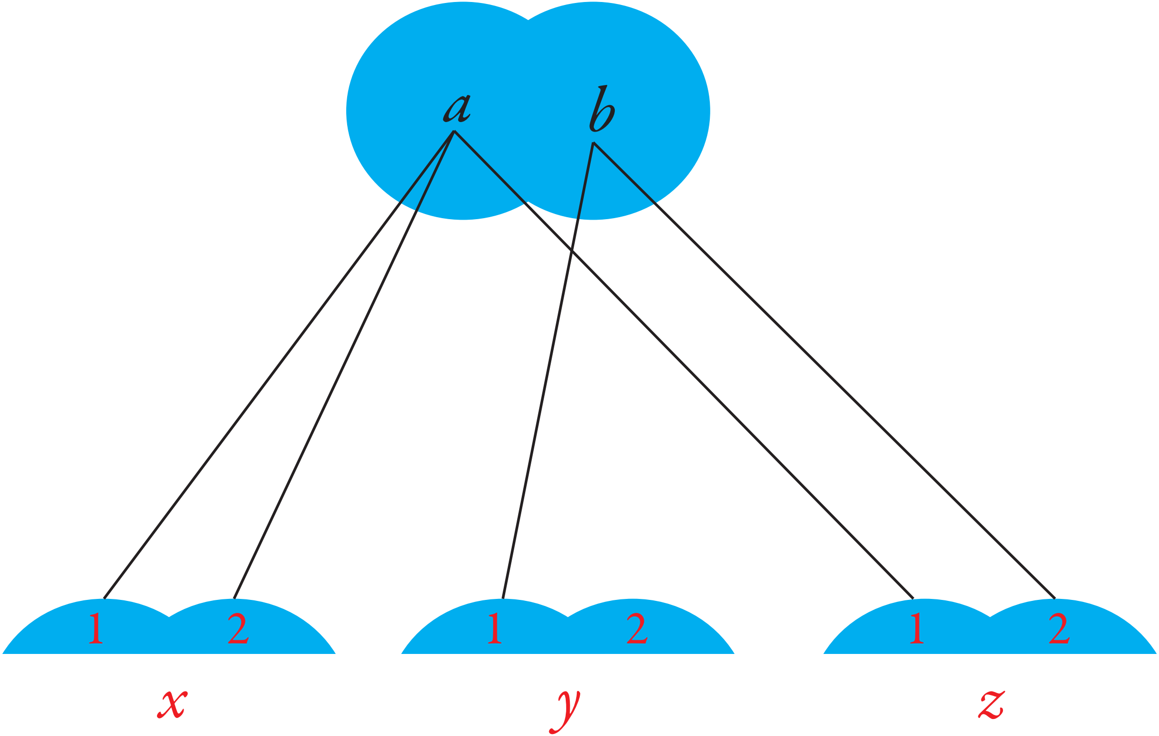
arity 0

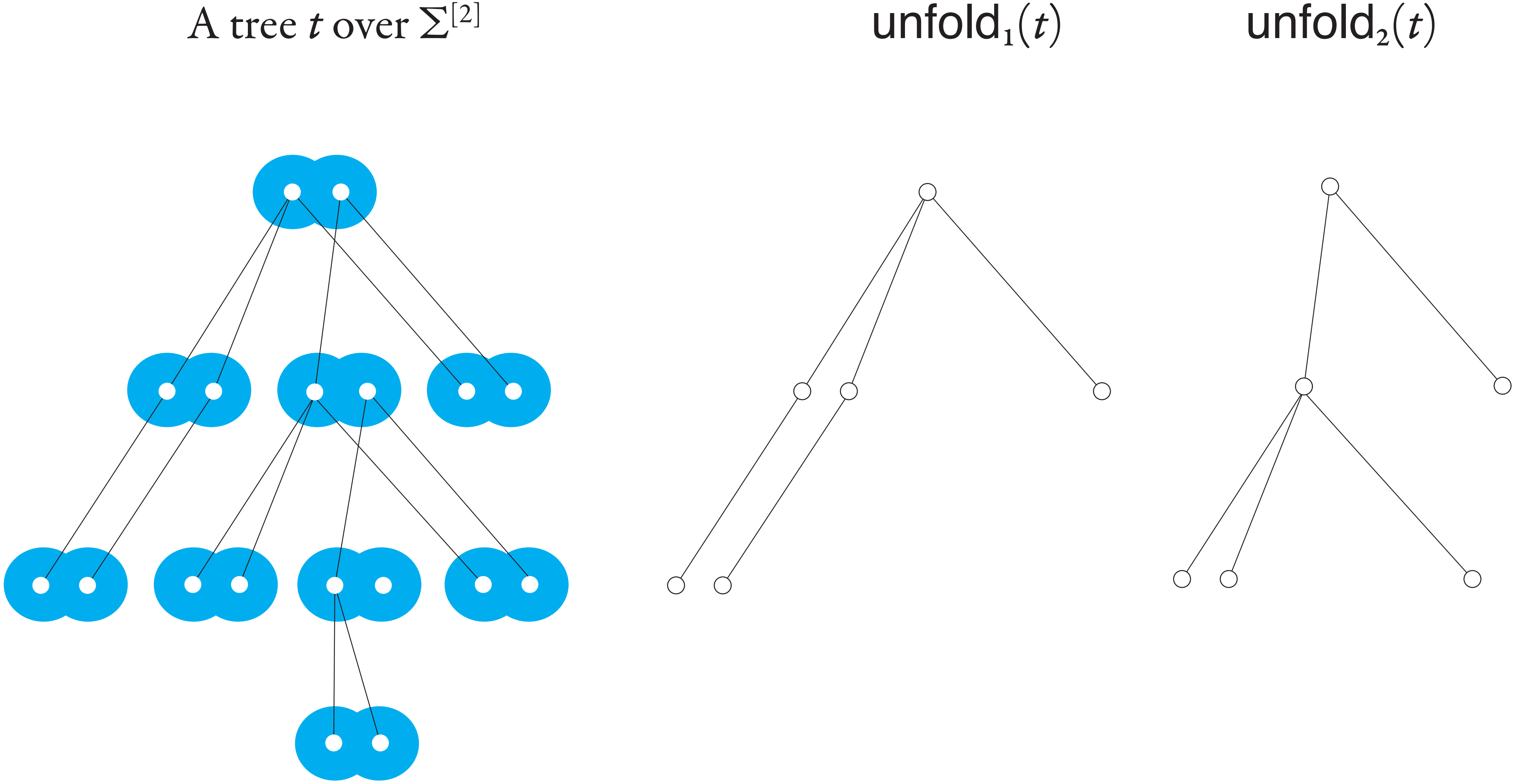
a tree



this node has a label of arity 2,
and therefore it has 2 children

this node is child 2
(children are ordered)

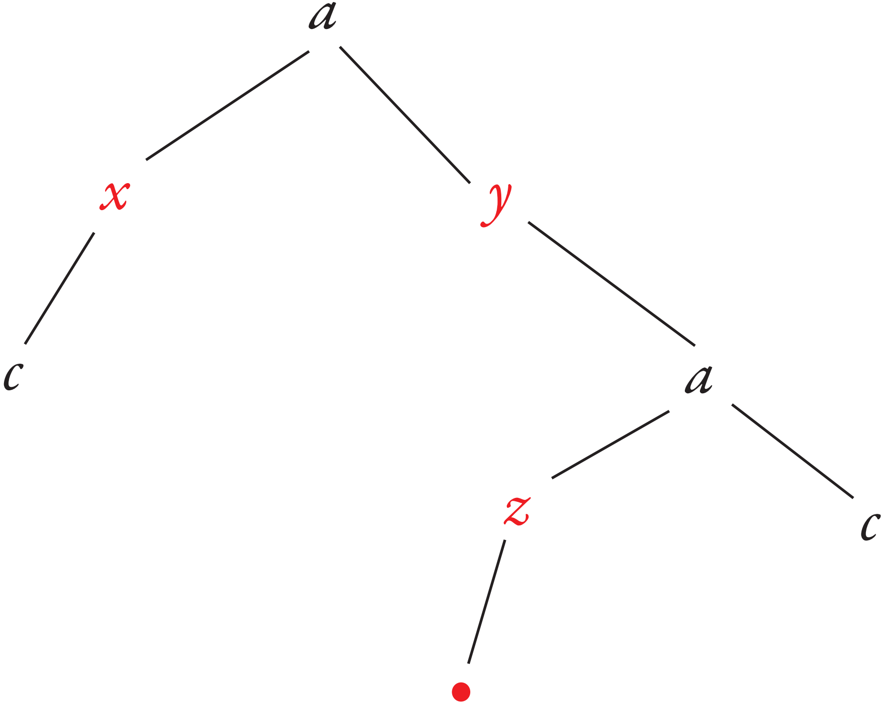

a
b
1
2
1
2
1
2
x
y
z

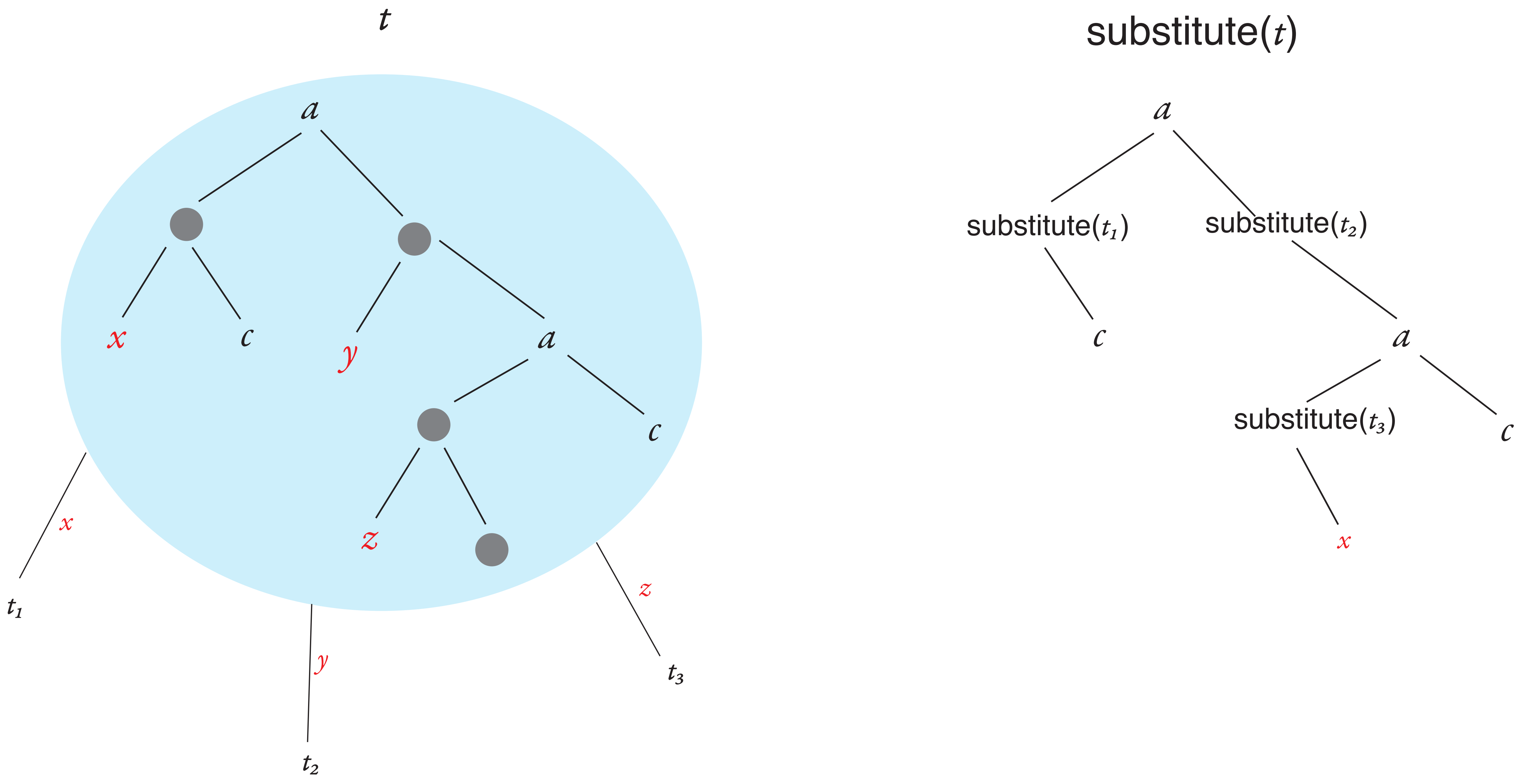


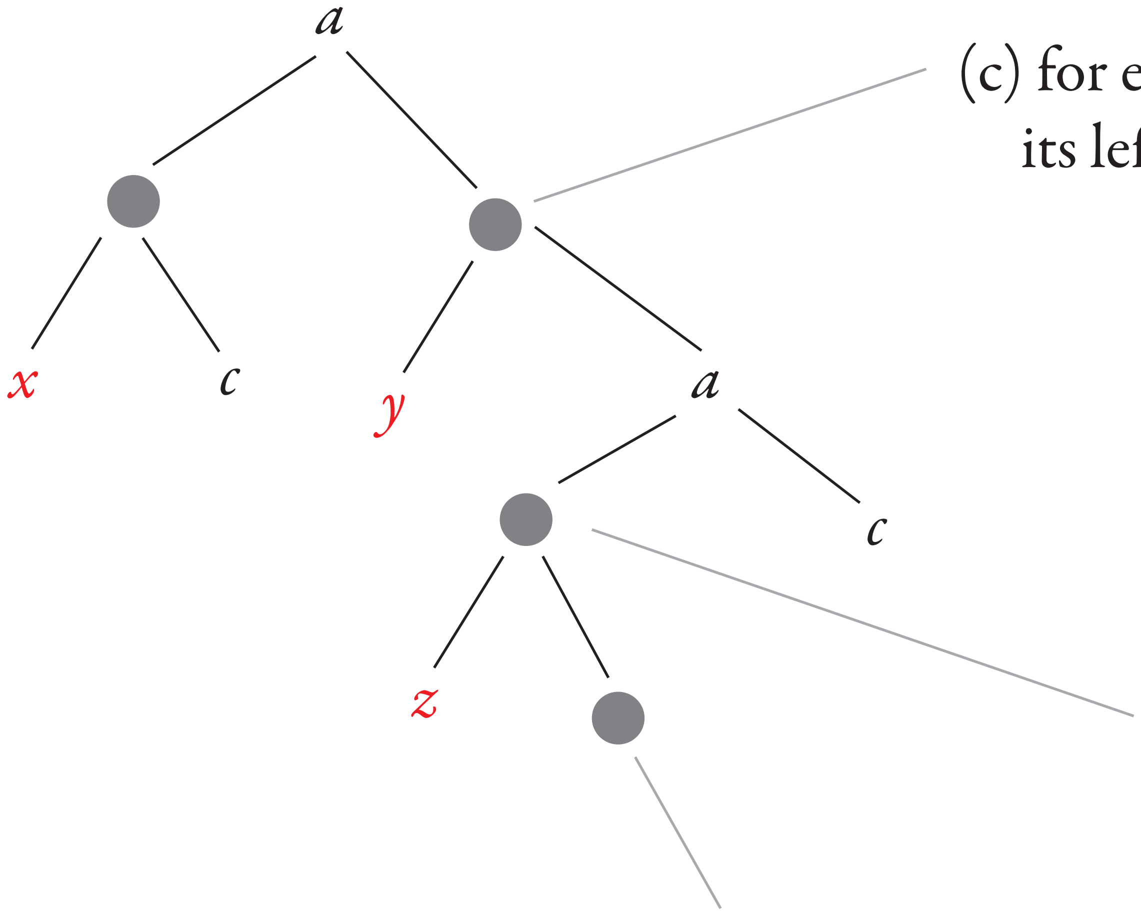
A tree t over $\Sigma^{[2]}$

$\mathsf{unfold}_1(t)$

$\mathsf{unfold}_2(t)$


a
x
y
c
a
z
c

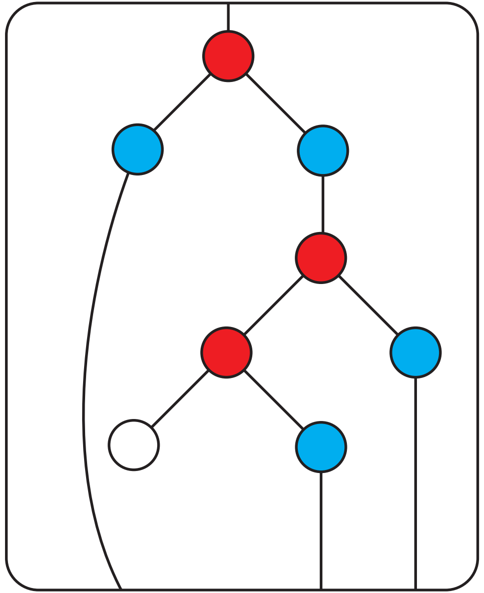

t
a
x
c
y
a
c
z
x
t1
y
t2
z
t3
substitute(t)
a
substitute(t1)
substitute(t2)
c
a
substitute(t3)
c
x



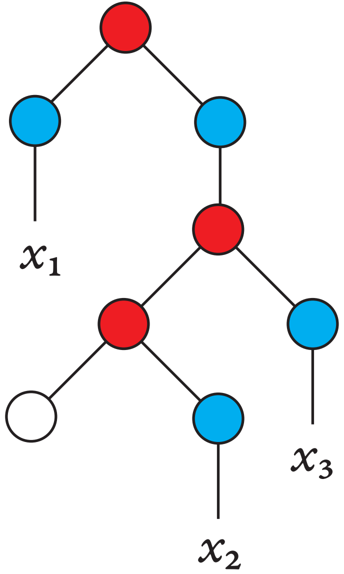
(c) for each binary letter from 1, its left child is a variable

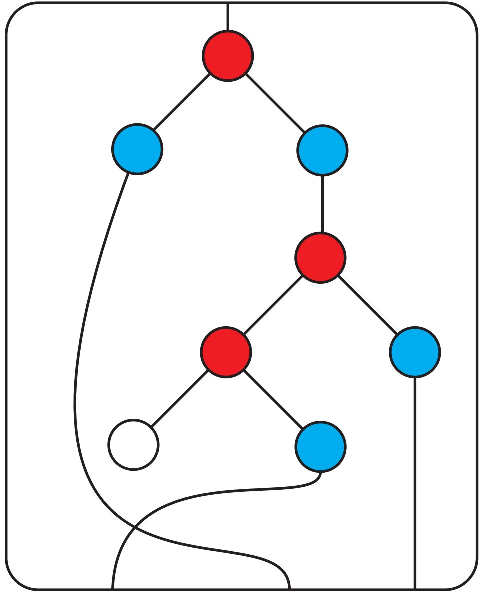
(a) only binary and nullary letters from 1 are allowed

(b) a nullary letter from 1 is allowed exactly once

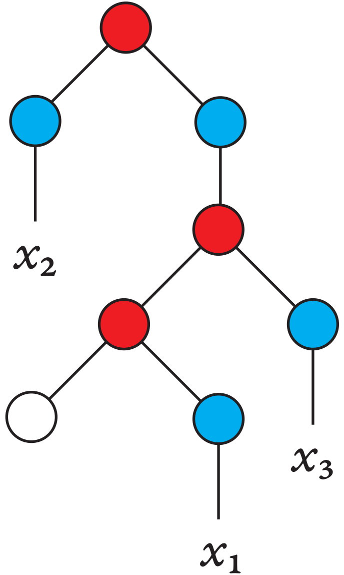


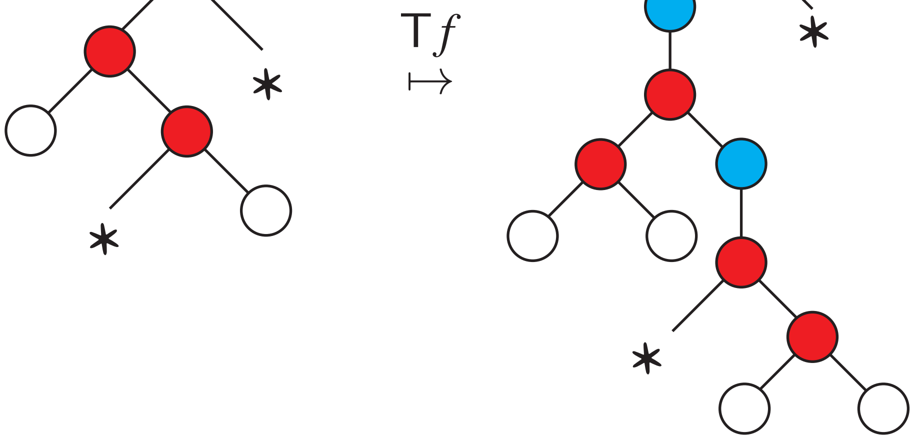
=

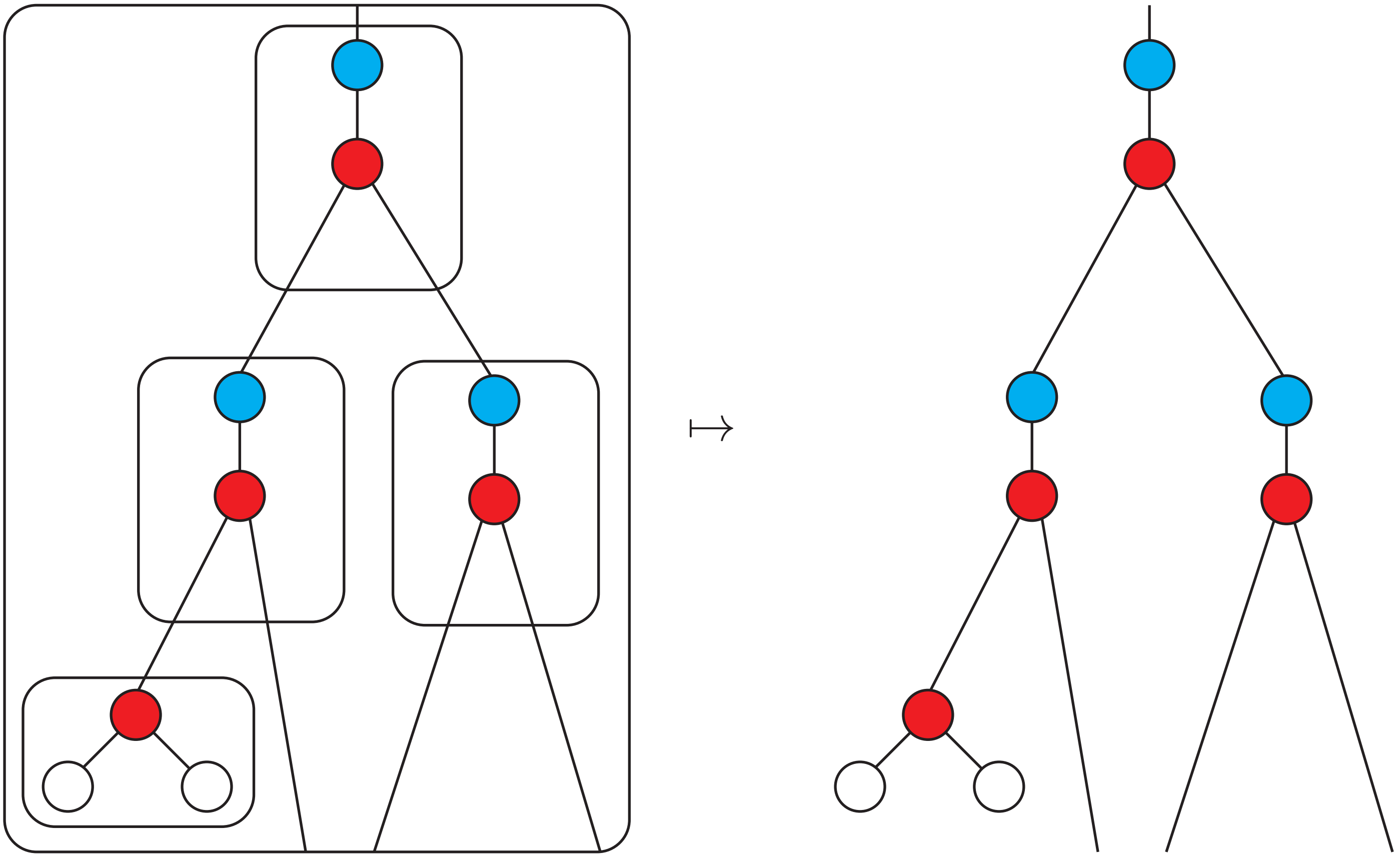

x_1
x_3
x_2

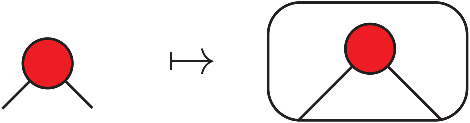


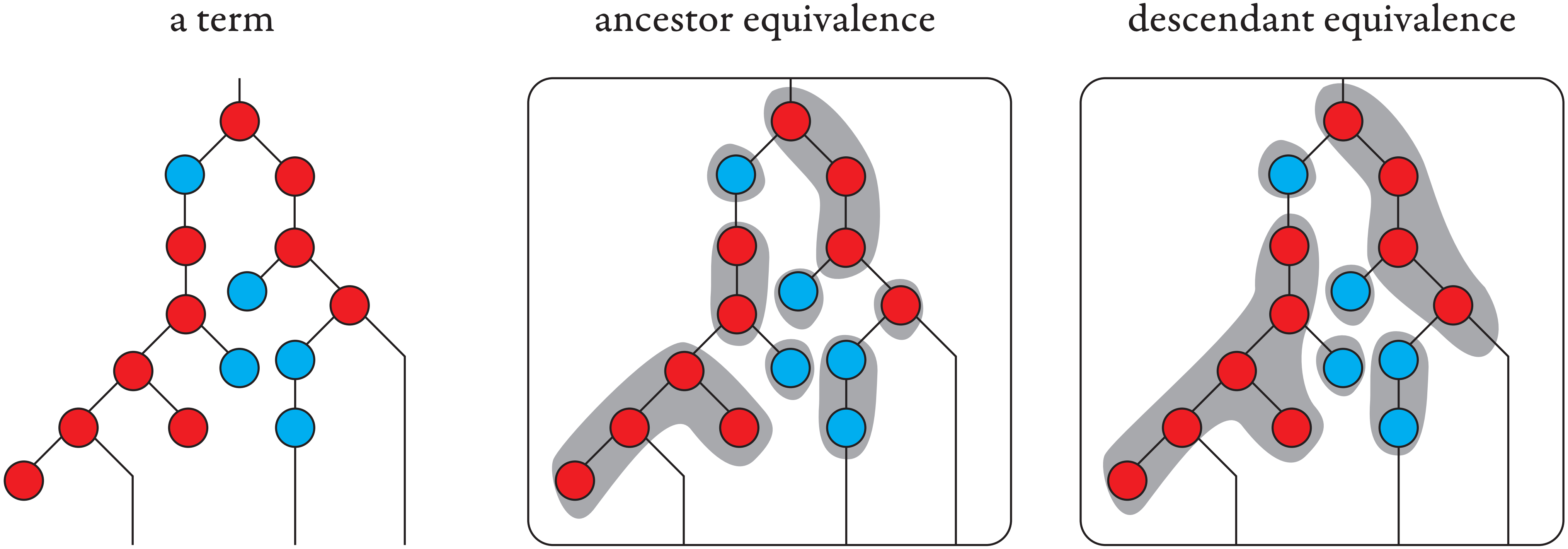
=


x_2
x_3
x_1


$\mathsf{T}f$
$\longmapsto$
$*$
$*$
$*$
$*$



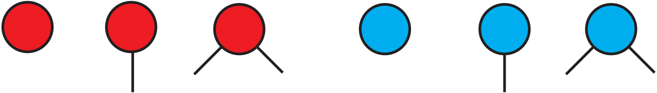


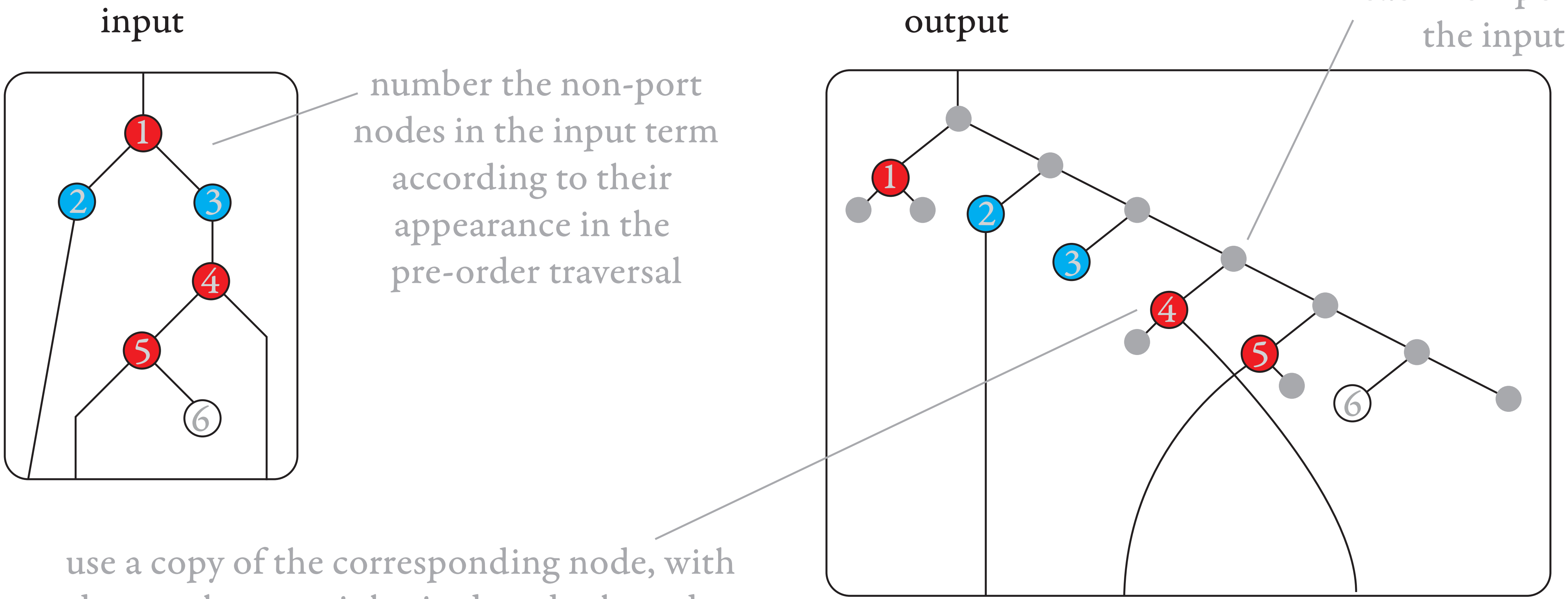


a term

ancestor equivalence

descendant equivalence



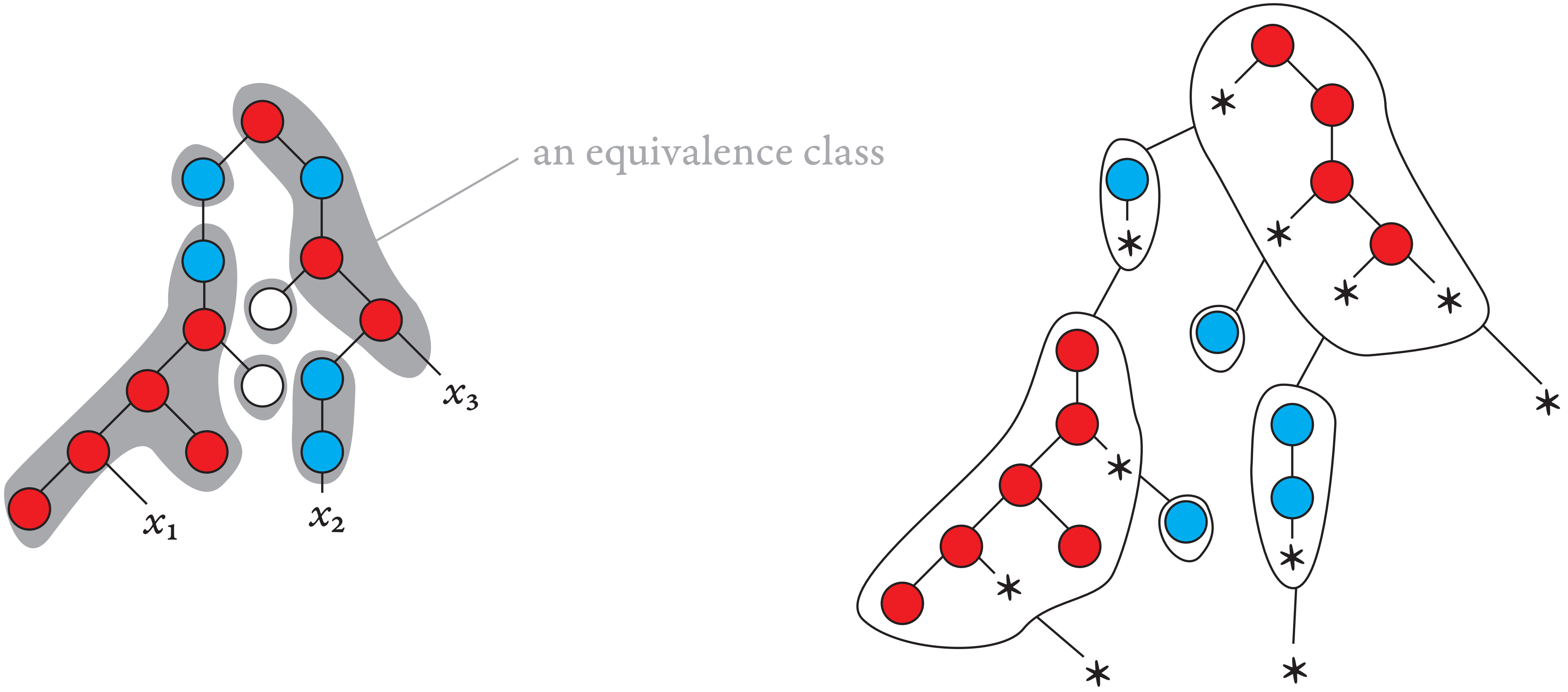


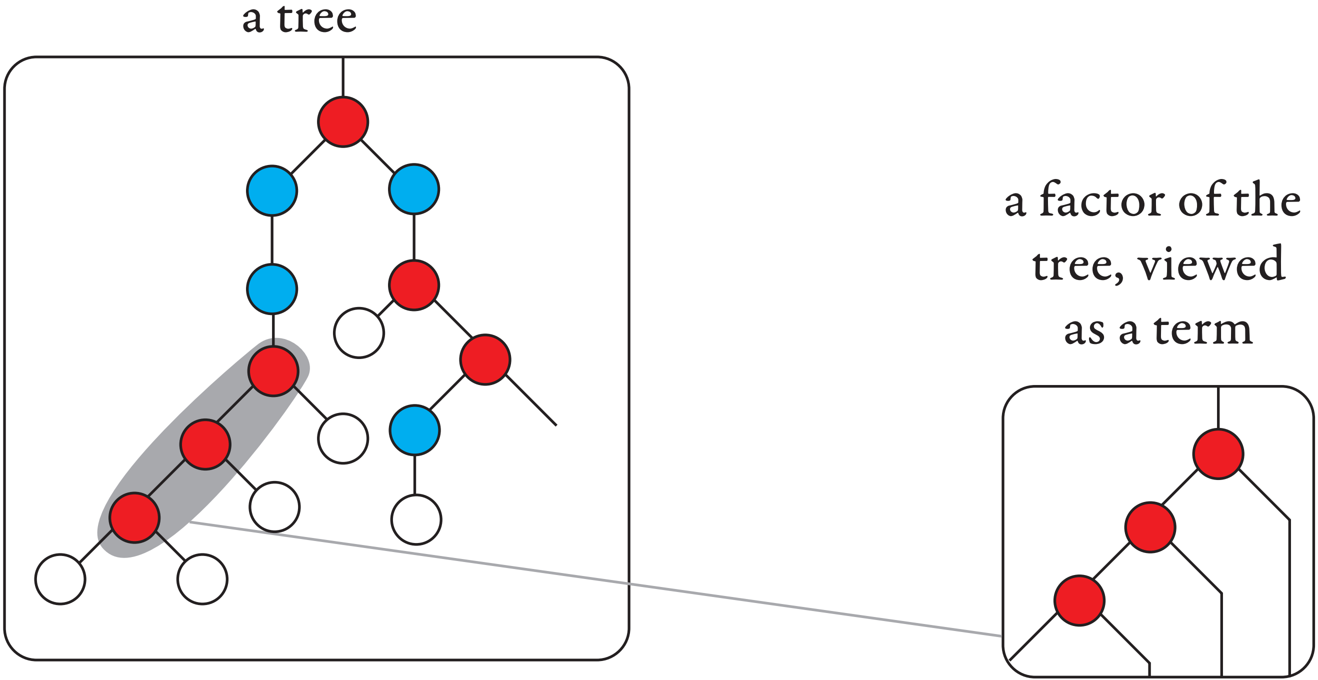
number the non-port nodes in the input term according to their appearance in the pre-order traversal

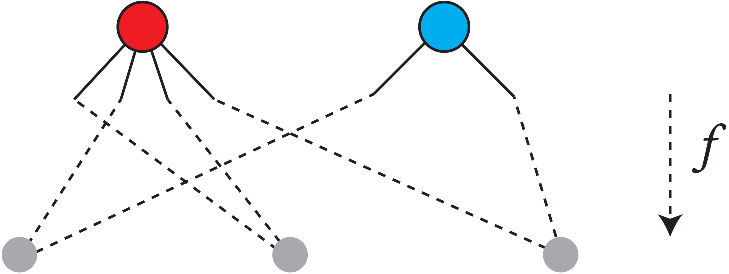
create a binary node for each non-port node in the input term

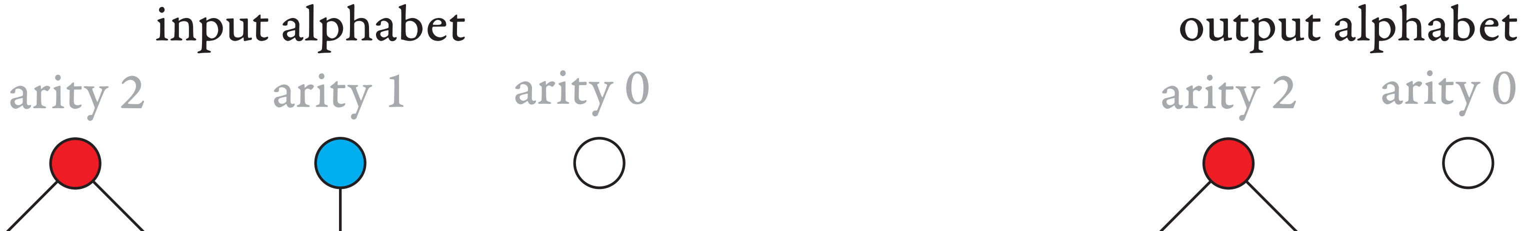
use a copy of the corresponding node, with edges to the ports inherited, and other edges plugged by ●

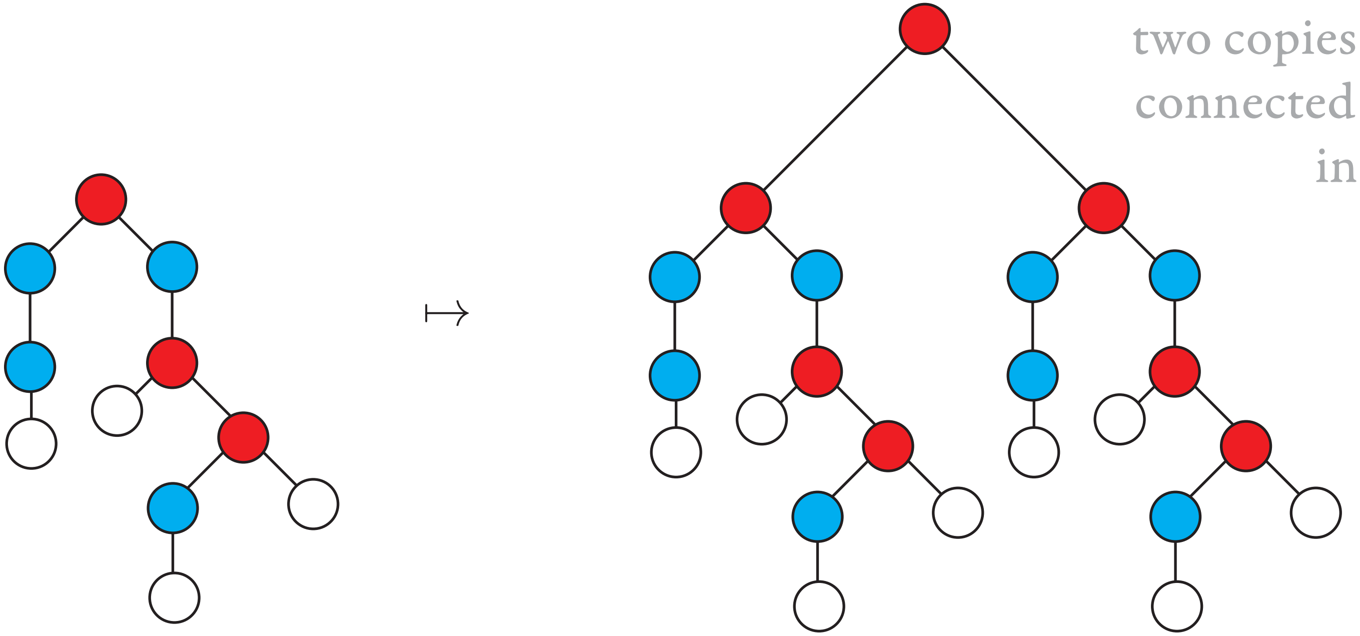
a factorisation equivalence



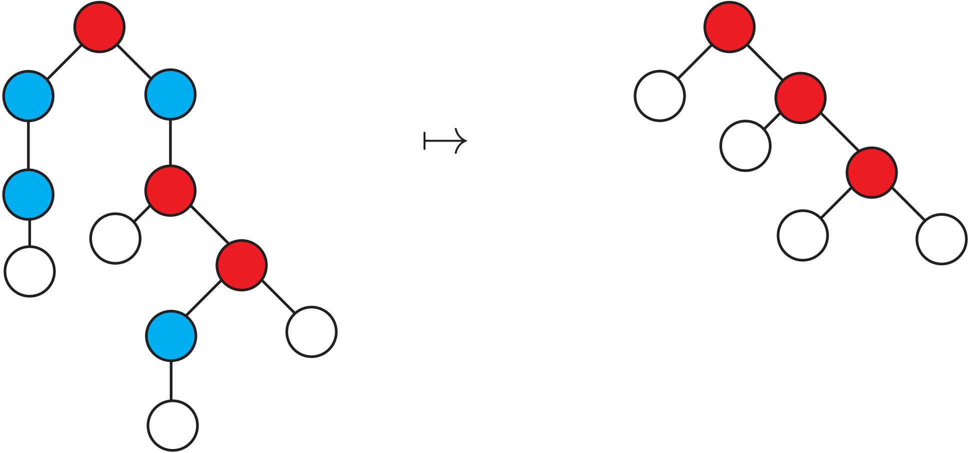

a tree
a factor of the tree, viewed as a term

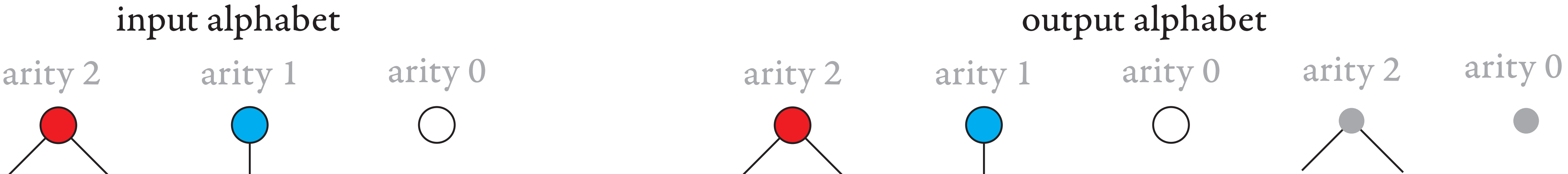

f

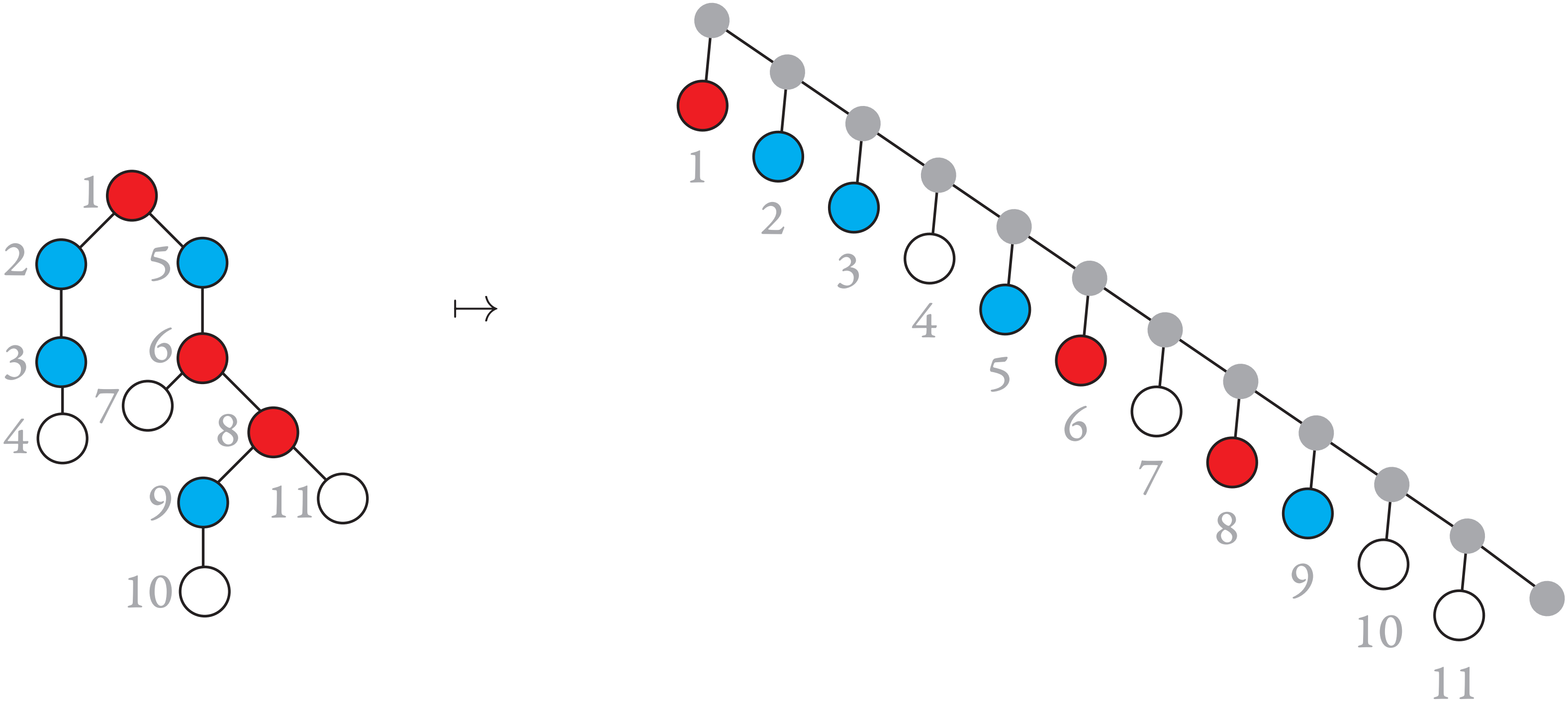

input alphabet
arity 2
arity 1
arity 0
output alphabet
arity 2
arity 0



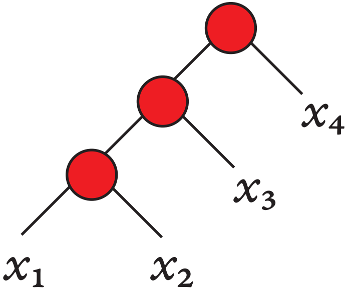
two copies of the input tree, connected by a binary node in the root



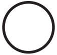

input alphabet
arity 2
arity 1
arity 0
output alphabet
arity 2
arity 1
arity 0
arity 2
arity 0

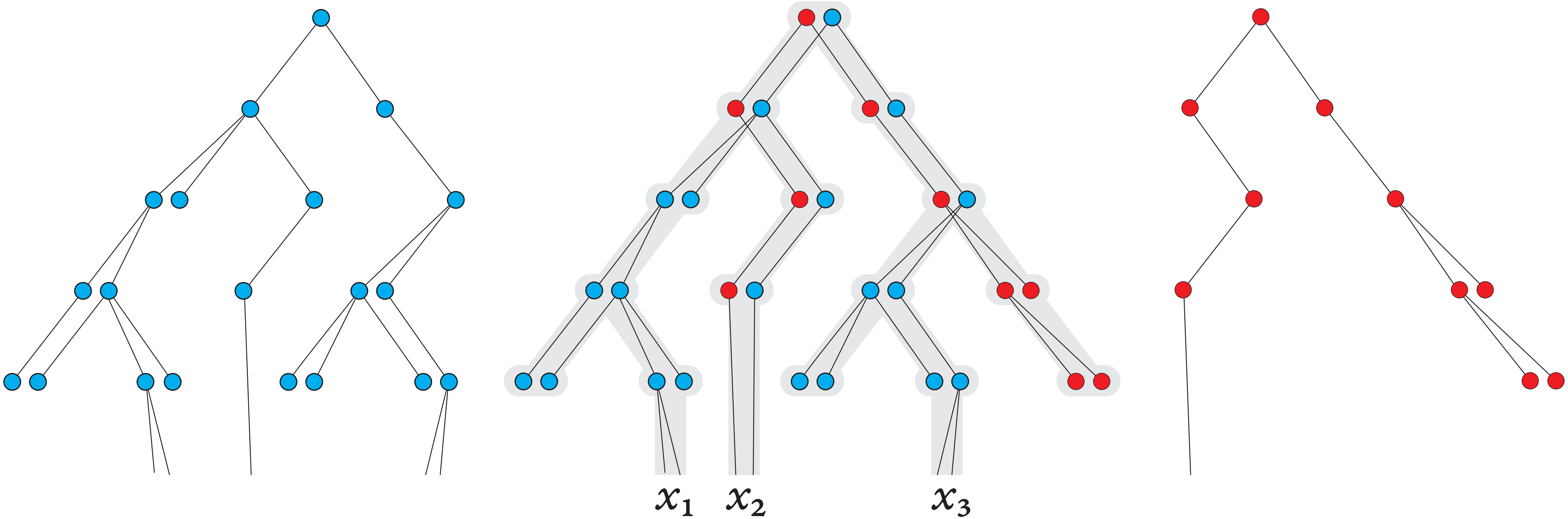

1
2
5
3
6
7
4
8
9
11
10
↦
1
2
3
4
5
6
7
8
9
10
11

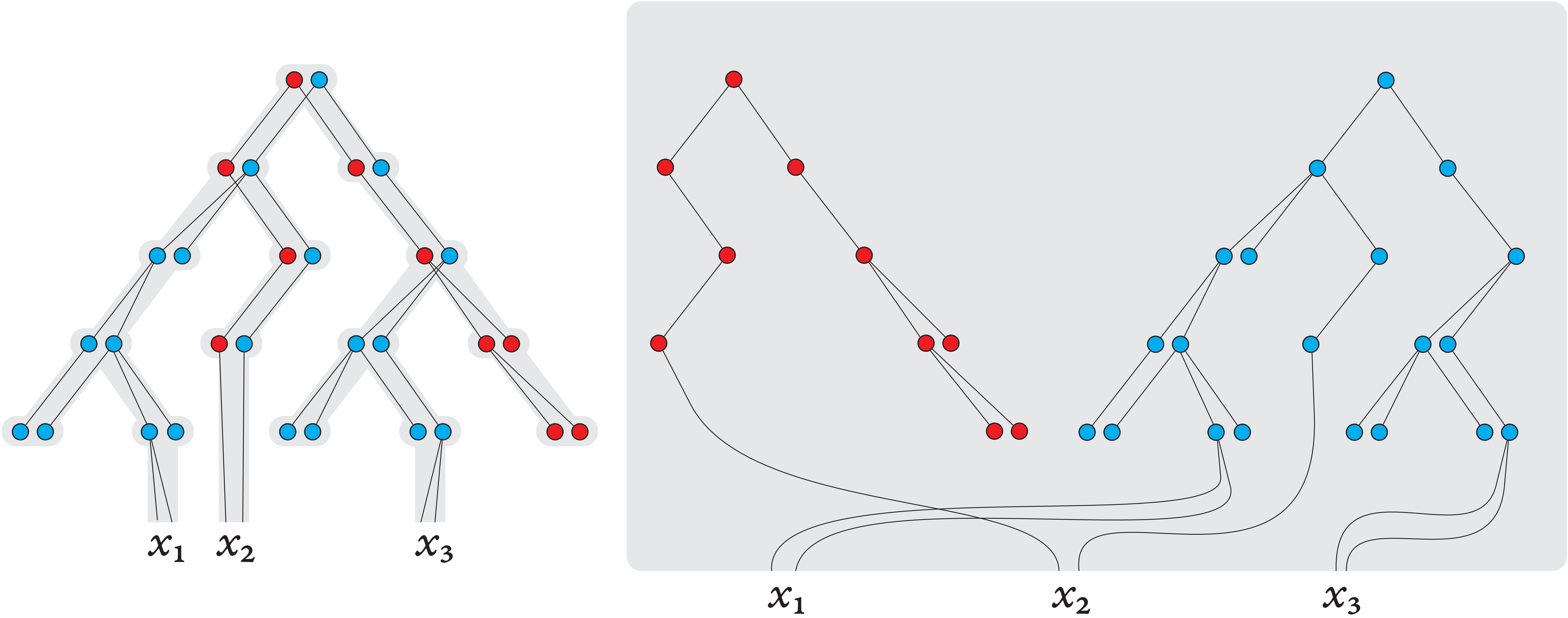
a term of arity 4

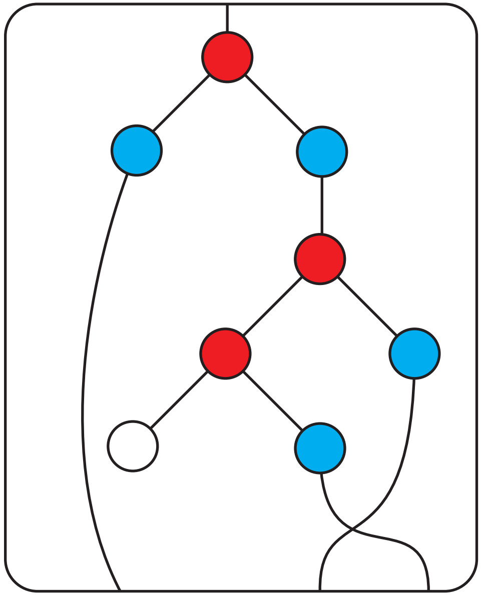


a term of arity 0




x_1
x_2
x_3

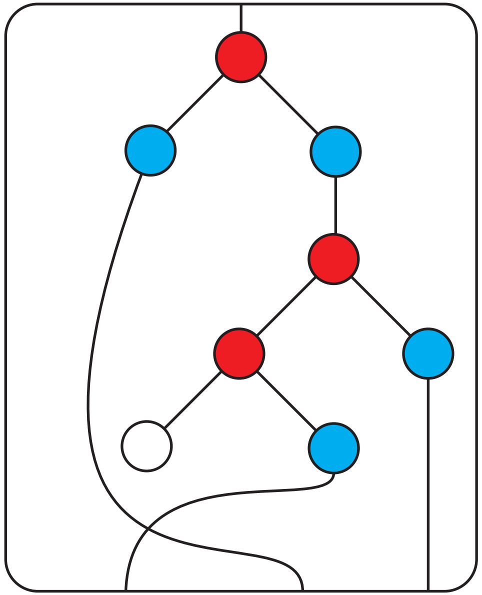

x_1
x_2
x_3
x_1
x_2
x_3



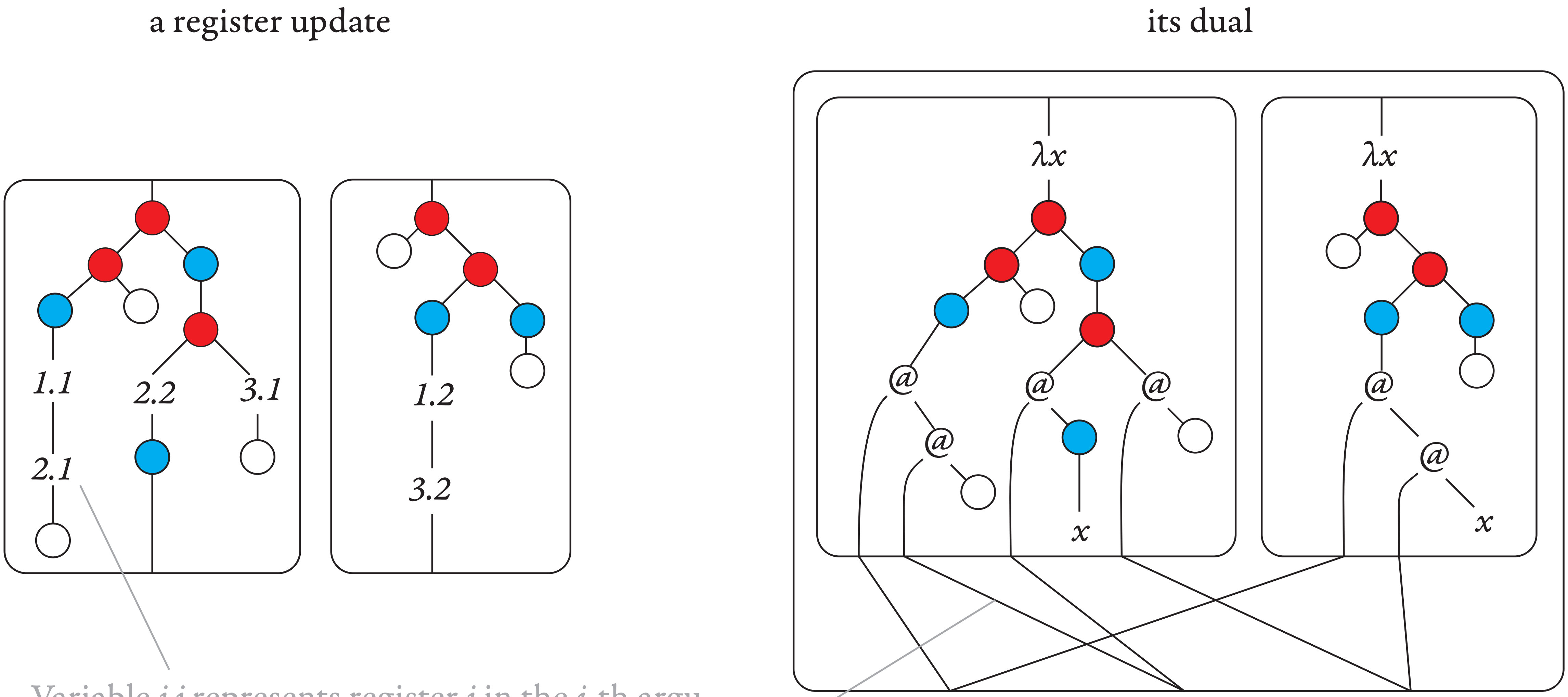
satisfies (*)

(*)

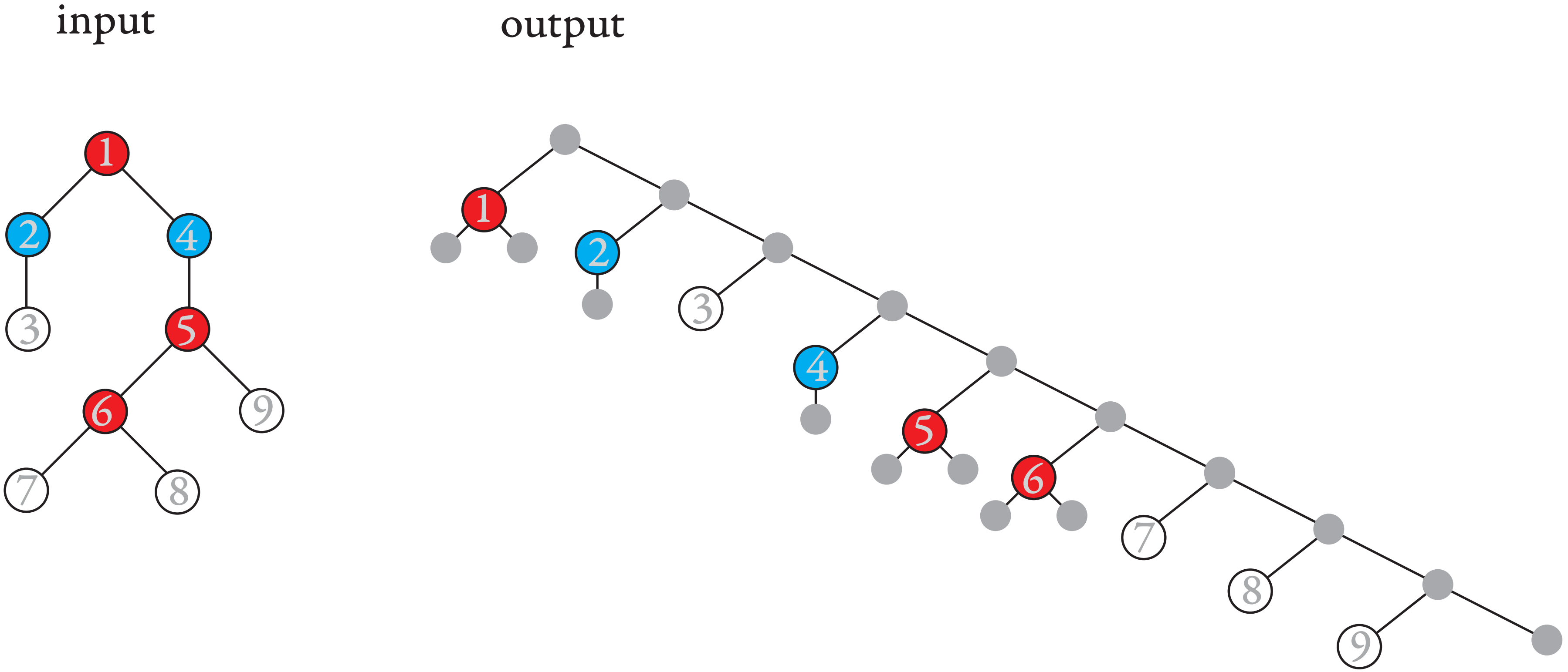
If the root has arity n,
and $1 \leq i < j \leq n$, then
all ports of the j-th
subterm of the root are
after all ports of the
i-th subterm of the root



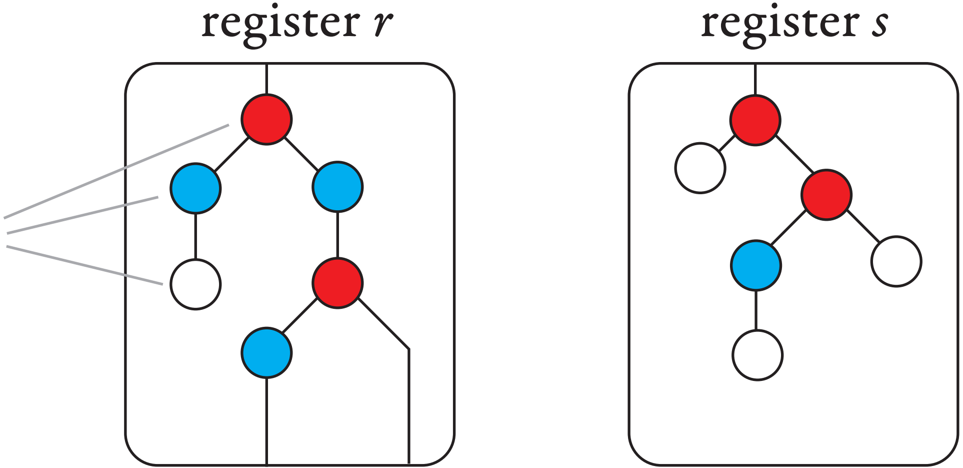
violates (*)

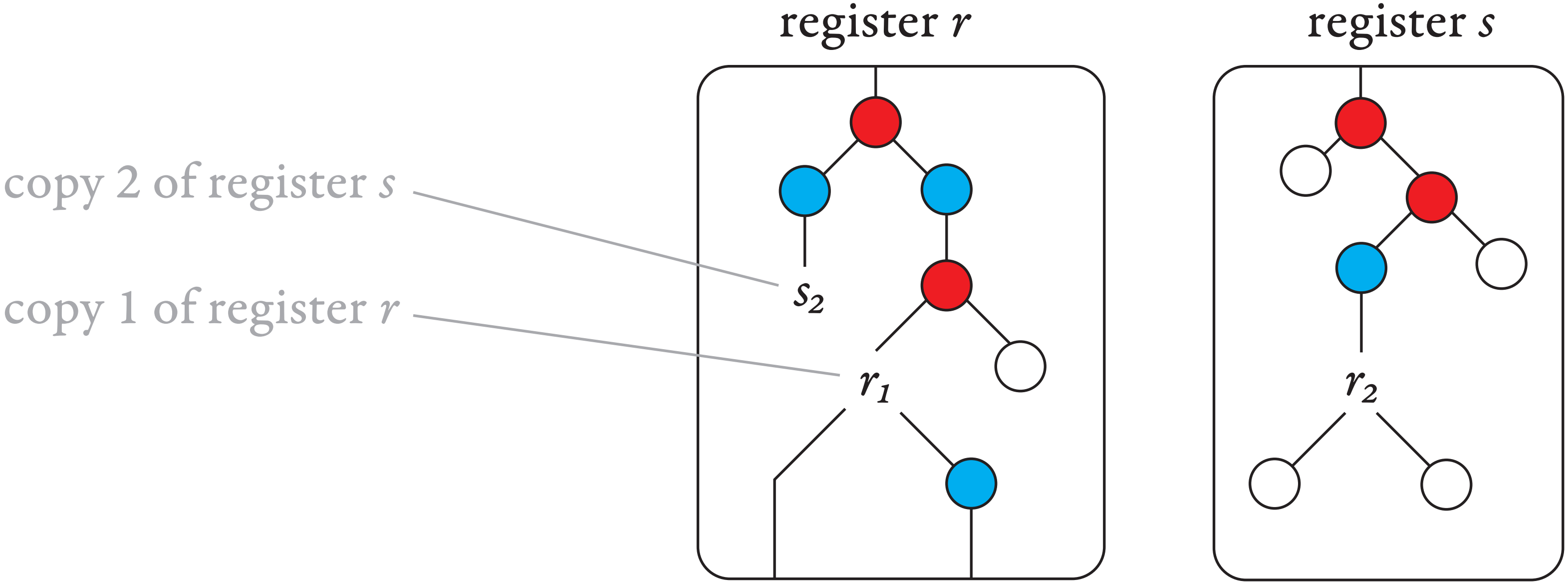


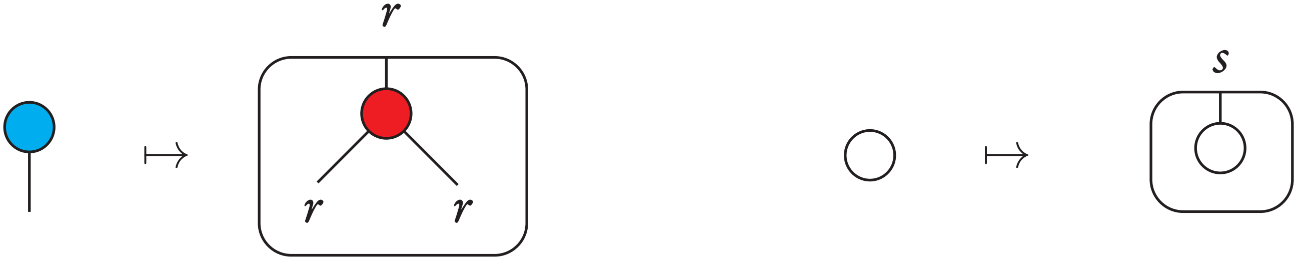
Variable *i.j* represents register *i* in the *j*-th argument of the reigster update.
In the dual, this variable is mapped to the *i*-th edge which enters the *j*-th port of the reducer.


input
output
1
2
4
3
5
6
9
7
8
1
2
3
4
5
6
7
8
9

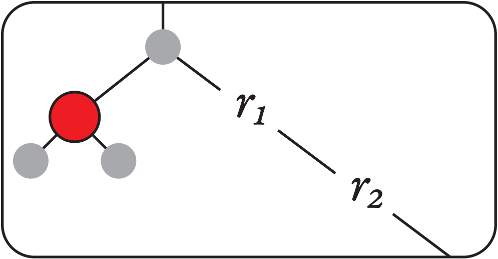
letters of the output alphabet

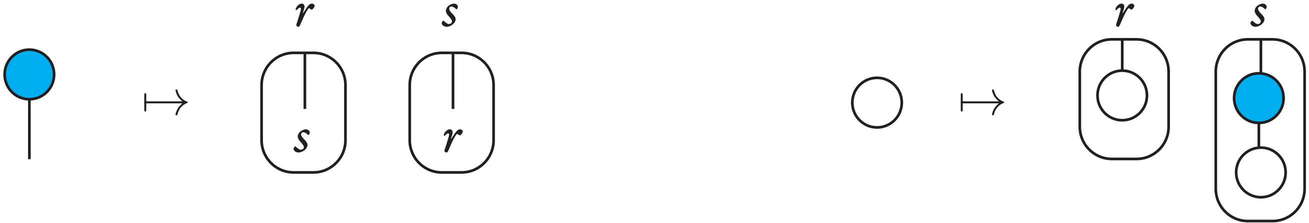


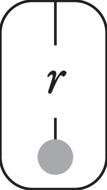

register *r*
register *s*
copy 2 of register *s*
copy 1 of register *r*
s_2
r_1
r_2



↦
r
r
r
↦
s

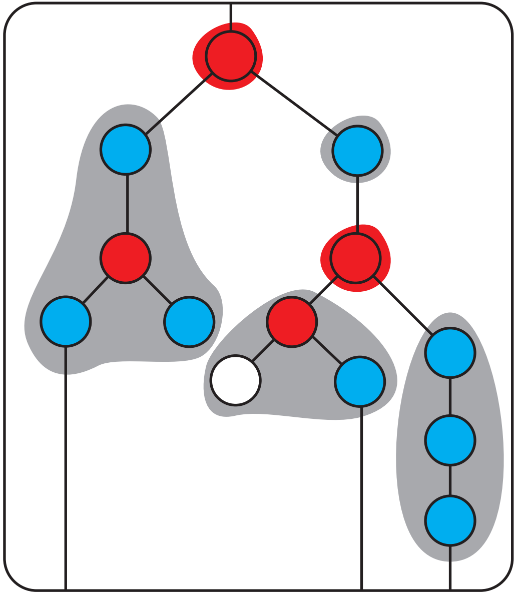

r_1
r_2

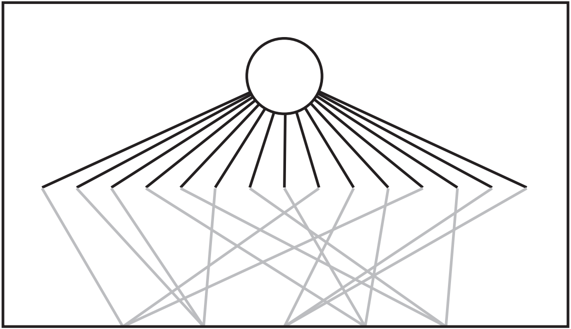

r
s
s
r
r
s

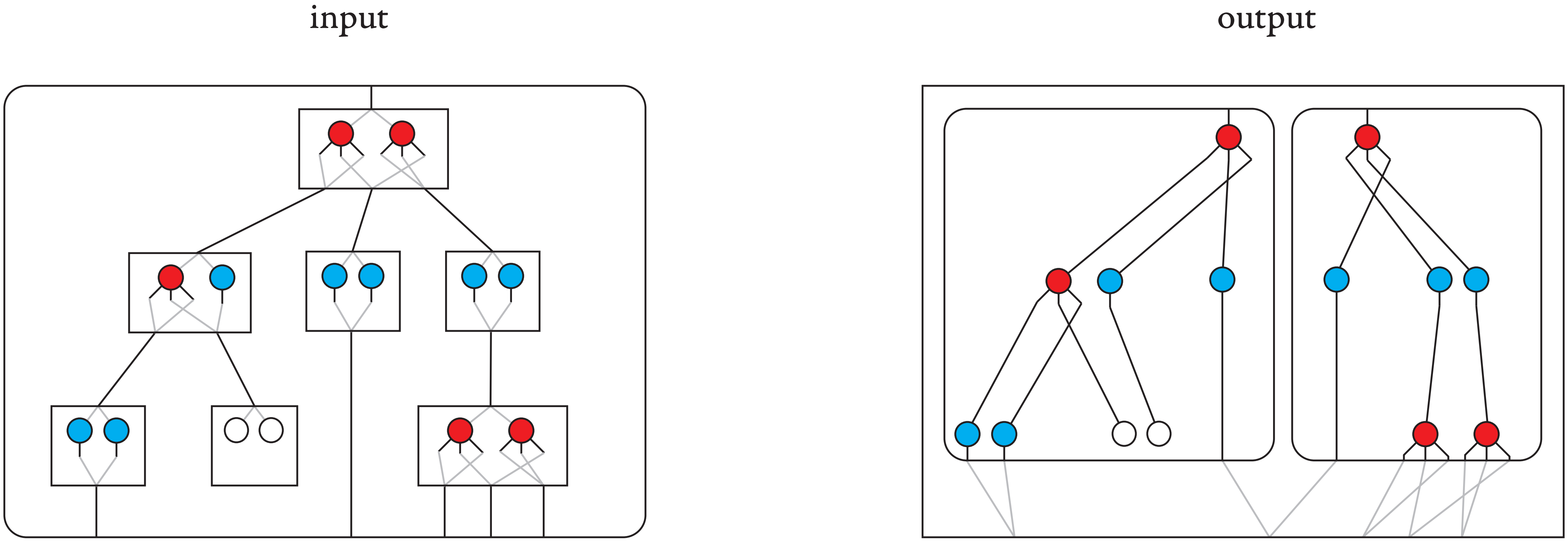

r

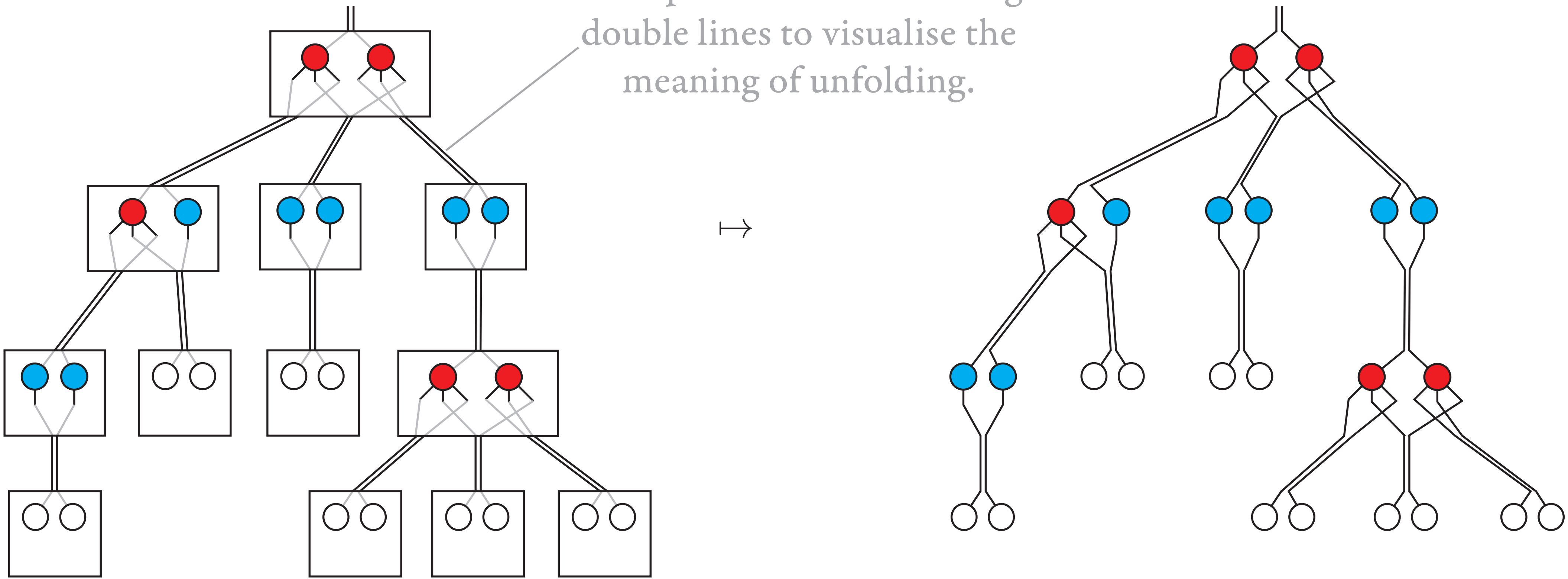
▨ factors without branching nodes

🟥 factors with branching nodes

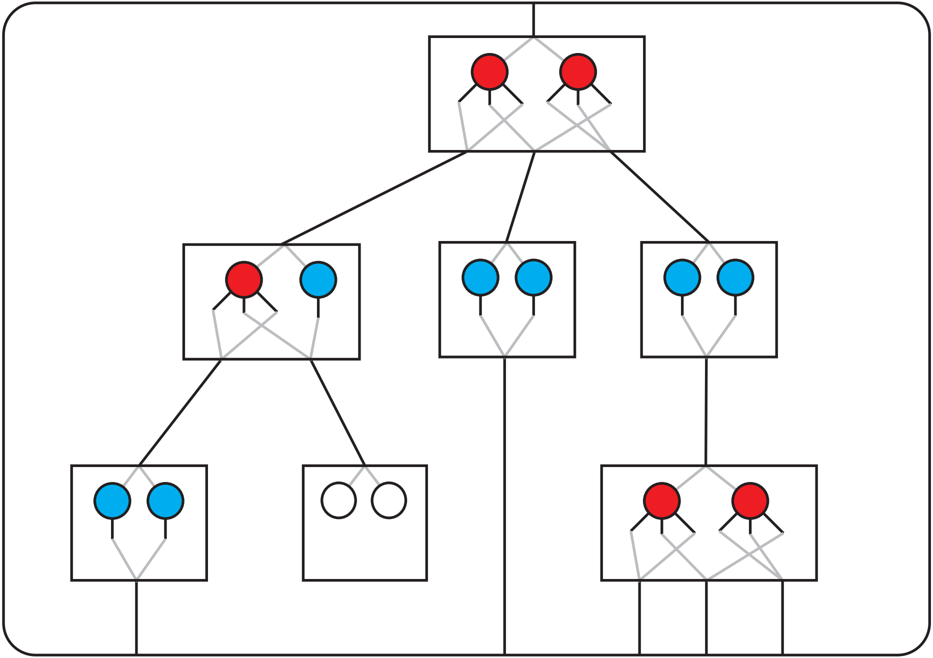


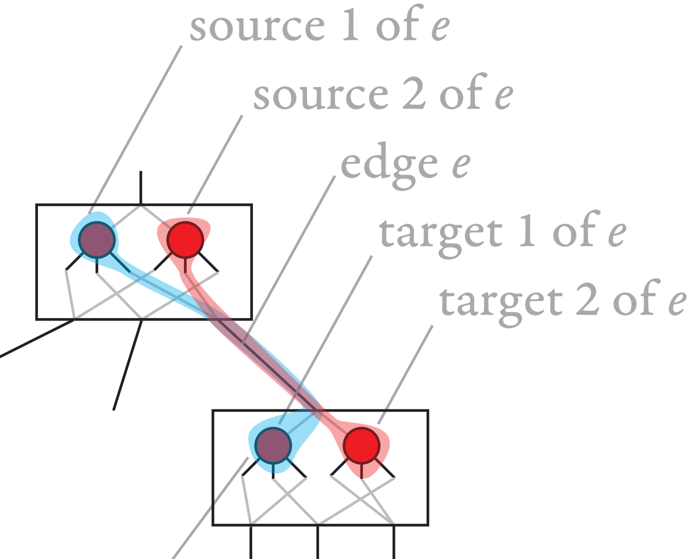


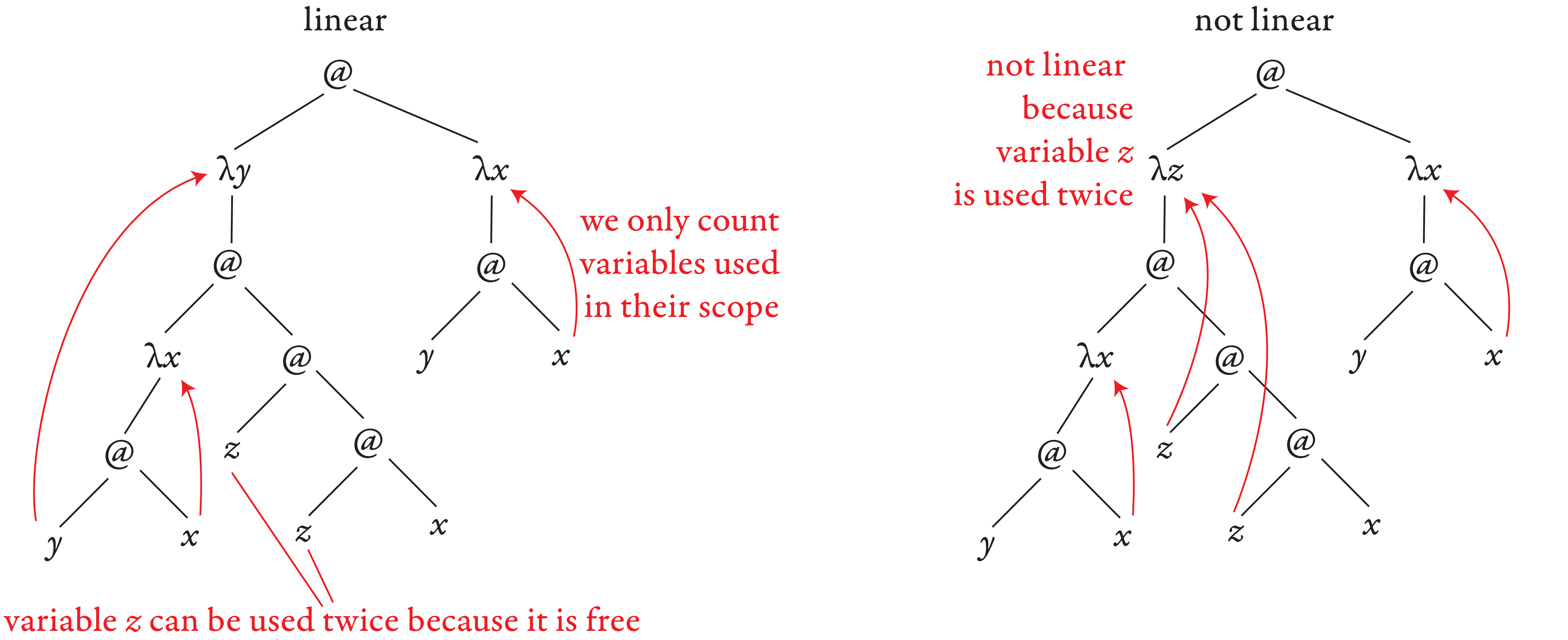

input
output



the parent-child relation in the input tree is drawn using double lines to visualise the meaning of unfolding.




source 1 of *e*
source 2 of *e*
edge *e*
target 1 of *e*
target 2 of *e*



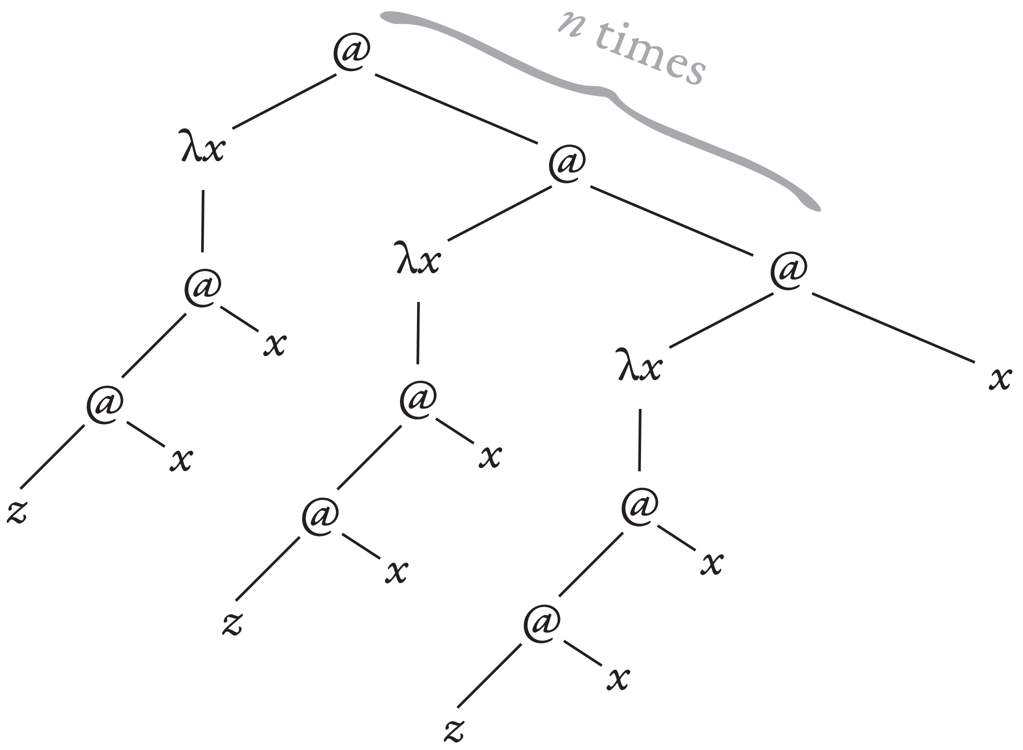
linear

not linear

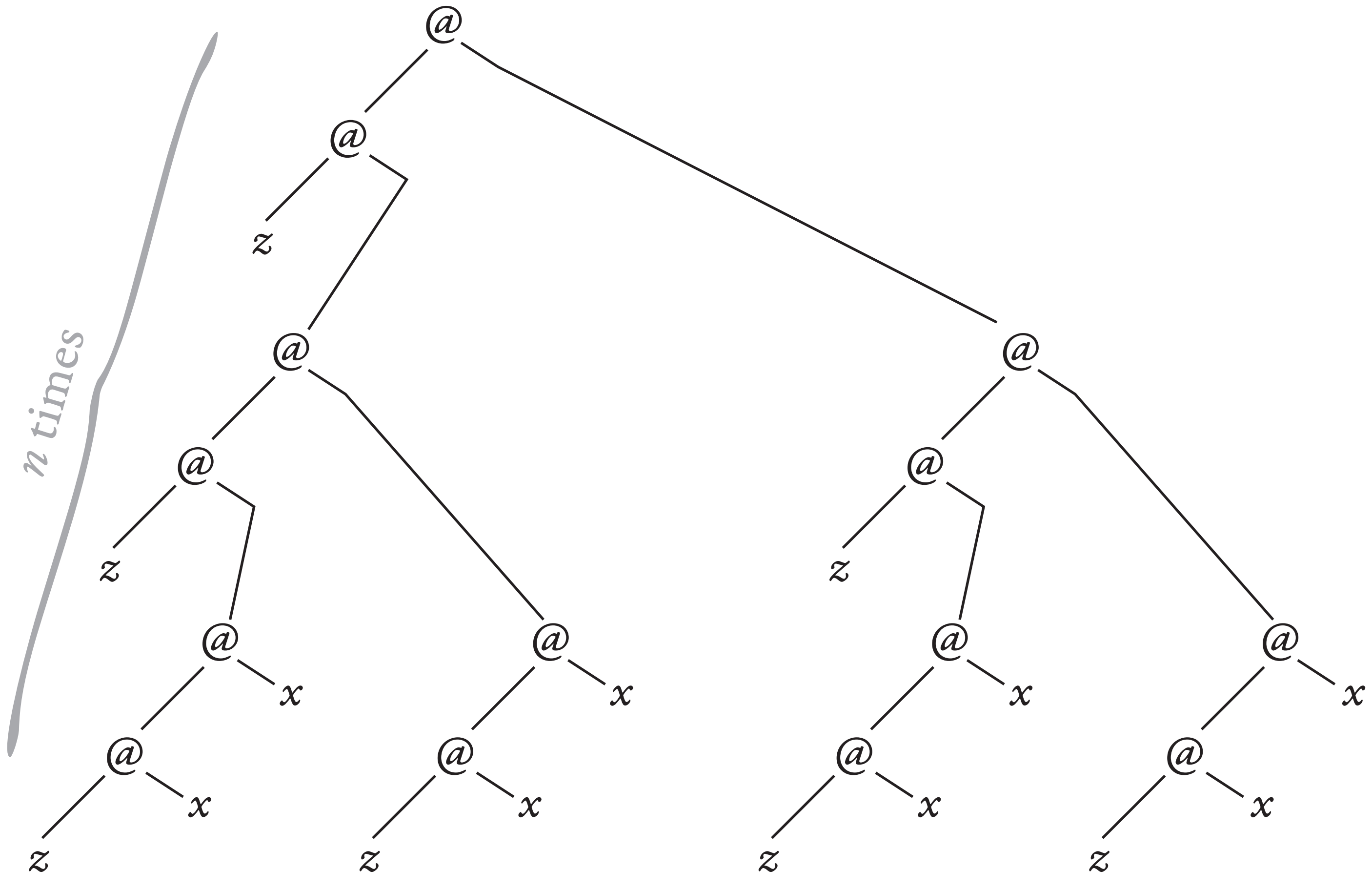
we only count variables used in their scope

variable z can be used twice because it is free

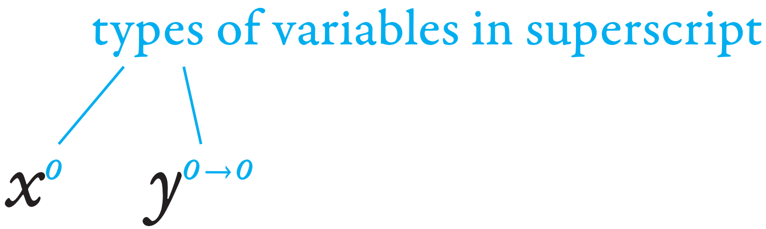
not linear because variable z is used twice a λ-term of size O(n)



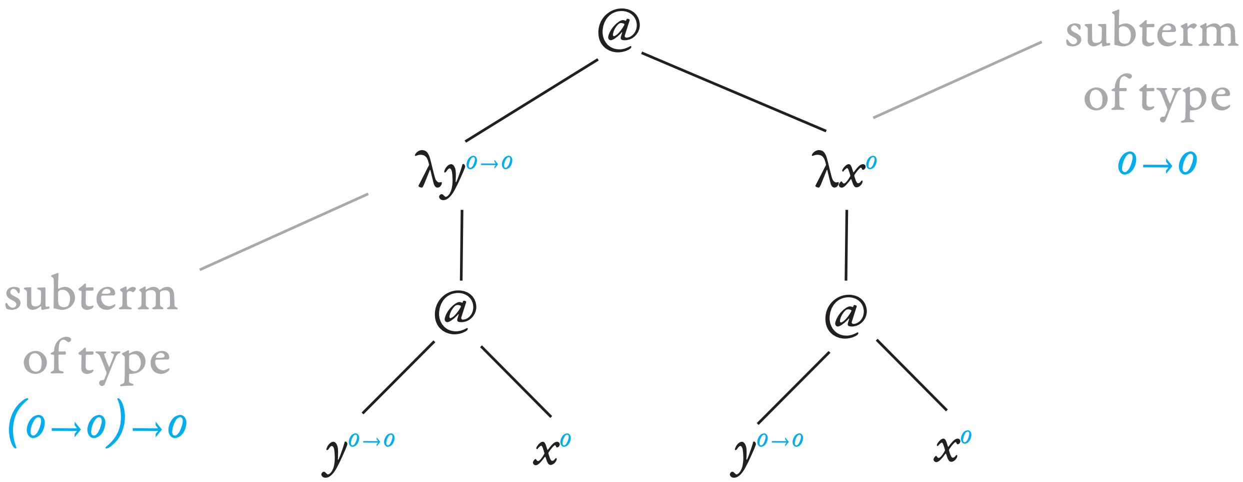
its normal form of size O(2^n)

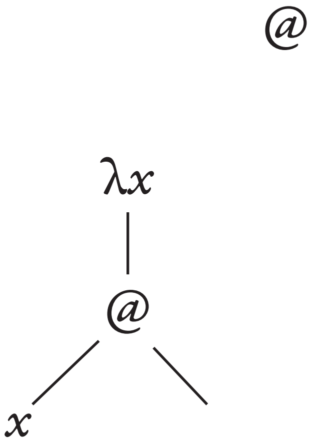


variables

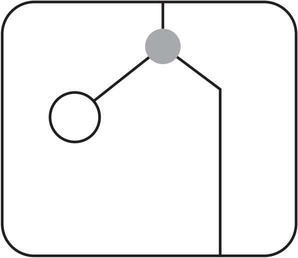


λ-term of type o

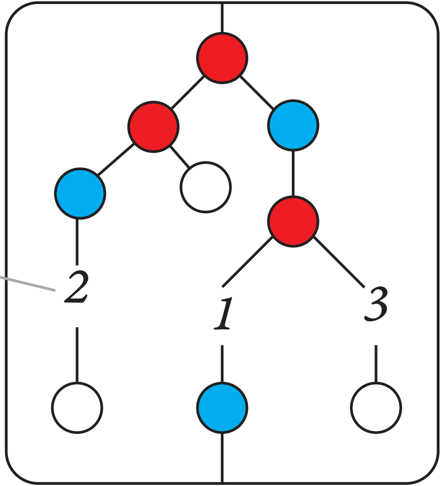




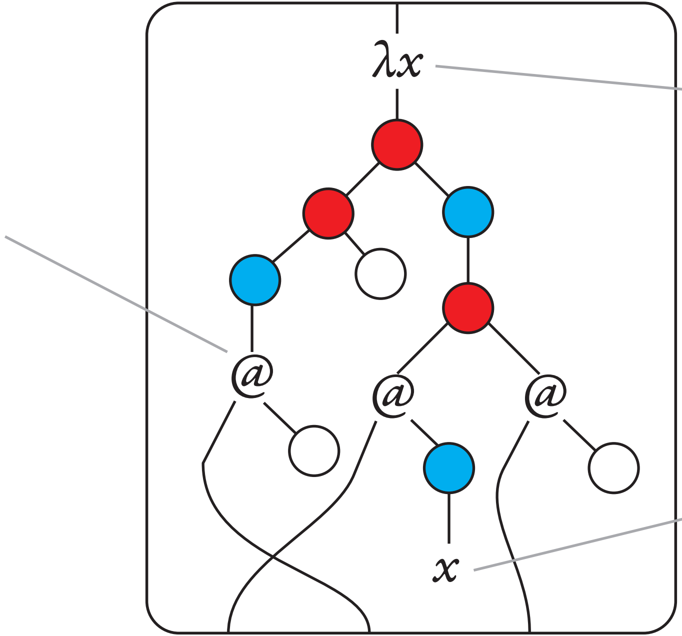
$\lambda x.$



r

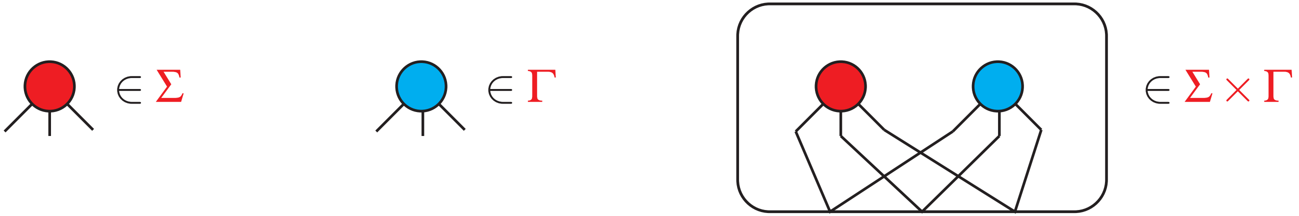


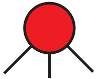
placeholder for the term
stored in the unique register
of the 2nd child

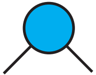


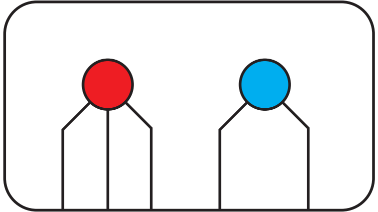
variable *x* is bound in the root

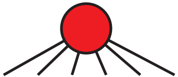
the original port is replaced by *x*


∈ Σ
∈ Γ
∈ Σ × Γ

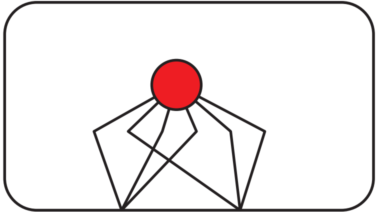
 $\in \Sigma$

 $\in \Gamma$

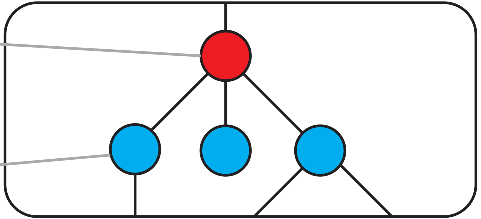
 $\in \Sigma \otimes \Gamma$



$\in \Sigma$

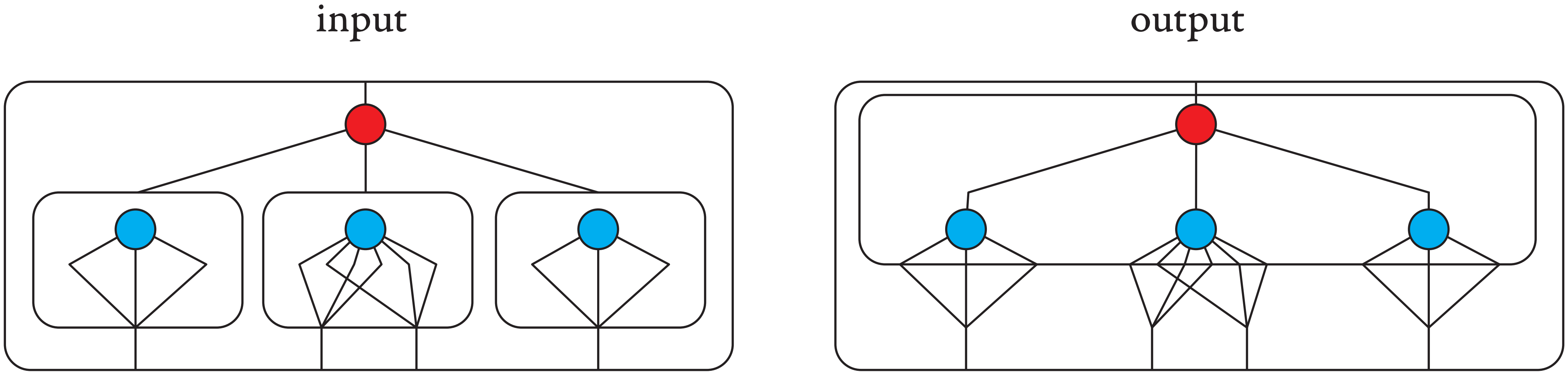


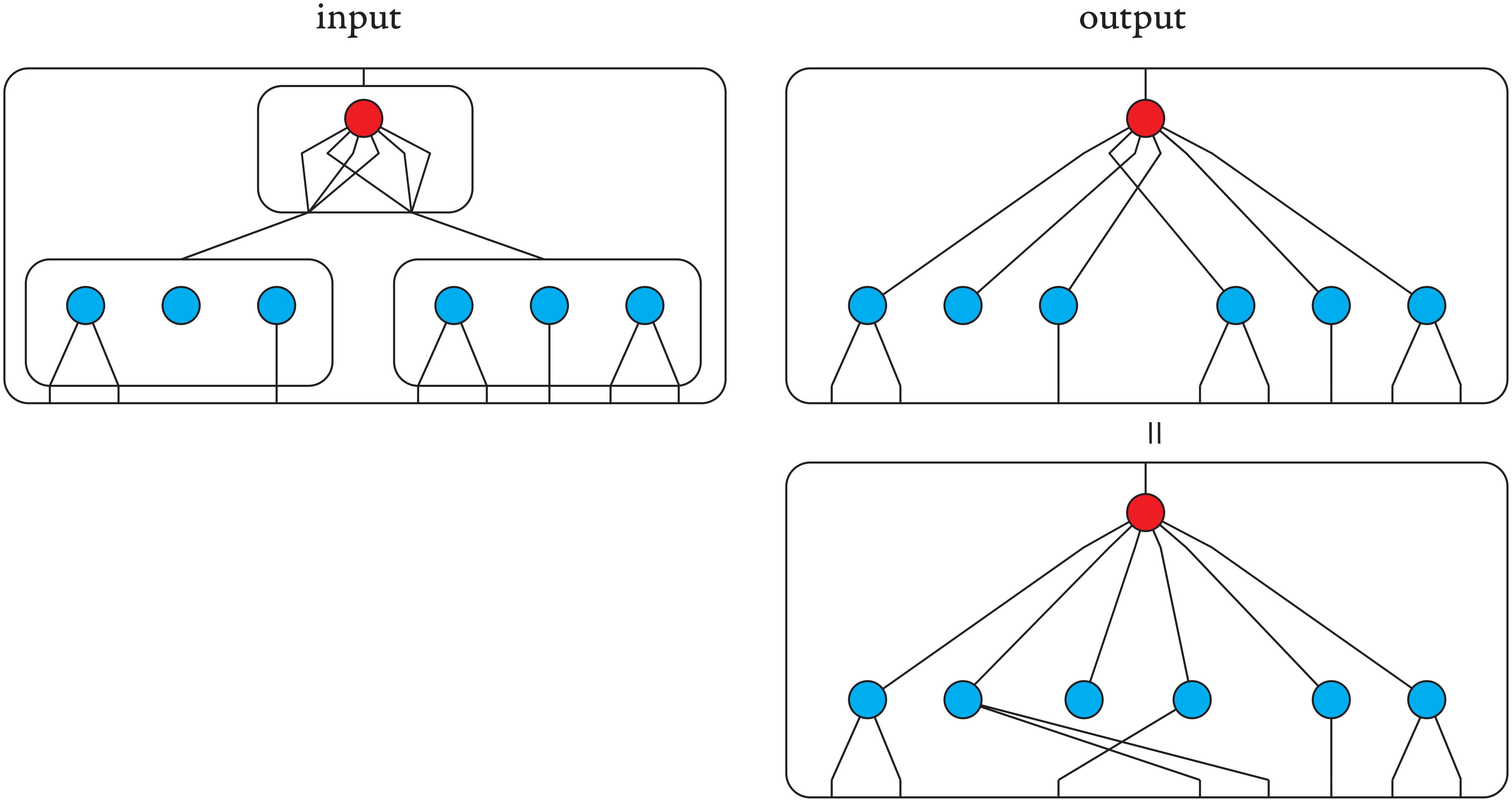
$\in F_3\Sigma$



the root is from Σ

all children are from Γ


input
output


input
output
=