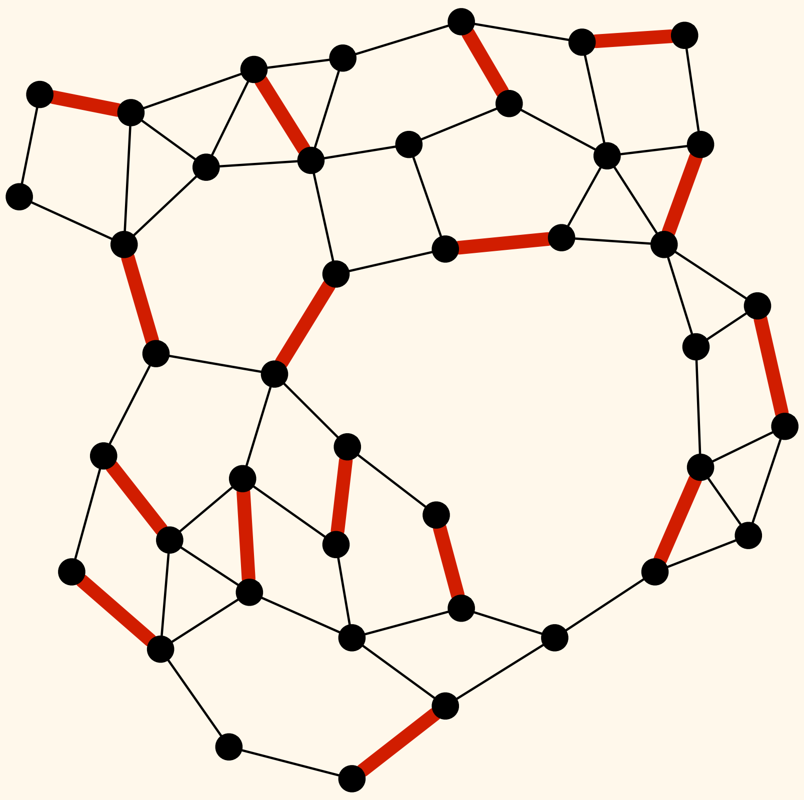
Distributed Maximal Matching: Greedy is Optimal

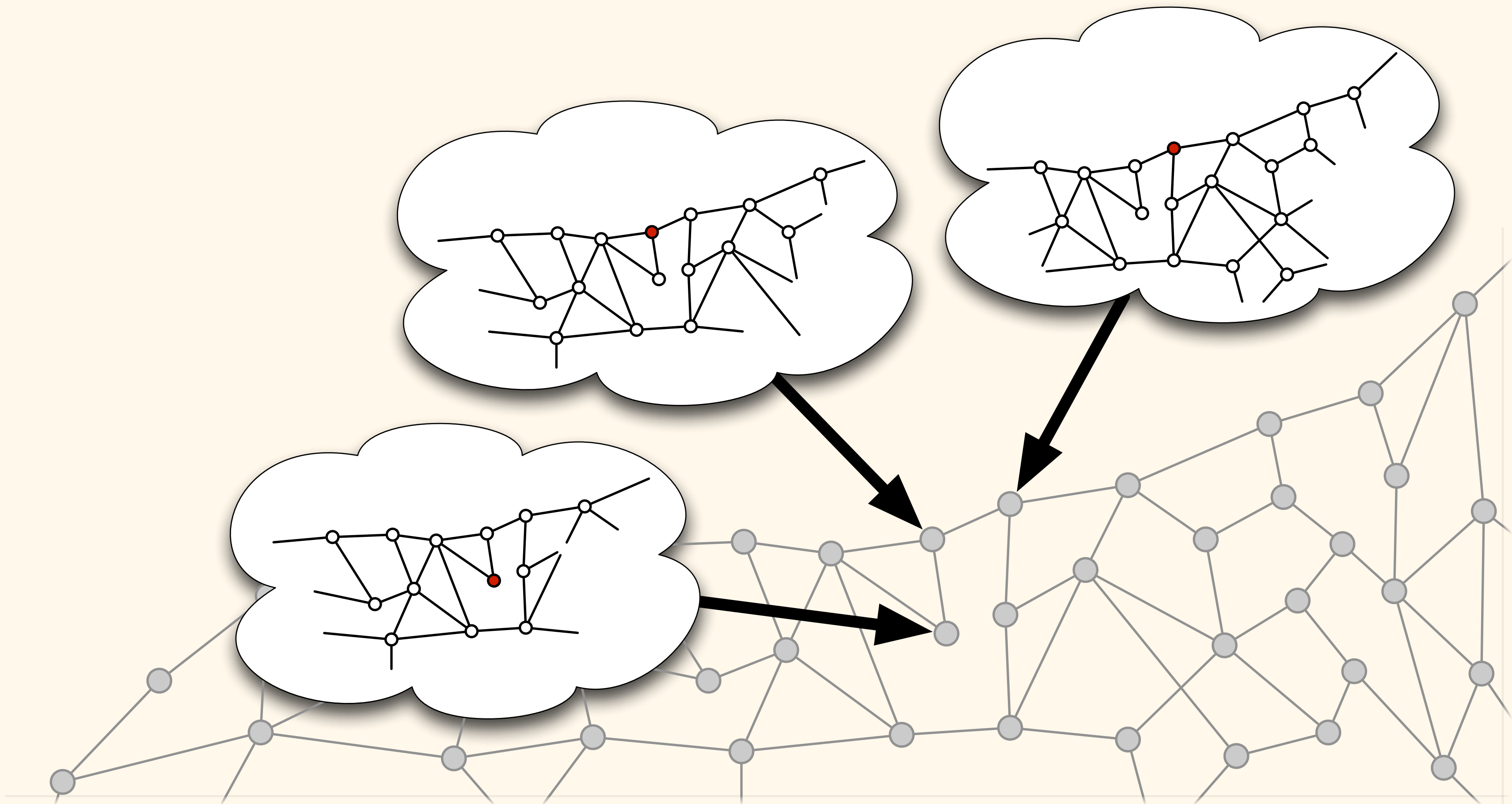
Jukka Suomela
Helsinki Institute for Information Technology HIIT
University of Helsinki

Joint work with **Juho Hirvonen**

Weizmann Institute of Science, 27 November 2011

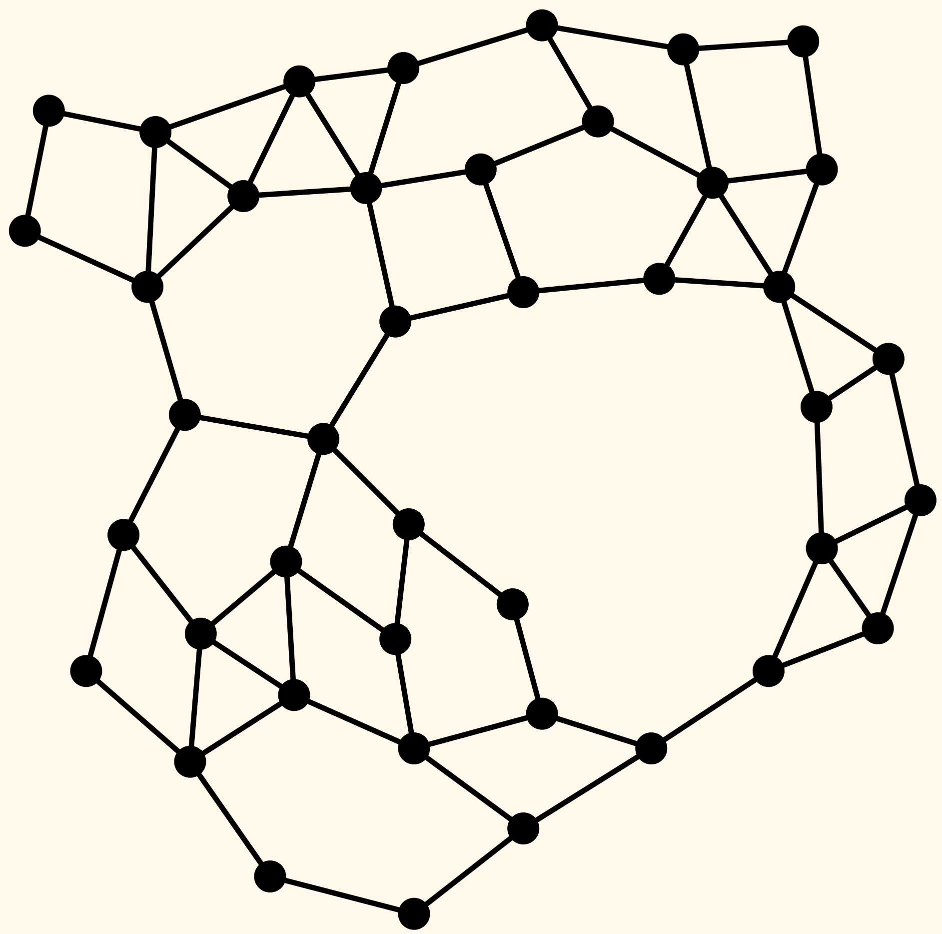


Background

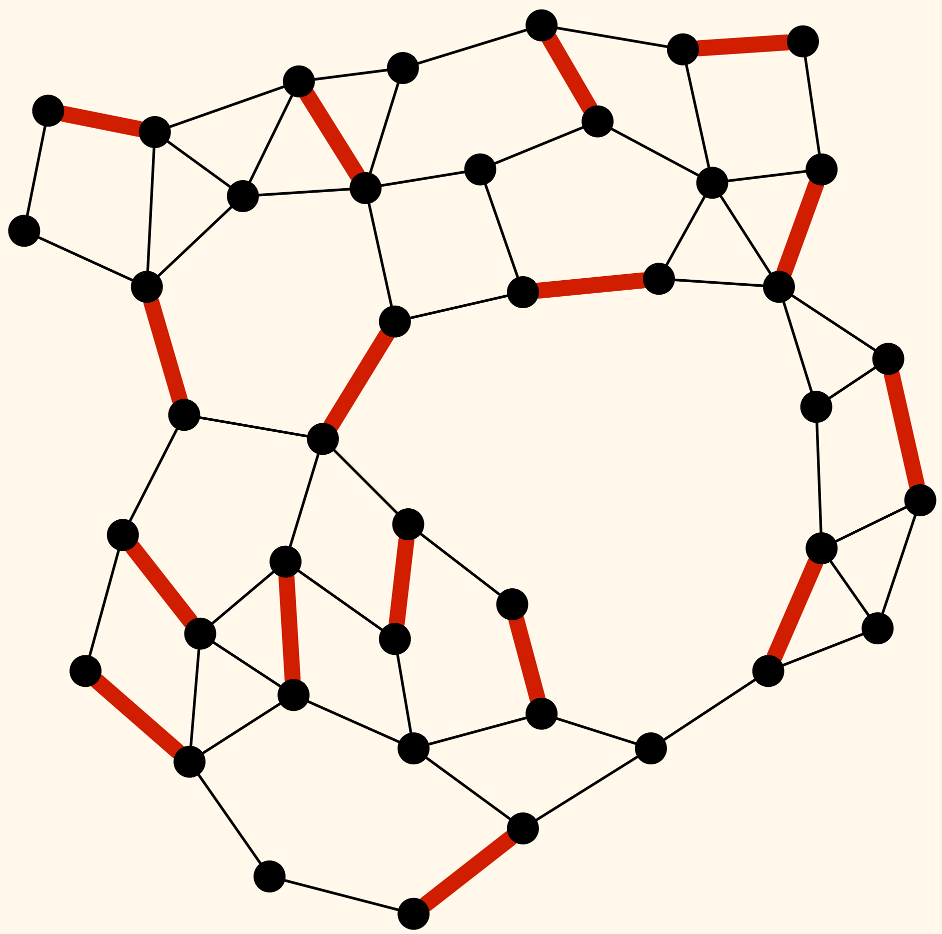


Maximal Matchings

Input

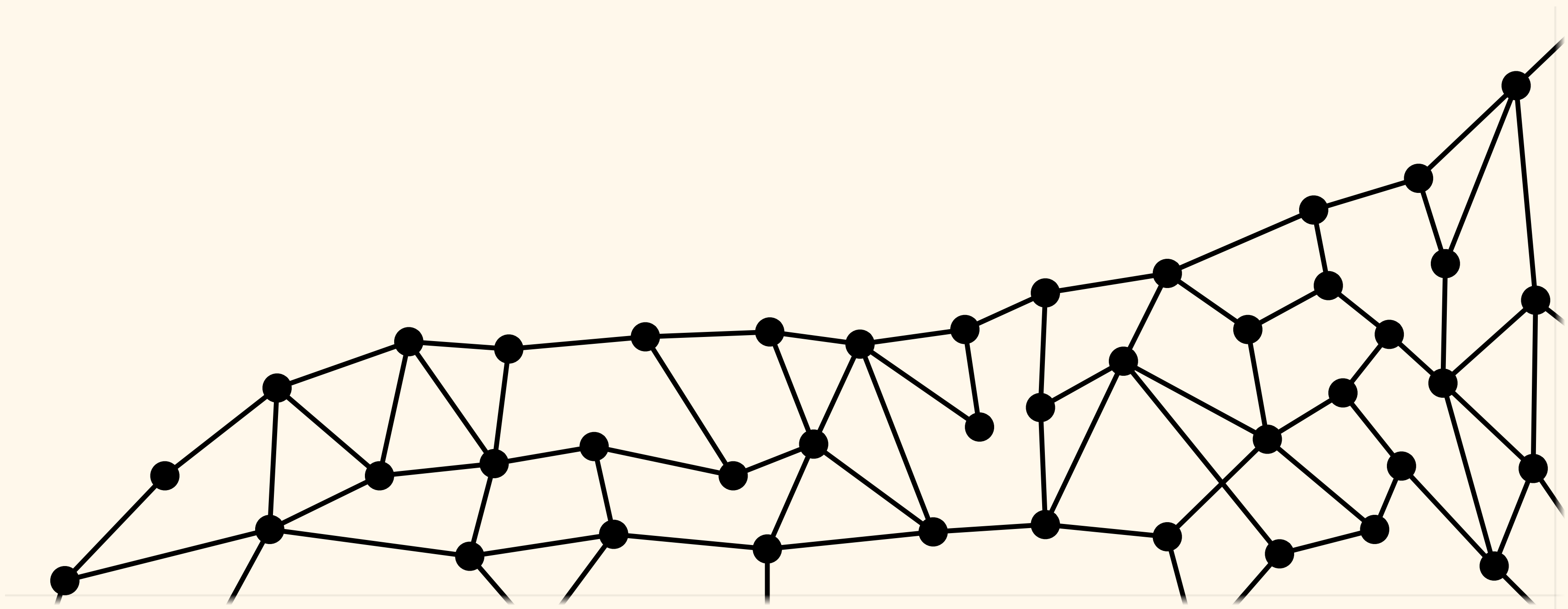


Output



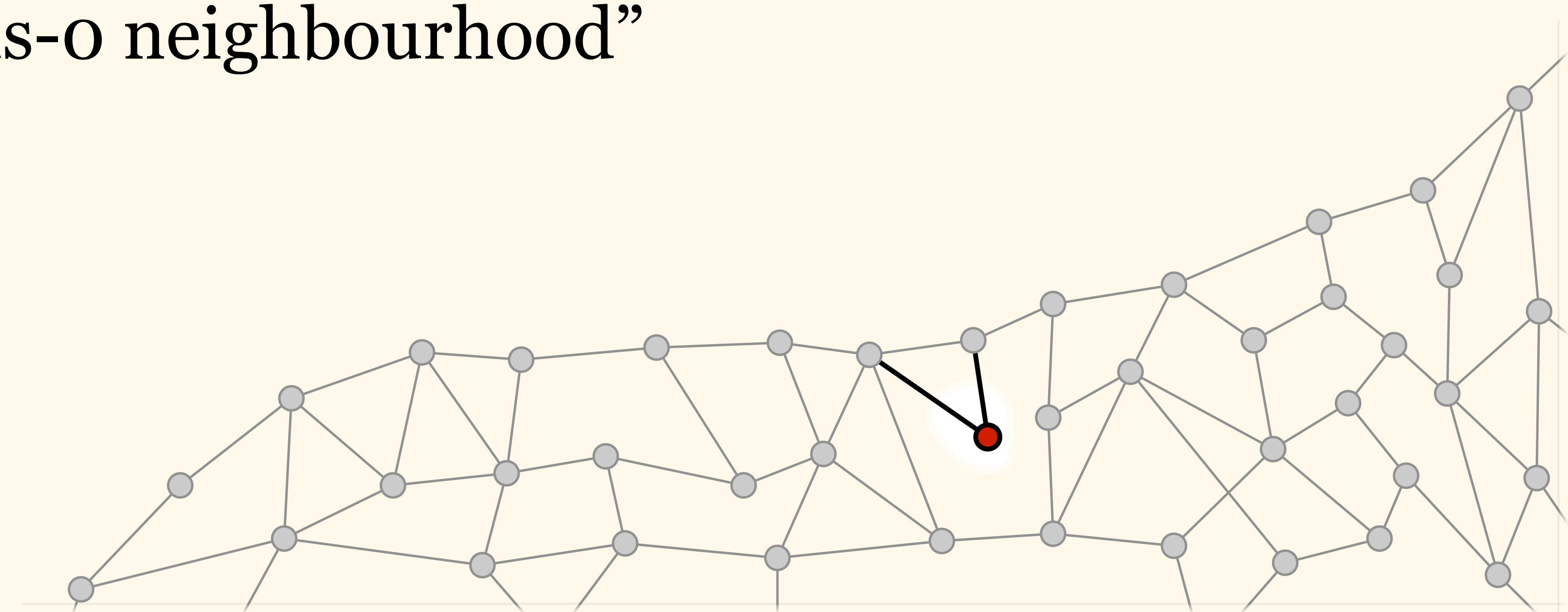
Distributed Algorithms

- Graph G = input = communication network
  - node = computer
  - edge = communication link



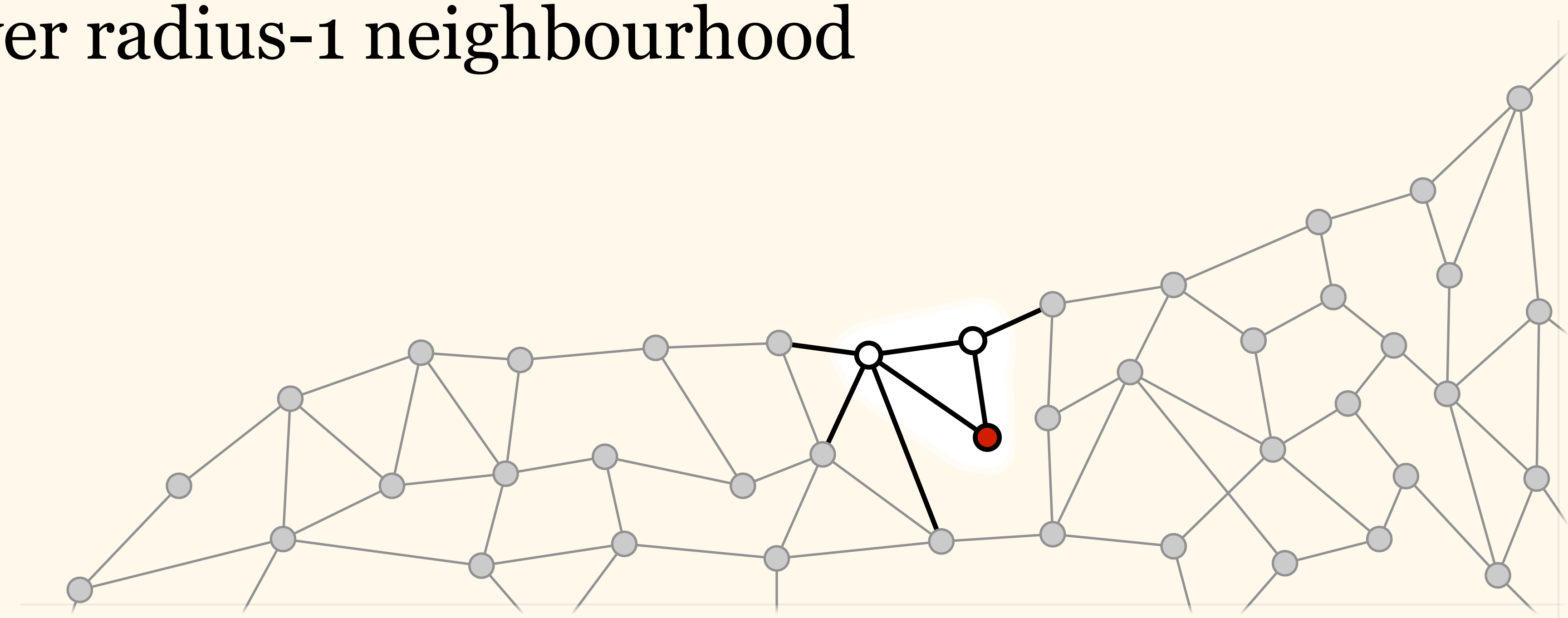
Distributed Algorithms

- Initially, each node only knows its incident edges
  - i.e., each node knows its "radius-0 neighbourhood"



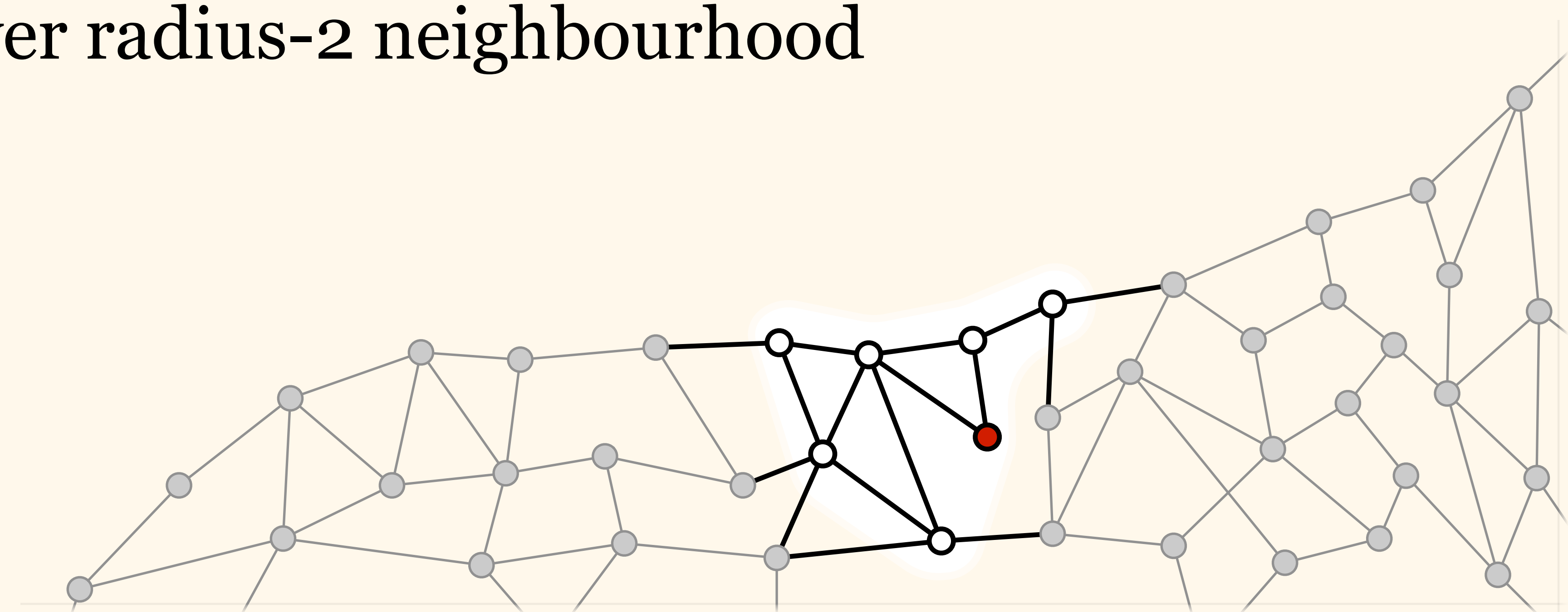
Distributed Algorithms

- Nodes can exchange messages to learn more about graph *G*…
    - 1 communication round: discover radius-1 neighbourhood



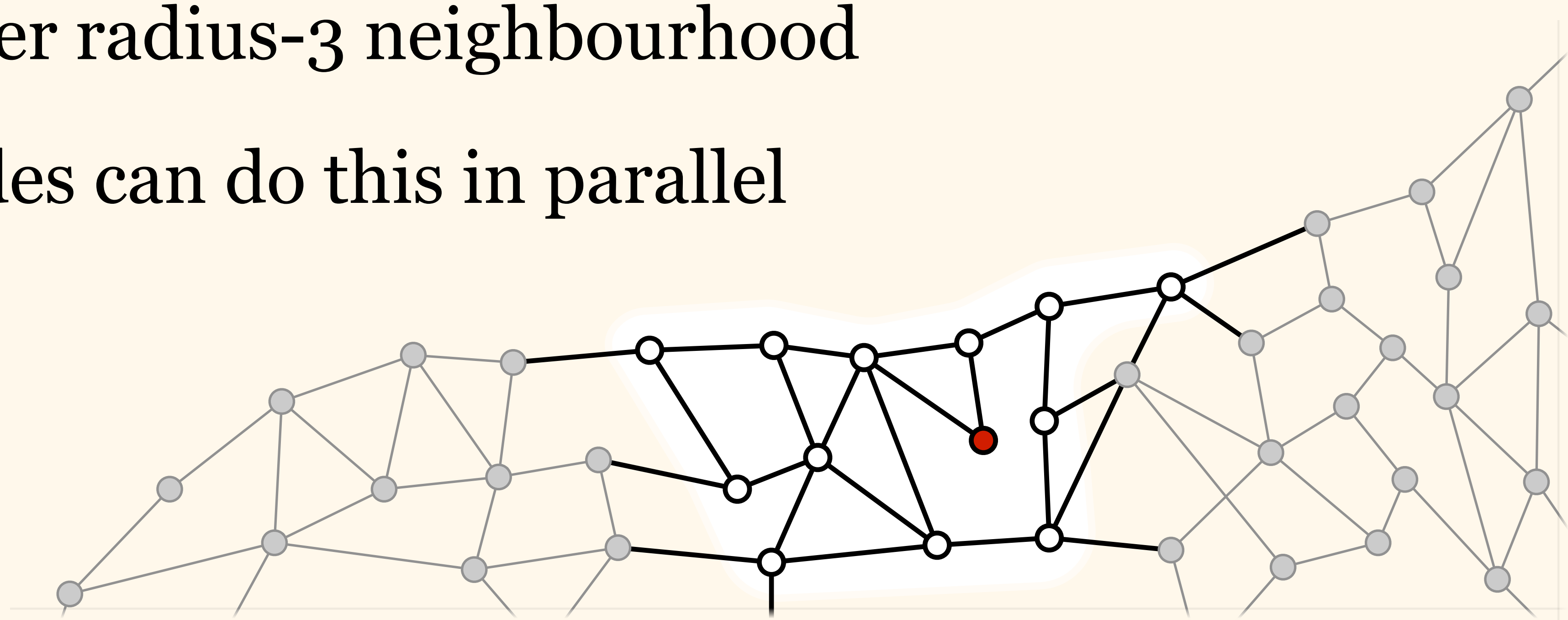
Distributed Algorithms

- Nodes can exchange messages to learn more about graph G...
  - 2 communication rounds: discover radius-2 neighbourhood



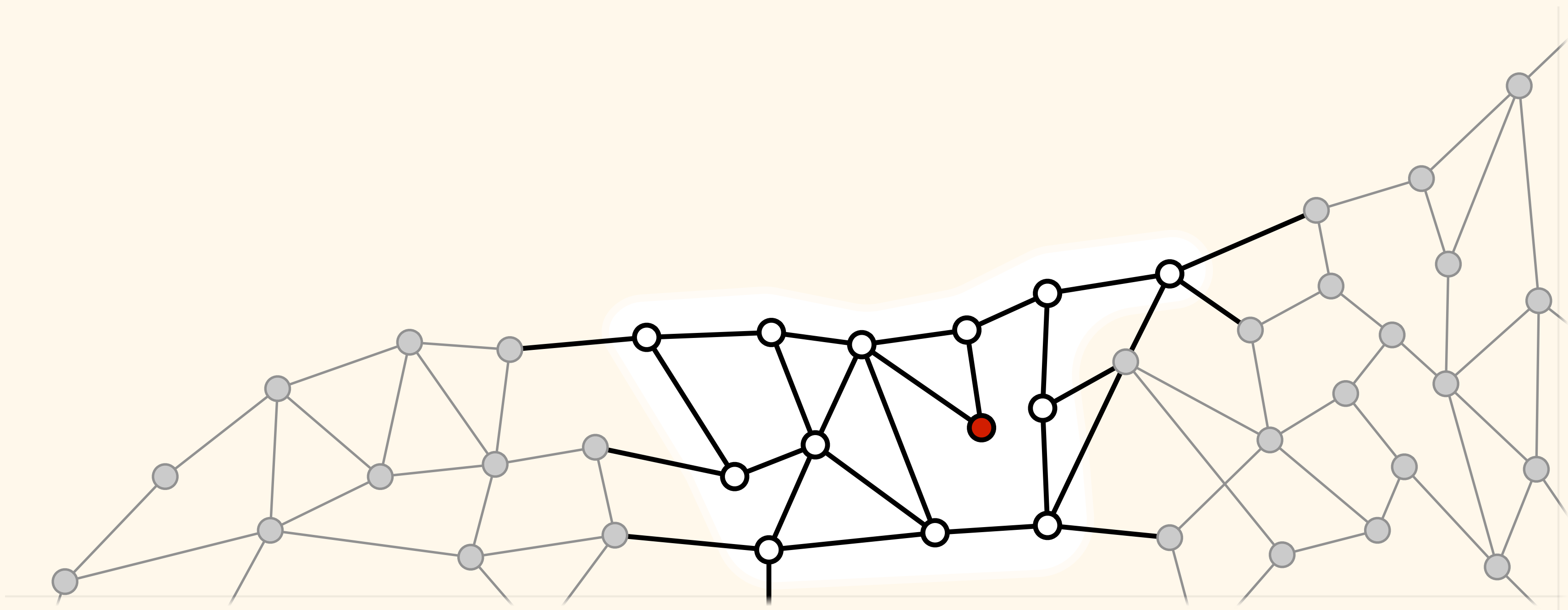
Distributed Algorithms

- Nodes can exchange messages to learn more about graph G...
    - 3 communication rounds: discover radius-3 neighbourhood
    - all nodes can do this in parallel



Distributed Algorithms

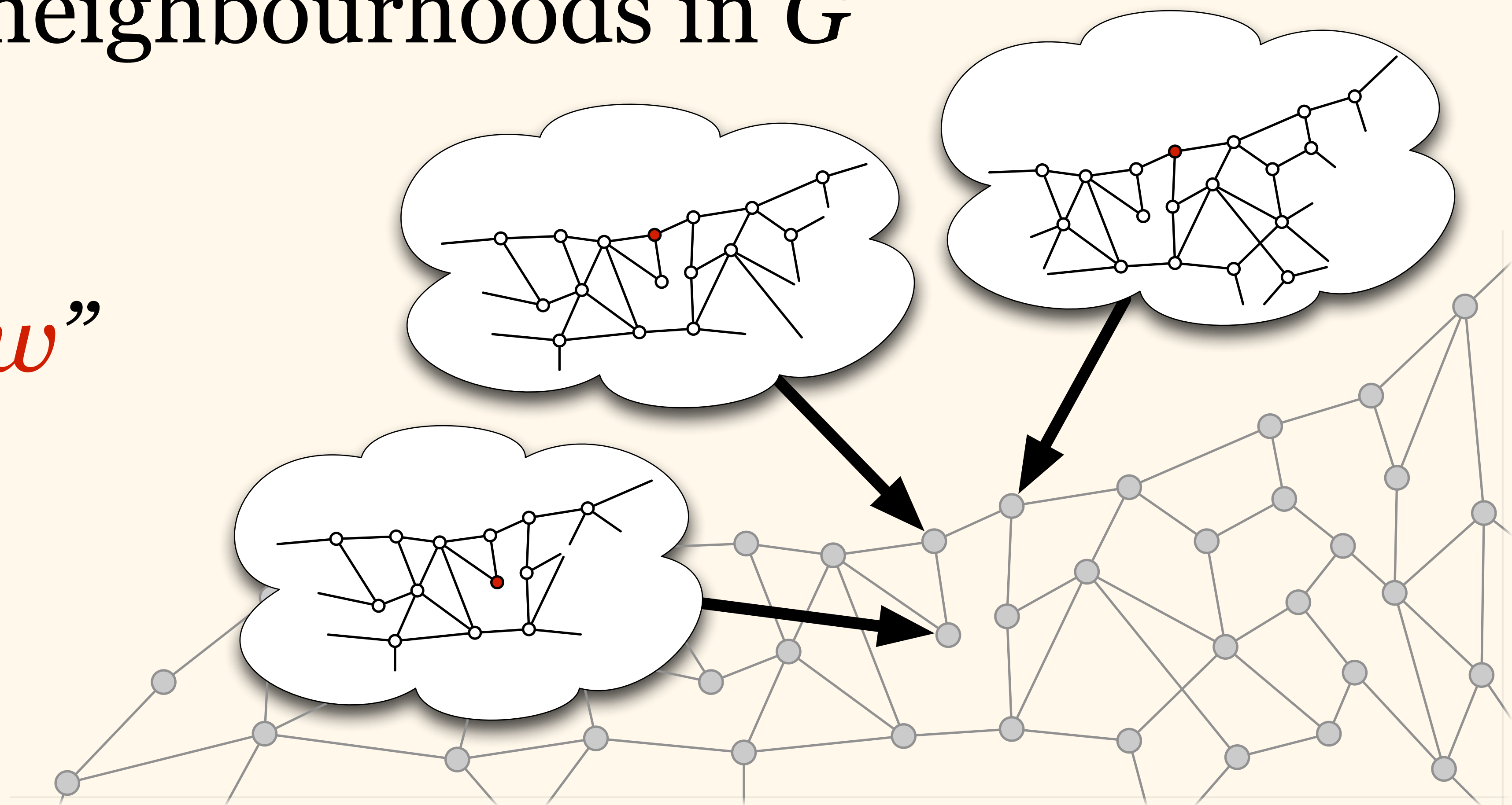
After T rounds, all nodes know their radius-T neighbourhoods in G



Distributed Algorithms

After T rounds, all nodes know their radius-T neighbourhoods in G

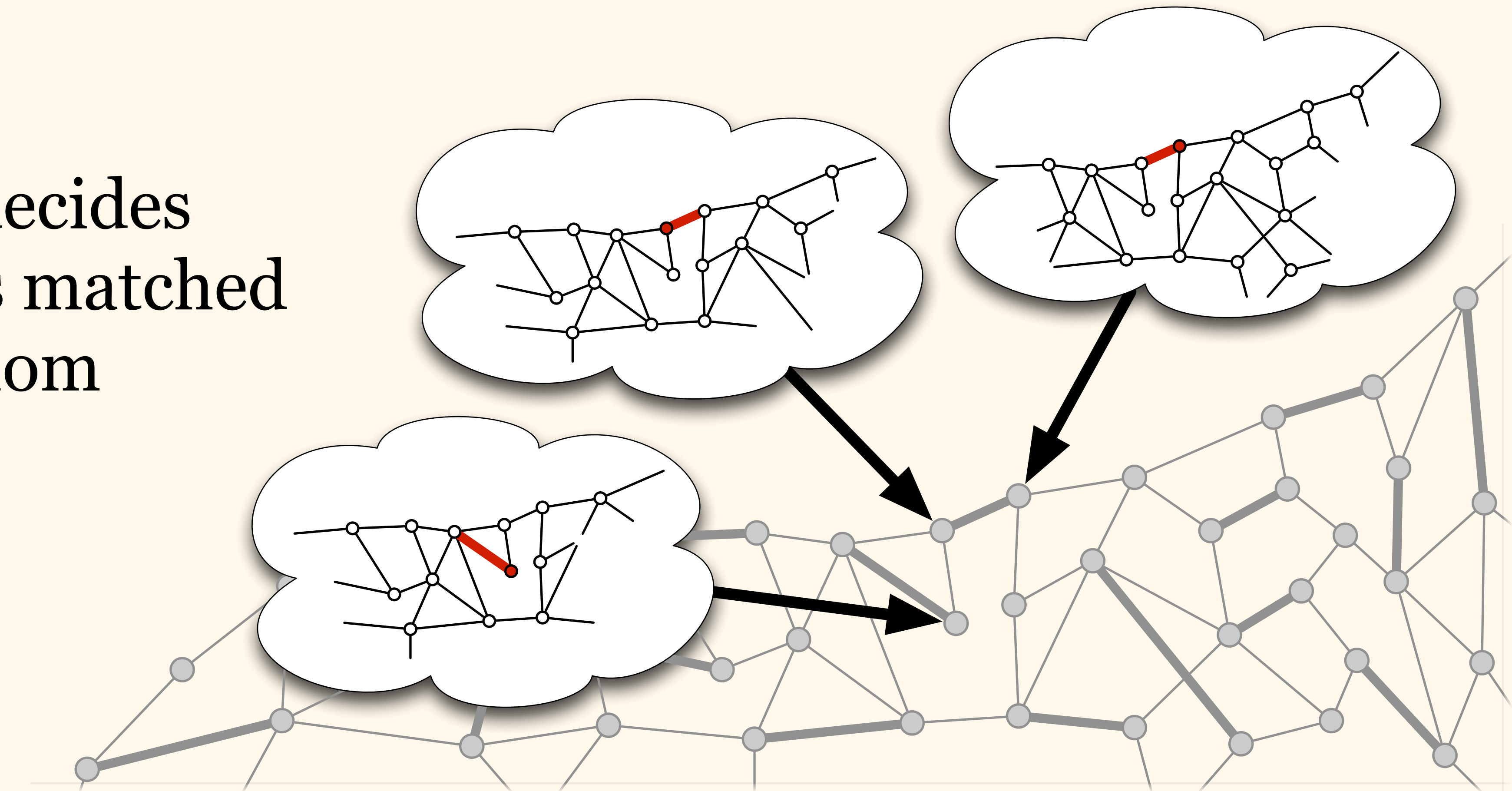
“*local view*”



Distributed Algorithms

Mapping: *local view* ↦ *local output*

Each node decides whether it is matched and with whom



Distributed Algorithms

- Time = number of communication rounds
- Equivalent:
  - Distributed algorithm that runs in time T
  - All nodes run the same algorithms;
    after T synchronous communication rounds
    all nodes announce their local outputs
  - Mapping from radius-T neighbourhoods
    to local outputs

Distributed Algorithms

- Time = number of communication rounds
- *How fast* can we find a maximal matching?
    - $O(n)$? $O(\log n)$? $O(1)$?

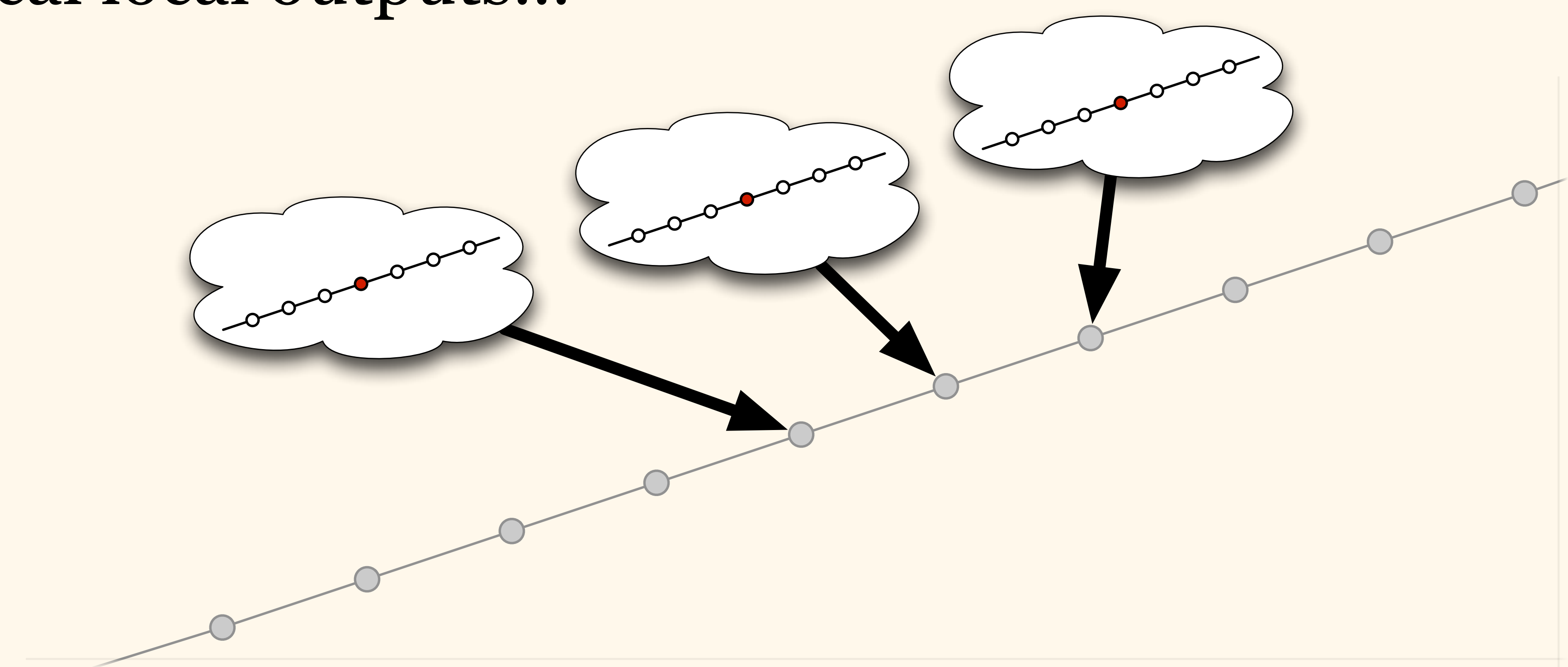
Distributed Algorithms

- Time = number of communication rounds
- *How fast* can we find a maximal matching?
    - $O(n)$? $O(\log n)$? $O(1)$?

- Maybe we should first make sure that we can find a maximal matching *at all*...

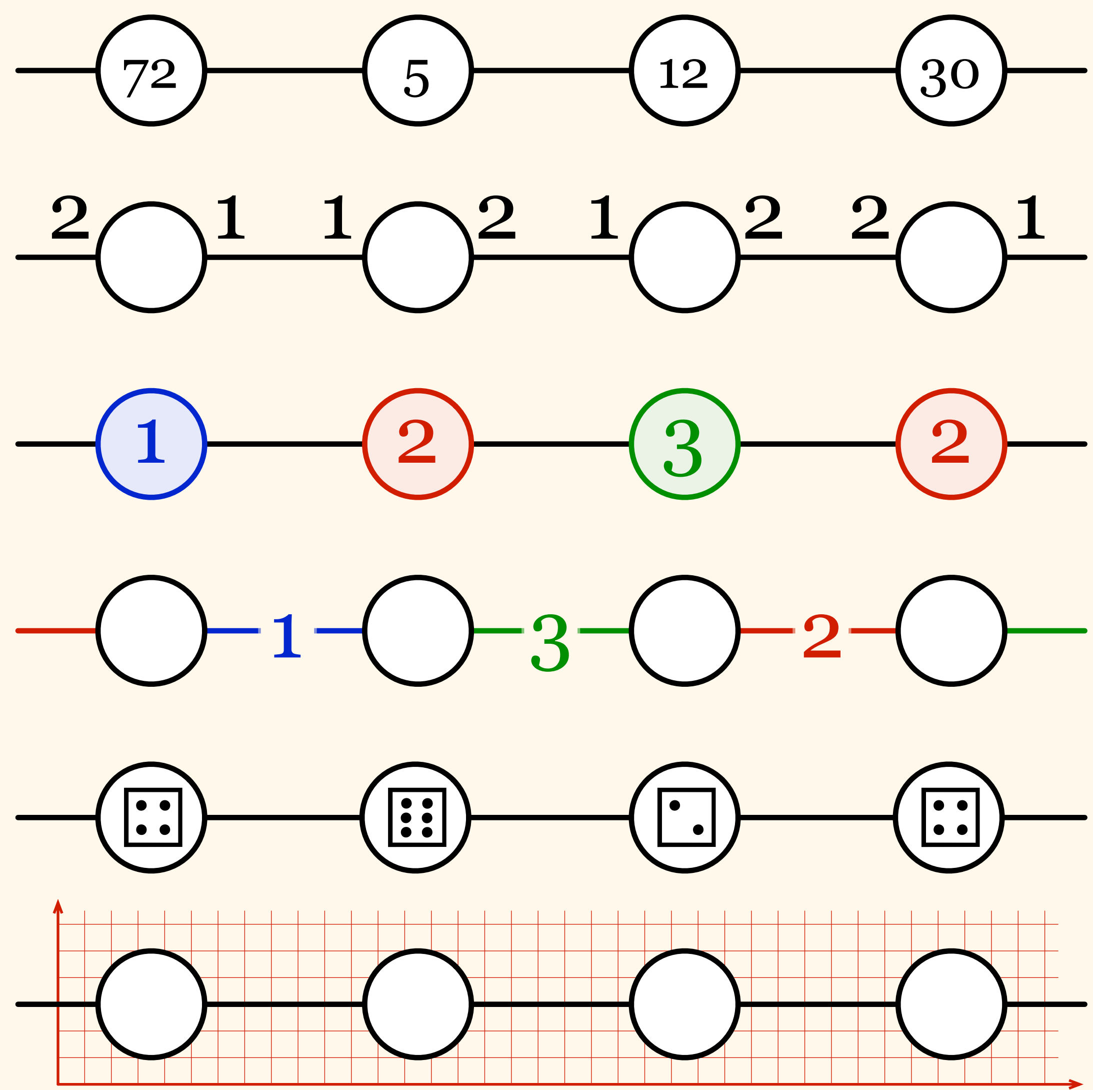
Symmetry Breaking

- Some kind of symmetry-breaking is needed!
  - identical local views,
    identical local outputs…



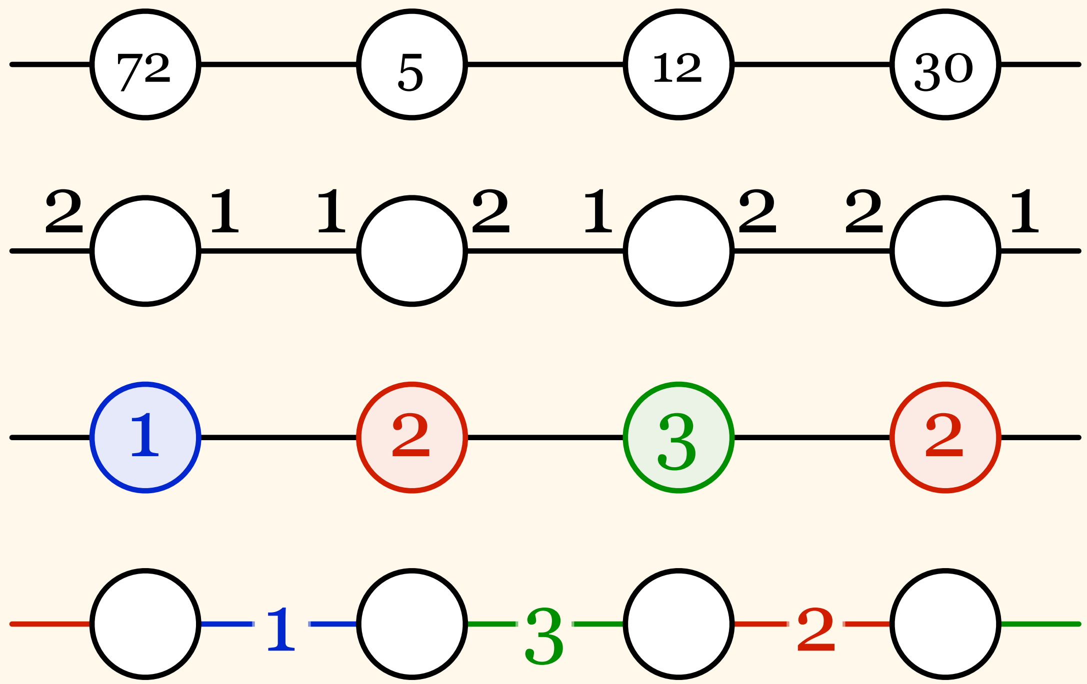
Symmetry Breaking

- Unique identifiers
- Port numbering
- Node colouring
- Edge colouring
- Randomness
- Geometry



Symmetry Breaking

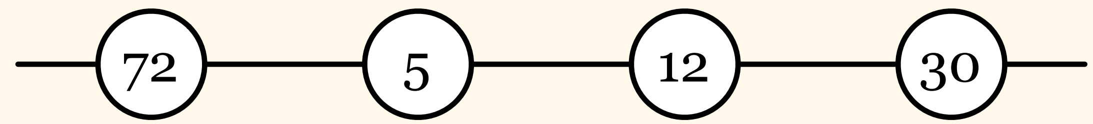
- Unique identifiers
- Port numbering
- Node colouring
- Edge colouring
- Randomness
- Geometry



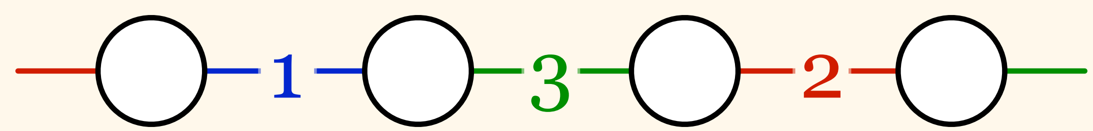
another world...

Symmetry Breaking

- Unique identifiers
- Port numbering
- Node colouring
- Edge colouring
- Randomness
- Geometry



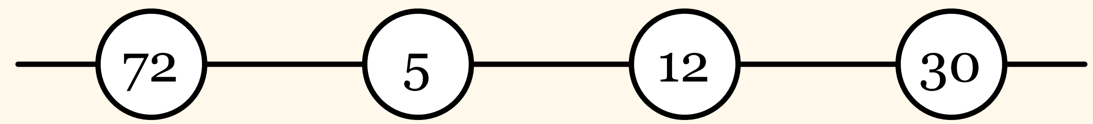
} not enough!



} another world...

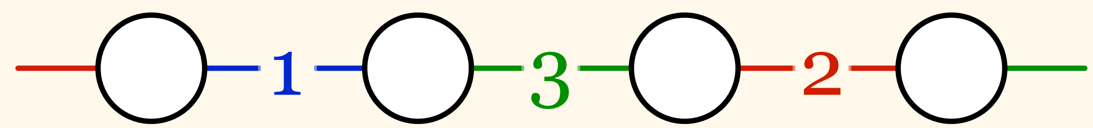
Symmetry Breaking

- Unique identifiers



  - n nodes: identifiers subset of {1, 2, ... poly(n)}
  - I will refer to this when discussing related work

- **Edge colouring**

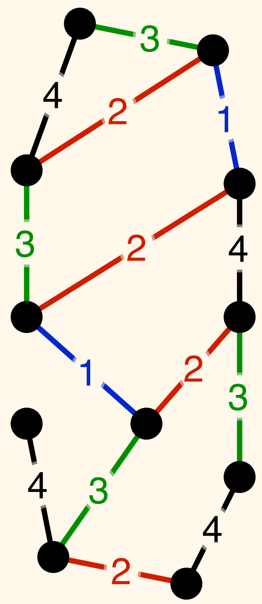


  - proper k-colouring of edges
  - enough for our purposes—this what we use today

Greedy Algorithm

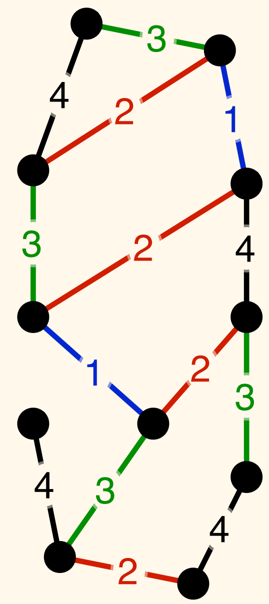
- Given: *k*-edge coloured graph

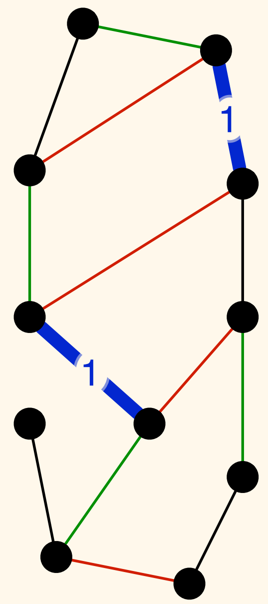
Input



Greedy Algorithm

- Greedily add edges of colour 1, …

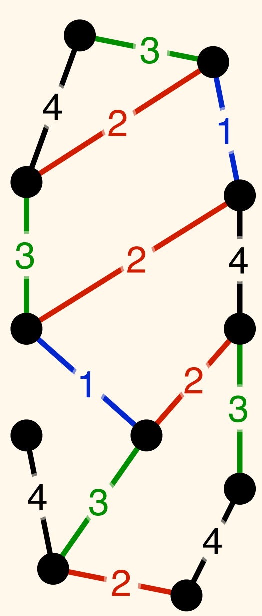




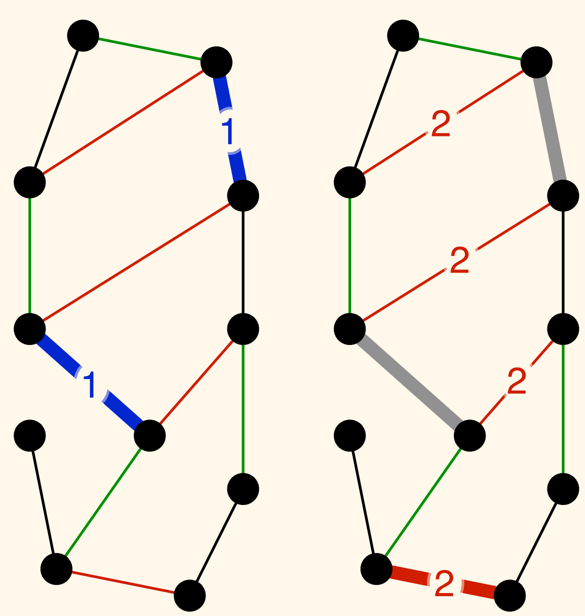
Greedy Algorithm

- Greedily add edges of colour 1, 2, ...

Input



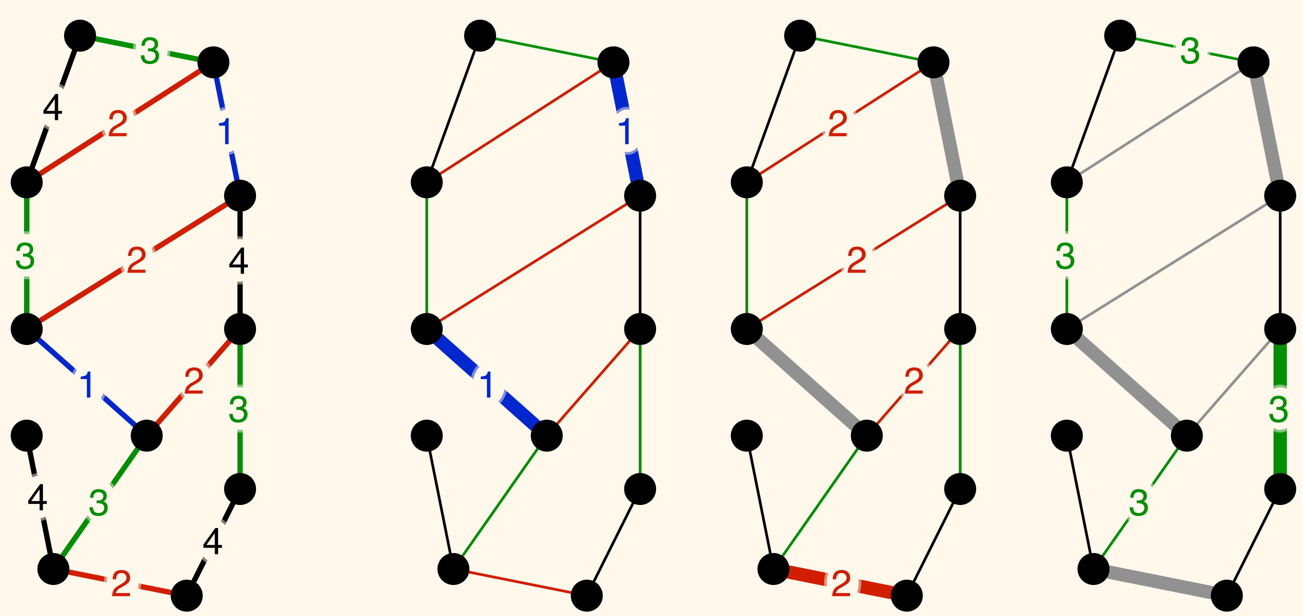
Greedy algorithm



Greedy Algorithm

- Greedily add edges of colour 1, 2, 3, …

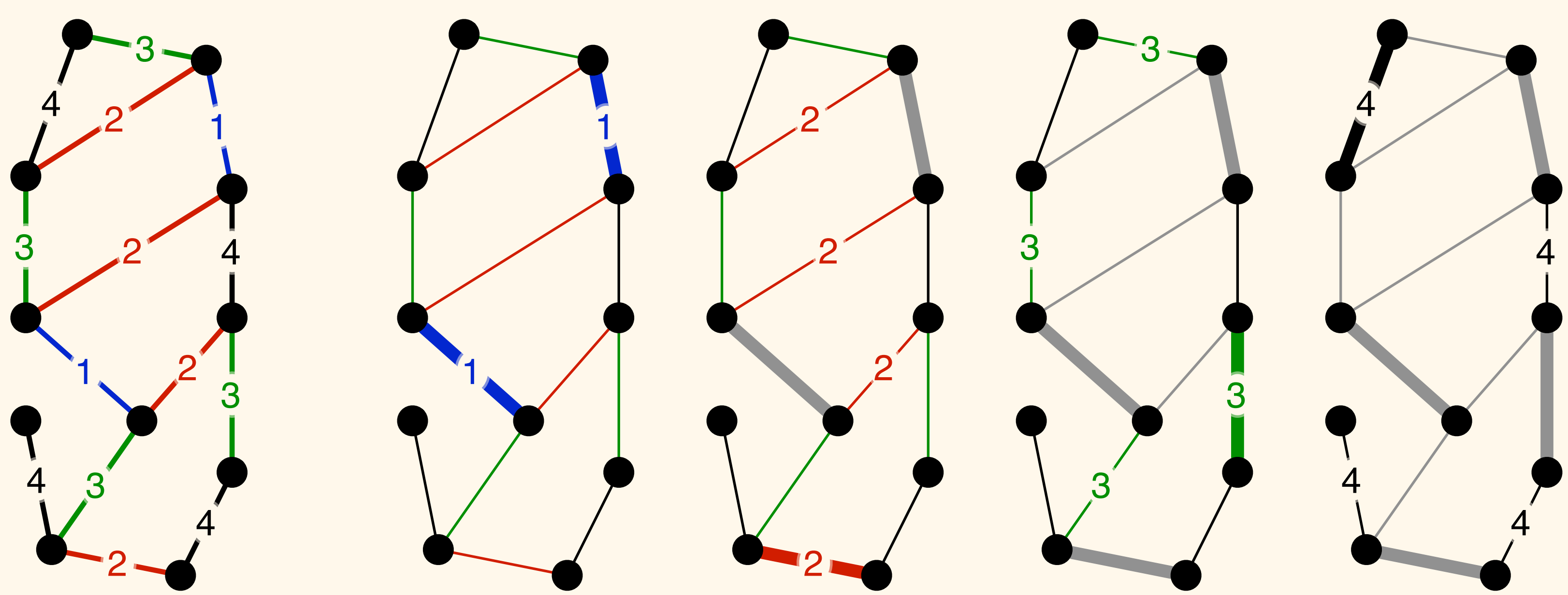
Input　　　Greedy algorithm



Greedy Algorithm

- Greedily add edges of colour 1, 2, …, k

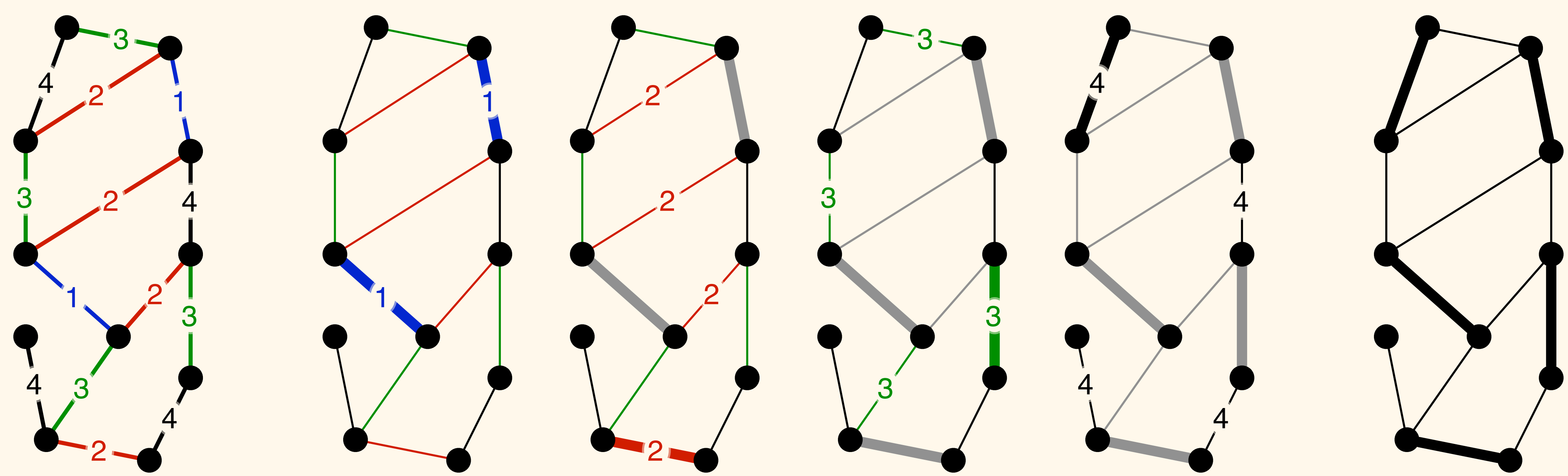
Input   Greedy algorithm



Greedy Algorithm

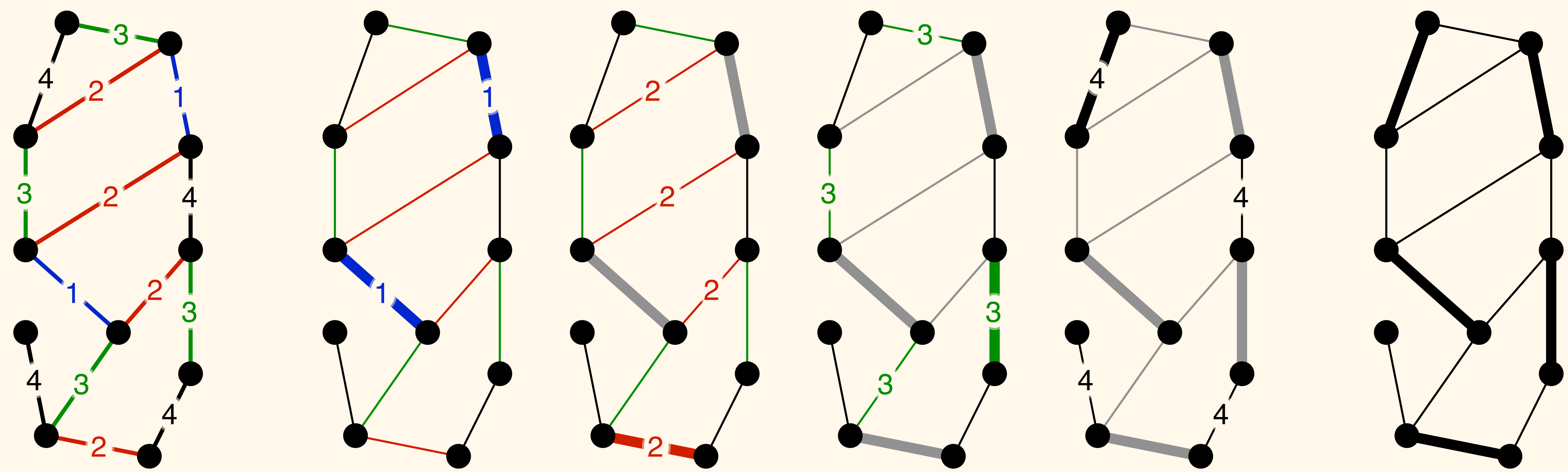
- That’s it – maximal matching in time $O(k)$

Input Greedy algorithm Output



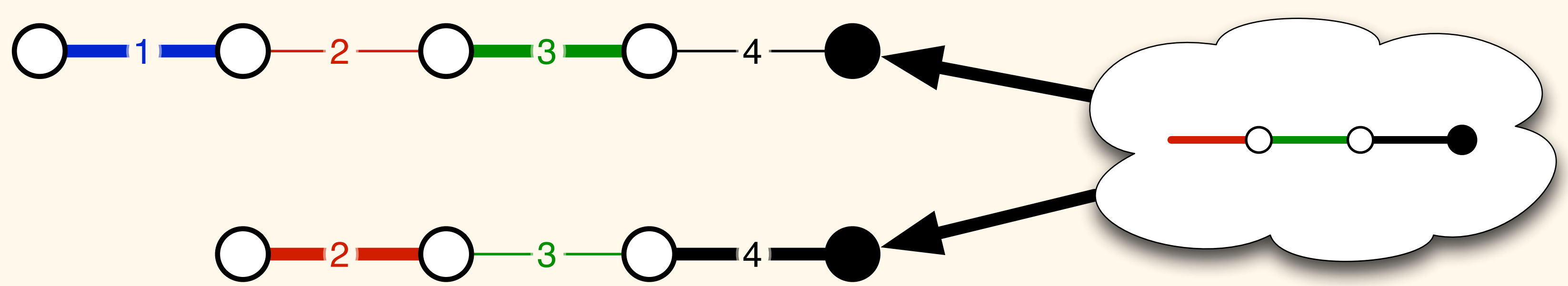
Greedy Algorithm

- Running time is exactly $k - 1$
  - initially each node knows the colours of incident edges



Greedy Algorithm

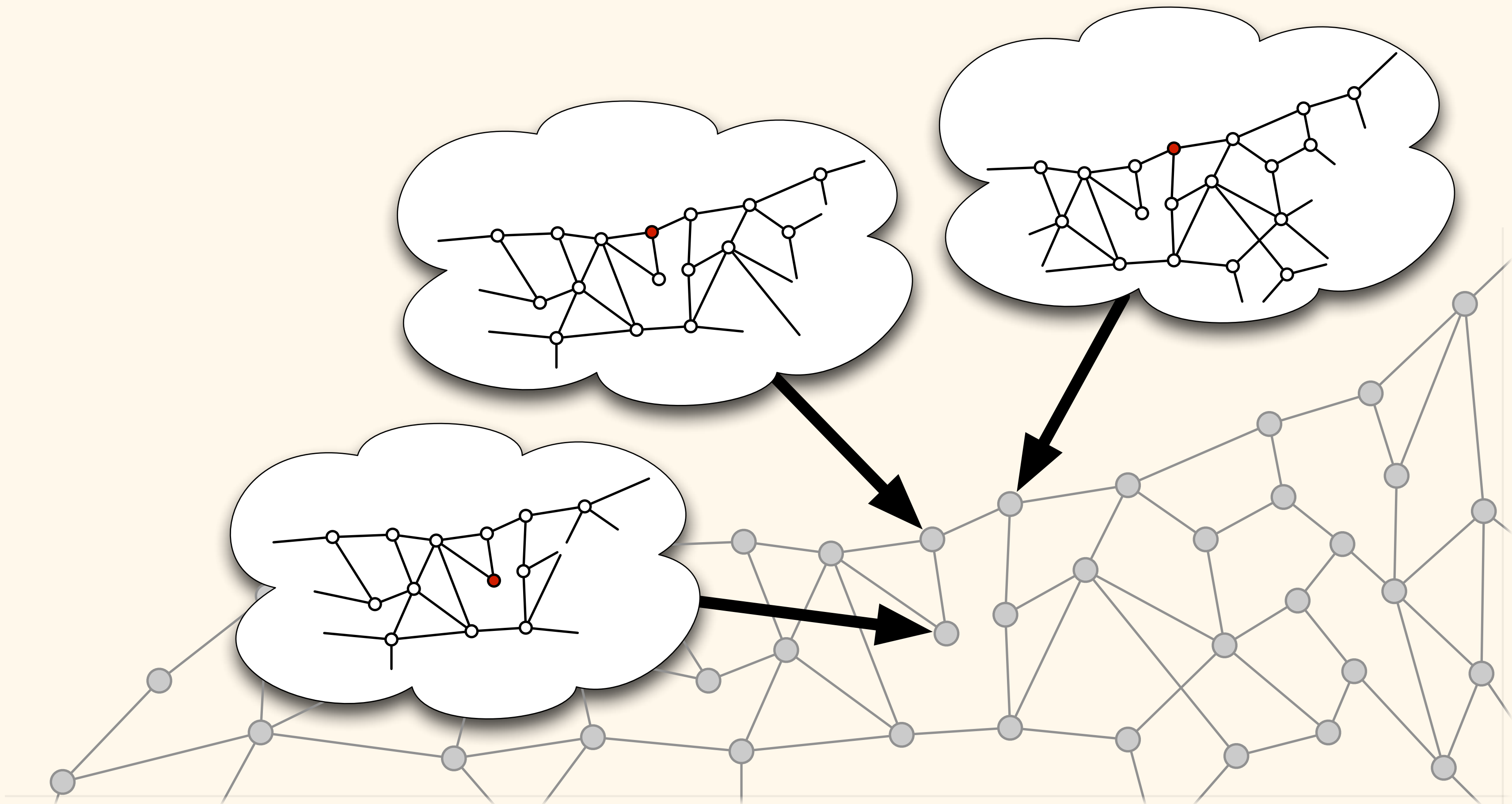
- Running time is exactly $k - 1$
- Analysis is tight
  - Example for case $k = 4$
  - Identical radius-2 neighbourhoods, different outputs:



Greedy Algorithm

- Running time is exactly $k - 1$
- Analysis is tight
- But *could we design a faster algorithm*?
  - turns out that this is connected to some fundamental open questions of the field…

Related Work



n and Δ

- Running time as a function of what?
- Two parameters commonly used:
  - n = number of nodes
  - Δ = maximum degree
- We often assume that n and Δ are known
  - or some upper bounds of them

Problem	Upper bound	Lower bound
maximal matching	$\Delta + \log^* n$	$\text{polylog}(\Delta) + \log^* n$
(Δ+1)-vertex colouring	$\Delta + \log^* n$	$\log^* n$
(2Δ-1)-edge colouring	$\Delta + \log^* n$	$\log^* n$
maximal edge packing	Δ	$\text{polylog}(\Delta)$
vertex cover 2-approx.	Δ	$\text{polylog}(\Delta)$

Positive: *Panconesi–Rizzi* (2001), *Barenboim–Elkin* (2009), *Kuhn* (2009), *Åstrand–Suomela* (2010), ...

Negative: *Linial* (1992), *Kuhn et al.* (2004, 2006)

n and Δ

- Time complexity is well-understood as a function of n
  - asymptotically tight upper and lower bounds
- But we do not really understand time complexity as a function of Δ
  - exponential gap

k and Δ

- What about edge-coloured graphs?
  - k = number of colours, Δ = maximum degree
- Maximal matching:
  - upper bound: $O(\Delta + \log^* k)$
  - lower bound: $\Omega(\text{polylog}(\Delta) + \log^* k)$
- Again, an exponential gap for Δ...

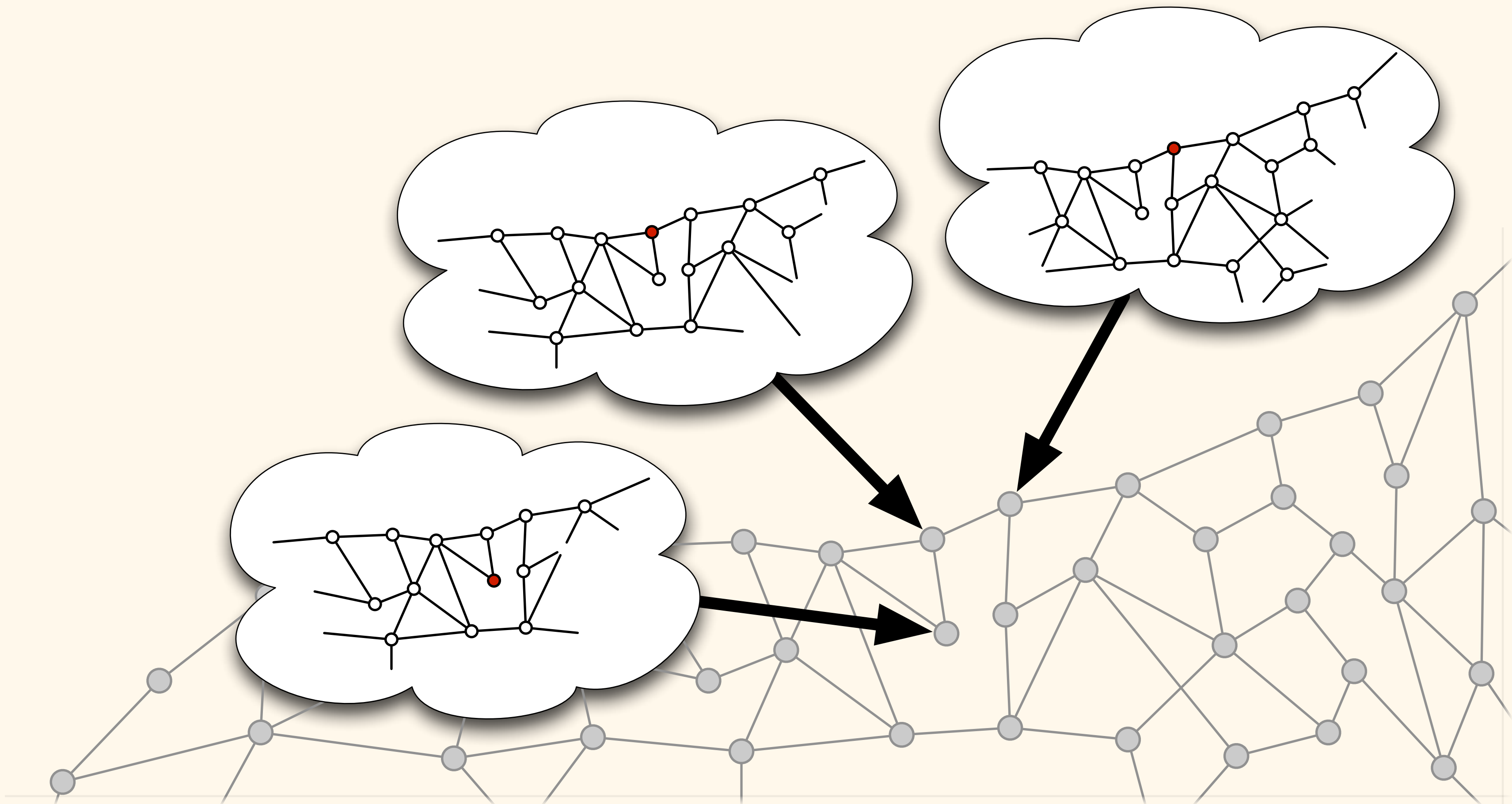
k and Δ

- What about edge-coloured graphs?
  - k = number of colours, Δ = maximum degree
- Maximal matching:
  - upper bound: $O(\Delta + \log^* k)$
  - lower bound: $\cancel{\Omega(\text{polylog}(\Delta) + \log^* k)}$ $\boxed{\Omega(\Delta + \log^* k)}$
- ~~Again, an exponential gap for Δ…~~

Contributions

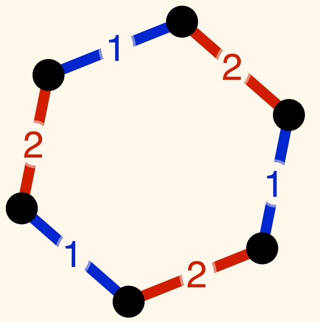
- Time complexity of finding maximal matchings in k-edge-coloured graphs
- General graphs: $\geq k - 1$
  - matching upper bound: $\leq k - 1$ (greedy)
- Bounded-degree graphs: $\Omega(\Delta + \log^* k)$
  - matching upper bound: $O(\Delta + \log^* \mathrm{k})$
    (an adaptation of *Panconesi–Rizzi* 2001)

Lower Bound

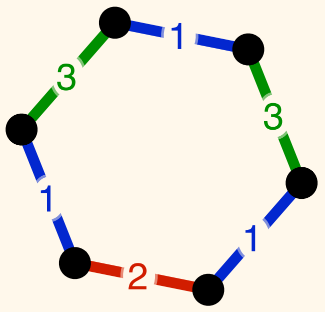


Plan

- Focus: d-regular k-edge-coloured graphs
- If $d = k$:
  - trivial to find a maximal matching in constant time (pick a colour class)



- If $d = k - 1$:
  - as difficult as the general case!
  - we show that we need at least d rounds



Plan

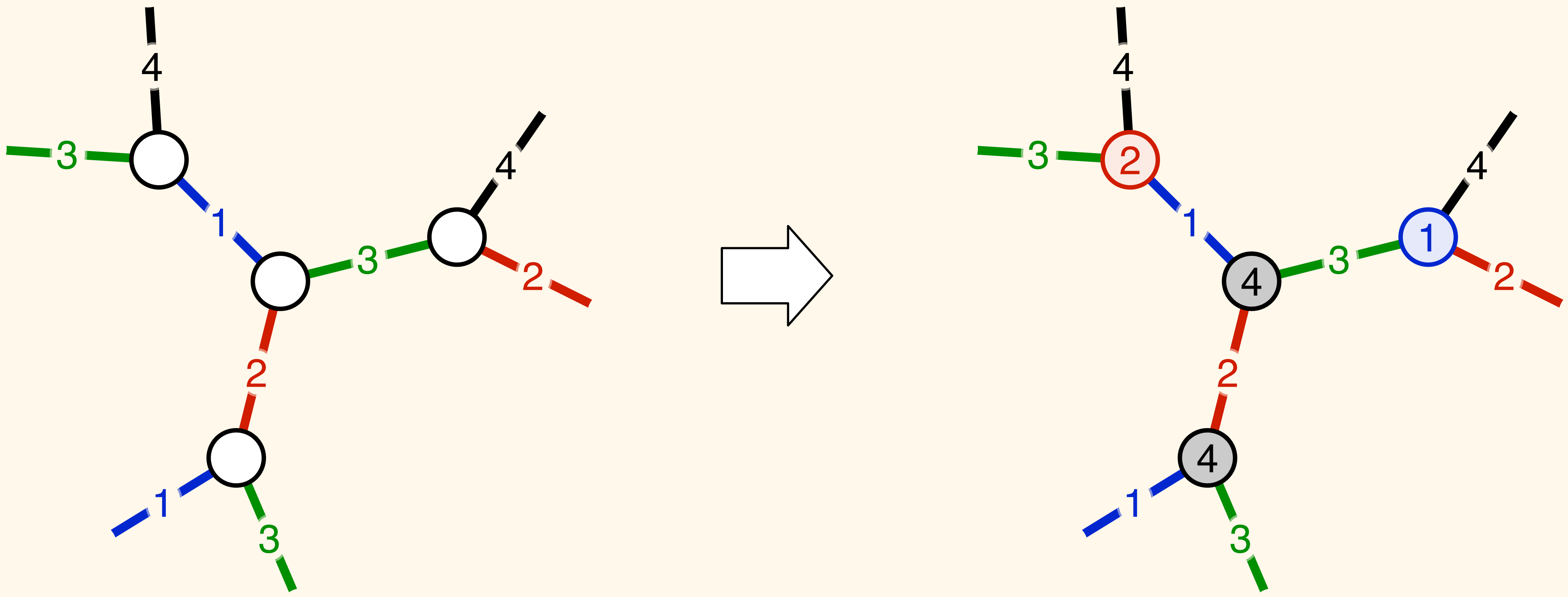
- Given $k \geq 3$, define $d = k - 1$, assume:
  - algorithm A finds a maximal matching in any d-regular k-edge-coloured graph
- We construct a pair of infinite trees T_1, T_2:
  - root nodes have identical $(k - 2)$-neighbourhoods
  - output of A: root of T_1 matched, root of T_2 unmatched
  - running time of A is at least $k - 1$

Tools

- Now we need tools for constructing and manipulating infinite edge-coloured trees…
- *Warning*:
  - the manuscript uses a very different formalism (with some group-theoretic constructions)
  - in this talk I'll try to keep everything lightweight, and just present the key ideas with illustrations

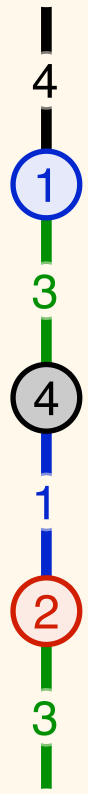
Node Colours

- Node colour = the unique “missing colour”



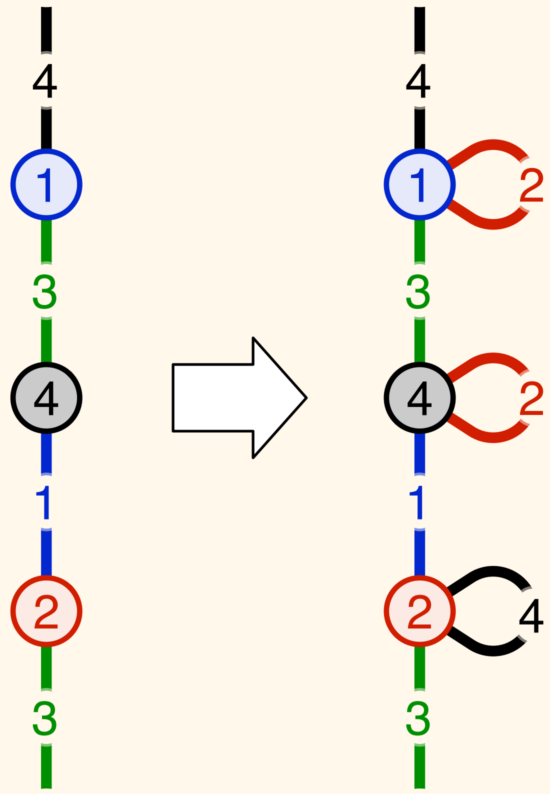
Templates

- Degree < d



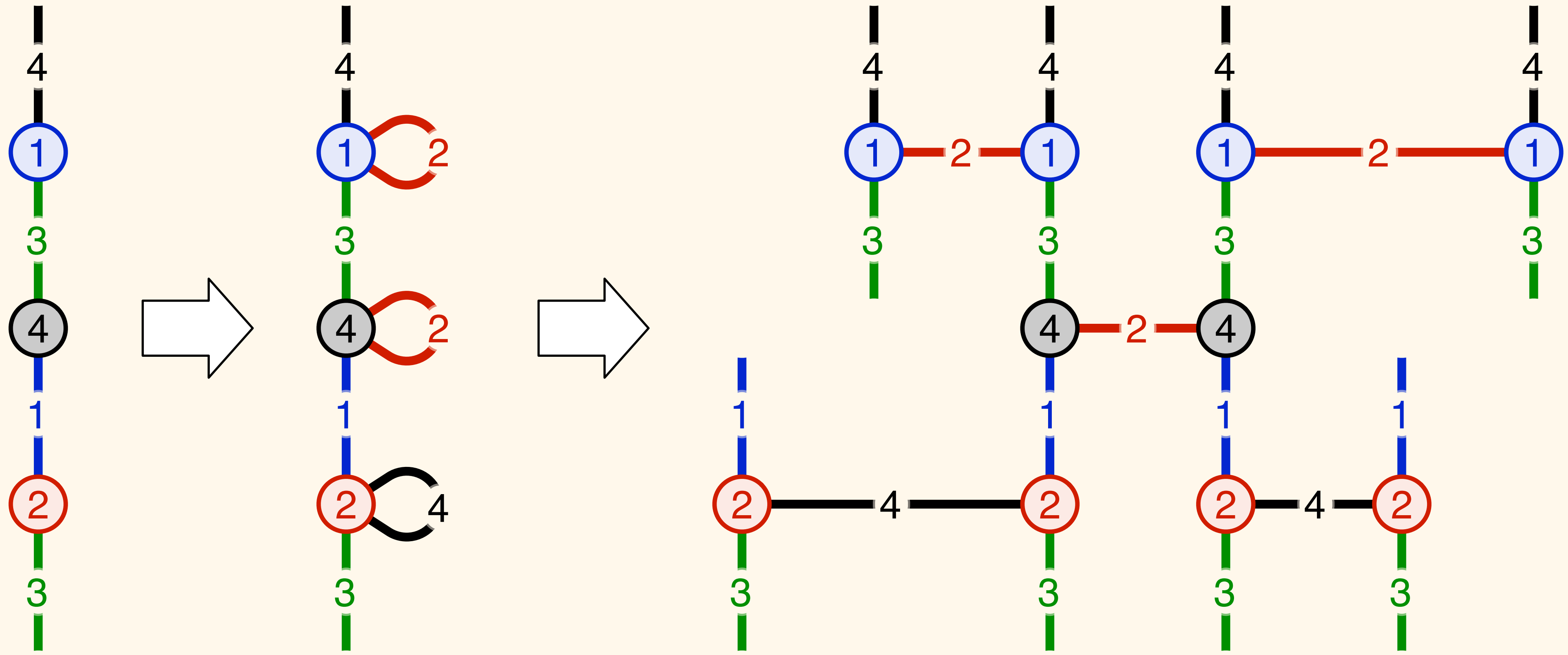
Templates

- Degree < d: add loops



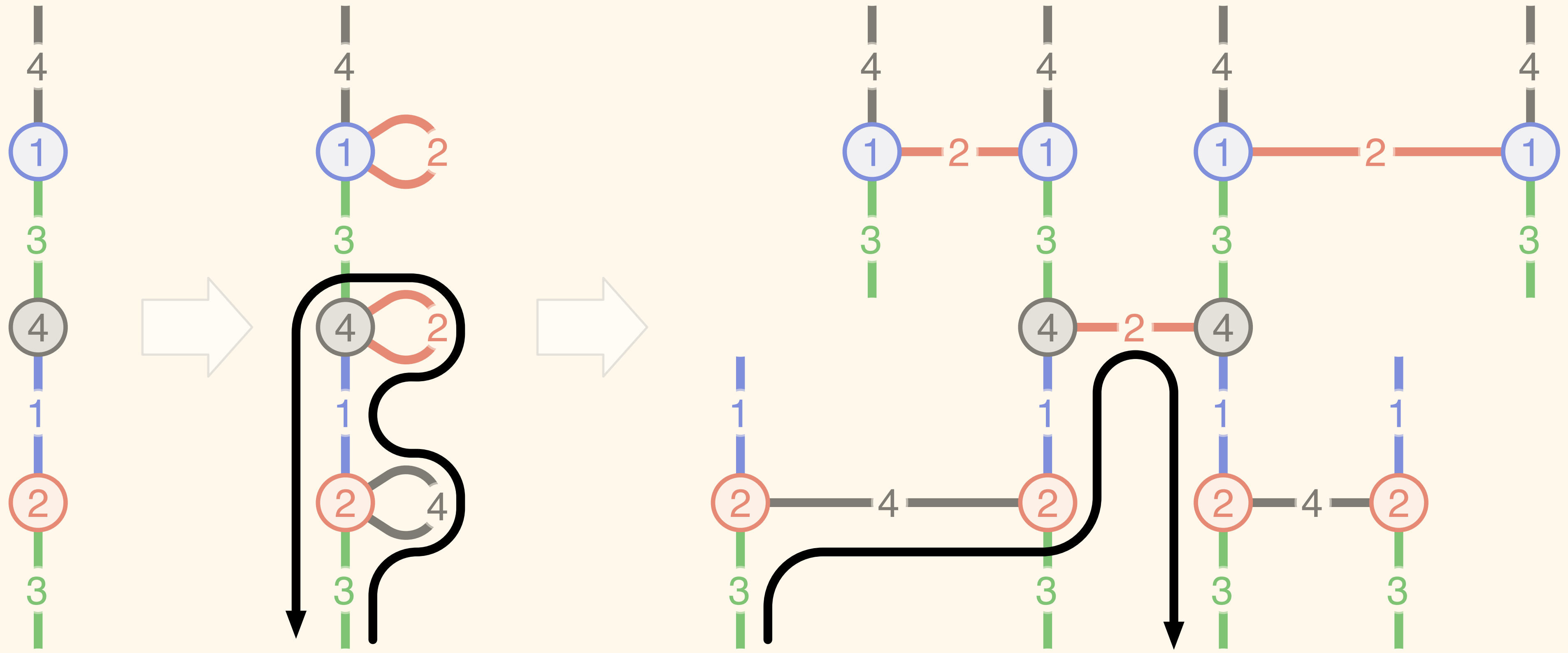
Templates

- Degree < d: add loops, unfold loops



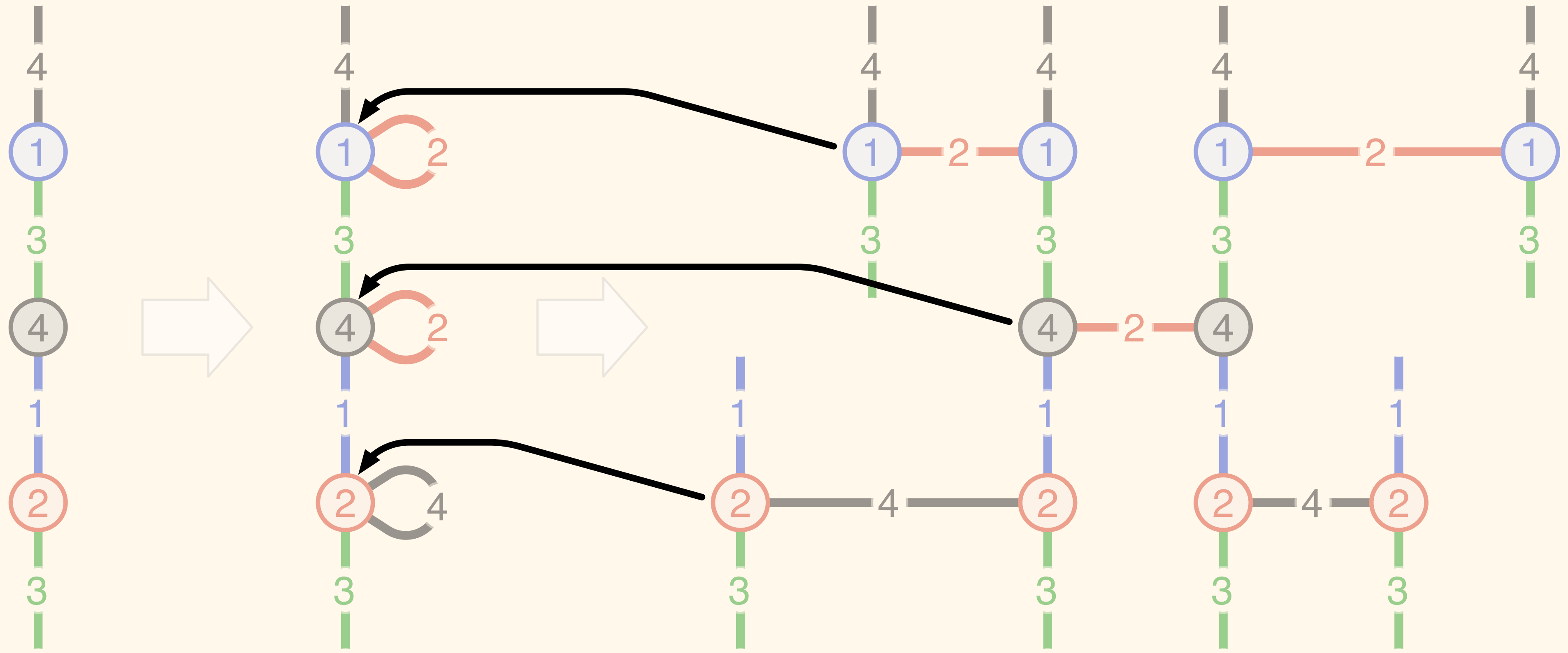
Templates

- Unfolding preserves traversals



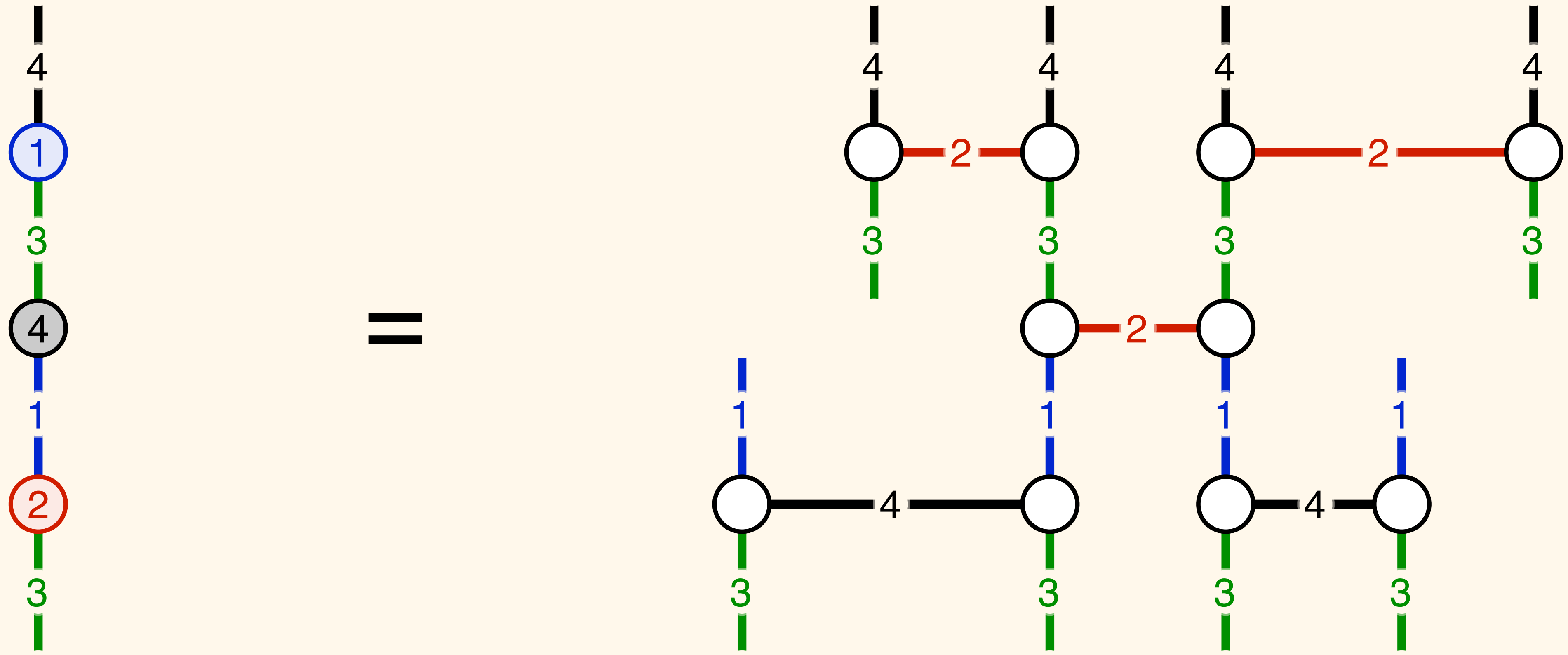
Templates

- Natural homomorphism

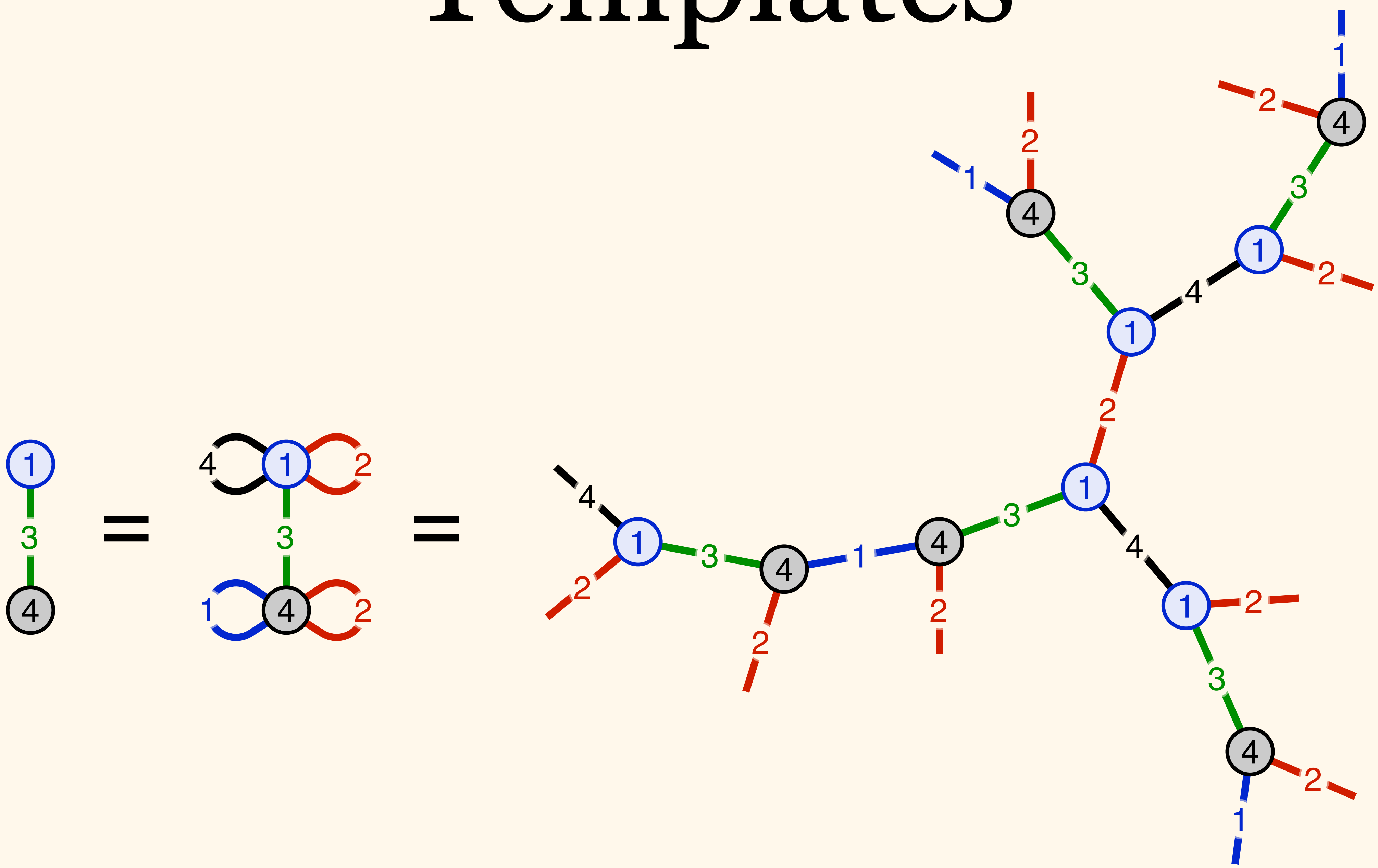


Templates

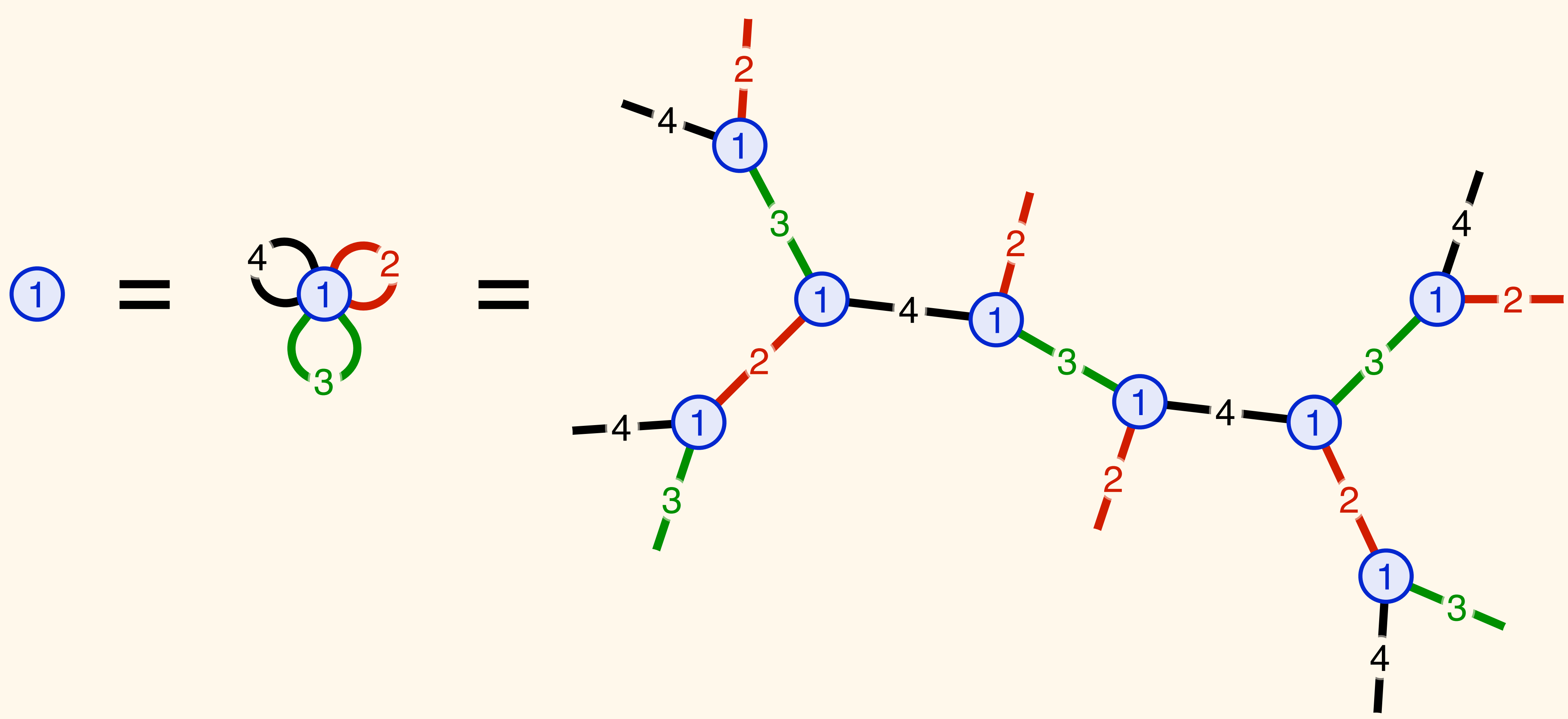
- Compact representations of trees



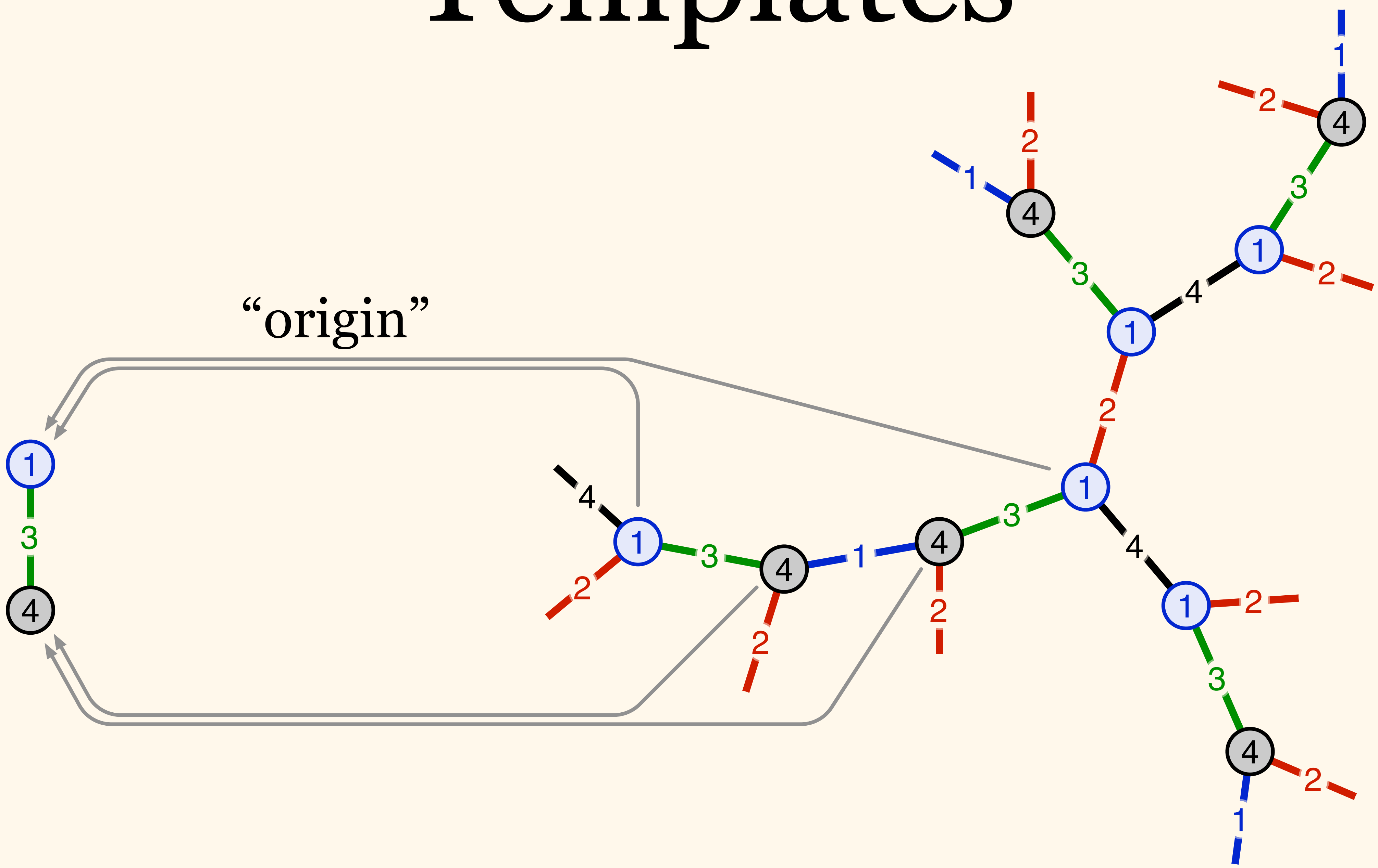
Templates



Templates

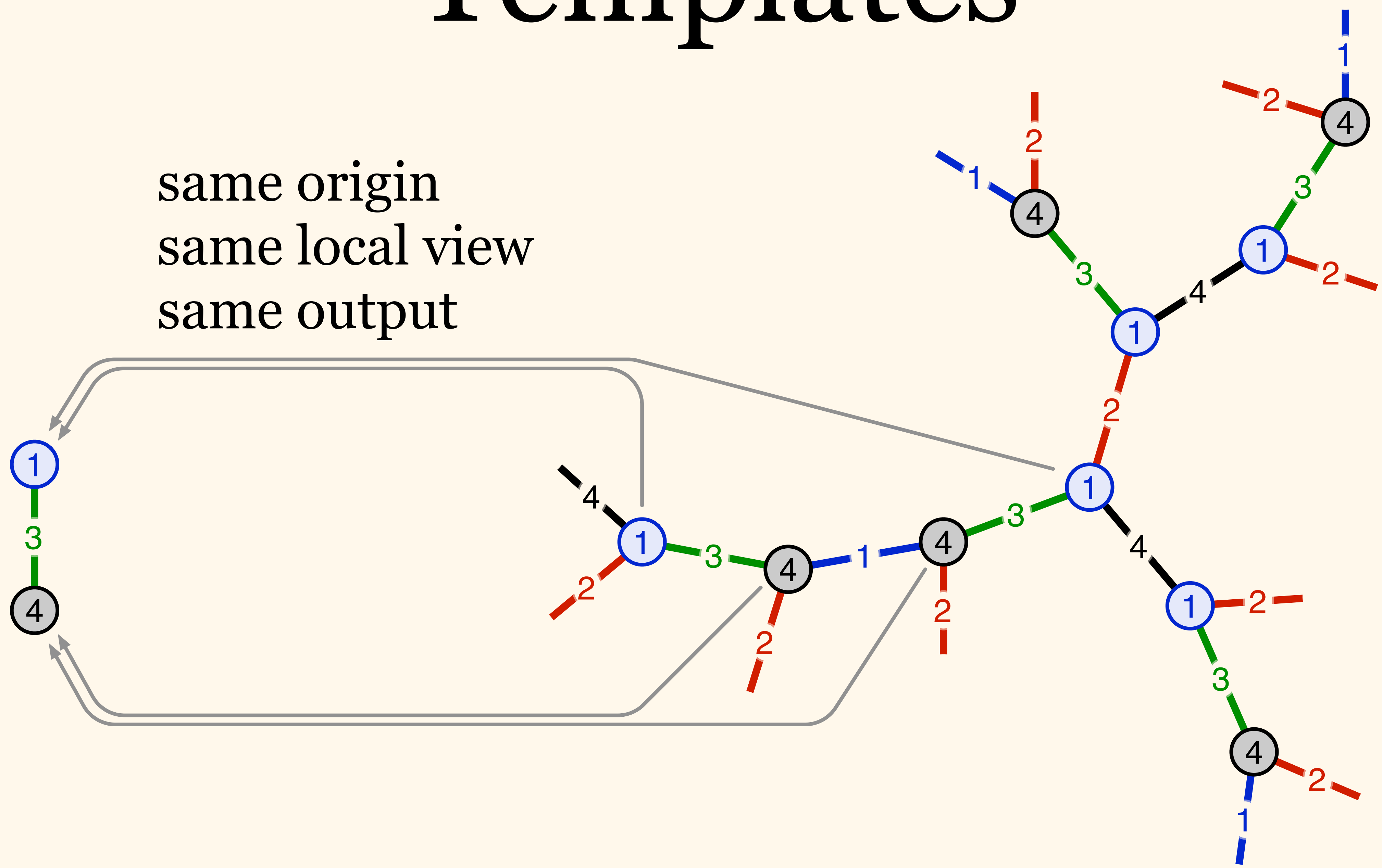


Templates

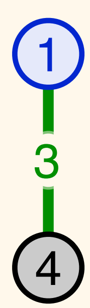


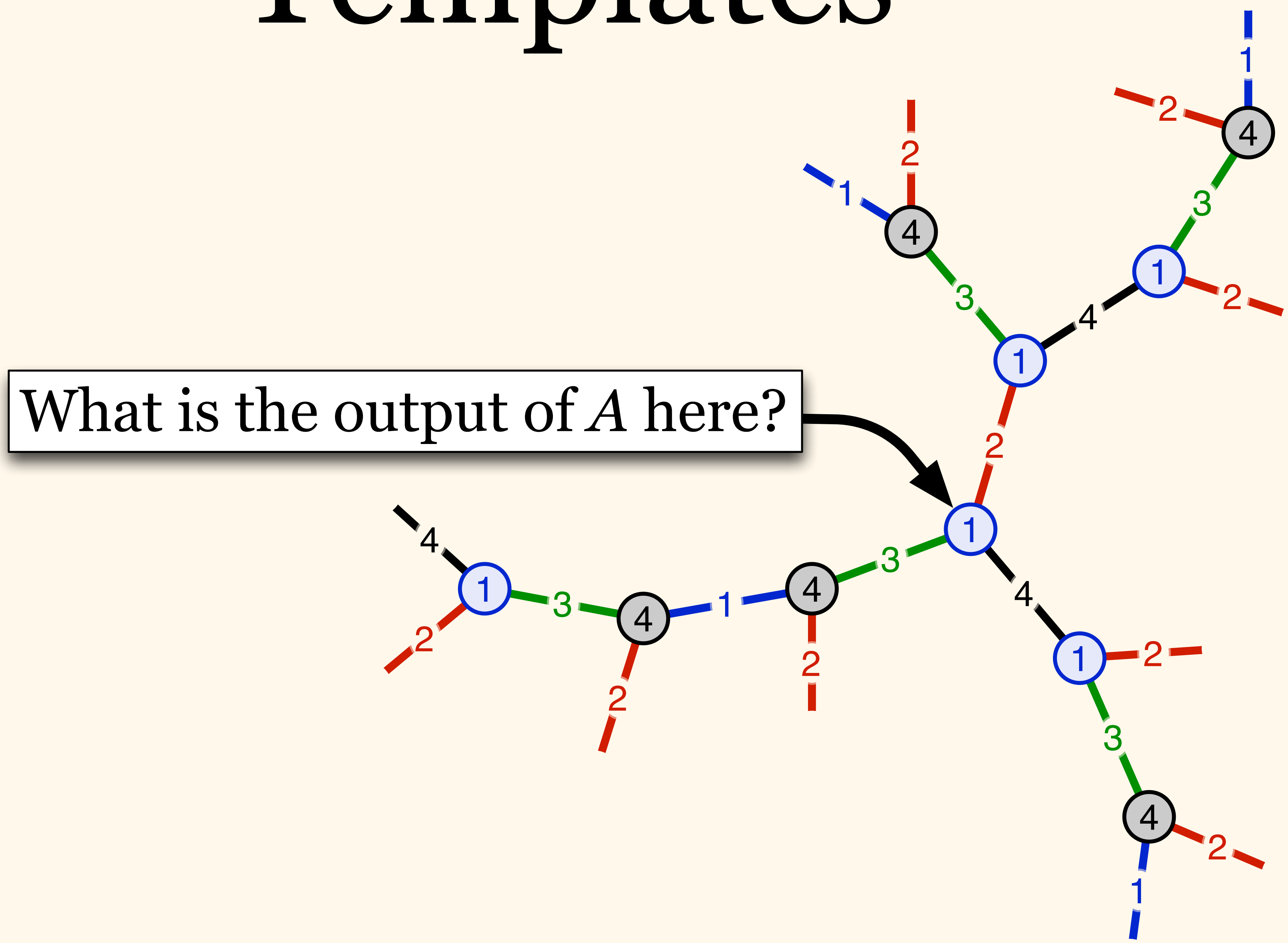
Templates

same origin
same local view
same output

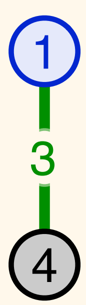


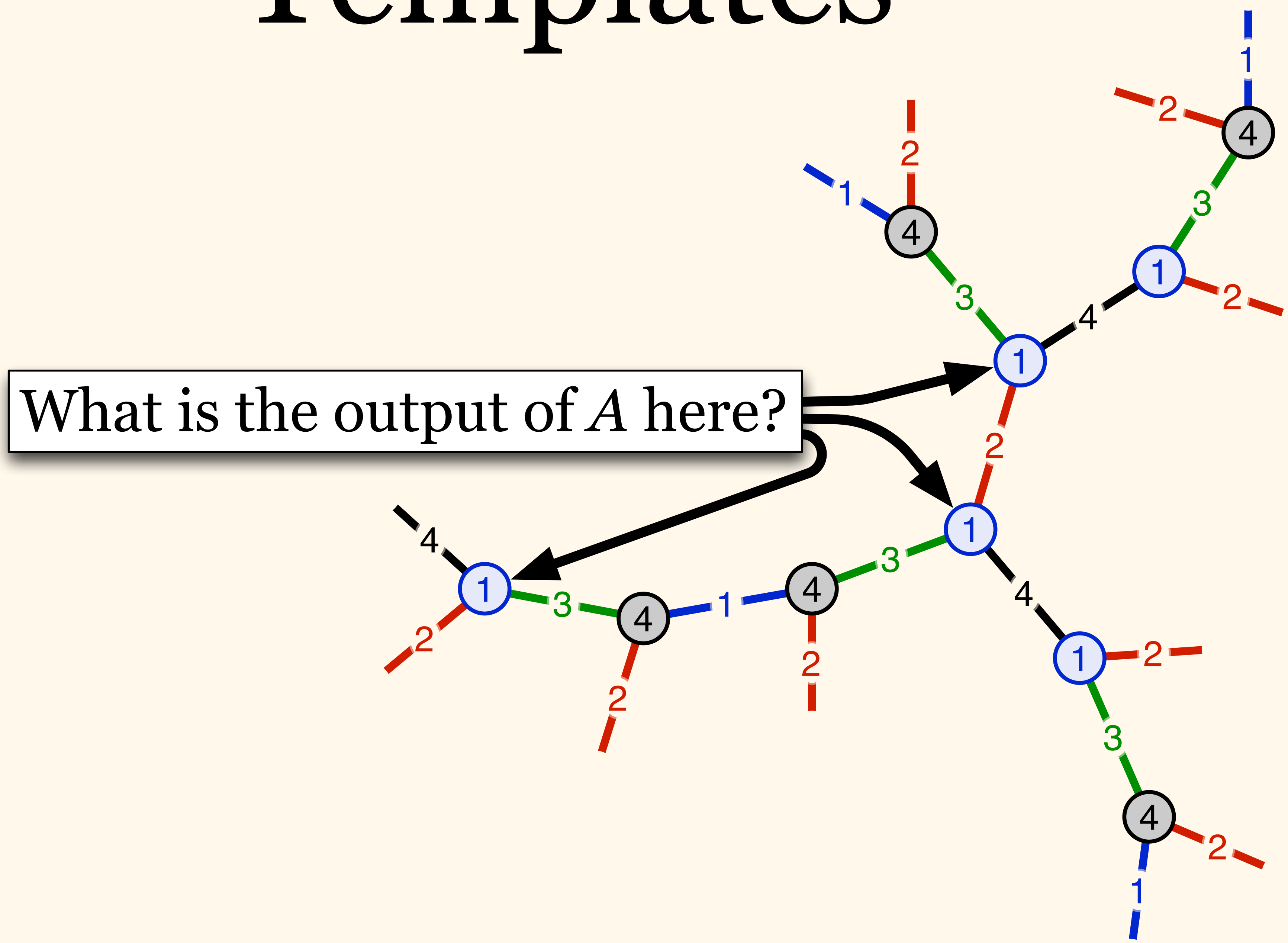
Templates





Templates



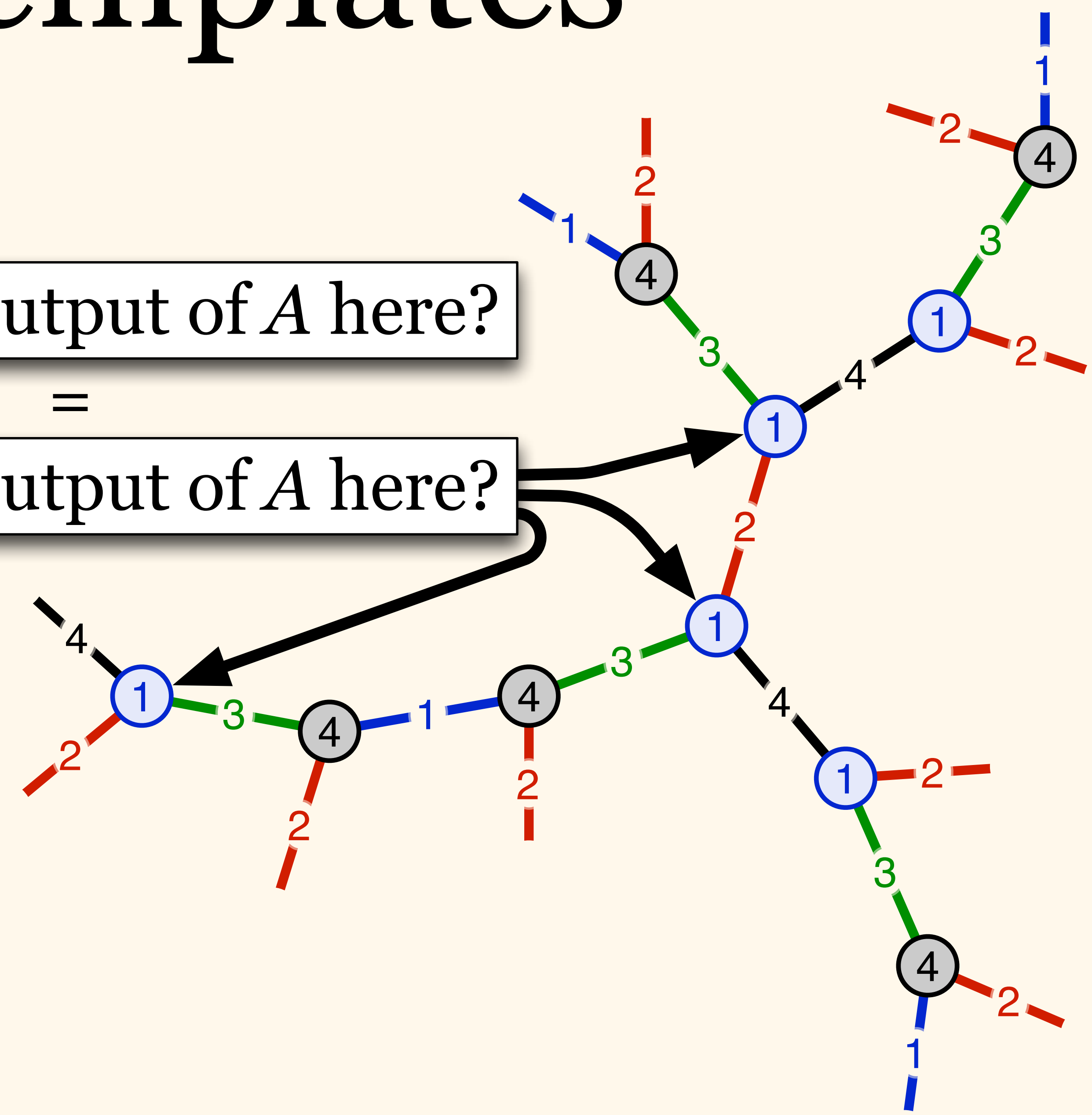


Templates

What is the output of A here?

=

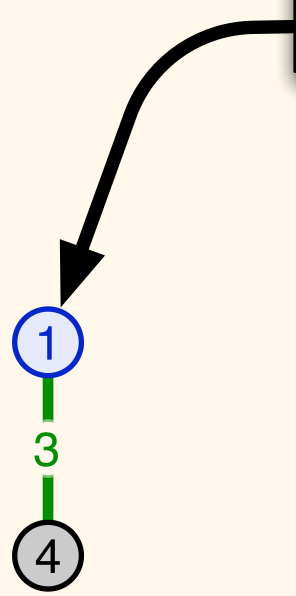
What is the output of A here?



Definition!

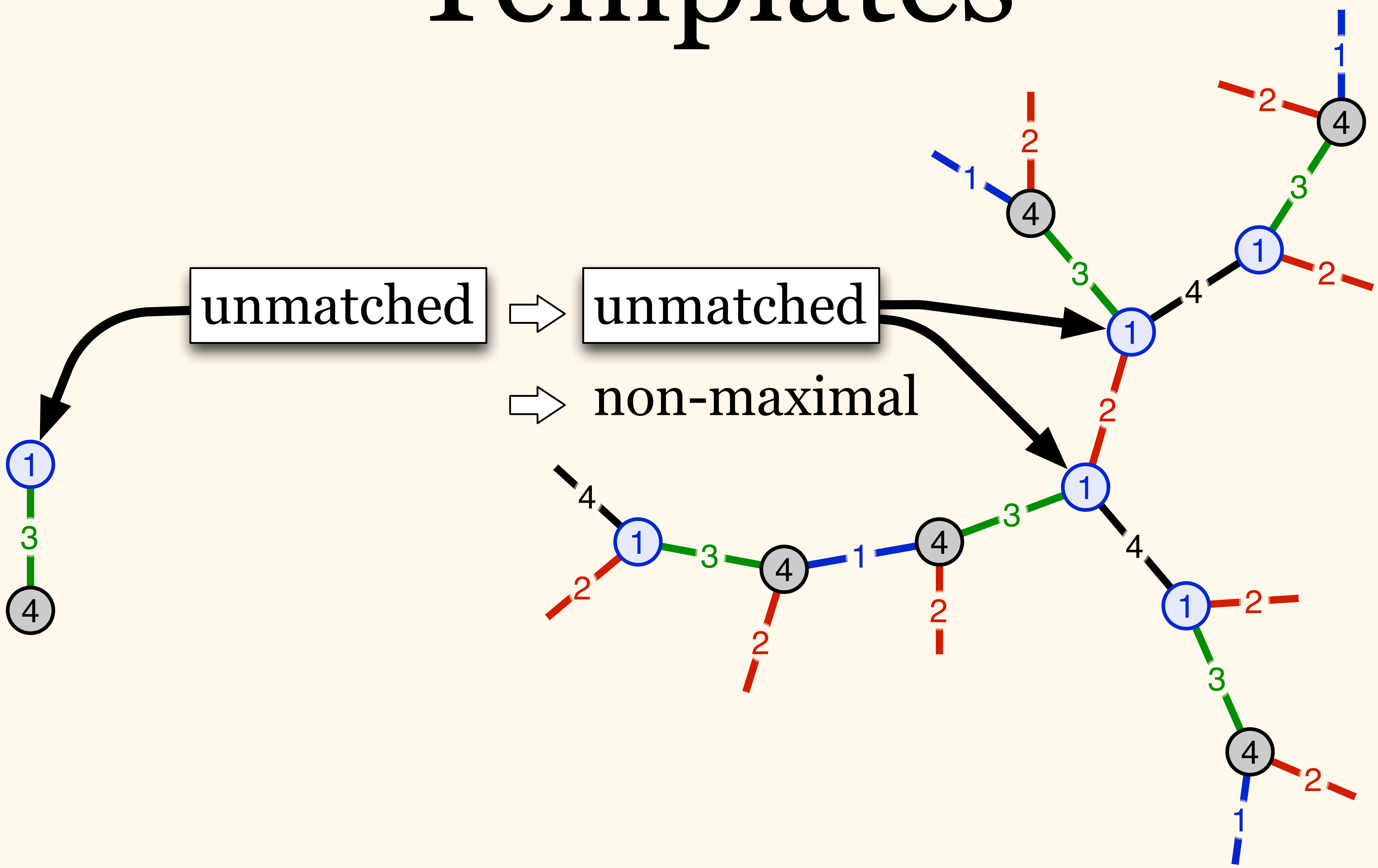
Templates

What is the output of A here?



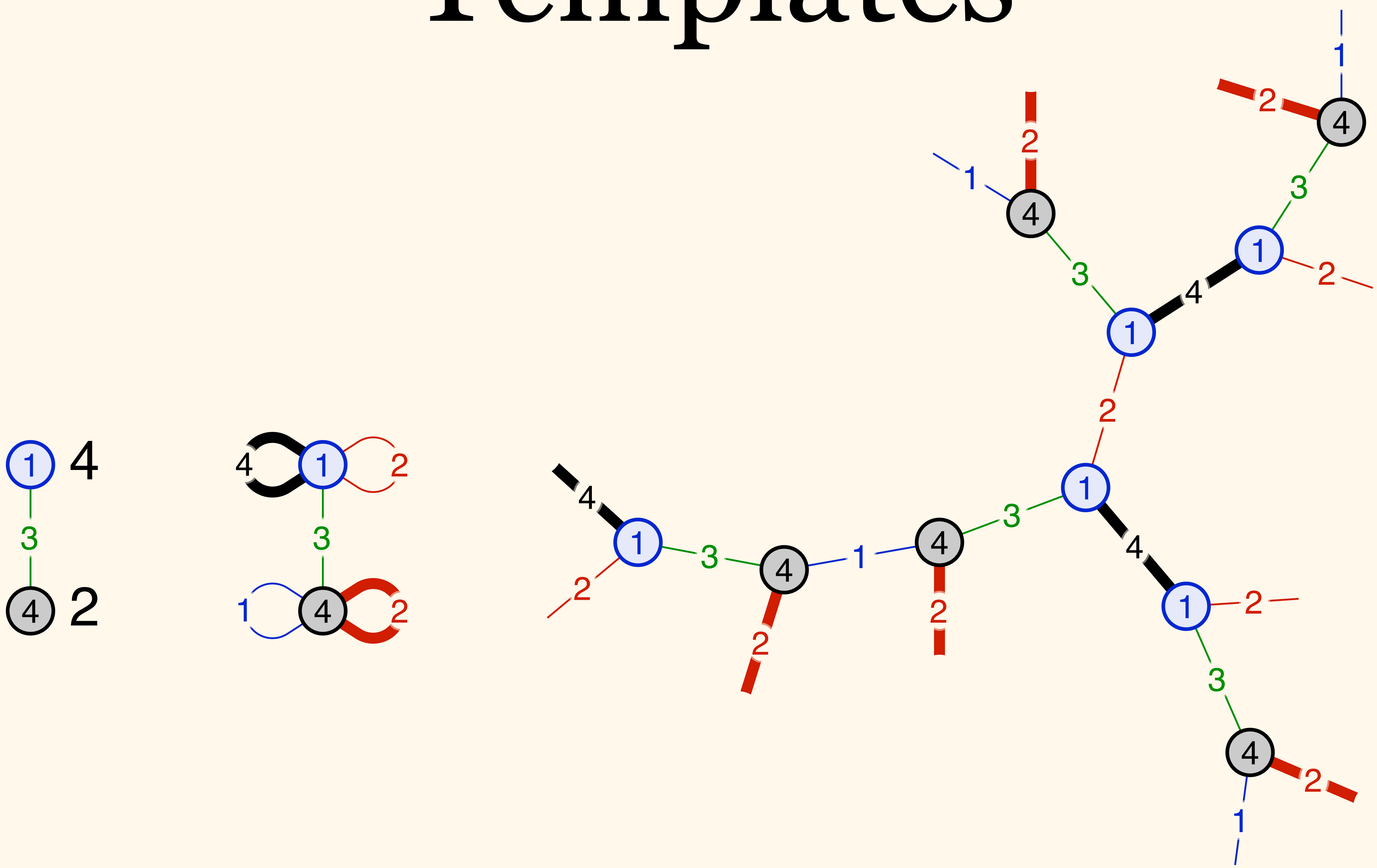
From now on we can study
the output of algorithm A on templates…

Templates



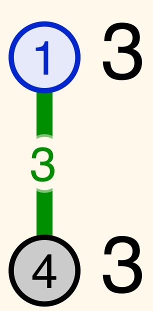
Template of degree < d: all nodes are matched

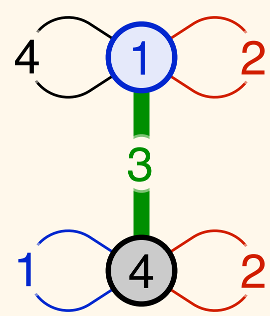
Templates

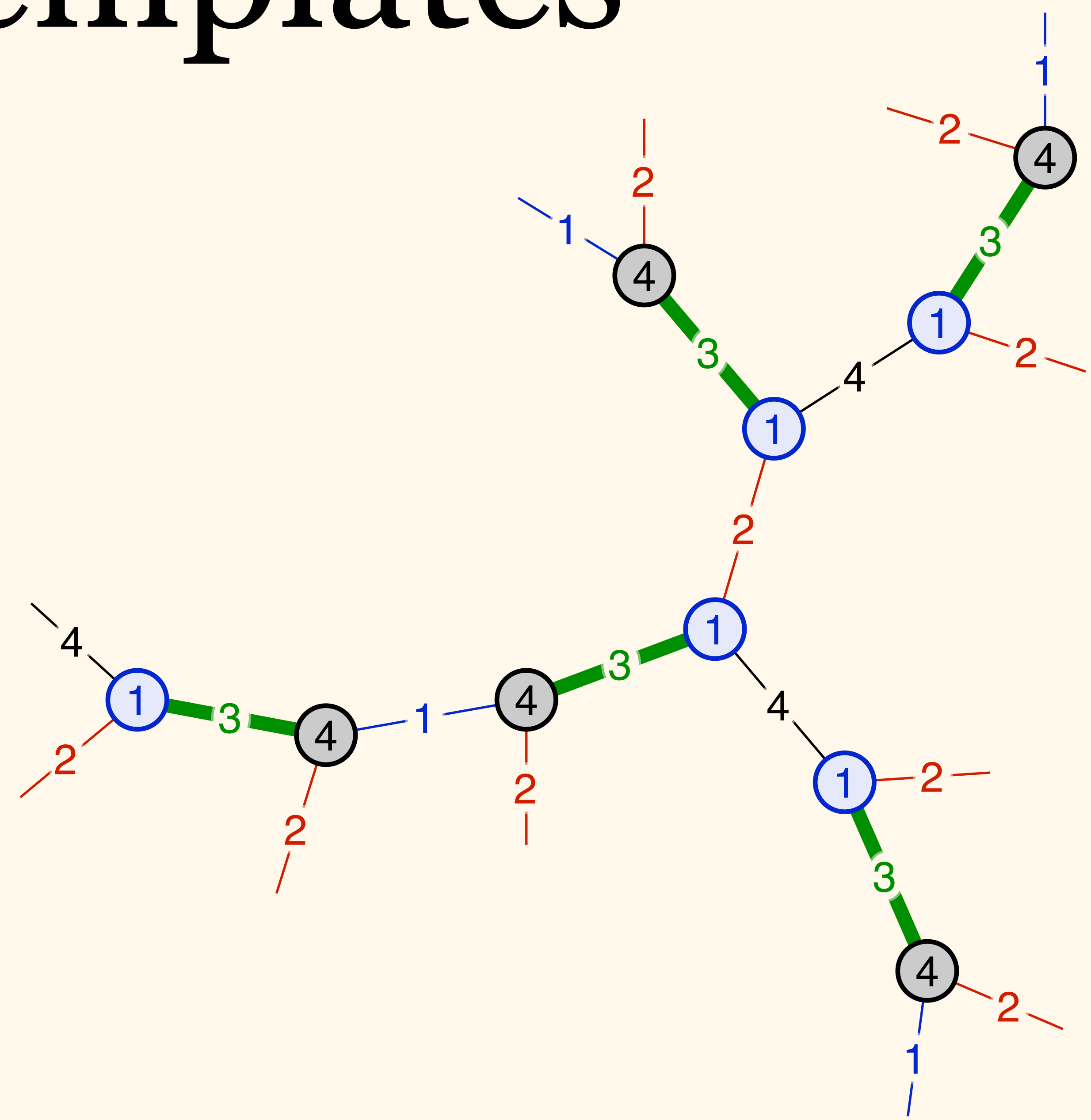


Output x: matched along the edge of colour x

Templates



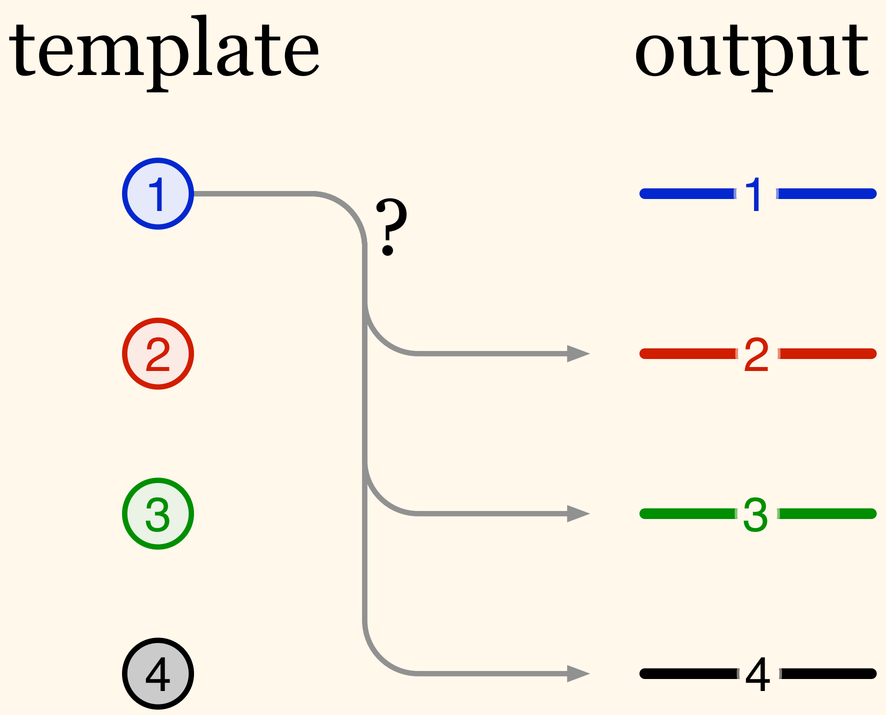




Output x: matched along the edge of colour x

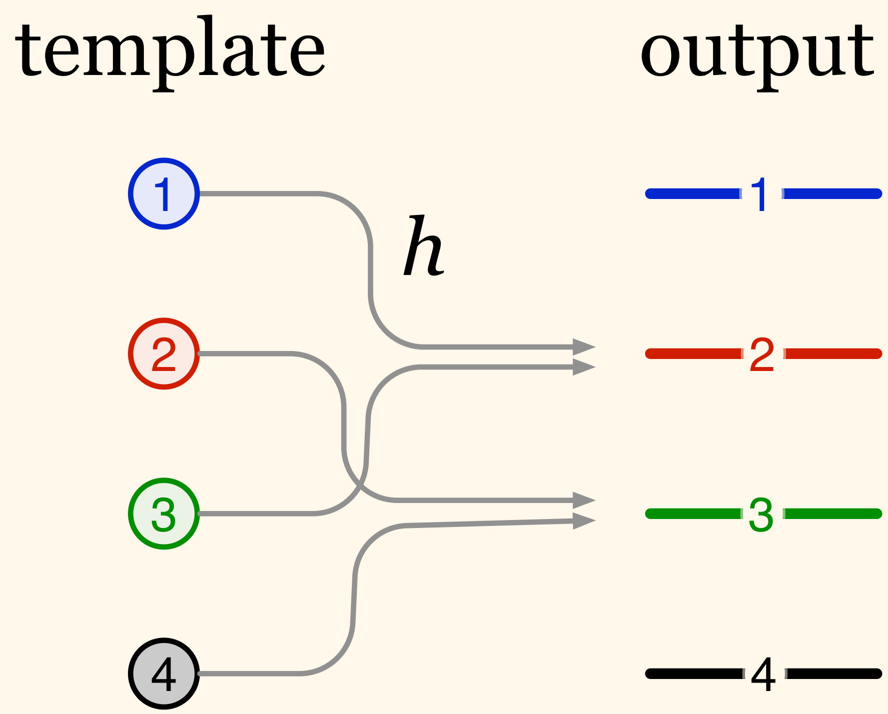
Base Case

- Apply algorithms A to templates of degree zero
- Defines a mapping from node colours to outputs



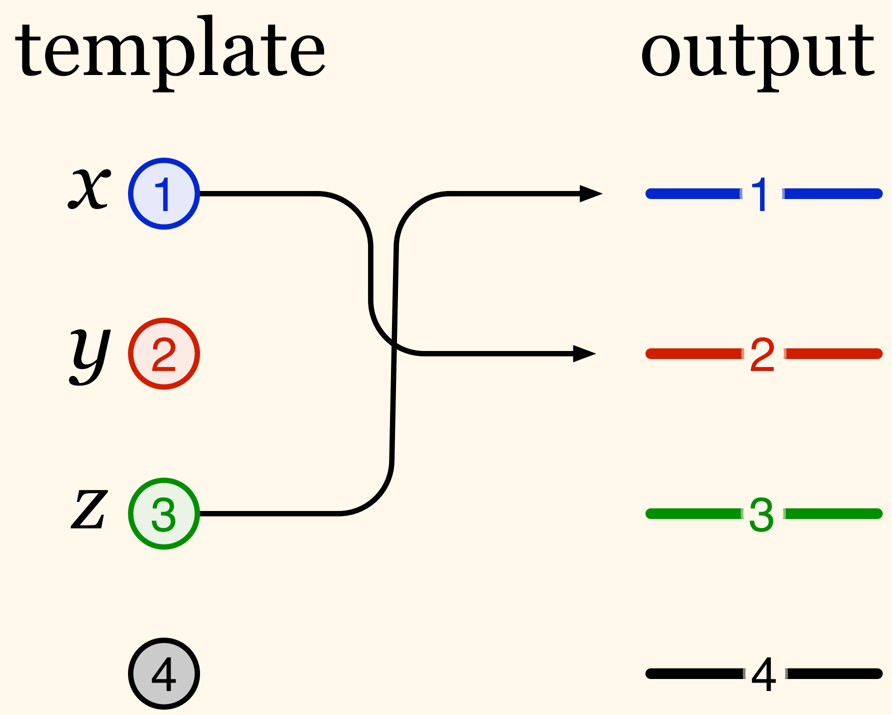
Base Case

- h: $\{1,2,\ldots,k\} \to \{1,2,\ldots,k\}$
- no fixed points



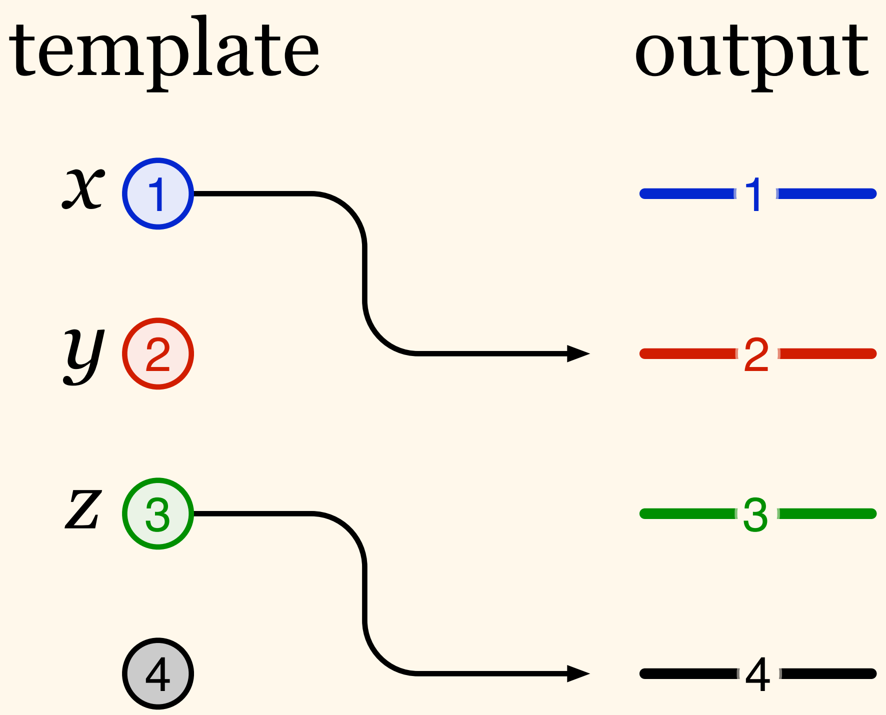
Base Case

- h: $\{1,2,\ldots,k\} \rightarrow \{1,2,\ldots,k\}$
- no fixed points
- there are distinct x, y, z with
  - $h(x) = y$
  - $h(z) \neq y$

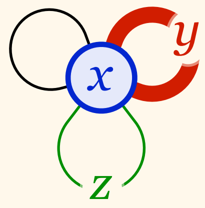


Base Case

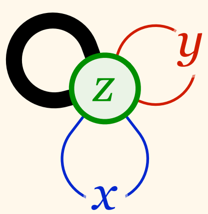
- h: $\{1,2,\ldots,k\} \to \{1,2,\ldots,k\}$
- no fixed points
- there are distinct x, y, z with
  - $h(x) = y$
  - $h(z) \neq y$



Base Case

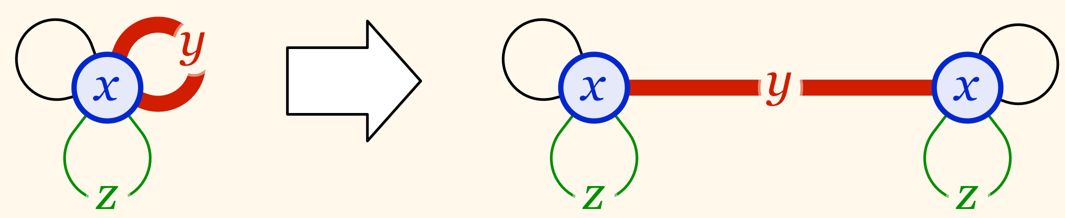


edge of colour y exists,
in matching

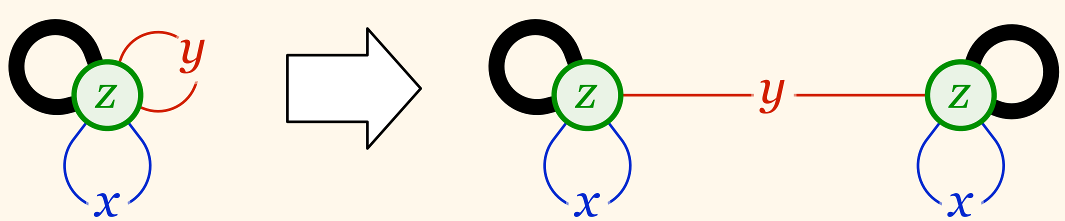


edge of colour y exists,
but not in matching

Base Case

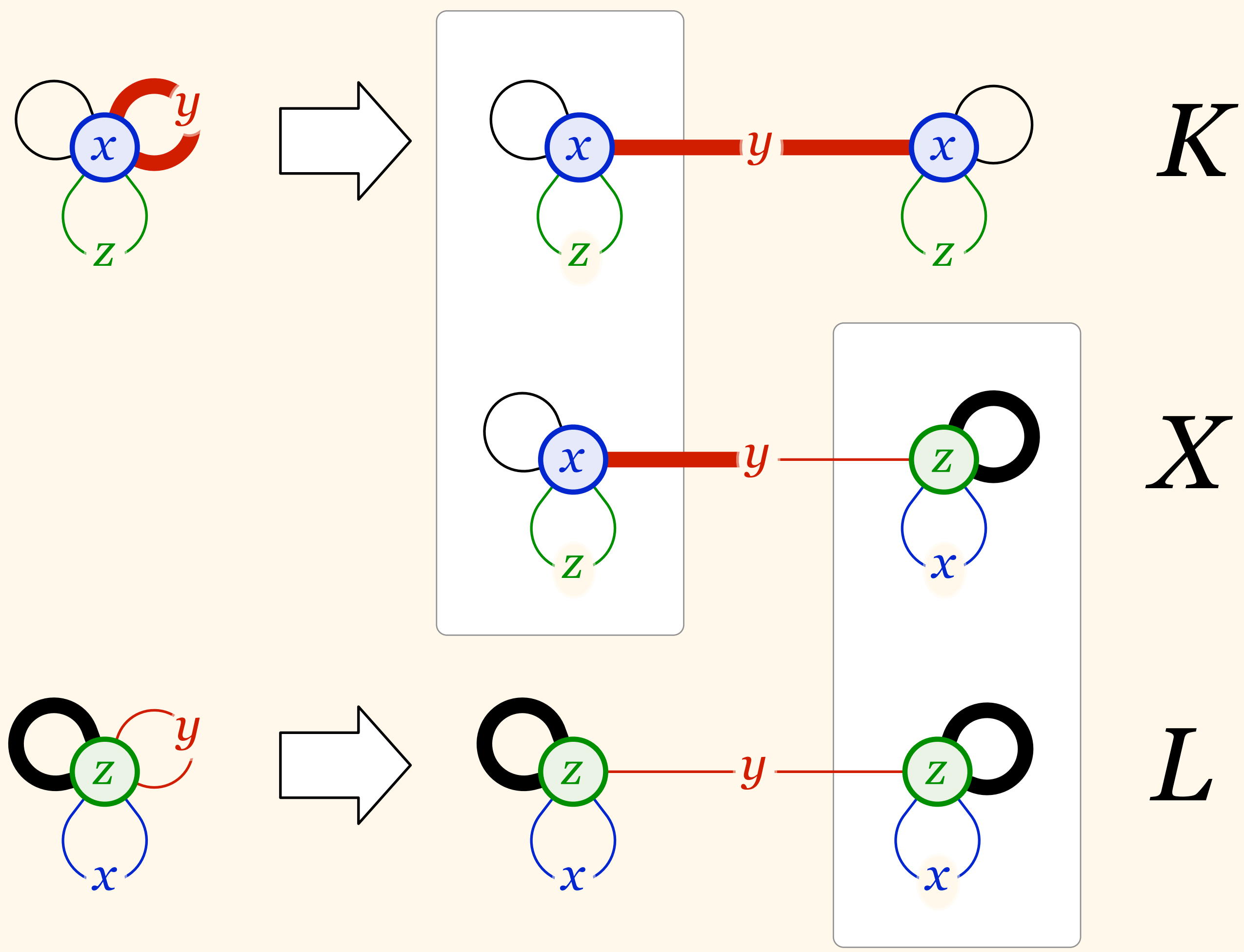


K



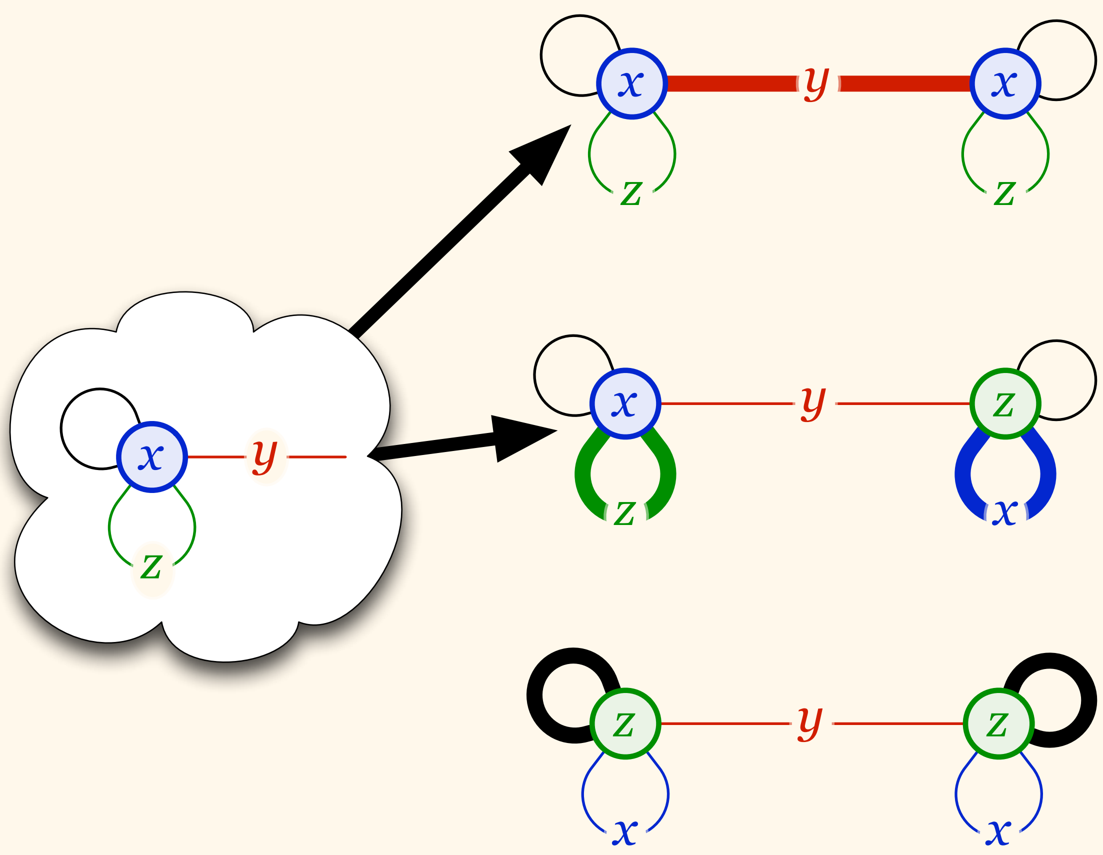
L

Base Case



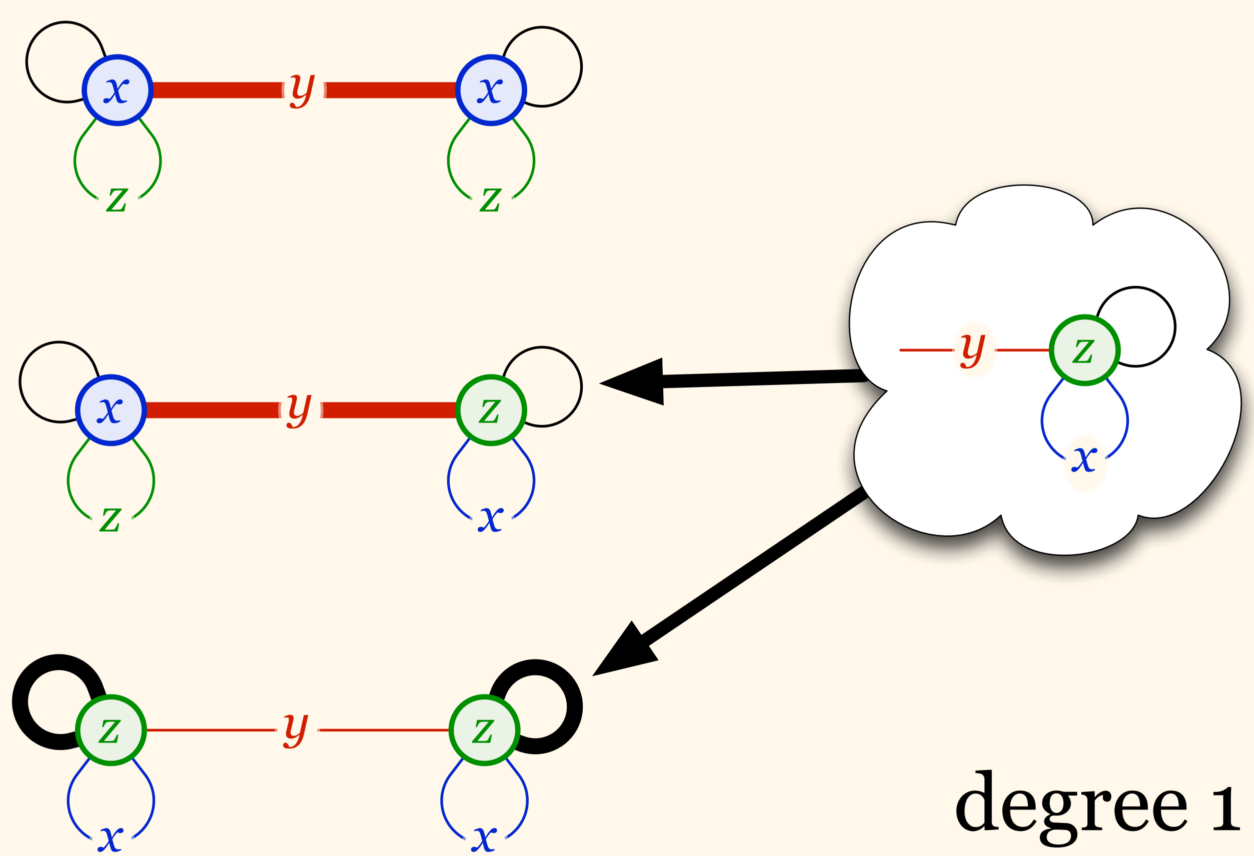
output in X cannot
be copied from K & L
– something must change!

Base Case



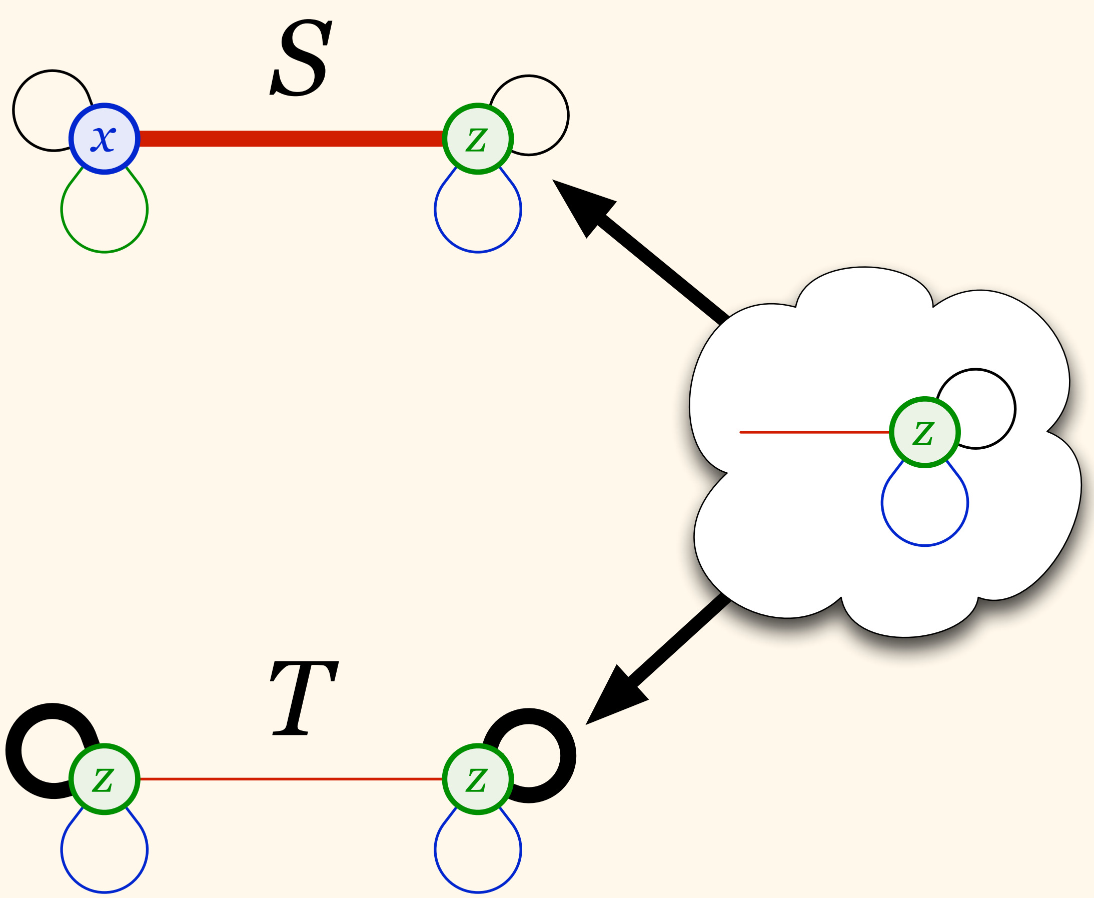
degree 1 templates,
same radius-0 view,
different output

Base Case



degree 1 templates,
same radius-0 view,
different output

Inductive Step

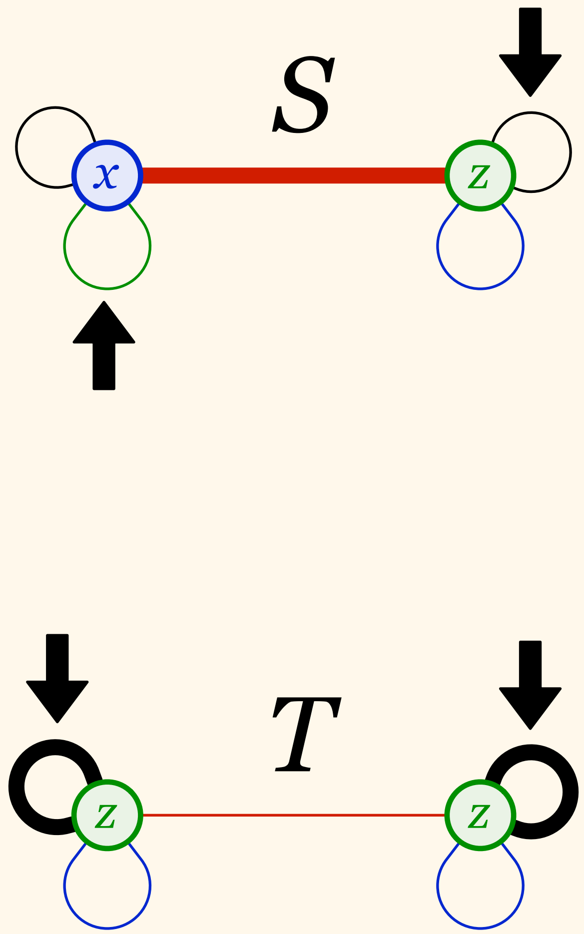


Given:
degree i templates,
same radius-(i–1) view,
different output

Construct:
degree i+1 templates,
same radius-i view,
different output

(here i = 1)

Inductive Step

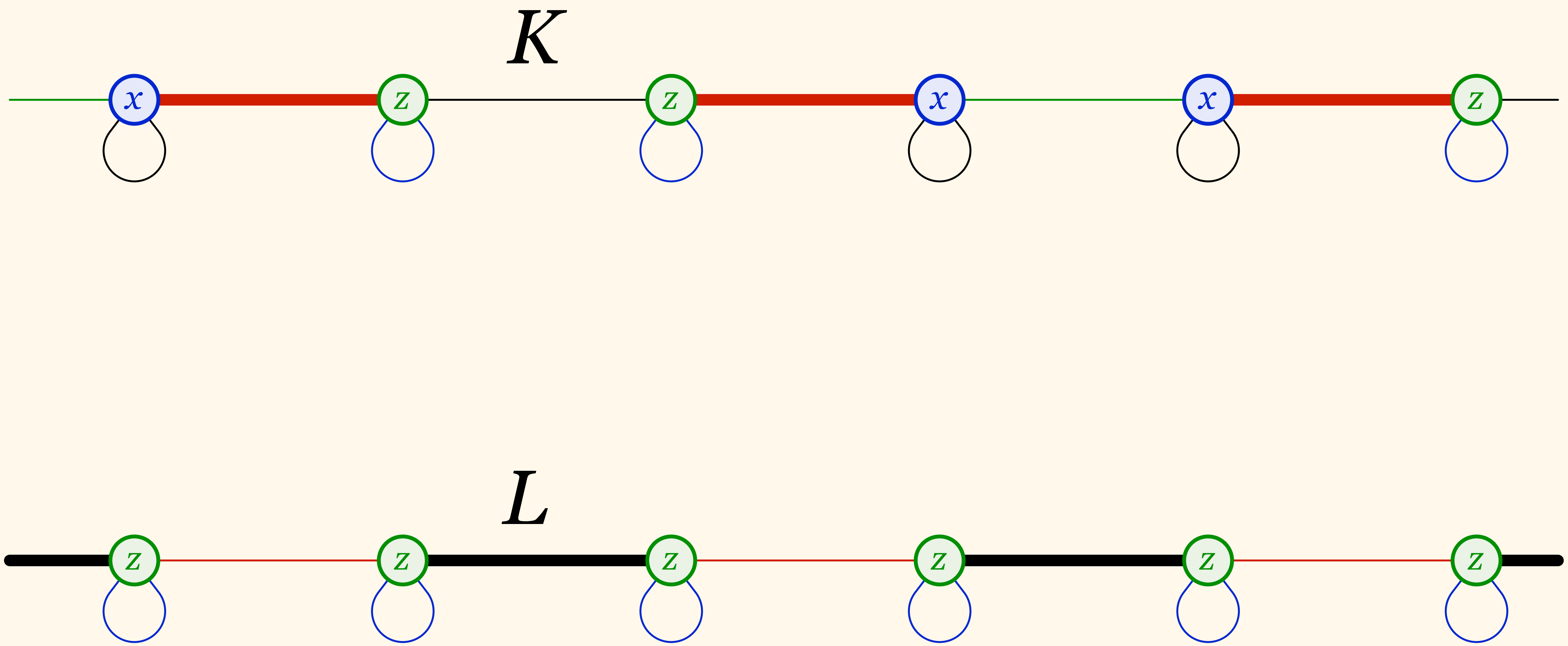


Choose one loop per node

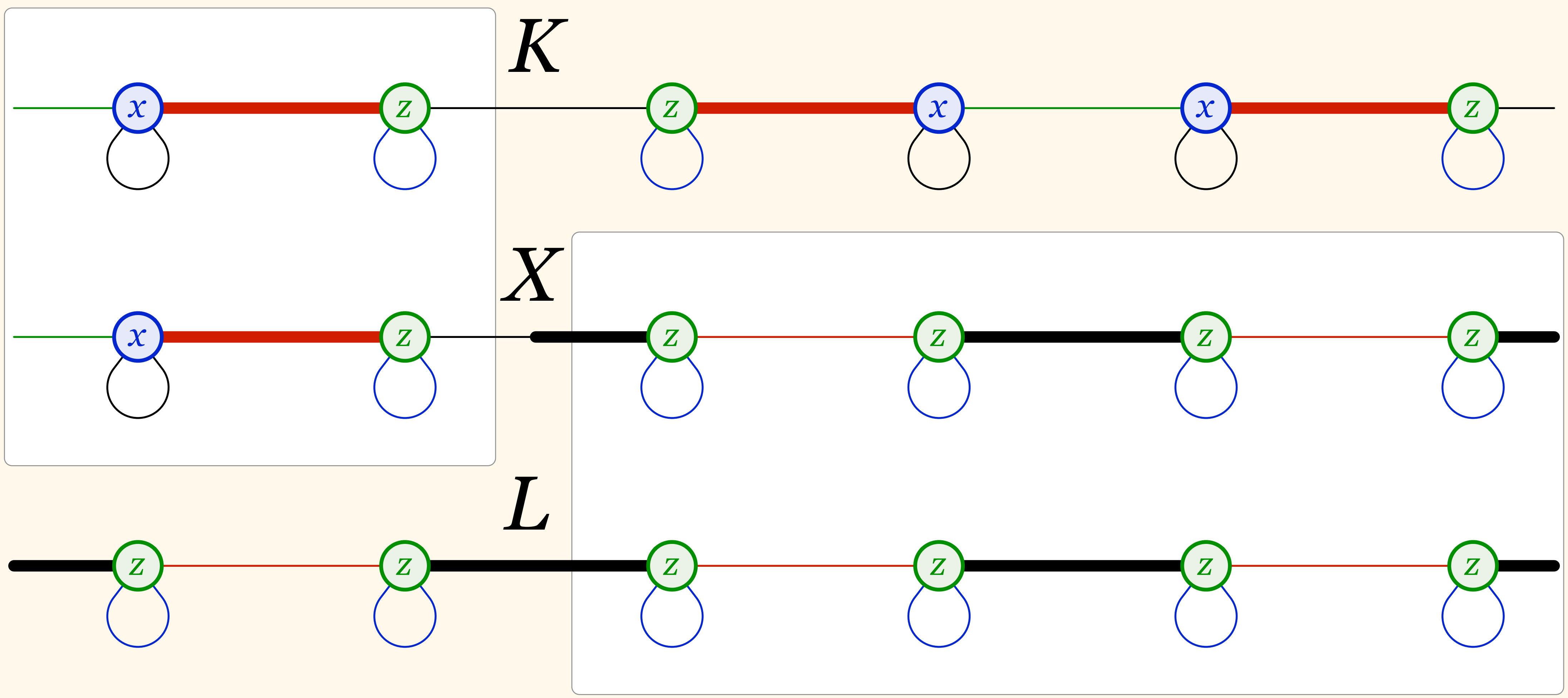
Prefer loops that are matched in T

Then unfold these loops...

Inductive Step

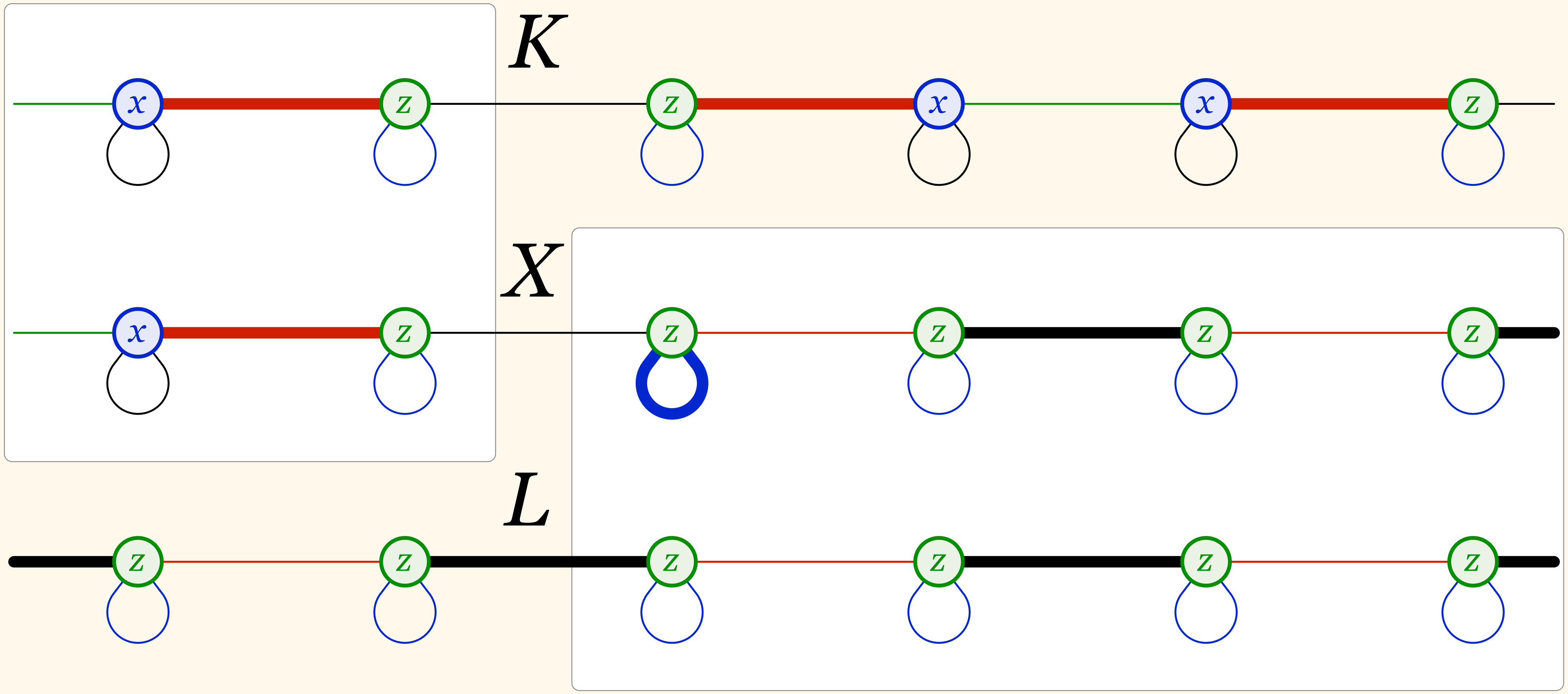


Inductive Step

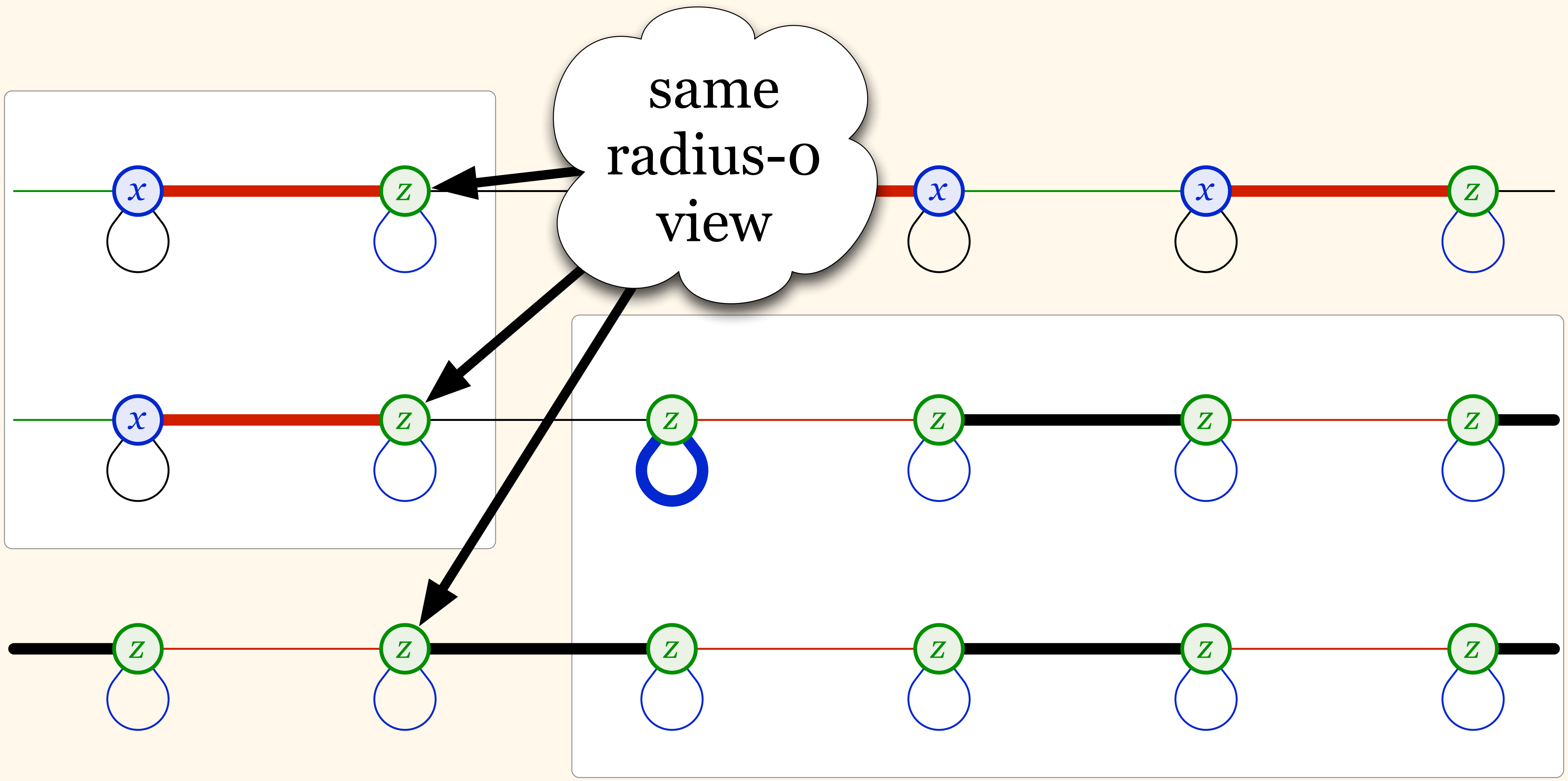


… again, something must change in the output!

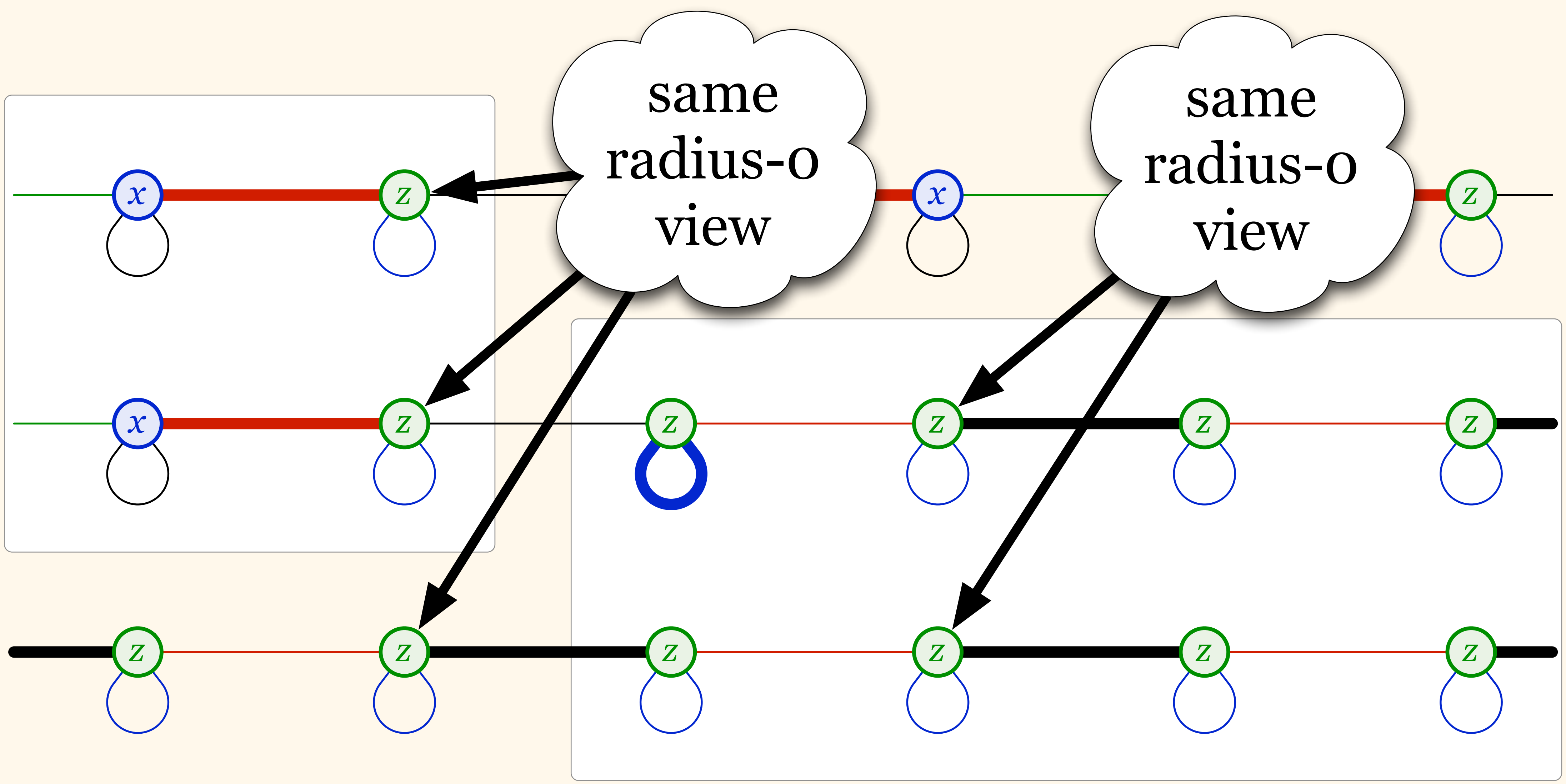
Inductive Step



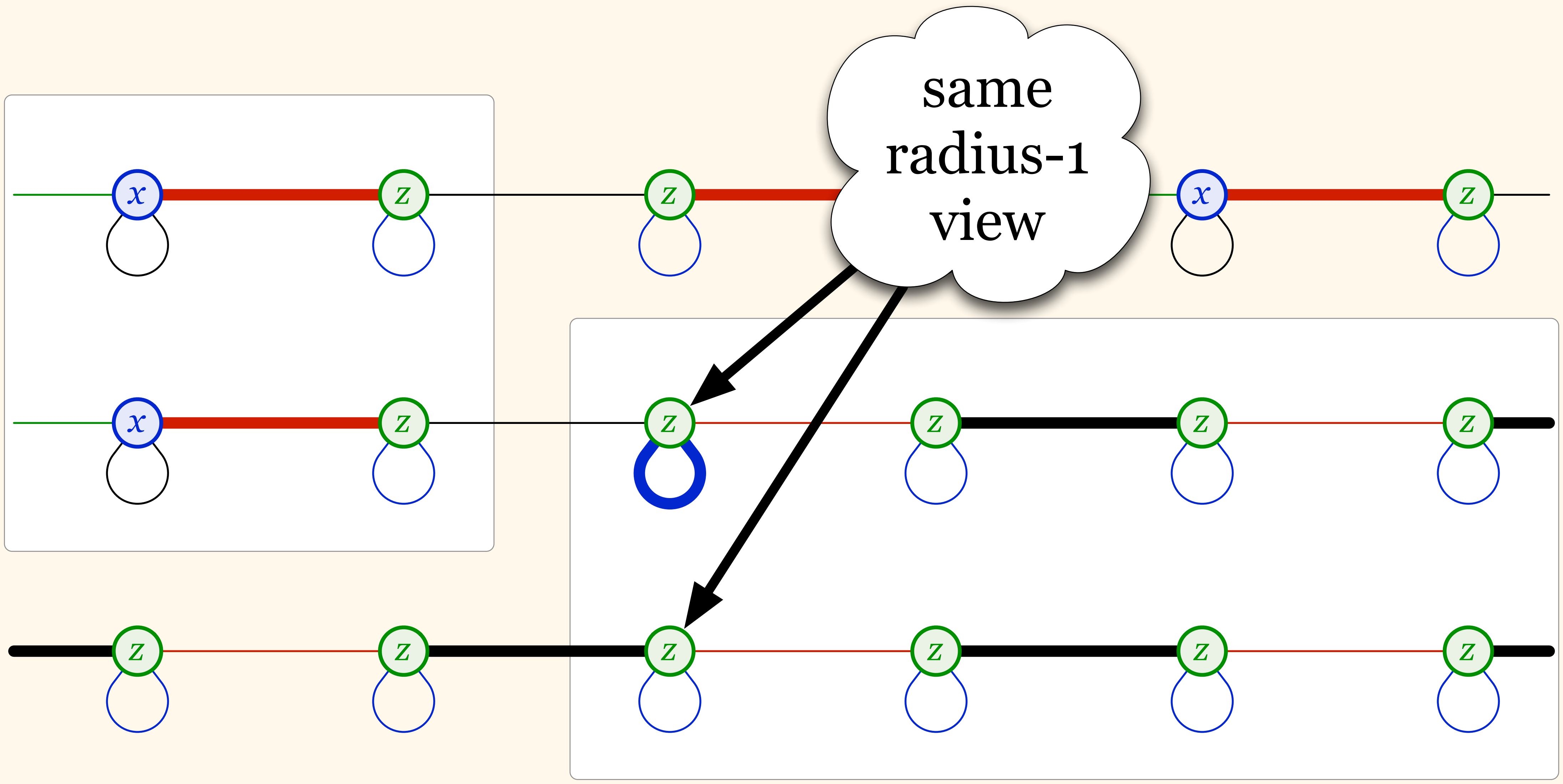
Inductive Step



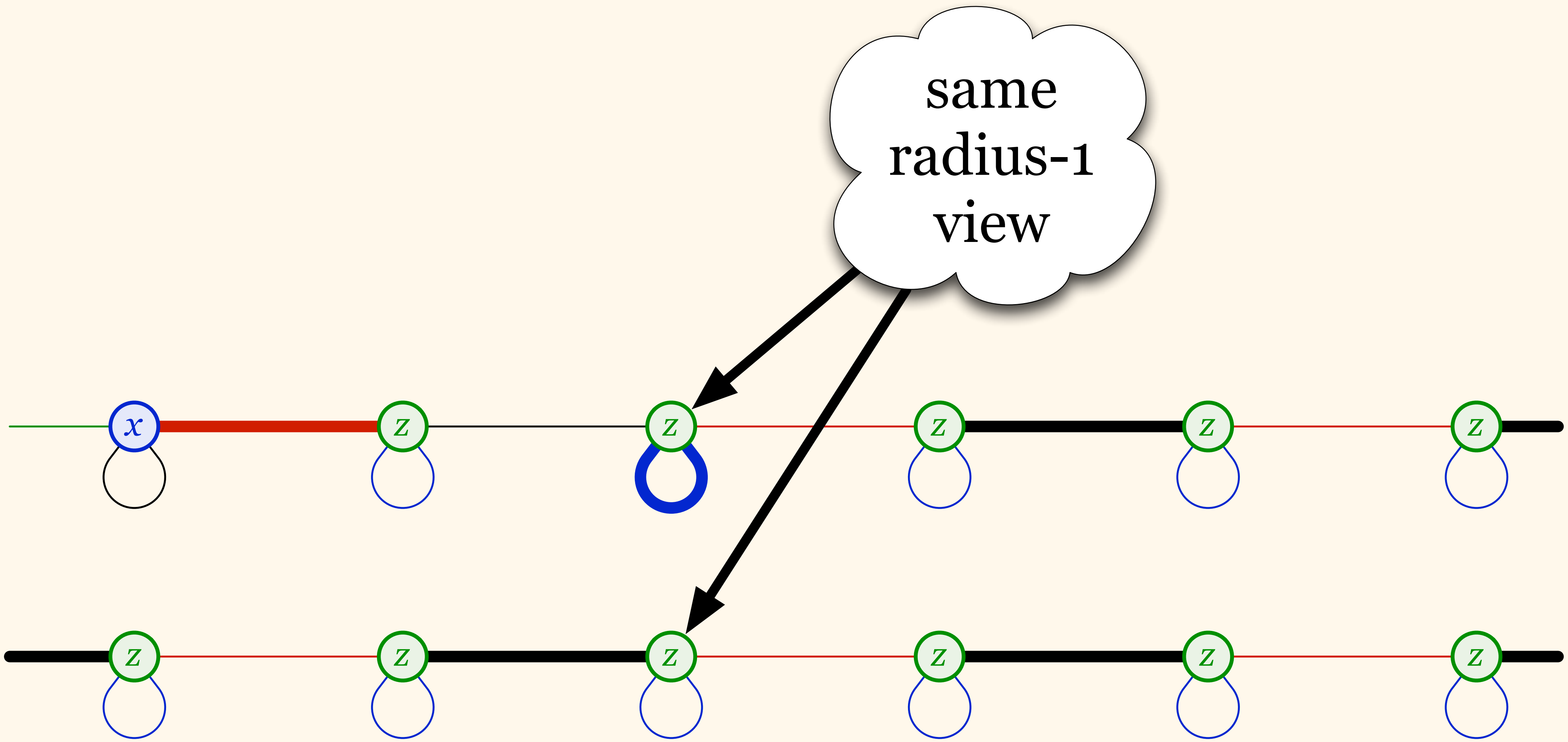
Inductive Step



Inductive Step

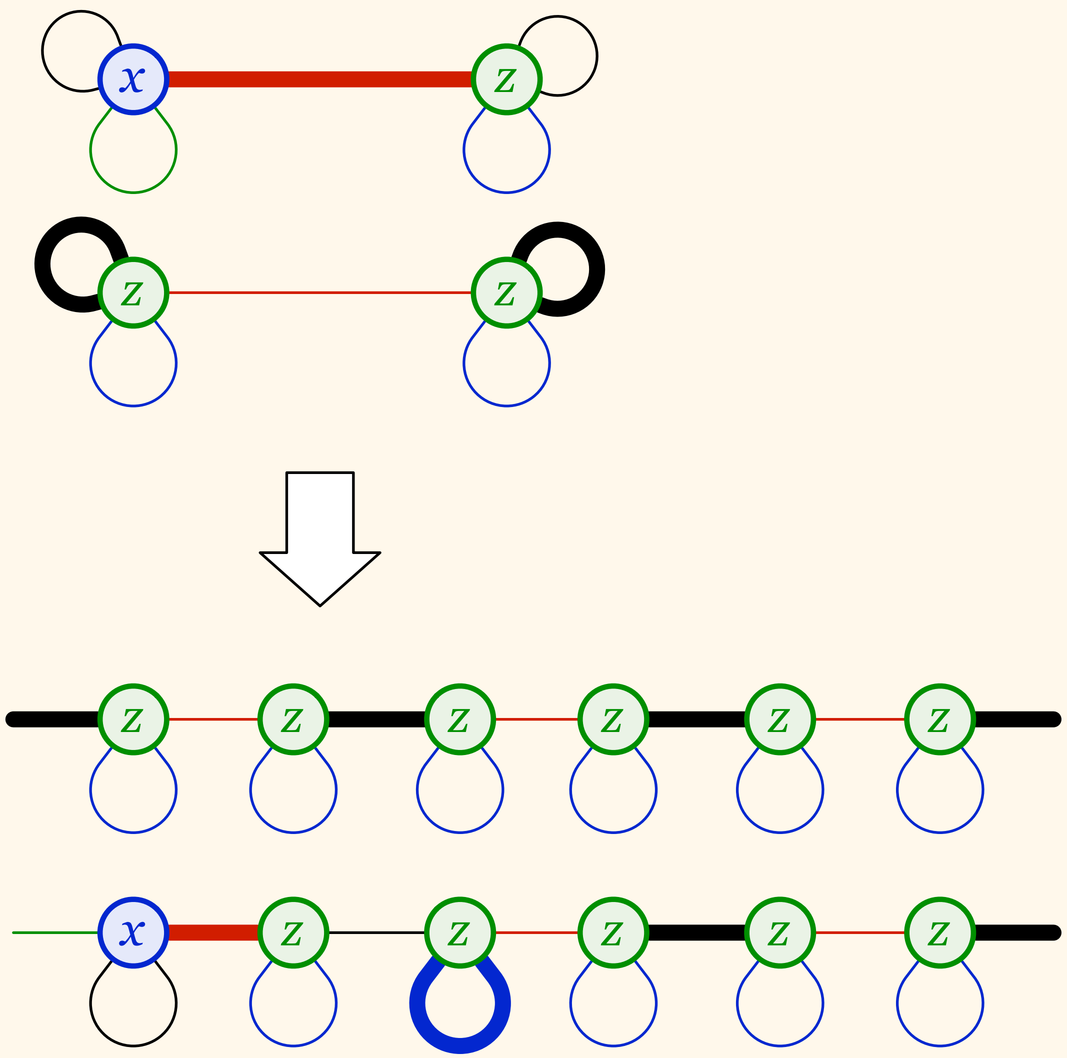


Inductive Step



degree 2 templates, same radius-1 view, different output

Inductive Step

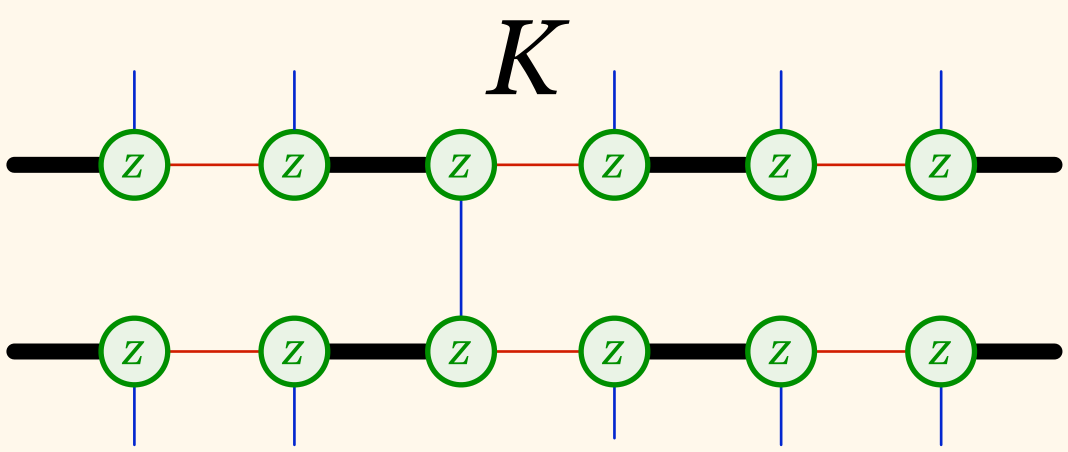


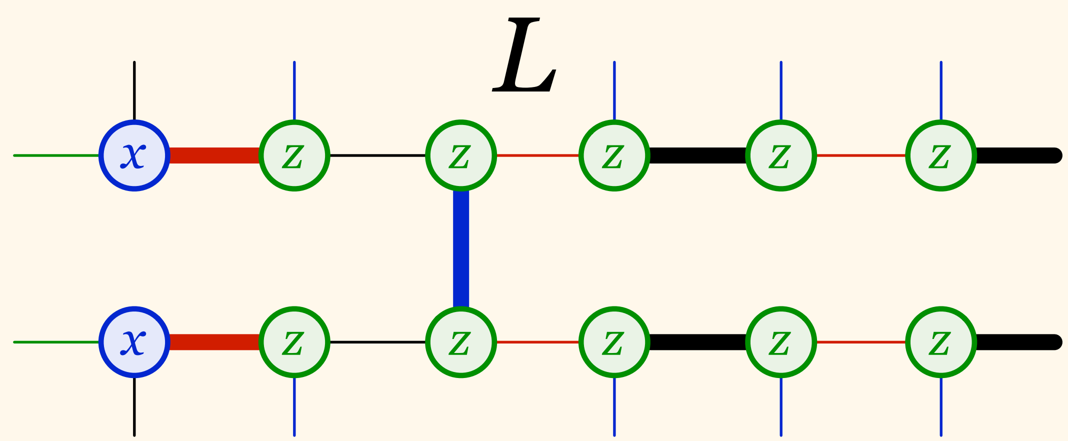
Given:
degree i templates,
same radius-(i–1) view,
different output

Construct:
degree i+1 templates,
same radius-i view,
different output

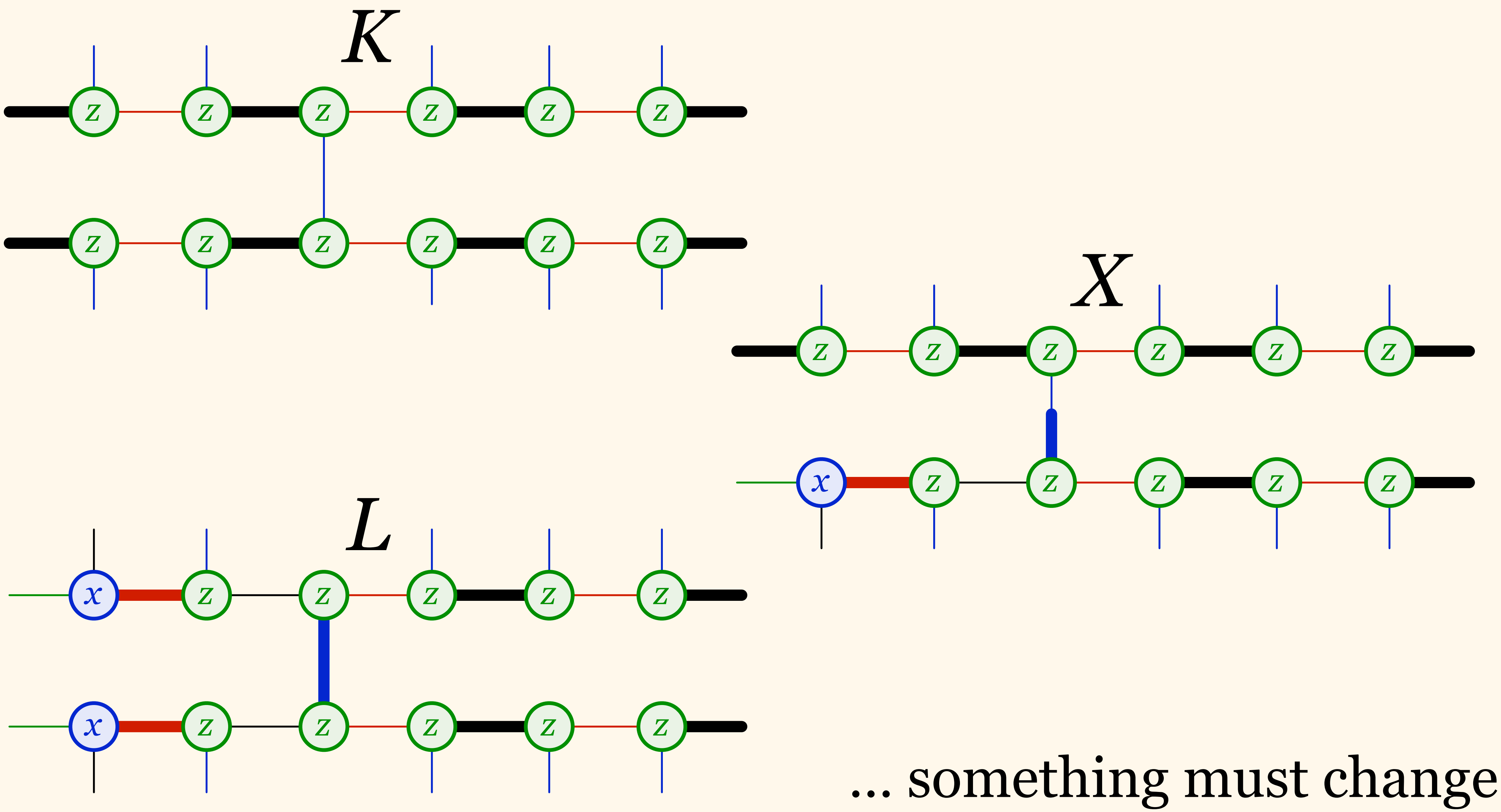
(here i = 1)

Inductive Step



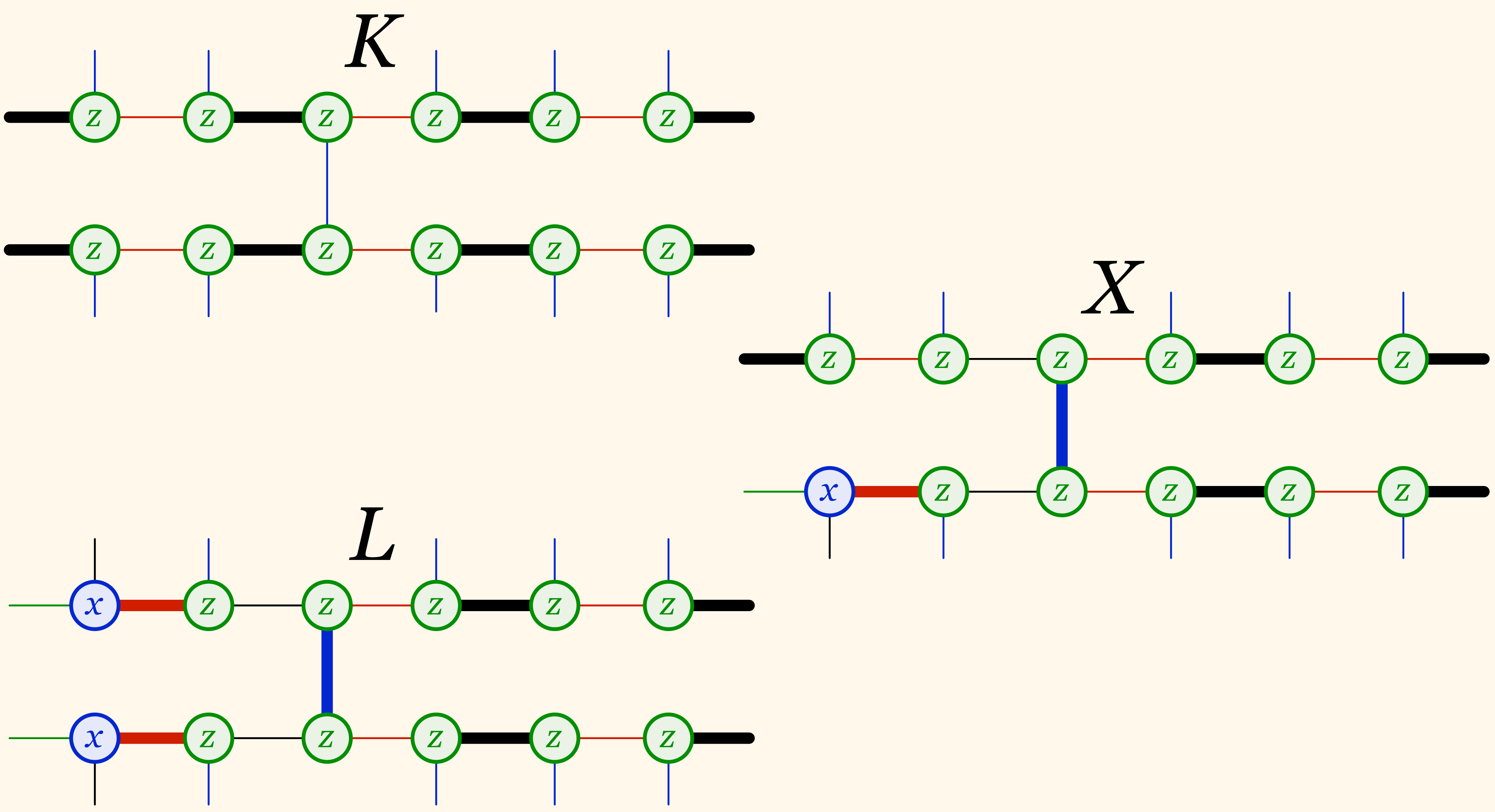


Inductive Step

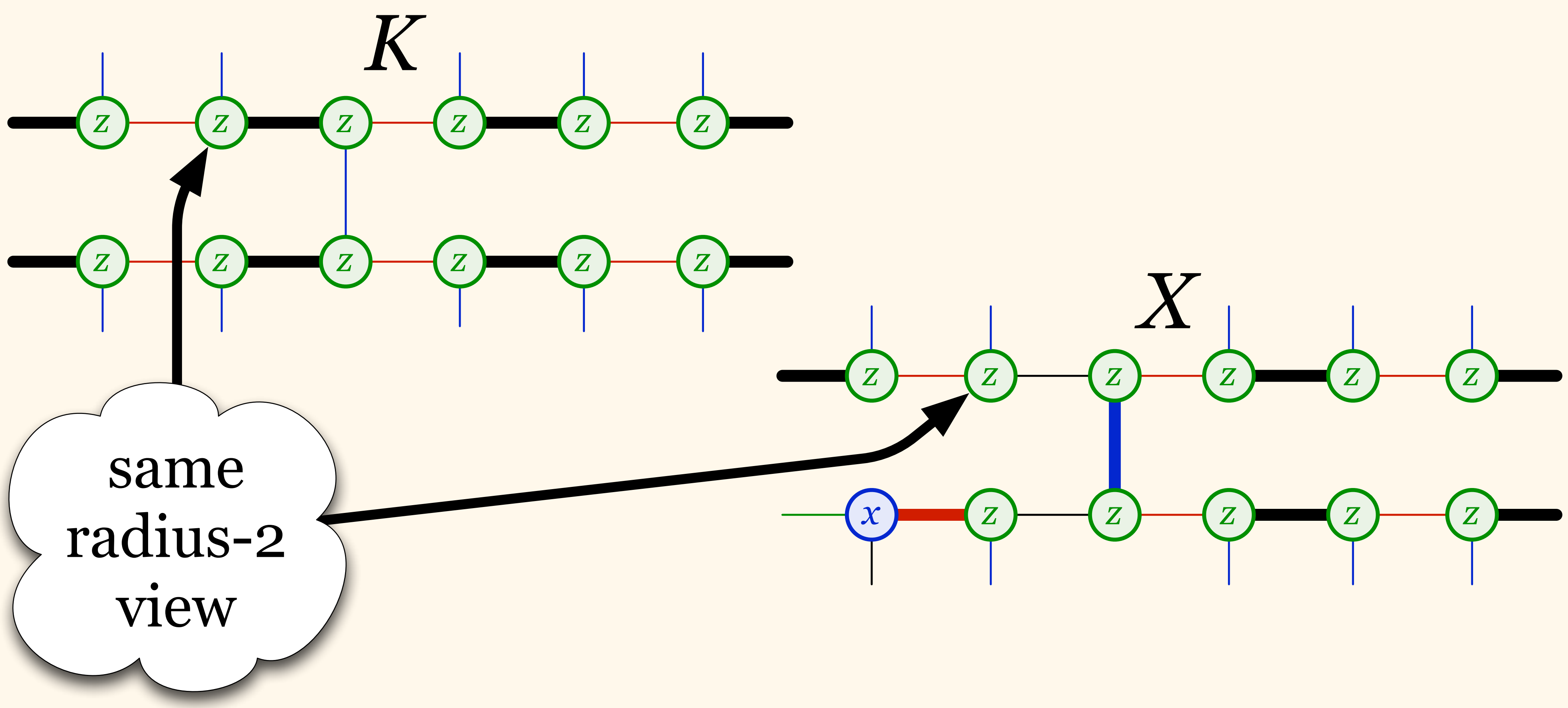


… something must change

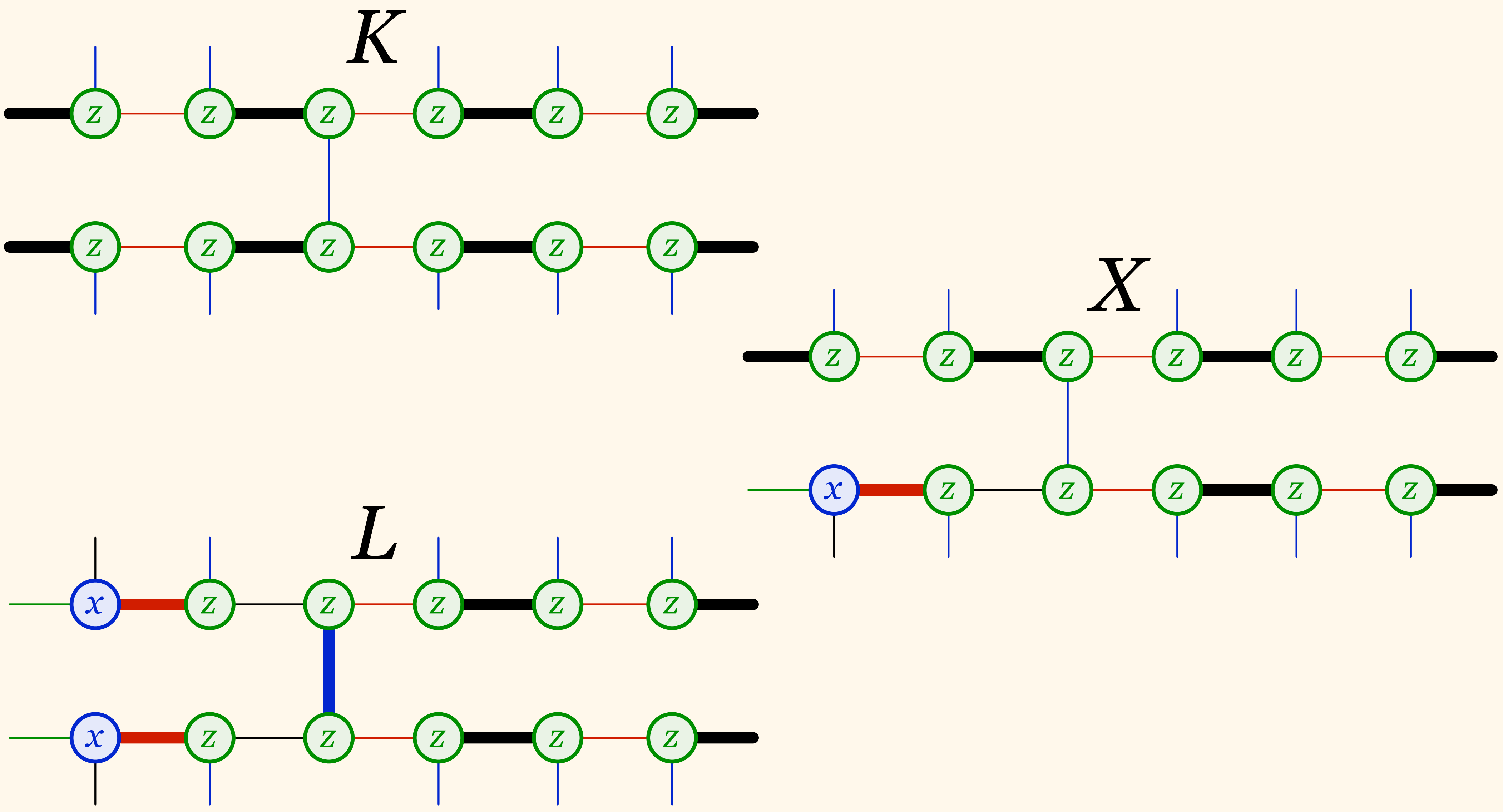
Inductive Step



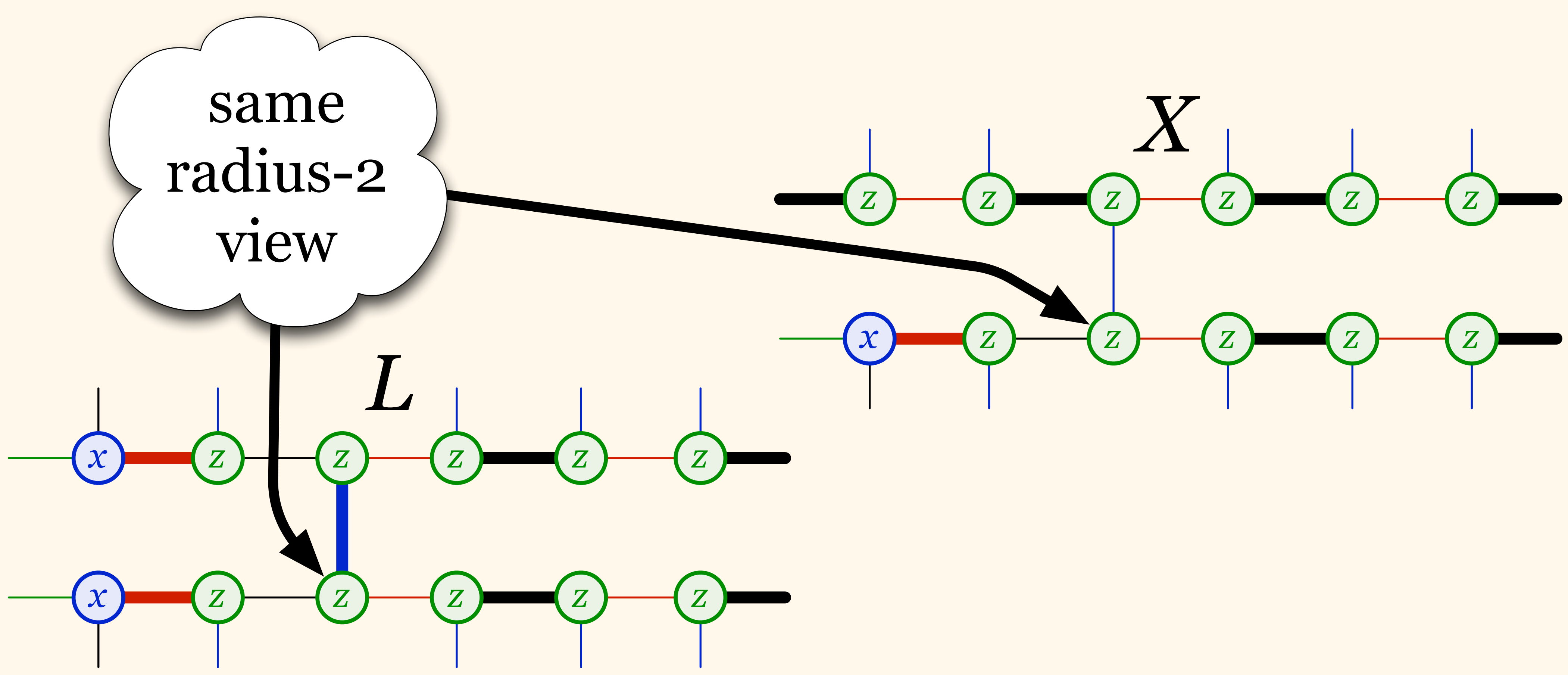
Inductive Step



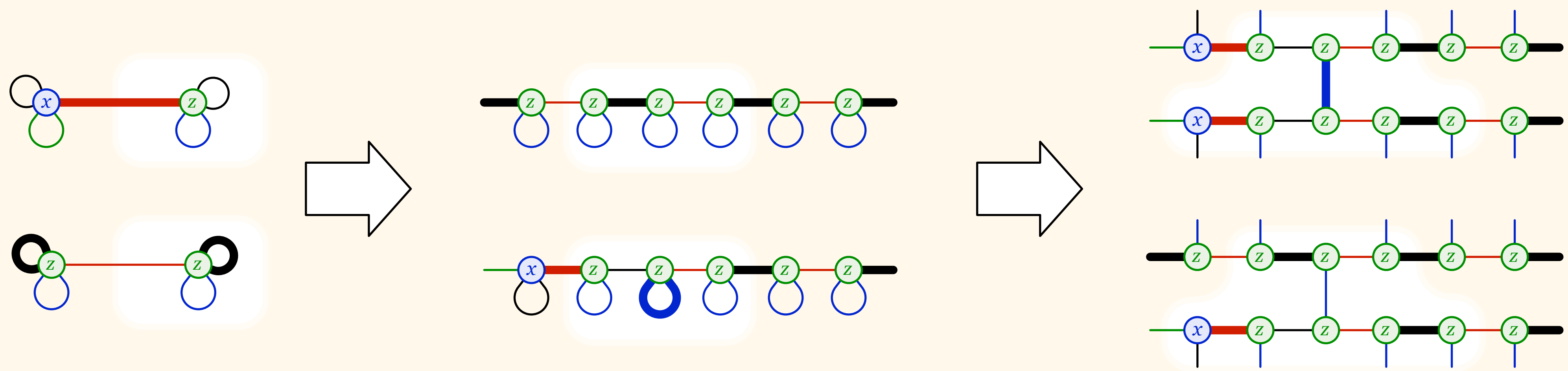
Inductive Step



Inductive Step

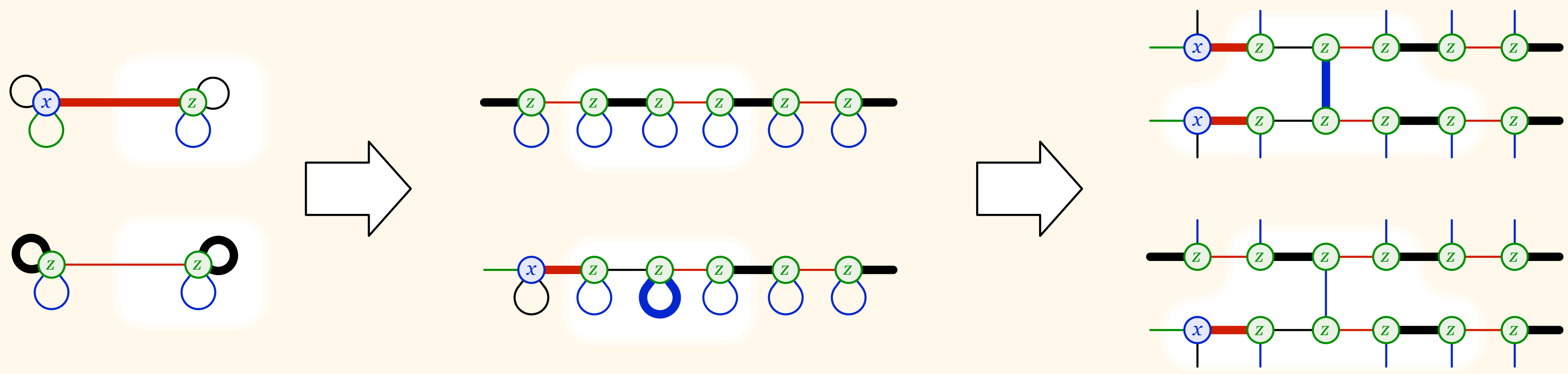


Conclusions



- By induction, we can construct:
  - two degree-d trees
  - same radius-$(d{-}1)$ view
  - different output

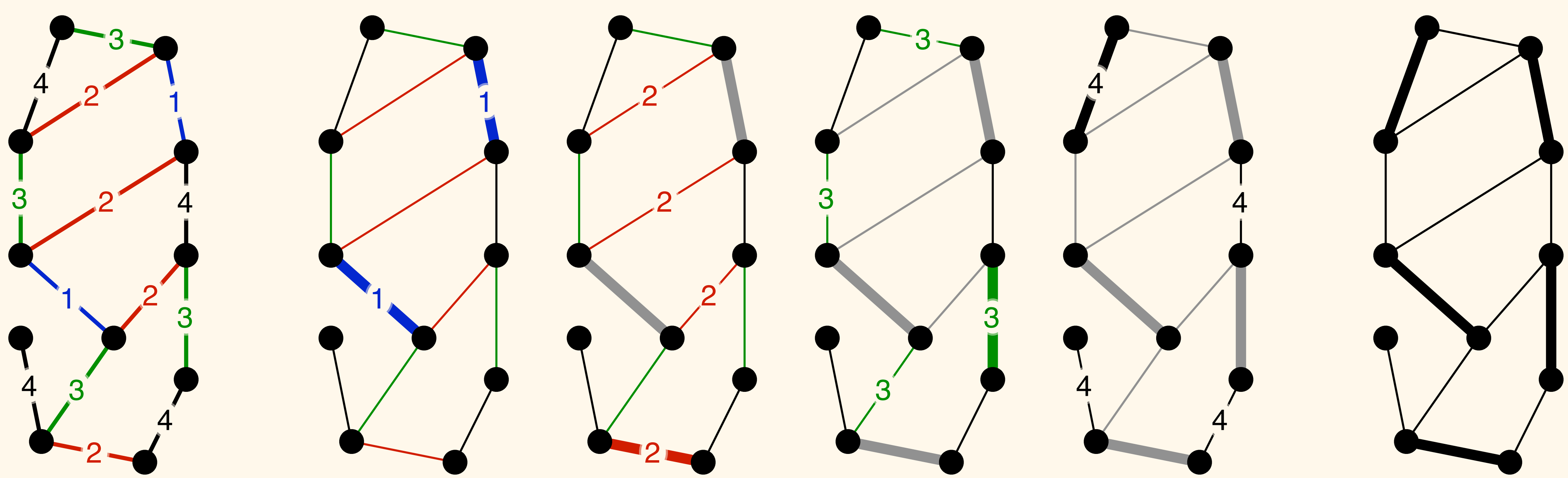
Conclusions



- Algorithm A requires at least $k - 1$ rounds in a k-edge-coloured graph
- Algorithm A cannot be faster than the greedy algorithm

Conclusions

- Greedy is optimal



Conclusions

- Maximal matching in k-edge-coloured graphs requires:
  - $k - 1$ communication rounds in general
  - $\Theta(\Delta + \log^* k)$ rounds in graphs of degree $\leq \Delta$
- Still open:
  - what if we have unique identifiers?
  - or both edge colouring and node colouring?