

Hierarchies of Polynomial Kernelization Hardness

Karolina Soktys

Max Planck Institute for Informatics, Saarbrücken

WoRKer 2011, 3 September 2011, Vienna

joint work with:

Danny Hermelin



Magnus Wahlström



Xi Wu



Excluding Small Kernels

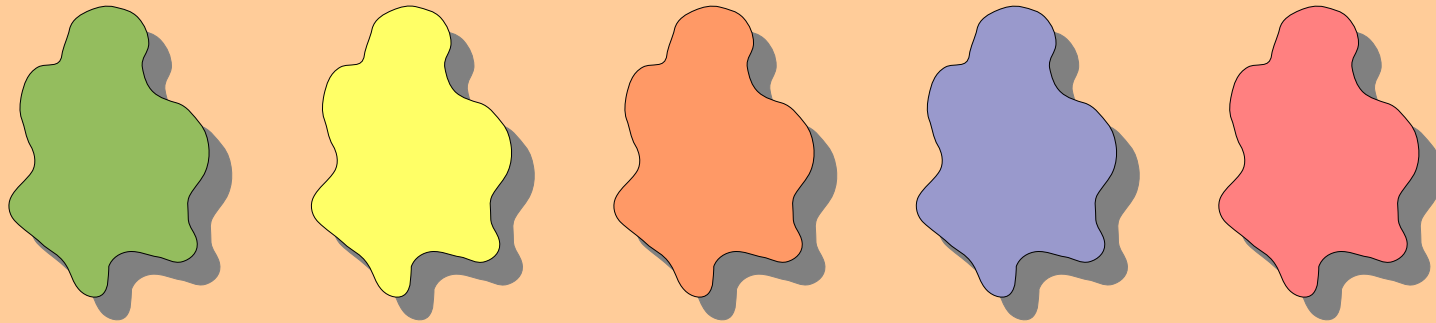
- a powerful framework for proving **inexistence of polynomial kernels** (unless PH collapses) (**BDFH, FS '08**)
- **lower bounds for many problems** (**k-Path, Connected Vertex Cover, Steiner Tree, ...**)
- **in essence: compositional problems have no polynomial kernels**

Compositional Problem

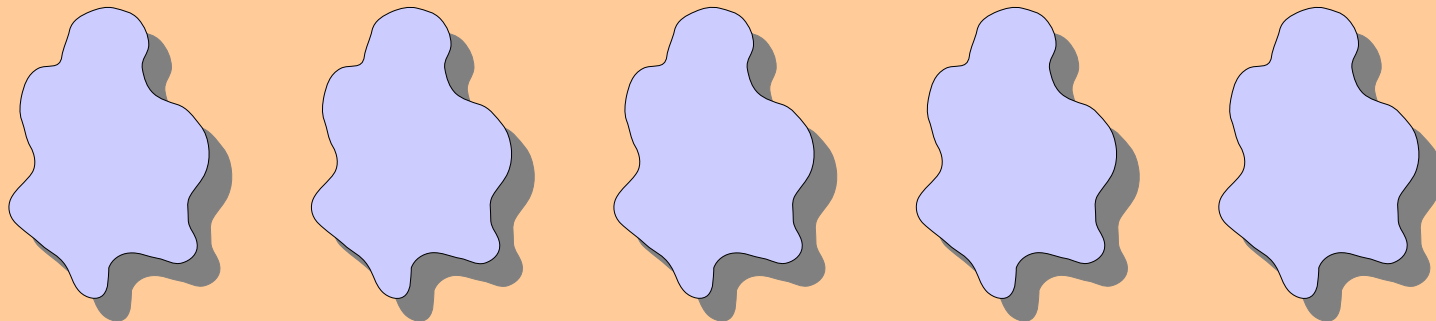
- **we can in polynomial time encode the OR of many instances of the problem into a single instance**
(potentially large) **with a small parameter**
- **example: k-Path**

Compositional Problem

- example: **k-Path**
- does **any** of these graphs contain a path of length **k**?

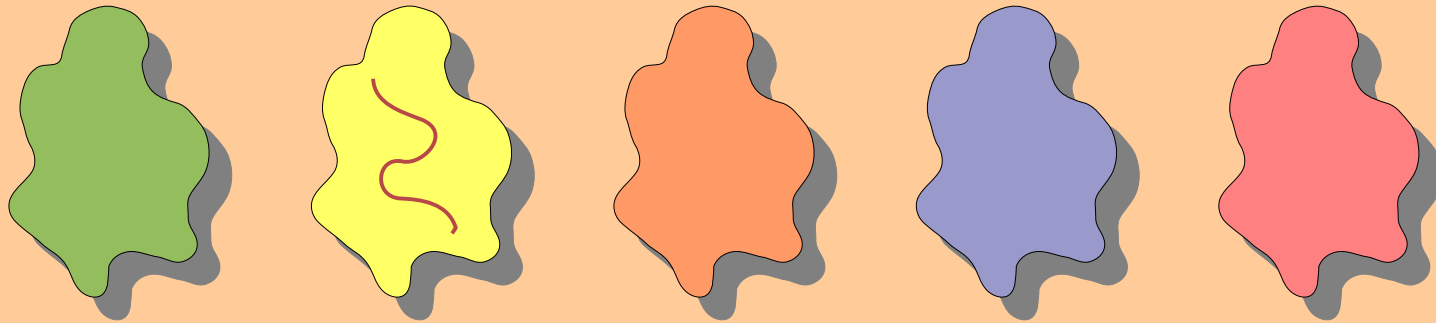


- does **this graph** contain a path of length **k**?

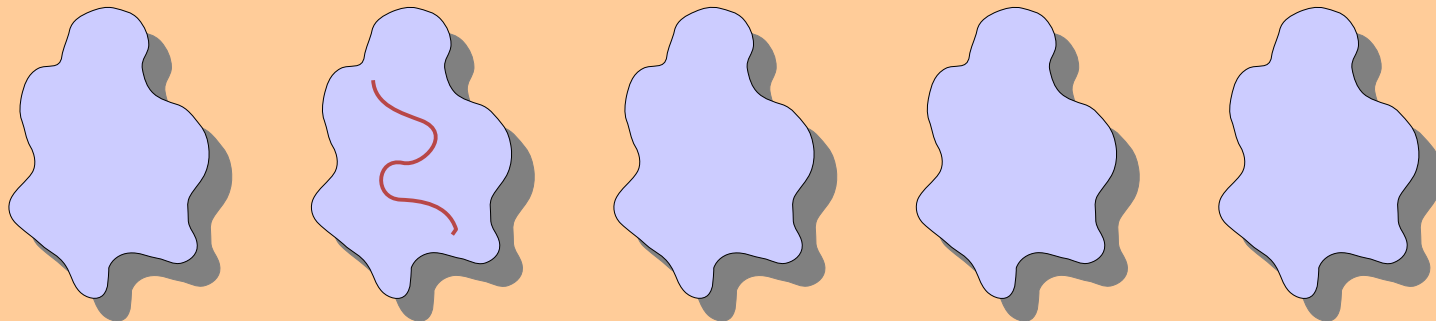


Compositional Problem

- example: **k-Path**
- does **any** of these graphs contain a path of length **k**?



- does **this graph** contain a path of length **k**?

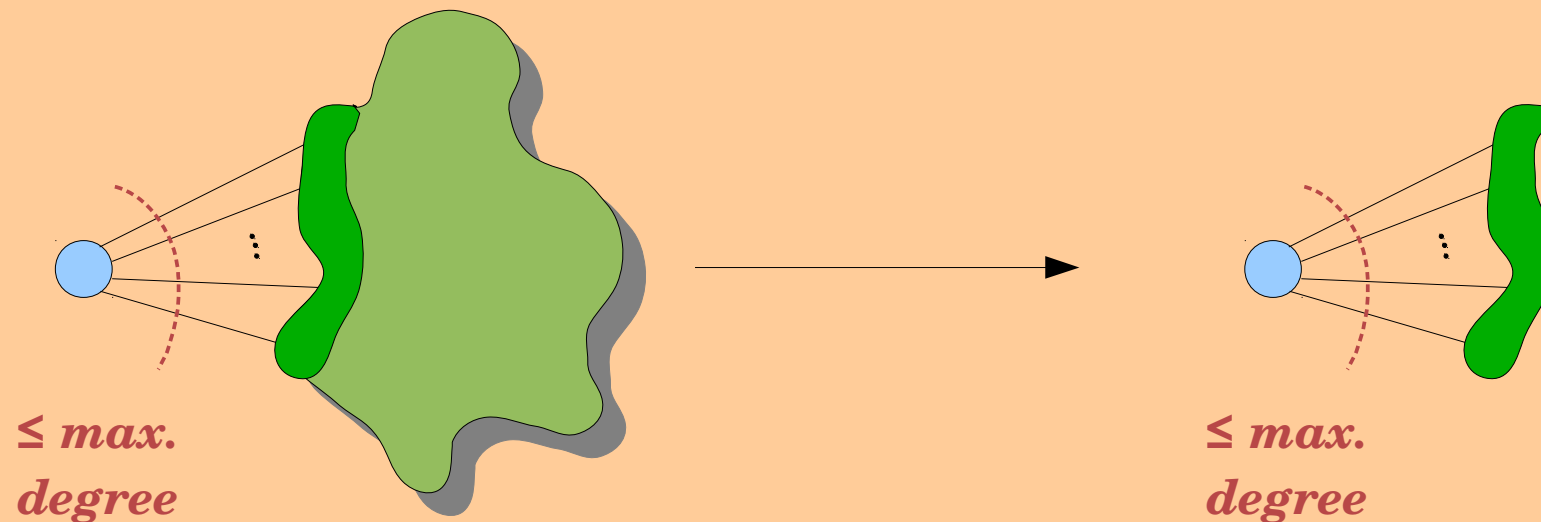


Excluding Small Kernels

- a powerful framework for proving **inexistence of polynomial kernels** (unless PH collapses) [BDFH, FS '08]
- but sometimes there exists a polynomial **“cheating” kernel: an OR-kernel** (Max-Leaf Out-Branching, Clique(Max. Degree)) **OR**, more generally, a **Turing Kernel**
- and **subexponential kernels** could also be possibly **useful in practice**
- **how to exclude such kernels?**

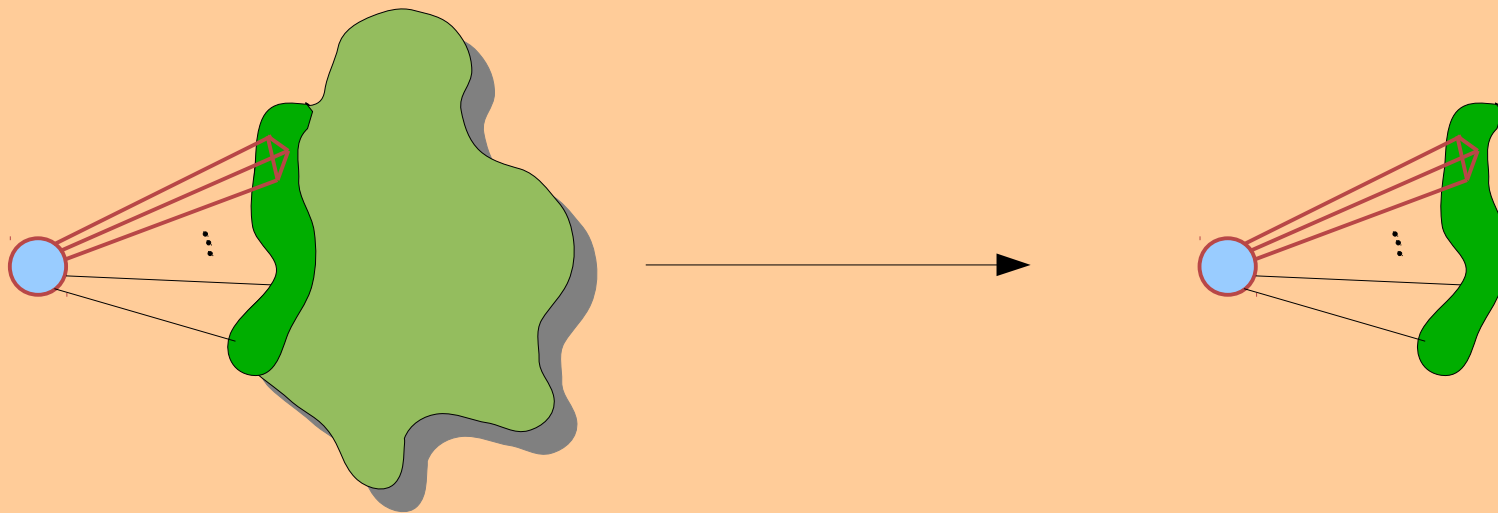
A “Cheating” Kernel

- **Clique(Max. Degree)** might not have a polynomial kernel...
- ... but it has one if we **localize the problem**: i.e. require that a given vertex v is in the clique



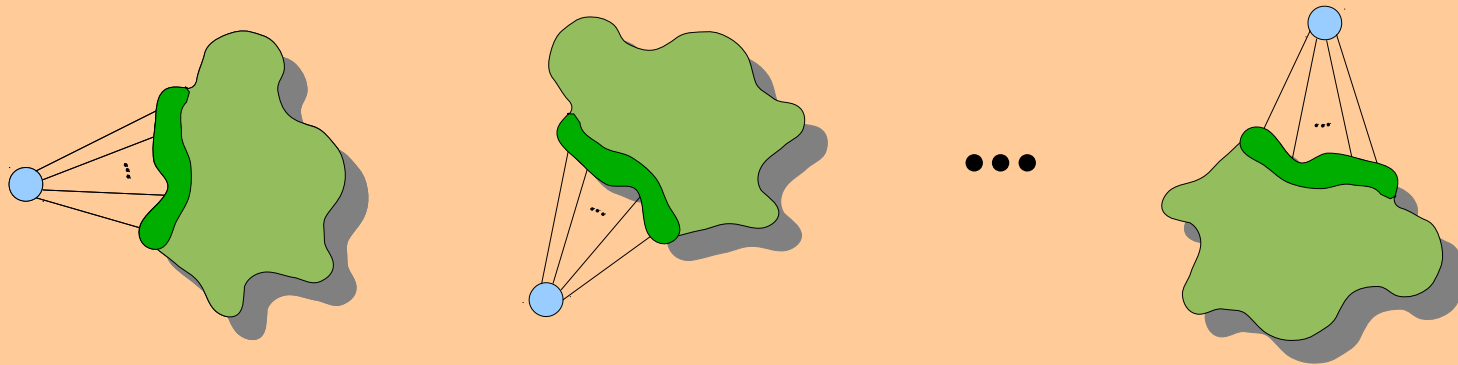
A “Cheating” Kernel

- **Clique(Max. Degree)** might not have a polynomial kernel...
- ... but it has one if we **localize the problem**: i.e. require that a given vertex v is in the clique



A “Cheating” Kernel

- **Clique(Max. Degree)** might not have a polynomial kernel...
- ... but it has one if we **localize the problem**: i.e. require that a given vertex v is in the clique
- ... and we can **ask this question about v_1, v_2, v_3, \dots**



- this “kernel” is still very **useful for FPT algorithms**

Excluding “Cheating” Kernels

- **seems difficult**
- **complexity theorist's approach:**
 - **find a good candidate for a hard problem**
 - **assuming it is indeed hard, prove hardness of other problems**
(so we need reductions preserving hardness)
 - **PPT reductions** (akin to standard FPT reductions, but changing the parameter at most polynomially) **do the trick**
 - **this yields a class of problems** (probably) **not admitting “cheating”** (Turing, subexponential) **kernels**

Hard Nuts to Kernelize

- **candidate hard problems**
 - **FPT reparametrizations of basic non-FPT problems:**
 - **Clique ($k \log n$)**
 - **NP problems verifiable in time polynomial in the witness size**
 - **k-Step Turing Machine Halting Problem**
 - **problems for which solution consistency is checked locally**
 - **Min Ones q-SAT**
 - **problems convenient to work with to use in future reductions :)**
 - **(Exact) Hitting Set (m)**

Hard Nuts to Kernelize

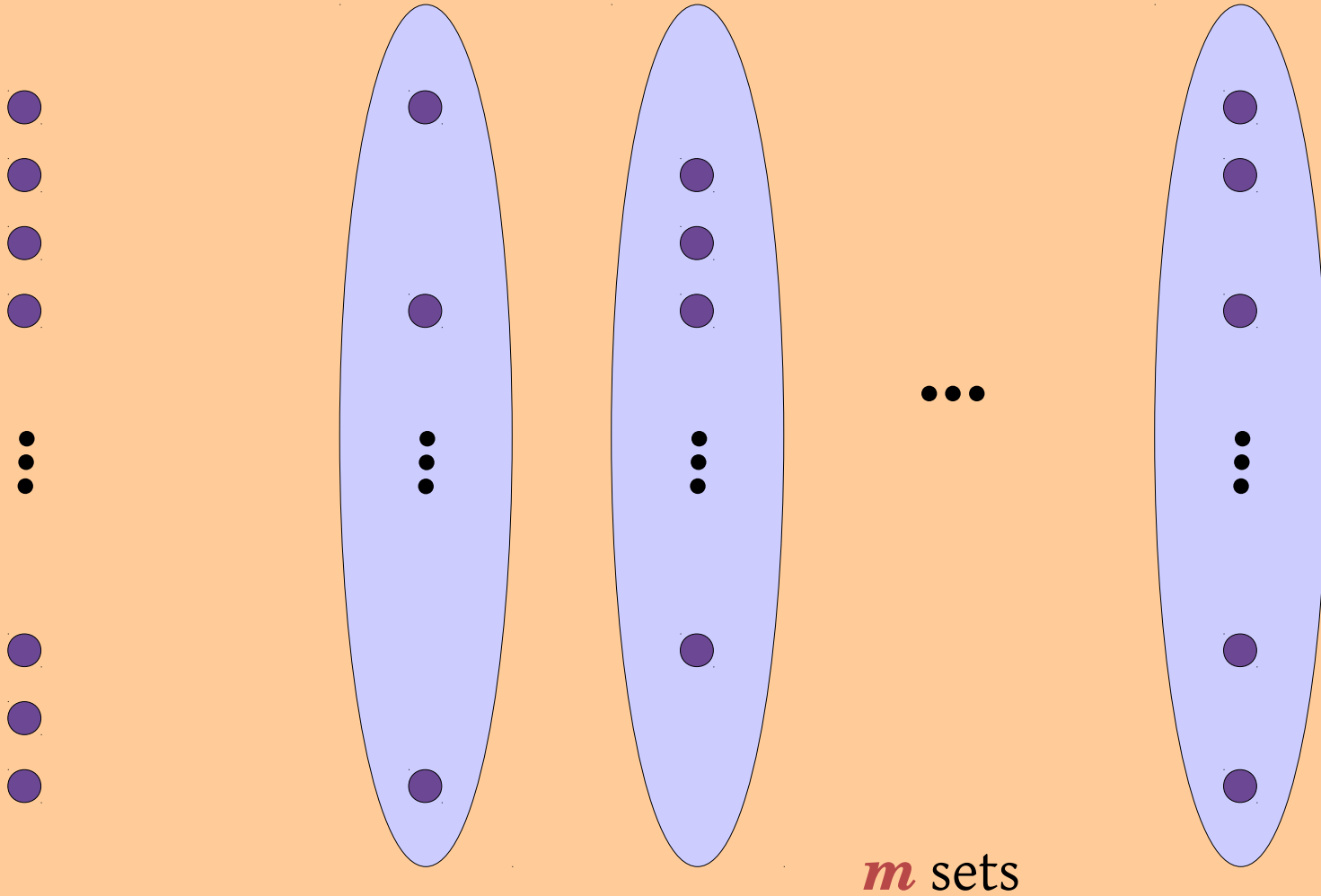
- **candidate hard problems**
 - **Clique ($k \log n$)**
 - **k-Step Turing Machine Halting Problem**
 - **Min Ones q-SAT**
 - **(Exact) Hitting Set (m)**
- **fortunately, we can prove they are all equivalent!**

Formal Definition

- **WK[t]** = [Weight-t Weighted SAT ($k \log n$)]_{≤ppt}
 - **WK[1]** is the class of problems that we discuss in this talk
- **MK[t]** = [Weight-t SAT (n)]_{≤ppt}
 - **MK[2]** captures Hitting Set (n)
- **PK** = **MK[1]** ⊆ **WK[1]** ⊆ **MK[2]** ⊆ **WK[2]** ⊆ ... ⊆ **EXPT**
- hierarchies for kernelizability analogous to W- and M-
hierarchies for fixed-parameter tractability

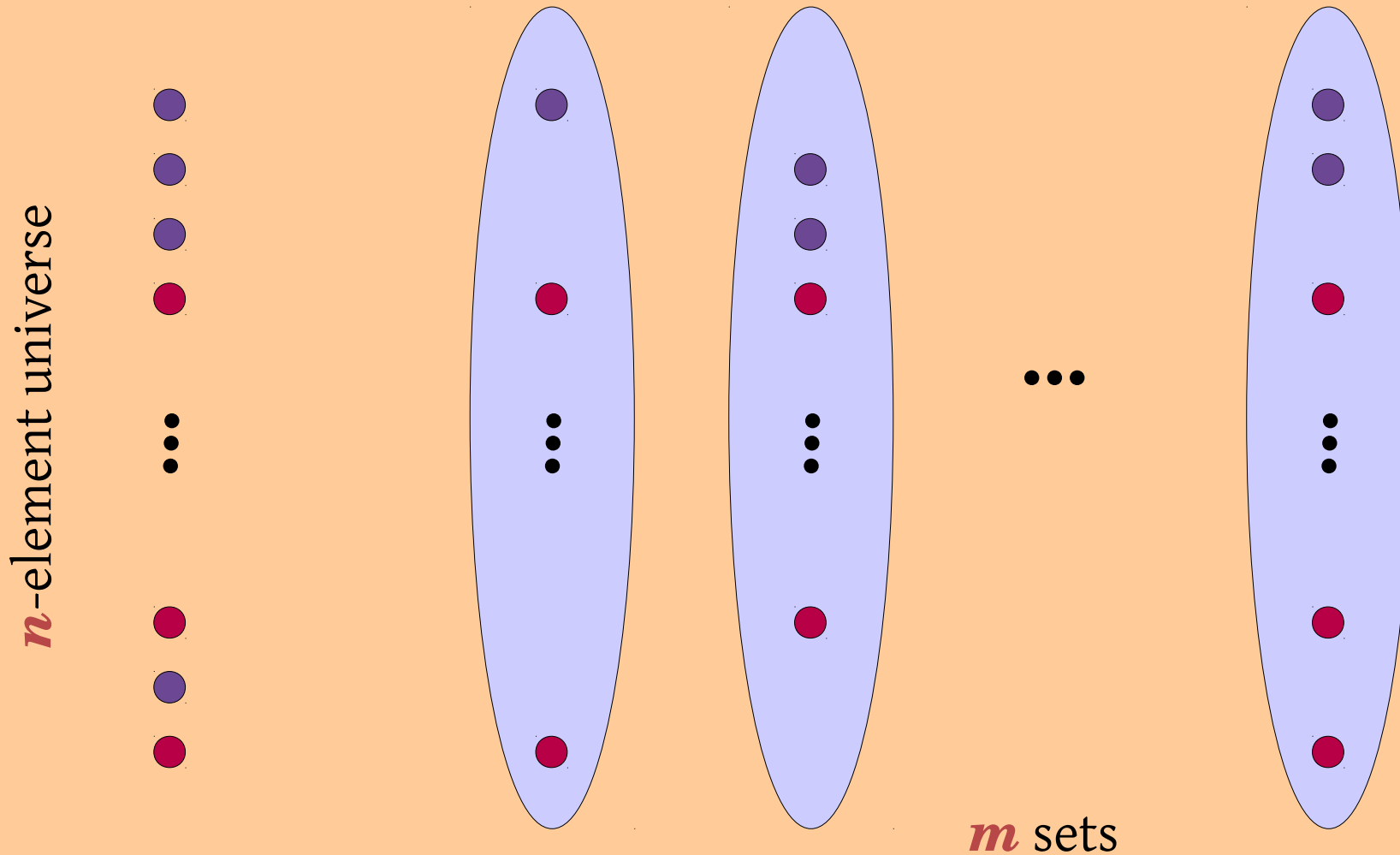
Hitting Set (m)

n -element universe



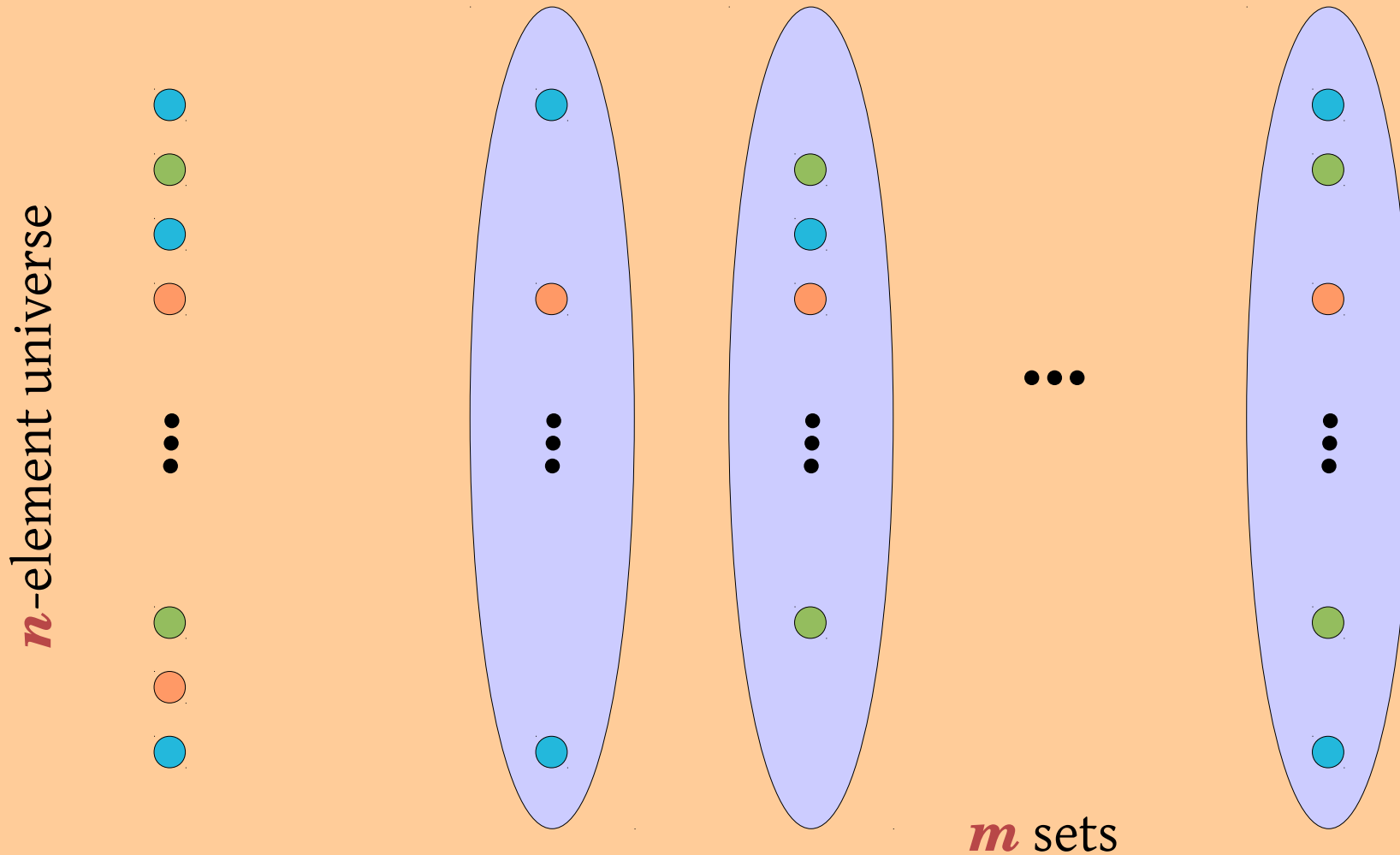
Hitting Set (m)

- select k elements to hit all sets



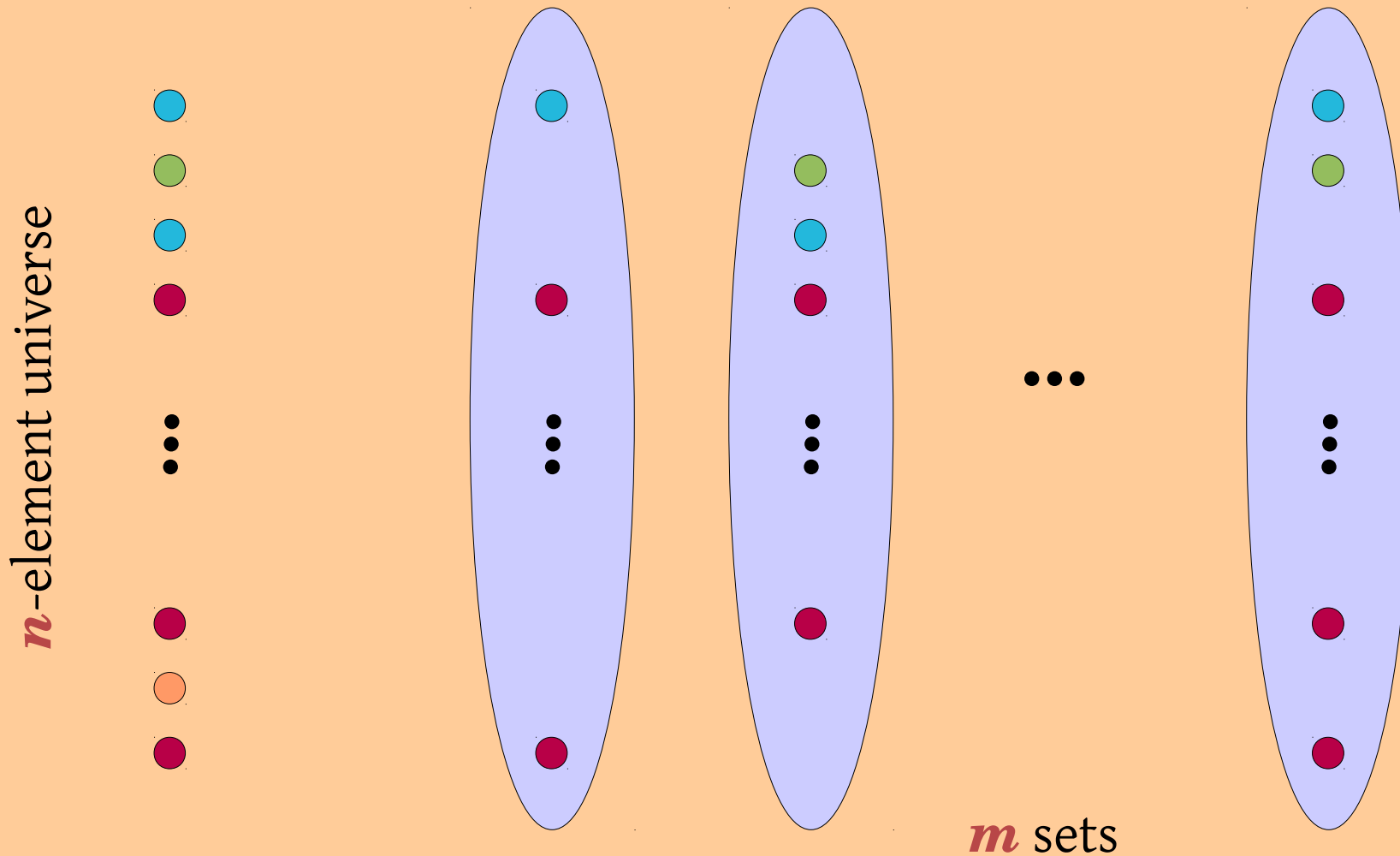
Multicolored Hitting Set (m)

- **technical modification** (elements come in k colors, select one of each)



Multicolored Hitting Set (m)

- **technical modification** (elements come in k colors, select one of each)



One Example of Equivalence

- (Multicolored) **Clique** ($k \log n$) \leq_{ppt} (Exact) **Hitting Set** (m)
 - we assign to vertices **binary IDs** (of length $\log n$)
 - elements correspond to edges
 - sets require exactly one edge linking a given pair of colors...
($\binom{k}{2}$ sets)
 - ... and ensure that any two edges using a color use the same
vertex ($\binom{k}{3} \log n$ sets)

One Example of Equivalence

- **Hitting Set** (m) \leq_{ppt} (Multicolored) **Clique** ($k \log n$)
 - w.l.o.g. we have at most exponential number of elements (n)
 - we create a graph with m color classes (one for every set)...
 - ...and $O(n)$ vertices (one for every pair (S,e) where $e \in S$)
 - a vertex (S,e) is selected to the clique iff S is hit by e
 - edges ensure that two sets are hit either by the same element, or by two elements from the symmetric difference

Other WK(1)-Hard Problems

- **Set Cover (n), Bipartite Dominating Set ($|T|$)**
- **Local Circuit SAT**
- **Disjoint Cycles, Disjoint Paths**
- **Chromatic Number ($|Vertex\ Cover|$), Weighted Feedback Vertex Set ($|Vertex\ Cover|$)**
- **Steiner Tree, Connected Vertex Cover, Capacitated Vertex Cover**
- **Small Subset Sum**
- **Tree-Contractibility**
- **3-SAT (Horn-backdoor), 3-SAT (2-CNF-backdoor)**
- **$\{0,1\}$ -CSP (width)**

Multicolored k-Path

- **Disjoint Factors**

- **input: n-character string over k-ary alphabet**
- **task: find k non-overlapping factors: subwords of the form $a \dots a$ for every symbol a**
- **binary selection gadget: $a \dots a \dots a$**
- **WK[1]-hard via a reduction from Hitting Set (m) – see blackboard**

- **Multicolored k-Path**

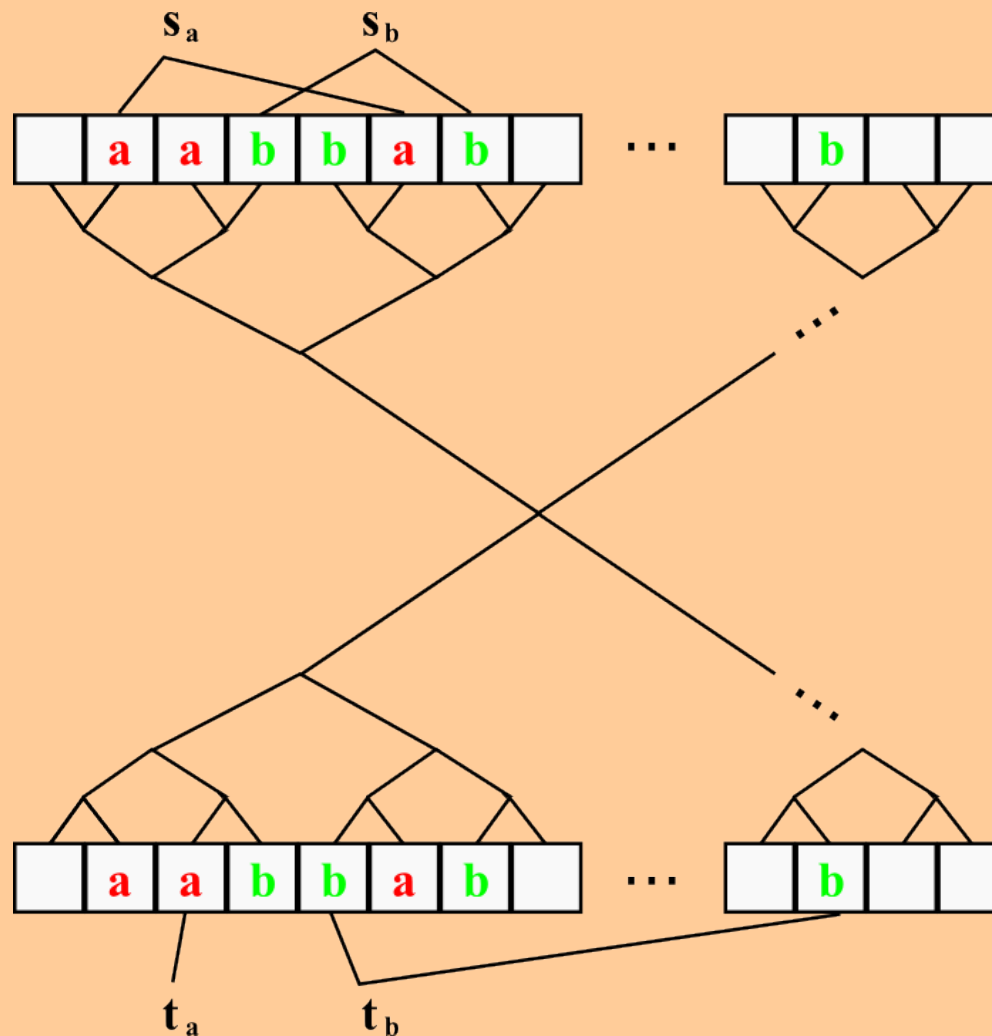
- **Multicolored Directed k-Path – easy reduction from Disjoint Factors – see blackboard**
- **we can go to the undirected case, but we still need colors**

Disjoint Short Paths

- **reduction from Non-Nested Factors...**
 - a technical gap problem which can be proven to be WK[1]-hard
- **...to Disjoint Short Paths**
 - we want to find a path of length k' for each of k terminal pairs
 - non-crossing paths will correspond to non-nested factors

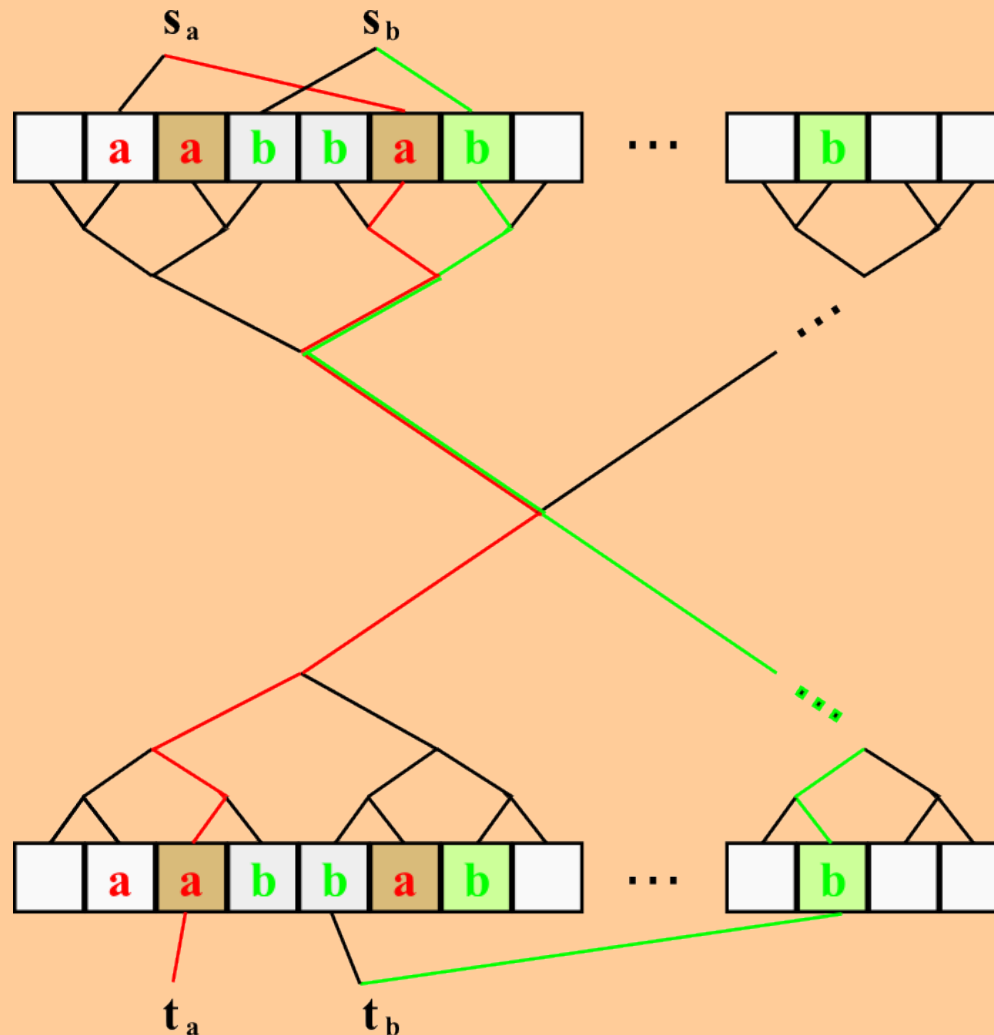
Disjoint Short Paths

- the high-level structure for Disjoint Short Paths



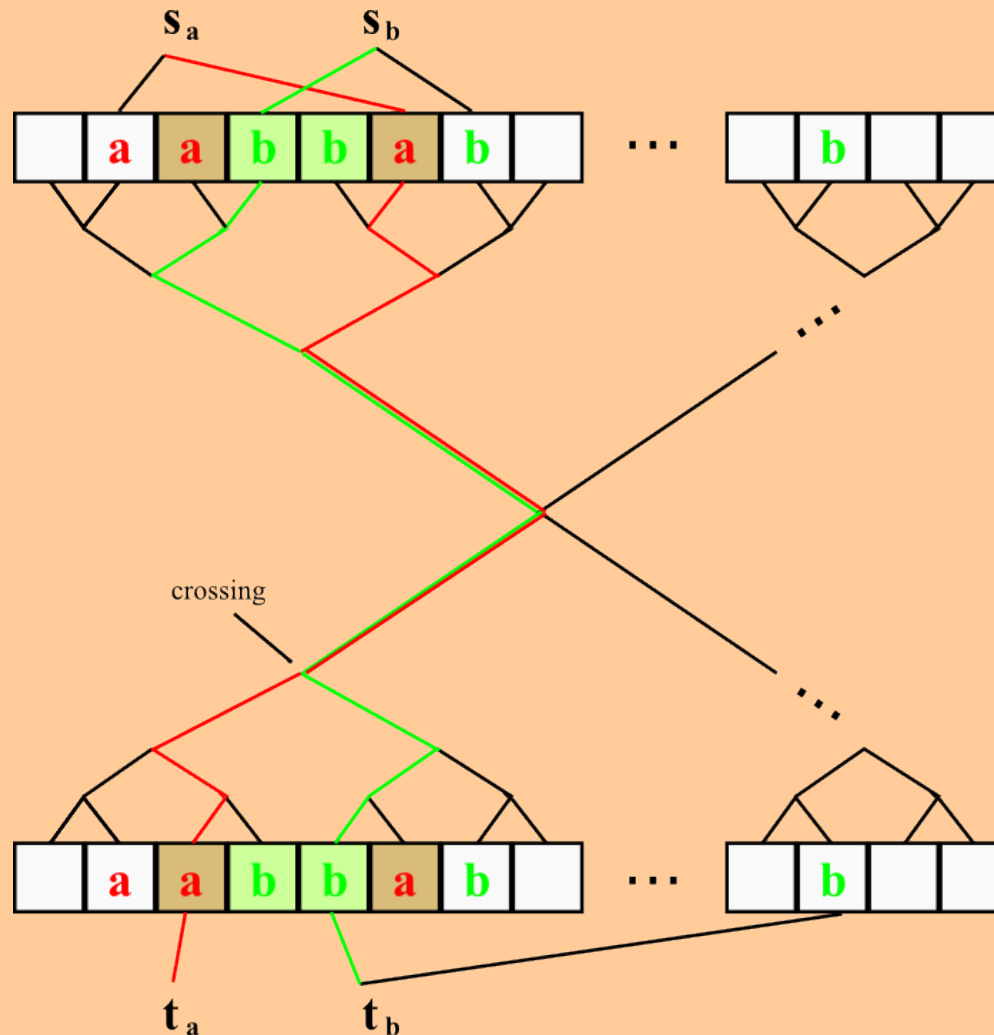
Disjoint Short Paths

- paths corresponding to **non-overlapping** factors can go **alongside**...



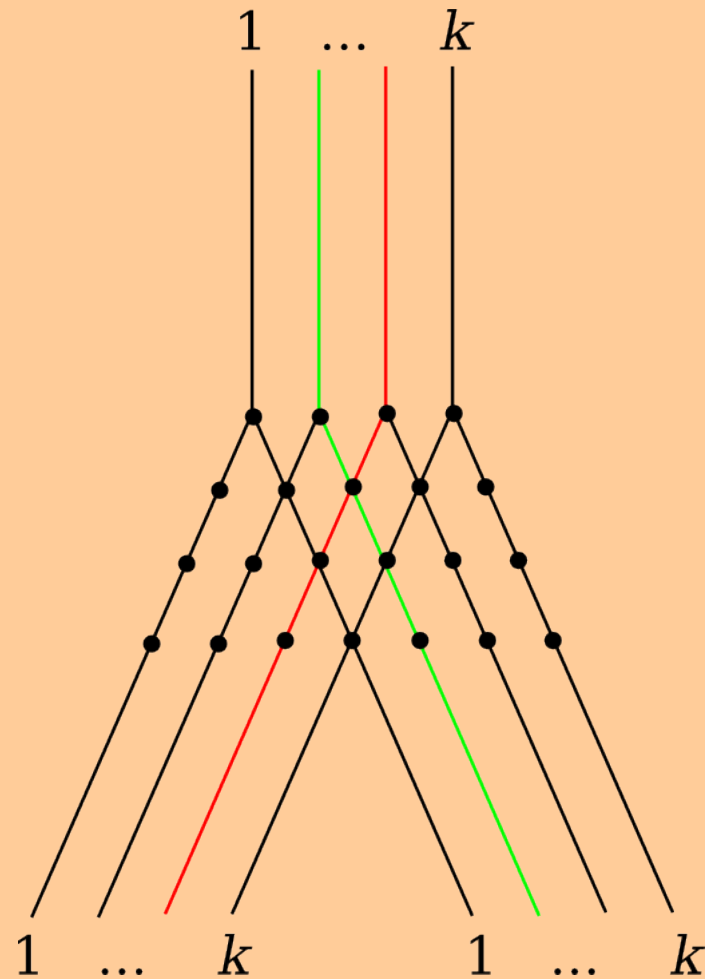
Disjoint Short Paths

- ...while paths corresponding to **nested** factors always have to **cross**



Disjoint Short Paths

- replacing edges by **k-lane highways** allows paths to go alongside



Open Problems

- **k-Path, Directed k-Path**
- **(might be difficult) prove some separation results**

Thank you! Questions?