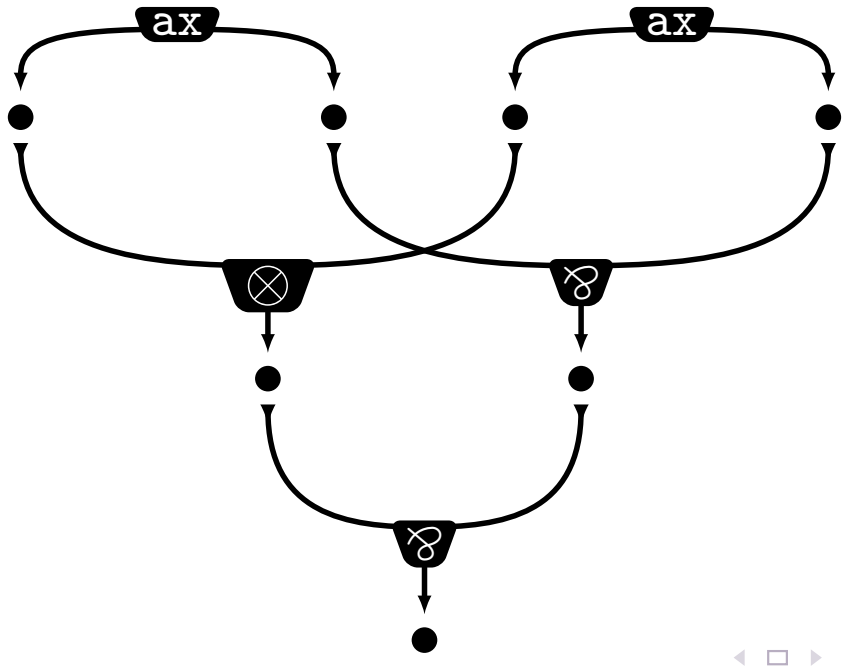
On the complexity of finding cycles in proof nets

Nguyễn Lê Thành Dũng
École normale supérieure de Paris & LIPN, Université Paris 13
nltd@nguyentito.eu

Developments in Implicit Computational Complexity (DICE)
Thessaloniki, April 14th, 2018

Proof structures and proof nets

- A *proof structure* is a sort of graph made of ax, $\parr$ and $\otimes$ links
- Represents a Multiplicative Linear Logic (MLL) proof
- A *proof net* is a proof structure which represents a *correct* proof
  - i.e. coming from a sequent calculus proof
  - equivalently, inductive definition of proof nets



The correctness problem for proof structures

Problem (Correctness)

Given a proof structure, decide whether it is a proof net.

- Related to *correctness criteria*: non-inductive combinatorial characterizations of proof nets among proof structures
- This talk: investigate the **computational complexity** of this problem for linear logic **with Mix**, using tools from **graph theory**

$$\text{Mix rule:} \quad \frac{\vdash \Gamma \qquad \vdash \Delta}{\vdash \Gamma, \Delta}$$

Partial timeline of correctness criteria

- 1986: Birth of linear logic, "long trip" criterion
- 1989: Danos–Regnier criterion (everybody uses this one!)
  - Delete 1 of the 2 premises of each $\parr$-link; do you always get a *tree*?
  - If so, then you've got an MLL proof net
- 1990: "contractibility" from Danos's PhD gives a *polynomial time* algorithm for correctness
- 1999: Guerrini implements contractibility in *linear time*
  - complicated graph parsing algorithm, somewhat ad-hoc
- 2000: another linear time criterion by Murawski & Ong
  - using mainstream graph theory (dominator trees)
- 2007: MLL correctness is *NL-complete* (Mogbil & Naurois)
- Lots of omissions in this list
  - At first, complexity was not the main focus
  - The subject seems "explored to death" …

The situation with Mix

- Variant of the Danos–Regnier criterion
  - Delete 1 of the 2 premises of each $\parr$-link; do you always get a *forest*?
  - If so, then you've got an *MLL+Mix* proof net
- Danos's PhD contains a *polynomial time* criterion for MLL+Mix (not contractibility)

The situation with Mix

- Variant of the Danos–Regnier criterion
  - Delete 1 of the 2 premises of each $\parr$-link; do you always get a *forest*?
  - If so, then you've got an *MLL+Mix* proof net
- Danos's PhD contains a *polynomial time* criterion for MLL+Mix (not contractibility)
- **No linear-time algorithm**
- **No sub-polynomial algorithm**
- **No X-completeness result**

The situation with Mix

- Variant of the Danos–Regnier criterion
  - Delete 1 of the 2 premises of each $\parr$-link; do you always get a *forest*?
  - If so, then you've got an *MLL+Mix* proof net
- Danos's PhD contains a *polynomial time* criterion for MLL+Mix (not contractibility)
- **No linear-time algorithm**
- **No sub-polynomial algorithm**
- **No X-completeness result**
- Maybe it's straightforward to adapt the MLL case?

The situation with Mix

- Variant of the Danos–Regnier criterion
  - Delete 1 of the 2 premises of each $\parr$-link; do you always get a *forest*?
  - If so, then you've got an *MLL+Mix* proof net
- Danos's PhD contains a *polynomial time* criterion for MLL+Mix (not contractibility)
- **No linear-time algorithm**
- **No sub-polynomial algorithm**
- **No X-completeness result**
- Maybe it's straightforward to adapt the MLL case?
  NO. It's actually more subtle than expected at first sight.

The situation with Mix

- Variant of the Danos–Regnier criterion
  - Delete 1 of the 2 premises of each $\otimes$-link; do you always get a *forest*?
  - If so, then you've got an *MLL+Mix* proof net
- Danos's PhD contains a *polynomial time* criterion for MLL+Mix (not contractibility)
- **No linear-time algorithm**
- **No sub-polynomial algorithm**
- **No X-completeness result**
- Maybe it's straightforward to adapt the MLL case?
  NO. It's actually more subtle than expected at first sight.
- Actually, MLL+Mix case interesting because of close connections with mainstream graph theory
  - mainstream $\neq$ "homemade" objects such as *paired graphs*

About connections with graph theory

- Indeed, why don't we juste use *graph algorithms*?
  - Proof nets are graph-like structures
  - Correctness criteria are decision procedures
  - Would let us leverage the work of algorithmists
- Not much has been done in this direction by the LL community

About connections with graph theory

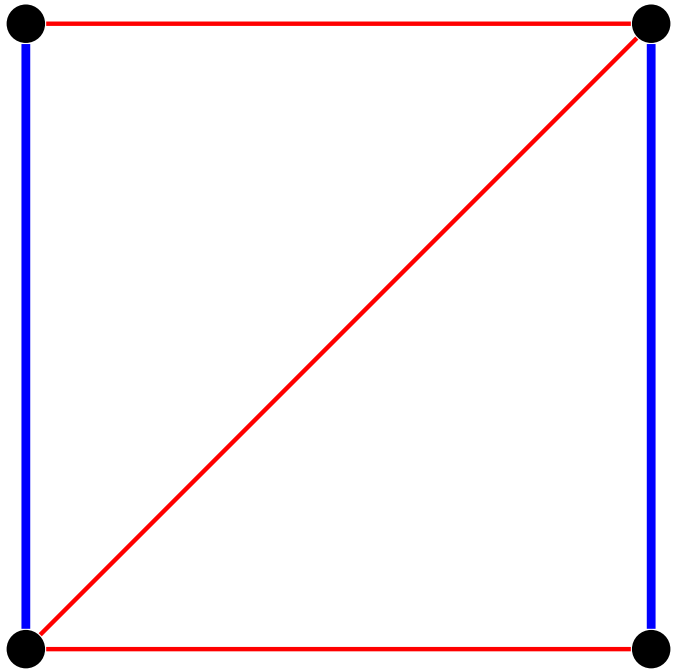
- Indeed, why don't we juste use *graph algorithms*?
  - Proof nets are graph-like structures
  - Correctness criteria are decision procedures
  - Would let us leverage the work of algorithmists
- Not much has been done in this direction by the LL community
- One exception: Christian Retoré's work
  - Seems to have been mostly ignored / forgotten until now
- 1993 PhD thesis: theory of "aggregates"
  - Also in unpublished report *Graph theory from linear logic: Aggregates*
  - Aggregates $\simeq$ edge-colored graphs / rainbow paths
  - We'll come back to this later

About connections with graph theory

- Indeed, why don't we juste use *graph algorithms*?
  - Proof nets are graph-like structures
  - Correctness criteria are decision procedures
  - Would let us leverage the work of algorithmists
- Not much has been done in this direction by the LL community
- One exception: Christian Retoré's work
  - Seems to have been mostly ignored / forgotten until now
- 1993 PhD thesis: theory of "aggregates"
  - Also in unpublished report *Graph theory from linear logic: Aggregates*
  - Aggregates $\simeq$ edge-colored graphs / rainbow paths
  - We'll come back to this later
- Later: *R&B-graphs* represent proof nets using *perfect matchings*
  - A classical topic in graph theory and combinatorial optimisation
  - Combine with algorithms for perfect matchings $\rightarrow$ profit!

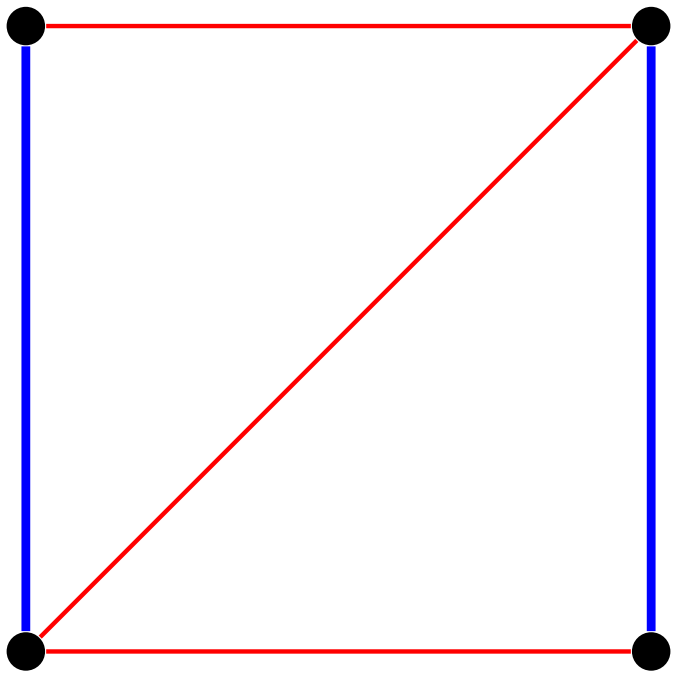
Perfect matchings: reminder (1)

- A *perfect matching* is a set of edges in an undirected graph such that each vertex is incident to exactly one edge in the matching
- Example below: blue edges form a perfect matching



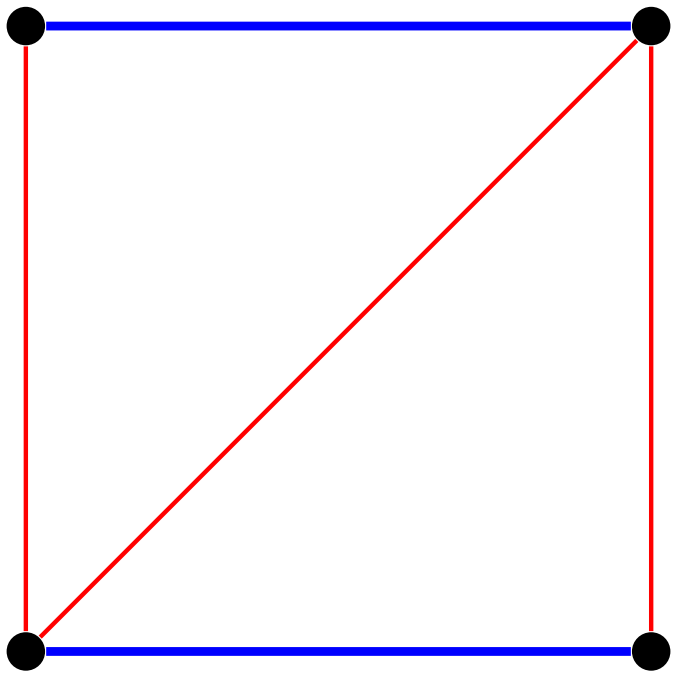
Perfect matchings: reminder (2)

- An *alternating path* is a path
  - without vertex repetitions
  - which alternates between edges inside and outside the matching
- Analogous notion of *alternating cycle*
- $\exists$ alternating cycle $\Leftrightarrow$ the perfect matching is not *unique*

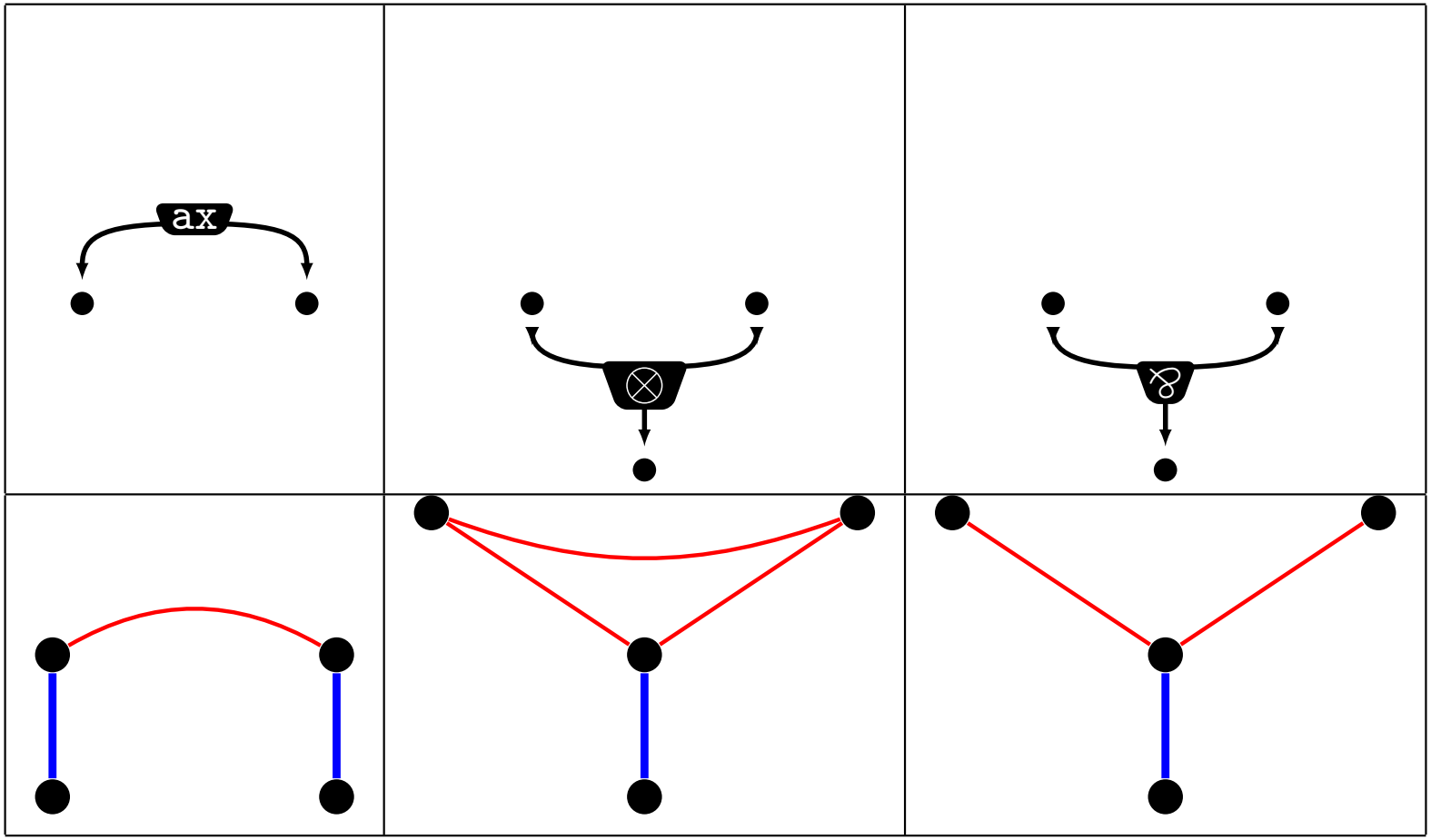


Perfect matchings: reminder (2)

- An *alternating path* is a path
  - without vertex repetitions
  - which alternates between edges inside and outside the matching
- Analogous notion of *alternating cycle*
- $\exists$ alternating cycle $\Leftrightarrow$ the perfect matching is not *unique*

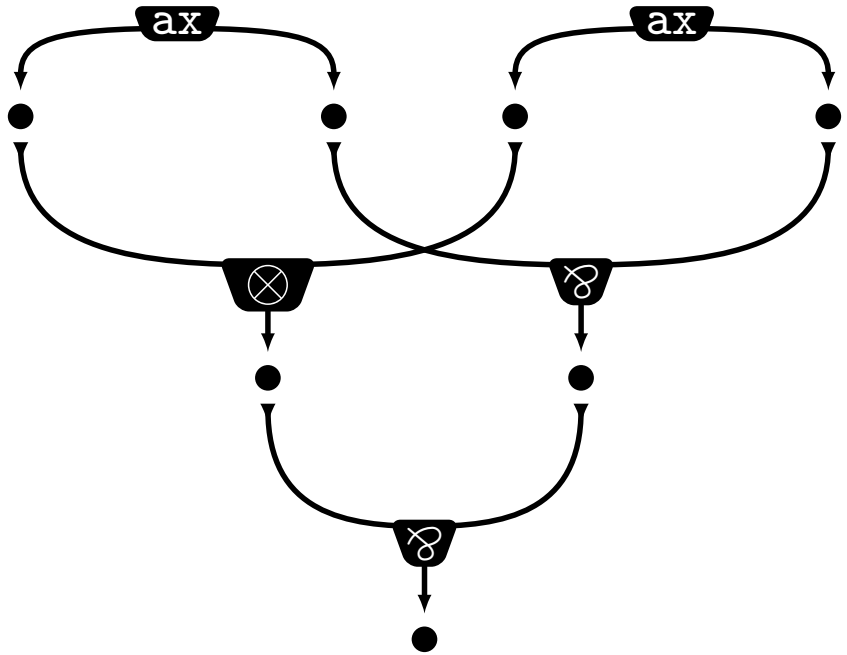


Retoré's *R&B-graphs*



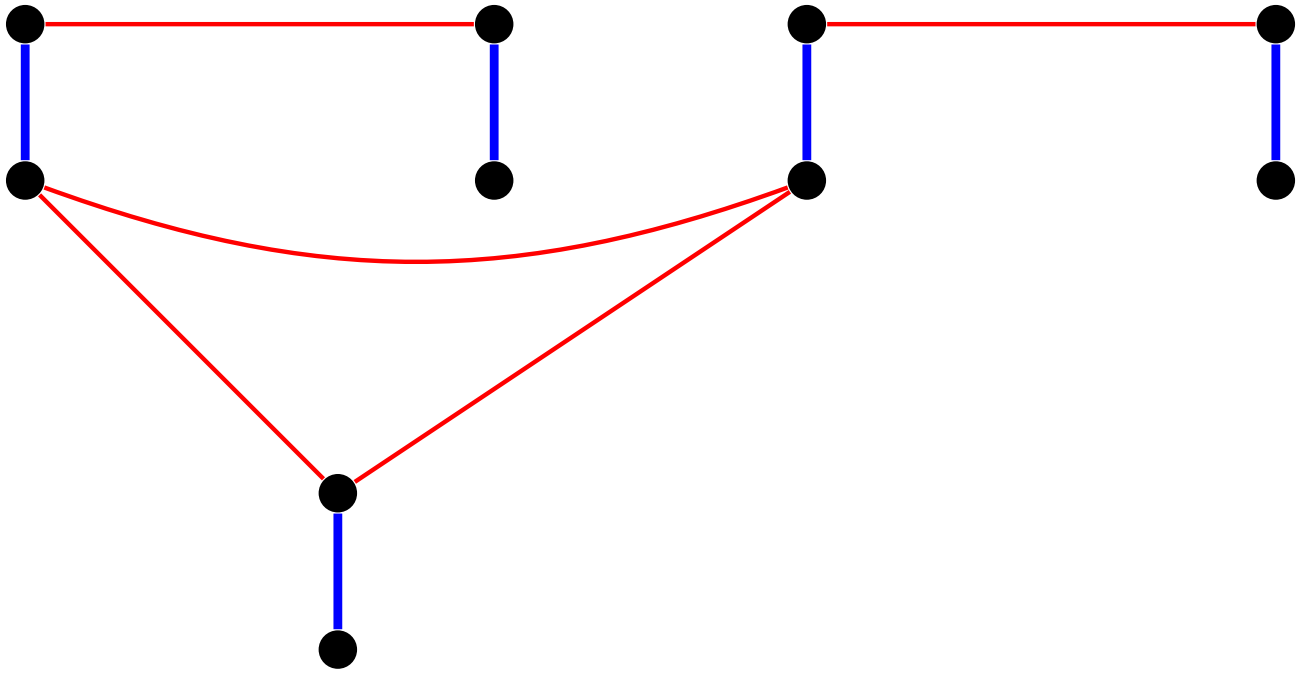
- Correctness criterion: matching is unique, i.e. no alternating cycle
- Corresponds to MLL+Mix correctness: no cycle crossing both premises of a $\parr$-link

R&B-graphs: example (1)

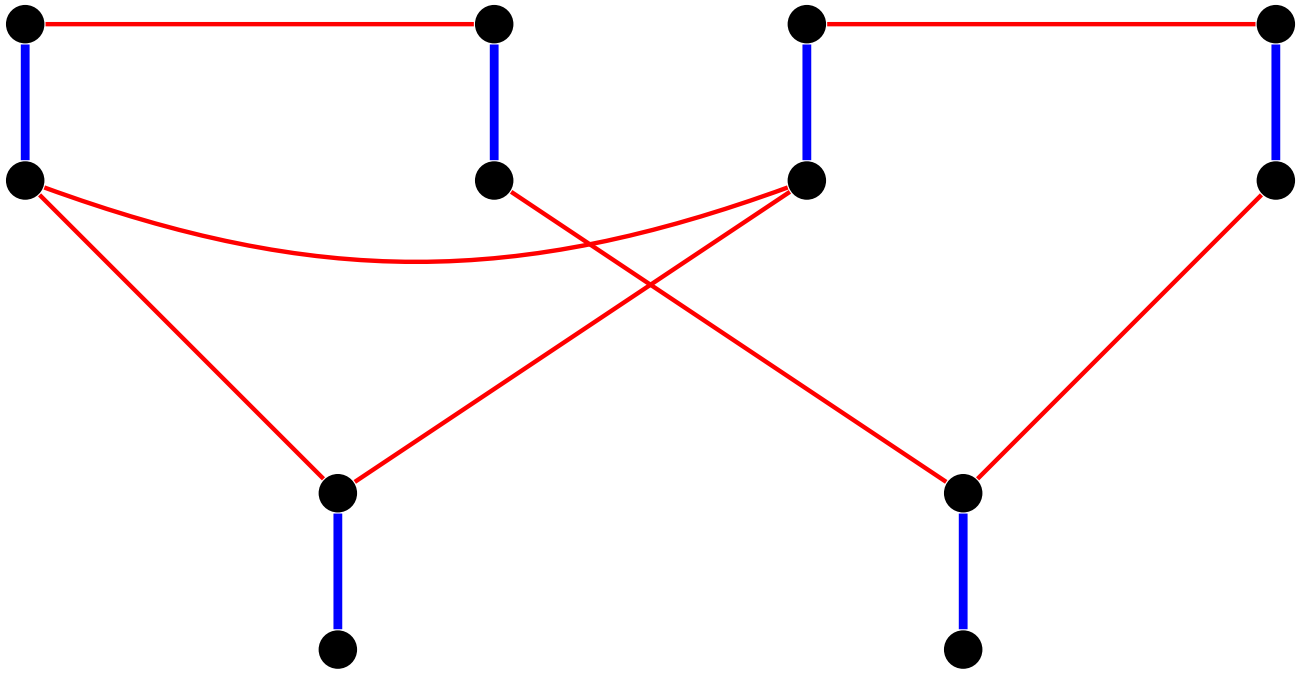


R&B-graphs: example (2)

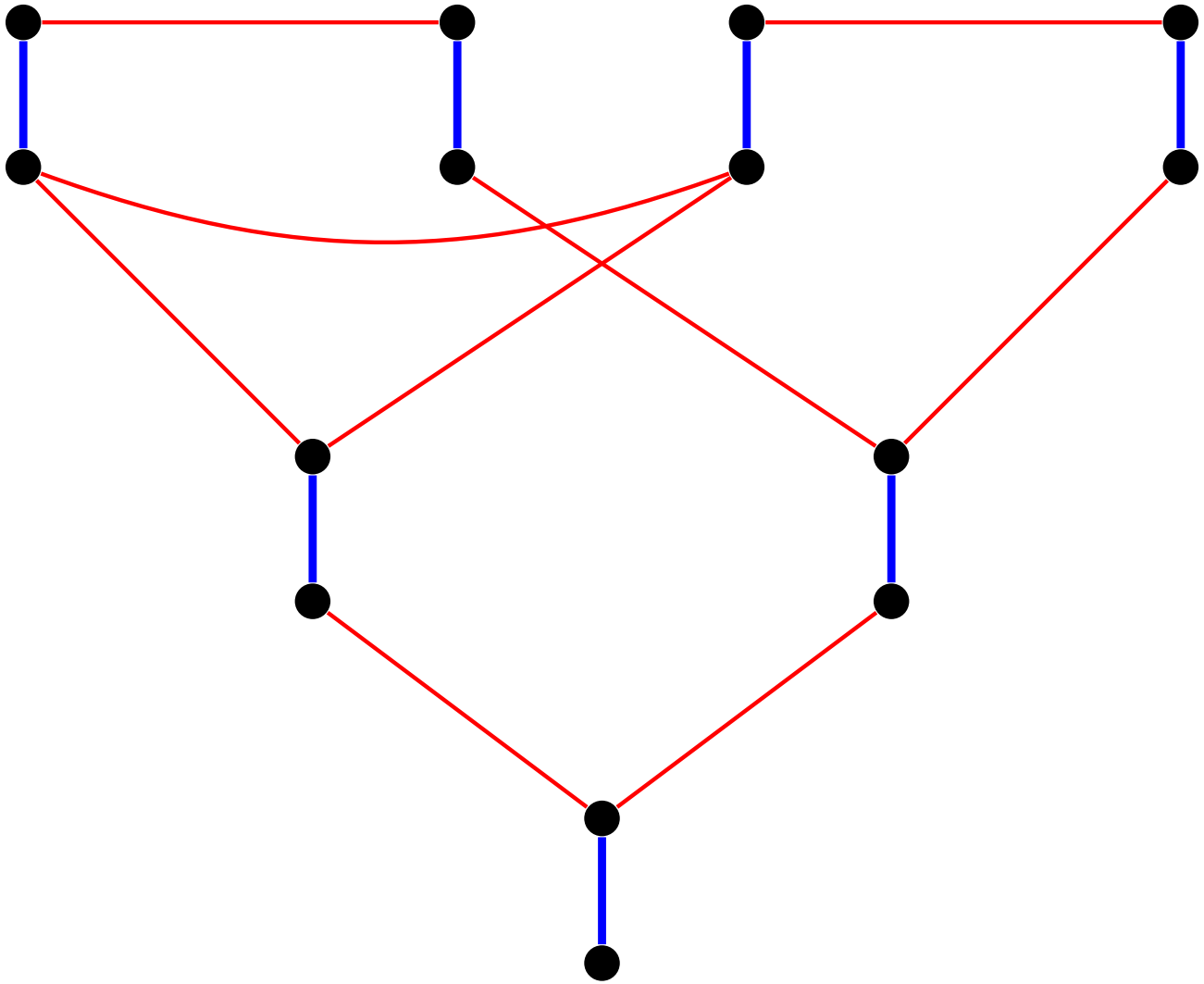
R&B-graphs: example (2)



R&B-graphs: example (2)



R&B-graphs: example (2)



An immediate consequence of R&B-graphs

- Alternating cycles for perfect matchings can be found in *linear time* (Gabow, Kaplan & Tarjan 1999)
- $\Rightarrow$ **Correctness for MLL+Mix can be decided in linear time**
  - First linear-time criterion for MLL+Mix
  - Also works for MLL without Mix (by Euler–Poincaré…), and simpler than other linear-time criteria: graph theory takes care of the difficult parts!

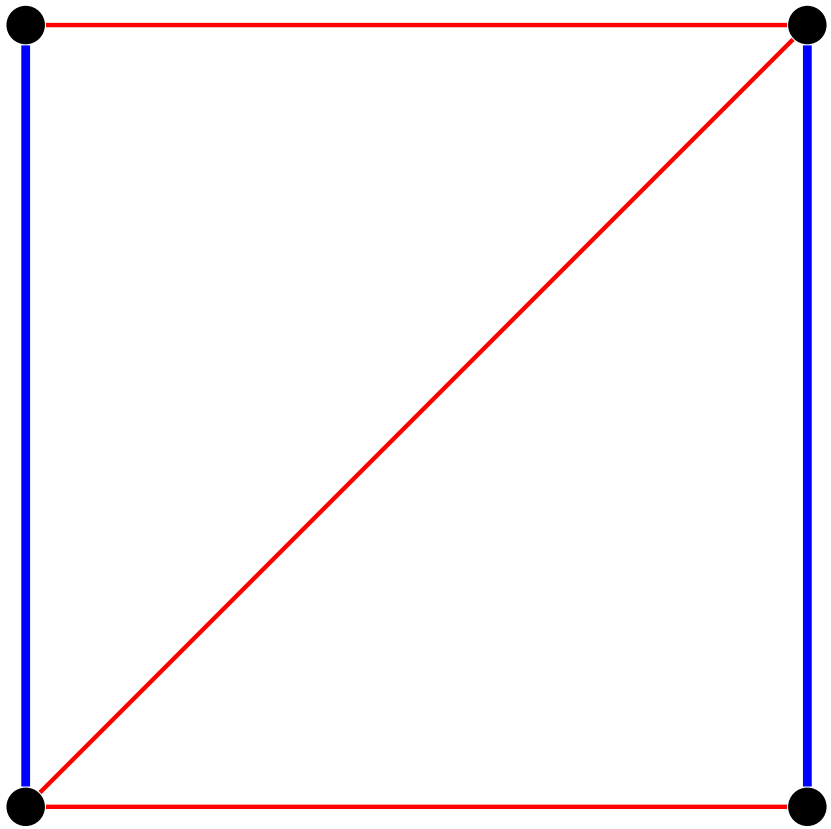
An immediate consequence of R&B-graphs

- Alternating cycles for perfect matchings can be found in *linear time* (Gabow, Kaplan & Tarjan 1999)
- $\Rightarrow$ **Correctness for MLL+Mix can be decided in linear time**
  - First linear-time criterion for MLL+Mix
  - Also works for MLL without Mix (by Euler–Poincaré…), and simpler than other linear-time criteria: graph theory takes care of the difficult parts!
- Timeline:
  - 1996: Linear Logic Tokyo Meeting, *Perfect matching and series-parallel graphs: multiplicative proof nets as R&B-graphs* (Retoré)
  - May 1999: STOC'99, *Unique maximum matching algorithms* (GKT)
  - July 1999: LICS'99, *Correctness of multiplicative proof nets is linear* (Guerrini)

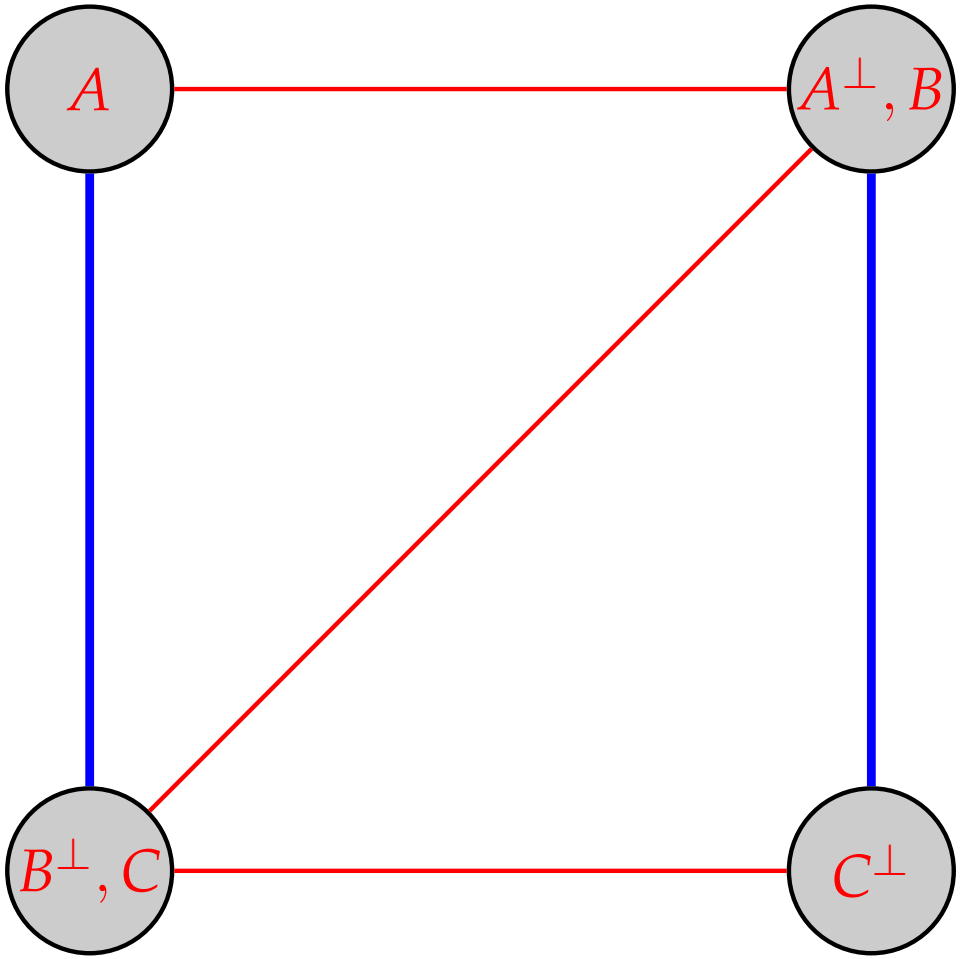
An immediate consequence of R&B-graphs

- Alternating cycles for perfect matchings can be found in *linear time* (Gabow, Kaplan & Tarjan 1999)
- $\Rightarrow$ **Correctness for MLL+Mix can be decided in linear time**
  - First linear-time criterion for MLL+Mix
  - Also works for MLL without Mix (by Euler–Poincaré…), and simpler than other linear-time criteria: graph theory takes care of the difficult parts!
- Timeline:
  - 1996: Linear Logic Tokyo Meeting, *Perfect matching and series-parallel graphs: multiplicative proof nets as R&B-graphs* (Retoré)
  - May 1999: STOC'99, *Unique maximum matching algorithms* (GKT)
  - July 1999: LICS'99, *Correctness of multiplicative proof nets is linear* (Guerrini)
- Also, an AC^0 reduction to the alternating cycle problem
  - What about the converse?

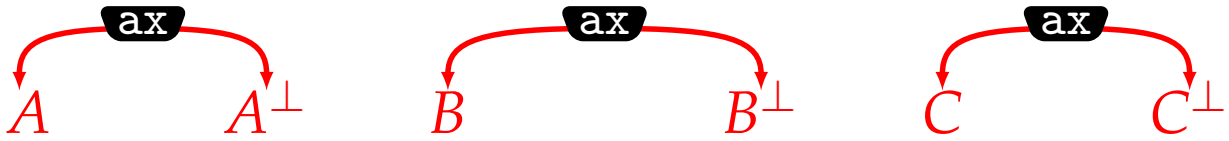
Alternating cycle $\rightarrow$ MLL+Mix correctness (1)



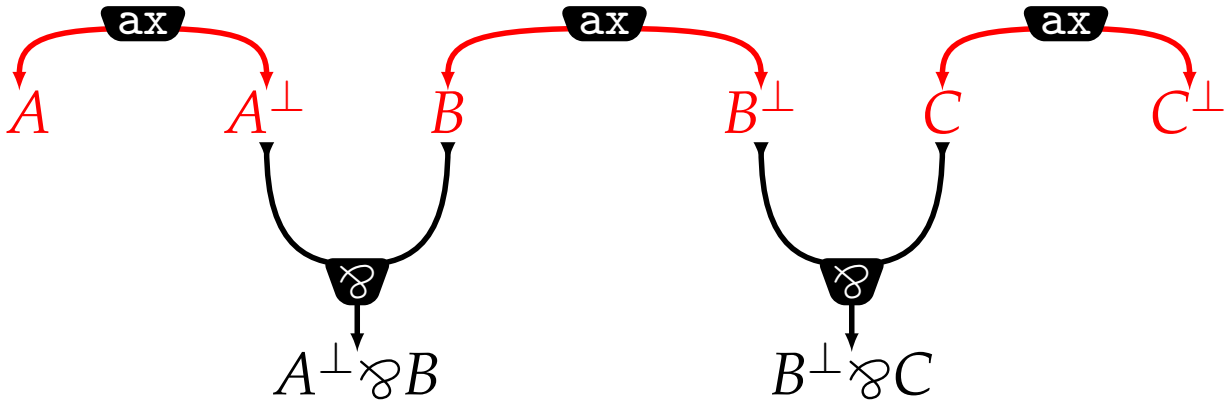
Alternating cycle $\to$ MLL+Mix correctness (1)



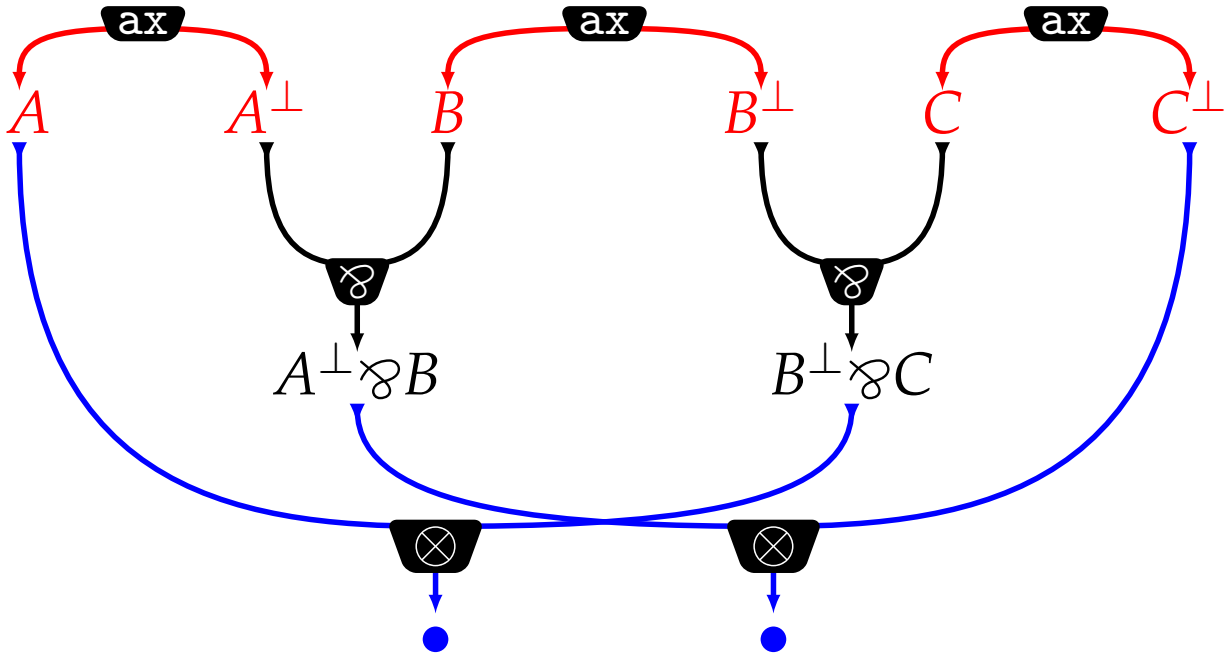
Alternating cycle → MLL+Mix correctness (2)



Alternating cycle → MLL+Mix correctness (2)



Alternating cycle → MLL+Mix correctness (2)



Perfect matchings and sub-polynomial complexity

- Finding an alternating cycle can be done in[1] *randomized* NC
  - sophisticated techniques (polynomial identity testing on determinants)
- *Deterministic* NC?
  - Would imply (non-trivially) that there exists a NC algorithm for deciding whether a graph has *exactly one* perfect matching
  - This is an *open problem* posed by Lovász in the 80's
- Recently: deterministic *quasi-NC* (Svensson & Tarnawski)
  - quasipolynomially many processors
  - very technical result, FOCS 2017 best paper award

[1] Reminder: NC is the class of problems computable in polylog(n) time with poly(n) processors, and NL $\subseteq$ NC.

On the complexity of MLL+Mix correctness

- Correctness for MLL+Mix is *equivalent* to the alternating cycle problem
- $\Rightarrow$ MLL+Mix correctness $\in$ NC would solve an open problem
  - If it were in NL, would be surprising
- Contrast with the *NL-completeness* of correctness for MLL
  - Explains why many criteria for MLL, e.g. contractibility, cannot be easily adapted to handle the Mix rule
- Still, **MLL+Mix correctness is in quasi-NC**

On the complexity of MLL+Mix correctness

- Correctness for MLL+Mix is *equivalent* to the alternating cycle problem
- $\Rightarrow$ MLL+Mix correctness $\in$ NC would solve an open problem
  - If it were in NL, would be surprising
- Contrast with the *NL-completeness* of correctness for MLL
  - Explains why many criteria for MLL, e.g. contractibility, cannot be easily adapted to handle the Mix rule
- Still, **MLL+Mix correctness is in quasi-NC**
- Conclusion for now: MLL+Mix correctness…
  - can be solved in **linear time**
  - is **probably harder** *(under* AC^0 *reductions)* than without Mix

On the complexity of MLL+Mix correctness

- Correctness for MLL+Mix is *equivalent* to the alternating cycle problem
- $\Rightarrow$ MLL+Mix correctness $\in$ NC would solve an open problem
  - If it were in NL, would be surprising
- Contrast with the *NL-completeness* of correctness for MLL
  - Explains why many criteria for MLL, e.g. contractibility, cannot be easily adapted to handle the Mix rule
- Still, **MLL+Mix correctness is in quasi-NC**
- Conclusion for now: MLL+Mix correctness…
  - can be solved in **linear time**
  - is **probably harder** *(under* AC^0 *reductions)* than without Mix
- Next:
  - Sequentialization
  - More graph theory stuff…
  - applied to the complexity of Pagani's "visible acyclicity"

A few words on sequentialization (1)

- The *sequentialization theorem*:
  correctness criterion $\Leftrightarrow$ inductive definition of proof nets
- A remark by Retoré: the "splitting link" lemmas in proofs of sequentialization are equivalent to the following

Theorem (Kotzig 1959)

Every unique perfect matching (i.e. w/o alt cycle) contains a bridge.

- Kotzig's theorem yields a "sequentialization theorem" for unique perfect matchings
- Not directly equivalent to proof net sequentialization via Retoré's R&B-graphs
  - Problem: R&B-graphs do not represent the premise $\rightarrow$ conclusion orientation of proof nets

A few words on sequentialization (2)

- To fix this, new translation proof structures $\to$ perfect matchings
- See my FSCD'18 paper *Unique perfect matchings and proof nets*
- Consequences:
  - A tighter bridge between linear logic and graph theory
  - Proof net sequentializations correspond to perfect matching sequentializations
  - Quasi-linear time *sequentialization algorithm* (still not as good as linear time for MLL without Mix)
  - Lot of stuff to say on the "kingdom ordering" of links
  - A new graph-theoretic result
- Not necessary when studying the complexity of correctness
  - Everything follows easily from Retoré's reduction

A few words on sequentialization (2)

- To fix this, new translation proof structures $\to$ perfect matchings
- See my FSCD'18 paper *Unique perfect matchings and proof nets*
- Consequences:
  - A tighter bridge between linear logic and graph theory
  - Proof net sequentializations correspond to perfect matching sequentializations
  - Quasi-linear time *sequentialization algorithm* (still not as good as linear time for MLL without Mix)
  - Lot of stuff to say on the "kingdom ordering" of links
  - A new graph-theoretic result
- Not necessary when studying the complexity of correctness
  - Everything follows easily from Retoré's reduction
  - But could we have guessed that such a reduction must exist?

A family of constraints on paths / cycles in graphs

- But could we have guessed that such a reduction must exist?
- *Yes*: the Danos–Regnier correctness criterion is a special case of a problem known to reduce to a matching problem
- Actually, MLL+Mix correctness belongs to a family of equivalent graph-theoretic problems
  - Find a path/cycle under some (tractable) constraints
  - With *edge-colored graphs*, the family resemblance with Danos–Regnier will be more obvious
- Let's revisit the usual *paired graphs* and see how they fit in mainstream graph theory

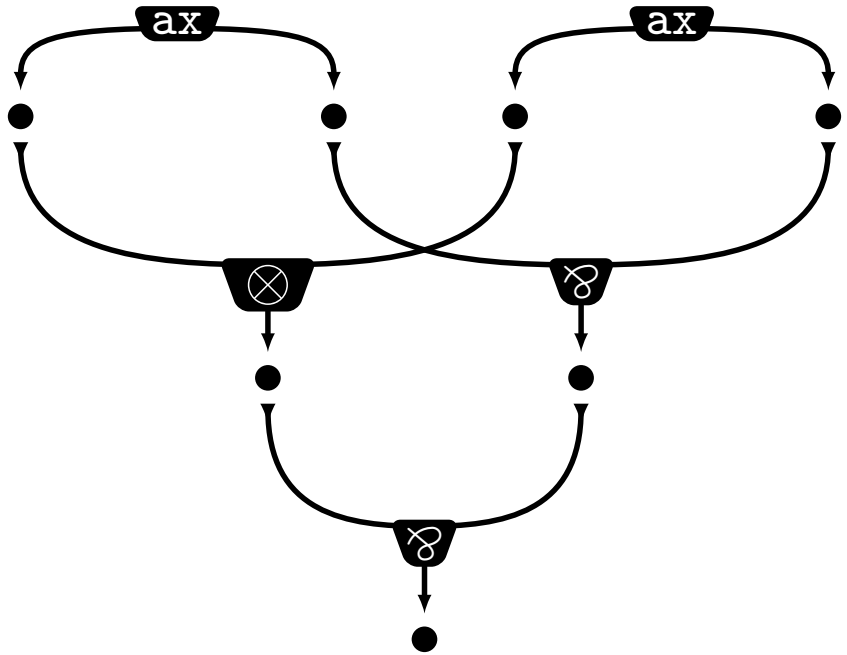
Paired graphs

Definition

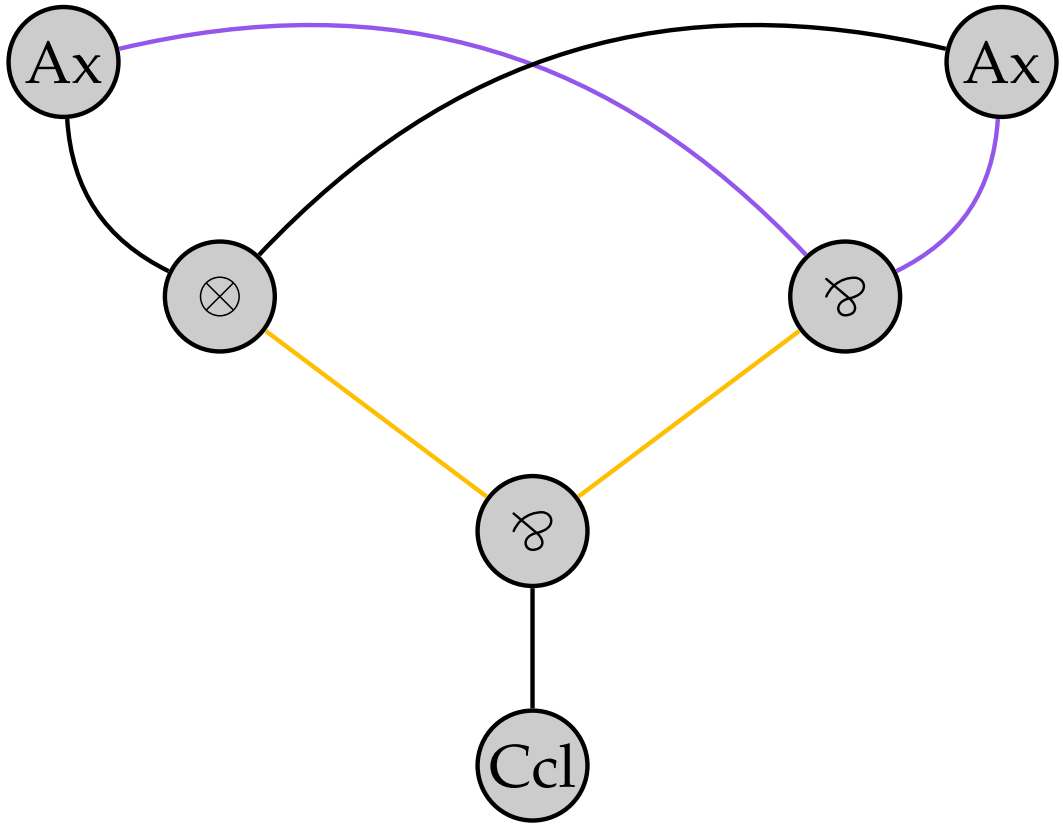
A *paired graph* is a graph equipped with a set of *disjoint* unordered pairs of *co-incident* edges.

- The usual graph-theoretic presentation of correctness criteria:
  - build a paired graph from a proof structure
  - Danos–Regnier: delete one edge in each pair; do you always get a tree (resp. a forest)?
  - contractibility: rewrite rules on paired graphs
- A remark: contractibility is actually an (efficient) algorithm to test D–R on general paired graphs
  - MLL case (tree): works even with non co-incident pairs!
  - MLL+Mix case (forest): does not work at all

Paired graph example

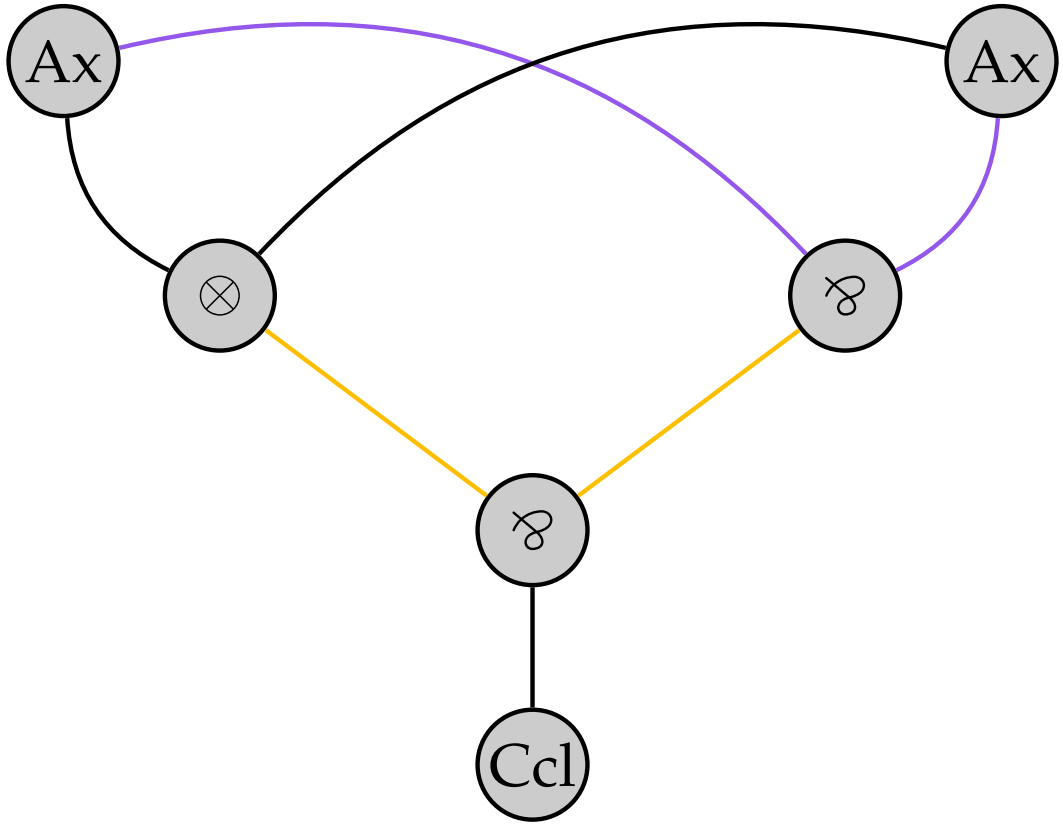


Paired graph example



- Black edges are unpaired
- Colors indicate pairs

Edge-colored graph example



- Now imagine the black edges actually have different unique colors
- This is an *edge-colored graph*

Rainbow paths in edge-colored graphs

- From now on we consider paired graphs as special cases of edge-colored graphs
- Benefit: lots of previous work on paths in such graphs

Definition

A *rainbow path* (resp. *cycle*) is a path (resp. cycle) in an edge-colored graph which crosses at most one edge of each color.

- Obviously, MLL+Mix correctness is rainbow acyclicity
- Bad news: finding rainbow paths in NP-complete
  - (Except for a special case encompassing Retoré's aggregates!)

Properly colored paths in edge-colored graphs

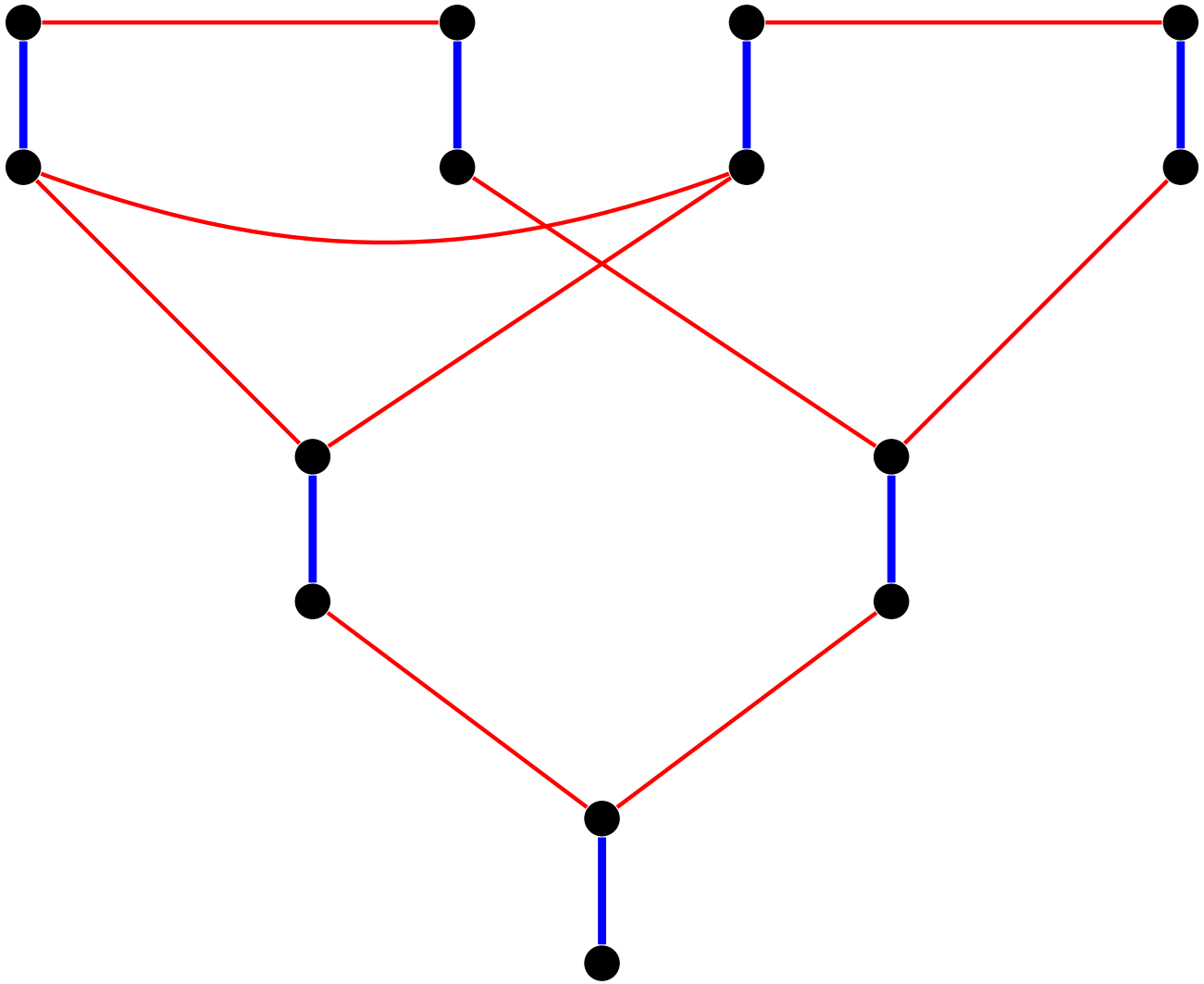
Definition

A *properly colored path* (resp. *cycle*) is a path (resp. cycle) in an edge-colored graph which never crosses *consecutively* two edges with the same color, and never visits the same vertex twice.

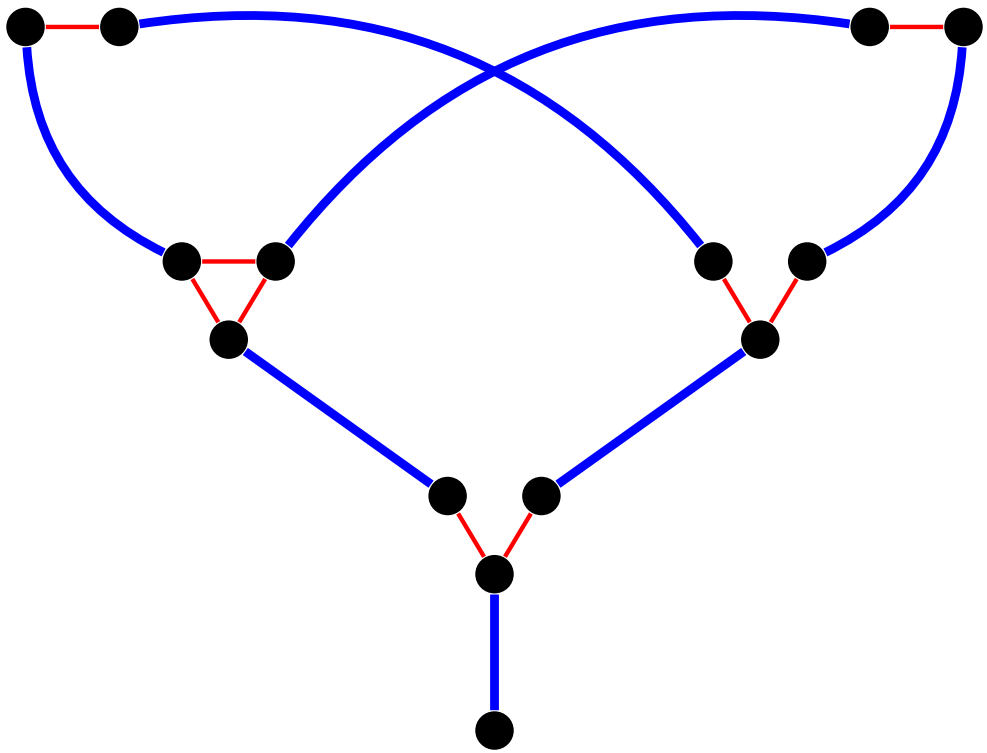
- **From now on, "paths" cannot visit a vertex twice**
- *In the case of paired graphs*, this is the same as rainbow paths
  - Because paired edges must share a vertex
- Good news: finding properly colored paths and cycles can be done in linear time
  - By reduction to alternating paths for perfect matchings!
- Again, MLL+Mix correctness can be tested in linear time

Analyzing Retoré's construction

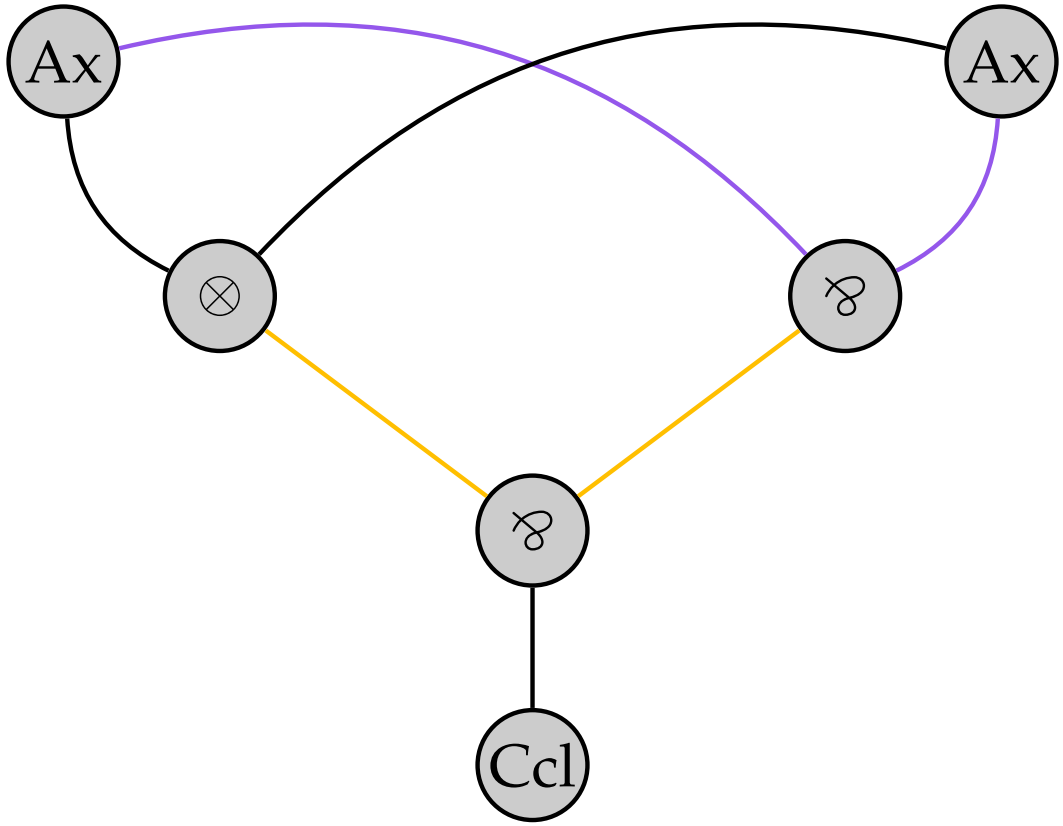
- Let's slightly deform this R&B-graph seen earlier…



Analyzing Retoré's construction



Analyzing Retoré's construction

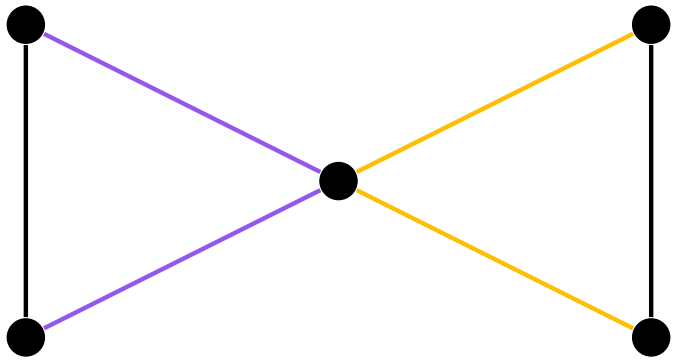


R&B-graphs for paired graphs

- Recipe: take the paired graph and
  - turn edges into matching edges
  - turn vertices into *cliques* outside the matching
  - delete non-matching edges corresponding to pairs
- Alternating paths correspond to *trails* not crossing paired edges consecutively
  - A trail can visit the same vertex multiple times, but crosses each *edge* at most once
- So Retoré's reduction actually tests for the existence of properly colored *closed trails*
  - Works on any *graph with forbidden transitions*, generalizing edge-colored graphs, cf. my TLLA'17 talk
  - This has new purely graph-theoretic consequences

Paths vs trails in paired graphs

- A paired graph without PC cycles can have a PC closed trail



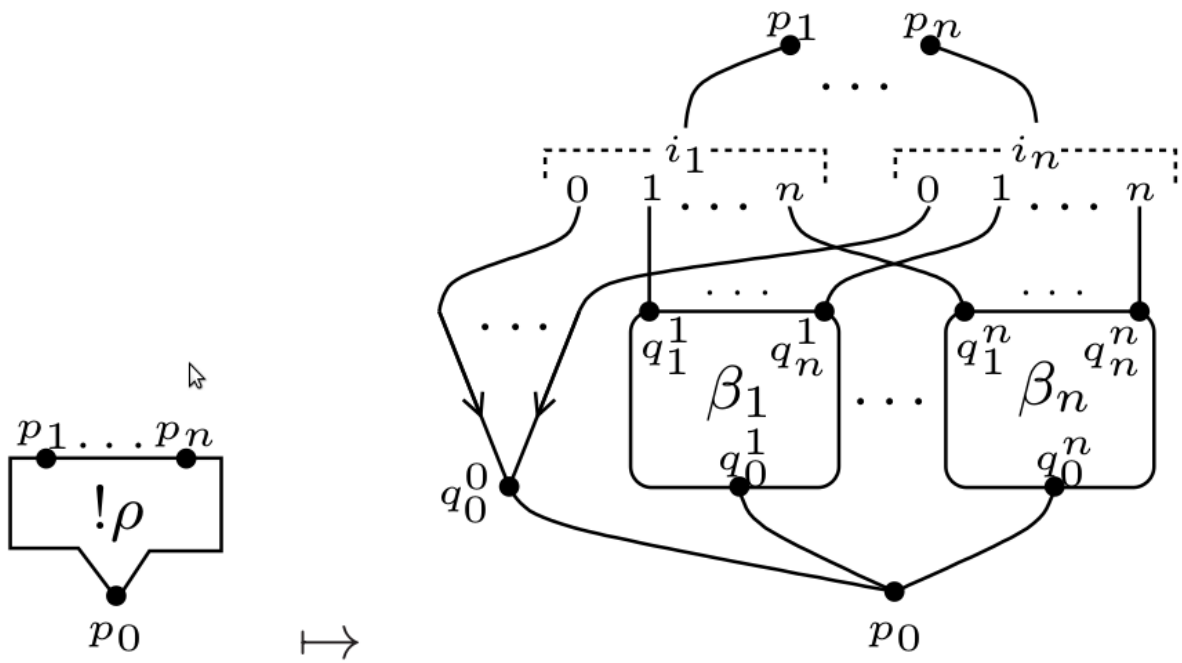
- But for paired graphs *coming from proof structures*, this doesn't happen
- Moral of the story: proof structures collapse different notions of constrained paths/cycles
  - Earlier, rainbow vs properly colored paths

Easy extensions beyond MLL+Mix

- With edge-colored graphs, it's clear how to represent:
  - Jumps
  - n-ary axiom links
- Jumps $\rightarrow$ correctness for quantifiers
  - Also, partially sequentialized proofs à la Di Giamberardino – Faggian
- n-ary axioms $\rightarrow$ exponential boxes $\rightarrow$ correctness for MELL+Mix
- Last but not least: what can we say about a less restrictive "correctness criterion" for MELL proof structures?

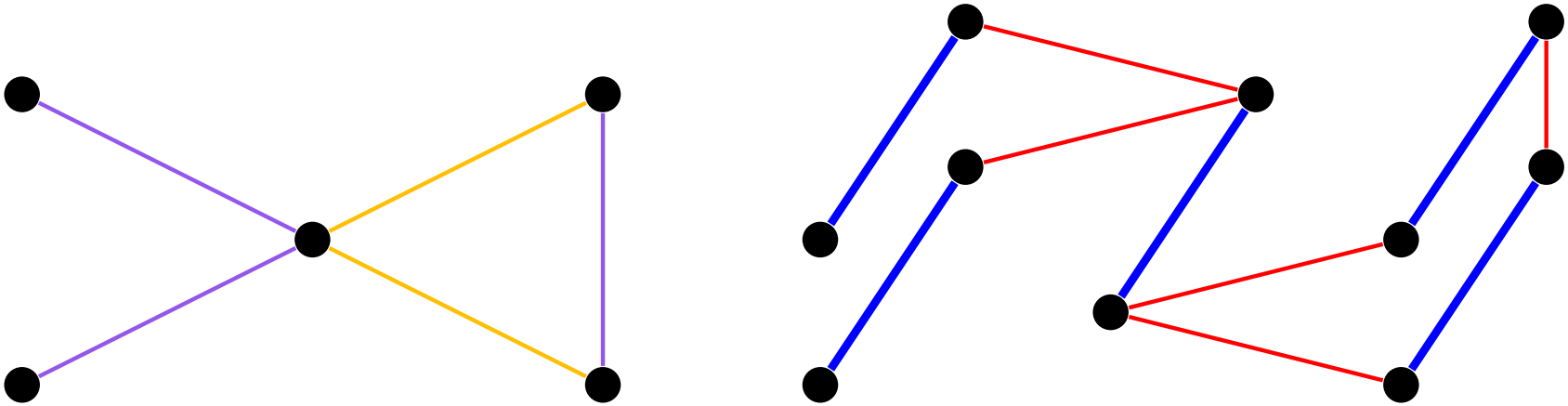
Visible acyclicity (1)

- Pagani's *visible acyclicity*:
  - No sequentialization theorem, but still guarantees strong normalization
  - Motivated by semantics
- The definition of visible cycles involves *directed* arcs



Visible acyclicity (2)

- 2-arc-colored digraphs can be reduced to the "visible digraphs" of MELL proof structures
  - inspired by 2-edge-colored graphs $\to$ perfect matchings $\to$ MLL



- Finding a properly colored cycle in a 2-arc-colored digraph is NP-complete
- So testing visible acyclicity is coNP-hard
  - Is it in coNP? I don't know

Some questions

- We seem to have found the right graph-theoretic counterpart for the statics of MLL+Mix proof nets; *what about MLL*?
- Mysteriously, all known linear-time correctness criteria, with or without Mix, rely on the same *incremental tree set union*[2] data structure… why?
- Combine edge-colored graphs + information on sequentialization?

[2] A special case of Union-Find.

Some questions

- We seem to have found the right graph-theoretic counterpart for the statics of MLL+Mix proof nets; *what about MLL*?
- Mysteriously, all known linear-time correctness criteria, with or without Mix, rely on the same *incremental tree set union*[2] data structure… why?
- Combine edge-colored graphs + information on sequentialization?

Thank you for your attention!

[2] A special case of Union-Find.