Proof nets through the lens of graph theory

Nguyễn Lê Thành Dũng (a.k.a. Tito) — nltd@nguyentito.eu
LIPN, Université Paris 13

Seminar of the "Gruppo di Logica e Geometria della Cognizione"
Università Roma Tre, May 17th, 2019

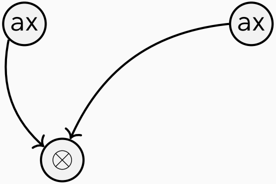
MLL proof nets

We work in Multiplicative Linear Logic (MLL)

A *proof net* is a sort of graph made of ax, $⅋$ and $\otimes$ links which represents a proof

- i.e. translated from a sequent calculus proof
- Equivalently, set of proof nets inductively generated

$$\dfrac{\dfrac{}{\vdash A, A^\perp}\,\mathsf{ax} \qquad \dfrac{}{\vdash B, B^\perp}\,\mathsf{ax}}{\vdash A \otimes B, A^\perp, B^\perp}\,\otimes$$



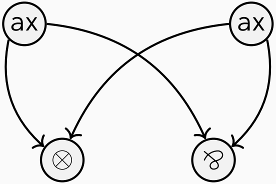
MLL proof nets

We work in Multiplicative Linear Logic (MLL)

A *proof net* is a sort of graph made of ax, $⅋$ and $\otimes$ links which represents a proof

- i.e. translated from a sequent calculus proof
- Equivalently, set of proof nets inductively generated

$$\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\,\mathsf{ax} \qquad \dfrac{}{\vdash B, B^{\perp}}\,\mathsf{ax}}{\vdash A \otimes B, A^{\perp}, B^{\perp}}\,\otimes}{\vdash A \otimes B, A^{\perp} ⅋ B^{\perp}}\,⅋$$



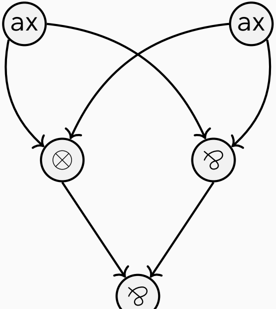
MLL proof nets

We work in Multiplicative Linear Logic (MLL)

A *proof net* is a sort of graph made of ax, $\parr$ and $\otimes$ links which represents a proof

- i.e. translated from a sequent calculus proof
- Equivalently, set of proof nets inductively generated

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\,\mathsf{ax} \qquad \dfrac{}{\vdash B, B^{\perp}}\,\mathsf{ax}}{\vdash A \otimes B, A^{\perp}, B^{\perp}}\,\otimes}{\vdash A \otimes B, A^{\perp} \parr B^{\perp}}\,\parr}{\vdash (A \otimes B) \parr (A^{\perp} \parr B^{\perp})}\,\parr$$



Proof nets vs proof structures

Proof structures: graphs made of ax-links, $\otimes$-links and $\parr$-links

- Proof structures $\supsetneq$ proof nets!
  Some are not images of any sequent calculus proof

Problem (Correctness)

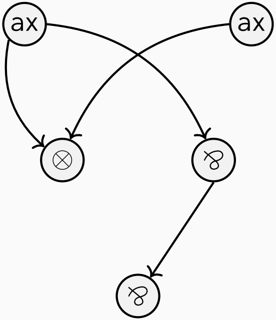
Given a proof structure, decide whether it is a proof net.

Related to *correctness criteria*: non-inductive combinatorial characterizations of proof nets among proof structures

A correctness criterion for MLL

Most common criterion: Danos–Regnier

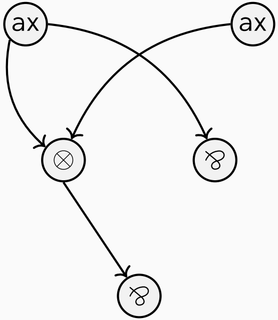
- Delete 1 of the 2 premises of each ⅋-link; do you always get an (undirected) *tree*?
- If so, then you've got an MLL proof net



A correctness criterion for MLL

Most common criterion: Danos–Regnier

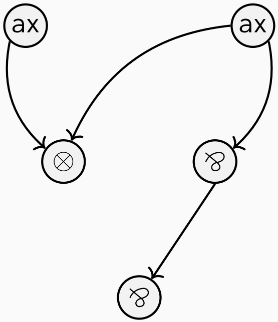
- Delete 1 of the 2 premises of each $\parr$-link; do you always get an (undirected) *tree*?
- If so, then you've got an MLL proof net



A correctness criterion for MLL

Most common criterion: Danos–Regnier

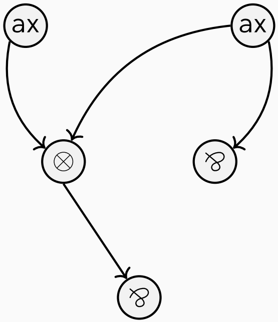
- Delete 1 of the 2 premises of each ⅋-link; do you always get an (undirected) *tree*?
- If so, then you've got an MLL proof net



A correctness criterion for MLL

Most common criterion: Danos–Regnier

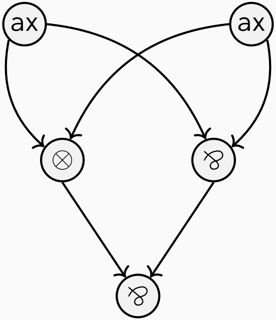
- Delete 1 of the 2 premises of each $\parr$-link; do you always get an (undirected) *tree*?
- If so, then you've got an MLL proof net



A correctness criterion for MLL

Most common criterion: Danos–Regnier

- Delete 1 of the 2 premises of each $\parr$-link; do you always get an (undirected) *tree*?
- If so, then you've got an MLL proof net



A correctness criterion for MLL+Mix

Most common criterion: Danos–Regnier

- Delete 1 of the 2 premises of each $\parr$-link; do you always get an (undirected) *tree* (resp. forest)?
- If so, then you've got an MLL (resp. MLL+Mix) proof net

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

Partial timeline of correctness criteria

- 1986: Birth of linear logic, “long trip” criterion
- 1989: Danos–Regnier criterion
- 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

- Danos–Regnier acyclicity
- Danos's PhD contains a *polynomial time* criterion for MLL+Mix (not contractibility)

The situation with Mix

- Danos–Regnier acyclicity
- 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

- Danos–Regnier acyclicity
- 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

- Danos–Regnier acyclicity
- 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

- Danos–Regnier acyclicity
- 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*

A graph-theoretic viewpoint

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

A graph-theoretic viewpoint

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

MLL+Mix correct = no cycle crossing both premises of a $\parr$-link

So this is a *constrained path-finding* problem

- Several such problems have been studied in graph theory
- Next: an example

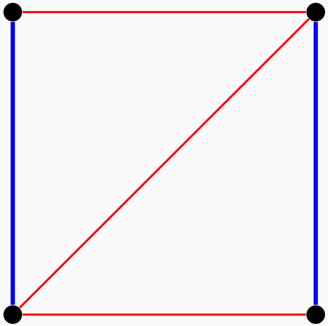
Perfect matchings (1)

Definition

A *perfect matching* is a set of edges in a graph such that each vertex is incident to exactly one edge in the matching.

A classical topic in combinatorics!

Example below: blue edges form a perfect matching

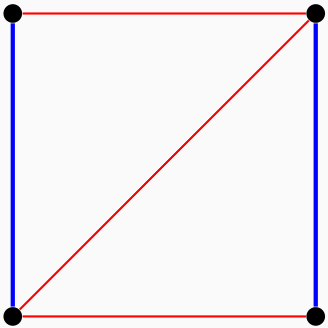


Perfect matchings (2)

An *alternating path* (resp. cycle) is a path (resp. cycle) which

- has no vertex repetitions
- alternates between edges inside and outside the matching

∃ alternating cycle ⇔ the perfect matching is not *unique*

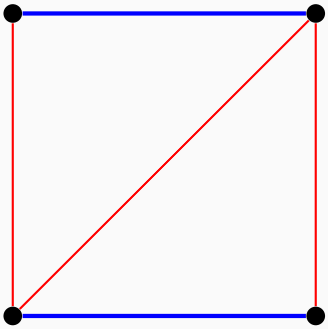


Perfect matchings (2)

An *alternating path* (resp. cycle) is a path (resp. cycle) which

- has no vertex repetitions
- alternates between edges inside and outside the matching

$\exists$ alternating cycle $\Leftrightarrow$ the perfect matching is not *unique*



Proof net correctness vs perfect matching uniqueness

- Alternating cycles in perfect matchings are equivalent to many[1] kinds of constrained cycles in graph theory
- Is it also the case for MLL+Mix correctness?

[1] See e.g. Szeider, On theorems equivalent with Kotzig's result on graphs with unique 1-factors, 2004

Proof net correctness vs perfect matching uniqueness

- Alternating cycles in perfect matchings are equivalent to many[1] kinds of constrained cycles in graph theory
- Is it also the case for MLL+Mix correctness? YES
- A connection was found by Christian Retoré in the 90's
- *R&B-graphs*: reduction {proof structures} $\rightarrow$ {graphs equipped with perfect matchings}

Theorem (Retoré's correctness criterion)

A proof structure is a MLL+Mix proof net iff the perfect matching of its R&B-graph is unique (i.e. has no alt. cycle).

[1] See e.g. Szeider, On theorems equivalent with Kotzig's result on graphs with unique 1-factors, 2004

An immediate application: complexity of correctness

Linear time algorithms for MLL correctness (Guerrini / Murawski & Ong) *cannot* be extended to MLL+Mix.

(Technical reason: take any subnet of a MLL net, contract it into a node, the net is still correct; not true with Mix.)

An immediate application: complexity of correctness

Linear time algorithms for MLL correctness (Guerrini / Murawski & Ong) *cannot* be extended to MLL+Mix.

(Technical reason: take any subnet of a MLL net, contract it into a node, the net is still correct; not true with Mix.)

Theorem (new!)

MLL+Mix correctness can be decided in linear time.

Proof.

Compute the R&B-graph, then test if it admits an alternating cycle. Both are in linear time. □

Also works for MLL (via "Euler–Poincaré" invariant)

Sophisticated linear time algorithm for finding an alt. cycle
$\longrightarrow$ Leverage the work of graph theorists as a black box!

Timeline

- 1996: LL Tokyo Meeting, *Perfect matchings and series-parallel*[2] *graphs: multiplicative proof nets as R&B-graphs* (Retoré)
- May 1999: STOC'99, *Unique maximum matching algorithms* (Gabow, Kaplan & Tarjan) $\longrightarrow$ alt. cycles in linear time
- July 1999: LICS'99, *Correctness of multiplicative proof nets is linear* (Guerrini)

[2] Refers to cographs, also related to combinatorial proofs; not discussed here.

Timeline

- 1996: LL Tokyo Meeting, *Perfect matchings and series-parallel*[2] *graphs: multiplicative proof nets as R&B-graphs* (Retoré)
- May 1999: STOC'99, *Unique maximum matching algorithms* (Gabow, Kaplan & Tarjan) $\longrightarrow$ alt. cycles in linear time
- July 1999: LICS'99, *Correctness of multiplicative proof nets is linear* (Guerrini)

OK, but Guerrini's algorithm also computes a *sequentialization* in linear time; can we do that for MLL+Mix?

[2]Refers to cographs, also related to combinatorial proofs; not discussed here.

Timeline

- 1996: LL Tokyo Meeting, *Perfect matchings and series-parallel*[2] *graphs: multiplicative proof nets as R&B-graphs* (Retoré)
- May 1999: STOC'99, *Unique maximum matching algorithms* (Gabow, Kaplan & Tarjan) $\longrightarrow$ alt. cycles in linear time
- July 1999: LICS'99, *Correctness of multiplicative proof nets is linear* (Guerrini)

OK, but Guerrini's algorithm also computes a *sequentialization* in linear time; can we do that for MLL+Mix?

Not quite, but we get close.
Before that, we need some preliminary work.

[2]Refers to cographs, also related to combinatorial proofs; not discussed here.

On sequentialization theorems

Sequentialization theorem: correct proof structures are proof nets, i.e. come from sequent calculus proofs

A remark by Retoré: analogously, unique perfect matchings admit an inductive characterization
(proof: use the theorem below)

Theorem (Kotzig 1959)

Every unique perfect matching (*i.e. without alternating cycle*) *contains a* bridge.

On sequentialization theorems

Sequentialization theorem: correct proof structures are proof nets, i.e. come from sequent calculus proofs

A remark by Retoré: analogously, unique perfect matchings admit an inductive characterization
(proof: use the theorem below)

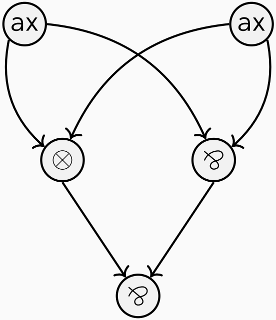
Theorem (Kotzig 1959)

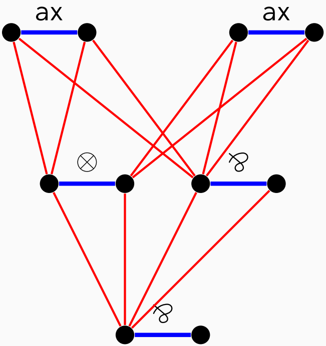
Every unique perfect matching (i.e. without alternating cycle) contains a bridge.

- A mismatch: {sequentializations of a proof net} $\not\cong$ {sequentializations of its "R&B-graph"}
- We fix this with another reduction {proof structures} $\to$ {graphs w/ PMs}: *graphification*

Graphification of proof structures (1)

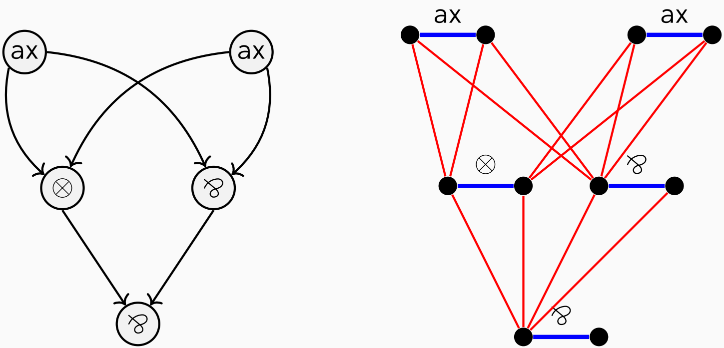
- Matching edges correspond to links
- *Bridges* correspond to *splitting terminal links*





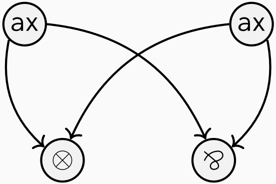
Graphification of proof structures (1)

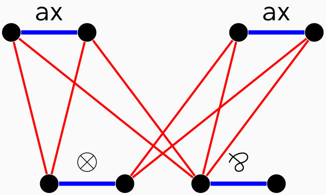
- Matching edges correspond to links
- *Bridges* correspond to *splitting terminal links*



Graphification of proof structures (1)

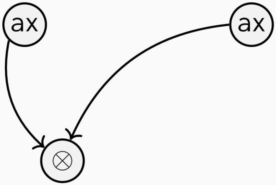
- Matching edges correspond to links
- *Bridges* correspond to *splitting terminal links*

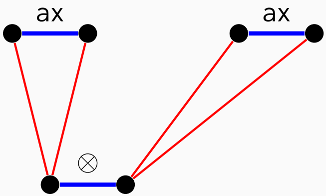




Graphification of proof structures (1)

- Matching edges correspond to links
- *Bridges* correspond to *splitting terminal links*



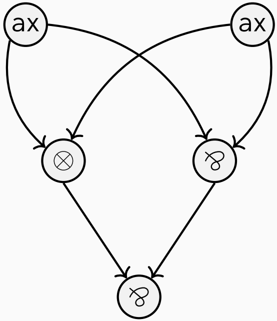


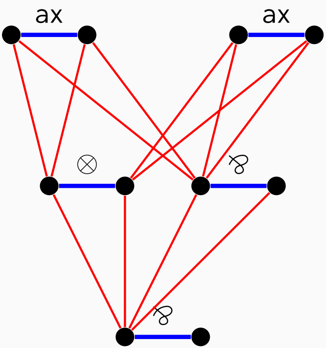
Graphification of proof structures (1)

- Matching edges correspond to links
- *Bridges* correspond to *splitting terminal links*

Graphification of proof structures (1)

- Matching edges correspond to links
- *Bridges* correspond to *splitting terminal links*





Correctness criterion is still uniqueness of PM i.e. no alt cycle

Graphifications of proof nets (2)

Theorem

The sequentializations of a proof structure are in bijection with the sequentializations of its graphification.

In particular if one set is $\neq \emptyset$ so is the other, therefore:

Corollary (Sequentialization theorem for MLL+Mix)

Danos–Regnier acyclic $\Leftrightarrow$ MLL+Mix sequentializable.

New proof, immediate from graph-theoretic analogue.

Graphifications of proof nets (2)

Theorem

The sequentializations of a proof structure are in bijection with the sequentializations of its graphification.

In particular if one set is $\neq \emptyset$ so is the other, therefore:

Corollary (Sequentialization theorem for MLL+Mix)

Danos–Regnier acyclic $\Leftrightarrow$ MLL+Mix sequentializable.

New proof, immediate from graph-theoretic analogue.

Next, let's *compute* a sequentialization.

Problem (Sequentialization)

Given a MLL+Mix proof net π, find a sequent proof which translates into π.

The naive sequentialization algorithm

1. Find a splitting link
2. Remove it
3. Recurse on remaining sub-proof net(s)

- Obvious implementation: quadratic time
    - Linear-time traversal to find a splitting link at each step

The naive sequentialization algorithm

1. Find a bridge in the graphification
2. Remove it
3. Recurse on remaining sub-proof net(s)

- Obvious implementation: quadratic time
  - Linear-time traversal to find a bridge at each step

The naive sequentialization algorithm

1. Find a bridge in the graphification
2. Remove it
3. Recurse on remaining sub-proof net(s)

- Obvious implementation: quadratic time
  - Linear-time traversal to find a bridge at each step
- Efficient implementation: how to find bridges quickly?
  - This has been studied by graph theorists

Quasi-linear sequentialization

Find bridges quickly using a dedicated data structure

- Fixed vertex set V, edge insertions/deletions, queries for bridges and connected components
- Latest improvement: SODA 2018 paper[3]
  $O((\log|V|)^2(\log\log|V|)^2)$ amortized complexity operations

[3] Holm, Rotenberg & Thorup, Dynamic bridge-finding in $\tilde{O}(\log^2 n)$ amortized time

Quasi-linear sequentialization

Find bridges quickly using a dedicated data structure

- Fixed vertex set V, edge insertions/deletions, queries for bridges and connected components
- Latest improvement: SODA 2018 paper[3]
  $O((\log|V|)^2(\log\log|V|)^2)$ amortized complexity operations

Theorem

MLL+Mix proof nets can be sequentialized in $O(n(\log n)^2(\log\log n)^2)$ time.

[3] Holm, Rotenberg & Thorup, Dynamic bridge-finding in $\tilde{O}(\log^2 n)$ amortized time

Mix makes things harder

Recap of new results on MLL+Mix (for now):

- correctness in linear time
- but sequentialization in *quasi*-linear time

Recall that MLL correctness is NL-complete.

So, is MLL+Mix in NL?

Mix makes things harder

Recap of new results on MLL+Mix (for now):

- correctness in linear time
- but sequentialization in *quasi*-linear time

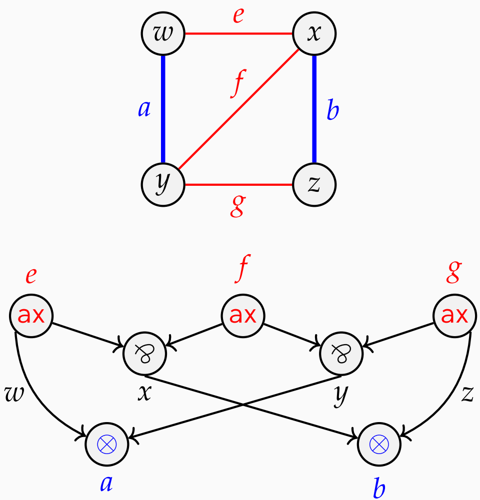
Recall that MLL correctness is NL-complete.

So, is MLL+Mix in NL?
This would solve an open problem in graph theory
(thus, either difficult or false)

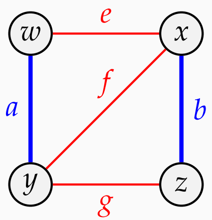
- Reduction from uniqueness of PMs (next slide)
- So the problems are actually *equivalent*

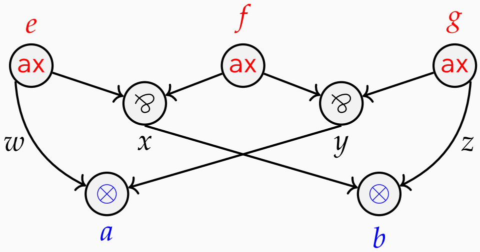
$\longrightarrow$ Both MLL+Mix correctness and sequentialization seem "harder" in some informal sense

Reduction perfect matchings → proof structures



Reduction perfect matchings → proof structures





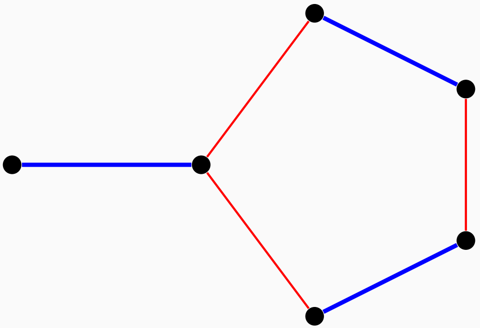
Next: a theorem on graphs inspired by linear logic

Blossoms in matching theory

- A key concept in combinatorial matching algorithms, e.g. testing PM uniqueness: *blossoms*[4]

Definition

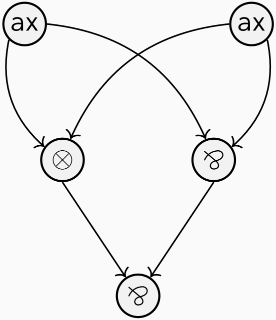
A *blossom* is a cycle with exactly 1 vertex matched outside.

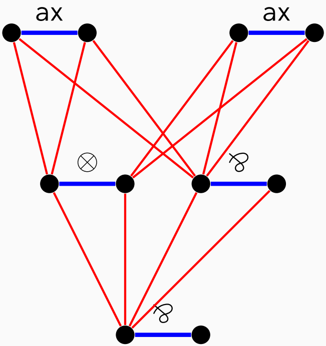


[4] Edmonds, *Paths, trees and flowers*, Canadian J. Math., 1965

Blossoms vs. dependencies

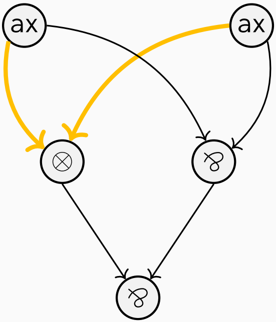
Blossoms of graphification $\rightsquigarrow$ subformulae and dependencies

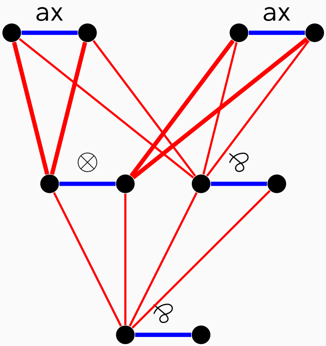




Blossoms vs. dependencies

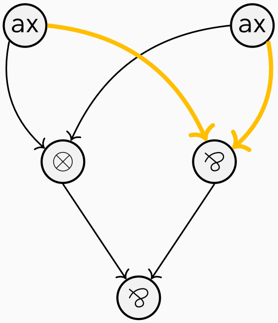
Blossoms of graphification $\rightsquigarrow$ **subformulae** and dependencies

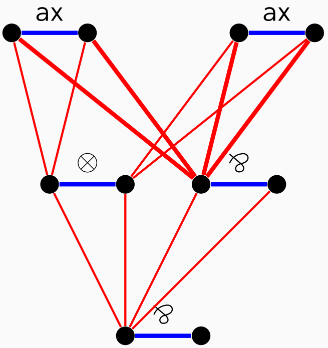




Blossoms vs. dependencies

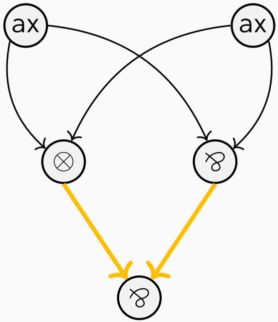
Blossoms of graphification $\rightsquigarrow$ **subformulae** and dependencies

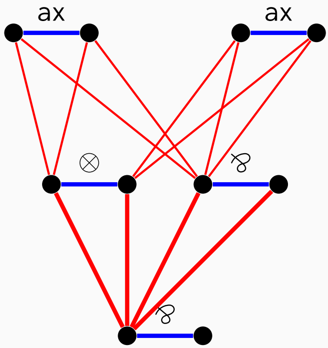




Blossoms vs. dependencies

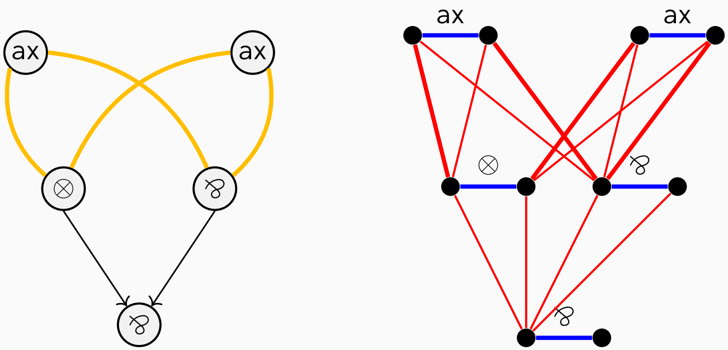
Blossoms of graphification ⇝ **subformulae** and dependencies





Blossoms vs. dependencies

Blossoms of graphification $\rightsquigarrow$ subformulae and **dependencies**

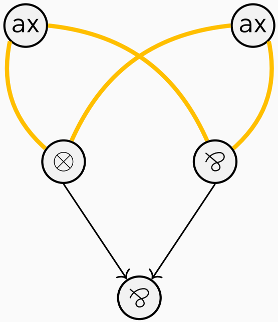


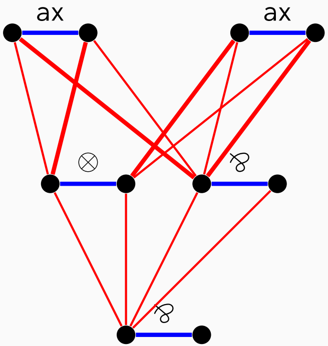
Definition

A ⅋-link l *depends upon* a link l' if there is a Danos–Regnier path between the premises of l going through l'.

Blossoms vs. dependencies

Blossoms of graphification $\rightsquigarrow$ subformulae and **dependencies**





Definition

A ⅋-link l *depends upon* a link l' if there is a Danos–Regnier path between the premises of l going through l'.

Kingdom ordering of proof nets and unique PMs

Definition (Kingdom ordering of a proof net)

Let l, l' be links of a MLL+Mix proof net π. We define $l \ll_\pi l$ iff every sequentialization of π introduces l above l'.

Theorem (Bellin 1997)

$\ll_\pi = ((\textit{subformula relation}) \cup (\textit{dependency relation}))^*$

Kingdom ordering of proof nets and unique PMs

Definition (Kingdom ordering of a proof net)

Let l, l' be links of a MLL+Mix proof net π. We define $l \ll_\pi l$ iff every sequentialization of π introduces l above l'.

Theorem (Bellin 1997)

$\ll_\pi = ((\textit{subformula relation}) \cup (\textit{dependency relation}))^*$

Kingdom ordering can be defined for unique PMs
(Natural concept, similar things studied in combinatorics

e.g. perfect elimination orderings of chordal graphs)

Theorem (Equivalent graph-theoretic version)

Kingdom ordering = "blossom reachability"

A non-artificial graph-theoretic result coming from LL;
simpler statement: transitive closure of only 1 relation!

Summary of first part

Unique perfect matchings: the right graph-theoretic counterpart for the statics of MLL+Mix proof nets
(Not a combinatorial bijection, but both algorithmic reductions and transfer of structural properties)

Consequences:

- Progress on central problems on MLL+Mix nets
  - Not mentioned: a *quasi-NC* correctness criterion
- New results in graph theory
  - Not just Bellin's theorem: also, connections with *edge-colored graphs* and *graphs with forbidden transitions*, see `https://arxiv.org/abs/1901.07028`

Next: graphs *embedded on surfaces* and *cyclic* linear logic

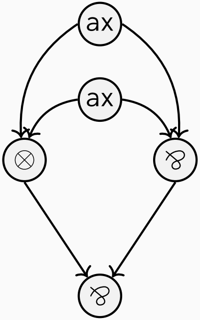
Cyclic MLL proof nets (1)

Sequent calculus for cyclic MLL: non-commutativity

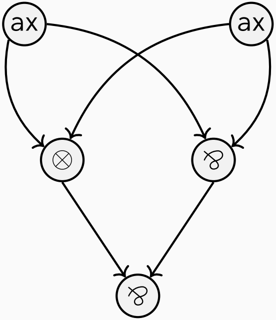
replace exchange $\dfrac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A\Delta}$ by $\dfrac{\vdash \Gamma, \Delta}{\vdash \Delta, \Gamma}$ cyclic exchange

Proof nets "drawn on the plane without crossings":

in cyMLL:



not in cyMLL:



Cyclic MLL proof nets (2)

Proof nets "drawn on the plane",
but the notion of *planar graph* is insufficient:
both graphs on the previous slide are the same…

From Nagayama & Okada 2003[5]:

> *Definition 3.2. A marked D–R graph drawing is said to be uniformly directed if the L-edge, R-edge and C-edge for a link is* drawn in a fixed cyclic order uniformly *for all tensor-links and par-links, or the links of degree 3.*

(emphasis mine)

[5] *A graph-theoretic characterization theorem for multiplicative fragment of non-commutative linear logic*

Combinatorial maps

So we must consider graphs endowed with a *rotation system*: for each vertex, a cyclic order on its incident edges. (This order is different for our two examples.)

Undirected graph + rotation system = *combinatorial map*.

[6] i.e. the faces are homeomorphic to disks.
[7] More precisely, compact oriented surfaces without boundary.

Combinatorial maps

So we must consider graphs endowed with a *rotation system*: for each vertex, a cyclic order on its incident edges. (This order is different for our two examples.)

Undirected graph + rotation system = *combinatorial map*.

Theorem (Heffter–Edmonds–Ringel principle)

(*Connected*) *combinatorial maps* $\cong$ *homeomorphism classes of* cellular[6] embeddings of graphs on surfaces[7].

Planar map = the surface is a sphere (compactified plane)
$\longrightarrow$ cyMLL proof nets must be planar maps

[6] i.e. the faces are homeomorphic to disks.
[7] More precisely, compact oriented surfaces without boundary.

A correctness criterion for cyMLL (1)

For *cut-free* proof structures,
MLL correctness + planarity = cyMLL correctness.

For proof nets with cuts, Melliès proposes a criterion based on "ribbons". My opinion: this is not the right point of view.

From embedded graph to ribbon:
take ε-neighborhood of graph drawing on the surface...
Conversely, one can recover the *faces* of a combinatorial map from its ribbon.

A correctness criterion for cyMLL (2)

Theorem (Melliès's criterion in mainstream language)

A proof structure with cuts is a cyMLL proof net iff

- *it is a MLL proof net and a planar map,*
- *all its conclusions are on the same face, and this face contains no upper corner of a $\parr$-link.*

A correctness criterion for cyMLL (2)

Theorem (Melliès's criterion in mainstream language)

A proof structure with cuts is a cyMLL proof net iff

- *it is a MLL proof net and a planar map,*
- *all its conclusions are on the same face, and this face contains no upper corner of a $\parr$-link.*

Corollary

Correctness for cyMLL with cuts is decidable in linear time.

Proof.

To decide planarity, compute the Euler characteristic of the surface. This takes linear time on the combinatorial map.

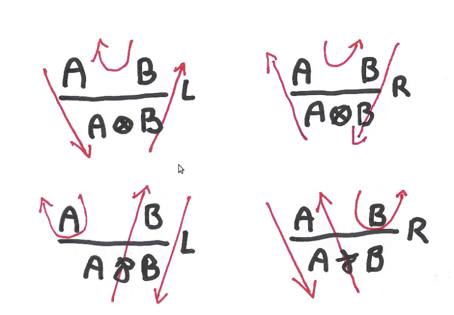
Traveling around all faces also takes linear time. □

Long trip switchings as embedded graphs (1)

Let's apply combinatorial maps to usual (commutative) MLL.

Girard's original *long trip* correctness criterion:

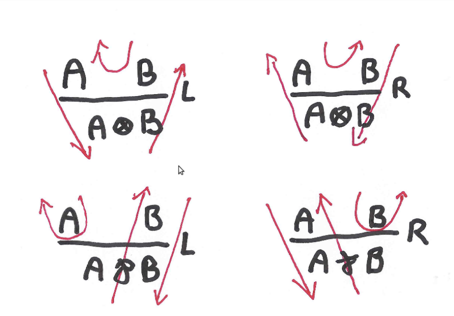
- On each $\otimes$-link and $\parr$-link, choose 1 of 2 possible set of routing instructions around the link
- Is the orbit a single cycle?



Long trip switchings as embedded graphs (1)

Let's apply combinatorial maps to usual (commutative) MLL.

Girard's original *long trip* correctness criterion:



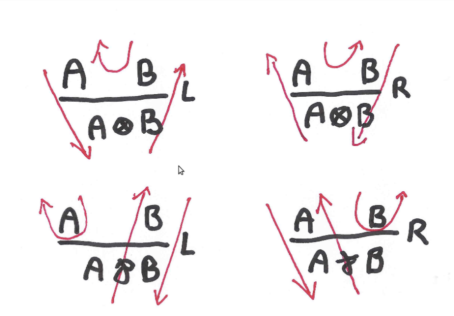
A long trip is a way to travel around a DR tree…

But how should we interpret the routing around a ⊗-link?

Long trip switchings as embedded graphs (1)

Let's apply combinatorial maps to usual (commutative) MLL.

Girard's original *long trip* correctness criterion:



A long trip is a way to travel around a DR tree...

But how should we interpret the routing around a ⊗-link?

It's a choice of *rotation system*! Trips are *faces*.

Long trip switchings as embedded graphs (2)

By the Heffter–Edmonds–Ringel principle, that the equivalence of the long trip and Danos–Regnier criteria is an instance of:

Theorem

A connected graph is a tree iff all its cellular embeddings on surfaces have a single face.

(Thanks to T. Seiller and É. Colin de Verdière.)

Proof.

- ($\Leftarrow$) is easy combinatorially
- ($\Rightarrow$) is obvious topologically

(For a purely combinatorial proof of ($\Rightarrow$), ask T. Seiller.) $\square$

Conclusion of second part

Moral of the story: finding the right widespread mathematical object makes a lot of things clearer!

Related:

- MLL with explicit exchange rule (Métayer 2001[8])
  - Rephrasing of main result:
    genus of surface ≤ number of exchange rules
  - Gaubert 2004[9] rediscovers a basic fact on embedded graphs
- Bijective combinatorics of (linear / planar / usual) λ-terms
  - e.g. N. Zeilberger's recent work
  - heavy use of combinatorial maps

---

[8] *Implicit exchange in multiplicative proofnets.*
[9] *Two-dimensional proof-structures and the exchange rule.*