A Formalization of Multiplicative Proof-Nets in Rocq

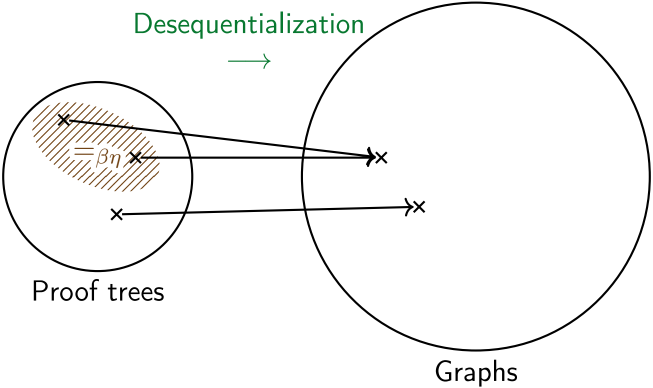
Rémi Di Guardia* Olivier Laurent†

*Paris, †Lyon

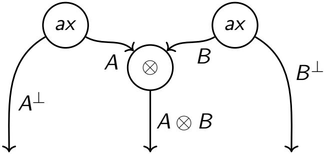
TLLA 2025, 19 July 2025

Introduction

Proof nets: graphical syntax for proofs of **Linear Logic**, *canonical*

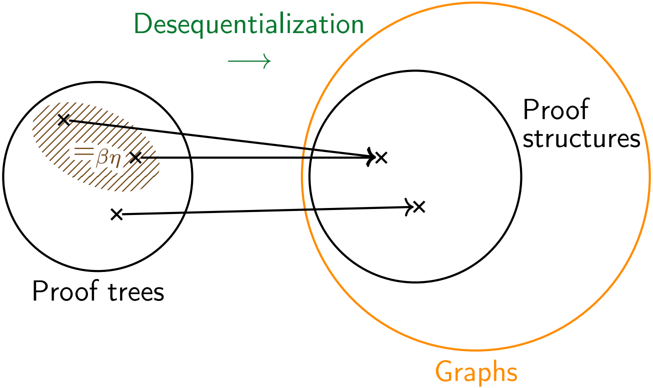


$$\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(ax) \qquad \dfrac{}{\vdash B^{\perp}, B}\,(ax)}{\vdash A^{\perp}, A \otimes B^{\perp}, B}\,(\otimes)$$

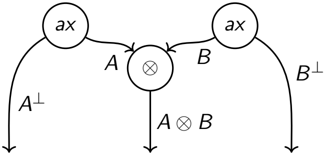


Introduction

<u>*Proof nets:*</u> graphical syntax for proofs of **Linear Logic**, *canonical*

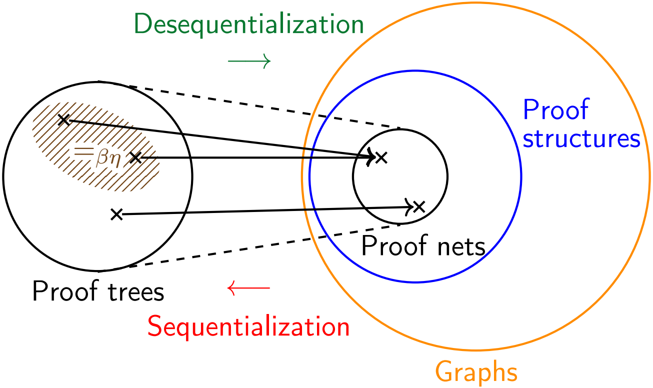


$$\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(ax) \qquad \dfrac{}{\vdash B^{\perp}, B}\,(ax)}{\vdash A^{\perp}, A \otimes B^{\perp}, B}\,(\otimes)$$

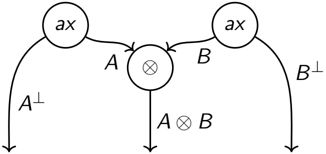


Introduction

<u>*Proof nets:*</u> graphical syntax for proofs of **Linear Logic**, *canonical*

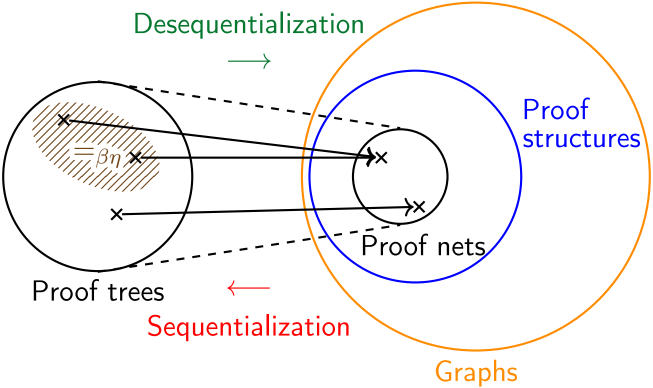


$$\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(ax) \qquad \dfrac{}{\vdash B^{\perp}, B}\,(ax)}{\vdash A^{\perp}, A \otimes B^{\perp}, B}\,(\otimes)$$

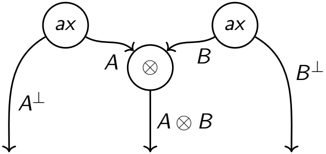


Introduction

Proof nets: graphical syntax for proofs of **Linear Logic**, *canonical*



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



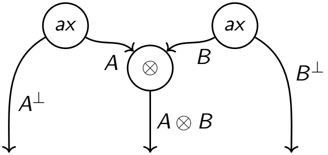
This talk: how to formalize proof nets in a proof assistant?

Around Formalizations of Linear Logic

Already many formalizations of linear logic in ...

- Rocq [Lau17; PW99; Xav+18; Bos+11; Péd; Sad03]
- Abella [CLR19; CLR17]
- Isabelle [KP95; Gro95]

... but always for *sequent calculus* and **never** for *proof nets*!



Around Formalizations of Linear Logic

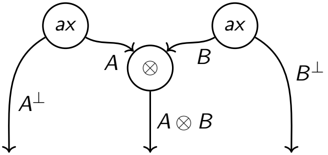
Already many formalizations of linear logic in ...

- Rocq [Lau17; PW99; Xav+18; Bos+11; Péd; Sad03]
- Abella [CLR19; CLR17]
- Isabelle [KP95; Gro95]

... but always for *sequent calculus* and **never** for *proof nets*!

What is the problem?

- Manipulations of **multi/hyper-graphs** and their isomorphisms, **non inductive** syntax
- **Complex** + **Several definitions** with strata – structures and nets



Around Formalizations of Linear Logic

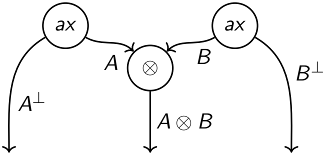
Already many formalizations of linear logic in ...

- Rocq [Lau17; PW99; Xav+18; Bos+11; Péd; Sad03]
- Abella [CLR19; CLR17]
- Isabelle [KP95; Gro95]

... but always for *sequent calculus* and **never** for *proof nets*!

What is the problem?

- Manipulations of **multi/hyper-graphs** and their isomorphisms, **non inductive** syntax
- **Complex** + **Several definitions** with strata – structures and nets
- **Geometric/Graphical/Implicit** arguments, with little drawings



Motivations & Goals

Why formalize proof nets now?

- **Because** of the liberties taken on paper!
- First step towards **complicated** proof nets (additive, first order, . . .)
- GraphTheory library in ROCQ (1st release in **2020**)
  - proves difficult results – treewidth, minors, . . .
  - **multigraphs** with needed operations: adding/removing vertices and edges, sub-graphs, isomorphisms, . . .

Motivations & Goals

Why formalize proof nets now?

- **Because** of the liberties taken on paper!
- First step towards **complicated** proof nets (additive, first order, . . .)
- GraphTheory library in ROCQ (1^{st} release in **2020**)
  - proves difficult results – treewidth, minors, . . .
  - **multigraphs** with needed operations: adding/removing vertices and edges, sub-graphs, isomorphisms, . . .

What did we formalize?

- **definition** of proof nets
- **desequentialization** from sequents to graphs + it yields a proof net
- **sequentialization**: proof-nets $\simeq$ images of desequentialization
- **cut-elimination** + gives proof net

Outline

- **Unit-free Multiplicative Linear Logic**
- **Formalization of Proof Nets**
  - Underlying Graphs
  - Correctness Criterion

Unit-free Multiplicative Linear Logic

Formulas

$$A ::= X \mid X^{\perp} \mid A \otimes A \mid A \parr A$$

Orthogonality

$$(X^{\perp})^{\perp} = X \qquad (A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp} \qquad (A \parr B)^{\perp} = A^{\perp} \otimes B^{\perp}$$

Sequents (lists)

$$\vdash A_1, A_2, \ldots, A_n$$

Rules

$$\frac{}{\vdash A^{\perp}, A}\ (ax) \qquad \frac{\vdash A^{\perp}, \Gamma \qquad \vdash A, \Delta}{\vdash \Gamma, \Delta}\ (cut) \qquad \frac{\vdash \Gamma}{\vdash \sigma(\Gamma)}\ (ex)$$

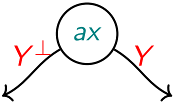
$$\frac{\vdash A, \Gamma \qquad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta}\ (\otimes) \qquad \frac{\vdash A, B, \Gamma}{\vdash A \parr B, \Gamma}\ (\parr)$$

Desequentialization

$$
\dfrac{\dfrac{}{\vdash X^{\perp}, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^{\perp}, Y}\,(ax) \qquad \dfrac{}{\vdash Z^{\perp}, Z}\,(ax)}{\vdash Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\dfrac{\dfrac{\vdash X^{\perp}, X \otimes Y^{\perp}, Y \otimes Z^{\perp}, Z}{\vdash X^{\perp}, X \otimes Y^{\perp}, (Y \otimes Z^{\perp}) \parr Z}\,(\parr)}{\vdash X^{\perp} \parr (X \otimes Y^{\perp}), (Y \otimes Z^{\perp}) \parr Z}\,(\parr)}\,(\otimes)
$$

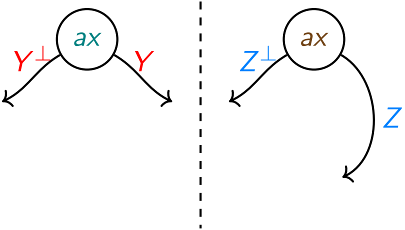
Desequentialization

$$\dfrac{\dfrac{}{\vdash X^\perp, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^\perp, Y}\,(ax) \qquad \dfrac{}{\vdash Z^\perp, Z}\,(ax)}{\vdash Y^\perp, Y \otimes Z^\perp, Z}\,(\otimes)}{\dfrac{\vdash X^\perp, X \otimes Y^\perp, Y \otimes Z^\perp, Z}{\dfrac{\vdash X^\perp, X \otimes Y^\perp, (Y \otimes Z^\perp) \parr Z}{\vdash X^\perp \parr (X \otimes Y^\perp), (Y \otimes Z^\perp) \parr Z}\,(\parr)}\,(\parr)}\,(\otimes)$$



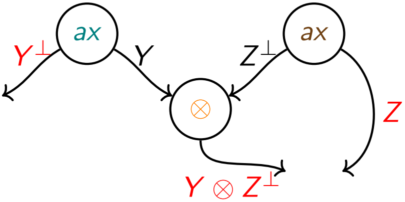
Desequentialization

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash X^{\perp}, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^{\perp}, Y}\,(ax) \quad \dfrac{}{\vdash Z^{\perp}, Z}\,(ax)}{\vdash Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, (Y \otimes Z^{\perp}) \parr Z}\,(\parr)}{\vdash X^{\perp} \parr (X \otimes Y^{\perp}), (Y \otimes Z^{\perp}) \parr Z}\,(\parr)$$



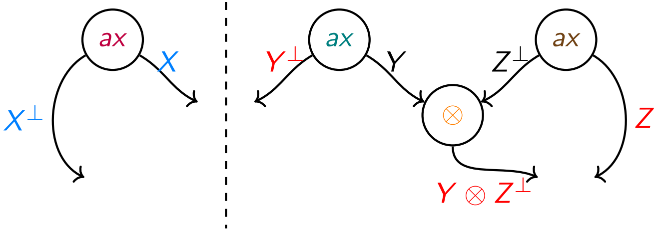
Desequentialization

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash X^{\perp}, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^{\perp}, Y}\,(ax) \qquad \dfrac{}{\vdash Z^{\perp}, Z}\,(ax)}{\vdash Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, (Y \otimes Z^{\perp}) \parr Z}\,(\parr)}{\vdash X^{\perp} \parr (X \otimes Y^{\perp}), (Y \otimes Z^{\perp}) \parr Z}\,(\parr)
$$

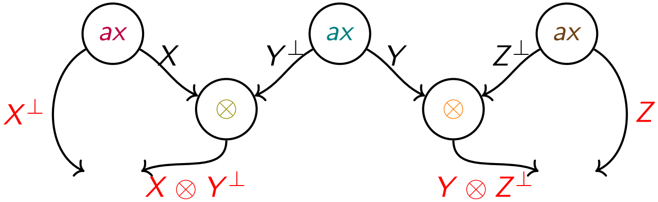


Desequentialization

$$
\dfrac{\dfrac{\dfrac{}{\vdash X^\perp, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^\perp, Y}\,(ax) \qquad \dfrac{}{\vdash Z^\perp, Z}\,(ax)}{\vdash Y^\perp, Y \otimes Z^\perp, Z}\,(\otimes)}{\dfrac{\vdash X^\perp, X \otimes Y^\perp, Y \otimes Z^\perp, Z}{\vdash X^\perp, X \otimes Y^\perp, (Y \otimes Z^\perp) \parr Z}\,(\parr)}\,(\otimes)}{\vdash X^\perp \parr (X \otimes Y^\perp), (Y \otimes Z^\perp) \parr Z}\,(\parr)
$$

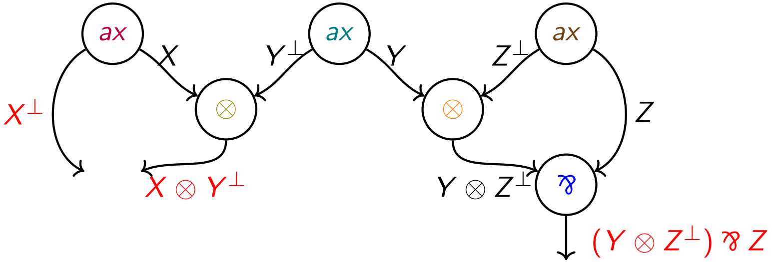


$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash X^{\perp}, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^{\perp}, Y}\,(ax) \qquad \dfrac{}{\vdash Z^{\perp}, Z}\,(ax)}{\vdash Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, (Y \otimes Z^{\perp}) \parr Z}\,(\parr)}{\vdash X^{\perp} \parr (X \otimes Y^{\perp}), (Y \otimes Z^{\perp}) \parr Z}\,(\parr)$$



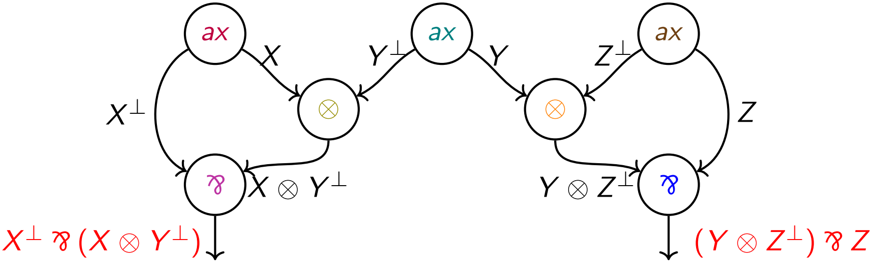
Desequentialization

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash X^{\perp}, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^{\perp}, Y}\,(ax) \qquad \dfrac{}{\vdash Z^{\perp}, Z}\,(ax)}{\vdash Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, Y \otimes Z^{\perp}, Z}\,(\otimes)}{\vdash X^{\perp}, X \otimes Y^{\perp}, (Y \otimes Z^{\perp}) \parr Z}\,(\parr)}{\vdash X^{\perp} \parr (X \otimes Y^{\perp}), (Y \otimes Z^{\perp}) \parr Z}\,(\parr)$$



Desequentialization

$$\dfrac{\dfrac{\dfrac{}{\vdash X^\perp, X}\,(ax) \qquad \dfrac{\dfrac{}{\vdash Y^\perp, Y}\,(ax) \qquad \dfrac{}{\vdash Z^\perp, Z}\,(ax)}{\vdash Y^\perp, Y \otimes Z^\perp, Z}\,(\otimes)}{\dfrac{\vdash X^\perp, X \otimes Y^\perp, Y \otimes Z^\perp, Z}{\vdash X^\perp, X \otimes Y^\perp, (Y \otimes Z^\perp) \parr Z}\,(\parr)}\,(\otimes)}{\vdash X^\perp \parr (X \otimes Y^\perp), (Y \otimes Z^\perp) \parr Z}\,(\parr)$$

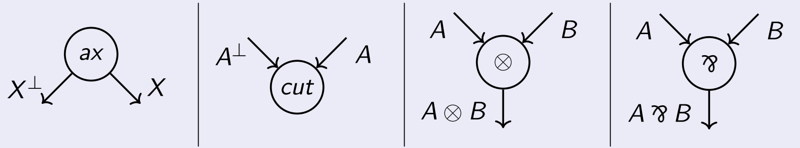


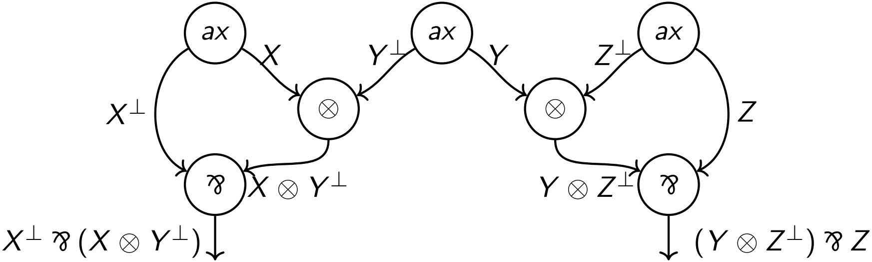
Proof Structure

Definition

Partial directed multigraph with labels on vertices $\rightarrow$ *ax* / *cut* / $\otimes$ / $\parr$

on edges $\rightarrow$ formula



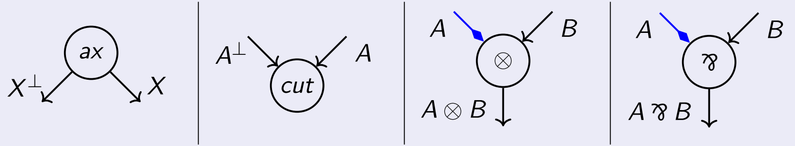


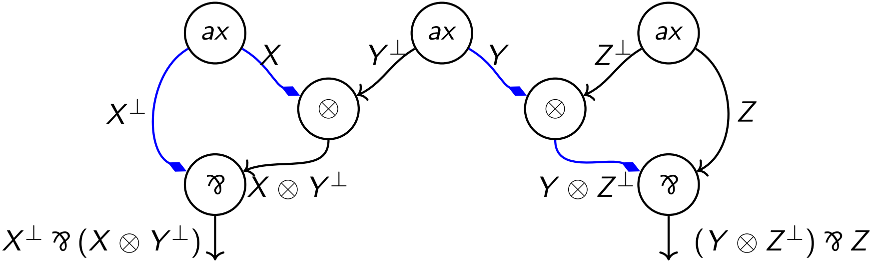
Proof Structure

Definition

Partial directed multigraph with labels on vertices $\rightarrow$ *ax* / *cut* / $\otimes$ / $\parr$

on edges $\rightarrow$ formula



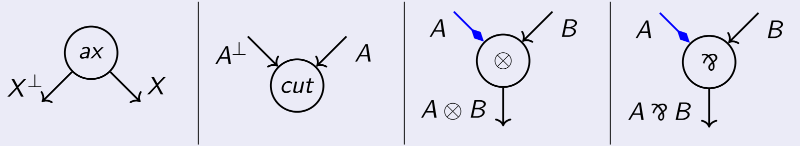


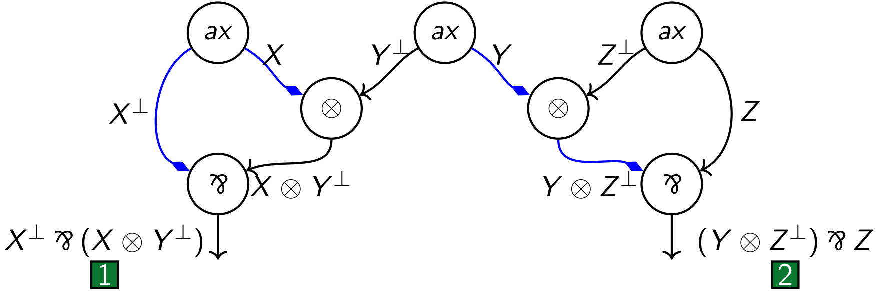
Proof Structure

Definition

Partial directed multigraph with labels on vertices $\rightarrow$ *ax* / *cut* / $\otimes$ / $\parr$
on edges $\rightarrow$ formula

and an ordering of the conclusions



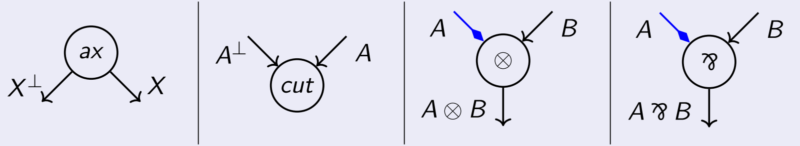


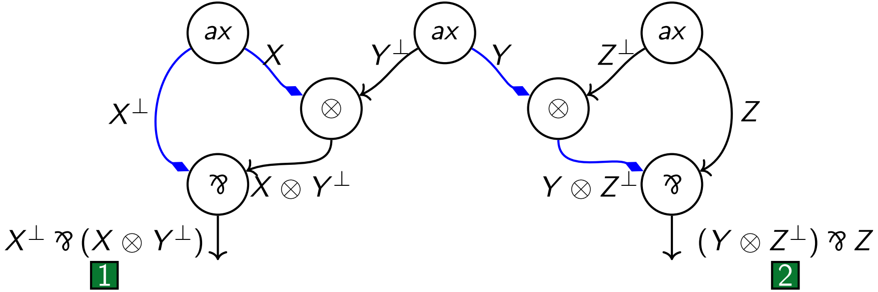
Definition

Partial directed multigraph with labels on vertices $\rightarrow$ *ax* / *cut* / $\otimes$ / $⅋$

on edges $\rightarrow$ formula

and an ordering of the conclusions





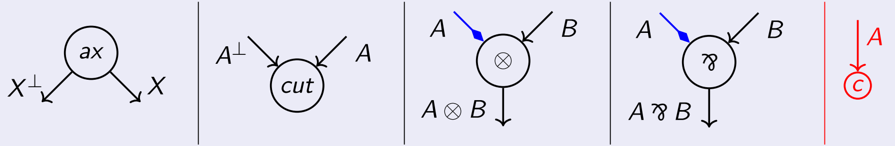
Proof Structure

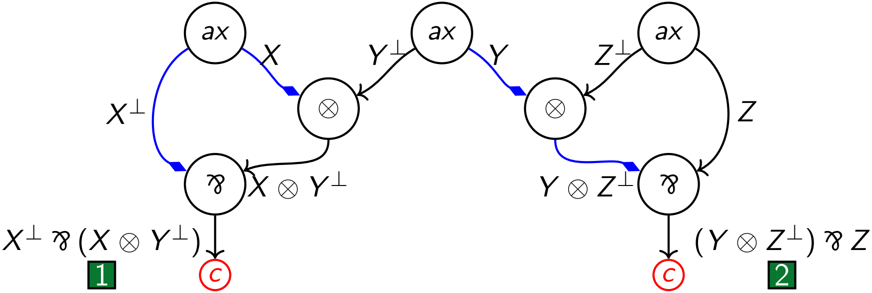
Definition

~~Partial~~ directed multigraph with labels on vertices $\rightarrow$ *ax* / *cut* / $\otimes$ / $⅋$ / c

on edges $\rightarrow$ formula

and an ordering of the conclusions





Correctness Criterion & Proof Net

Definition: Correctness Graph

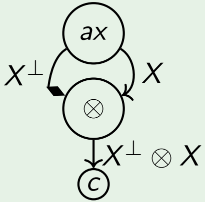
Remove one in-edge of each $⅋$, forget the orientation of edges

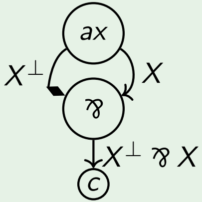
Definition: Correctness Criterion (Danos-Regnier)

Correct $=$ all correctness graphs are **acyclic** and **connected**

Proof Nets $=$ correct proof structures

Toy Examples





Correctness Criterion & Proof Net

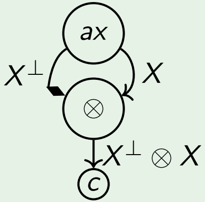
Definition: Correctness Graph

Remove one in-edge of each $\parr$, forget the orientation of edges

Definition: Correctness Criterion (Danos-Regnier)

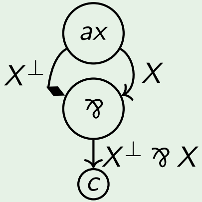
Correct = all correctness graphs are **acyclic** and **connected**
Proof Nets = correct proof structures

Toy Examples



not acyclic (but connected)

INCORRECT



Correctness Criterion & Proof Net

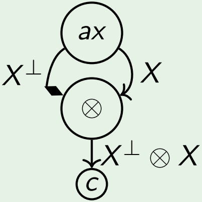
Definition: Correctness Graph

Remove one in-edge of each $⅋$, forget the orientation of edges

Definition: Correctness Criterion (Danos-Regnier)

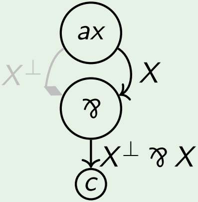
Correct $=$ all correctness graphs are **acyclic** and **connected**
Proof Nets $=$ correct proof structures

Toy Examples



not acyclic (but connected)

INCORRECT



acyclic and connected

Correctness Criterion & Proof Net

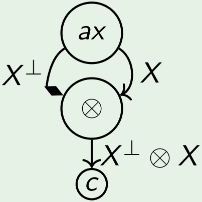
Definition: Correctness Graph

Remove one in-edge of each $\parr$, forget the orientation of edges

Definition: Correctness Criterion (Danos-Regnier)

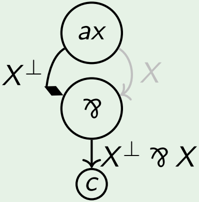
Correct = all correctness graphs are **acyclic** and **connected**
Proof Nets = correct proof structures

Toy Examples



not acyclic (but connected)

INCORRECT

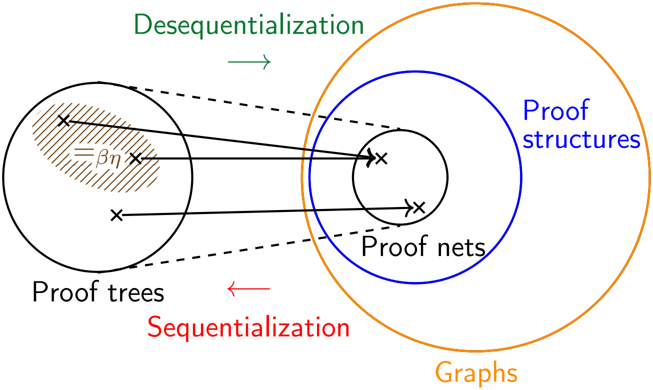


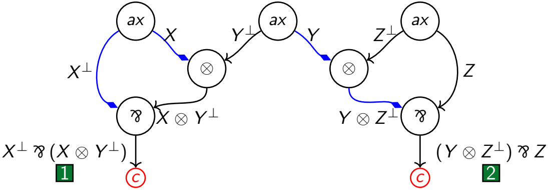
acyclic and connected

CORRECT

Outline

- Unit-free Multiplicative Linear Logic
- **Formalization of Proof Nets**
  - Underlying Graphs
  - Correctness Criterion

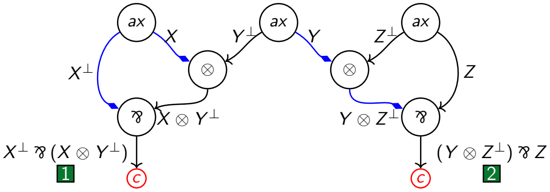




Direct Traduction:

```
Record graph_data : Type :=
  Graph_data {
    graph_of :> graph rule formula;
     left : { v : vertex graph_of | vlabel v == ⊗ || vlabel v == ⅋ }
          → edge graph_of;
     right : { v : vertex graph_of | vlabel v == ⊗ || vlabel v == ⅋ }
          → edge graph_of;
     order : { v : vertex graph_of | vlabel v == c } →
          'I_#|{ v : vertex graph_of | vlabel v == c }|;
  }.
```

Implementation of the underlying graphs



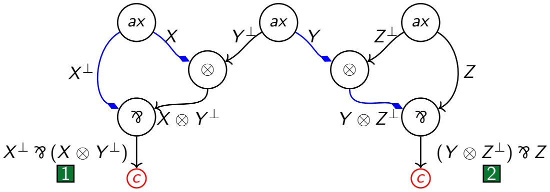
Direct Traduction:

```
Record graph_data : Type :=
  Graph_data {
    graph_of :> graph rule formula;
     left : { v : vertex graph_of | vlabel v == ⊗ || vlabel v == ⅋ }
         → edge graph_of;
     right : { v : vertex graph_of | vlabel v == ⊗ || vlabel v == ⅋ }
         → edge graph_of;
     order : { v : vertex graph_of | vlabel v == c } →
         'I_#|{ v : vertex graph_of | vlabel v == c }|;
  }.
```

$\longrightarrow$ Dependant types quickly too complex

$\rightsquigarrow$ To **define** adding a vertex, need to **prove** that each $\otimes$ before is still a $\otimes$ after

Implementation of the wanted graphs bis

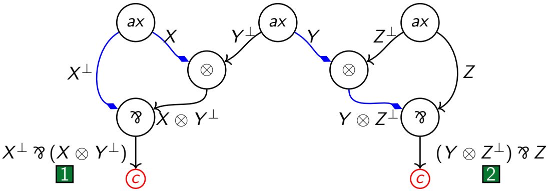


Adopted Solution:

Record *graph_data* : Type :=
  Graph_data {
    graph_of :> *graph rule formula*;
    left : *edge graph_of* → *bool*;
    order : *seq* (*vertex graph_of*);
  }.

- A boolean to mark left arrows
- A list of vertices for ordering the conclusions
- Properties to check:
  - no two marked edges for a same vertex
  - the list contains exactly c-vertices
  - ...

Implementation of the wanted graphs bis



Adopted Solution:

```
Record graph_data : Type :=
  Graph_data {
    graph_of :> graph rule formula;
    left : edge graph_of → bool;
    order : seq (vertex graph_of);
  }.
```

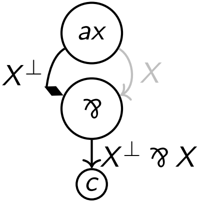
- A boolean to mark left arrows
- A list of vertices for ordering the conclusions
- Properties to check:
  - no two marked edges for a same vertex
  - the list contains exactly c-vertices
  - ...

$\longrightarrow$ Remove difficulties from **definitions** to put them in the **proofs**

Formalization of the Correctness Criterion

Reminder

Correctness Graph: remove one in-edge of each $\parr$, forget the orientation
Correctness: all correctness graphs are **acyclic** and **connected**



On computer: **horrible!**

$\longrightarrow$ correctness graphs of $G + v \simeq$ (correctness graphs of G) $+ v$

$\longrightarrow$ not the same edges in G and in its correctness graphs

$\Longrightarrow$ Not trivial to show that adding v preserves correctness!

Correctness Criterion without Correctness Graphs

Idea: correctness directly in the **proof structure**

Acyclicity

$\iff$ every (undirected) cycle uses both in-edges of a same $\parr$

Correctness Criterion without Correctness Graphs

Idea: correctness directly in the **proof structure**

Acyclicity

$\iff$ every (undirected) cycle uses both in-edges of a same $\parr$ 

Lemma

$acyclic \implies \#cc = \#vertices - \#edges$

$\longrightarrow$ all correctness graphs connected iff the one without left edges is

Connectedness

$\iff$ every pair of vertices are linked by a path not using left in-edges of $\parr$

Correctness Criterion without Correctness Graphs

Idea: correctness directly in the **proof structure**

Acyclicity

$\iff$ every (undirected) cycle uses both in-edges of a same $\parr$ 

Lemma

acyclic $\implies$ $\#cc = \#vertices - \#edges$

$\longrightarrow$ all correctness graphs connected iff the one without left edges is

Connectedness

$\iff$ every pair of vertices are linked by a path not using left in-edges of $\parr$

$\rightsquigarrow$ Study of *particular paths*:

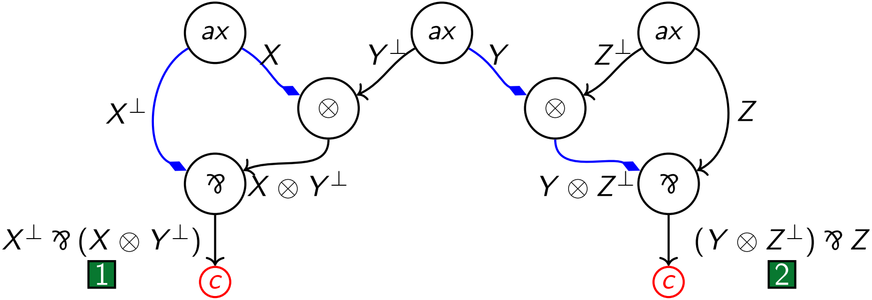
- some edges are incompatible
- some edges are forbidden

Conclusion & Perspectives

What do we have now? *What do we want?*

- **definition** of proof nets
- **desequentialization** from sequents to graphs + it yields a proof net
- **sequentialization**: proof-nets $\simeq$ images of desequentialization
  $\longrightarrow$ the one presented at FSCD yesterday [Di+25]
- **cut-elimination** + gives proof net + convergence + same as sequents
- idem for **axiom-expansion**
- quotient by **rule commutations**
- theory of proof nets: kingdoms, empires
- theorems using proof nets, *e.g.* isomorphisms of MLL [BD99]
- Proof nets for larger systems: MELL, MALL
- ...
- Intermediate results not in **GraphTheory** (undirected paths, etc)

Thank you!



ax
ax
ax
X
$Y^\perp$
Y
$Z^\perp$
$X^\perp$
$\otimes$
$\otimes$
Z
$⅋$
$X \otimes Y^\perp$
$Y \otimes Z^\perp$
$⅋$
$X^\perp ⅋ (X \otimes Y^\perp)$
$(Y \otimes Z^\perp) ⅋ Z$
1
c
c
2

References I

[BD99] Vincent Balat and Roberto Di Cosmo. "A Linear Logical View of Linear Type Isomorphisms". In: *Computer Science Logic*. Ed. by Jörg Flum and Mario Rodríguez-Artalejo. Vol. 1683. Lecture Notes in Computer Science. Springer, 1999, pp. 250–265.

[Bos+11] Anne-Gwenn Bosser, Pierre Courtieu, Julien Forest, and Marc Cavazza. "Structural Analysis of Narratives with the Coq Proof Assistant". In: *2nd International Conference on Interactive Theorem Proving (ITP)*. Ed. by Marko van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk. Vol. 6898. Lecture Notes in Computer Science. Springer, 2011, pp. 55–70. DOI: 10.1007/978-3-642-22863-6_7. URL: https://github.com/Matafou/ill_narratives.

References II

[CLR17] Kaustuv Chaudhuri, Leonardo Lima, and Giselle Reis. "Formalized Meta-Theory of Sequent Calculi for Substructural Logics". In: *Electronic Notes in Theoretical Computer Science* 332 (June 2017). LSFA 2016 - 11th Workshop on Logical and Semantic Frameworks with Applications (LSFA), pp. 57–73. DOI: 10.1016/j.entcs.2017.04.005. URL: https://www.sciencedirect.com/science/article/pii/S1571066117300154.

References III

[CLR19] Kaustuv Chaudhuri, Leonardo Lima, and Giselle Reis. "Formalized meta-theory of sequent calculi for linear logics". In: *Theoretical Computer Science* 781 (2019). Code source at https://github.com/meta-logic/abella-reasoning, pp. 24–38. DOI: https://doi.org/10.1016/j.tcs.2019.02.023. URL: https://www.sciencedirect.com/science/article/pii/S030439751930129X.

References IV

[Di+25] Rémi Di Guardia, Olivier Laurent, Lorenzo Tortora de Falco, and Lionel Vaux Auclair. "Yeo's Theorem for Locally Colored Graphs: the Path to Sequentialization in Linear Logic". In: *International Conference on Formal Structures for Computation and Deduction (FSCD)*. Ed. by Maribel Fernández. Vol. 337. Leibniz International Proceedings in Informatics (LIPIcs). Also available on https://hal.science/hal-04082204. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, July 2025, 16:1–16:18. DOI: 10.4230/LIPIcs.FSCD.2025.16. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.FSCD.2025.16.

References V

[Gro95] Philippe de Groote. "Linear logic with Isabelle: Pruning the proof search tree". In: *Theorem Proving with Analytic Tableaux and Related Methods*. Ed. by Peter Baumgartner, Reiner Hähnle, and Joachim Possega. Springer, Sept. 1995, pp. 263–277. DOI: 10.1007/3-540-59338-1_41. URL: https://link.springer.com/chapter/10.1007/3-540-59338-1_41.

[KP95] Sara Kalvala and Valeria de Paiva. "Mechanizing Linear Logic in Isabelle". In: *Isabelle Users Workshop*. Sept. 1995. URL: https://www.cl.cam.ac.uk/~lp15/papers/Workshop/papers/kalvala-linear.pdf.

[Lau17] Olivier Laurent. *Yalla: Yet Another deep embedding of Linear Logic in Coq*. Coq library. July 2017. URL: https://perso.ens-lyon.fr/olivier.laurent/yalla/.

References VI

[Péd] Pierre-Marie Pédrot. *ll_coq*. Coq library. URL: https://github.com/ppedrot/ll-coq.

[PW99] James Power and Caroline Webster. "Working with Linear Logic in Coq". In: *Theorem Proving in Higher Order Logics: Emerging Trends*. An implementation is available at https://github.com/ComputerAidedLL/PowerWebster_ILL. Sept. 1999. URL: http://www-sop.inria.fr/croap/TPHOLs99/proceeding.html.

[Sad03] Mehrnoosh Sadrzadeh. "Modal Linear Logic in Higher Order Logic, an experiment in Coq". In: *Theorem Proving in Higher Order Logics (01/09/03)*. Ed. by D Basin and W Burkhart. Sept. 2003, pp. 75–93. URL: https://eprints.soton.ac.uk/261814/.

References VII

[Xav+18] Bruno Xavier, Carlos Olarte, Giselle Reis, and Vivek Nigam. "Mechanizing Focused Linear Logic in Coq". In: *12th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2017)*. Ed. by Sandra Alves and Renata Wassermann. Vol. 338. Code source available at https://github.com/meta-logic/coq-ll. 2018, pp. 219–236. DOI: 10.1016/j.entcs.2018.10.014. URL: https://www.sciencedirect.com/science/article/pii/S157106611830080X.

Linear Logic Formalizations in Proof Assistants

[1] https://github.com/olaure01/yalla

[2] https://github.com/ComputerAidedLL/PowerWebster_ILL

[3] https://github.com/meta-logic/coq-ll

[4] https://github.com/Matafou/ill_narratives

[5] https://github.com/ppedrot/ll-coq

[6] https://eprints.soton.ac.uk/261814/

[7] https://github.com/meta-logic/abella-reasoning

[8] https://www.cl.cam.ac.uk/~lp15/papers/Workshop/papers/kalvala-linear.pdf

[9] https://link.springer.com/chapter/10.1007/3-540-59338-1_41

Strata of definitions for proof nets in Rocq (1/2)

```
Notation base_graph := (graph (flat rule) (flat (formula × bool))).
Definition flabel {G : base_graph} (e : edge G) : formula :=
  fst (elabel e).
Definition llabel {G : base_graph} (e : edge G) : bool :=
  snd (elabel e).

Record graph_data : Type :=
  Graph_data {
    graph_of :> base_graph;
    order : seq (edge graph_of);
  }.

Definition sequent (G : graph_data) : seq formula :=
  [seq flabel e | e ← order G].
```

Strata of definitions for proof nets in Rocq (2/2)

```
Record proof_structure : Type := Proof_structure {
    graph_data_of :> graph_data;
    p_deg : proper_degree graph_data_of;
    p_ax_cut : proper_ax_cut graph_data_of;
    p_tens_parr : proper_tens_parr graph_data_of;
    p_noleft : proper_noleft graph_data_of;
    p_order_full : proper_order_full graph_data_of;
    p_order_uniq : proper_order_uniq graph_data_of;
  }.
```

```
Definition proper_tens_parr (G : base_graph) :=
  ∀ (b : bool) (v : G), vlabel v = (if b then ⅋ else ⊗) →
  ∃ el er ec, el \in edges_at_in v ∧ llabel el ∧
    er \in edges_at_in v ∧ ¬llabel er ∧ ec \in edges_at_out v ∧
    flabel ec = (if b then ⅋ else ⊗) (flabel el) (flabel er).
```

```
Record proof_net : Type := ...
```

f-simple Paths

Study undirected paths respecting some conditions:

- **Acyclicity** $\longrightarrow$ some edges are incompatible
- **Connectedness** $\longrightarrow$ some edges are forbidden

$\implies$ new notion of undirected paths

f-simple Paths

Study undirected paths respecting some conditions:

- **Acyclicity** $\longrightarrow$ some edges are incompatible
- **Connectedness** $\longrightarrow$ some edges are forbidden

$\Longrightarrow$ new notion of undirected paths

f-simple Paths

Given an edge-coloring $f : E \longrightarrow I \cup \{\perp\}$, a path p is f-simple when:

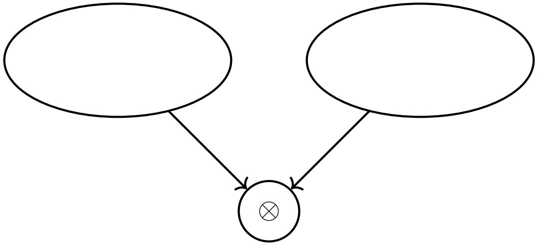
- f is injective on the edges of p
- no edge of p has the forbidden color $\perp$ has an image for f

Difficulties

- Relatively young graph library, lacks some **concepts** for multigraphs: undirected paths, . . .
- **Explicit** manipulations of graphs and their isomorphisms
- Formalizing graph theory reasonnings and their **implicit** arguments

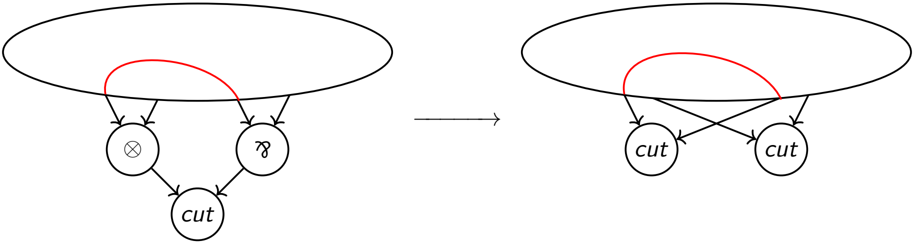
Explicit manipulations of graphs

Proof of Sequentialization: find a splitting vertex almost as easily as on paper, but concluding from there is way more complex!



Explicit manipulations of graphs

Proof of Sequentialization: find a splitting vertex almost as easily as on paper, but concluding from there is way more complex!

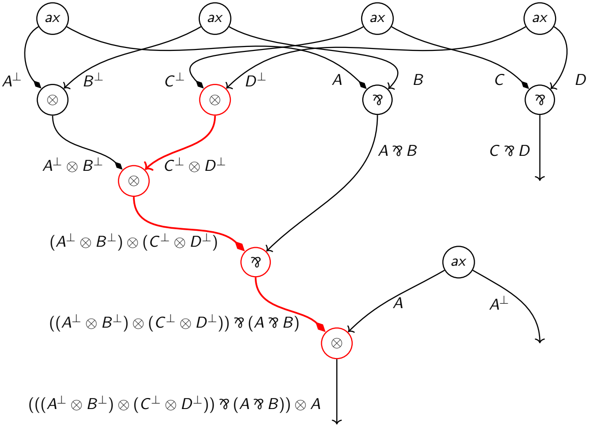


Transfert of paths when adding or removing a vertex, ...

Informal reasoning in graph theory

Descending path

A *descending path* is one whose target has only out-edges towards c-vertices



Formal reasonning in graph theory

Lemma

A proof structure is a DAG (directed acyclic multigraph)

Lemma

The relation "being linked by a directed path" is well-founded in a DAG

Proposition

For a vertex (not c) in a proof structure, there exists a directed path from it to a vertex whose out-edges are all towards c-vertices

Implementation of the wanted graphs bis

Without Dependant Type:

```
Record graph_data : Type :=
  Graph_data {
    graph_of :> graph rule formula;
    left : vertex graph_of → edge graph_of;
    right : vertex graph_of → edge graph_of;
    order : vertex graph_of → int;
  }.
```

Implementation of the wanted graphs bis

Without Dependant Type:

```
Record graph_data : Type :=
  Graph_data {
    graph_of :> graph rule formula;
    left : vertex graph_of → edge graph_of;
    right : vertex graph_of → edge graph_of;
    order : vertex graph_of → int;
  }.
```

⟶ Giving arbitrary values for irrelevant arguments can be the longest part of some definitions / proofs!

Implementation of the wanted graphs bis

Without Dependant Type:

```
Record graph_data : Type :=
  Graph_data {
    graph_of :> graph rule formula;
    left : vertex graph_of → option edge graph_of;
    right : vertex graph_of → option edge graph_of;
    order : vertex graph_of → option int;
  }.
```

⟶ Giving arbitrary values for irrelevant arguments can be the longest part of some definitions / proofs!

⟶ With option types get a boring pattern matching everywhere