On Lambek Grammars

Makoto Kanazawa and Sylvain Salvati

makoto.kanazawa@nii.ac.jp sylvain.salvati@labri.fr

National Institute of Informatics, Tokyo
INRIA-futurs, université de Bordeaux, LaBRI

Fourth Workshop on λ-calculus and formal Grammars

Are LG and CFG strongly equivalent?

- Pentus showed that the languages defined by Lambek Grammars (LG) *were weakly equivalent* to Context Free Grammars (CFGs),

Are LG and CFG strongly equivalent?

- Pentus showed that the languages defined by Lambek Grammars (LG) *were weakly equivalent* to Context Free Grammars (CFGs),
- then the question of *strong equivalence* of the two formalisms has been left open,

Are LG and CFG strongly equivalent?

- Pentus showed that the languages defined by Lambek Grammars (LG) *were weakly equivalent* to Context Free Grammars (CFGs),
- then the question of *strong equivalence* of the two formalisms has been left open,
- an obvious answer to this question is that derivations of LGs are proofs and the proofs of CFGs are trees and that therefore the respective strong generating capacities of the two formalisms cannot be compared.

Are LG and CFG strongly equivalent?

- Pentus showed that the languages defined by Lambek Grammars (LG) *were weakly equivalent* to Context Free Grammars (CFGs),
- then the question of *strong equivalence* of the two formalisms has been left open,
- an obvious answer to this question is that derivations of LGs are proofs and the proofs of CFGs are trees and that therefore the respective strong generating capacities of the two formalisms cannot be compared.

We adopt here a different perspective (or better two different perspectives):

- we show that the relations defined between syntax and semantics the LGs and CFGs can define are actually the same,

Are LG and CFG strongly equivalent?

- Pentus showed that the languages defined by Lambek Grammars (LG) *were weakly equivalent* to Context Free Grammars (CFGs),
- then the question of *strong equivalence* of the two formalisms has been left open,
- an obvious answer to this question is that derivations of LGs are proofs and the proofs of CFGs are trees and that therefore the respective strong generating capacities of the two formalisms cannot be compared.

We adopt here a different perspective (or better two different perspectives):

- we show that the relations defined between syntax and semantics the LGs and CFGs can define are actually the same,
- we give a language theoretic characterization of the set of *normal derivations* of LGs.

Are LG and CFG strongly equivalent?

- Pentus showed that the languages defined by Lambek Grammars (LG) *were weakly equivalent* to Context Free Grammars (CFGs),
- then the question of *strong equivalence* of the two formalisms has been left open,
- an obvious answer to this question is that derivations of LGs are proofs and the proofs of CFGs are trees and that therefore the respective strong generating capacities of the two formalisms cannot be compared.

We adopt here a different perspective (or better two different perspectives):

- we show that the relations defined between syntax and semantics the LGs and CFGs can define are actually the same,
- we give a language theoretic characterization of the set of *normal derivations* of LGs.

Remark: the initial idea of this work was already in Pentus' proof.

Lambek grammars (1)

Given a finite set of categories Cat, Lambek calculus handles formulae NL_{Cat} such that:

$$NL_{Cat} := Cat|(NL_{Cat}/NL_{Cat})|(NL_{Cat}\backslash NL_{Cat})$$

Lambek grammars (1)

Given a finite set of categories Cat, Lambek calculus handles formulae NL_{Cat} such that:

$$NL_{Cat} := Cat|(NL_{Cat}/NL_{Cat})|(NL_{Cat}\backslash NL_{Cat})$$

A LG is 4-tuple $G_{NL} = (W, Cat, \chi, S)$ where:

Lambek grammars (1)

Given a finite set of categories Cat, Lambek calculus handles formulae NL_{Cat} such that:

$$NL_{Cat} := Cat|(NL_{Cat}/NL_{Cat})|(NL_{Cat}\backslash NL_{Cat})$$

A LG is 4-tuple $G_{NL} = (W, Cat, \chi, S)$ where:

- W is a set of *words*,

Lambek grammars (1)

Given a finite set of categories Cat, Lambek calculus handles formulae NL_{Cat} such that:

$$NL_{Cat} := Cat|(NL_{Cat}/NL_{Cat})|(NL_{Cat}\backslash NL_{Cat})$$

A LG is 4-tuple $G_{NL} = (W, Cat, \chi, S)$ where:

- W is a set of *words*,
- Cat is a set of *atomic categories*,

Lambek grammars (1)

Given a finite set of categories Cat, Lambek calculus handles formulae NL_{Cat} such that:

$$NL_{Cat} := Cat|(NL_{Cat}/NL_{Cat})|(NL_{Cat}\backslash NL_{Cat})$$

A LG is 4-tuple $G_{NL} = (W, Cat, \chi, S)$ where:

- W is a set of *words*,
- Cat is a set of *atomic categories*,
- χ is a function from W to finite subsets of NL_{Cat}, and

Lambek grammars (1)

Given a finite set of categories Cat, Lambek calculus handles formulae NL_{Cat} such that:

$$NL_{Cat} := Cat|(NL_{Cat}/NL_{Cat})|(NL_{Cat}\backslash NL_{Cat})$$

A LG is 4-tuple $G_{NL} = (W, Cat, \chi, S)$ where:

- W is a set of *words*,
- Cat is a set of *atomic categories*,
- χ is a function from W to finite subsets of NL_{Cat}, and
- $S \in Cat$.

Lambek Grammars (2)

For LGs, derivations are proofs which establish judgements of the form $\Gamma \vdash A$ where:

- Γ is a list of hypotheses,
- A is a category.

Lambek Grammars (2)

For LGs, derivations are proofs which establish judgements of the form $\Gamma \vdash A$ where:

- Γ is a list of hypotheses,
- A is a category.

The language defined by $G_{NL} = (W, Cat, \chi, S)$ is the set $\{w_1 \dots w_n \vdash \exists \Gamma . \overline{\Gamma} = \langle w_1, A_1 \rangle \cdots \langle w_n, A_n \rangle \wedge \Gamma \vdash S\}$,

Lambek Grammars (2)

For LGs, derivations are proofs which establish judgements of the form $\Gamma \vdash A$ where:

- Γ is a list of hypotheses,
- A is a category.

The language defined by $G_{NL} = (W, Cat, \chi, S)$ is the set $\{w_1 \dots w_n \vdash \exists \Gamma . \overline{\Gamma} = \langle w_1, A_1 \rangle \cdots \langle w_n, A_n \rangle \wedge \Gamma \vdash S\}$,
and derivations are obtained from the following logical rules:

$$\frac{A \in \chi(w)}{\langle w, A \rangle \vdash A}\, Lex \qquad \frac{}{A \vdash A}\, Ax \qquad \frac{\Gamma \vdash B \quad \Delta_1, B, \Delta_2 \vdash A}{\Delta_1, \Gamma, \Delta_2 \vdash A}\, Cut$$

$$\frac{\Gamma \vdash A \quad \Delta_1, B, \Delta_2 \vdash C}{\Delta_1, \Gamma, A \backslash B, \Delta_2 \vdash C}\, L\backslash \qquad \frac{\Gamma \vdash A \quad \Delta_1, B, \Delta_2 \vdash C}{\Delta_1, B/A, \Gamma, \Delta_2 \vdash C}\, L/$$

$$\frac{B, \Gamma \vdash A}{\Gamma \vdash B \backslash A}\, R\backslash \qquad \frac{\Gamma, B \vdash A}{\Gamma \vdash A/B}\, R/$$

Curry-Howard correspondence

We represent proofs on an NL-grammar as terms built on the signature $\Sigma_{G_{NL}}$ where:

- $\mathcal{A}_{G_{NL}} = Cat$,
- $\mathcal{C}_{G_{NL}} = \{\langle w, A\rangle | w \in W \wedge A \in \chi(w)\}$ and
- $\tau_{G_{NL}}(\langle w, A\rangle) = A^{\dagger}$ where $A_1 \backslash A_2{}^{\dagger} = A_2 / A_1{}^{\dagger} = A_1{}^{\dagger} \rightarrow A_2{}^{\dagger}$.

The terms are obtained using the following rules:

$$\frac{A \in \chi(w)}{\langle w, A\rangle \vdash \langle w, A\rangle : A}\, Lex \qquad \frac{}{x : A \vdash x : A}\, Ax$$

$$\frac{\Gamma \vdash t1 : B \quad \Delta_1, x : B, \Delta_2 \vdash t_2 : A}{\Delta_1, \Gamma, \Delta_2 \vdash t_2[x := t_1] : A}\, Cut$$

$$\frac{\Gamma \vdash t_1 : A \quad \Delta_1, x : B, \Delta_2 \vdash t_2 : C}{\Delta_1, \Gamma, y : op(A, B), \Delta_2 \vdash t_2[x := (y t_1)] : C}\, L\ op$$

$$\frac{x : B, \Gamma \vdash t : A}{\Gamma \vdash \lambda x.t : B \backslash A}\, R \backslash \qquad \frac{\Gamma, x : B \vdash t : A}{\Gamma \vdash \lambda x.t : A / B}\, R /$$

Pentus' proof (1)

The proof crucially uses the following very strong interpolation theorem:
If $\Gamma \vdash A$ is derivable then:

- either $\Gamma = \Gamma_1, A_1, A_2, \Gamma_2 \vdash A$ and there is E such that:
  - contains at most as many atoms as the biggest formula in Γ, A,
  - $A_1, A_2 \vdash E$ and $\Gamma_1, E, \Gamma_2 \vdash A$ are derivable, and
  - $$\dfrac{\begin{array}{cc}\vdots & \vdots \\ A_1, A_2 \vdash E & \Gamma_1, E, \Gamma_2 \vdash S\end{array}}{\Gamma \vdash A}\ cut$$ normalizes to the same proof as the former proof of $\Gamma \vdash A$,
- or $\Gamma = \Gamma', B$ and there is an interpolant E such that $\Gamma' \vdash E$ and $E, B \vdash A$ are derivable with the same normalization property as in the previous case,
- or Γ contains at most two formulae.

Pentus' proof (2)

A repeated use of the interpolation theorem proves that if $\Gamma \vdash A$ is derivable then there is a proof of the form:

$$\dfrac{\dfrac{}{A_1^1, A_2^1 \vdash E_1} \quad \dfrac{\dfrac{A_2^n, A_2^n \vdash E_n \quad A_1^{n+1}, A_2^{n+1} \vdash C}{\vdots}\, cut}{\Gamma_1, E_1, \Gamma_2 \vdash S}\, cut}{\Gamma \vdash S}\, cut$$

- this proof normalizes to the initial proof of $\Gamma \vdash A$
- the E_k and the A_1^l, and A_2^l contain at most as much atoms as the biggest formula in Γ, A.

Given an LG G, it easily follows that the CFG whose rules are:

- $[E] \to [A_1][A_2]$ with E, A_1 and A_2 are formulae containing at most as many atoms as the biggest formula in the lexicon of G,
- $[S] \to [A]$ where A is a formula of the lexicon of G,
- $[A] \to w$ if $\langle w, A \rangle$ is in the lexicon of G

defines the same language as G.

Another consequence

If we use the Curry-Howard correspondence we have that whenever $\Gamma \vdash t : A$ is derivable then there is a derivation:

$$\dfrac{\dfrac{}{x_1^1 : A_1^1, x_2^1 : A_2^1 \vdash t_1 : E_1} \quad \dfrac{\dfrac{x_1^n : A_2^n, x_2^n : A_2^n \vdash t_n : E_n \quad x_1^{n+1} : A_1^{n+1}, x_2^{n+1} : A_2^{n+1} \vdash u : C}{\vdots}\, cut}{\Gamma_1, E_1, \Gamma_2 \vdash w : S}\, cut}{\Gamma \vdash v : S}\, cut$$

such that $v \xrightarrow{*}_{\beta} t$.

Another consequence

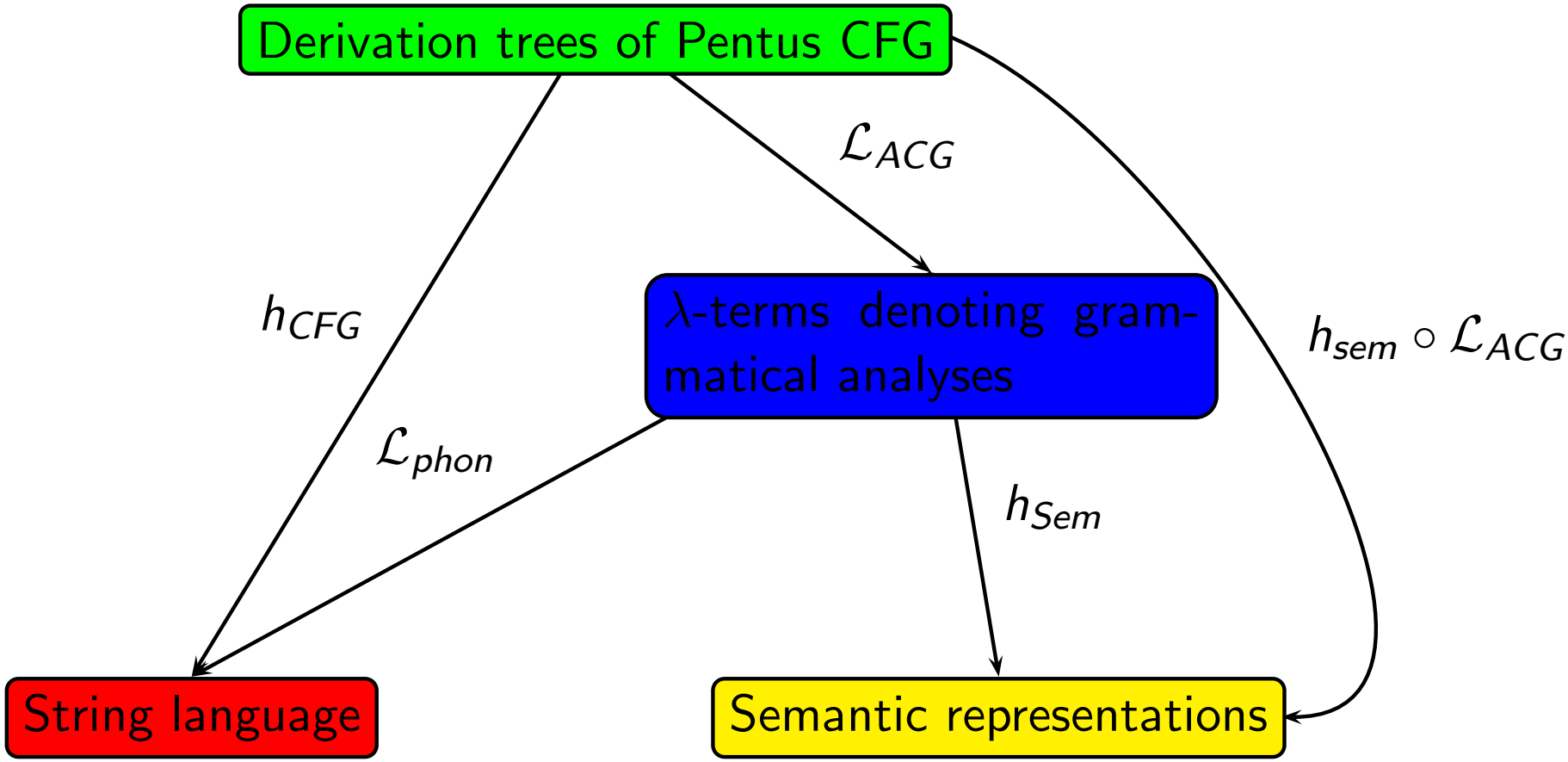
For a Lambek grammar G we may define a second order ACG whose lexicon is of the form:

- $[x_1 : A_1, x_2 : A_2 \vdash t : E] := \lambda x_1 x_2 . t : [A_1] \multimap [A_2] \multimap [E]$ where $x_1 : A_1, x_2 : A_2 \vdash t : E$ is a derivation where A_1, A_2 and E contain at most as many atoms as the biggest formula in the lexicon of G,
- $[x : A \vdash t : S] := \lambda x . t : [A] \multimap [S]$ where $x : A \vdash t : S$ is a derivation and A is a formula of the lexicon of G,
- $[w, A] := \langle w, A \rangle : [A]$

such the set of terms denoting grammatical analyses of G is exactly the set of object terms the lexicon associates to closed abstract terms of type S.

Another consequence

Any relation between syntax and semantics that a LG can define can also be defined on the derivation tree of the context free grammars Pentus gave:



On strong equivalence

If one tries to model the relation between syntax and semantics and understands strong equivalence as the two following properties:

- weak equivalence, and
- the class of semantics interpretation the deep structures allow to define,

then, from the previous remarks, LGs and CFGs are strongly equivalent.

On strong equivalence

If one tries to model the relation between syntax and semantics and understands strong equivalence as the two following properties:

- weak equivalence, and
- the class of semantics interpretation the deep structures allow to define,

then, from the previous remarks, LGs and CFGs are strongly equivalent.

If one is only interested in syntax, then strong equivalence can be characterized only by the relation surface structures and deep structures a formalism defines.

It is then known that (Tiede 99):

- natural deduction trees of Lambek calculus (without the knowledge of the assumption discharged by introduction rules) define unambiguously a unique proof,
- the set of normal natural deduction trees representing grammatical analyses of an LG are not regular.

In this view, LGs are stronger than CFGs. A question remains on a language theoretic characterisation of this set of trees.

A wrong conjecture

It was claimed in Tiede 99 that the set of normal natural deduction trees representing grammatical analyses of an LG are context free. This claim is wrong. Here is a counter-example:
$G_L = (\{a; b; c; d\}, \{e; l; s_1; s_2; f\}, \chi, d)$ with:
$\chi(e) = b, \chi(l) = b\backslash a\backslash b, \chi(s_1) = d/c, \chi(s_2) = b\backslash c,$
$\chi(f) = d/(d/((a\backslash c)\backslash c))$
Natural deduction trees of this grammar have a branch of the form

$$h_1^n(g_1(h_2^n(g_2(h_3^n(h))))))$$

and in any mode of derivation (IO or OI) the branches of trees belonging to context free tree grammars are context free.

A wrong conjecture

It was claimed in Tiede 99 that the set of normal natural deduction trees representing grammatical analyses of an LG are context free.
This claim is wrong. Here is a counter-example:
$G_L = (\{a; b; c; d\}, \{e; l; s_1; s_2; f\}, \chi, d)$ with:
$\chi(e) = b, \chi(l) = b\backslash a\backslash b, \chi(s_1) = d/c, \chi(s_2) = b\backslash c,$
$\chi(f) = d/(d/((a\backslash c)\backslash c))$
Natural deduction trees of this grammar have a branch of the form

$$h_1^n(g_1(h_2^n(g_2(h_3^n(h)))))$$

and in any mode of derivation (IO or OI) the branches of trees belonging to context free tree grammars are context free.

By transforming with a linear lexicon $\mathcal{L}_{ND}$ the λ-terms representing derivations of LGs we may define the language of those trees as a second order ACG whose lexicon is $\mathcal{L}_{ND} \circ \mathcal{L}_{ACG}$.
It is known (Kanazawa 2007) that such trees are trees of hyperedge replacement grammars.

Some more consequences of Pentus' proof

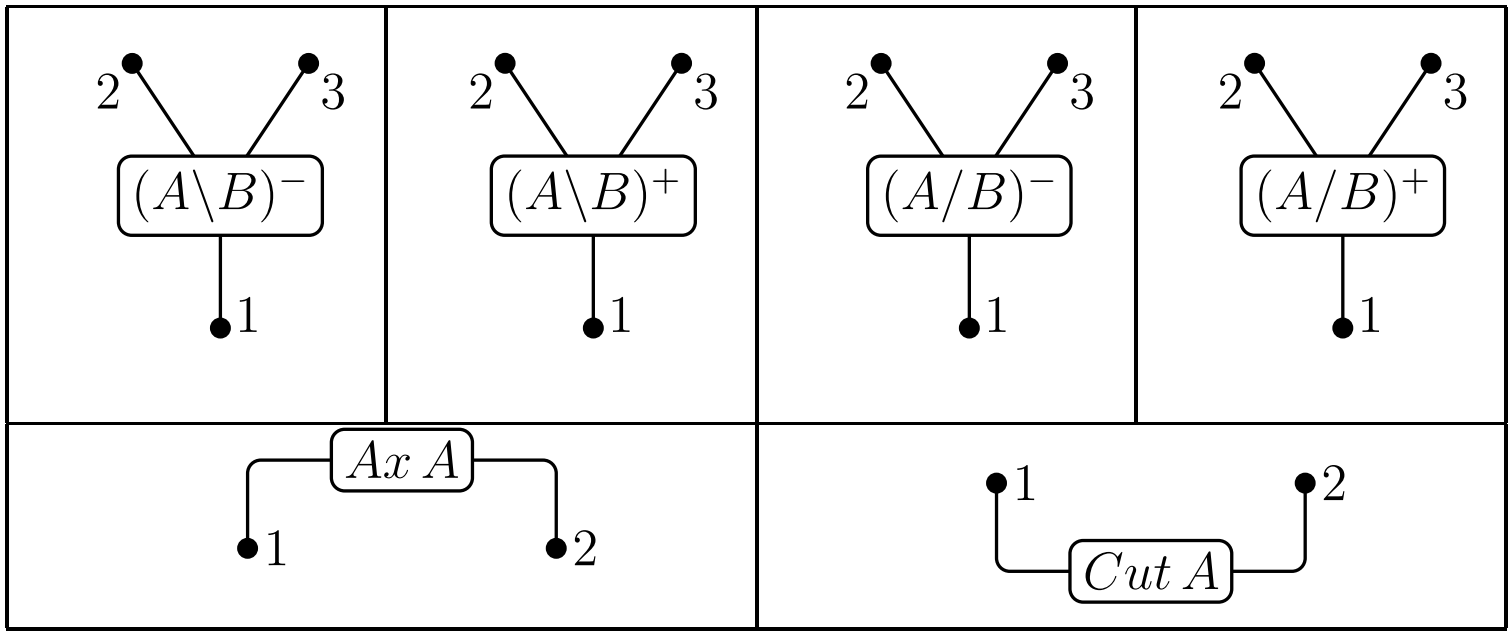
In pentus proof, if derivations are represented as proof-nets, we can reprensent every cut-free proof-nets denoting grammatical analyses by:

- combining a finite number of proof-nets with cut links,
- normalizing the obtained proof-net.

This and Kanazawa 2007 give the insight that this set of proof-nets can be described as the language of a hyperedge replacement grammar (HR).

Proof-nets as hypergraphs

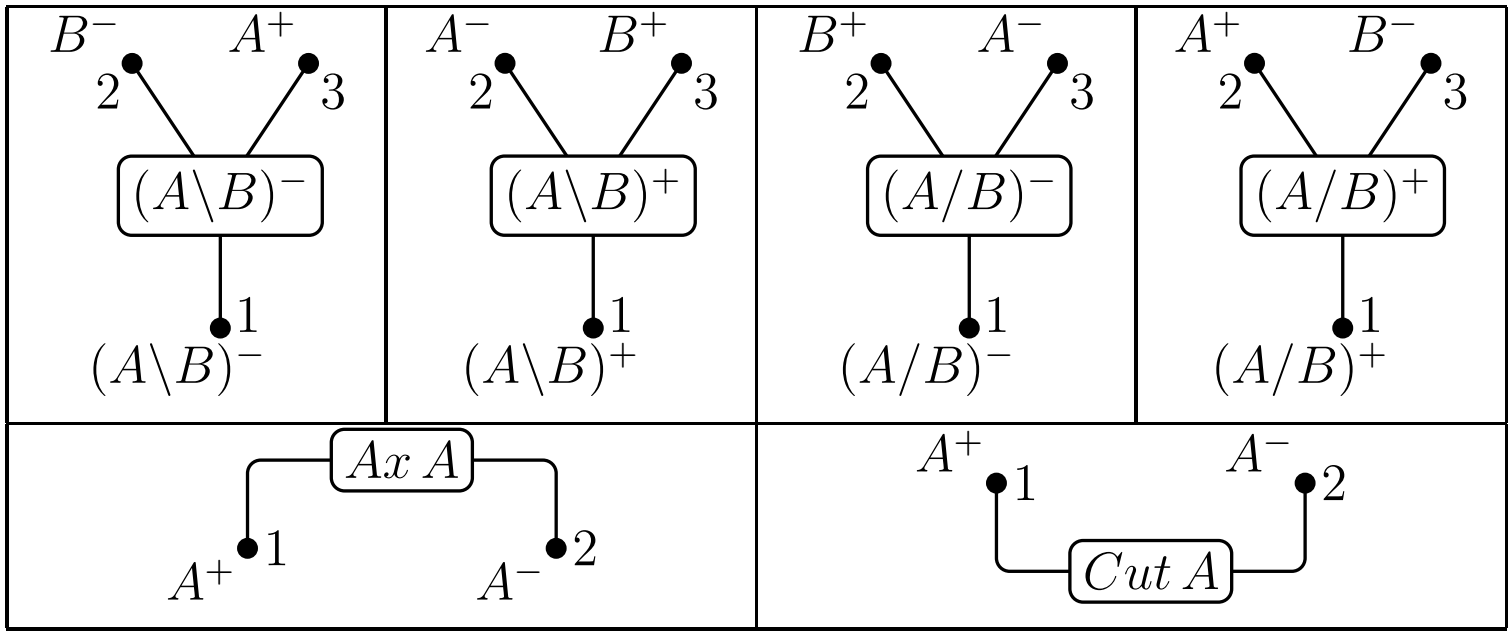
Links may be represented by the following hyperedges:



The subformula property entails that for a given LG cut-free proof-nets denoting grammatical analyses may be construct from a finite number of hyperedges.

Proof-nets as hypergraphs

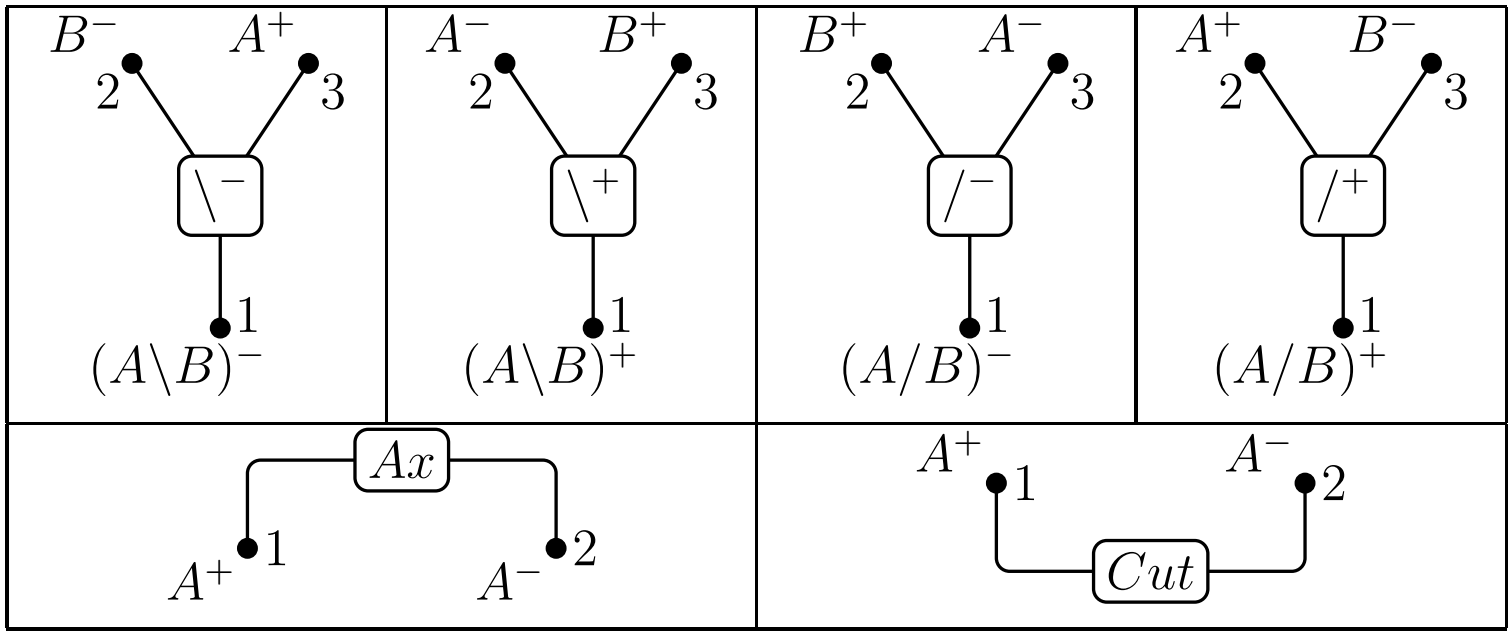
Links may be represented by the following hyperedges:



The subformula property entails that for a given LG cut-free proof-nets denoting grammatical analyses may be construct from a finite number of hyperedges.

Proof-nets as hypergraphs

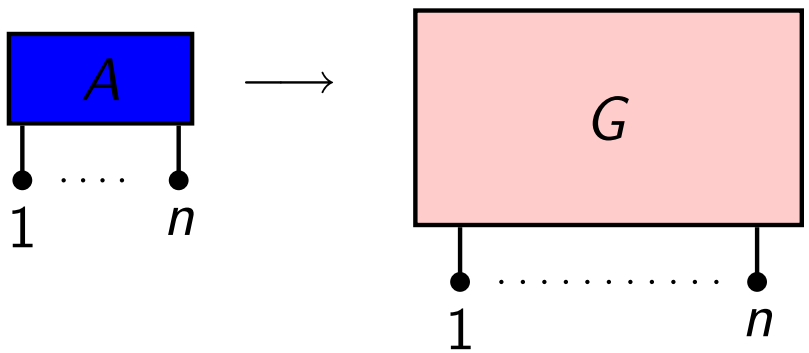
Links may be represented by the following hyperedges:



The subformula property entails that for a given LG cut-free proof-nets denoting grammatical analyses may be construct from a finite number of hyperedges.

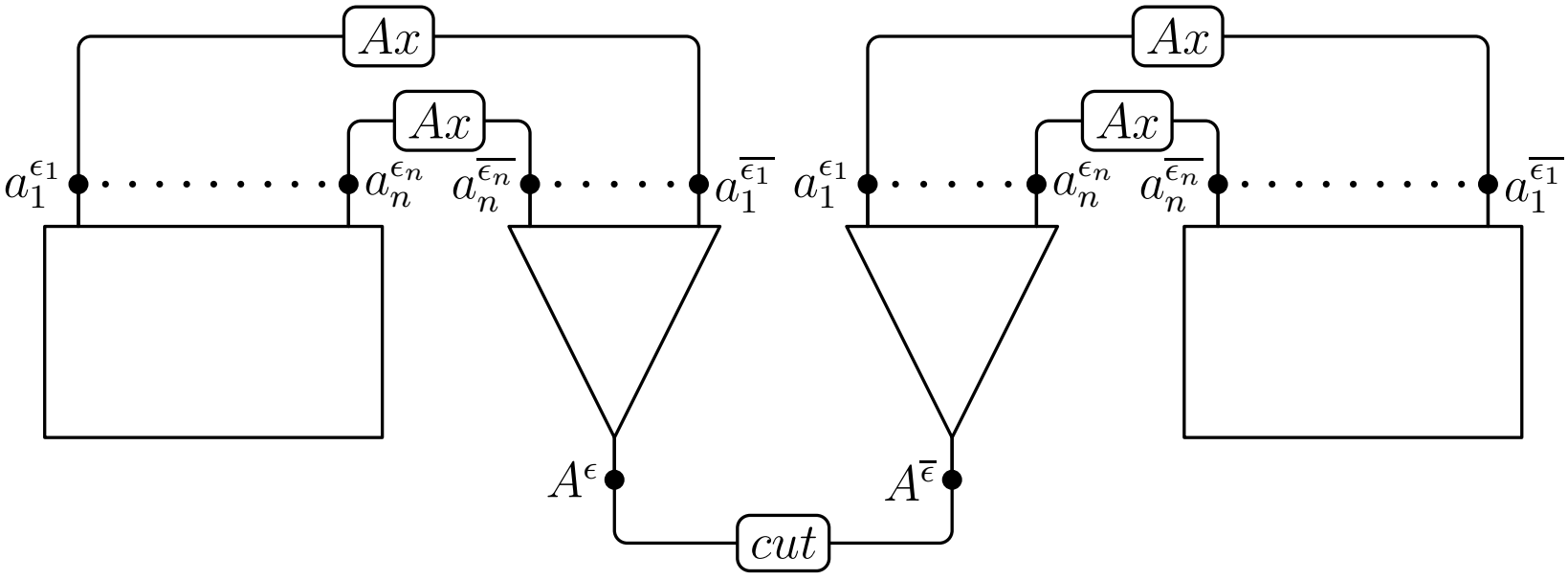
Hyperedge replacement grammars

HRs are grammars with production rules of the form:

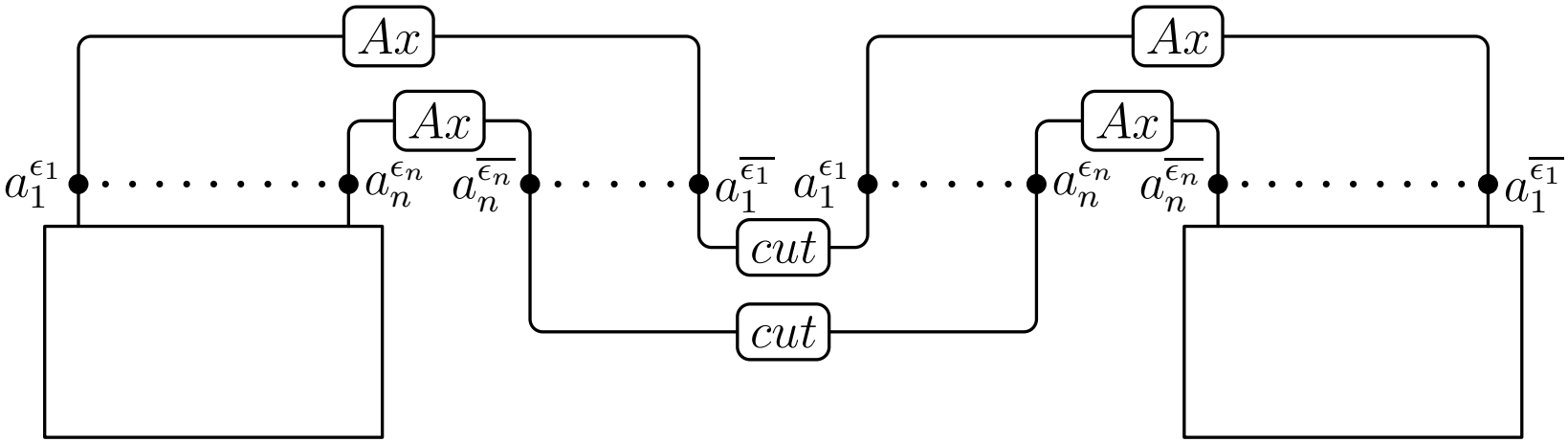


Such a rule is used to replace the non-terminal hyperedge A by the graph G in a hypergraph by merging the external nodes of G with the corresponding nodes connected to A.
Remark that there is no way in this formalism to perform the dynamic of cut-elimination.

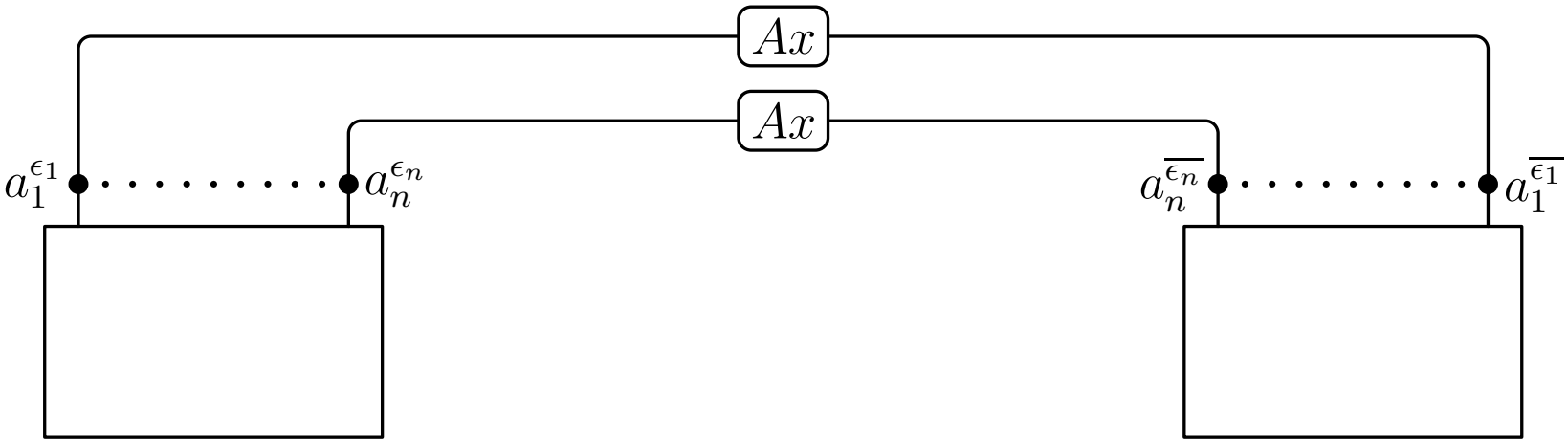
Pre-computing cut-elimination



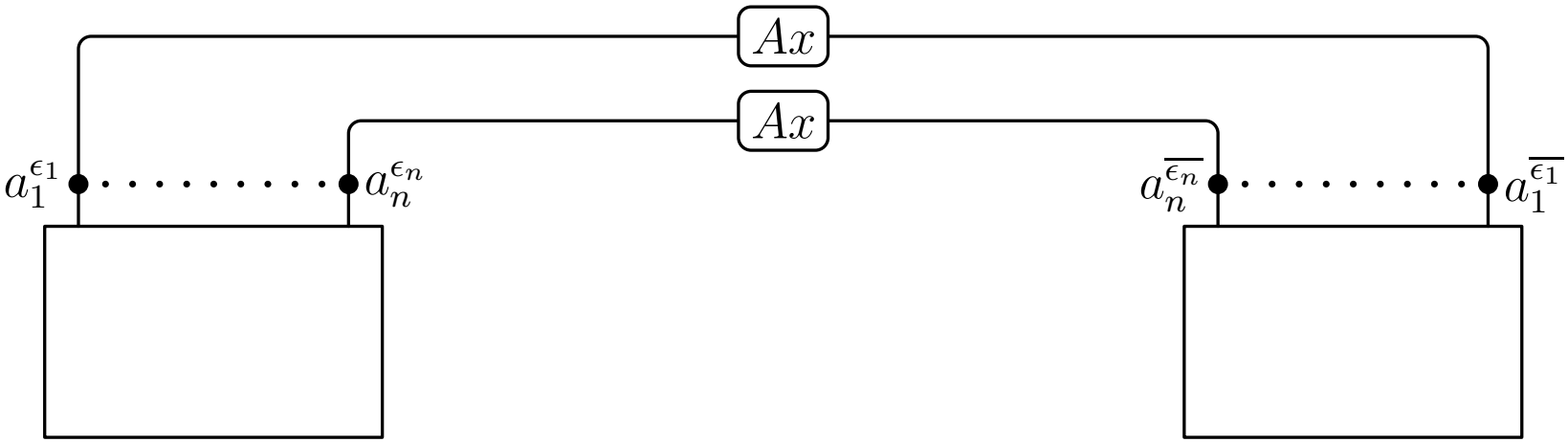
Pre-computing cut-elimination



Pre-computing cut-elimination



Pre-computing cut-elimination



The cut formula establishes which formulae of the two proof-nets have to be the conclusions of the same axiom link. This gives a hint on how a hyperedge replacement grammar will *glue* axiom links of two proof-nets correctly.

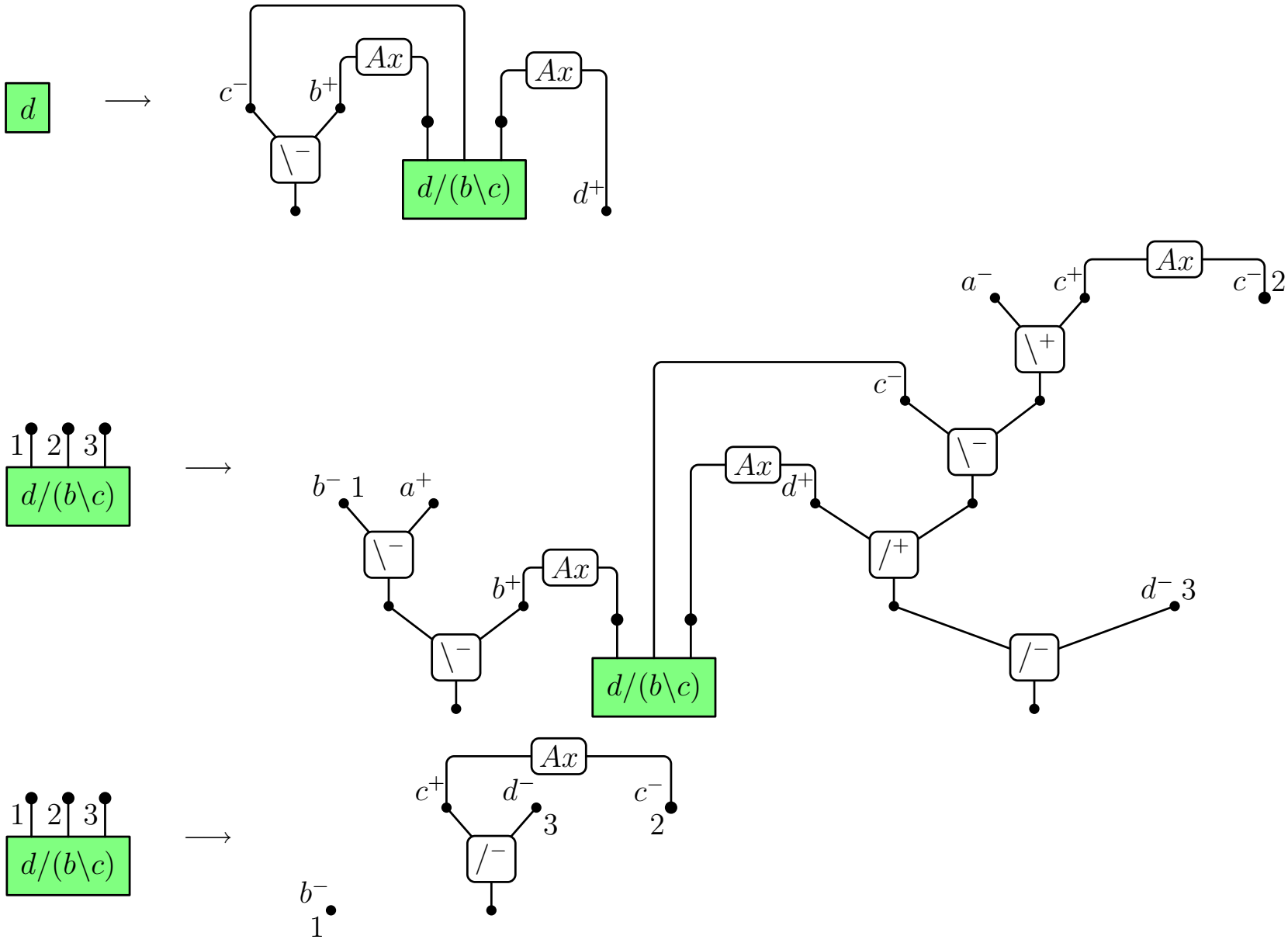
Example (1)

We give a HR whose language is the set of cut-free proof-nets of the LG $G_L = (\{a; b; c; d\}, \{e; l; s_1; s_2; f\}, \chi, d)$ with:
$\chi(e) = b, \chi(l) = b\backslash a\backslash b, \chi(s_1) = d/c, \chi(s_2) = b\backslash c,$
$\chi(f) = d/(d/((a\backslash c)\backslash c))$
We first note that the sequents:

- $d/(b\backslash c), b\backslash c \vdash d,$
- $d/(d/((a\backslash c)\backslash c)), d/(b\backslash c), b\backslash a\backslash b \vdash d/(b\backslash c)$
- $d/c, b \vdash d/(b\backslash c)$

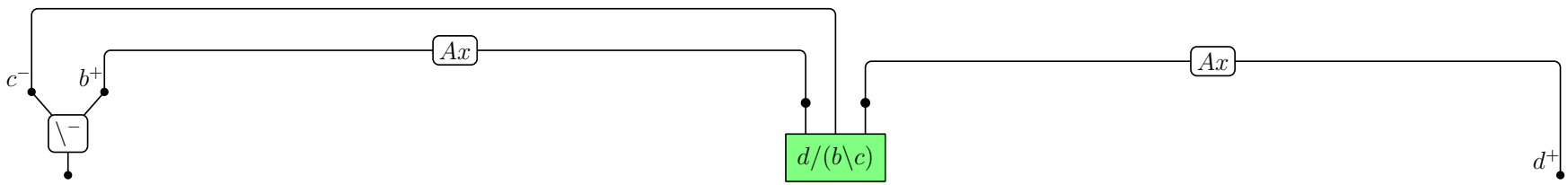
are all derivable and that putting them in cut with $d/(b\backslash c)$ as cut-formula gives all the possible derivations.

Example (2): the rules

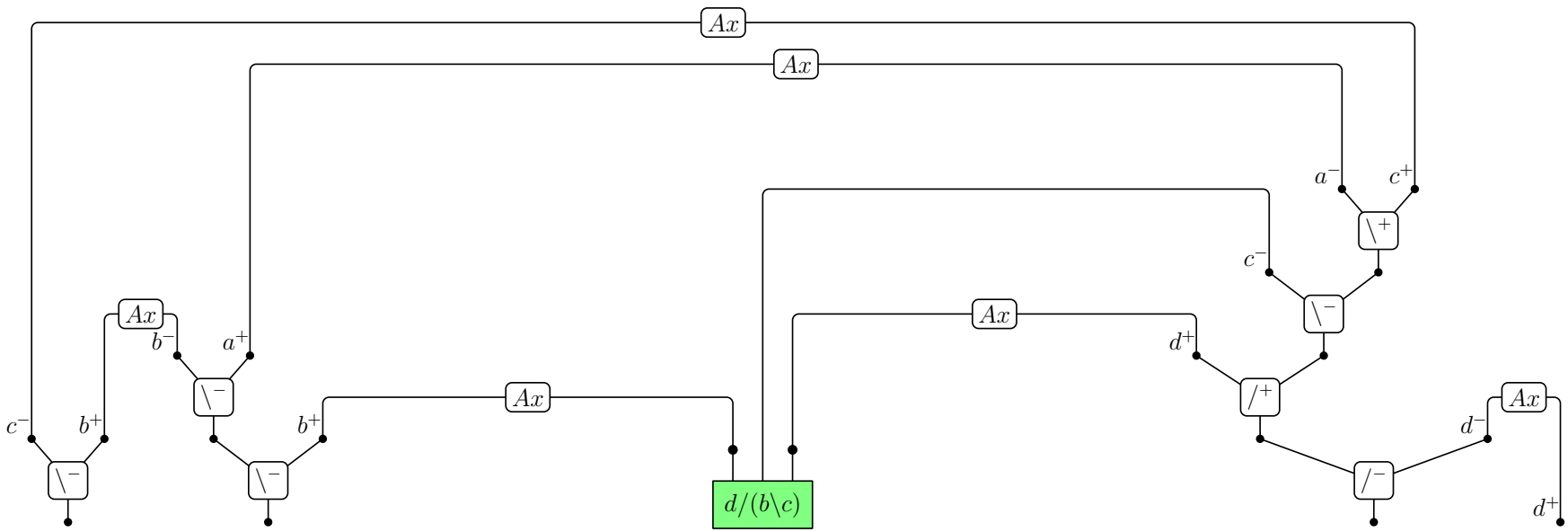


Example (3): a derivation

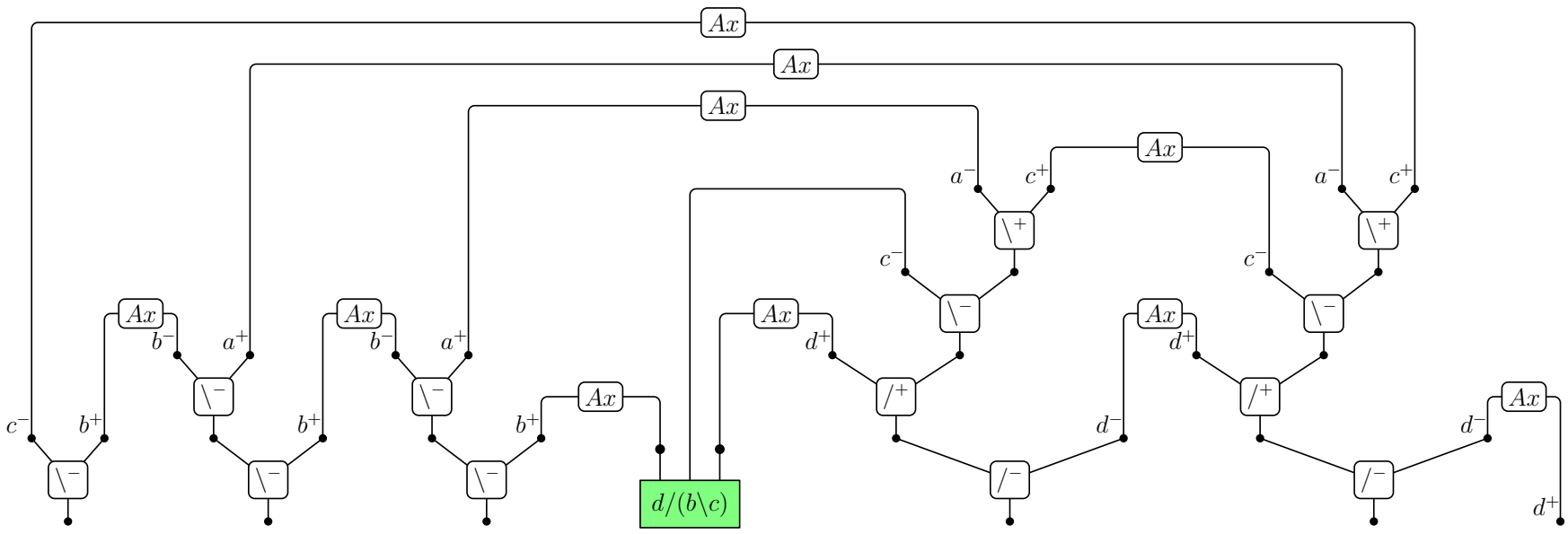
Example (3): a derivation



Example (3): a derivation



Example (3): a derivation



Example (3): a derivation