Retractile Proof Nets of MALL

(Purely **M**ultiplicative and **A**dditive Fragment of **L**inear **L**ogic)

Roberto Maieli

Università degli Studi “Roma Tre”
maieli@uniroma3.it

LPAR’07 Conference - Yerevan, 19th October 2007

Overview

Overview

- **Topic** : we compare the Parallel Syntax (proof-nets, graphs) w.r.t. the Sequential Syntax (sequent-proofs, trees) for proofs of Linear Logic (Girard, 1987).

Overview

- **Topic** : we compare the Parallel Syntax (proof-nets, graphs) w.r.t. the Sequential Syntax (sequent-proofs, trees) for proofs of Linear Logic (Girard, 1987).
- **Question** : finding an intrinsic (geometrical, non inductive) criterion for detecting those graphs (*proof-nets*) that correspond to sequential proofs of the purely multiplicative and additive fragment of linear logic (*MALL*)

Overview

- **Topic** : we compare the Parallel Syntax (proof-nets, graphs) w.r.t. the Sequential Syntax (sequent-proofs, trees) for proofs of Linear Logic (Girard, 1987).
- **Question** : finding an intrinsic (geometrical, non inductive) criterion for detecting those graphs (*proof-nets*) that correspond to sequential proofs of the purely multiplicative and additive fragment of linear logic (*MALL*)
- **Answer** : a correctness criterion formulated like an *algorithm* which implements simple *graph rewriting rules*.

Overview

- **Topic** : we compare the Parallel Syntax (proof-nets, graphs) w.r.t. the Sequential Syntax (sequent-proofs, trees) for proofs of Linear Logic (Girard, 1987).
- **Question** : finding an intrinsic (geometrical, non inductive) criterion for detecting those graphs (*proof-nets*) that correspond to sequential proofs of the purely multiplicative and additive fragment of linear logic (*MALL*)
- **Answer** : a correctness criterion formulated like an *algorithm* which implements simple *graph rewriting rules*.
- **Hint** : an *initial idea* of a retraction correctness criterion for proof nets of MLL, the purely multiplicative fragment of linear logic (Danos, 1990).

MALL

MALL

- *Formulas* $A, B, \ldots$ are built from *literals* by the binary connectives $\otimes$ (*tensor*), $⅋$ (*par*), $\&$ (*with*) and $\oplus$ (*plus*).

MALL

- *Formulas* $A, B, \ldots$ are built from *literals* by the binary connectives $\otimes$ (*tensor*), $⅋$ (*par*), $\&$ (*with*) and $\oplus$ (*plus*).
- *Negation* $(.)^\perp$ extends to any formula by de Morgan laws:

$$(A \otimes B)^\perp = (B^\perp ⅋ A^\perp) \qquad (A ⅋ B)^\perp = (B^\perp \otimes A^\perp)$$
$$(A \& B)^\perp = (B^\perp \oplus A^\perp) \qquad (A \oplus B)^\perp = (B^\perp \& A^\perp)$$

MALL

- *Formulas* $A, B, \ldots$ are built from *literals* by the binary connectives $\otimes$ (*tensor*), $\parr$ (*par*), $\&$ (*with*) and $\oplus$ (*plus*).
- *Negation* $(.)^\perp$ extends to any formula by de Morgan laws:

$$(A \otimes B)^\perp = (B^\perp \parr A^\perp) \qquad (A \parr B)^\perp = (B^\perp \otimes A^\perp)$$
$$(A \& B)^\perp = (B^\perp \oplus A^\perp) \qquad (A \oplus B)^\perp = (B^\perp \& A^\perp)$$

- *Sequents* Γ, Δ are sets of formula occurrences $A_1, \ldots, A_{n \geq 1}$, proved using the following rules (we omit $\vdash$):

MALL

- *Formulas* $A, B, \ldots$ are built from *literals* by the binary connectives $\otimes$ (*tensor*), $⅋$ (*par*), $\&$ (*with*) and $\oplus$ (*plus*).
- *Negation* $(.)^\perp$ extends to any formula by de Morgan laws:

$$(A \otimes B)^\perp = (B^\perp ⅋ A^\perp) \qquad (A ⅋ B)^\perp = (B^\perp \otimes A^\perp)$$
$$(A \& B)^\perp = (B^\perp \oplus A^\perp) \qquad (A \oplus B)^\perp = (B^\perp \& A^\perp)$$

- *Sequents* Γ, Δ are sets of formula occurrences $A_1, \ldots, A_{n \geq 1}$, proved using the following rules (we omit $\vdash$):
  - identity:

$$\frac{}{A, A^\perp}\ \text{ax} \qquad \frac{\Gamma, A \qquad \Delta, A^\perp}{\Gamma, \Delta}\ \text{cut}$$

MALL

- *Formulas* $A, B, \ldots$ are built from *literals* by the binary connectives $\otimes$ (*tensor*), $\parr$ (*par*), $\&$ (*with*) and $\oplus$ (*plus*).
- *Negation* $(.)^\perp$ extends to any formula by de Morgan laws:

$$(A \otimes B)^\perp = (B^\perp \parr A^\perp) \qquad (A \parr B)^\perp = (B^\perp \otimes A^\perp)$$
$$(A \& B)^\perp = (B^\perp \oplus A^\perp) \qquad (A \oplus B)^\perp = (B^\perp \& A^\perp)$$

- *Sequents* Γ, Δ are sets of formula occurrences $A_1, \ldots, A_{n \geq 1}$, proved using the following rules (we omit $\vdash$):
  - identity:
    $$\frac{}{A, A^\perp}\,\text{ax} \qquad \frac{\Gamma, A \qquad \Delta, A^\perp}{\Gamma, \Delta}\,\text{cut}$$
  - multiplicatives:
    $$\frac{\Gamma, A \qquad \Delta, B}{\Gamma, \Delta, A \otimes B}\,\otimes \qquad \frac{\Gamma, A, B}{\Gamma, A \parr B}\,\parr$$

MALL

- *Formulas* $A, B, \ldots$ are built from *literals* by the binary connectives $\otimes$ (*tensor*), $⅋$ (*par*), $\&$ (*with*) and $\oplus$ (*plus*).
- *Negation* $(.)^\perp$ extends to any formula by de Morgan laws:

$$(A \otimes B)^\perp = (B^\perp ⅋ A^\perp) \qquad (A ⅋ B)^\perp = (B^\perp \otimes A^\perp)$$
$$(A \& B)^\perp = (B^\perp \oplus A^\perp) \qquad (A \oplus B)^\perp = (B^\perp \& A^\perp)$$

- *Sequents* Γ, Δ are sets of formula occurrences $A_1, \ldots, A_{n \geq 1}$, proved using the following rules (we omit $\vdash$):
  - identity: $\dfrac{}{A, A^\perp}\,\text{ax}$ $\qquad \dfrac{\Gamma, A \qquad \Delta, A^\perp}{\Gamma, \Delta}\,\text{cut}$
  - multiplicatives: $\dfrac{\Gamma, A \qquad \Delta, B}{\Gamma, \Delta, A \otimes B}\,\otimes$ $\qquad \dfrac{\Gamma, A, B}{\Gamma, A ⅋ B}\,⅋$
  - additives: $\dfrac{\Gamma, A \qquad \Gamma, B}{\Gamma, A \& B}\,\&$ $\qquad \dfrac{\Gamma, A}{\Gamma, A \oplus B}\,\oplus_1$ $\qquad \dfrac{\Gamma, B}{\Gamma, A \oplus B}\,\oplus_2$

Examples of proofs

Examples of proofs

$$
\dfrac{\dfrac{\dfrac{A, A^\perp}{A, B^\perp \oplus A^\perp}\,\oplus_2 \qquad \dfrac{B, B^\perp}{B, B^\perp \oplus A^\perp}\,\oplus_1}{A \& B, B^\perp \oplus A^\perp}\,\& \qquad \dfrac{C, C^\perp}{C \parr C^\perp}\,\parr}{A \& B, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\,\otimes
$$

Examples of proofs

$$\dfrac{\dfrac{\dfrac{A, A^\perp}{A, B^\perp \oplus A^\perp}\oplus_2 \qquad \dfrac{B, B^\perp}{B, B^\perp \oplus A^\perp}\oplus_1}{A \& B, B^\perp \oplus A^\perp}\,\& \qquad \dfrac{C, C^\perp}{C \parr C^\perp}\parr}{A \& B, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\otimes$$

$$\dfrac{\dfrac{\dfrac{A, A^\perp}{A, B^\perp \oplus A^\perp}\oplus_2 \qquad \dfrac{C, C^\perp}{C \parr C^\perp}\parr}{A, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\otimes \qquad \dfrac{\dfrac{B, B^\perp}{B, B^\perp \oplus A^\perp}\oplus_1 \qquad \dfrac{C, C^\perp}{C \parr C^\perp}\parr}{B, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\otimes}{A \& B, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\,\&$$

Equivalence of proofs

Equivalence of proofs

- We would like consider equivalent (in some sense) two sequent proofs (*trees*) when they differ only for the order in which the derivation rules are applied.

Equivalence of proofs

- We would like consider equivalent (in some sense) two sequent proofs (*trees*) when they differ only for the order in which the derivation rules are applied.
- This idea leads to the notion of proof structure (*graph*).

Equivalence of proofs

- We would like consider equivalent (in some sense) two sequent proofs (*trees*) when they differ only for the order in which the derivation rules are applied.
- This idea leads to the notion of proof structure (*graph*).
- In particular, some proof structures (proof nets) can be seen as quotients of classes of sequent proofs that are *equivalent modulo irrelevant permutation of sequent rules*.

Proof Structures (PS)

A PS is an oriented graph s.t. each edge is labelled by a MALL formula and built on the set of nodes according to the following typing constraints:

Proof Structures (PS)

A PS is an oriented graph s.t. each edge is labelled by a MALL formula and built on the set of nodes according to the following typing constraints:

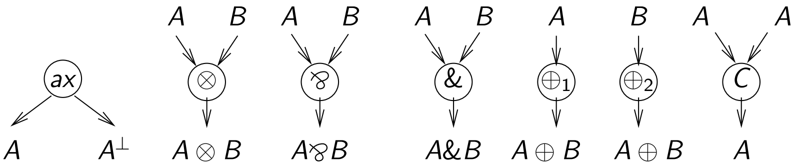


Figure: MALL Links

- entering (resp., exiting) edges are *premises* (resp., *conclusions*)
- pending edges are called *conclusions* of PS

Request for a Correctness Criterion

Request for a Correctness Criterion

- We are interested in finding an intrinsic (geometrical, non inductive) criterion for detecting those proof structures that are correct, i.e. that correspond to proofs of MALL.

Request for a Correctness Criterion

- We are interested in finding an intrinsic (geometrical, non inductive) criterion for detecting those proof structures that are correct, i.e. that correspond to proofs of MALL.
- For doing that we need to go trough some more abstract objects (*Abstract Proof Structures*) which allow us to get rid of some concrete matters of proof structures

Towards a Correctness Criterion: Abstract Proof Structures

Towards a Correctness Criterion: Abstract Proof Structures

– an APS is a non oriented graph G equipped with a set $\mathcal{C}(G)$ of pairwise disjoint pairs of coincident edges labelled by formulas;

Towards a Correctness Criterion: Abstract Proof Structures

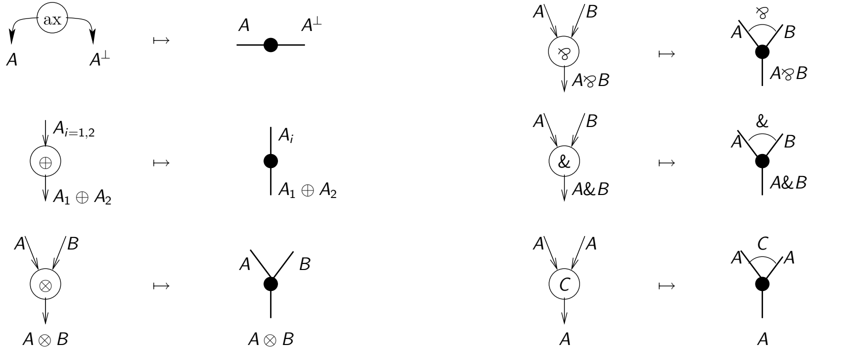
– an APS is a non oriented graph G equipped with a set $\mathcal{C}(G)$ of pairwise disjoint pairs of coincident edges labelled by formulas;
– a base of a pair is a common vertex, labelled by a $\wp, \&$ or C-arc;

Towards a Correctness Criterion: Abstract Proof Structures

– an *APS* is a non oriented graph G equipped with a set $\mathcal{C}(G)$ of pairwise disjoint pairs of coincident edges labelled by formulas;

– a base of a pair is a common vertex, labelled by a $\wp$, & or C-arc;

– a PN is mapped into an APS as follows:



The Correctness Criterion: Proof Nets

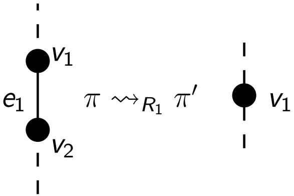
A PS π with conclusions $A_1, ..., A_n$, with $n \geq 1$, is *correct* (i.e., it is a proof net) if its corresponding APS π^* *retracts* to a single node $\bullet$, by iterating the following retraction rules $(R_1, ..., R_5)$

The Correctness Criterion: Proof Nets

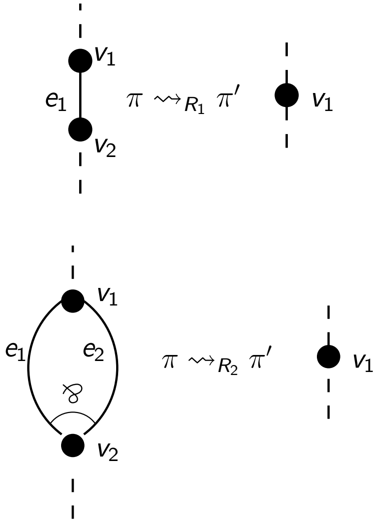
A PS π with conclusions $A_1, ..., A_n$, with $n \geq 1$, is *correct* (i.e., it is a proof net) if its corresponding APS π^* *retracts* to a single node $\bullet$, by iterating the following retraction rules $(R_1, ..., R_5)$

$$\text{a PS } \pi \text{ is a PN iff its corresponding APS } \pi^* \rightsquigarrow^* \bullet$$

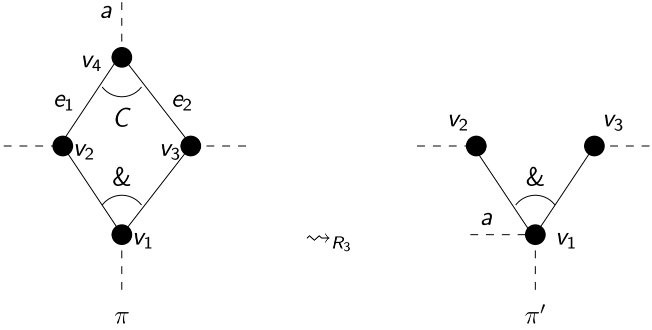
Multiplicative Retraction Rules: R_1 and R_2



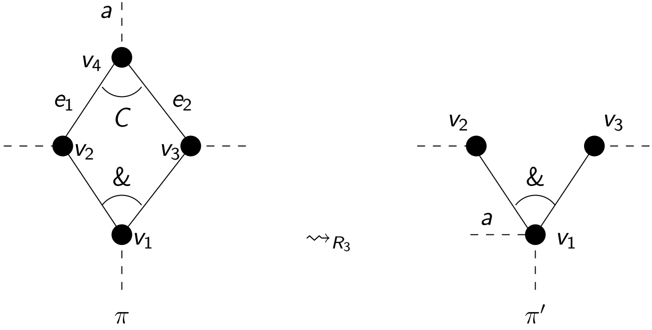
Multiplicative Retraction Rules: R_1 and R_2

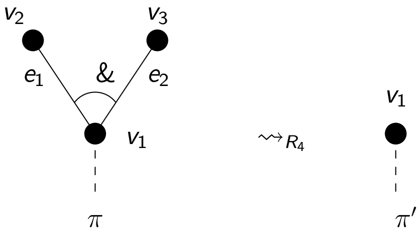


Additive Retraction Rules: R_3 and R_4

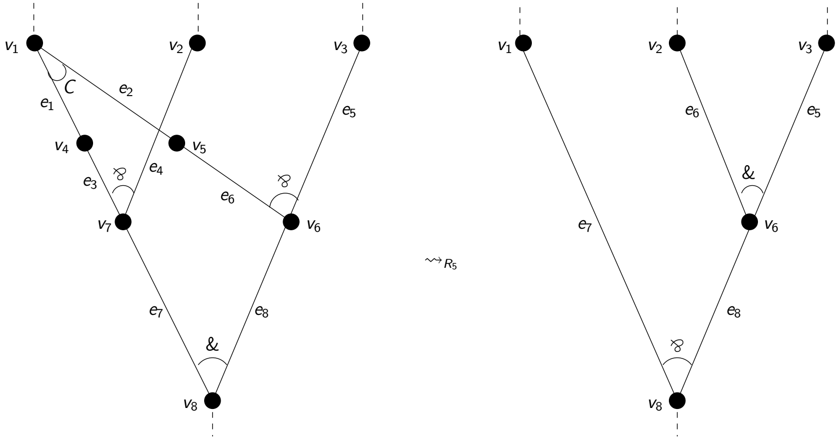


Additive Retraction Rules: R_3 and R_4



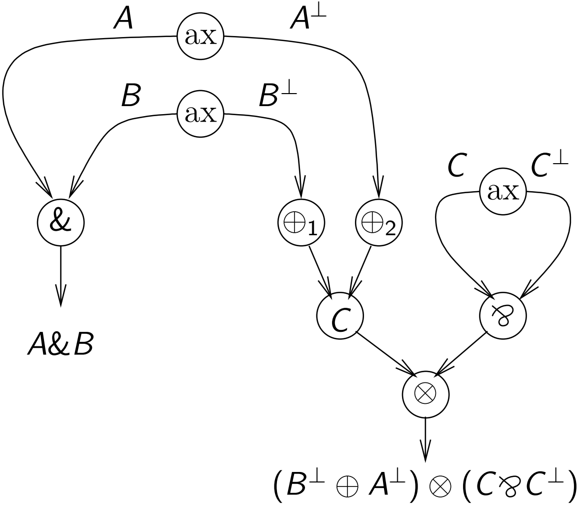


Distributive Retraction Rule: R_5



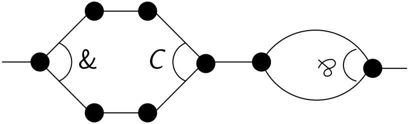
$$(b ⅋ c) \& (b ⅋ d) \vdash b ⅋ (c \& d)$$

An example of Proof Net π (1/2)

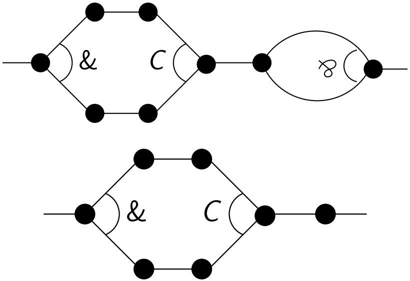


An example of retraction sequence for π^* (2/2)

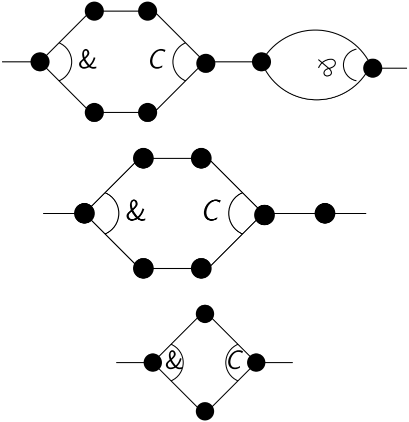
An example of retraction sequence for π^* (2/2)



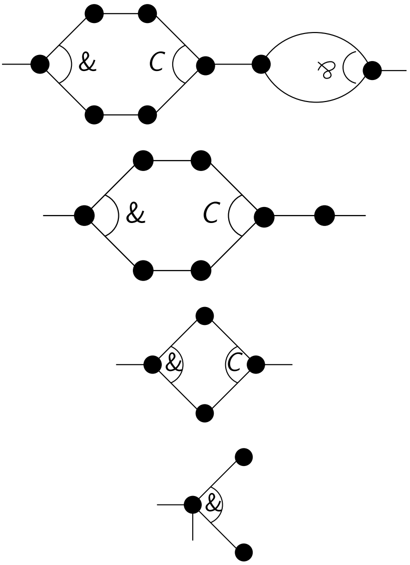
An example of retraction sequence for π^* (2/2)



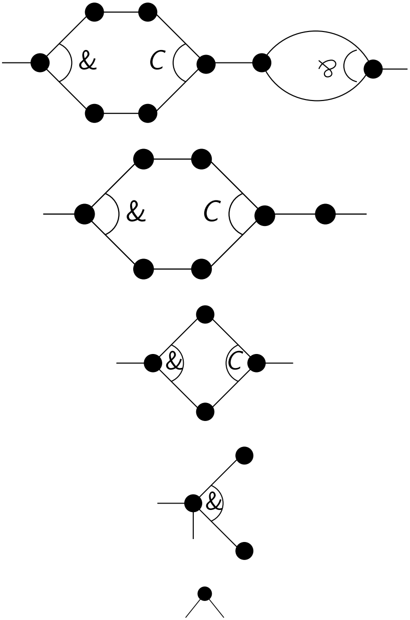
An example of retraction sequence for π^* (2/2)



An example of retraction sequence for π^* (2/2)



An example of retraction sequence for π^* (2/2)



De-squentialization Theorem

Theorem

A proof π *of a sequent* Γ *can be de-sequentialized in to a proof net* π^{-} *with same conclusion.*

De-sequentialization Theorem

Theorem

A proof π *of a sequent* Γ *can be de-sequentialized in to a proof net* π^- *with same conclusion.*

Proof.

We associate a link to each derivation rule, then we proceed by induction on π. □

Squentialization Theorem

A proof net π of a sequent Γ can be sequentialized in to a proof π^{-} with same conclusion.

Squentialization Theorem

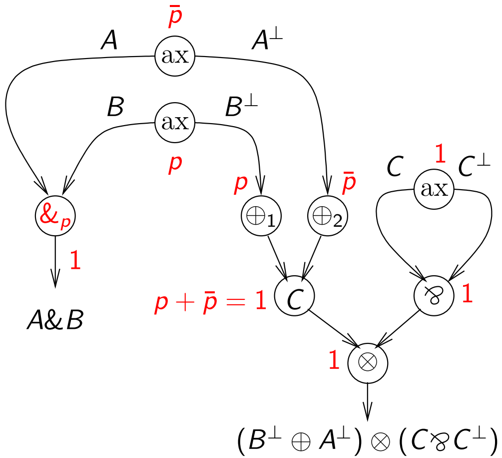
A proof net π of a sequent Γ can be sequentialized in to a proof π^- with same conclusion.

Proof: we use weights, i.e. elements$\neq 0$ of the Boolean Algebra generated by the set eigen variables indexing the &-nodes of π.

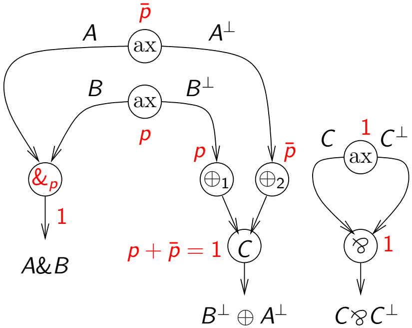
Squentialization Theorem

A proof net π of a sequent Γ can be sequentialized in to a proof π^- with same conclusion.

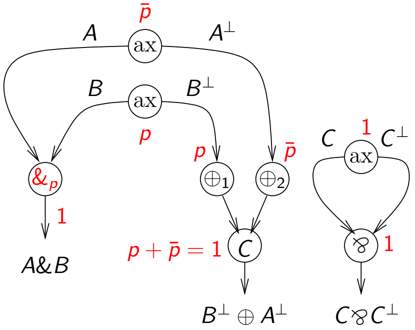
Proof: we use weights, i.e. elements$\neq 0$ of the Boolean Algebra generated by the set eigen variables indexing the &-nodes of π.



A splitting squentialization: ⊗ first (bottom-up)

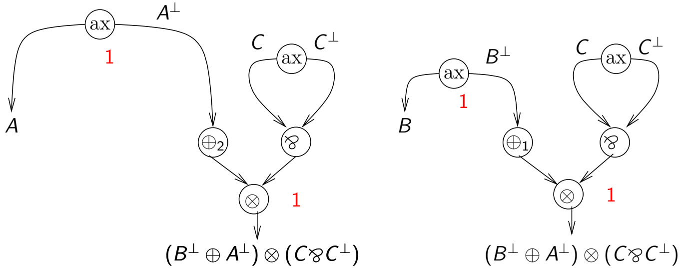


A splitting squentialization: $\otimes$ first (bottom-up)

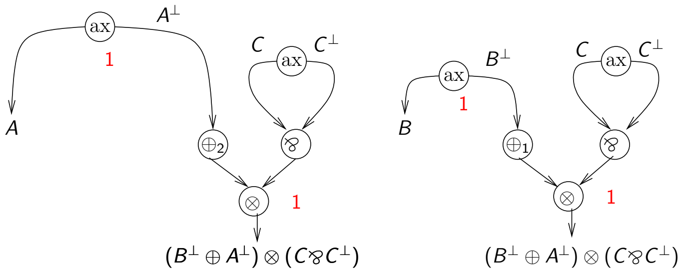


$$\dfrac{\dfrac{\dfrac{A, A^\perp}{A, B^\perp \oplus A^\perp}\,\oplus_2 \qquad \dfrac{B, B^\perp}{B, B^\perp \oplus A^\perp}\,\oplus_1}{A \& B, B^\perp \oplus A^\perp}\,\& \qquad \dfrac{C, C^\perp}{C ⅋ C^\perp}\,⅋}{A \& B, (B^\perp \oplus A^\perp) \otimes (C ⅋ C^\perp)}\,\otimes$$

A slicing squentialization: & first (bottom-up)



A slicing squentialization: & first (bottom-up)



$$\dfrac{\dfrac{\dfrac{A, A^\perp}{A, B^\perp \oplus A^\perp}\,\oplus_2 \qquad \dfrac{C, C^\perp}{C \parr C^\perp}\,\parr}{{\color{red}A}, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\,\otimes \qquad \dfrac{\dfrac{B, B^\perp}{B, B^\perp \oplus A^\perp}\,\oplus_1 \qquad \dfrac{C, C^\perp}{C \parr C^\perp}\,\parr}{{\color{red}B}, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\,\otimes}{{\color{red}A \& B}, (B^\perp \oplus A^\perp) \otimes (C \parr C^\perp)}\,\&$$

Confluence

Theorem

If a PN π retracts to $\bullet$, then all retraction sequences start with π^* and terminate with $\bullet$.

Conclusions

Retractile Correctness Criteria for PN :

Conclusions

Retractile Correctness Criteria for PN :

- can be seen as concurrent (parsing) algorithms for proof-search;

Conclusions

Retractile Correctness Criteria for PN :

- can be seen as concurrent (parsing) algorithms for proof-search;
  - alternative to sequential algorithms performed on sequent calculi;

Conclusions

Retractile Correctness Criteria for PN :

- can be seen as concurrent (parsing) algorithms for proof-search;
  - alternative to sequential algorithms performed on sequent calculi;
  - more efficient and compact, since they are performed on PN (class of equivalent proofs, modulo permutability of rules);

Conclusions

Retractile Correctness Criteria for PN :

- can be seen as concurrent (parsing) algorithms for proof-search;
  - alternative to sequential algorithms performed on sequent calculi;
  - more efficient and compact, since they are performed on PN (class of equivalent proofs, modulo permutability of rules);
  - with low complexity (linear, quadratic, ...)

Conclusions

Retractile Correctness Criteria for PN :

- can be seen as concurrent (parsing) algorithms for proof-search;
  - alternative to sequential algorithms performed on sequent calculi;
  - more efficient and compact, since they are performed on PN (class of equivalent proofs, modulo permutability of rules);
  - with low complexity (linear, quadratic, ...)
- lead to possible applications like Transactional Systems, navigation of Formal Ontologies ...