A Formal Model of A New Linear Time Correctness Condition for Multiplicative Linear Logic

Satoshi Matsuoka
AIST
Motivation

• Can specify many computational problems by Linear Logic formulas
• Be able to specify many NP-complete problems using MLL formulas
• Can Use Proof Search as Problem Solving
• Proof Net Construction: a method of LL Proof Search
  ✓ Construct a proof structure associated with the input specification formula, and then check whether it is a proof net using a correctness condition
  ✓ Efficient (Linear Time) Correct Condition is a Key
Brief History of CCs of MLL proof nets

• 1987: Girard
  Introduction of Proof Nets
  Long Trip Condition for unit-free Multiplicative Linear Logic (MLL)
• 1989: Danos & Regnier
  Acyclic & Connected Condition over DR-graphs
• 1999: Guerrini
  The First Linear Time Correctness Condition based on Danos’s Contractability Condition
• 2000: Murawski & Ong
  Another Linear Time Correctness Conditions based on Lamarche’s Condition of IMLL proof nets
• 2007: de Naurois & Mogbil
  New Correctness Condition Establishing NL-completeness: we only need just one DR-graph
Our result

- A new linear time correctness condition based on de Naurois & Mogbil's work
- Introduction of deNM-trees
- Rewriting system over deNM-trees with three rewrite rules
- Use of union-find data structures in order to establish linear time termination
- Introduction of deadlock prevention mechanism in order to establish completeness of the algorithm
- Easy to implement (in Proof Net Calculator)
- A formal model (a finite transition system) based on the implementation
The system MLL

MLL formulas: $A ::= p \mid p^\perp \mid A \otimes B \mid A \& B$

negation: $(p^\perp)^\perp := p$, $(A \otimes B)^\perp := B^\perp \& A^\perp$, $(A \& B)^\perp := B^\perp \otimes A^\perp$

Inference rules:

\[
\text{ID} \quad \frac{}{\Gamma \vdash A, A^\perp}
\]

\[
\otimes \quad \Gamma \vdash \Sigma_1, A \quad \Gamma \vdash B, \Sigma_2
\]

\[
\frac{}{\Gamma \vdash \Sigma_1, \Sigma_2, A \otimes B}
\]

\[
\& \quad \Gamma \vdash \Sigma, A, B
\]

\[
\frac{}{\Gamma \vdash \Sigma, A \& B}
\]

\[
\text{Cut} \quad \Gamma \vdash \Sigma_1, A \quad \Gamma \vdash A^\perp, \Sigma_2
\]

\[
\frac{}{\Gamma \vdash \Sigma_1, \Sigma_2}
\]

Cut is almost $\otimes$

$\Sigma, \Sigma_1, \Sigma_2$ are multisets of MLL formulas
Links

- **ID-Links**
- **Par-Links**
- **Tensor-Links**

- Multiplicative-Links

Premises
Conclusions
MLL proof structures

• A set of links
• Satisfying two conditions:
  1. any formula (occurrence) is a conclusion of exactly one link
  2. any formula (occurrence) is at most one premise of a link
MLL Proof Nets

• Abstract formal proofs of Multiplicative Linear Logic
• Sequentializable MLL proof structures
Definition of Proof Nets: Sequentializability

(1) ID-links are sequentializable

(2) \( A \otimes B \) is sequentializable if

\[
\begin{align*}
\begin{array}{c}
\text{sequentializable if} \\
\end{array}
\end{align*}
\]

\( A \otimes B = A \otimes B \)

and both \( A \) and \( B \) are sequentializable

(3) \( A \otimes B \) is sequentializable if

\[
\begin{align*}
\begin{array}{c}
\text{sequentializable if} \\
\text{sequentializable}
\end{array}
\end{align*}
\]
A DR-switching S for a proof structure $\Theta$

- A function from the set of par-links in $\Theta$ to \{Left, Right\}
The DR graph $\Theta S$ induced by DR-switching $S$

(1) if $A \overline{A}$ occurs in $\mathcal{H}$ then $A \overline{A}$ is an edge of $\mathcal{H}_S$

(2) if $A \otimes B$ occurs in $\mathcal{H}$ then $A \otimes B$ and $A \otimes B$ are two edges of $\mathcal{H}_S$

(3) if $A \not\Rightarrow B$ occurs in $\mathcal{H}$ and $S(\overline{A \not\Rightarrow B}) = L$ then $A \not\Rightarrow B$ is an edge of $\mathcal{H}_S$

(4) if $A \not\Rightarrow B$ occurs in $\mathcal{H}$ and $S(\overline{A \not\Rightarrow B}) = R$ then $A \not\Rightarrow B$ is an edge of $\mathcal{H}_S$
A Graph-theretic characterization of proof nets

• Theorem (Girard, Danos-Regnier)
  A MLL proof structure $\Theta$ is a MLL proof net iff
  for any DR-switching $S$ for $\Theta$,
  the DR-graph $\Theta S$ is acyclic and connected
Our new linear time correctness condition

1. Given an MLL proof structure $\Theta$
2. Select a DR-switching $S$ for $\Theta$
3. If the DR-graph $S(\Theta)$ is not acyclic and connected, then $\Theta$ is not a proof net
4. Otherwise, construct the $deNM$-tree $T$ for $\Theta$ and $S$,
   1. check whether or not $T$ can reduce to exactly one node $deNM$-tree using three rewrite rules
   2. If it succeeds, then $\Theta$ is a proof net
   3. Otherwise, not a proof net
deNM-trees

- Trees consisting of the following two types of nodes:

  - labeled-node
    - $t$
    - $\ldots$
    - $S$ is a set of labels $l_L$ or $r_L$ where $L$ is a $\aleph$-link
    - $t \geq 0$

  - $\aleph$-node (degree 1)  $\aleph$-node (degree 2)
    - $\aleph$ $
    - $\aleph$ $
    - $L$
    - $L$
Example 1: An MLL proof net
Example 1: its deNM-tree
Rewriting System

- Rewriting System over deNM-trees
- Only three rewrite rules
- Choose an active (labeled) node arbitrarily
Three rewrite rules

union

\[ S_1 \cup S_2 \]

\( \mathcal{G} \)-elimination

\[ \mathcal{G}_L \]

local jump

\[ \mathcal{G}_L \]

\[ S_1 \]

\[ S_2 \]
Example 1: its deNM-tree
Example 1: the reduced one node deNM-tree

\{ l_1, r_1, l_2, r_2, l_3, r_3, l_4, r_4 \}
Example 2: An MLL proof structure, but not PN
Example 2: its deNM-tree
Example 2: the reduced deNM-tree
Remark

• Any naïve implementation has worst cases terminating in quadratic time

• Need of more sophisticated data structures: union-find data structures
Union-Find Data Structures

• Maintenance of a partition on a finite set $S = \{a_1, a_2, \ldots, a_n\}$:
  • Each partition (subset) $S_a$ of $S$ has the representative element $a$
  • $\text{union}(a, b)$: makes the union set $S_a \cup S_b$, where $S_a$ (resp. $S_b$) is the current subset including $a$ (resp. $b$)
  • $\text{find}(a)$: return the representative element $b$ of $S_b$ that includes $a$

• Can check whether two elements $a$ and $b$ belong to the same partition

• Runs in almost linear time (in the sense of amortized cost)

• Runs in linear time over special cases, especially over trees
Our finite transition system

• States \( \langle a, N, n, P, S_{\text{elim}}, \text{num}_{\text{labeled}}, \text{num}_\forall \rangle \)
  - \(a\) denotes the current active node
  - \(N_i\) maintains the information about labeled node \(i\)
  - \(n_i\) denotes the representative element of the partition including \(i\)
  - \(P_j\) maintains the information about \(\forall\) -node \(j\)
  - etc.
Our transition system (Cont.)

• Four transition rules:
  - elimination rewrite rule is divided into two transition rules
• Use of union-find data structures
  - union operation in union transition rule
  - find operation in two elimination transition rules for the elimination condition
  - find operation in local jump transition rule in order to find the next active node
• Each transition rule only uses constant number of Union-Find operations
  ⇒ Linear time termination
Deadlock prevention mechanism

- Without it, we would judge that correct proof nets were not: can not establish completeness of the algorithm
- Realized by *queue data structures* and *union-find data structures*
- Indices for not yet eliminated $\mathcal{R}$-nodes are maintained in queues
- Its correctness (deadlock freedom) can be stated by a *liveness property*:
  
  “Indices for not yet eliminated $\mathcal{R}$-nodes *must* be always in some queues”
Additional Results

• (Yet another) linear time sequentialization algorithm
  Easy to implement
  The basic idea is to repeat linear time CC twice
• Linear time algorithm for automated generation of planar proof nets
  can be reduced to decremental graph connectivity problem
Our implementation

• Can be downloaded from

https://staff.aist.go.jp/s-matsuoka/PNCalculator/index.html

as a software package called *Proof Net Calculator*
Future Work

• Incorporation of backtracking mechanism
  Seeking elegant (verifiable) implementation
• Semi-persistent data structures?
• Extensions to other fragments and/or variants of LL
Thank you