Encoding MELL Cut Elimination into a Hierarchical Graph Rewriting Language

Kento Takyu* Kazunori Ueda

Waseda University

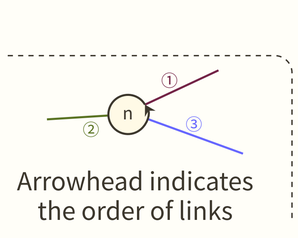
Abstract and Contributions

- ✓ We show that several key language constructs of the hierarchical graph rewriting language LMNtal correspond directly to the operations required for the cut elimination of MELL (Multiplicative Exponential Linear Logic) Proof Nets.
- ✓ This implies that LMNtal serves as a useful workbench for Proof Nets.

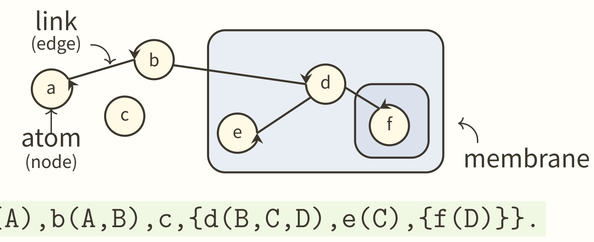
1. LMNtal: a Hierarchical Graph Rewriting Language

LMNtal is a programming language that simultaneously supports two structuring mechanisms, Graphs (connectivity) and Membranes (hierarchy).

Graphs with Membranes:

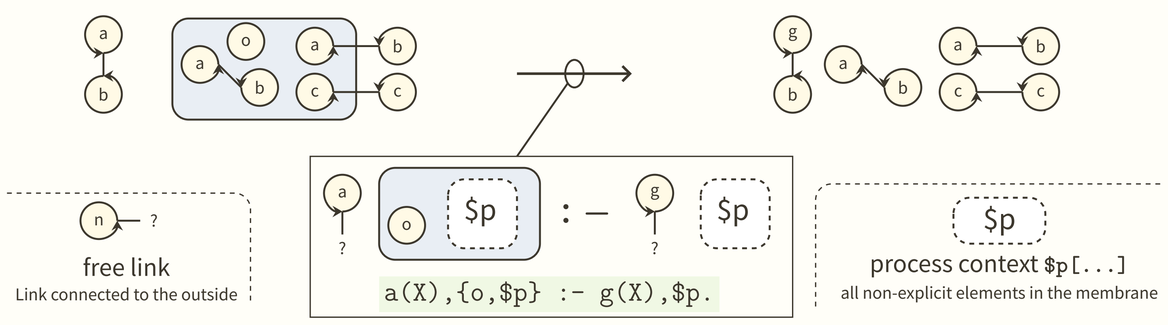


Arrowhead indicates the order of links




```
a(A),b(A,B),c,{d(B,C,D),e(C),{f(D)}}.
```

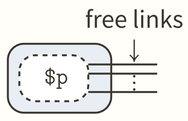
Application of a Rewriting Rule:



Operations on Membranes:

Protection (from rewriting), copy, remove, move hierarchy.

These operations should handle **free links** properly.



Non-determinism and Model Checking:

The LMNtal toolkit is publicly available: https://github.com/lmntal/

- **State space** of graph rewriting can be constructed and **visualized**.
- Furthermore, the toolkit provides an **LTL model checker**.

2. MELL Proof Nets

Multiplicative Exponential Linear Logic (MELL):

Multiplicative Linear Logic (MLL) + Exponential operator !, ?

$$\text{formula } F ::= X(atomic) \mid F^{\perp} \mid F \otimes F \mid F \parr F \mid !F \mid ?F$$

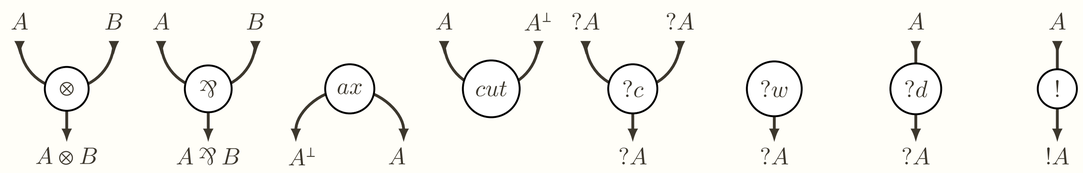
$$A^{\perp\perp} := A \quad (A \otimes B)^{\perp} := A^{\perp} \parr B^{\perp} \quad (A \parr B)^{\perp} := A^{\perp} \otimes B^{\perp} \quad (!A)^{\perp} := ?(A^{\perp}) \quad (?A)^{\perp} := !(A^{\perp}) \quad A \multimap B := A^{\perp} \parr B$$

$$A \to B \mapsto (!A \multimap B)$$

Proof Nets:

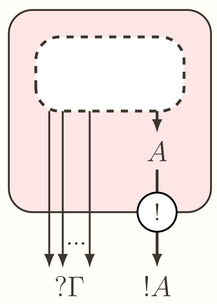
Proofs represented as Graphs, which could be reduced to more concise forms through cut elimination.

MELL Proof Nets is constructed by the following elements:



- Directed, acyclic, multigraph.
- Cells are labelled with inference rules; wires are labelled with formulae.
- **Only cells ⊗, ⅋ have two ordered inputs** (not cut and $?c$).

Promotion Box:



- Corresponding to !-rule of the sequent calculus.
- Must be either disjoint or nested.
- Protects the inner items from reduction.

※ $?\Gamma$ is an unspecified number of formulas with "?".

Cut Elimination Rules:

Cut elimination is expressed as a set of rewrite rules of Proof Nets that involves **copying, removing** or **moving Promotion Boxes.**

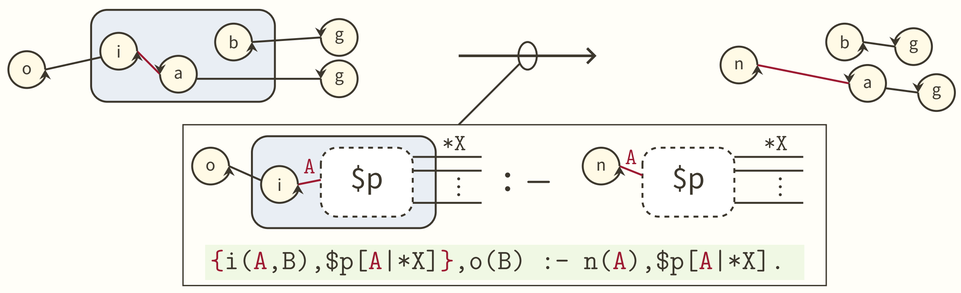
? **How can we express reduction rules for nested structures with unspecified number of edges?**

3. Encoding Cut Elimination into LMNtal

Bundle *X: Construct to handle an unspecified number of free links.

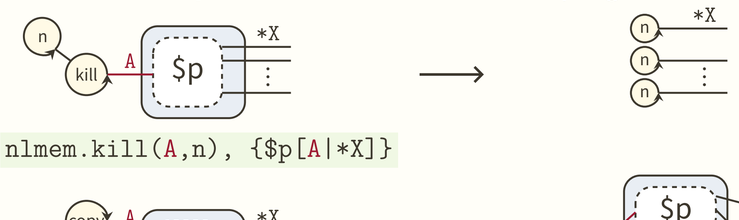
The args of a process context `$p[...]` describe the conditions on its free link.

e.g., `$p[A|*X]` includes the link `A` and zero or more free links represented by `*X`.

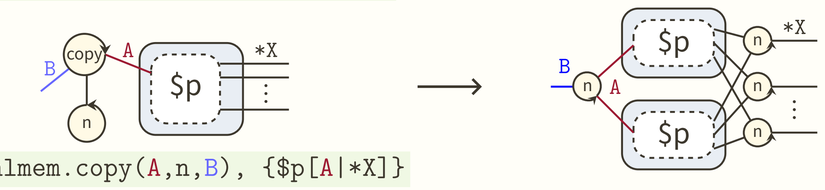


The membrane can be **copied** or **removed** while keeping bundle `*X`:

remove :



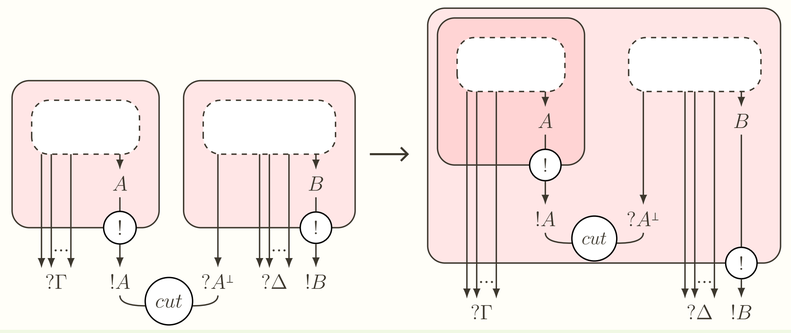
copy :



➔ **With $?\Gamma$ represented as `*X`, Promotion Box can be handled directly!**

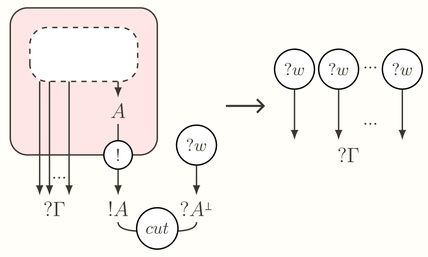
Encoding Cut Elimination Rules:

① Structure where multiple boxes are connected via a cut




```
cut_elimination_nested@@
{'!'(A,B),$p1[A|*X]},{$p2[C|*X]},cut{+(B),+(C)}
    :- {{'!'(A,B),$p1[A|*X]},$p2[C|*X],cut{+(A),+(B)}}.
```

② Structure where box is connected to weakening via a cut



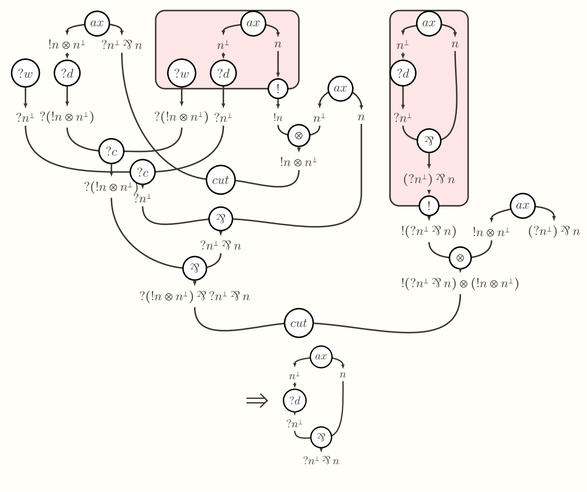

```
cut_elimination_weakening@@
{'!'(A,B),$p[A|*X]},'?w'(C).
cut{+(B),+(C)}
  :- nlmem.kill(
      {trash(A),$p[A|*X]},'?w'
     ).
```

All the other rules including contraction can be encoded in the same way.

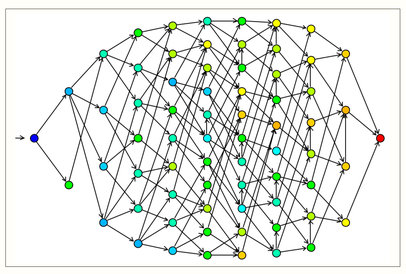
➔ **Cut elimination can be encoded concisely as an LMNtal program.**

Execution Example:

Proof Net:



State Space Visualization by LaViT (LMNtal Visual Tool):



➔ The state space is confluent and strongly normalizing.

4. Conclusion and Future Work

- ✓ We have shown that Membranes, Process contexts, and Bundles of LMNtal correspond directly to the operations required for the cut elimination of Proof Nets.
- ✓ We encoded the cut elimination rules in LMNtal concisely and tested them with various examples.
- ? Fine-grained encoding of cut elimination inspired by the scope management of [5] that uses membranes in a different manner.
- ? Adapting and extending the framework to a workbench for lambda calculus.

References

[1] J.Y. Girard. Linear logic. Theor. Comput. Sci., Vol. 50, No. 1, pp. 1–101, 1987.

[2] J.Y. Girard, Y. Lafont, and P. Taylor. Proofs and Types. No. 7 in Cambridge Tracts in Theoretical Computer Science. 1989.

[3] M. Gocho, T. Hori, and K. Ueda. Evolution of the LMNtal runtime to a parallel model checker. Computer Software, Vol. 28, No. 4, pp. 137–157, 2011.

[4] S. Guerrini, S. Martini, and A. Masini. Proof nets, garbage, and computations. Theor. Comput. Sci., Vol. 253, No. 2, pp. 185–237, 2001.

[5] K Ueda. Encoding the Pure Lambda Calculus into Hierarchical Graph Rewriting. In Rewriting Techniques and Applications, pp. 392–408, 2008.

[6] K. Ueda. LMNtal as a hierarchical logic programming language. Theor. Comput. Sci., Vol. 410, No. 46, pp. 4784–4800, 2009.