Critical Pairs in Network Rewriting

Lars Hellström
lars.hellstrom@residenset.net

Division of Applied Mathematics, Mälardalen University, Västerås, Sweden

International Workshop on Confluence (IWC),
FLoC/VSL 2014, Wien
Informal definition of network

- A term is the same as a tree (for suitable CS-style definition of tree).
- A word arises as the special case that the above tree is a path (open in both ends).
- A network is the generalisation where instead of a tree you allow a directed acyclic graph (DAG), symmetrically with respect to incoming and outgoing.
- Not the same as a term graph, since that treats incoming and outgoing edges at a vertex differently.
Example of network
Why networks?

• The intuition for a vertex with out-degree > 1 is that it is like a subroutine with more than one out-parameter.

• The intuition for a vertex with out-degree 0 is that it is like a subroutine has an effect on some global state.

• There are mathematical theories (bialgebras, Hopf algebras, quantum gate arrays, etc.) where operations with not exactly one out-parameter are essential parts of the core structure. Expressions for these theories cannot (easily) be expressed as terms, but they can in these cases be expressed as networks.

• Which is good, because I’d like to do universal algebra for some of these theories.
Networks as expressions to evaluate

- **Words** can be evaluated in a *monoid*.
- **Terms** can be evaluated in an $\Omega$-*algebra* (for appropriate signature $\Omega$).
- **Networks** can be evaluated in a *PROP/symmetric monoidal category*.

That an algebraic structure supports a network evaluation map may be taken as the definition of a PROP.

- Can drop the ‘acyclic’ condition on networks while keeping ability to evaluate by adding a ‘traced’ on ‘symmetric monoidal category’, but that loses many applications.
- There are other classes of monoidal categories, which corresponds to networks with various types of embeddings.
At first sight, it should be straightforward: replace part of a network (a subnetwork, playing the role of subexpression in this theory) by another network.

But there are two different candidates for the concept of subnetwork:

1. A *subnetwork is something that can be contracted to a vertex.*

   Supported by: free category formalisation, double pushout formalism (easiest formulation).

2. A *subnetwork is an arbitrary set of vertices (and edges between them)*, not just those that are convex.

   Supported by: abstract index notation formalisation, graph theory.
Network Rewriting

At first sight, it should be straightforward: replace part of a network (a subnetwork, playing the role of subexpression in this theory) by another network.

But there are two different candidates for the concept of subnetwork:

1. A subnetwork is something that can be contracted to a vertex.
   Supported by: free category formalisation, double pushout formalism (easiest formulation).

2. A subnetwork is an arbitrary set of vertices (and edges between them), not just those that are convex.
   Supported by: abstract index notation formalisation, graph theory.
Example nonconvex rewrite step

\[
\begin{bmatrix}
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
\end{bmatrix}
\]

applies

\[
\begin{bmatrix}
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
  \square & \circ & \square & \square \\
\end{bmatrix}
\rightarrow [\times]: \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}
\]
Transferrence

Nonconvex rules introduce the problem of how to preserve acyclicity.

**Definition**
The transferrence of an $n$-in-$m$-out network is an $m \times n$ boolean matrix. There is a 1 in position $(i, j)$ iff there is a directed path in the network from the $j$th input to the $i$th output.

Knowing the transferrences of all involved networks and subnetworks make it possible to tell whether a particular replacement would create a cycle or not.
Network rewrite rules

A network rewrite rule is defined by three pieces of data:

- A nominal transferrence \( q \).
  For derived rules created during completion, this is usually the minimal transferrence at which the rule can be derived.

- A left hand side \( l \), which is a network whose transferrence may not exceed \( q \).

- A right hand side \( r \), which is a network whose transferrence may not exceed \( q \).

Rules operate on networks tagged with a nominal transferrence \( p \) (which may be larger than their actual transferrence, but not smaller).

A redex requires (i) that the left hand side is matched as a subnetwork and (ii) that putting the nominal transferrence \( q \) into the actual context does not result in a combined transferrence that exceeds \( p \).
Network rewrite rules

A network rewrite rule is defined by three pieces of data:

- A **nominal transferrence** $q$.
  For derived rules created during completion, this is usually
  the minimal transferrence at which the rule can be derived.

- A **left hand side** $l$, which is a network whose transferrence
  may not exceed $q$.

- A **right hand side** $r$, which is a network whose transferrence
  may not exceed $q$.

Rules operate on **networks tagged with a nominal transferrence** $p$ (which may be larger than their actual transferrence, but not smaller).

A redex requires (i) that the left hand side is matched as a
subnetwork and (ii) that putting the nominal transferrence $q$
into the actual context does not result in a combined
transferrence that exceeds $p$. 
Ambiguities and shadows

Any time you have a network $\mu$ of nomimal transferrence $p$, and two rules $l_1 \rightarrow r_1 : q_1$ and $l_2 \rightarrow r_2 : q_2$ that can act on it, you have an ambiguity. $\mu$ is called the site of the ambiguity.

An ambiguity $A_1$ is a shadow of an ambiguity $A_2$ if $A_1$ is just $A_2$ placed into some extra context. Every sequence of rewrite steps that resolve $A_2$ can be mechanically rewritten to a sequence that resolve $A_1$. 
No extra vertex

If the site of an ambiguity contains a vertex that is not part of either redex, then that vertex *can be deleted*, i.e., that ambiguity is a shadow of an ambiguity whose site does not have that vertex.

This relies heavily on being able to consider ambiguities whose transferrence is not all-ones.
No extra edge, and edges can be cut

- If an edge in the site of an ambiguity is not part of either redex, then that edge can be deleted (in the same sense as on previous slide).

- If an edge in the site of an ambiguity is part of a redex, but neither redex covers the full length of the edge, then that edge may be cut; the ambiguity is a shadow of an ambiguity where the separate redex segments belong to separate legs of the network.

These conditions put bounds on the size of the site of an interesting ambiguity formed by two given rules, so we know that there are always finitely many.
Intuition from word rewriting is that the only types of ambiguities that need resolving are overlaps and inclusions. What is missing from the above is to eliminate ambiguities where the redexes are disjoint (no vertex or edge in common).

Definition
A rule is **sharp** if its nominal transferrence is equal to the transferrence of its left hand side.

An ambiguity formed by two **sharp** rules where the redexes are disjoint is always resolvable.
But for non-sharp rules it need not be!
Example

If applying all criteria from above, the example site should be

\[
\begin{bmatrix}
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0
\end{bmatrix}
\]

but it is easier to look at the edges-not-cut version

\[
\begin{bmatrix}
1 & 1 \\
1 & 1 \\
1 & 1 \\
1 & 1
\end{bmatrix}
\cdot
\begin{pmatrix}
1 \\
1
\end{pmatrix}
\]
Example (cont.)

Given rules

\[
\begin{align*}
\begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} & \xrightarrow{s_1} \begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} : \left(\begin{array}{cc} 1 & 1 \\ 1 & 1 \end{array}\right), \\
\begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} & \xrightarrow{s_2} \begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} : \left(\begin{array}{cc} 1 & 1 \\ 1 & 1 \end{array}\right)
\end{align*}
\]

there is the ambiguity

\[
\begin{align*}
\begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} & \xleftarrow{s_1} \begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} = \begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} = \begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix} & \xrightarrow{s_2} \begin{bmatrix}
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{bmatrix}
\end{align*}
\]

where the leftmost network no longer has a redex for \( s_2 \) (and vice versa), even though it would have a redex for a different rule with the same left hand side but transferrence \( \left(\begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array}\right) \).
Future work

- Sometimes I believe (and other times I doubt) that if one could just refine the transferrence system further then I could get back to just overlaps and inclusions. Which would be nice.

But that last example is still exhibiting a nontrivial consequence of the rules, so there would have to be some kind of critical pair here that needs to be examined.

- Either way, some method of enumerating all critical ambiguities of this new variety is needed. (I have a program that does completion on systems of network rewrite rules, but at the moment it only considers overlaps and inclusions, not wraps.)
Future work

• Sometimes I believe (and other times I doubt) that if one could just refine the transferrence system further then I could get back to just overlaps and inclusions. Which would be nice.

But that last example is still exhibiting a nontrivial consequence of the rules, so there would have to be some kind of critical pair here that needs to be examined.

• Either way, some method of enumerating all critical ambiguities of this new variety is needed. (I have a program that does completion on systems of network rewrite rules, but at the moment it only considers overlaps and inclusions, not wraps.)
Future work

- Sometimes I believe (and other times I doubt) that if one could just refine the transferrence system further then I could get back to just overlaps and inclusions. Which would be nice.

But that last example is still exhibiting a nontrivial consequence of the rules, so there would have to be some kind of critical pair here that needs to be examined.

- Either way, some method of enumerating all critical ambiguities of this new variety is needed.
  (I have a program that does completion on systems of network rewrite rules, but at the moment it only considers overlaps and inclusions, not wraps.)
More reading


Network rewriting program homepage: http://www.mdh.se/ukk/personal/maa/lhm03/sw/rewriting

Thank you for your attention!