Quine graphs

Marius Buliga

Institute of Mathematics, Romanian Academy
P.O. BOX 1-764, RO 014700
Bucureşti, Romania
Marius.Buliga@imar.ro , mbuliga@protonmail.ch

31.10.2019

Abstract

Given a graph rewrite system, any graph has at least one maximal collections of non-conflicting matches (MNCM) of left patterns of graphs rewrites. A graph G is a quine graph if it has a non-void MNCM such that after the rewrites from the collection are done in parallel we obtain a graph isomorphic with G.

In this note we introduce quine graphs. We invite you to explore some of their properties, with examples taken from lambda calculus, interaction combinators and chemlambda.

1 Patterns, mols, graph rewrite systems and quines

We are concerned with graphs which have a finite number of nodes. Nodes are connected with edges via node ports. The valence of a node is the number of its node ports. Each edge connects two different node ports. Every node port is connected to an edge. Each node has a type from the list $NodeTypes$ of node types. The type $A \in NodeTypes$ of a node gives the valence $ar(A)$ of the node, as well as a numbering of the node ports (from 0 to $ar(A) - 1$).

Any graph made of a finite number of nodes, each node of a type from the list of node types, admits a mol notation. We need an infinite alphabet $Edge$ of symbols which we use to decorate the edges of the graph, such that every two different edges have different decorations. We arbitrarily order the nodes of the graph. Then we produce the mol notation of the graph which is a list of node items, one for each node. Each item is a list which starts with the node type, then with the edge decorations of each edge incident to a node port, in the order of the node ports.

For example the list:

$$((A, a, b, c), (A, b, d, e), (B, c, a, d), (A, e, f, f))$$

describes a graph with 4 nodes, 3 of them of type A and one of type B. All nodes have valence 3. There are 6 edges, decorated with the symbols a,b,c,d,e,f. (By using the order of the nodes given by the list) we know that the edge "a" connects the 1st port of the first node with the 2nd port of the second node, and so on until the last edge, "f", which connects the 2nd and 3rd ports of the 4th node.

Further we shall write node items and node words in the following format: one node item per line, without parantheses or commas, like this

$$\begin{array}{cccc} A & a & b & c \\ A & b & d & e \\ B & c & a & d \\ A & e & f & f \end{array}$$

Notice that each edge decoration appears exactly twice.

The mol notation of a graph is unique up to the renaming of the edges (decorations) and up to permutation of the node items. Similarly, two graphs are isomorphic if they admit the same mol notation, again up to renaming of the edges and up to permutation of the node items.

A pattern is a collection of nodes of the graph, together with the internal edges (joining the nodes of the pattern) and with their external half-edges, but seen in the same mol notation. For the example of a graph taken previously, any collection of lines in the mol notation is a pattern, like this one:

$$\begin{array}{cccc} A & a & b & c \\ B & c & a & d \end{array}$$

In the mol notation of a pattern, each edge decoration appears at most twice. Those edge decorations which appear once are the external half-edges.

A match of a pattern into a graph (described by its mol notation) is a pair of functions (f, g), where g is an injective function from the nodes (lines in the list) of the pattern to the nodes of the graph, and f is a renaming of the edge decorations, such that on those those decorations which appear twice in the pattern the renaming is injective, and on the external half-edges the renaming is at most 2-to-1.

For example if we take the pattern:

$$\begin{array}{cccc} A & 1 & 2 & 3 \\ A & 3 & 4 & 5 \end{array}$$

then there is a match with the graph taken as an example:

$$\begin{array}{cccc} A & 1 & 2 & 3 \\ A & 3 & 4 & 5 \end{array} \quad \rightarrow \quad \begin{array}{cccc} A & a & b & c \\ \mathbf{A} & \mathbf{b} & \mathbf{d} & \mathbf{e} \\ B & c & a & d \\ \mathbf{A} & \mathbf{e} & \mathbf{f} & \mathbf{f} \end{array}$$

Indeed, here the function g matches the 1st line of the pattern with the 2nd line of the mol notation of the graph and the 2nd line of the pattern with the 4th line of the mol notation of the graph. The function f is defined as:

$$f(1) = b \,.\, f(2) = d\, f(3) = e\, f(4) = f(5) = f$$

A graph rewrite $LHS \rightarrow RHS$ is defined by two patterns with the same external half-edges, called the LHS and the RHS patterns of the rewrite. For our example suppose that we have the following graph rewrite:

$$\begin{array}{cccc} A & 1 & 2 & 3 \\ A & 3 & 4 & 5 \end{array} \quad \rightarrow \quad \begin{array}{ccc} C & 1 & 2 \\ C & 4 & 5 \end{array}$$

In the usual style of DPO rewrites (adapted to the mol notation) the application of the graph rewrite is a 3 parts process, where:

- we start from a match of the LHS pattern into the graph, for example the one considered previously,

- we produce an intermediary mol notation by deleting the lines in the mol notation of the graph which are in the scope of the match thus obtaining

$$\begin{array}{cccc} A & a & b & c \\ B & c & a & d \end{array}$$

- we concatenate the intermediary mol notation with a copy of the RHS pattern, such that all the internal edges have fresh decorations (not the case in our example, where the RHS does not have internal edges) and such that the external edges inherit the decoration from the match of the LHS pattern. In our example we get:

$$\begin{array}{llll} A & a & b & c \\ B & c & a & d \\ \mathbf{C} & \mathbf{b} & \mathbf{d} & \\ \mathbf{C} & \mathbf{f} & \mathbf{f} & \end{array}$$

In order to allow graphs with some free edges we suppose that we always have in $NodeTypes$ 1-valent nodes named "FR*" where "*" is a word of our choice. For example a pattern which has external half-edges, like

$$\begin{array}{llll} A & a & b & c \\ A & c & d & e \end{array}$$

could be seen as the mol notation of a graph with free half-edges a, b, d, e, by adding to it 4 FR* nodes, like this:

$$\begin{array}{llll} A & a & b & c \\ A & c & d & e \\ FRIN & a & & \\ FRIN & b & & \\ FROUT & c & & \\ FROUT & d & & \end{array}$$

so that now each edge decoration appears exactly twice. This completion of a pattern to a mol notation does not erase the difference between mols and patterns though.

There are two kinds of graph rewrites which we want to allow. We want to allow parallel graph rewrites which don't share nodes, but they do share half-edges. Also we want to allow graph rewrites which delete nodes. That is why we suppose that we always have in $NodeTypes$ 2-valent nodes named "Arrow*". Moreover we shall always add to the graph rewrite systems considered a family of graph rewrites called "COMB", which consist into the elimination of the Arrow* elements. For example suppose we have a graph rewrite which deletes a pair of neighboring nodes and joins the dangling half-edges. This can be done by a graph rewrite of the form

$$\begin{array}{llll} A & 1 & 2 & 3 \\ B & 3 & 4 & 5 \end{array} \quad \rightarrow \quad \begin{array}{lll} Arrow & 1 & 5 \\ Arrow & 4 & 2 \end{array}$$

This allows to apply several such rewrites in parallel, without confusion concerning the wiring of the edges. After the rewrites are done (one or more in parallel), we end up with the following two situations. A node connected to an Arrow*, or an Arrow* with both ports connected to the same edge. THe COMB rewrites are then used to erase the Arrow* elements. For any $A \in NodeTypes$

$$\begin{array}{llll} A & ... & e_i & ... \\ Arrow* & e_i & b & \end{array} \quad \rightarrow \quad A \quad ... \quad b \quad ...$$

and

$$Arrow* \quad a \quad a \qquad \rightarrow \emptyset$$

In the case of sequential application of rewrites, we may always absorb the COMB rewrites into the particular graph rewrites considered.

Finally, a graph is a quine (graph), relative to a graph rewrite system, if it has a maximal collections of non-conflicting matches (MNCM) of left patterns of graphs rewrites, with the property that after we apply the graph rewrites in parallel, and after we eliminate all the Arrow* elements by COMB rewrites, we end up with the same (i.e. isomorphic) graph.

2 Detailed description

Let $NodeTypes$ be a non empty finite set and $ar : NodeTypes \to \mathbb{N}$ a function called arity. Let $Edge$ be an infinite set. A node item is a list

$$N = (A, e_0, ..., e_{ar(A)-1}) \, , \, A \in NodeTypes \, , \, e_0, ..., e_{ar(A)-1} \in Edge$$

Let $NodeItems$ be the set of node items and the set of node words $NodeItems^*$ is the set of words over the alphabet $NodeItems$.

The function $type : NodeItems \to NodeTypes$ gives the type of a node item:

$$type((A, e_0, ..., e_{ar(A)-1})) = A$$

and the function $occ : NodeItems^* \to (Edge \to \mathbb{N})$ gives the number of occurences of an element $e \in Edge$ in the word $w \in NodeItems^*$. This function is defined as:

- $occ(\emptyset)(e) = 0$ for any $e \in Edge$, where $\emptyset$ is the empty word,
- $occ(w_1 w_2)(e) = occ(w_1)(e) + occ(w_2)(e)$ for any $e \in Edge$ and $w_1, w_2 \in NodeItems^*$
- for any node item $N = (A, e_0, ..., e_{ar(A)-1})$ and for any $e \in Edge$ we have

$$occ(N)(e) = \sum_{i=0}^{ar(A)-1} \chi_e(e_i)$$

Here $\chi_e : Edge \to \{0, 1\}$ is the characteristic function $\chi_e(a) = 1$ if and only if $e = a$.

For a non-empty node word w the set $Occ(w) \subset Edge$ is made of all $e \in Edge$ such that $occ(w)(e) \neq 0$. Also, for any $n \in \mathbb{N}$ we define

$$Occ^n(w) = \{e \in Edge \mid occ(w)(e) = n\}$$

The length $\mid w \mid$ of a node word $w \in NodeItems^*$ is simply the length of it as a word over the alphabet $NodeItems$.

We can define now the category $NodeWords(NodeTypes, ar, Edge)$ of node words which has $NodeItems^*$ as objects. Consider two arbitrary non-empty node words $w_1 = N_1 ... N_p$ and $w_2 = M_1 ... M_q$, where

$$N_i = (A^i, e^i_0, ..., e^i_{ar(A^i)-1})$$

A morphism from w_1 to w_2 is a pair (f, g) of functions, where

- $f : Occ(w_1) \to Occ(w_2)$ and $g : \{1, ..., \mid w_1 \mid\} \to \{1, ..., \mid w_2 \mid\}$
- $M_{g(i)} = (A^i, f(e^i_0), ..., f(e^i_{ar(A^i)-1}))$.

This category has two sub-categories which are of interest to us. The first one is the category $Patterns(NodeTypes, ar, Edge)$ of patterns. It has as objects those node words w with the property that for any $e \in Edge$

$$occ(w)(e) \leq 2$$

It follows that for a pattern w we have

$$Occ(w) = Occ^1(w) \cup Occ^2(w)$$

We name the elements of $Occ^1(w)$ the external edges and the elements of $Occ^2(w)$ the internal edges.

The morphisms between two patterns are only those node word morphisms (f, g) such that:

- g is an injective function,

- f is injective on $Occ^2(w)$ and at most 2-to-1 on $Occ^1(w)$.

The category of mols (from "molecules") $Mol(NodeTypes, ar, Edge)$, has as objects those node words w with the property that for any $e \in Edge$

$$occ(w)(e) \in \{0, 2\}$$

Thus a mol is a pattern without external edges. The morphisms between two mols are the morphisms of patterns.

To the isomorphism class $[w]$ of a mol w in $Mol(NodeTypes, ar, Edge)$ is associated a graph. Indeed, for a mol w the graph has $\mid w \mid$ nodes, where the node corresponding to the node item (letter in w) $(A, e_0, ..., e_{ar(A)-1})$ has the type A, valence $ar(A)$ and an ordered collection of node ports. The set of edges of the graph may be taken as $Occ(w)$ and an edge $e \in Occ(w)$ connects the node port a of the node (corresponding to the node item) $(A^1, e^1_0, ..., e^1_{ar(A^1)-1})$ with the node port b of the node (corresponding to the node item) $(A^2, e^2_0, ..., e^1_{ar(A^2)-1})$ if

$$e^1_a \, = \, e^2_b \, = \, e$$

Each edge of the graph is well defined, because any $e \in Occ(w)$ connects exactly two node (ports). Two isomorphic mols define the same graph, up to a permutation of nodes and up to a renaming of edges.

In order to allow for mols with free half-arrows, as well as to easily be able to perform some rewrites, we shall assume that $NodeTypes$ always contains one or more "free" nodes, denoted "FR*" (where "*" may be a word of choice), with arity 1, as well as one or more "arrow" nodes, denoted "Arrow*", with arity 2.

By using the FR* nodes we can always transform a pattern into a mol, simply by concatenating the pattern word w with an word $'$ made of node items of the form $(FR*, e)$ where $e \in Occ^1(w)$. This transformation does not respect pattern morphisms.

A match of a pattern w in a pattern m is a pattern morphism from w to m. A node item of m is in the scope of the match if it is of the form $N_{g(i)}$ for an $i \in \{1, ..., \mid w \mid\}$.

A graph rewrite is defined in a similar way to the usual DPO rewrites. More precisely, in our setting, a graph rewrite is defined by a pair of patterns, notation $w^{left} \rightarrow w^{right}$ which have the same external edges

$$Occ^1(w^{left}) = Occ^1(w^{right})$$

Then the graph rewrite application to a mol w consists into:

- the identification of a match $(f, g) : w^{left} \rightarrow w$
- the deletion of node items of w which are of the form $N_{g(i)}$ with $i \in \{1, ..., \mid w^{left} \mid\}$, which gives an intermediary pattern w',
- the identification of an morphism of patterns $(f', g') : w^{right} \rightarrow w$" such that g is surjective, $Occ(w') \cap Occ^2(w") = \emptyset$ and for any $e \in Occ^1(w^{right})$ we have $f'(e) = f(e)$
- concatenation of w" to w', which produces the new mol $w'w$".

The graph rewrite is well defined. Given a mol w, a graph rewrite and a match of w^{left} in w, the result of the rewrite is unique up to isomorphism.

A graph rewrite system GRS is a finite collection of graph rewrites. We shall always suppose that our GRS contains the following family of rewrites called "COMB", which involves Arrow* node types. For any $A \in NodeTypes$

$$\begin{array}{llll} A & ... & e_i & ... \\ Arrow* & e_i & b & \end{array} \quad \rightarrow \quad A \; ... \; b \; ...$$

and

$$Arrow* \quad a \quad a \qquad \rightarrow \quad \emptyset$$

Consider a mol w. A collection $\mathcal{C}$ of matches of left patterns of rewrites in GRS on w, not containing any COMB rewrite, is non conflicting if any node item of w is in most one scope of these matches. Because w has a finite number of node items, it follows that there are maximal collections of non-conflicting matches (MNCM). We can therefore perform all the (matched) graph rewrites in parallel, followed by a sequence of COMB rewrites which eliminates all Arrow* elements. The result is called the derived mol

$$w' = \mathcal{C}(w)$$

If w' obtained after performing these rewrites is isomorphic with w then we call w a 1-quine (and the graph corresponding to $[w]$ is called a quine graph or a 1-quine graph).

More generally, for a positive number n, an n-quine (relative to a GRS) is a mol w for which there is a sequence of n pairs $(w_i, \mathcal{C}_{i+1})$, for $i = 0$ to $n-1$ such that:

- w_0 is isomorphic with w
- w_i is a mol and $\mathcal{C}_{i+1}$ is a MNCM of w_i
- for any $i < n-1$ $w_{i+1} = \mathcal{C}_{i+1}(w_i)$
- $\mathcal{C}_n(w_{n-1})$ is isomorphic to w.