University of Glasgow | School of Computing Science

RPOs in Place Graphs of Epimorphic Bigraphs with sharing *

W. David Frohlingsdorf

10. September 2016

Abstract

The Bigraphical Reactive System (BRS), introduced by Robin Milner, can be used to model systems with locality and connectivity. Bigraphs with sharing is an extension of the BRS and allows agents to share locations. This is achieved by a redefinition of the underlying spatial model. The sharing concept is a very useful extension, because in many systems -for example social interactions or wireless signals- the spatial location of agents can overlap.

In his work Milner lined out why it is important to find relative pushouts (RPOs) in order to spot possible conflicts or potential reactions. Furthermore he outlined an algorithm to find RPOs in bigraphs. However, the algorithm can not be applied to bigraphs with sharing due to the changed spatial model. The aim of this project was therefore to introduce and implement an algorithm to find RPOs in the spatial model of epimorphic bigraphs with sharing as well as proving its correctness.

*This report is part of a 10-week summer internship at the University of Glasgow. Supervised by Dr. Michele Sevegnani, funded by the EPSRC.

Contents

1 Introduction

In a more and more connected world it is important to have models which can capture connections and positions of agents in a complex system, as well as changes based on statistical methods. For this reason Robin Milner introduced the idea of bigraphs to capture the state of a system at any given time. Moreover Milner introduced the Bigraphical Reactive System (BRS) as well as the refinement, Stochastical Bigraphical Reactive System (SBRS), to model bigraph system state changes based on reaction rules and in case of the SBRS also on probabilities [2].
Bigraphs are structures which consist of nodes which may contain other nodes or sites (abstracted part of the bigraph) and are themselves inside a parent node or root. Moreover, nodes can be connected to another with hyperedges. Hence, a bigraph consists of a place graph which captures the position of nodes, as well as a link graph which captures the connections of the nodes. The structures used for place and link graphs are a forest and a hypergraph respectively.
This model, however, can not capture systems with overlapping spatial positions (for example wireless networks or the like). For this reason Michele Sevegnani introduce bigraphs with sharing which is an improvement of the bigraphs introduced by Milner as it allows agents to share positions [3][4]. This new property is achieved by changing the underlying place graph from a forest to a directed acyclic graph (DAG). There is no need to change the link graph, as the linking capacity of nodes is not affected by the change. On the downside, however, operations on standard bigraphs can not directly be applied to bigraphs with sharing as the underlying structure is fundamentally different. For this reason all operations have to be redefined for bigraphs with sharing. This project was about the redefinition of one specific operation, namely to find relative pushouts (RPOs) in place graphs of epimorphic bigraphs with sharing. This operation is useful for finding possible conflicts of reaction rules as well as potential reactions in the BRS.
Robin Milner lined out an algorithm to find RPOs in link and place graphs in his work. First, we will review this algorithm in Section 2. In the next step we will introduce introduce an algorithm to find RPOs in epimorphic place graphs with sharing in Section 3 (note that Milner's link graph algorithm can be used for standard as well as bigraphs with sharing the like). In Section 4 and 5 we will respectively show an example of the working algorithm and line out a pseudo code of the algorithm. A mathematical proof of the algorithm's correctness can be found in Appendix A During the course of this project we also implemented the outlined pseudo code in the BigraphER tool [5] which has been invented and is maintained by Michele Sevegnani. Finally, we will close this paper with a discussion of RPOs in general place graphs with sharing in Section 6.

2 Review of Milner's Algorithm

Robin Milner gives in [2](Construction 5.9) an outline of an algorithm which can be used to find RPOs in place graphs of standard bigraphs (without sharing). In this section we will simply review his algorithm and reword some of his formulations to improve clarity. The reader is advised to use Figure 1 as a reference point.

2.1 Notation

The number of the roots of A_0 and A_1 (that is $A_i (i = 0, 1)$) are the two ordinals (natural numbers) m_i. To iterate over those roots we will use r_i.
V_0 and V_1 denote the sets of nodes of A_i.
$V_2 = V_0 \cap V_1$ is the set of all nodes in common between A_0 and A_1.
We will use w to iterate over $h \uplus V_2$. That is, to iterate over the -shared- sites of A and shared nodes (but obviously not the roots).
We use $A(w)$ to denote $prnt_A(w)$ that is, to receive the parent of w in A (equally $D(w)$).

2.2 Overview - How to built an RPO

In a concrete place graph 'PG; if there is a bound such that: $\overrightarrow{A} : h \rightarrow \overrightarrow{m}$ (span), and $\overrightarrow{D} : \overrightarrow{m} \rightarrow p$ (cospan), one can built a RPO ($\overrightarrow{B} : \overrightarrow{m} \rightarrow \hat{m}, B : \hat{m} \rightarrow p$) with the following steps.

2.2.1 Recall: Relative Pushout -RPO-

This section contains a very informal definition of an RPO.

A RPO is a relative bound plus an arrow to some bound. It therefore consists of three arrows -pushout-. What makes the RPO special is the fact that **any** other relative bound to the given bound can be "reached" half way from the RPO with a **unique** arrow -j-. That means, the RPO is the closest cut-off of the respective bound that is anyhow possible (called a minimal bound). Therefore, it does not give anything away and no other relative bound can cut closer to the bound, hence they can **all** be reached via the RPO (see Figure 1)[2, 1].

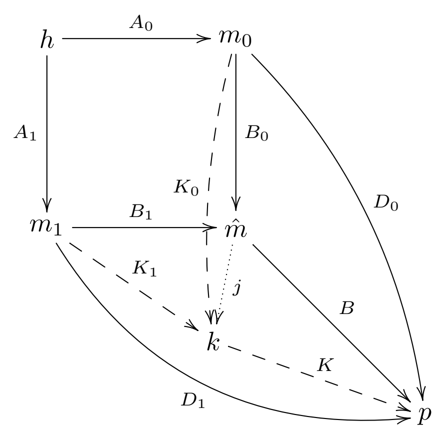


Figure 1: RPO for the bound $\overrightarrow{A}$, $\overrightarrow{D}$ consisting of bigraphs B_0, B_1 and B. Any other relative bound can be reached with a unique morphism -j-.

2.3 Nodes

Let V_3 be the set of nodes which are not in A_0 nor in A_1 and are therefore in both of D_i. In terms of nodes, the set V_3 should only be added to the bigraph in the last step, that is in

B, to ensure a minimal bound $\overrightarrow{B}$. The arrows B_i only add the missing nodes to bring the bound to $V_0 \cup V1$. Note that $(V_0 \cup V_1) \cap V_3 = \emptyset$ are disjoint sets.

2.3.1 Equivalence Class

The equivalence class denotes the single parent site is connected to. We are particularly interested in the equivalence class of the sites in m_i which are connected to V_3 or a root in p. That means if two sites $r_0 \cong r_1$ (they belong to the same $\cong$-equivalence class), then they have the same parent in $\overrightarrow{D}$ and this parent is in $V_3 \uplus p$.

2.4 Interfaces

Now consider the object $\hat{m}$ (that is the roots in common in arrows B_0 and B_1). Essentially $\hat{m}$ is a subset of the roots/sites of $m_0 \cup m1$. In particular: Take all the sites in m_i which have as their parent in D_i either a node of the set V_3 or a root of p (these are the ones which can be directly mapped into $\hat{m}$ as V_3 is only touched in bigraph B).

$$\text{Formal: } m_i' := \{r \in m_i | D_i(r) \in V_3 \uplus p\}$$

Before we can decide on $\hat{m}$ we need to find out how many $\cong$-equivalence classes there are. Iterate through all shared places such that $w \in h \uplus V_2$. Whenever the shared place's parent is a root in **both** bigraphs ($A_0(w) = r_0$ and $A_1(w) = r_1$), then the two roots are in the same equivalence class which is denoted by: $(0, r_0) \cong (1, r_1)$. Now, this is in particular interesting when $r_0 \in m_0'$ and $r_1 \in m'1$. With this at hand we can define up to isomorphism (see also: Discussion 3.4.1 and Proposition A.7):

$$\hat{m} := (m_0' + m_1')/\cong$$

In words, $\hat{m}$ consists of all $\cong$-equivalence classes of the disjoint sum $m_0' + m_1'$, that is their roots.
For a site $r \in m_i'$ we can denote its $\cong$-equivalence class as $\widehat{i,r}$.

2.5 Parents

So far we have got the sites from B_0 (m_0) and B_1 (m_1), the interfaces $\hat{m}$ (that is, the common roots of B_0 and B_1, which are equal to the sites of B) and of course p, as well as three sets of nodes assigned to the bigraphs $\overrightarrow{B}$ (i.e. B_0, B_1) and B.
What is left to do is to define parents for the sites and nodes as follows:
For B_0:

$$\text{For } r \in m_0 \text{ -the roots of } A_0\text{-:}$$

$$B_0(r) := \begin{cases} \text{if } r \in m_0'\text{: } \widehat{0,r} \\ \text{else: } D_0(r) \end{cases}$$

$$\text{For } v \in V_1/V_2 \text{ -the missing nodes of } A_0\text{-:}$$

$$B_0(v) := \begin{cases} \text{if } A_1(v) = r \in m_1\text{: } \widehat{1,r} \\ \text{else: } D_0(r) \end{cases}$$

The parents of B_1 are equally created with the according 0s and 1s "flipped over".
Finally, we can define the parents in B which simulates the common part of D_0 and D_1:

$$\text{For } \hat{r} \in \hat{m} \text{ -the sites of } B \text{ from interface } \hat{m}\text{-:}$$

$$B(\hat{r}) := D_i(r) \text{ where } \widehat{i,r} = \hat{r}$$

$$\text{For } v \in V_3\text{:}$$

$$B(v) := D_i(v)$$

Note that in the node case i could either be 0 or 1. The result, however, is the same and does not depend on i as B represents the common part of D_0 and D_1.

3 RPOs in epimorphic Bigraphs with sharing

3.1 Overview

We will now introduce an algorithm to find RPOs in epimorphic place graphs with sharing. As proved by Szmajdjuch [6] it is possible to find RPOs for all epimorphic bigraphs with sharing.
The outlined algorithm by Milner for standard bigraphs can not directly be used for bigraphs with sharing. This is due to the fact that Milner's algorithm is largely based on parent relations which are ambiguous in bigraphs with sharing. Therefore we will focus on rewriting the algorithm where this is necessary. Note that link graphs are identical in bigraphs with and without sharing. The RPO algorithm for link graphs outlined by Milner (Construction 5.5 [2]) can therefore be directly used for link graphs of bigraphs with sharing. In this paper we will therefore only discuss algorithms for place graphs of bigraphs with sharing. The algorithm outlined in this section can only be used for epimorphic bigraphs with sharing, but we will discuss the general case at the end of this paper in Section 6.

3.2 Notation

Hereinafter, we will use the notation $D_i^{\{\}}(v)$ to denote the set of all parents of node v in D_i (to avoid confusion with $D_i(r)$ which denotes a -single- parent in standard bigraphs).
Moreover we define

$$M := V_3 \uplus p$$

for convenience.
With the notation $A \ni r$ we mean: Set A contains an r. Furthermore, we will use $\bar{\imath}$ to denote the counterpart of i. Specifically, $i = 0 \Rightarrow \bar{\imath} = 1$ and $i = 1 \Rightarrow \bar{\imath} = 0$. With $|B|$ we mean the support, that is the set of nodes, in bigraph B.

3.3 Nodes

Milner's algorithm can be used without any changes for epimorphic bigraphs with sharing with regard to nodes as briefly outlined in section 2.3.

3.4 Interface

Definition 3.1. Just like in standard bigraphs the interface $\hat{m}$ is a subset of the interfaces $m_0 \cup m1$. For this we first have to find the disjoint sum $m_0' + m_1'$. We can do this by reducing the interfaces m_0 and m_1. In particular, if a shared place $w \in h \uplus V_2$ has a root as parent in **one, but not both** A_0 and A_1 (note that a node can not have more than one parent as we are in the class of epimorphic bigraphs -$\widetilde{SPg^e}(\mathcal{K})$-), then we know that this root can not be in $\hat{m}$, because w has already the complete set of parents in the corresponding other graph of A. We can therefore reduce m_i by this root. Since a reduction of m_i changes the interface, we have to iterate this process until no further changes of m_i have occurred. The reduced interfaces of m_i correspond to m_i'.
Having the interfaces m_0' and m_1' defined, we can now continue by defining $\cong$-equivalence classes over m_i'. We do so by iterating through all the shared places $w \in h \uplus V_2$. If a shared places has a root of m_i' as a parent in A_0, then it must also have one in A_1 (by the definition of m_i') and those roots are furthermore $\cong$-equivalent. We denote $\cong$-equivalence of two roots r_0 and r_1 with $(0, r_0) \cong (1, r_1)$. Roots of the same $\cong$-equivalence class form the same root in $\hat{m}$ which is denoted by $\widehat{i, r}$ for any root (i, r) belonging to this class. Note also that more than two roots can belong to a $\cong$-equivalence class (see example Section 4).

Formally:

$$m'(m) := \begin{cases} \text{if } A_i^{\{\}}(w) \ni r_i \wedge A_{\bar{1}}^{\{\}}(w) \not\ni r_{\bar{1}} \\ \ |w \in h \uplus V_2 \wedge r_i \in m_i \wedge r_{\bar{1}} \in m_{\bar{1}} \wedge (i = 0 \vee 1) \\ \ :\ m'((m_i - r_i) + m_{\bar{1}}) \\ \text{else } m \end{cases}$$

We denote the formal definition of interface $\hat{m}$, up to isomorphism, as:

$$\hat{m} := (m_0' + m_1')/ \cong$$

3.4.1 Discussion: Roots from $\cong$-quivalence classes are unique, up to isomorphism

The order of the roots in $\hat{m}$ is not of importance for the algorithm. This is because the definition of the interfaces and the parent relation is only based on the $\cong$-equivalence class and not the concrete position of the roots in $\hat{m}$. Moreover, note that the morphism of the RPO pointing to another relative bound (which might equally be a RPO), j, can change the order of the roots accordingly -also known as permutation-, if required. To be consistent with Milner's terminology we denote this property with "up to isomorphism" which means that the ordinal number or label assigned to each root in $\hat{m}$ is not unique and therefore freely interchangeable.
On the other hand, however, for the graphical notation it would be nice to have a root order which guarantees maximal clarity.

3.5 Parents

The connection to the parents is fairly self-explanatory. We will first give the formal definitions and then give the definition in words.

Definition 3.2. For $r \in m_0$ -the roots of A_0-:

$$B_0^{\{\}}(r) := \begin{cases} \text{if } r \in m_0' \text{: } \widehat{0,r} \\ \text{else: } \emptyset \end{cases} \quad \uplus (D_0^{\{\}}(r)/M)$$

For each site $r \in m_0$: connect to all the parent nodes as in D_0 apart from the shared nodes V_3 and also connect to a root of $\hat{m}$ if $r \in m_0'$.

For $v \in V_1/V_2$ -the missing nodes of A_0-:

$$B_0^{\{\}}(v) := \begin{cases} \text{if } A_1^{\{\}}(v) \ni r | r \in m_1 \text{: } \widehat{1,r} \\ \text{else: } \emptyset \end{cases} \quad \uplus (D_0^{\{\}}(r)/M)$$

For each node $v \in V_1/V_2$: connect it to all its parents which are not in $V_3 \uplus p$ and if v had a parent r in A_1 such that r is a root, connect v also to the corresponding root in $\hat{m}$ which covers the $\cong$-equivalence class $\widehat{1,r}$.
The parents of B_1 are equally created with the according 0s and 1s "flipped over".
Finally, we can define the parents in B which simulates the common part of D_0 and D_1:

For $\hat{r} \in \hat{m}$ -the sites of B from interface $\hat{m}$-:

$$B^{\{\}}(\hat{r}) := D_i^{\{\}}(r) \cap M \text{ where } \widehat{i,r} = \hat{r}$$

For $v \in V_3$:

$$B^{\{\}}(v) := D_i^{\{\}}(v)$$

4 Example of working algorithm

We will now illustrate how the introduced algorithm works with a concrete example.

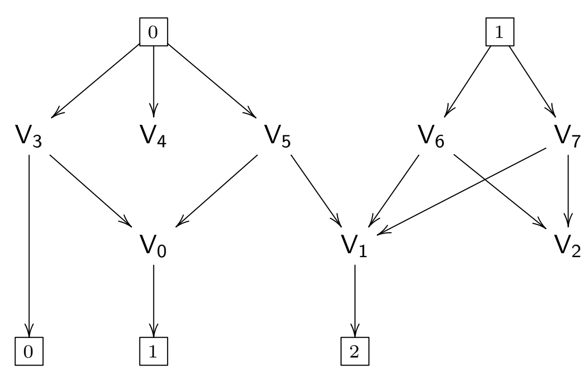
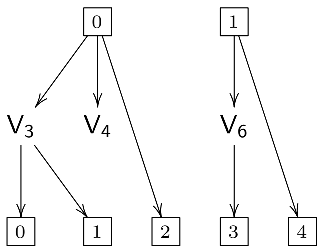
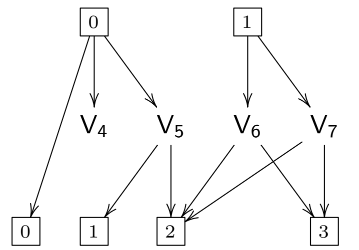


Figure 2: Complete place graph used in this example

Lets consider a bound $\overrightarrow{A} : h \to \overrightarrow{m}$ (span), and $\overrightarrow{D} : \overrightarrow{m} \to p$ (cospan) -Figure: 4 and 3- over the place graph introduced in Figure 2. Our ultimate goal is to create a RPO for the bound such that any other relative bound can be reached by the RPO (see Figure 1).

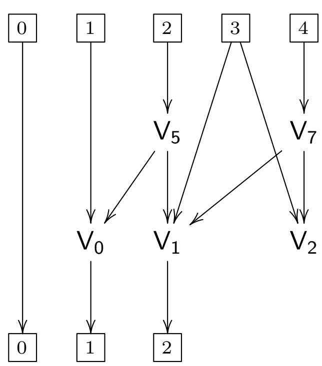


(a) D_0

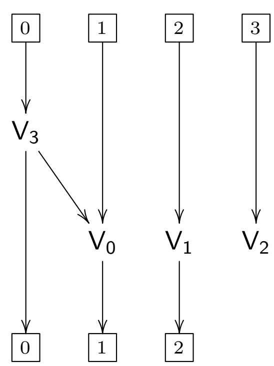


(b) D_1

Figure 3



(a) A_0



(b) A_1

Figure 4

4.1 Nodes

Let us now define the node sets V_{0-3} which is straightforward.

$$V_0 = \{v_0, v_1, v_2, v_5, v_7\}$$
$$V_1 = \{v_0, v_1, v_2, v_3\}$$
$$V_2 = \{v_0, v_1, v_2\}$$
$$V_3 = \{v_4, v_6\}$$

The bigraph B_0 will therefore consist of nodes $V_1/V_2 = \{v_3\}$ and B_1 of $V_0/V_2 = \{v_5, v_7\}$. Bigraph B consists of $V_3 = \{v_4, v_6\}$.

4.2 Interface

We can continue by defining the interface $\hat{m}$ which lies between B_i and B. For this we first require m'_0 and m'_1. We do this by reducing all roots which have a shared place as a child, and this shared place has only in one of the two As a root as a parent, from m_i. Therefore m'_i is the set $\{1, 2, 3, 4\}$ (root 0 is not in m_0 as site 0 is not connected to a root parent in A_1) and $\{0, 1, 2, 3\}$ for $i = 0, 1$ respectively. Whenever two roots (of m'_0 and m'_1) have a child in common, they belong to the same $\cong$-equivalence class. Therefore we can determine up to isomorphism (i.e. the order is not important) the $\cong$-equivalence classes for $\hat{m} := (m'_0 + m'_1)/\cong$ which are:

$$(1, 0)$$
$$(0, 1) \cong (1, 1)$$
$$(0, 2)$$
$$(0, 3) \cong (1, 2) \cong (1, 3)$$
$$(0, 4)$$

With this at hand we can conclude the interfaces for the place graphs are:

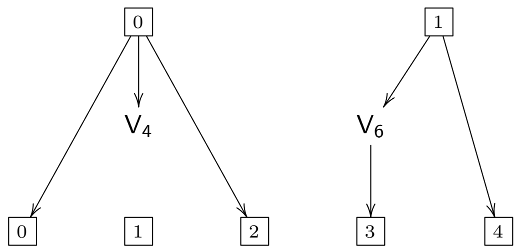
$$B_0^P : 5 \to 5$$
$$B_1^P : 4 \to 5$$
$$B^P : 5 \to 2$$

4.3 Parents

All that is left to do now is to connect the places to the parents with the formulas introduced in Section 3. Lets start with the sites in B_0. Sites 1 to 4 are all in m'_0 and have therefore a corresponding $\cong$-equivalence root as one of their parents. The same is the case for all four sites of B_1. Furthermore, the sites 0 and 1 of B_0 as well as the sites 1, 2 and 3 of B_1 are connected to nodes, in D_0 and D_1 respectively, which are **not** in V_3. Therefore, those nodes are parents of the sites in B_i.

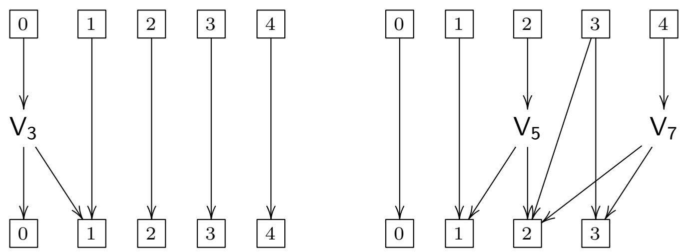
We continue with the nodes in B_0. The node v_3 has a parent in A_1 which is a root. Hence, we connect v_3 to the corresponding $\cong$-equivalence class root in $\hat{m}$. v_3 has no further parents in D_0. The nodes v_5 and v_7 of bigraph B_1 are similar.

Eventually, we can define the parents for places in bigraph B. This process is very straightforward. Lets consider one concrete example: The site of the $\cong$-equivalence class $(0, 3) \cong (1, 2) \cong (1, 3)$. The parents of the corresponding sites in D_0 and D_1 are the sets $\{v_6\}$, $\{v_5, v_6, v_7\}$ and $\{v_6, v_7\}$. However, the intersection with M gives a distinct result, namely $\{v_6\}$. Note that the intersection with M might be the empty set ($\emptyset$), in this case the site



(a) B

Figure 5



(a) B_0 (b) B_1

Figure 6

becomes an orphan (for example site $\widehat{0,1}$). The parent relation of nodes in B is even simpler and it does not matter if D_0 or D_1 is used as the reference point. Note that we do not require the intersection with M for this relation, because if note v is neither in A_0 nor in A_1, then it is guaranteed that all parents of v are also not in A_i and therefore automatically in M.

The final place graphs of the bigraphs B_i and B can be seen in Figures 6 and 5 respectively.

5 Pseudo Code of algorithm

In this section we introduce the pseudo code emerging from the introduced algorithm. We will leave parts of the code uncommented as it should be fairly self-explanatory. Further functions which are necessary for this pseudo code are introduced in Appendix B The reader is advised to use Section 3 as a reference. We will continue with discussing some of the aspects introduced in the pseudo code afterwards.

5.1 Arguments

The input of the algorithm are the four epimorphic bigraphs A_0, A_1, D_0 and D_1. In particular we make use of the notation V_X which gives the set of all nodes in bigraph X as well as $prnt_X(v)$ which gives the set of parents of place v in bigraph X. Moreover we note that an interface h is a bound over the natural numbers such that $[0, h) \in \mathbb{N}$. The entire algorithm is built upon those principles only.

Algorithm 1 RPO algorithm for epimorphic place bigraphs with sharing

function rpo(A_0, A_1, D_0, D_1) ▷ Input: $\overrightarrow{A} : h \to m_i$, $\overrightarrow{D} : m_i \to p$
    $V_{B_0} := V_{A_1} - V_{A_0}$ ▷ Nodes
    $V_{B_1} := V_{A_0} - V_{A_1}$
    $V_B := V_{D_0} \cap V_{D_1}$
▷ Interface
    $(red_0, red_1) =$ buildRed$(A_0, A_1, \emptyset, \emptyset)$
    $\hat{M} := \emptyset$ ▷ $\hat{M}$ is a set of sets of tuples (i, r)
    for all $i = [0, 1]$ **do**
        for all $r \in m_i$ **do**
            if $r \notin red_i$ **then**
                $\hat{M} += \{(i, r)\}$
            end if
        end for
    end for
▷ Determine $\cong$-equivalence relations
    for all $w \in (V_{A_0} \cap V_{A_1}) \uplus h$ **do**
        if rootPrnt$(w, A_0, red_0) =$ SOME $r_0 \wedge$ rootPrnt$(w, A_1, red_1) =$ SOME r_1 **then**
            $s :=$ rootEqui$(\hat{M}, (0, r_0))$
            $t :=$ rootEqui$(\hat{M}, (1, r_1))$
            $\hat{M} -= s$
            $\hat{M} -= t$
            $\hat{M} += s \cup t$
        end if
    end for
    $B_0 := m_0 \to [0, |\hat{M}|) \in \mathbb{N}$
    $B_1 := m_1 \to [0, |\hat{M}|) \in \mathbb{N}$
    $B := [0, |\hat{M}|) \in \mathbb{N} \to p$

    $\hat{m} :=$createMappping$(\hat{M}, [0, |\hat{M}|) \in \mathbb{N})$
▷ Parent Relations
    for all $i = [0, 1] \wedge \bar{\imath} = [1, 0]$ **do** ▷ B_0 and B_1
        for all $r \in m_i$ **do** ▷ Sites
            if rootEqui$(\hat{M}, (i, r)) \neq \emptyset$ **then**
                $prnt_{B_i}(r) = \{$map$(\hat{m},$rootEqui$(\hat{M}, (i, r)))\}$
            else
                $prnt_{B_i}(r) = \emptyset$
            end if
            for all $par \in prnt_{D_i}(r)$ **do**
                if $par \in V_{A_0} \cup V_{A_1}$ **then**
                    $prnt_{B_i}(r) += par$
                end if
            end for
        end for

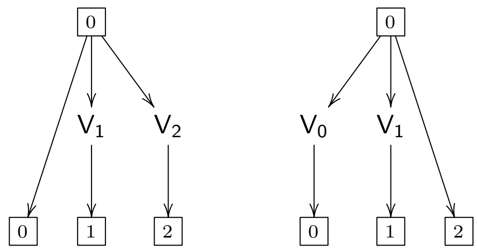
44: **for all** $v \in V_{A_{\bar{\imath}}} - V_{A_i}$ **do** $\triangleright$ Nodes
45: **if** rootPrnt$(v, A_{\bar{\imath}}, red_{\bar{\imath}})$ = SOME r **then**
46: $prnt_{B_i}(v) = \{\text{map}(\hat{m},\text{rootEqui}(\hat{M}, (\bar{\imath}, r)))\}$
47: **else**
48: $prnt_{B_i}(v) = \emptyset$
49: **end if**
50: **for all** $par \in prnt_{D_i}(v)$ **do**
51: **if** $par \in V_{A_0} \cup V_{A_1}$ **then**
52: $prnt_{B_i}(v) += par$
53: **end if**
54: **end for**
55: **end for**
56: **end for**
57: $\triangleright$ B
58: **for all** $\hat{r} \in \hat{m}$ **do**
59: $(i, r) = takeFirst(\hat{r}.KEY)$ $\triangleright$ Take the next best site
60: $prnt_B(\hat{r}.ORDINAL) = \emptyset$ $\triangleright$ Initialize parent set
61: **for all** $par \in prnt_{D_i}(r)$ **do**
62: **if** $par \in (V_{D_0} \cap V_{D_1}) \uplus p$ **then**
63: $prnt_B(\hat{r}.ORDINAL) += par$
64: **end if**
65: **end for**
66: **end for**
67: **for all** $v \in V_{D_0} \cap V_{D_1}$ **do**
68: $prnt_B(v) = prnt_{D_0}(v)$
69: **end for**
70: **return** B_0, B_1, B
71: **end function**

The algorithm returns the RPO triple (B_0, B_1, B).

5.2 Data structures

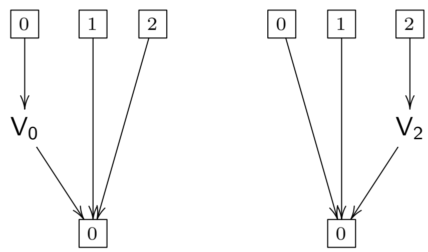
As previously mentioned the notation $prnt_X(v0)$ is the set of all parent-nodes and -roots of node $v0$ in bigraph X. If we write $prnt_X(v0) += v1$ we say that $v1$ is added to the parent set of $v0$ ($v1$ becomes a parent of $v0$). For the interested reader: Note that this means $prnt_X += (v0, v1)$ with regard to definition 3.2.1 of source [3].
In the algorithm the set $\hat{M}$ denotes a set of sets of tuples, such that each set (i.e. each element of $\hat{M}$) is one $\cong$-equivalence class. We require $\hat{M}$ to be dynamical as a lot of changes are required in lines 16 to 30 (therefore we postpone the mapping). For the example in section 4 $\hat{M} = \{\{(0,2)\}, \{(0,4)\}, \{(1,0)\}\}$ on line 14 and

$$\begin{gathered}\hat{M} = \{\{(1,0)\},\\ \{(0,1),(1,1)\},\\ \{(0,2)\},\\ \{(0,3),(1,2),(1,3)\},\\ \{(0,4)\}\}\end{gathered}$$



(a) D_0 (b) D_1

Figure 7



(a) A_0 (b) A_1

Figure 8

on line 31 (note that the specific order does not matter as it is a set). Next we assign an ordinal for each member ($\cong$-equivalence class) of $\hat{M}$. This could certainly be done in different ways, however, we will use tuples such that the first field (denoted as KEY) is a set of tuples (the members of $\hat{M}$) and the second field (denoted as ORDINAL) contains a mapping to a unique natural number. For the example in Section 4 we have therefore:

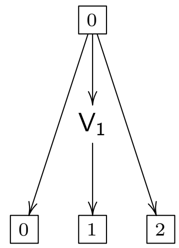
$$\begin{gathered}\hat{m} = \{(\{(1,0)\},0),\\ (\{(0,1),(1,1)\},1),\\ (\{(0,2)\},2),\\ (\{(0,3),(1,2),(1,3)\},3),\\ (\{(0,4)\},4)\}\end{gathered}$$

Note that the mapping is a bijection between the members of $\hat{M}$ and the bound $[0, |\hat{M}|) \in \mathbb{N}$.

6 RPOs in generic Bigraphs with sharing

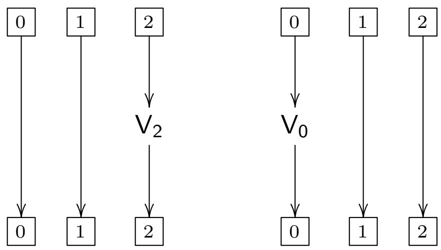
As we have seen in Section 3 it is possible to reason and construct RPOs in epimorphic bigraphs. We will now consider an example which suggests that this might not be possible for all bigraphs. For this, consider the bound shown in figures 7 and 8.

If there exists a RPO it should clearly contain the nodes V_2 and V_0 in B_0 and B_1 respectively. Let us now consider the candidate triple shown in Figures 9 and 10 which looks promising to be a RPO at the first glance.



(a) B

Figure 9



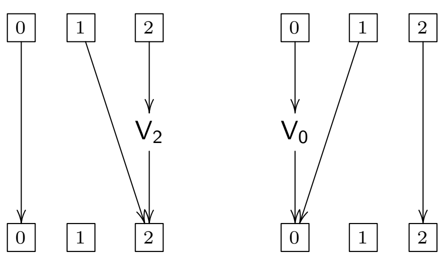
(a) B_0 (b) B_1

Figure 10

The introduced candidate triple fulfils all criteria to be a bound for $\overrightarrow{A}$ relative to $\overrightarrow{D}$. In particular: $B_0 \circ A_0 = B_1 \circ A_1$, $B \circ B_0 = D_0$ and $B \circ B_1 = D_1$. However, as we will see the triple B_i, B can not be a RPO as there are relative bounds which can not be reached from the triple. For example consider the bound shown in Figure 11 with $K = B$.

It is fairly easy to see that a bigraph j which would fulfil the criteria $j \circ B_0 = K_0$ leads to $\Rightarrow j \circ B_1 \neq K_1$. And equally vice versa $j' \circ B_1 = K_1 \Rightarrow j' \circ B_0 \neq K_0$. On the other hand it is also not possible to find a unique morphism from K_i to B_i which proves that neither B nor K can be a RPO.

It seems like that there is for any given bound to A relative to D another relative bound



(a) B_0 (b) B_1

Figure 11

which can not be reached from the former. We also note the interesting property: $B_i \circ A_i = K_i \circ A_i \wedge B_i \neq K_i$. The reader is advised to try her/himself further examples with the bound $\overrightarrow{D} \circ \overrightarrow{A}$ and use this as a benchmark test for further studies.
With regard to the given example the most likely scenario is that from each relative bound there is only a certain number of other relative bounds which can be reached with a unique morphism j. However, this is only a suggestion and further research is required to properly reason about this. A further discussion about monomorphic bigraphs can be found in Appendix C.

7 Conclusion

7.1 Discussion

In this report we showed the construction of an algorithm to find RPOs in place graphs of epimorphic bigraphs with sharing. Bigraphs with sharing is an extension, introduced by Michele Sevegnani, of Robin Milner's BRS. We briefly touched why this algorithm is useful and sketched an overview of the broader context. We showed with an example how the introduced algorithm works and gave an outline of a pseudo code which has been used to implement the algorithm in the BigraphER tool. Finally, we briefly discussed RPOs in bigraphs with sharing in general as well as in the special case of monomorphic graphs.

7.2 Future work

Further research will be needed in the field of RPOs in bigraphs with sharing in general. In Section 6 we sketched out an example which suggests that it might not be possible to find RPOs for all bigraphs with sharing. However, we omitted a proof or further discussion. More work needs to be done to fully understand the problem.
A further field of research is the subcategory of monomorphic bigraphs with sharing. In Appendix C we briefly outline a potential algorithm for finding RPOs in place graphs of monomorphic bigraphs with sharing. However, more work is needed to assure correctness of this algorithm as well as an implementation in the BigraphER tool.

Appendices

A Proof of epimorphic algorithm

In this appendix we will prove that the outlined algorithm of Section 3 is correct, in particular that the result is indeed a RPO for the bound. First we will prove that the produced outcome is in fact a relative bound to the bound $\overrightarrow{D}$. Secondly, we will show that for any other relative bound $\overrightarrow{K}, K$ there is a unique morphism j such that $j \circ B_i = K_i$. The reader is advised to use Figure 1 as a reference.

A.1 Proof relative bound

To prove that $\overrightarrow{B}, B$ is in fact a relative bound to $\overrightarrow{D}$ one can simply prove $B \circ B_i = D_i$ and $B_0 \circ A_0 = B_1 \circ A_1$. By the definition of equality of place graphs with sharing two place graphs are equal iff their node sets are equal, their interfaces are equal and their parent relations are equal. We will first show that these three properties hold for $B \circ B_i = D_i$ and secondly for $B_0 \circ A_0 = B_1 \circ A_1$.

A.1.1 B relative to D

Nodes

Proposition A.1. The set of nodes of $B \circ B_i$ is equal to D_i.

Proof. By the construction 5.9 of source [2] which has been used in Section 3.3 we have the following three properties.

$$|D_i| = (V_{\bar{\imath}}/V_2) \uplus V_3 \tag{1}$$

$$|B_i| = V_{\bar{\imath}}/V_2 \tag{2}$$

$$|B| = V_3 \tag{3}$$

By the definition 3.2.2 of source [3] the composition of two bigraphs has the support $|G \circ F| = V_F \uplus V_G$. Therefore we can change Equation 1 in the following manner.

$$\begin{gathered} |D_i| = (V_{\bar{\imath}}/V_2) \uplus V_3 \\ \Leftrightarrow |D_i| = |B_i| \uplus |B| \\ \Leftrightarrow |D_i| = |B \circ B_i| \end{gathered}$$

□

Interfaces

Proposition A.2. If $B \circ B_i : a_i \to b$ and $D_i : c_i \to d$ then $a_i = c_i$ and $b = d$.

Proof. By the definition given in Section 3.4 we note the following three properties.

$$D_i : m_i \to p \tag{4}$$

$$B_i : m_i \to \hat{m} \tag{5}$$

$$B : \hat{m} \to p \tag{6}$$

By the definition 3.2.2 of source [3] the composition of two bigraphs, $F : k \to m$ and $G : m \to n$, is equal to $G \circ F : k \to n$ therefore

$$B \circ B_i : m_i \to p$$

As given in Equation 4.

□

Parent Relation

Proposition A.3. The parent relation of the composition $B \circ B_i$ is equal to the parent relation of D_i.

We require to note the following in order to prove Proposition A.3.

Definition A.1. All parent relations in $prnt_{D_i}$ are member of one, and only one, of these parent relations: A place which is not in M to a parent equally not in M, a place which is not in M to a parent which is in M and a member of M to a different member of M. We shall denote those relations with $prnt^{PP}$, $prnt^{PM}$ and $prnt^{MM}$ respectively. Therefore:

$$prnt_{D_i} = prnt_{D_i}^{PP} \uplus prnt_{D_i}^{PM} \uplus prnt_{D_i}^{MM} \tag{7}$$

By the definition 3.2.2 of source [3] the composition of $B \circ B_i$ has the parent relation

$$prnt := prnt_B^{\triangleleft} \uplus prnt_{\circ} \uplus prnt_{B_i}^{\triangleright} \tag{8}$$

To prove that Equations 7 and 8 are equal we will show that their components are equal with the following three lemmas.

Lemma A.3.1. $prnt_{D_i}^{PP} = prnt_{B_i}^{\triangleright}$

Proof. By the definition given in Section 3.5 all places (sites and nodes) in B_i take over all the parent relations of D_i except of those which are in M. Therefore directly:

$$prnt_{D_i}^{PP} = prnt_{B_i}^{\triangleright}$$

□

Lemma A.3.2. $prnt_{D_i}^{PM} = prnt_{\circ}$

Proof. First of all, we note that for each site which has a parent relation in $prnt^{PM}$ there is a unique root in $\hat{m}$ which will provide all connections to M (see Proposition A.7). B connects the sites of $\hat{m}$ to the corresponding members of M according to the origin of the site (i.e. one of $\cong$-equivalence sites of D. As proved in Proposition A.7 all sites in a $\cong$-equivalence class have the same parents in M). Note that if a site of $\hat{m}$ does not have a parent relation to M (even though it had the potential from the viewpoint of A), then it is an orphan in B. Orphan sites and their relations can be discarded in the composition $prnt_{\circ}$. Therefore, $prnt_{\circ}$ contains only relations from P to M as required. □

Lemma A.3.3. $prnt_{D_i}^{MM} = prnt_B^{\triangleleft}$

Proof. By the definition given in Section 3.5 all nodes in B take over all the parent relations of D_i. Moreover, by the definition in Section 3.3, B has only nodes of $V_3 \subseteq M$ Therefore directly:

$$prnt_{D_i}^{MM} = prnt_B^{\triangleleft}$$

□

A.1.2 B bound for A

Nodes

Proposition A.4. The set of nodes of $B_0 \circ A_0$ is equal to $B_1 \circ A_1$.

Proof. By the construction 5.9 of source [2] which has been used in Section 3.3 we have the following three properties.

$$|A_i| = V_i \tag{9}$$

$$|B_i| = V_{\bar{\imath}}/V_2 \tag{10}$$

$$V_2 = V_i \cap V_{\bar{\imath}} \tag{11}$$

By the definition 3.2.2 of source [3] the composition of two bigraphs has the support $|G \circ F| = V_F \uplus V_G$. Therefore we can change the equations in the following manner.

$$\begin{gathered}
|B_i \circ A_i| = V_i \uplus (V_{\bar{\imath}}/V_2) \\
\Leftrightarrow |B_i \circ A_i| = V_i \uplus (V_{\bar{\imath}}/(V_i \cap V_{\bar{\imath}})) \\
\Leftrightarrow |B_i \circ A_i| = V_i \uplus (V_{\bar{\imath}}/V_i) \\
\Leftrightarrow |B_i \circ A_i| = V_i \uplus V_{\bar{\imath}} \\
\Leftrightarrow |B_i \circ A_i| = V_{\bar{\imath}} \uplus V_i \\
\Leftrightarrow |B_i \circ A_i| = |B_{\bar{\imath}} \circ A_{\bar{\imath}}|
\end{gathered}$$

□

Interfaces

Proposition A.5. If $B_i \circ A_i : a \to b$ and $B_{\bar{\imath}} \circ A_{\bar{\imath}} : c \to d$ then $a = c$ and $b = d$.

Proof. By the definition given in Section 3.4 we note the following two properties.

$$A_i : h \to m_i \tag{12}$$

$$B_i : m_i \to \hat{m} \tag{13}$$

By the definition 3.2.2 of source [3] the composition of two bigraphs, $F : k \to m$ and $G : m \to n$, is equal to $G \circ F : k \to n$ therefore directly

$$B_i \circ A_i : h \to \hat{m}$$

As required.

□

Parent Relation

Proposition A.6. The parent relation of the composition $B_i \circ A_i$ is equal to the parent relation of $B_{\bar{\imath}} \circ A_{\bar{\imath}}$.

Proof. By the definition given in Section 3.5 all parent relations are directly taken from D which is in itself a bound for A and must therefore be consistent. □

A.2 $\cong$-quivalence classes

Proposition A.7. Roots which are $\cong$-equivalent have the same -if any- parents in M. Therefore each $\cong$-equivalence class denotes one unique set of parents in M.

Proof. Each shared place $w \in h \uplus V_2$ which **might** have a parent in M must have a root of m'_i as a parent in both A_0 and A_1, because by definition M is the shared part of D and can therefore not be in A. Because we are in the category of epimorphic bigraphs, those roots must connect to the very same nodes and roots in M as otherwise $\overrightarrow{A}$, $\overrightarrow{D}$ would not be a bound. Therefore, $\cong$-equivalent roots/sites have the same parents in M. $\square$

A.2.1 Orphan sites in B

It is essential to keep orphan sites in the RPO. If we would discard them it would be possible to find other candidate triples for the RPO (with orphans), where there is no unique **epimorphic** morphism j from our triple candidate (without orphans) to the one (with orphans). Hence, it is clearly necessary to keep all possible orphans.

A.3 Proof RPO

As we have shown the triple (B_0, B_1, B) is indeed a relative bound for $\overrightarrow{A}$ to $\overrightarrow{D}$. In order for the triple to be a RPO we have to show that $\overrightarrow{B}$ is the closest cospan to span $\overrightarrow{A}$ possible and that all three bigraphs of the triple are guaranteed to be epimorphic [6]. We will hereinafter assume that the given bigraphs $\overrightarrow{A}$ and $\overrightarrow{D}$ are epimorphic.

Proposition A.8. The triple (B_0, B_1, B) is guaranteed to be epimorphic.

Proof. By the definition in Section 3.5 all roots and parent relations in B are taken from D_i and all sites from m_i with a potential for being connected to a root (see Preposition A.7) are also in B. Therefore, no root can be idle nor can there be two roots which are partners in B. By definition, each place of B_i is connected to at most one root of $\hat{m}$. Moreover, there can not be an idle root in B_i, because each root has to be a parent of either a site or a node. $\square$

Proposition A.9. Cospan $\overrightarrow{B}$ is the closest possible bound for span $\overrightarrow{A}$ relative to $\overrightarrow{D}$.

Proof. By definition, $|B_i| = |A_{\bar{\imath}}|/|A_i|$ therefore directly $|B_i \circ A_i| = |A_0| \cup |A_1|$ hence no more nodes are added to the composite. Moreover, each place in A_0 or A_1 which has the potential to have a parent which is not in A (i.e. the place has always a root of m'_i as a parent) has this property preserved by the definition of interface $\hat{m}$. Since no more places are added through B_i and **all** potential parent relations are preserved. The bound $\overrightarrow{B} \circ \overrightarrow{A}$ must therefore be the closest bound to the span $\overrightarrow{A}$ relative to $\overrightarrow{D}$. $\square$

B Additional pseudo code functions

Algorithm 2 Root parent function. Gives the single root *-if any-* of a place

function rootPrnt(v, C, red) ▷ Input: Node v, bigraph $C : k \to l$, reduction red
  for all $p \in prnt_C(v)$ **do**
    if $p \in l$ **then**
      if $p \in red$ **then**
        return NONE
      else
        return SOME p
      end if
    end if
  end for
  return NONE
end function

Algorithm 3 Recursive algorithm to build reduction sets

function buildRed(A_0, A_1, red_0, red_1) ▷ Input: Two bigraphs, two reduction sets
  for all $w \in (V_{A_0} \cap V_{A_1}) \uplus h$ **do**
    if rootPrnt(w, A_0, red_0) = SOME r_0 **then**
      if rootPrnt(w, A_1, red_1) = NONE **then**
        return buildRed$(A_0, A_1, red_0 + r_0, red_1)$
      end if
    else if rootPrnt(w, A_1, red_1) = SOME r_1 **then**
      return buildRed$(A_0, A_1, red_0, red_1 + r_1)$
    end if
  end for
  return (red_0, red_1)
end function

Algorithm 4 Root equivalence function. Gives the set which contains the root

function rootEqui$(\hat{M}, r)$ ▷ Input: Set of sets $\hat{M}$, root identifier tuple r
  for all $s \in \hat{M}$ **do**
    if $r \in s$ **then**
      return s
    end if
  end for
  return $\emptyset$
end function

Algorithm 5 Creates an injective mapping from $\hat{M}$ to N

function createMappping$(\hat{M}, N)$ ▷ Input: Set $\hat{M}$, Numbers N
  $\hat{m} := \emptyset$
  for all $m \in \hat{M} \wedge n \in N$ **do**
    $\hat{m} += (m, n)$
  end for
  return $\hat{m}$
end function

Algorithm 6 Maps a distinct set to an ordinal

function map($\hat{m}, S$) ▷ Input: Set of mapping $\hat{m}$, root set S
  for all $R \in \hat{m}$ **do** ▷ R is a tuple such that (KEY, ORDINAL)
    if $R.KEY = S$ **then**
      return $R.ORDINAL$
    end if
  end for
end function

Algorithm 7 Returns the first tuple found in the set

function takeFirst(S) ▷ Input: Set of tuples S
  for all $s \in S$ **do**
    return s
  end for
end function

C Outline of monomorphic algorithm

In this appendix we will discuss some of the features an algorithm for monomorphic bigraphs with sharing would need to have. In particular we will focus on the construction of the mediating interface $\hat{m}$ as all other parts should be fairly similar to the introduced algorithm for epimorphic bigraphs with sharing. We will omit a further proof of the correctness and further work will be needed before this algorithm can be implemented. The reader is advised to use Figure 1 as a reference.

We note that in a monomorphic bound each site of m_i has a unique parents set of parents in M which is disjoint with all the other parent sets of m_i for $i = 0, 1$ respectively (that is, if sides are considered separately). Furthermore we notice that a morphism j can always perform splits on sites, however never merges as this would violate the monomorphic restriction (no two sites are siblings). Therefore $\hat{m}$ must provide the fewest number of sites possible.

Hereinafter we will always mean parent set of M if we talk about the parent set, unless otherwise stated.

Let us first consider m_0. As mentioned each site has its own parent set which is disjoint with all other parent sets. We can drop all empty parent sets as it can not connect to a root in $\hat{m}$ as this root would be an idle site in B (violation of monorphism).

Let us now consider m_1. Each parent set will intersect with a number of the parent sets of m_0 but never with another parent set of m_1. By considering the parent sets and their intersection we can create concrete parent sets such that each intersection of a parent set has a unique concrete parent set. As before, we can drop all empty concrete parent sets, in particular only intersections will be non-empty. Those concrete parent sets represent the bare minimum of roots/sites needed for $\hat{m}$. In particular we create for each concrete parent set (i.e. each intersection) a site in $\hat{m}$ and connect the site in B to the corresponding parents of M. In B_0 and B_1 the sites connect to the corresponding roots such that all concrete parent sets of the site's parent set are connected to the site.

References

[1] Horst Herrlich and George E. Strecker. *Category Theory*. Allyn and Bacon, Inc., 1 edition, 1973.

[2] Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 1 edition, 2009.

[3] Michele Sevegnani. *Bigraphs with sharing and applications in wireless networks*. PhD thesis, University of Glasgow, 2012.

[4] Michele Sevegnani and Muffy Calder. Bigraphs with sharing. *Theoretical Computer Science*, 577:44–73, April 2015.

[5] Michele Sevegnani and Muffy Calder. Bigrapher: rewriting and analysis engine for bigraphs. *CAV 2016, Lecture Notes in Computer Science*, 9780(II):494–501, July 2016.

[6] Mariusz Szmajduch. Rpos for bigraphs with sharing. Research Report: http://www.academia.edu/21610250/RPOs_for_bigraphs_with_sharing [Accessed on: 18 July 2016], 2015.