A CSP implementation of the bigraph embedding problem

Marino Miculan marino.miculan@uniud.it
Marco Peressotti marco.peressotti@uniud.it
Laboratory of Models and Applications of Distributed Systems
Department of Mathematics and Computer Science
University of Udine, Italy

Abstract

A crucial problem for many results and tools about bigraphs and bigraphical reactive systems is bigraph embedding. An embedding is more informative than a bigraph matching, since it keeps track of the correspondence between the various components of the redex (guest) within the agent (host). In this paper, we present an algorithm for computing embeddings based on a reduction to a *constraint satisfaction* problem. This algorithm, that we prove to be sound and complete, has been successfully implemented in LibBig, a library for manipulating bigraphical reactive systems. This library can be used for implementing a wide range of tools, and it can be adapted to various extensions of bigraphs.

1 Introduction

Bigraphical Reactive Systems (BRSs) [14, 19] are a flexible and expressive meta-model for ubiquitous computation. System states are represented by *bigraphs*, which are compositional data structures describing at once both the locations and the logical connections of (possibly nested) components of a system. Like graph rewriting [24], the dynamic behaviour of a system is defined by a set of *(parametric) reaction rules*, which can modify a bigraph by replacing a *redex* with a *reactum*, possibly changing agents' positions and connections.

BRSs have been successfully applied to the formalization of a broad variety of domain-specific calculi and models, from traditional programming languages to process calculi for concurrency and mobility, from context-aware systems to web-service orchestration languages, from business processes to systems biology; a non exhaustive list is [2,4,5,8,16,18]. Very recently bigraphs have been used in structure-aware agent-based computing for modelling the structure of the (physical) world where the agents operates (e.g., drones, robots, etc.) [20].

Beside their normative and expressive power, BRSs are appealing because they provide a range of interesting general results and tools, which can be readily instantiated with the specific model under scrutiny: simulation tools, systematic construction of compositional bisimulations [14], graphical editors [9], general model checkers [23], modular composition [22], stochastic extensions [15], etc.

In this paper, we give an implementation for a crucial problem that virtually all these tools have to deal with, i.e., the matching a bigraph inside an agent. Roughly, this can be stated as follows: given R and A, we have to find (all, or some) C, D such that $A = C \circ R \circ D$. Clearly this is required by any simulation tool (in order to apply a reaction rule, we have to match the redex inside the agent, and then replace it with the reactum), but also in other tools, e.g., for implementing "find&replace" in graphical editors, for occurrence checks in sortings [1] and model checkers, for refinements in architectural design tools, etc.

Like the similar and well-known *subgraph isomorphism* problem, bigraph matching is NP-complete (see [3]). However, using the theory of Fixed Parameter Tractability it can be shown that the exponential explosion depends only on the size (more precisely, the width) of the redex to be found, and not on the size of the agent. In most practical cases, this width is constant and small (e.g. ≤ 3), hence the problem becomes feasible.

A (rather *ad-hoc*) implementation of bigraph matching has been given in the *BPLTool* [7]; this was based on a term-based representation of agents and rules, in the spirit of term rewriting systems. More recently, a more graphical-oriented approach has been preferred. Højsgaard have introduced the notion of *bigraph embedding* [11], which is a function from nodes and edges of the redex to nodes and edges of the agent, describing how the former is embedded in the latter. Although embeddings and matchings are basically equivalent, embeddings turn out to be more useful especially in connection with Gillespie-like algorithms for stochastic simulations [11], because they allow for a simpler calculation of interference between redexes. In fact, embeddings are at the core of the *Bigraphic Abstract Machine* [21], a general abstract machine for implementing various kinds of BRSs, with several possible execution strategies.

For these reasons, in this paper we focus on the *bigraph embedding problem*. More precisely, we translate the embedding problem to a *constraint satisfaction problem* (CSP), whose solutions correspond to bigraph embeddings. Instead of defining directly a CSP for the bigraph embedding problem we take advantage of bigraphs being the "merge" of two graphical structures (called *link graphs* and *place graphs* respectively): initially we define the encoding for embeddings of these two orthogonal structures separately and then combine them by means of some consistency constraints reflecting the interplay between link and place structures. This split mimics the peculiar structure of bigraphs and allows us to factor the exposition of the problem, its encoding and the accompanying adequacy results. An implementation based on the *CHOCO* solver is available in *LibBig* (available at `http://mads.dimi.uniud.it/`), an extensible library for manipulating bigraphical reactive systems. However, this is an implementation choice mainly due to the use of Java, but the results of this paper can be implemented in any solver capable of handling the integer solutions of a linear equation system.

We do not provide an exhaustive discussion of experimental results because the encoding proposed is "solver-independent" and moreover, because there is no widely-acknowledged benchmark suit for this problem. In fact, finding a reasonably representative set of instances still is an open question.

Synopsis In Section 2 we briefly recall the notion of bigraphs and bigraphical reactive systems and in Section 3 present the bigraph embedding problem, its complexity and how it can be divided in two sub-problems, by taking advantage of the components of bigraphs. The implementation of the bigraph embedding problem as a constraint satisfaction problem and the adequacy results are presented in Sections 4. Conclusions, with some experimental evaluations, and some directions for future work are in Section 5.

2 Bigraphical reactive systems

In this section we briefly recall the notion of Bigraphical Reactive Systems (BRS) referring the interested reader to [19].

The key point of BRSs is that "the model should consist in some sort of reconfigurable space". Agents may interact in this space, even if they are spatially separated. This means that two agents may be adjacent in two ways: they may be at the same *place*, or they may be connected by a *link*. This leads to the definition of *bigraphs* as a data structure for representing the state of the

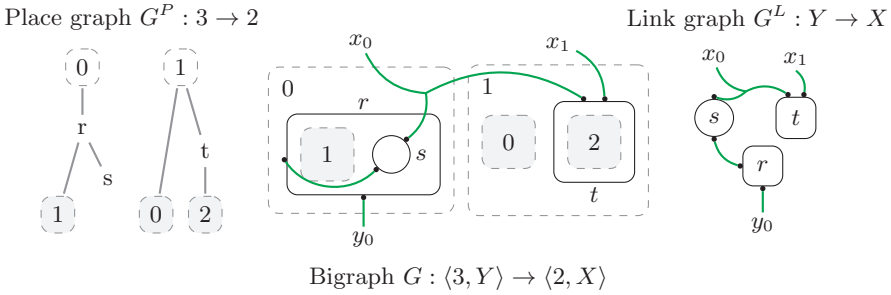


Figure 1: Forming a bigraph from a place graph and a link graph.

system. A bigraph can be seen as enriched hyper-graph combining two independent graphical structures over the same set of *nodes*: a hierarchy of *places*, and a hyper-graph of *links*.

Definition 1 (Bigraph [19, Def. 2.3])**.** *A* bigraph graph G *over a given signature* Σ *(i.e. a set of types, called controls, denoting a finite arity) is a an object*

$$(V_G, E_G, \mathsf{ctrl}_G, \mathsf{prnt}_G, \mathsf{link}_G) : \langle n_G, X_G \rangle \to \langle m_G, Y_G \rangle$$

composed by two substructures (Figure 1): a place graph $G^P = (V_G, \mathsf{ctrl}_G, \mathsf{prnt}_G) : n_G \to m_G$ *and a* link graph $G^L = (V_G, E_G, \mathsf{ctrl}_G, \mathsf{link}_G) : X_G \to Y_G$*. The set* V_G *is a finite set of nodes and to each of them is assigned a control in* Σ *by the* control map $\mathsf{ctrl}_G : V_G \to \Sigma$*. The set* E_G *is a finite set of names called* edges*.*

These structures presents an inner interface (composed by n_G *and* X_G*) and an outer one (*m_G*,* Y_G*) along which can be composed with other of their kind as long as they do not share any node or edge. In particular,* X_G *and* Y_G *are finite sets of names and* n_G *and* m_G *are finite ordinals that index* sites *and* roots *respectively.*

On the side of G^P*, nodes, sites and roots are organized in a forest described by the* parent map $\mathsf{prnt}_G : V_G \uplus n_G \to V_G \uplus m_G$ *such that sites are leaves and roots are exactly* m_G*.*

On the side of G^L*, nodes, edges and names of the inner and outer interface forms a hyper-graph described by the* link map $\mathsf{link}_G : P_G \uplus X_G \to E_G \uplus Y_G$ *which is a function from* X_G *and ports* P_G *(i.e. elements of the finite ordinal associated to each node by its control) to edges* E_G *and names in* Y_G*.*

The dynamic behaviour of a system is described in terms of *reactions* of the form $a \rightarrowtriangle a'$ where a, a' are agents, i.e. bigraphs with inner interface $\langle 0, \emptyset \rangle$. Reactions are defined by means of graph rewrite rules, which are pairs of bigraphs (R_L, R_R) equipped with a function η from the sites of R_R to those of R_L called *instantiation rule*. A bigraphical encoding for the open reaction rule of the Ambient Calculus is shown in Figure 2 where redex and reactum are the bigraph on the left and the one on the right respectively and the instantiation rule is drawn in red. A rule fires when its redex can be embedded into the agent; then, the matched part is replaced by the reactum and the parameters (i.e. the substructures determined by the sites of the redex) are instantiated accordingly with η.

3 Bigraph embeddings

In this Section we briefly recall the notion of *bigraph embedding*. The following definitions are taken from [11], with minor modification to simplify the presentation of the equivalent CSP

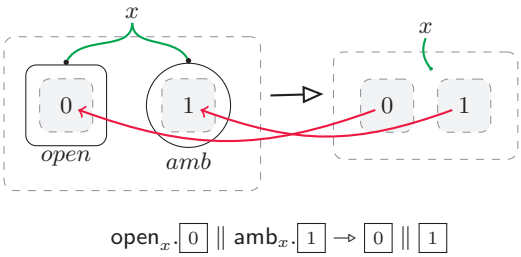


Figure 2: Open reaction rule of the Ambient Calculus.

formulation (cf. Section 4). As usual, we shall exploit the orthogonality of the link and place graphs, by defining *link and place graph embeddings* separately and then combine them to extend the notion to bigraph.

Link graph Intuitively an embedding of link graphs is a structure preserving map from one link graph (the *guest*) to another (the *host*). As one would expect from a graph embedding, this map contains a pair of injections: one for the nodes and one for the edges (i.e., a support translation). The remaining of the embedding map specifies how names of the inner and outer interfaces should be mapped into the host link graph. Outer names can be mapped to any link; here injectivity is not required since a context can alias outer names. Dually, inner names can mapped to hyper-edges linking sets of points in the host link graph and such that every point is contained in at most one of these sets.

Definition 2 (Link graph embedding [11, Def 7.5.1])**.** *Let $G : X_G \to Y_G$ and $H : X_H \to Y_H$ be two concrete link graphs. A* link graph embedding *$\phi : G \hookrightarrow H$ is a map $\phi \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{e}} \uplus \phi^{\mathsf{i}} \uplus \phi^{\mathsf{o}}$ (assigning nodes, edges, inner and outer names respectively) subject to the following conditions:*

(LGE-1) *$\phi^{\mathsf{v}} : V_G \rightarrowtail V_H$ and $\phi^{\mathsf{e}} : E_G \rightarrowtail E_H$ are injective;*

(LGE-2) *$\phi^{\mathsf{i}} : X_G \rightarrowtail \wp(X_H \uplus P_H)$ is fully injective: $\forall x \neq x' : \phi^{\mathsf{i}}(x) \cap \phi^{\mathsf{i}}(x') = \emptyset$;*

(LGE-3) *$\phi^{\mathsf{o}} : Y_G \to E_H \uplus Y_H$ in an arbitrary partial map;*

(LGE-4) *$\mathit{img}(\phi^{\mathsf{e}}) \cap \mathit{img}(\phi^{\mathsf{o}}) = \emptyset$ and $\mathit{img}(\phi^{\mathsf{port}}) \cap \bigcup \mathit{img}(\phi^{\mathsf{i}}) = \emptyset$;*

(LGE-5) *$\phi^{\mathsf{p}} \circ \mathit{link}_G^{-1}\big|_{E_G} = \mathit{link}_H^{-1} \circ \phi^{\mathsf{e}}$;*

(LGE-6) *$\mathit{ctrl}_G = \mathit{ctrl}_H \circ \phi^{\mathsf{v}}$;*

(LGE-7) *$\forall p \in X_G \uplus P_G : \forall p' \in (\phi^{\mathsf{p}})(p) : (\phi^{\mathsf{h}} \circ \mathit{link}_G)(p) = \mathit{link}_h(p')$*

where $\phi^{\mathsf{p}} \triangleq \phi^{\mathsf{i}} \uplus \phi^{\mathsf{port}}$, $\phi^{\mathsf{h}} \triangleq \phi^{\mathsf{e}} \uplus \phi^{\mathsf{o}}$ and $\phi^{\mathsf{port}} : P_G \rightarrowtail P_H$ is $\phi^{\mathsf{port}}(v, i) \triangleq (\phi^{\mathsf{v}}(v), i)$.

The first three conditions are on the single sub-maps of the embedding. Condition (LGE-4) ensures that no components (except for outer names) are identified; condition (LGE-5) imposes that points connected by the image of an edge are all covered. Finally, conditions (LGE-6) and (LGE-7) ensure that the guest structure is preserved i.e. node controls and point linkings are preserved.

Place graph Like link graph embeddings, place graph embeddings are just a structure preserving injective map from nodes along with suitable maps for the inner and outer interfaces. In particular, a site is mapped to the set of sites and nodes that are "put under it" and a root is mapped to the host root or node that is "put over it" splitting the host place graphs in three parts: the guest image, the context and the parameter (which are above and below the guest image).

Definition 3 (Place graph embedding [11, Def 7.5.4])**.** *Let $G : n_G \to m_G$ and $H : n_H \to m_H$ be two concrete place graphs. A* place graph embedding *$\phi : G \hookrightarrow H$ is a map $\phi \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{s}} \uplus \phi^{\mathsf{r}}$ (assigning nodes, sites and regions respectively) subject to the following conditions:*

(PGE-1) *$\phi^{\mathsf{v}} : V_G \rightarrowtail V_H$ is injective;*

(PGE-2) *$\phi^{\mathsf{s}} : n_G \rightarrowtail \wp(n_H \uplus V_H)$ is fully injective;*

(PGE-3) *$\phi^{\mathsf{r}} : m_G \to V_H \uplus m_H$ in an arbitrary map;*

(PGE-4) *$\mathsf{img}(\phi^{\mathsf{v}}) \cap \mathsf{img}(\phi^{\mathsf{r}}) = \emptyset$ and $\mathsf{img}(\phi^{\mathsf{v}}) \cap \bigcup \mathsf{img}(\phi^{\mathsf{s}}) = \emptyset$;*

(PGE-5) *$\forall r \in m_G : \forall s \in n_G : \mathsf{prnt}^*_H \circ \phi^{\mathsf{r}}(r) \cap \phi^{\mathsf{s}}(s) = \emptyset$;*

(PGE-6) *$\phi^{\mathsf{c}} \circ \mathsf{prnt}_G^{-1}\big|_{V_G} = \mathsf{prnt}_H^{-1} \circ \phi^{\mathsf{v}}$;*

(PGE-7) *$\mathsf{ctrl}_G = \mathsf{ctrl}_H \circ \phi^{\mathsf{v}}$;*

(PGE-8) *$\forall c \in n_G \uplus V_G : \forall c' \in \phi^{\mathsf{c}}(c) : (\phi^{\mathsf{f}} \circ \mathsf{prnt}_G)(c) = \mathsf{prnt}_H(c')$;*

where $\mathsf{prnt}^*_H(c) = \bigcup_{i<\omega} \mathsf{prnt}^i(c)$, $\phi^{\mathsf{f}} \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{r}}$, and $\phi^{\mathsf{c}} \triangleq \phi^{\mathsf{v}} \uplus \phi^{\mathsf{s}}$.

Conditions in the above definition follows the structure of Definition 2, the main notable difference is (PGE-5) which states that the image of a root can not be the descendant of the image of another. Conditions (PGE-1), (PGE-2) and (PGE-3) are on the three sub-maps composing the embedding; conditions (PGE-4) and (PGE-5) ensure that no components are identified; (PGE-6) imposes surjectivity on children and the last two conditions require the guest structure to be preserved by the embedding map.

Bigraph Finally, bigraph embeddings can now be defined as maps being composed by an embedding for the link graph with one for the place graph consistently with the interplay of these two substructures. In particular, the interplay is captured by a single additional condition ensuring that points in the image of an inner names reside in the parameter defined by the place graph embedding (i.e. are inner names or ports of some node under a site image).

Definition 4 (Bigraph embedding [11, Def 7.5.14])**.** *Let $G : \langle n_G, X_G\rangle \to \langle m_G, Y_G\rangle$ and $H : \langle n_H, X_H\rangle \to \langle m_H, Y_H\rangle$ be two concrete bigraphs. A* bigraph embedding *$\phi : G \hookrightarrow H$ is a map given by a place graph embedding $\phi^{\mathsf{P}} : G^P \hookrightarrow H^P$ and a link graph embedding $\phi^{\mathsf{L}} : G^L \hookrightarrow H^L$ subject to the consistency condition:*

(BGE-1) *$\mathsf{img}(\phi^{\mathsf{i}}) \subseteq X_H \uplus \{(v, i) \in P_H \mid \exists s \in n_G : k \in \mathbb{N} : \mathsf{prnt}^k_H(v) \in \phi^{\mathsf{s}}(s)\}$.*

NP-completeness Despite their apparent complexity, the conditions maps have to satisfy to be considered bigraph embeddings may give some information and guidance in the construction of these maps. However the problem remains NP-complete as demonstrated in [3]. We recall their results to make this paper self contained. The authors focus on labelled forest embedding which covers the case of place graphs embeddings but not link graphs. In Section 4.1 we prove that the link graph embedding problem corresponds to an admissibility problem for a specific flow network. Therefore, the result presented in [3] will suffice to justify our approach.

To prove that the labelled forest pattern is NP-complete, in [3, §3] a reduction from 3-Sat is provided. The proposed reduction uses the RainbowAntichain problem as a middle step (introduced in *loc. cit.*). An instance of this problem is a tree $\mathcal{T}(\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$, and a finite set of colours $\mathcal{P}$, said palette. Some of the nodes in $\mathcal{T}$ have been coloured with one or more colours taken from $\mathcal{P}$. The problem asks to decide whatever exists a colourful subset of nodes $\mathcal{R} \subset \mathcal{V}$ where each colour c of $\mathcal{P}$ has exactly one representative node coloured with c and for no pair of $u, v \in \mathcal{R}$ of distinct nodes u is an ancestor of v.

Theorem 1 ([3, Th. 8])**.** *The* RainbowAntichain *problem is NP-complete.*

It is the straightforward to see that an instance $\mathcal{T}$, $\mathcal{P} = (c_0, \ldots, c_{n-1})$ of RainbowAntichain can be reduced to a forest pattern matching, namely, one that embeds the forest $(c_0[0], \ldots, c_{n-1}[n-1])$ – every tree has only a node, labelled with a colour of the palette, and a hole/site – into $\mathcal{T}$. This states that the forest pattern matching problem is NP-complete. Formally,

Theorem 2 ([3, Th. 9])**.** *The labelled forest embedding problem is NP-complete.*

This proves that deciding the existence of a place graph embedding (which can be seen as labelled forest pattern matching) of a given guest into a given host is NP-complete. Moreover, we are interested in listing all of them thus making CSP a viable approach.

4 Implementing the embedding problem in CSP

In this Section we present the main contribution of the paper i.e. a constraint satisfaction problem that models bigraph embedding problem. The encoding is based solely on integer linear constraints and is proven to be sound and complete.

Initially, we present the encoding for the link graph embedding problem and for the place graph embedding problem. Then we combine them providing some additional "gluing constraints" to ensure the consistency of the two sub-problems. The resulting encodings contains 34 constraint families (reflecting the size of the problem definition, cf. in Section 3) and hence taking advantage of the orthogonality of link and place structures is mandatory for the sake of both exposition and adequacy proofs. We shall remark that, despite the constraint families are quite numerous, the overall number of variables and constraints produced by the encoding is polynomially bounded with respect to the support cardinality of the involved bigraphs.

4.1 Link Graphs

Let us fix the guest and host concrete bigraphs: $G : X_G \to Y_G$ and $H : X_H \to Y_H$. We characterize the embeddings of G into H as the solutions of a suitable multi-flux problem which we denote as $\mathrm{LGE}[G, H]$. The main idea is to see the host points (i.e. ports and inner names) and handles (i.e. edges and outer names) as sources and sinks respectively: each point outputs a flux unit and each handle inputs one unit for each point it links. Units flows towards each point

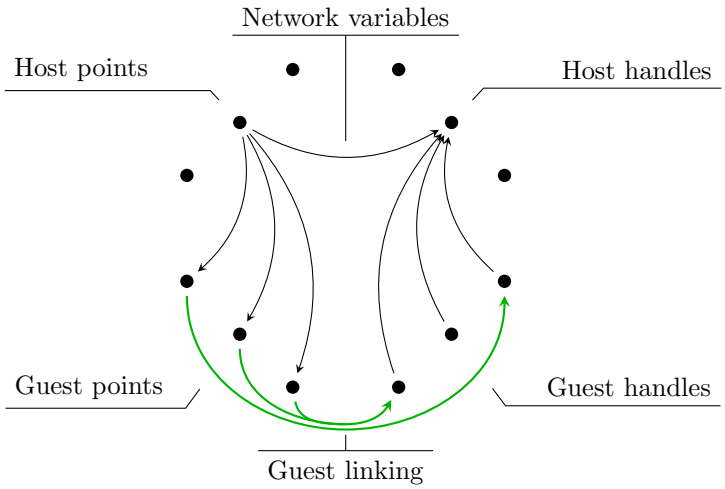


Figure 3: Schema of the multi-flux network encoding.

handle following H hyper-edges and optionally taking a "detour" along the linking structure of the guest G (provided that some conditions regarding structure preservation are met). Figure 3 and Figure 4 contain a sketch of the flux problem and its formal definition respectively.

The flux network reflects the linking structure and contains an edge connecting each point to its handle; these edges have an integer capacity limited to 1 and are represented by the variables defined in (2). The remaining edges of the network are organised in two complete biparted graphs: one between guest and host handles and one between guest and host points. Edges of the first sub-network are described by the variables in (1) and their capacity is bounded by the number of points linked by the host handle since this is the maximum acceptable flux and corresponds to the case where each point passes through the same hyper-edge of the guest link graph. Edges of the second sub-network are described by the variables in (3) and, like the first group of links, have their capacity limited to 1; to be precise, some of these variables will never assume a value different from 0 because host points can receive flux from anything but the host ports (as expressed by constraint (9)). Edges for the link structure of the guest are presented implicitly in the flux preservation constraints (see constraint (7)). In order to fulfil the injectivity conditions of link embeddings, some additional *flux variables* (whereas the previous are *network variables*) are defined by (4). These are used to keep track and separate each flux on the bases of the points handle[1].

The constraint families (5) and (6) define the outgoing and ingoing flux of host points and handles respectively. The firsts have to send exactly one unit considering every edge they are involved into and the seconds receive one unit for each of their point regardless if this unit comes from the point directly or from a handle of the guest.

The linking structure of the guest graph is encoded by the constraint family (7) which states that flux is preserved while passing through the guest i.e. the output of each handle has to match the overall input of the points it connects.

Constraints (8), (9), (18), (19) and (20) shape the flux in the sub-network linking guest and host points. Specifically, (8) requires that each point from the guest receives exactly one unit or, the other way round, that guest points are assigned with disjoint sets of points from the host.

[1]The problem can be presented without the additional flux variables, but we found this approach more readable.

$$N_{h,h'} \in \{0, \ldots, |\mathsf{link}_H(h')|\} \qquad h \in E_G \uplus Y_G,\ h' \in E_H \uplus Y_H \tag{1}$$

$$N_{p,h'} \in \{0,1\} \qquad h' \in E_H \uplus Y_H,\ p \in \mathsf{link}_H^{-1}(h') \tag{2}$$

$$N_{p,p'} \in \{0,1\} \qquad p' \in X_G \uplus P_G,\ p \in X_H \uplus P_H \tag{3}$$

$$F_{h,h'} \in \{0,1\} \qquad h \in E_G \uplus Y_G,\ h' \in E_H \uplus Y_H \tag{4}$$

$$\sum_k N_{p,k} = 1 \qquad p \in X_H \uplus P_H \tag{5}$$

$$\sum_k N_{k,h} = |\mathsf{link}_H(h)| \qquad h \in E_H \uplus Y_H \tag{6}$$

$$\sum_k N_{h,k} = \sum_{p \in \mathsf{link}_G^{-1}(h)} \sum_k N_{k,p} \qquad h \in E_G \uplus Y_G \tag{7}$$

$$\sum_k N_{k,p} = 1 \qquad p \in X_G \uplus P_G \tag{8}$$

$$N_{p,p'} = 0 \qquad p' \in P_G,\ p \in X_H \tag{9}$$

$$\frac{N_{h,h'}}{|\mathsf{link}_H^{-1}(h')|} \leq F_{h,h'} \leq N_{h,h'} \qquad \begin{array}{l} h \in E_G \uplus Y_G,\ h' \in E_H \uplus Y_H, \\ |\mathsf{link}_G^{-1}(h)| > 0,\ |\mathsf{link}_H^{-1}(h')| > 0 \end{array} \tag{10}$$

$$N_{p,p'} \leq F_{h,h'} \qquad \begin{array}{l} h \in E_G \uplus Y_G,\ h' \in E_H \uplus Y_H, \\ p \in \mathsf{link}_G^{-1}(h),\ p' \in \mathsf{link}_H^{-1}(h') \end{array} \tag{11}$$

$$F_{h,h'} \leq \sum_{\substack{p \in \mathsf{link}_G^{-1}(h) \\ p' \in \mathsf{link}_H^{-1}(h')}} N_{p,p'} \qquad \begin{array}{l} h \in E_G \uplus Y_G,\ h' \in E_H \uplus Y_H, \\ \mathsf{link}_G^{-1}(h) \cup \mathsf{link}_H^{-1}(h') \neq \emptyset \end{array} \tag{12}$$

$$\sum_k F_{h,k} = 1 \qquad h \in E_G \uplus Y_G \tag{13}$$

$$N_{p,h'} + F_{h,h'} \leq 1 \qquad h \in E_G,\ h' \in E_H \uplus Y_H,\ p \in \mathsf{link}_H^{-1}(h') \tag{14}$$

$$F_{h,h'} + F_{h'',h'} \leq 1 \qquad h \in E_G,\ h' \in Y_H,\ h'' \in Y_G \tag{15}$$

$$F_{h,h'} = 0 \qquad h \in E_G,\ h' \in Y_H \tag{16}$$

$$F_{h,h'} \leq 1 \qquad h \in E_G \uplus Y_G,\ h' \in E_H \tag{17}$$

$$N_{p,p'} = 0 \qquad \begin{array}{l} v \in V_G,\ v' \in V_H, \mathsf{ctrl}_G(v) = \mathsf{ctrl}_H(v) = c \\ i \neq i' \leq c,\ p = (v,i),\ p' = (v',i') \end{array} \tag{18}$$

$$N_{p,p'} = 0 \qquad \begin{array}{l} v \in V_G,\ v' \in V_H, \mathsf{ctrl}_G(v) \neq \mathsf{ctrl}_H(v), \\ p = (v,i),\ p' = (v',i') \end{array} \tag{19}$$

$$\sum_{j \leq c} N_{(v,j),(v',j)} = c \cdot N_{p,p'} \qquad \begin{array}{l} v \in V_G,\ v' \in V_H, \mathsf{ctrl}_G(v) = \mathsf{ctrl}_H(v) = c, \\ i \leq c,\ p = (v,i),\ p' = (v',i) \end{array} \tag{20}$$

Figure 4: Constraints of LGE$[G, H]$.

Constraints (9), (18) and (19) disable edges between guest ports and host inner names, between mismatching ports of matching nodes and between ports of mismatching nodes. Finally, the flux of ports belonging to the same node has to act compactly i.e. if there is flux between the i-th ports of two nodes, then, there should be flux between every other matching ports as expressed by (20).

Constraints (10), (11) and (12) relates flux and network variables ensuring that the formers assume a true value if, and only, if there is actual flux between the corresponding guest and host handles. In particular, (11) propagates the information about the absence of flux between handles disabling the sub-network linking handles points and, *vice versa*, (12) propagates the information in the other way disabling flux between handles if there is no flux between their points.

The remaining constraints prevent fluxes from mixing. Constraint (13) requires guest handles to send their output to exactly one destination thus renders the sub-network between handles a function assigning guest handles to host handles. This mapping is subject to some additional conditions when edges are involved: (16) and (17) ensure that the edges are injectively mapped to edges only, (15) forbids host outer names to receive flux from an edge and an outer name at the same time. Finally, constraint (14) states that the output of host points cannot bypass the guest if there is flux between its handle and an edge from the guest.

Adequacy Let $\vec{N}$ be a solution of $\mathrm{LGE}[G, H]$. The corresponding link graph embedding $\phi : G \hookrightarrow H$ is defined as follows:

$$\begin{aligned}
\phi^{\mathsf{v}}(v) &\triangleq v' \in V_H : \exists i : N_{(v,i),(v',i)} = 1 \\
\phi^{\mathsf{e}}(e) &\triangleq e' \in E_H : F_{e,e'} = 1 \\
\phi^{\mathsf{o}}(y) &\triangleq y' \in Y_H : F_{y,y'} = 1 \\
\phi^{\mathsf{i}}(x) &\triangleq \{x' \in X_H \uplus P_H \mid N_{x',x} = 1\}
\end{aligned}$$

The components of ϕ just defined are well-given and compliant with Definition 2. On the other way round, let $\phi : G \hookrightarrow H$ be a link graph embedding. The corresponding solution $\vec{N}$ of $\mathrm{LGE}[G, H]$ is defined as follows:

$$N_{p,p'} \triangleq \begin{cases} 1 & \text{if } p' \in X_G \wedge p \in \phi^{\mathsf{i}}(p') \\ 1 & \text{if } p' = (v, i) \wedge p = (\phi^{\mathsf{v}}(v), i) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{p,h'} \triangleq \begin{cases} 1 & \text{if } h' = \mathsf{link}_H p \wedge \nexists p' : N_{p,p'} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{h,h'} \triangleq \begin{cases} 1 & \text{if } h \in E_H \wedge h' \in E_G \wedge h = \phi^{\mathsf{e}}(h') \\ 1 & \text{if } h \in Y_H \wedge h' \wedge h = \phi^{\mathsf{o}}(h') \\ 0 & \text{otherwise} \end{cases}$$

Clearly $F_{h,h'} = 1 \overset{\triangle}{\iff} N_{h,h'} \neq 0$. Then it is easy to check that every constraint of $\mathrm{LGE}[G, H]$ is satisfied by the solution just defined.

The constraint satisfaction problem in Figure 4 is sound and complete with respect to the link graph embedding problem given in Definition 2.

Proposition 3 (Adequacy of LGE)**.** *For any two concrete link graphs G and H, there is a bijective correspondence between the link graph embeddings of G into H and the solutions of* $\mathrm{LGE}[G, H]$.

4.2 Place Graphs

Let us fix the guest and host place graphs: $G : n_G \to m_G$ and $H : n_H \to m_H$. We characterize the embeddings of G into H as the solutions of the constraint satisfaction problem in Figure 5.

$$M_{h,g} \in \{0,1\} \qquad g \in n_G \uplus V_G \uplus m_G,\ h \in n_H \uplus V_H \uplus m_H \tag{21}$$

$$M_{h,g} = 0 \qquad g \in n_G \uplus V_G,\ h \in m_H \tag{22}$$

$$M_{h,g} = 0 \qquad g \in V_G \uplus m_G,\ h \in n_H \tag{23}$$

$$M_{h,g} = 0 \qquad g \in V_G,\ h \in V_H,\ \mathsf{ctrl}_G(g) \neq \mathsf{ctrl}_H(h) \tag{24}$$

$$M_{h,g} = 0 \qquad g \in m_G,\ h \notin m_H,\ v \in \mathsf{prnt}^*_H(h) \cap V_G,\ \mathsf{ctrl}_G(v) \notin \Sigma_a \tag{25}$$

$$M_{h,g} \leq M_{h',g'} \qquad g \notin m_G,\ g' \in \mathsf{prnt}_G(g),\ h \notin m_H,\ h' \in \mathsf{prnt}_H(h) \tag{26}$$

$$\sum_{h \in V_H \uplus m_H} M_{h,g} = 1 \qquad g \in m_G \tag{27}$$

$$\sum_{h \in n_H \uplus V_H} M_{h,g} = 1 \qquad g \in V_G \tag{28}$$

$$m_G \cdot \sum_{g \in n_G \uplus V_G} M_{h,g} + \sum_{g \in m_G} M_{h,g} \qquad h \in V_H \tag{29}$$

$$|\mathsf{prnt}_H^{-1}(h)| \cdot M_{h,g} \leq \sum_{\substack{h' \in \mathsf{prnt}_H^{-1}(h),\\ g' \in \mathsf{prnt}_G^{-1}(g)}} M_{h',g'} \qquad g \in V_G,\ h \in V_H \tag{30}$$

$$|\mathsf{prnt}_G^{-1}(g) \setminus n_G| \cdot M_{h,g} \leq \sum_{\substack{h' \in \mathsf{prnt}_H^{-1}(h) \setminus n_h,\\ g' \in \mathsf{prnt}_G^{-1}(g) \setminus n_g}} M_{h',g'} \qquad g \in m_G,\ h \in V_H \tag{31}$$

$$M_{h,g} + \sum_{\substack{h' \in \mathsf{prnt}^*_H(h),\\ g' \in m_G}} M_{h',g'} \leq 1 \qquad g \in V_G,\ h \in V_H \tag{32}$$

Figure 5: Constraints of $\mathrm{PGE}[G, H]$.

The problem is a direct encoding of Definition 3 as a matching problem presented, as usual, as a biparted graph. Sites, nodes and roots of the two place graphs are represented as nodes and parted into the guest and the host ones. For convenience of exposition, graph is complete.

Edges are modelled by the boolean variables defined in (21); these are the only variables used by the problem. So far a solution is nothing more than a relation between the components of guest and host containing only those pairs connected by an edge assigned a non-zero value. To capture exactly those assignments that are actual place graph embeddings some conditions have to be imposed.

Constraints (22) and (23) prevent roots and sites from the host to be matched with nodes or sites and nodes or roots respectively. (24) disables matching between nodes decorated with different controls. Constraint (25) prevents any matching for host nodes under a passive context (i.e. have an ancestor labelled with a passive control). (26) propagates the matching along the parent map from children to parents. Constraints (27) and (28) ensure that the matching is a function when restricted to guest nodes and roots (the codomain restriction follows by (22) and

(23)). (29) says that if a node from the host cannot be matched with a root or a node/site from the guest at the same time; moreover, if the host node is matched with a node then it cannot be matched to anything else.

The remaining constraints are the counterpart of (26) and propagate matchings from parents to children. (30) applies on matchings between nodes and says that if parents are matched, then children from the host node are covered by children from the guest node. In particular, the matching is a perfect assignment when restricted to guest children that are nodes (because of (29)) and is a surjection on those that are sites. (31) imposes a similar condition on matchings between guest roots and host nodes. Specifically, it says that the matching have to cover child nodes from the guest (moreover, it is injective on them) leaving child sites to match whatever remains ranging from nothing to all unmatched children. Finally, (32) prevent matching from happening inside a parameter.

Adequacy Let $\vec{M}$ be a solution of $\mathrm{PGE}[G, H]$. The corresponding place graph embedding $\phi : G \hookrightarrow H$ is defined as follows:

$$\begin{aligned}
\phi^{\mathsf{v}}(g) &\triangleq h \in V_H : \exists i : M_{h,g} = 1 \\
\phi^{\mathsf{s}}(g) &\triangleq \{h \in n_h \uplus V_H \mid M_{h,g} = 1\} \\
\phi^{\mathsf{r}}(g) &\triangleq h \in m_H \uplus V_H : M_{h,g} = 1
\end{aligned}$$

The components of ϕ just defined are well-given and compliant with Definition 3.

On the opposite direction, let $\phi : G \hookrightarrow H$ be a place graph embedding. The corresponding solution $\vec{M}$ of $\mathrm{PGE}[G, H]$ is defined as follows:

$$M_{h,g} \triangleq \begin{cases} 1 & \text{if } g \in V_G \wedge h = \phi^{\mathsf{v}}(g) \\ 1 & \text{if } g \in m_G \wedge h = \phi^{\mathsf{r}}(g) \\ 1 & \text{if } g \in n_G \wedge h \in \phi^{\mathsf{s}}(g) \\ 0 & \text{otherwise} \end{cases}$$

It is easy to check that every constraint of $\mathrm{PGE}[G, H]$ is satisfied by the solution just defined.

The constraint satisfaction problem in Figure 5 is sound and complete with respect to the place graph embedding problem given in Definition 3.

Proposition 4 (Adequacy of PGE)**.** *For any two concrete place graphs G and H, there is a bijective correspondence between the place graph embeddings of G into H and the solutions of* $\mathrm{PGE}[G, H]$.

4.3 Bigraphs

Let $G : \langle n_G, X_G \rangle \to \langle m_G, Y_G \rangle$ and $H : \langle n_H, X_H \rangle \to \langle m_H, Y_H \rangle$ be two concrete bigraphs. By taking advantage of the orthogonality of the link and place structure we can define the constraint satisfaction problem capturing bigraph embeddings by simply composing the constraints given above for the link and place graph embeddings and by adding just two consistency constraints to relate the solutions of the two problems.

These additional constraint families are reported in Figure 6. The family (33) ensures that solutions for $\mathrm{LGE}[G, H]$ and $\mathrm{PGE}[G, H]$ agree on nodes since the map ϕ^{v} has to be shared by the corresponding link and place embeddings. The family (34) ensures that ports are in the image of inner names (i.e. send their flux unit to them) only if their node is part of the parameter i.e. only if it is matched to a site from the guest or it descends from a node that is so.

$$M_{v,v'} = N_{p,p'} \qquad v \in V_H,\ v' \in V_G,\ p = (v,k),\ p' = (v',k) \tag{33}$$

$$\sum_{p' \in X_G} N_{p,p'} \leq \sum_{\substack{h \in \mathsf{prnt}^*_H(v), \\ g \in n_G}} M_{h,g} \qquad v \in V_H,\ p = (v,k) \tag{34}$$

Figure 6: Constraints of BGE$[G, H]$.

It is easy to check that (34) corresponds exactly to condition (BGE-1). Therefore, from Proposition 3 and Proposition 4, the constraint satisfaction problem defined by Figures 4, Figure 5 and Figure 6 is sound and complete with respect to the bigraph embedding problem given in Definition 4 as stated by below.

Proposition 5 (Adequacy of BGE)**.** *For any two concrete bigraphs G and H, there is a bijective correspondence between the bigraph embeddings of G into H and the solutions of* BGE$[G, H]$.

5 Conclusions and future works

In this paper, we have presented a sound and complete algorithm for solving the bigraph embedding problem, based on a constraint satisfaction problem. The resulting model is compact and composed by a number of variables and linear constraints, polynomially bounded by the size of the guest and host bigraphs. Remarkably, this algorithm does not require hosts to be ground.

The CSP approach offers a great flexibility, e.g. allowing to move execution strategies upstream into CSP calculation. An example can be found in this paper: in practice, the notion of active/passive contexts defines an execution strategy.

This approach naturally suggests several interesting extensions of the bigraphic model itself. We can think of *weighted* bigraphs, where nodes and edges can be given "weights", and firing a rule has a cost related to the nodes involved. This extension would lead us to consider "approximated embeddings": a guest can be embedded up-to some distance based on costs (e.g. missing controls, or quantitative differences between controls,...). In both cases, our implementation can be straightforwardly adapted just by adding a cost function, in order to give the optimal matchings; this would be far more efficient and easier to implement than dealing with these issues at the level of strategies (if possible at all).

The proposed approach can be easily applied also to extensions of the bigraphs, e.g. directed bigraphs [10], bigraphs with sharing [6] or local bigraphs. An interesting direction would be to extend the algorithm also to stochastic and probabilistic bigraphs [15] which will offer useful modelling and verification tools for quantitative aspects, e.g. for biological problems [2, 8].

The algorithm has been successfully integrated into LibBig, an extensible library for manipulating bigraphical reactive systems. This library can be used for implementing a wide range of tools and it can be adapted to support several extensions of bigraphs. Approximated and weighted embeddings are supported too but are still an experimental feature. In fact, the theoretical foundations and implications of these extensions have not been fully investigated yet suggesting another line of research.

The empirical evaluation of the implementation available in LibBig looks promising. It cannot be considered a rigorous experimental validation mainly because performance depends on the solver, and the model is not optimized for any particular solver. Moreover, up to now there are no "official" (or "widely recognized") benchmarks for the bigraph embedding problem. An algorithm for the bigraph matching problem is proposed in [7], based on a translation into a term

matching problem; this algorithm is at the core of BPLTool. In [25] Sevegnani *et. al.* presented a SAT based algorithm for deciding the matching problem in bigraphs with sharing. To the best of the authors knowledge, the approach of [25] is the nearest to the solution presented in this paper. However, an accurate and fair comparison of algorithms for computing bigraph embeddings/matchings is difficult because of the aforementioned reasons and because of the different bigraph variants these algorithms deal with (e.g., the matching algorithm of the model checker BigMC does not support inner names).

A decentralized algorithm for computing bigraphical embeddings has been proposed in [17] and is at the core of the *distributed bigraphical machine*.

Acknowledgements We thank Alessio Mansutti and the participants to MeMo'14 for fruitful discussions on preliminary version of this paper. This work is partially supported by MIUR PRIN project 2010LHT4KM, *CINA*.

References

[1] G. Bacci and D. Grohmann. On the decidability of bigraphical sorting. In M. Haveraaen, M. Lenisa, J. Power, and M. Seisenberger, editors, *Proc. CALCO Young Researchers Workshop*, number 05/2010 in Technical Report, pages 1–14, 2009.

[2] G. Bacci, D. Grohmann, and M. Miculan. Bigraphical models for protein and membrane interactions. In G. Ciobanu, editor, *Proc. MeCBIC*, volume 11 of *Electronic Proceedings in Theoretical Computer Science*, pages 3–18, 2009.

[3] G. Bacci, M. Miculan, and R. Rizzi. Finding a forest in a tree. In *Proc. TGC*, Lecture Notes in Computer Science. Springer, 2014.

[4] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, and H. Niss. Bigraphical models of context-aware systems. In L. Aceto and A. Ingólfsdóttir, editors, *Proc. FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2006.

[5] M. Bundgaard, A. J. Glenstrup, T. T. Hildebrandt, E. Højsgaard, and H. Niss. Formalizing higher-order mobile embedded business processes with binding bigraphs. In D. Lea and G. Zavattaro, editors, *Proc. COORDINATION*, volume 5052 of *Lecture Notes in Computer Science*, pages 83–99. Springer, 2008.

[6] M. Calder and M. Sevegnani. Process algebra for event-driven runtime verification: A case study of wireless network management. In J. Derrick, S. Gnesi, D. Latella, and H. Treharne, editors, *Proc. IFM*, volume 7321 of *Lecture Notes in Computer Science*, pages 21–23. Springer, 2012.

[7] T. C. Damgaard, A. J. Glenstrup, L. Birkedal, and R. Milner. An inductive characterization of matching in binding bigraphs. *Formal Aspects of Computing*, 25(2):257–288, 2013.

[8] T. C. Damgaard, E. Højsgaard, and J. Krivine. Formal cellular machinery. *Electronic Notes in Theoretical Computer Science*, 284:55–74, 2012.

[9] A. J. Faithfull, G. Perrone, and T. T. Hildebrandt. BigRed: A development environment for bigraphs. *ECEASST*, 61, 2013.

[10] D. Grohmann and M. Miculan. Directed bigraphs. In *Proc. MFPS*, volume 173 of *Electronic Notes in Theoretical Computer Science*, pages 121–137. Elsevier, 2007.

[11] E. Højsgaard. *Bigraphical Languages and their Simulation.* PhD thesis, IT University of Copenhagen, 2012.

[12] J. E. Hopcroft and R. E. Tarjan. A v^2 algorithm for determining isomorphism of planar graphs. *Inf. Process. Lett.*, 1(1):32–34, 1971.

[13] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of Conference Record of 6th Annual ACM Symposium on Theory of Computing, (STOC '74)*, pages 172–184, 1974.

[14] O. H. Jensen and R. Milner. Bigraphs and transitions. In A. Aiken and G. Morrisett, editors, *POPL*, pages 38–49. ACM, 2003.

[15] J. Krivine, R. Milner, and A. Troina. Stochastic bigraphs. In *Proc. MFPS*, volume 218 of *Electronic Notes in Theoretical Computer Science*, pages 73–96, 2008.

[16] A. Mansutti, M. Miculan, and M. Peressotti. Multi-agent systems design and prototyping with bigraphical reactive systems. In K. Magoutis and P. Pietzuch, editors, *Proc. DAIS*, volume 8460 of *Lecture Notes in Computer Science*, pages 201–208. Springer, 2014.

[17] A. Mansutti, M. Miculan, and M. Peressotti. Towards distributed bigraphical reactive systems. In R. Echahed, A. Habel, and M. Mosbah, editors, *Proc. GCM'14*, page 45, 2014. Workshop version.

[18] M. Miculan and M. Peressotti. Bigraphs reloaded: a presheaf presentation. Technical Report UDMI/01/2013, Dept. of Mathematics and Computer Science, Univ. of Udine, 2013.

[19] R. Milner. *The Space and Motion of Communicating Agents.* Cambridge University Press, 2009.

[20] E. Pereira, C. M. Kirsch, J. B. de Sousa, and R. Sengupta. BigActors: a model for structure-aware computation. In C. Lu, P. R. Kumar, and R. Stoleru, editors, *ICCPS*, pages 199–208. ACM, 2013.

[21] G. Perrone. *Domain-Specific Modelling Languages in Bigraphs.* PhD thesis, IT University of Copenhagen, 2013.

[22] G. Perrone, S. Debois, and T. T. Hildebrandt. Bigraphical refinement. In J. Derrick, E. A. Boiten, and S. Reeves, editors, *Proc. REFINE*, volume 55 of *Electronic Proceedings in Theoretical Computer Science*, pages 20–36, 2011.

[23] G. Perrone, S. Debois, and T. T. Hildebrandt. A model checker for bigraphs. In S. Ossowski and P. Lecca, editors, *Proc. SAC*, pages 1320–1325. ACM, 2012.

[24] G. Rozenberg, editor. *Handbook of graph grammars and computing by graph transformation*, volume 1. World Scientific, River Edge, NJ, USA, 1997.

[25] M. Sevegnani, C. Unsworth, and M. Calder. A SAT based algorithm for the matching problem in bigraphs with sharing. Technical Report TR-2010-311, Department of Computer Science, University of Glasgow, 2010.

A NP-completeness of the bigraph embedding problem

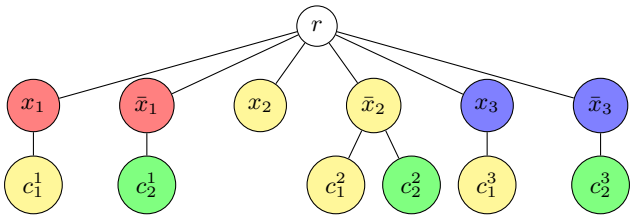
In [3] the authors proved that the labelled forest embedding problem is NP-Complete. This result covers the case of place graphs embeddings but not link graphs. However, the latter correspond to an admissibility problem for a specific flow network and hence their results will suffice to justify our approach.

To prove that the labelled forest pattern is NP-complete, in [3, §3] a reduction from 3-SAT is provided. The proposed reduction uses the RAINBOWANTICHAIN problem as a middle step. An instance of this problem (introduced in [3]) is a tree $\mathcal{T}(\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$, and a finite set of colours $\mathcal{P}$, said palette. Some of the nodes in $\mathcal{T}$ have been coloured with one or more colours taken from $\mathcal{P}$. The problem asks to decide whatever exists a colourful subset of nodes $\mathcal{R} \subset \mathcal{V}$ where each colour c of $\mathcal{P}$ has exactly one representative node coloured with c and for no pair of $u, v \in \mathcal{R}$ of distinct nodes u is an ancestor of v.

Theorem 6 ([3, Th. 8])**.** *The* RAINBOWANTICHAIN *problem is NP-complete.*

Proof. RAINBOWANTICHAIN is in NP, since, given a set of nodes $\mathcal{R}$, checking whatever $\mathcal{R}$ us a rainbow anti-chain for $\mathcal{T}$ can be easily done in polynomial time by breadth-first visit of $\mathcal{T}$, and for each $v \in \mathcal{R}$ found, firsti increase the node counter nc, then the colour counter $p[i]$ (with $1 \leq i \leq |\mathcal{P}|$) if v has a colour $c_i \in \mathcal{P}$. The check fails whether $nc > |\mathcal{P}|$ or $p[j] = 0$ for some j, otherwise $\mathcal{R}$ is a rainbow anti-chain for $\mathcal{T}$.

Let $C = \{c_1, \dots c_m\}$ be an instance of 3-SAT on variables $\{x_1, \dots, x_n\}$. From C we define a coloured tree $\mathcal{T}$ as follows. Let r be the root node which is left uncoloured. For each variable x_i let x_i and $\bar{x}_i$ be child nodes of r, and color them with fresh colour c_{x_i}, distinct for each variable. For each clause $c_j \in C$. let c_j^1, c_j^2 and c_j^3 be children nodes of l_i in $\mathcal{T}$ if c_j contains l_i as negated, and assign to each of them a fresh colour c_{c_j}, distinct for each clause. An example of construction for $c_1 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3), c_2 = (x_1 \vee x_2 \vee x_3)$ is shown below.



Let φ be a truth assignment satisfying the formula C. By construction, selecting only literal nodes l_i which are satisfied by φ, we obtain a rainbow anti-chain $\mathcal{R}'$ in $\mathcal{T}$ for the palette $\{c_{x_i} \mid 1 \leq i \leq n\}$. Now, we extend $\mathcal{R}'$ to $\mathcal{R}$ adding all clause nodes which are not children of an element in $\mathcal{R}'$. Such $\mathcal{R}$ is clearly an anti-chain for $\mathcal{T}$, but we must ensure that is colourful and no more than one representative per colour is taken. To do this, it suffices to prove that $\mathcal{R}$ is colourful, indeed if a colour occurs more than one in $\mathcal{R}$ we remove the others. By hypothesis, each clause c_J is satisfied by φ, hencec_J has at least one literal l_i such that $\varphi(l_i)$ is true. By construction of $\mathcal{T}$, there exist a node c_j^k, with $1 \leq k \leq 3$, child of $\bar{l}_i$, hence already in $\mathcal{R}$. This holds for all clauses c_j, hence $\mathcal{R}$ is colourful.

Controversy, let $\mathcal{R}$ be a rainbow anti-chain for $\mathcal{T}$. Let the boolean function φ over $\{x_1, \dots, x_n\}$ be defined by $\varphi(x_i) = \mathbb{T}$ if x_i is a node in $\mathcal{R}$ and $\varphi(x_i) = \mathbb{F}$ otherwise. Since $\mathcal{R}$ has exactly one representative per colour, no opposite literals are in $\mathcal{R}$, hence φ is a truth assignment for C. By colourfulness of $\mathcal{R}$, for all colours c_{c_J} $(1 \leq j \leq m)$ there exists a node $c_j^k \in \mathcal{R}$ $(1 \leq k \leq 3)$ such that c_j^k has a colour c_{c_j}. By construction on $\mathcal{T}$, each $c_j^K \in \mathcal{R}$ is a children of a literal node $l_i \notin \mathcal{R}$,

and moreover the clause c_j contains $\bar{l}_j$. Since $l_j \notin \mathcal{R}$, by definition $\varphi(\bar{l}_i) = \mathbb{T}$, hence $\varphi(c_j) = \mathbb{T}$. This holds for all $1 \leq j \leq m$, hence φ satisfies C. □

In [3] labelled trees are described by terms of a language inspired to the ambient calculus and quotiented by the usual structural congruence. Then ambients corresponds to labelled subtrees; the null process to the empty tree; variables are leaves; parallel processes to siblings (with the additional requirement for disjoint variables to ensure a tree structure). This allows to represent grafting as (simultaneous) substitution. An embedding of a labelled forest (with variables $\vec{Z}$) $\vec{S}(\vec{Z})$ into a tree T (denoted as $T \succeq \vec{S}$) can be described by a tree $C(\vec{X})$, said context, and a forest $\vec{D}$, whose trees are called parameters, such that $T \equiv (C\{\vec{S}/\vec{X}\})\{\vec{D}/\vec{Z}\}$.

It is the straightforward to see that an instance $\mathcal{T}$, $\mathcal{P} = (c_0, \ldots, c_{n-1})$ of RainbowAntichain can be reduced to a forest pattern matching, namely, one that embeds the forest $(c_0[0], \ldots, c_{n-1}[n-1])$ – every tree has only a node, labelled with a colour of the palette, and a hole/site – into $\mathcal{T}$. This states that the forest pattern matching problem is NP-complete. Formally,

Theorem 7 ([3, Th. 9])**.** *The labelled forest embedding problem is NP-complete.*

Proof. Given a solution $(C, \vec{D})$ for $T \succeq \vec{S}$, checking that $T \equiv (C\{\vec{S}/\vec{X}\})\{\vec{D}/\vec{Z}\}$ corresponds to a tree isomorphism test, which is in P for [12, 13]. Let a coloured tree $\mathcal{T}$ and a palette $\mathcal{P} = \{c_1, \ldots, c_n\}$ be and instance of the RainbowAntichain problem. Let us transform $\mathcal{T}$ into a tree term T as follows. If T is a single node v (a leaf) T is the empty tree labelled with m ($m[\mathsf{nil}]$) where $m = c$ if v has color c, otherwise $m = *$, a fresh name not in $\mathcal{P}$ denoting an uncoloured node. If $\mathcal{T}$ has root r and $\mathcal{T}_1, \ldots, \mathcal{T}_k$ are the (children) subtrees of r, T is $m[T_1| \ldots |T_k]$, where m is as above for r, and $T_1, \ldots, T_k$ are transformed trees of $\mathcal{T}_1, \ldots, \mathcal{T}_k$. Suppose $(C, \vec{D})$ be a solution for $T \succeq \vec{S} = (c_1[x_1], \ldots, c_k[x_n])$. In C, each $c_i[x_i]$ is grafted into a variable $z_i \in vars(C)$. Since variables can appear in terms only as leaves, in the transformation $\mathcal{T}$ of T, we have found a rainbow anti-chain for $\mathcal{P}$, since the matching forest $\vec{S}$ has all the colors in $\mathcal{P}$ exactly once.

Assume that $\mathcal{T}$ has a rainbow anti-chain $\mathcal{R}$. In order to recover context C and parameters $\vec{D}$, which are a solution for $T \succeq (c_1[x_1], \ldots, c_k[x_n])$, it suffices to apply the construction explained above with some adjustments: we obtain C applying the transformation from the root of T, but if a node in $\mathcal{R}$ is reached it is transformed by a fresh variable z_i $(1 \leq i \leq n)$ one for each element in $\mathcal{R}$; D_j's are recovered applying the original transformation starting from the subtrees rooted at the children of nodes in $\mathcal{R}$. It is straightforward to prove that $T \equiv (C\{c_1[x_1]/z_1, \ldots, c_k[x_n]/z_n\})\{\vec{D}/\vec{X}\}$, for $X = \{x_1, \ldots, x_n\}$. □

This proves that deciding the existence of a place graph embedding (which can be seen as labelled forest pattern matching) of a given redex into an agent is NP-complete. Moreover, we are interested in listing all of them thus making CSP a viable approach.