Concise Graphs and Functional Bisimulations

Ling Cheung and Jesse Hughes

Department of Computer Science, University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
Email: {lcheung,jesseh}@cs.kun.nl

Abstract

We investigate the conditions under which least bisimulations exist with respect to set inclusion. In particular, we describe a natural way to remove redundant pairs from a given bisimulation. We then introduce the *conciseness* property on process graphs, which characterizes the existence of least bisimulations under the aforementioned method.

Subsequently, we consider the category of process graphs and functional bisimulations. This category has all coequalizers. Binary products and coproducts can be constructed with some further assumptions. Moreover, the full subcategory of concise graphs is a reflective subcategory.

Keywords: functional bisimulation, process graph, least bisimulation, concise graph, product, quotient graph

1 Introduction

In [1], Zena M. Ariola and Jan Willem Klop investigated structural features of term graphs and functional bisimulations. There they defined an order relation $\leq_{\mathsf{FB}}$ on the collection of term graphs:

$$G \leq_{\mathsf{FB}} H \Leftrightarrow \exists \text{ functional bisimulation } f : G \longrightarrow H .$$

It was shown that $\leq_{\mathsf{FB}}$ is a partial order (up to graph isomorphism). More surprisingly, for any term graph G, the collection of all term graphs bisimilar to G form a complete lattice with respect to $\leq_{\mathsf{FB}}$.

The research in this paper began as an exercise to generalize these results to process graphs. We follow Ariola and Klop in taking functional bisimulations as our morphisms (although, unlike them, we do not investigate the related skeletal category). This yields the category P. Functional bisimulations seem to be an interesting (if non-traditional) choice, because they are closely related to history relations. Indeed, the $\leq_{\mathsf{FB}}$ relation of ibid corresponds to the opposite of $\leq_{\mathsf{H}}$ in [2]. In fact, a direct application of Lynch and

This is a preliminary version. The final version will be published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Vaandrager's Proposition 5.4 yields: There is a functional bisimulation $A \twoheadrightarrow B$ iff A is essentially obtained by adding a history variable to B. The function $A \twoheadrightarrow B$ is the effect of "forgetting" that variable.

Having taken history relations as our starting point, we investigate the basic features of the resulting category. Our aim is to define a product (with respect to functional bisimulations) of two process graphs via minimal bisimulation, and a coproduct via a quotient of the corresponding coproduct in **Set** (the category of sets and functions).

Since the structure of term graphs is much more rigid than that of process graphs, we encounter some non-trivial difficulties, among which the existence of a suitable minimal bisimulation between two bisimilar process graphs. These graphs may fail to have any minimal bisimulation between them (Fig. 1), or there may be non-isomorphic minimal bisimulations (Fig. 2).

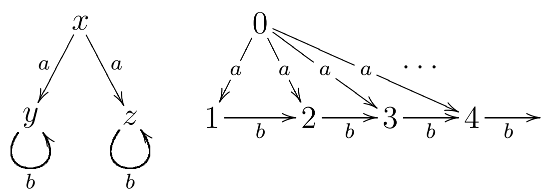


Fig. 1. No minimal bisimulation.

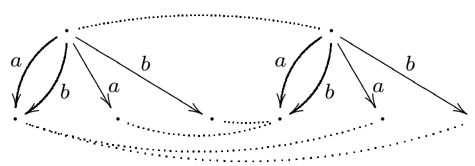


Fig. 2. Non-isomorphic minimal bisimulations: R (indicated by dotted lines) and the identity relation Δ.

We solve this problem by introducing the notion of *concise* graphs (Sect. 3). If G is concise, then one can construct the least bisimulation between G and H for any bisimilar H (which need not be concise). More precisely, we start with any bisimulation R between G and H and remove the pairs that are not reachable when R is given the transition structure described in [1].

We devote Sect. 4 to understanding basic features of P. This category has all coequalizers; hence, given a bisimulation R on process graph G, we can construct the quotient process G/R, using the least equivalence relation generated by R. We use this fact to construct binary coproducts of bisimilar graphs, provided one of the graphs is concise. Lastly, we move to the subcategory of *restricted* graphs and construct binary products under similar assumptions.

In Section 5, we prove that the full subcategory of concise graphs is a reflective subcategory of P. Given an arbitrary process graph G, there is a "best" way to identify nodes in G so that the result is concise. This operation can be viewed as a closure operator on (the skeleton of) P and the subcategory of concise graphs corresponds to the closure system generated by this operator. In general, a reflective subcategory is the categorical generalization of closure systems in posets (see discussion in [3]).

Section 6 explores the situation without conciseness. We show, by Zorn's lemma, that minimal bisimulations exist between image finite process graphs. However, there is no uniqueness guaranteed.

Section 7 discusses briefly the prospects of checking conciseness. We propose a modified definition called *obvious conciseness*, which allows for more efficient checking.

2 Preliminaries

Process graphs are labeled transition systems. (We assume an alphabet $\mathcal{A}$ of action labels.) Explicitly, a process graph is a triple

$$G = \langle G,\, e_G\colon G \longrightarrow \mathcal{P}(G)^{\mathcal{A}},\, \mathsf{roots}(\mathrm{G}) \subseteq G\rangle.$$

Elements of the set G are the *nodes* of process graph, denoted s, t, u, v, etc. We denote the actions (elements of $\mathcal{A}$) by a, b, c, etc. and write $s \xrightarrow{a} t$ if $t \in e_G(s)(a)$.

A *bisimulation* between two process graphs G and H is a relation $R \subseteq G \times H$ satisfying:

(i) For all $\langle s,\, s'\rangle \in R$, if $s \xrightarrow{a} t$ in G then there is a $t' \in H$ such that $s' \xrightarrow{a} t'$ and $\langle t,\, t'\rangle \in R$.

(ii) For all $\langle s,\, s'\rangle \in R$, if $s' \xrightarrow{a} t'$ in H then there is a $t \in G$ such that $s \xrightarrow{a} t$ and $\langle t,\, t'\rangle \in R$.

(iii) For all $r \in \mathsf{roots}(\mathrm{G})$, there is an $r' \in \mathsf{roots}(\mathrm{H})$ such that $\langle r,\, r'\rangle \in R$.

(iv) For all $r' \in \mathsf{roots}(\mathrm{H})$, there is an $r \in \mathsf{roots}(\mathrm{G})$ such that $\langle r,\, r'\rangle \in R$.

We say that G and H are *bisimilar* just in case there is a bisimulation between them. An interested reader can refer to [4] for an introduction to various semantics of concurrency including bisimilarity.

Process graphs are evidently coalgebras for the functor $FX = \mathcal{P}(X)^{\mathcal{A}}$, together with the extra structure of designated roots. Viewed in this way, a bisimulation is the same as an F-bisimulation (in the coalgebraic sense) satisfying (iii) and (iv). Note: one may be tempted to use the isomorphism between subsets of a set G and arrows $G \to 2$ to represent a process graph G as a coalgebra

$$G \xrightarrow{[e_G, \mathsf{roots}(\mathrm{G})]} \mathcal{P}(G)^{\mathcal{A}} \times 2,$$

that is, as a coalgebra for the functor $F'X = \mathcal{P}(X)^{\mathcal{A}} \times 2$. However, our definition of bisimulation is not the same as F'-bisimulations in the coalgebraic sense. The latter requires (in place of (iii) and (iv) above)

- For all $\langle s, t\rangle \in R$, we have $s \in \mathsf{roots}(\mathrm{G})$ iff $t \in \mathsf{roots}(\mathrm{H})$.

It is well-known that bisimulation relations are closed under arbitrary union. We use the symbol $\underline{\leftrightarrow}$ to denote the union of all bisimulations, i.e., the greatest bisimulation with respect to set inclusion. A bisimulation R is said to be *minimal* if no proper subset of R is again a bisimulation. It is said to be *functional* if it coincides with the graph of some function f. We write $\Phi(f)$ for the graph of f. For a given set map $f{:}G{\to}H$, we have that $\Phi(f)$ is a functional bisimulation iff f is an F-homomorphism (in the coalgebraic sense) satisfying

- f preserves roots, i.e., if $r \in \mathsf{roots}(\mathrm{G})$, then $f(r) \in \mathsf{roots}(\mathrm{H})$;
- $f\ \restriction_{\mathsf{roots}(\mathrm{G})} : \mathsf{roots}(\mathrm{G}) \twoheadrightarrow \mathsf{roots}(\mathrm{H})$ is surjective, i.e., for each $r' \in \mathsf{roots}(\mathrm{H})$, there is an $r \in \mathsf{roots}(\mathrm{G})$ such that $f(r) = r'$.

We also call f a functional bisimulation whenever $\Phi(f)$ is a functional bisimulation.

2.1 Paths and Reachability

The letters p, q, etc., are used to denote paths in a process graph. We write $s \overset{p}{\rightsquigarrow} t$ for "the path p starts at s and ends at t." Note the distinction between paths and traces: a path p has *trace* σ if σ is the sequence of action labels from edges in p (in the appropriate order).

Any relation on nodes of process graphs gives rise to a relation on paths in a natural way:

Definition 2.1 Let R be any relation between process graphs G and H. Let $p = s_0a_1s_1 \ldots s_{n-1}a_ns_n$ and $q = t_0a_1t_1 \ldots t_{n-1}a_nt_n$ be two paths in G and H, respectively. Then p and q are said to be *R-related* if, for all $0 \le i \le n$, $\langle s_i, t_i\rangle \in R$.

Notice that R-related paths necessarily have the same length and trace. Using this induced relation, we observe that bisimulations can be defined in terms of paths (instead of single steps).

Lemma 2.2 *Let R be a relation between process graphs G and H such that each root of G is related to some root of H and vice versa. Then R is a bisimulation if and only if, for all $\langle s, t\rangle \in R$ and $s \overset{p}{\rightsquigarrow} s'$, there is path $t \overset{q}{\rightsquigarrow} t'$ in H such that p and q are R-related and vice versa.*

Proof. The "if" part is trivial by taking a single step as a path with length 1. The converse can be proven easily by induction on the length of p.

□

The following definition of access paths is adapted from [1]. In the literature, they are also referred to as *runs* or *executions*.

Definition 2.3 Let s be a node in G. A path p in G is called an *access path* of s if $r \overset{p}{\rightsquigarrow} s$, where r is a root of G. The set of access paths of G is denoted $\mathsf{AccPath}(G)$. A node s is said to be *reachable* if it has an access path. Let $\mathsf{reach}(G)$ denote the set of reachable nodes in G (with transitions inherited from G).

Here we state a few basic facts about functional bisimulations and minimal bisimulations.

Lemma 2.4 *Let $f : G \twoheadrightarrow H$ be a functional bisimulation.*

- *If $H = \mathsf{reach}(H)$, then f is surjective.*
- *If $G = \mathsf{reach}(G)$, then $\Phi(f)$ is a minimal bisimulation.*

Lemma 2.5 *Let R be a minimal bisimulation between G and H. Then we have $\langle s, t\rangle \in R$ if and only if there exist R-related access paths p and q of s and t, respectively.*

Proof. Define R' to be the set of pairs $\langle s, t\rangle \in R$ satisfying the condition in the statement of this lemma. By minimality of R, it suffices to show that R' is also a bisimulation. We omit the details.

□

Corollary 2.6 *If R is a minimal bisimulation and $\langle s, t\rangle$ is in R, then s is reachable in G and t is reachable in H.*

2.2 Transition Structures on Bisimulations

The following is a well-known characterization of bisimulation.

Theorem 2.7 *Let G and H be process graphs and $R \subseteq G \times H$. Then R is a bisimulation iff there is a transition structure on R (i.e., a function $e_R : R \twoheadrightarrow \mathcal{P}(R)^{\mathcal{A}}$ and a subset $\mathsf{roots}(\mathrm{R}) \subseteq R$) such that the projections $\pi_1 : R \twoheadrightarrow G$ and $\pi_2 : R \twoheadrightarrow H$ are functional bisimulations.*

Notice, if R itself is functional, then π_1 is a bijection. In that case, R is isomorphic to its domain.

Lemma 2.8 *Let R be any bisimulation between G and H and fix a transition structure on R making the projections functional bisimulations, as in Theorem 2.7. Then $\mathsf{reach}(R)$ is again a bisimulation between G and H.*

Proof. The projections $\pi_1 : \mathsf{reach}(R) \twoheadrightarrow G$ and $\pi_2 : \mathsf{reach}(R) \twoheadrightarrow H$ are functional bisimulations. Apply Theorem 2.7. □

Theorem 2.7 implies that, for any bisimulation, there is a transition structure making the projections homomorphisms. We are particularly interested in the largest such structure, explicitly defined here.

Definition 2.9 Let R be any bisimulation between G and H. Define the *maximal labeled transition system on* R as follows:

(i) $\langle r_1, r_2 \rangle$ is a root of R if and only if r_1 is a root of G and r_2 is a root of H;

(ii) $\langle s, t \rangle \xrightarrow{a} \langle s', t' \rangle$ if and only if $s \xrightarrow{a} s'$ in G and $t \xrightarrow{a} t'$ in H.

The next theorem states that the maximal LTS on R satisfies the condition of Theorem 2.7. (This result is also noted in [1].) It is routine to verify this is in fact the largest such structure. In the special case that R is functional, the maximal LTS is the *only* LTS making both projections functional bisimulations.

Theorem 2.10 *Let R be any bisimulation between G and H. The projections $\pi_1 : R \twoheadrightarrow G$ and $\pi_2 : R \twoheadrightarrow H$ are functional bisimulations with respect to the maximal LTS on R.*

Hereafter, we shall always impose the maximal LTS on a bisimulation R, unless stated otherwise. We should emphasize the distinction between this definition and the *synchronous product* of two transition systems (cf. [5]): the synchronous product is uniquely determined between each pair of graphs G and H (not necessarily bisimilar), whereas each bisimulation R between G and H (necessarily bisimilar) yields its own maximal LTS.

Lemma 2.11 *Let R be any bisimulation between G and H. Then $\langle s, t \rangle \in \mathsf{reach}(R)$ if and only if there exist R-related access paths p and q of s and t, respectively.*

Combined with Lemma 2.5, we can see that R is a minimal bisimulation implies $R = \mathsf{reach}(R)$. If R is not minimal, then it's possible (but not necessary) to have unreachable pairs. For example, one can safely augment a bisimulation R with all pairs $\langle s, t \rangle$ such that s and t are termination nodes (i.e., those without out-going edges). Call the resulting bisimulation R'. Depending on the histories of s and t, the pair $\langle s, t \rangle$ may or may not be reachable in R'.

3 Concise Graphs

Conciseness is a condition on the branching structure of a process graph. As we shall see in Theorem 3.8, conciseness limits branching flexibility just enough to guarantee existence of the least bisimulation under the construction in Lemma 2.8. This least bisimulation is crucial in subsequent categorical developments.

Definition 3.1 A process graph G is said to be *concise* if G contains no distinct but bisimilar roots and for all s, t_1, t_2 in $\mathsf{reach}(G)$,

$$(s \xrightarrow{a} t_1 \text{ and } s \xrightarrow{a} t_2 \text{ and } t_1 \leftrightarrows t_2) \Rightarrow t_1 = t_2.$$

Diagram (1) illustrates the forbidden situation. The intuition here is that “redundant” branches are not allowed in a concise graph (hence the name).

$$\begin{array}{ccc} & s & \\ {}^{a}\swarrow & & \searrow^{a} \\ t_1 & \underline{\leftrightarrow} & t_2 \end{array} \tag{1}$$

In practice, this situation may arise in the following way: a program performs a boolean test “`if` ***bexp*** `then` A `else` B,” where A and B exhibit the same behaviors (i.e., they are bisimilar states). An algorithm to suppress such useless boolean tests is presented in [6].

Conciseness is much weaker than determinism, because we are still allowed to take two different a-steps from the same node, as long as the target nodes are not bisimilar. Note also that for a graph to be concise, it is not necessary to identify all bisimilar nodes. In other words, with conciseness we can have distinct but bisimilar nodes, provided those nodes are not reachable via $\underline{\leftrightarrow}$-related paths.

We give an alternative characterization of conciseness. It takes the form of a proof principle on $\mathsf{AccPath}(G)$: the relation “$\underline{\leftrightarrow}$-related” coincides with identity. This proof principle is valid for $\mathsf{AccPath}(G)$ iff G is concise. This is analogous to coinduction for coalgebras: the relations $\underline{\leftrightarrow}$ and identity on a coalgebra C coincide iff C is a subcoalgebra of the final coalgebra.

Lemma 3.2 *A process graph G is concise if and only if, for all access paths p and q in G, we have p is $\underline{\leftrightarrow}$-related to q iff $p = q$.*

Notice that a functional bisimulation can be viewed as an operation that identifies certain bisimilar nodes in the domain, hence it can never create a non-concise situation. I.e., functional bisimulations preserve conciseness.

Lemma 3.3 *Let $f : G \twoheadrightarrow H$ be a functional bisimulation. If G is concise, then so is H.*

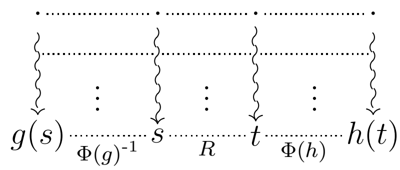
Proof. Let r_1 and r_2 be bisimilar roots of H. Since $\Phi(f)$ is a bisimulation, we can choose r_1' and r_2' in $\mathsf{roots}(\mathrm{G})$ such that $f(r_1') = r_1$ and $f(r_2') = r_2$. Hence r_1' and r_2' are bisimilar. By conciseness of G, $r_1' = r_2'$, therefore $r_1 = r_2$.

Now suppose we have (in $\mathsf{reach}(H)$) $s \xrightarrow{a} t_1$ and $s \xrightarrow{a} t_2$ with t_1 and t_2 bisimilar. By Lemma 2.4, all of these nodes are in the range of f. Choose $s' \in G$ such that $f(s') = s$. Since $\Phi(f)$ is a bisimulation, we may choose t_1' with $s' \xrightarrow{a} t_1'$ and $f(t_1') = t_1$. Similarly for t_2. Since t_1 and t_2 are bisimilar, so are t_1' and t_2'. By conciseness of G, we conclude that $t_1' = t_2'$; hence $t_1 = t_2$. $\square$

The following lemma will be used to construct coproducts in Sect. 4.

Lemma 3.4 *Let G, H and S be process graphs with S concise. Suppose $g : G \twoheadrightarrow S$ and $h : H \twoheadrightarrow S$ are functional bisimulations and $R \subseteq G \times H$ is a minimal bisimulation. Then for all $\langle s, t\rangle \in R$, $g(s) = h(t)$.*

Proof. Let $\langle s, t\rangle \in R$ be given. By Lemma 2.5, we have R-related access paths p and q of s and t, respectively. Since $\Phi(g)$ is functional, there must be access path l of $g(s)$ such that l and p are $\Phi(g)^{-1}$-related. Similarly there is access path l' of $h(t)$ such that q and l' are $\Phi(h)$-related. Therefore l and l' are $\Phi(h) \circ R \circ \Phi(g)^{-1}$-related; here $\circ$ denotes relational composition. By conciseness of S, this implies $g(s) = h(t)$.



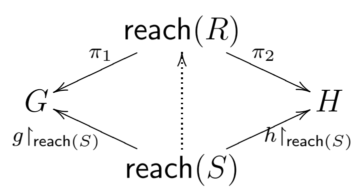
□

There is an alternative proof of Lemma 3.4 using Theorem 3.8. We can view $\Phi(h) \circ R \circ \Phi(g)^{-1}$ as a relation on $\mathsf{reach}(S)$. It's easily shown to be minimal, hence must coincide with $\Delta_{\mathsf{reach}(S)}$.

3.1 Existence of the Least Bisimulation

For the first step, we observe the following universal property.

Theorem 3.5 *Let R be any bisimulation between G and H. Suppose either G or H is concise. Let S be any process graph with functional bisimulations $g{:}S{\twoheadrightarrow}G$ and $h{:}S{\twoheadrightarrow}H$. Then the restrictions of g and h to $\mathsf{reach}(S)$ factors (necessarily uniquely) through $\mathsf{reach}(R)$, as shown in Diag. (2).*



(2)

Proof. Without loss of generality, we assume that G is concise.

Let $s \in \mathsf{reach}(S)$ be given. Choose an access path p of s. By Lemma 2.2 and the fact that $\Phi(h)$ is a bisimulation, there must be an access path q in H such that p and q are $\Phi(h)$-related. Similarly, there is an access path l in G such that l and p are $\Phi(g)^{-1}$-related.

On the other hand, since R is a bisimulation, there must be an access path l' such that l' and q are R-related. Hence l and l' are $(R^{-1} \circ \Phi(h) \circ \Phi(g)^{-1})$-related. By Lemma 3.2, we have that $g(s) = \mathsf{end}(l) = \mathsf{end}(l')$, so $\langle g(s), h(s)\rangle = \langle \mathsf{end}(l'), \mathsf{end}(q)\rangle \in R$. Apply Lemma 2.11 to conclude $\langle g(s), h(s)\rangle$ is reachable in R. □

As we will see in Theorem 4.8, the map $\mathsf{reach}(S){\twoheadrightarrow}R$ is in fact a functional bisimulation.

Consider the exemplary non-concise graph G, as shown in Diag. (1). Let R be the identity relation Δ_G. There are two functional bisimulations from G to itself: the identity function $g = \mathsf{id_G}$ and the *swap* function h mapping s to s, t_1 to t_2, and t_2 to t_1. The conclusion of Theorem 3.5 fails for this example, because $\langle g(t_1), h(t_1)\rangle = \langle t_1, t_2\rangle \notin R$. This suggests that, for a pair of non-concise graphs, the binary product cannot be constructed in a straightforward way using an arbitrary minimal bisimulation.

The following is an immediate consequence of Theorem 3.5.

Corollary 3.6 *If G is concise and H is bisimilar to G, then* $\mathsf{reach}(R) = \mathsf{reach}(S)$ *for any bisimulations R and S between G and H.*

Corollary 3.6 implies that $\mathsf{reach}(R)$ (for any R) is included in the intersection of all bisimulations between G and H. Combined with Lemma 2.8, this intersection must be exactly $\mathsf{reach}(R)$. In summary, we have the following theorem.

Theorem 3.7 *If G is concise, then for any H bisimilar to G and any bisimulation R between G and H,* $\mathsf{reach}(R)$ *is the least bisimulation between G and H (with respect to set inclusion).*

This gives a rather strong hint on the extent to which conciseness restricts the branching structure of process graphs; i.e., there is essentially only one way to construct a bisimulation between a concise graph and any other bisimilar graph.

Consider the special case in which H coincides with G and R is the identity relation Δ_G. Then Theorem 3.7 says every bisimulation from G to itself must include $\mathsf{reach}(\Delta_G)$. (Notice $\mathsf{reach}(\Delta_G)$ is just Δ_G restricted to the reachable nodes of G.) This fails easily when G is not concise: in (1), the identity relation is not included in the swap relation $\{\langle s, s\rangle, \langle t_1, t_2\rangle, \langle t_2, t_1\rangle\}$.

Moreover, in the same graph G, the relation $R := \Delta_G \cup \{\langle t_1, t_2\rangle\}$ is a bisimulation and $\mathsf{reach}(R) = R$; but clearly R is not a minimal bisimulation. Hence Theorem 3.7 fails in a different way.

We now strengthen Theorem 3.7 by showing its converse. This gives yet another characterization of conciseness.

Theorem 3.8 *Let G be a process graph. The following are equivalent:*

- *G is concise;*
- *for any process graph H bisimilar to G and any bisimulation R between them,* $\mathsf{reach}(R)$ *is the least bisimulation between G and H.*

Proof. The forward implication is Theorem 3.7. Conversely, suppose G is not concise. Let $\underline{\leftrightarrow}_G$ be the greatest bisimulation from G to G. We claim that we can find $\langle t_1, t_2\rangle$ in $\mathsf{reach}(\underline{\leftrightarrow}_G)$ such that t_1 is distinct from t_2:

(i) If G has distinct but bisimilar roots, then we set $\langle t_1, t_2\rangle$ to be a pair of such roots. By definition, this is a root of $\underline{\leftrightarrow}_G$; therefore it must be reachable.

(ii) Otherwise, we choose witnesses s, t_1 and t_2 (in $\mathsf{reach}(G)$) such that $s \xrightarrow{a} t_1$, $s \xrightarrow{a} t_2$, and t_1 and t_2 are distinct but bisimilar. Let p be any access path of s. Then pat_1 and pat_2 are $\underline{\leftrightarrow}_G$-related access paths. By Lemma 2.11, $\langle t_1, t_2 \rangle$ is reachable in $\underline{\leftrightarrow}_G$.

Now consider the bisimulation Δ_G. Certainly $\langle t_1, t_2 \rangle$ is not in Δ_G. This implies $\mathsf{reach}(\underline{\leftrightarrow}_G)$ is not a subset of Δ_G and hence not the least bisimulation from G to G. □

Before concluding this section, we raise the following question: when is the least bisimulation R in Theorem 3.8 a functional bisimulation?

Definition 3.9 We say that a process graph H is a *forest* if each node in H has exactly one access path. (In particular, this implies that $\mathsf{reach}(H) = H$.)

Intuitively, if t in H has more than one access paths, then a bisimulation R may be required to relate t to multiple nodes in G, because each access path in H must have an R-related counterpart in G. Therefore, in order to prove existence of functional bisimulations, we require $\mathsf{reach}(H)$ to be a forest. The corollary which follows is immediate.

Theorem 3.10 *Assume G is concise and H is bisimilar to G, Moreover, assume that every node in H has at most one access path (i.e., $\mathsf{reach}(H)$ is a forest). Then the least bisimulation R (from Theorem 3.7) between G and H is a partial function from H to G, total on $\mathsf{reach}(H)$.*

Proof. Let $t \in \mathsf{reach}(H)$ be given. Then t must be related (by R) to some node in G. Let s_1 and s_2 be two such nodes. It suffices to show that $s_1 = s_2$.

By assumption on H, there is a unique access path p of t. Since R is minimal, we can apply Lemma 2.5, to get access paths q_1 and q_2 of s_1 and s_2, respectively. Furthermore, q_1 and q_2 are $(R^{-1} \circ R)$-related. Now applying conciseness of G and Lemma 3.2, we can conclude that $s_1 = s_2$, hence R is the graph of a partial function from $\mathsf{reach}(H)$ to G.

□

Corollary 3.11 *Let $G \underline{\leftrightarrow} G'$, where G' is concise, and $H \underline{\leftrightarrow} H'$, where H' is a forest. Then $G \underline{\leftrightarrow} H$ iff there is a functional bisimulation $H' \twoheadrightarrow G'$. In particular, this applies when H' is the unfolding of H, and G' is the canonical concise graph for G (constructed in Section 5).*

4 Categories of Process Graphs

In this section, we define four closely related categories of process graphs. The most fundamental of these is the category P of process graphs and functional bisimulations. The category CP is the full subcategory of P consisting of concise process graphs.

Call a process graph G *restricted* if $\mathsf{reach}(G) = G$. We denote the full subcategory of restricted process graphs by RP and the full subcategory of concise, restricted process graphs by CRP.

We shall explore the relationship between these four categories in more detail in Section 5. Presently, we show that each of these categories inherits coequalizers from the category **Set**. Coequalizers will be used to construct coproducts (Sect. 4.2) and the reflection $\mathsf{P} \to \mathsf{CP}$ (Section 5). We postpone the treatment of binary products until the end of this section, because it requires that we work in (subcategories of) RP.

4.1 Coequalizers

In this section, we consider a common categorical construction: coequalizers, the standard generalization of quotients by equivalence relations in the category **Set**. Since we will explicitly use the construction of coequalizers in **Set** to show that we have coequalizers in our categories, we review that construction here.

Let $X \underset{g}{\overset{f}{\rightrightarrows}} Y$ be given (in **Set**.). Define a relation $\sim$ on Y by

$$\sim \; = \Phi(g) \circ (\Phi(f))^{-1} \;,$$

i.e., $y \sim y'$ iff there is an $x \in X$ such that $f(x) = y$ and $g(x) = y'$. Let $\equiv$ be the least equivalence relation containing $\sim$. Then, the map $Y \to Y/\equiv$ is a coequalizer of f and g.

In order to show that the same construction yields a coequalizer in our settings, we rely on the following.

Lemma 4.1 *Let G be a process graph and R a bisimulation on G. Let $\equiv$ be the least equivalence relation containing R. Then $\equiv$ is also a bisimulation.*

Proof. For this, we explicitly construct $\equiv$ in the usual way. Namely,

$$\begin{aligned} \equiv_0 &= R \cup R^{-1} \cup \Delta, \\ \equiv_{n+1} &= \equiv_n \circ \equiv_n, \\ \equiv &= \bigcup \equiv_n \;. \end{aligned}$$

Since Δ and R^{-1} are also bisimulations, and bisimulations are closed under unions, $\equiv_0$ is a bisimulation. Bisimulations are closed under composition, and so each $\equiv_n$ is a bisimulation. Again, we appeal to closure under unions to conclude that $\equiv$ is a bisimulation. □

Theorem 4.2 *The category* P *has all coequalizers and these coequalizers are preserved by the forgetful functor taking a graph to its set of nodes. In other words, the forgetful functor* $\mathsf{P} \to$ **Set** *create coequalizers. Similarly for the categories* CP, RP *and* CRP.

Proof. We prove the result for the category P of process graphs. For the remaining categories, it suffices to show the operation taking G to $G/\equiv$ preserves restrictedness and conciseness. We omit those easy proofs.

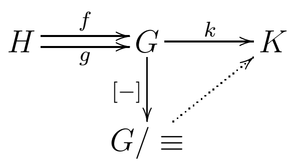
Let $H \overset{f}{\underset{g}{\rightrightarrows}} G$ be functional bisimulations. Define $\sim$ and $\equiv$ as relations on G as above. Since bisimulations are closed under composition, $\sim$ is a bisimulation, and hence so is $\equiv$.

We impose an LTS structure on $G/\equiv$ by first defining

$$\mathsf{roots}(\mathrm{G}/\equiv) = \{[r] \mid r \in \mathsf{roots}(\mathrm{G})\} \ ,$$

where $[r]$ denotes the coset (i.e., equivalence class) of r. We define a transition $[s] \xrightarrow{a} [t]$ just in case there is a transition $s \xrightarrow{a} t'$ in G for some $t' \equiv t$. This is well-defined, since $\equiv$ is a bisimulation. Furthermore, it is easy to see that the quotient map $[-]\colon G \to G/\equiv$ is a bisimulation under this definition.

Suppose that $k\colon G \to K$ is a functional bisimulation making the top row of the diagram below commute. We must show that there is a unique functional bisimulation, shown as a dashed arrow, making the triangle commute.

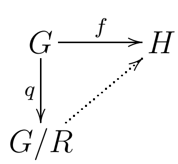


Clearly, it is necessary and sufficient to show that the set function $G/\equiv \to K$ defined by $[s] \mapsto k(s)$ is a functional bisimulation. The graph of this function is the relational composition $\Phi(k) \circ \Phi([-])^{-1}$, and hence is a bisimulation.

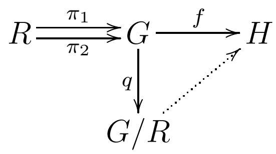
□

Given a bisimulation R on G and a functional bisimulation $f\colon G \to H$, we say that f *respects* R if, whenever sRt, we have $f(s) = f(t)$.

Corollary 4.3 *Let R be a bisimulation on a process graph G. There is a process graph G/R and a functional bisimulation $q\colon G \to G/R$ such that every functional bisimulation $f\colon G \to H$ respecting R factors through q uniquely, as shown.*



Proof. We regard R as a process graph, with its maximal LTS (Sect. 2). Note that f respects R iff f coequalizes the projections $R \rightrightarrows G$. On the other hand, these projections are functional bisimulations, so we may apply Theorem 4.2 to obtain their coequalizer q. Let G/R be the codomain of q. Therefore, there is a unique functional bisimulation from G/R to H making the diagram commute.



□

Remark 4.4 Explicitly, G/R is constructed as follows. Take $\equiv$ as the equivalence relation generated by R. Then the nodes of G/R are the cosets of $\equiv$. A coset is a root of R if it contains some root of G. For each s, t in G, there is a transition $[s] \xrightarrow{a} [t]$ iff there is a transition $s \xrightarrow{a} t'$ for some $t' \equiv t$.

4.2 Binary Coproduct

We now turn to binary coproducts of bisimilar process graphs. We approach this by first taking the coproduct $G + H$ in **Set** (i.e., disjoint union) for bisimilar G and H. There is an evident process graph structure on $G+H$, but the resulting graph is *not* the coproduct of G and H in P (or its subcategories RP, CP, CRP). Instead, we define a bisimulation $\overline{R}$ on $G + H$ and show that $(G + H)/\overline{R}$ (as given in Corollary 4.3) satisfies the universal property of coproducts. For this, we assume that at least one of G and H is concise.

Let's first make precise the evident transition structure on $G + H$:

- $\mathsf{roots}(\mathrm{G} + \mathrm{H}) = \mathsf{roots}(\mathrm{G}) \cup \mathsf{roots}(\mathrm{H})$;
- $\mathsf{in}_{\mathrm{G}}(s) \xrightarrow{a} \mathsf{in}_{\mathrm{G}}(t)$ if and only if $s \xrightarrow{a} t$ in G;
- similarly for $\mathsf{in}_{\mathrm{H}}(s) \xrightarrow{a} \mathsf{in}_{\mathrm{H}}(t)$.

Note that the inclusions $\mathsf{in}_{\mathrm{G}} : G \to G + H$ and $\mathsf{in}_{\mathrm{H}} : G \to G + H$ are *not* bisimulations in general, and so $G + H$ cannot be the coproduct of G and H in P.

Consider the least bisimulation R between G and H as given by Theorem 3.8. It can be viewed as a relation R^{G+H} on $G + H$ in the obvious way. Namely,

$$R^{G+H} = \Phi(\mathsf{in}_{\mathrm{H}}) \circ R \circ (\Phi(\mathsf{in}_{\mathrm{G}}))^{-1} \ ,$$

where in_{G} and in_{H} are the canonical inclusions of G and H, respectively, in $G + H$. Let $\overline{R}$ denote the relation

$$\overline{R} = R^{G+H} \cup (R^{G+H})^{-1} \cup \Delta_{G+H}.$$

In order to construct $(G + H)/\overline{R}$, we need to verify that $\overline{R}$ is a bisimulation on $G + H$.

Lemma 4.5 *Let R be any bisimulation between G and H. Then $\overline{R}$, defined as above, is a bisimulation on $G + H$.*

Proof. Clearly, $\overline{R}$ relates every root of $G + H$ to itself. Suppose $s \xrightarrow{a} t$ in $G + H$ and $\langle s, s' \rangle \in \overline{R}$. We consider three cases.

$\langle s, s' \rangle \in R^{G+H}$ Then $s \in G$ and $s' \in H$ and $\langle s, s' \rangle \in R$. Since R is a bisimulation, there is a $t' \in H$ such that $s' \xrightarrow{a} t'$ and $\langle t, t' \rangle \in R$. Hence, $\langle t, t' \rangle \in R^{G+H} \subseteq \overline{R}$ and $s' \xrightarrow{a} t'$ in $G + H$.

$\langle s, s' \rangle \in (R^{G+H})^{-1}$ Similar.

$\langle s, s' \rangle \in \Delta_{G+H}$ Then $s = s'$ and $\langle t, t \rangle \in \overline{R}$.

□

Let $\kappa_1 : G \to (G + H)/\overline{R}$ be the composite

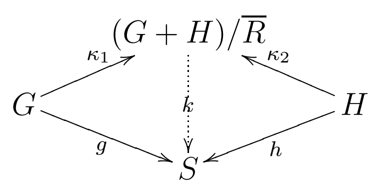
$$G \longrightarrow G + H \longrightarrow (G + H)/\overline{R},$$

and $\kappa_2 : H \to (G + H)/\overline{R}$ the analogous map for H. The lemma below (stated without proof) establishes that these maps are in fact morphisms in P. We then prove that $\langle (G + H)/\overline{R}, \kappa_1, \kappa_2 \rangle$ form a coproduct of G and H in P.

Lemma 4.6 *The maps κ_1 and κ_2 are functional bisimulations.*

Theorem 4.7 *Let G and H in* P *(*RP, CP, CRP*, resp.) be bisimilar. Assume either graph is concise. Then the coproduct of G and H exists in* P *(*RP, CP, CRP*, resp.).*

Proof. We first prove the result for P. Let $\langle (G + H)/\overline{R}, \kappa_1, \kappa_2 \rangle$ be given as above. Let S be a graph with functional bisimulations $g : G \to S$ and $h : H \to S$. We show that there is a (necessarily unique) map $k : (G + H)/\overline{R} \to S$ making the following diagram commute.



Let $m : G + H \to S$ be the unique **Set** map such that $m \circ \mathsf{in}_G = g$ and $m \circ \mathsf{in}_H = h$. It is easy to check that m is a functional bisimulation. We will show that m respects the bisimulation $\overline{R}$. By Corollary 4.3, this gives the desired unique map $k : (G + H)/\overline{R} \to S$.

We prove m respects R^{G+H}. The proof is similar for $\langle s, t \rangle \in (R^{G+H})^{-1}$, and trivial for $\langle s, t \rangle \in \Delta_{G+H}$. By definition, $\langle s, t \rangle \in R^{G+H}$ implies $s \in G$, $t \in H$ and $\langle s, t \rangle \in R$. By Lemma 3.4, $g(s) = h(t)$, i.e., $m(s) = m(t)$.

We have completed the proof that this construction yields a coproduct in P. Suppose, now, that G and H are in RP (CP, CRP, resp.). Then, $(G + H)/\overline{R}$ is also in RP (CP, CRP, resp.). By fullness and faithfulness of the inclusion, $(G + H)/\overline{R}$ is a coproduct in RP (CP, CRP, resp.). □

This coproduct construction may fail without conciseness. Consider again the graph G in Diag. (1) and let R be the swap relation. Then, in $(G + G)/\overline{R}$, the two leaf nodes are identified. This is not the coproduct, because there is no functional bisimulation from $(G + G)/\overline{R}$ to G.

4.3 Binary Product

The naive way to construct a product of two process graphs is to start with the Cartesian product $G \times H$ and try to define a transition structure so that the projections are functional bisimulations. Very quickly, one realizes this plan is not feasible. If the projections π_1 and π_2 were functional bisimulations, then $s \leftrightarroweq \langle s, t\rangle \leftrightarroweq t$ for all s, t in G. Clearly, that is not the case in general. We arrive at the conclusion that binary product in RP should not contain pairs of non-bisimilar nodes. Naturally, bisimulation relations become candidates for products.

The situation with products is different from those with coequalizers and coproducts, namely that our construction works only in RP and its subcategory CRP.

Theorem 4.8 *Let G and H be restricted process graphs. Assume that G is concise and H is bisimilar to G. Then the binary product of G and H exists in* RP.

Proof. By Theorem 3.8, we have the least bisimulation R between G and H. By Theorem 2.10, the projections π_1 and π_2 are functional bisimulations. We will show that $\langle R, \pi_1, \pi_2\rangle$ forms a product of G and H.

Let S be any restricted process graph and let $g : S \twoheadrightarrow G$ and $h : S \twoheadrightarrow H$ be functional bisimulations. By Theorem 3.5, we can define $m(s) = \langle g(s), h(s)\rangle$ for every reachable s in S. Since S is restricted, m is a total function. We claim that $\Phi(m)$ is a bisimulation.

Indeed, let $\langle r, r'\rangle$ be a root in R. Then $r \in \mathsf{roots}(\mathrm{G})$ and $r' \in \mathsf{roots}(\mathrm{H})$. Since $\Phi(h)$ is a bisimulation, we can choose $r'' \in \mathsf{roots}(\mathrm{S})$ such that $h(r'') = r'$. Notice that $g(r'') \in \mathsf{roots}(\mathrm{G})$. Moreover, $g(r'') \leftrightarroweq r'' \leftrightarroweq r' \leftrightarroweq r$. By conciseness of G, we conclude that $g(r'') = r$; hence there is a root r'' of S such that

$$m(r'') = \langle g(r''), h(r'')\rangle = \langle r, r'\rangle.$$

The proof that $\Phi(m)$ satisfies the transition conditions (i) and (ii) from Section 2 proceeds similarly using conciseness of G and definition of the maximal LTS on R.

Uniqueness of m follows from the fact that π_1 and π_2 are jointly monic in **Set**.

□

In the proof of Theorem 4.8, we used the assumption that $S = \mathsf{reach}(S)$ to establish totality of m. Without this assumption, $g(s)$ may be unreachable in G, in which case $\langle g(s), h(s)\rangle$ must not be in R (due to minimality of R).

In other words, m may not be well-defined for unreachable nodes in S. This is the reason for considering only restricted graphs.

It is easy to check that the least bisimulation R between two concise graphs is a concise graph; hence the construction in Theorem 4.8 also works in CRP.

5 A Categorical Comparison

In this section, we discuss the relationship between the various categories: P (process graphs), CP (concise process graphs), RP (restricted process graphs) and CRP (concise, restricted process graphs). Our main aim is to show that CP is a reflective subcategory of P. This gives a canonical means of constructing, for each process graph G, a bisimilar concise graph H. This construction should be viewed as an analogue to closure operators on partial orders, with one caveat: The graph H is constructed by taking a quotient of G, not by enlarging G. Following this task, we comment on the categories of restricted process graphs.

We will define a functor conc taking a process graph G to $G/{\sim^G}$, where $\sim^G$ is the least bisimulation such that $G/{\sim^G}$ is concise. We begin by describing the bisimulation $\sim^G$.

Let G be a process graph. We define a relation $\sim^G$ on G as follows.

$$\begin{aligned}
\sim^G_0 &= \underline{\leftrightarrow} \cap (\mathsf{roots}(\mathrm{G}) \times \mathsf{roots}(\mathrm{G})) \\
\sim^G_{n+1} &= \{\langle s,\, t\rangle \mid \exists v, u, a\ u \xrightarrow{a} s,\ v \xrightarrow{a} t,\ u \sim^G_n v \text{ and } s \,\underline{\leftrightarrow}\, t\} \\
\sim^G &= \bigcup \sim^G_n
\end{aligned}$$

Pictorially, the second clause says that, in a situation

$$\begin{array}{ccc} u & \sim^G_n & v \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle a} \\ s & \underline{\leftrightarrow} & t \end{array} \tag{3}$$

we require $s \sim^G_{n+1} t$. Note, in particular, that $s \sim^G t$ implies both s and t are reachable.

We omit the proof of the lemma below. It involves induction on the construction of $\sim^G$.

Lemma 5.1 *For each process graph G, the relation $\sim^G$ is a bisimulation.*

The process graph $G/{\sim^G}$ is constructed according to Corollary 4.3. For each $G \in \mathsf{P}$, we define $\mathsf{conc}(-) = G \mapsto G/{\sim^G}$ and η_G to be the surjection $G \twoheadrightarrow G/{\sim^G}$. As we will see in Lemma 5.3, $\mathsf{conc}(G)$ is concise. Note that G is essentially obtained, then, by adding a history variable to $\mathsf{conc}(G)$, its "concisification". Put another way: $\mathsf{conc}(G)$ is constructed by "forgetting" a (fictional) history variable in G.

First, we show that conc is functorial and η is natural.

Lemma 5.2 *Let $f : G \twoheadrightarrow H$ be a functional bisimulation. For each s, t in G, if $s \sim^G t$, then $f(s) \sim^H f(t)$. Consequently, the operator conc is functorial and η is a natural transformation $\mathsf{Id}_{\mathsf{P}} \Rightarrow \mathsf{conc}$.*

Proof. The first statement can be proved by an easy induction on the definition of $\sim^G$.

For the second, we must define, for each functional bisimulation $f : G \twoheadrightarrow H$, a functional bisimulation $\mathsf{conc}(f) : G/\!\sim^G \twoheadrightarrow H/\sim^H$. By Corollary 4.3, it suffices to show that the composite

$$G \xrightarrow{f} H \xrightarrow{\eta_H} H/\sim^H$$

respects $\sim^G$. This is equivalent to the first statement of the present lemma: for all s and t, $s \sim^G t$ implies $f(s) \sim^H f(t)$.

Naturality of η follows trivially from our definition of $\mathsf{conc}(f)$.

□

Lemma 5.3 *The graph $G/\!\sim^G$ is concise.*

Proof. If $[r]$, $[r']$ are roots of $G/\!\sim^G$ with $[r] \mathrel{\underline{\leftrightarrow}} [r']$, then $r \mathrel{\underline{\leftrightarrow}} [r] \mathrel{\underline{\leftrightarrow}} [r'] \mathrel{\underline{\leftrightarrow}} r'$ and hence $r \sim^G r'$, i.e., $[r] = [r']$. We must show, then, that for every $[s]$ reachable in $G/\!\sim^G$, if $[s] \xrightarrow{a} [t]$ and $[s] \xrightarrow{a} [t']$ and $[t] \mathrel{\underline{\leftrightarrow}} [t']$, then $[t] = [t']$. It suffices to show, for any s reachable in G with transitions $s \xrightarrow{a} t$ and $s \xrightarrow{a} t'$, we have $t \mathrel{\underline{\leftrightarrow}} t'$ implies $t \sim^G t'$.

Let $r \overset{p}{\leadsto} s$ be given, where $r \in \mathsf{roots}(\mathrm{G})$. A simple proof by induction shows that $s \sim^G_n s$, where n is the length of p. Thus, $t \sim^G_{n+1} t'$ and hence $t \sim^G t'$.

□

Theorem 5.4 CP *is a reflective subcategory of* P*. Explicitly, the functor* $\mathsf{conc} : \mathsf{P} \twoheadrightarrow \mathsf{CP}$ *is left adjoint to the inclusion* $\mathsf{CP} \hookrightarrow \mathsf{P}$*.*

Proof. Let H be concise. By [7, §IV.3], it suffices to show that every functional bisimulation $f : G \twoheadrightarrow H$ factors through η_G, i.e., that η_G is universal from G to conc.

By Corollary 4.3, it is enough to prove such f respects the bisimulation $\sim^G$: for all s and t, $s \sim^G t$ implies $f(s) = f(t)$. We proceed by induction on the definition of $\sim^G$.

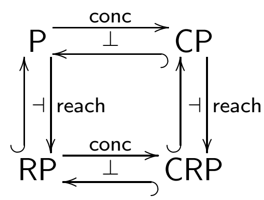
If $s \sim^G_0 t$, then s and t are bisimilar roots, and the result follows by conciseness of H.

Suppose that $s \sim^G_{n+1} t$. Then we have u, v, as in (3). By inductive hypothesis, $f(u) = f(v)$, and so we have

$$\begin{array}{ccc} & f(u) & \\ {\scriptstyle a}\swarrow & & \searrow{\scriptstyle a} \\ f(s) & \underline{\leftrightarrow} & f(t) \end{array}$$

in H. Since $u \sim^G v$, we have u and v are reachable; hence $f(u)$ reachable. Now we apply conciseness of H to conclude $f(s) = f(t)$. □

One can check that the category RP is a co-reflective subcategory of P via the functor reach : P→RP. Because coequalizers in RP are inherited from P, the reflection conc : P→CP restricts to a reflection RP→CRP. Similarly, the coreflection reach restricts to a coreflection CP→CRP. Thus, we have the following commutative square of adjoint functors.



6 The General Case: Without Conciseness

In this section we prove that minimal bisimulations exist provided both graphs are image finite. Unlike the situation with concise graphs, these minimal bisimulations are not necessarily unique (and hence not least).

We use a variation of Zorn's Lemma, listed as **M′2** in [8].

Lemma 6.1 *Let $\sqsubset$ be a transitive relation on the set S such that every $T \subseteq S$ well-ordered by $\sqsubset$ has an upper bound and let $s \in S$. Then there is a maximal $s' \in S$ such that $s \sqsubset s'$ or $s = s'$.*

The relevant order here is reverse inclusion. Therefore, we will show that any ordinally-indexed, decreasing chain $\{R_\beta \mid \beta \lneqq \alpha\}$ of bisimulations has a lower bound and conclude that there exists a minimal bisimulation. In fact, the situation here is stronger: Any ordinally-indexed, decreasing chain of bisimulations between image finite graphs has a greatest lower bound, given by their set-intersection. Of course, this claim is trivial for chains indexed by successor ordinals. We therefore concentrate on limit ordinals. We begin with some preliminary facts about such ordinals.

Definition 6.2 Let α be a limit ordinal. Let $\xi = (x_\beta \mid \beta \lneqq \alpha)$ be a sequence with range X. Then $x \in X$ is said to occur *cofinally in the sequence* ξ if for every $\beta \lneqq \alpha$, there is γ such that $\beta \lneqq \gamma \lneqq \alpha$ and $x_\gamma = x$.

It is easy to see that, if we have an α-sequence (α a non-zero limit ordinal) with a finite range, then at least one element in the range must occur cofinally in α in that sequence. This is used to prove the following lemma.

Lemma 6.3 *Let α be a limit ordinal. Let $S_0 \supseteq S_1 \supseteq \ldots \supseteq S_\beta \supseteq \ldots$ be a decreasing chain of sets indexed by the ordinals below α. Let $\xi = (x_\beta \mid \beta \lneqq \alpha)$ be a sequence such that $x_\beta \in S_\beta$ for every $\beta \lneqq \alpha$. If ξ has a finite range then there is $\overline{\beta}$ such that $x_{\overline{\beta}} \in \bigcap\{S_\beta \mid \beta \lneqq \alpha\}$.*

Proof. Write $x_1, \ldots, x_n$ for the elements of the range of ξ. There must be $1 \leq i \leq n$ such that x_i occurs cofinally in ξ. Choose such i. Then for each $\beta \lneqq \alpha$, we can find $\beta \leq \gamma \lneqq \alpha$ such that $x_\gamma = x_i$ (thus $x_i \in S_\gamma$). Since $\{S_\beta \mid \beta \lneqq \alpha\}$ is a decreasing chain, this implies $x_i \in S_\beta$. Therefore, $x_i \in S_\beta$ for all $\beta \lneqq \alpha$, i.e., $x_i \in \bigcap\{S_\beta \mid \beta \lneqq \alpha\}$. Now let $\overline{\beta}$ be any β such that $x_\beta = x_i$. □

In fact, this lemma holds for any sequence ξ whose range has a cardinality strictly below the cofinality of α. For our purposes, a finite range is appropriate. This allows us to prove that the intersection of a decreasing chain of bisimulations is still a bisimulation.

Lemma 6.4 *Let G and H be image finite process graphs. Let α be a limit ordinal. Let $\{R_\beta \mid \beta \lneqq \alpha\}$ be a decreasing chain of bisimulations between G and H. Then $R_\alpha := \bigcap\{R_\beta \mid \beta \lneqq \alpha\}$ is a bisimulation.*

Proof. Let $r \in \mathsf{roots}(\mathrm{G})$ be given. For each $\beta \lneqq \alpha$, there exists r'_β such that $\langle r,\, r'_\beta\rangle$ is in R_β. Notice that H has finitely many roots (because H is image finite). Hence $\{r'_\beta \mid \beta \lneqq \alpha\}$ is also finite. Be Lemma 6.3, we can choose $\overline{\beta}$ such that $\langle r,\, r'_{\overline{\beta}}\rangle \in \bigcap\{R_\beta \mid \beta \lneqq \alpha\} = R_\alpha$. Therefore, there exists r' (namely $r'_{\overline{\beta}}$) such that $\langle r,\, r'\rangle \in R_\alpha$.

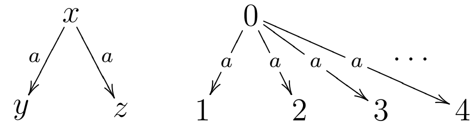
Now suppose we have $u \xrightarrow{a} v$ in G and $\langle u,\, u'\rangle$ in R_α. For all $\beta \lneqq \alpha$, there exists v'_β such that $u' \xrightarrow{a} v'_\beta$ and $\langle v,\, v'_\beta\rangle$ is in R_β. Since H is image finite, we have $\{v' \mid u' \xrightarrow{a} v'\}$ is finite; hence $\{v'_\beta \mid \beta \lneqq \alpha\}$ is also finite. By an argument similar to that in the root case, there exists v' such that $u' \xrightarrow{a} v'$ and $\langle v,\, v'\rangle \in R_\alpha$.

The direction from H to G follows by symmetry. □

In this lemma, it is necessary that both graphs are image finite. The following illustrates a counterexample in which one of the graphs is not image finite. For each $n \in N$, define R_n to be

$$\{\langle 0,\, 0\rangle\} \cup \{\langle y,\, i\rangle \mid i \in \mathbb{N}\} \cup \{\langle z,\, i\rangle \mid i \geq n\}\ .$$

This defines a decreasing ω-chain of bisimulations, but its intersection is the set $\{\langle 0,\, 0\rangle\} \cup \{\langle y,\, i\rangle \mid i \in \mathbb{N}\}$, which is not a bisimulation.



Lemmas 6.4 and 6.1 yield the following.

Theorem 6.5 *Let G and H be image finite process graphs. Suppose R is a bisimulation between them. Then there is $R' \subseteq R$ such that R' is a minimal bisimulation between G and H.*

Again it is necessary for both graphs to be image finite. We have already seen a counterexample in Fig. 1.

It is not hard to find examples in which minimal bisimulations are not unique. Figure 2 and Diag. (1) give two such examples. Figure 3 provides another.

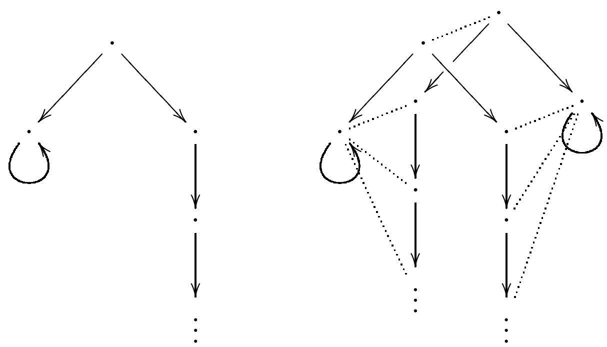
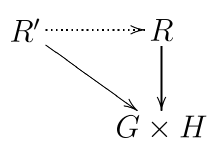


Fig. 3. Non-isomorphic minimal bisimulations: the identity relation Δ and the relation R indicated by the dotted lines

Notice that in Diag.(1), although the identity relation and the swap relation are different sets of ordered pairs, the maximal LTS's on them are isomorphic. However, in Figs. 2 and 3, the minimal bisimulations differ in an irreparable way (i.e., the maximal LTS's on them are not isomorphic).

Unfortunately, since these minimal bisimulations are not least, we cannot extend directly the results in Sect. 3 about binary products and coproducts. However, our conjecture is that there is always a suitable minimal bisimulation that will give rise to product. A bisimulation R on G and H is "suitable" if, for every bisimulation R' on G and H, there is a (necessarily unique) functional bisimulation $R' \rightarrowtail R$ making the diagram (in **Set**) below commute.



Here, we are viewing R and R' as process graphs with maximal LTS's, as in Sect. 2. In other terms, R is suitable iff it is the "greatest" bisimulation under $\leq_{\mathsf{FB}}$ (but, note, this is quite different than the greatest bisimulation under $\subseteq$).

In every example we have considered so far, such a suitable minimal bisimulation does exist, but we have failed to produce a proof (or preferably an algorithm to search for such a bisimulation).

7 Checking Conciseness

Finally, we consider a practical issue, namely how to decide whether a given graph is concise. For that end, one can modify the definition of conciseness in the following way:

Definition 7.1 Let $I(s)$ denote the set of initial actions of node s in G. A graph G is said to be *obviously concise* if

(i) for distinct roots r_1 and r_2 of G, $I(r_1) \neq I(r_2)$;

(ii) given s, t_1 and t_2,

$$(s \xrightarrow{a} t_1 \text{ and } s \xrightarrow{a} t_2 \text{ and } t_1 \neq t_2) \Rightarrow I(t_1) \neq I(t_2).$$

With the original definition, deciding conciseness has the same complexity as deciding bisimilarity. This modified definition eliminates the need to check $t_1 \underline{\leftrightarrow} t_2$; instead, the checking algorithm needs only look up and compare the two records $I(t_1)$ and $I(t_2)$. In other words, the modified definition is a local property of the individual nodes, whereas the original definition is much more global.

Assuming the action alphabet is finite, there is an algorithm to traverse a finite graph and perform the local check described above. This algorithm will be linear in the size of the graph.

It is clear that obviously concise graphs are concise. In practice, the specification of a concurrent system often generates a relatively small state graph; hence it will be feasible to check whether the specification is (obviously) concise. This raises hope that we can apply our results about least bisimulations to prove properties between a specification and its implementation, even though the latter may not be concise.

8 Conclusions and Future Work

To begin, it is clear that our work here is preliminary. The final judgment on whether concise process graphs are useful or interesting requires more investigation. We are hopeful that constructions involving concise graphs will lead to proof principles for history relations, but we have no confidence that this is so. At present, we offer an introductory investigation into what seems a natural category of process graphs.

Aside from the broad aims of investigating concise graphs, our work here leaves open a number of specific questions.

In a sense, we are not completely satisfied with Theorem 3.8. It says that conciseness characterizes the existence of least bisimulations under the reachable part construction. However, it is unclear whether conciseness is also a characterization for general existence, i.e., without reference to any particular construction. It would be nice to find a necessary and sufficient condition on G so that the least bisimulation between G and H exists for any bisimilar H.

Also, as mentioned at the end of Section 6, it is not known to us whether there is always a greatest (with respect to $\leq_{\mathsf{FB}}$) bisimulation. If that answer is positive, we can extend the binary product and coproduct constructions to image finite bisimilar graphs.

Another natural extension of this work is to incorporate τ steps and to study some form of weak functional bisimulations. The definition of conciseness needs to be reformulated to take into account nodes on a τ-path (i.e., internal states). For example, we may consider functional branching bisimulation and require, in addition to conciseness, that a process graph contains no inert τ-steps. In order to reuse our proofs, we must also find an appropriate notion of path correspondence analogous to Lemma 2.2. A good candidate is that of index relations, introduced by Griffioen and Vaandrager in [10].

Finally, we would like to compare our categories of process graphs to the well-developed models of parallel computation in [11]. On the one hand, one may ask whether conciseness leads to useful subcategories in their setting. Does conciseness yield interesting constructions if we take our morphisms to be (generalizations of) functional simulations, as in ibid? On the other hand, does the further consideration of weak bisimulations in [12] shed light on remaining work in our setting?

9 Acknowledgments

We thank Jan Willem Klop, whose ideas initiated this exercise, for many useful discussions. We also thank Frits Vaandrager for reading a draft of this paper and providing valuable suggestions. The definition of obviously concise graphs and subsequent observations in Sect. 7 are also due to him.

Finally we thank the anonymous referees of this paper and those members of the concurrency mailing list (available at `concurrency@cwi.nl`) who responded to our questions regarding this paper.

References

[1] Ariola, Z.M., Klop, J.W.: Equational term graph rewriting. Fundamenta Informaticae **26** (1996) 207–240

[2] Lynch, N., Vaandrager, F.: Forward and backward simulations part I: Untimed systems. Information and Computation **121** (1995) 214–233

[3] Taylor, P.: Practical Foundations of Mathematics. Cambridge University Press (1999)

[4] Glabbeek, R.v.: The linear time – branching time spectrum I; the semantics of concrete, sequential processes. In Bergstra, J., Ponse, A., Smolka, S., eds.: Handbook of Process Algebra. Elsevier (2001) 3–99 Available at `http://boole.stanford.edu/pub/spectrum1.ps.gz`.

[5] Fernandez, J.C., Jard, C., Jéron, T., Mounier, L.: "On the fly" verification of finite transition systems. Formal Methods in System Design (1992)

[6] Caspi, P., Fernandez, J.C., Girault, A.: An algorithm for reducing binary branchings. In: FST-TCS Bangalore. Lecture Notes in Computer Science, Springer Verlag (1995)

[7] Mac Lane, S.: Categories for the Working Mathematician. Graduate Texts in Mathematics. Springer-Verlag (1971)

[8] Rubin, H., Rubin, J.: Equivalents of the Axiom of Choice. North-Holland Publishing Company (1963)

[9] Glabbeek, R.v., Weijland, W.: Branching time and abstraction in bisimulation semantics. Journal of the ACM **43** (1996) 555–600

[10] Griffioen, W., Vaandrager, F.: A theory of normed simulations. Technical Report CSI-R0013, Computing Science Institute, University of Nijmegen (2000)

[11] Joyal, A., Nielsen, M., Winskel, G.: Bisimulation from open maps. Technical Report RS-94-7, BRICS (1994)

[12] Fiore, M.P., Cattani, G.L., Winskel, G.: Weak bisimulation and open maps. In: Logic in Computer Science. (1999) 67–76