Modular Termination of Graph Transformation

Detlef Plump

University of York, United Kingdom

Abstract. We establish a machine-checkable condition which ensures that the union of two terminating hypergraph transformation systems is terminating. The condition is based on so-called sequential critical pairs which represent consecutive rule applications that are not independent. In contrast to a corresponding modularity result for term rewriting, no restrictions on the form of rules are required. Our result applies to both systems with injective rules and systems with rules that merge nodes or edges.

1 Introduction

In the area of graph transformation, the problem of proving that a system will terminate (admit only a finite number of rule applications) on arbitrary host graphs has received surprisingly little attention. This is in contrast to the central role of termination research in the area of term rewriting [1,2].

Work on proving termination of general graph transformation systems by various methods includes [16,3,5,4]. There are also some papers on termination in the specialised settings of term graph rewriting [15,17,14] and cycle rewriting [23,21].

In this paper, we are interested in the problem of finding conditions that guarantee that the combination of terminating (hyper-)graph transformation systems yields again a terminating system. The corresponding problem for term rewriting systems received considerable attention after Toyama showed that even the combination of systems with disjoint function symbols need not preserve termination [22]. Interestingly, the latter phenomenon vanishes into thin air for acyclic term graph rewriting [15,14] because rewrite steps create shared subgraphs instead of copying subterms.

We prove in this paper that the union of two general hypergraph transformation systems preserves termination if there are no critical overlaps between the right-hand sides of one system and the left-hand sides of the other system. This idea is inspired by a result of Dershowitz [7] which shows that the corresponding property for term rewriting systems holds if one of the systems is left-linear and the other system is right-linear. In the case of graph transformation, it turns out that no restrictions on the form of rules are needed.

The rest of this paper is organized as follows. In Section 2, we review the basics of hypergraph transformation. In particular, we recall the concept of sequential independence which is crucial for our main result. Sequential critical pairs are introduced in Section 3, in analogy to standard critical pairs for graph

transformation. Our main result is proved in Section 4 where we also give examples to demonstrate the application of the modularity criterion. We conclude in Section 5.

2 Hypergraph Transformation

In this section, we recall some definitions and results of the double-pushout approach to graph transformation which can be found, for example, in [11]. For generality, we use here the setting of hypergraphs.

2.1 Hypergraphs

Our hypergraphs are directed and labelled. We use a type system where the label of a hyperedge can restrict the number of incident nodes and their labels. A *signature* $\Sigma = \langle \Sigma_{\mathrm{V}}, \Sigma_{\mathrm{E}}, \mathrm{Type} \rangle$ consists of a set Σ_{V} of *node labels*, a set Σ_{E} of *hyperedge labels* and a function Type assigning to each $l \in \Sigma_{\mathrm{E}}$ a set of strings $\mathrm{Type}(l) \subseteq \Sigma_{\mathrm{V}}^*$. This kind of typing allows to "overload" hyperedge labels by specifying different admissible attachment sequences. Typing regimes covered by this approach include the case of a singleton set $\mathrm{Type}(l)$ for each label l (as in [6]) and the case of "untyped" hypergraphs given by $\mathrm{Type}(l) = \Sigma_{\mathrm{V}}^*$ for each l (as in [10]). Unless stated otherwise, we denote by Σ an arbitrary but fixed signature over which all hypergraphs are labelled.

A *hypergraph* over Σ is a system $G = \langle \mathrm{V}_G, \mathrm{E}_G, \mathrm{mark}_G, \mathrm{lab}_G, \mathrm{att}_G \rangle$ consisting of two finite sets V_G and E_G of *nodes* (or *vertices*) and *hyperedges*, two labelling functions $\mathrm{mark}_G \colon \mathrm{V}_G \to \Sigma_{\mathrm{V}}$ and $\mathrm{lab}_G \colon \mathrm{E}_G \to \Sigma_{\mathrm{E}}$, and an attachment function $\mathrm{att}_G \colon \mathrm{E}_G \to \mathrm{V}_G^*$ such that $\mathrm{mark}_G^*(\mathrm{att}_G(e)) \in \mathrm{Type}(\mathrm{lab}_G(e))$ for each hyperedge e. (The extension $f^* \colon A^* \to B^*$ of a function $f \colon A \to B$ maps the empty string to itself and $a_1 \dots a_n$ to $f(a_1) \dots f(a_n)$.)

In pictures, nodes and hyperedges are drawn as circles and boxes, respectively, with labels inside. Lines represent the attachment of hyperedges to nodes, where numbers specify the left-to-right order in the attachment string. For example, Figure 1 shows a hypergraph with four nodes (all labelled with •) and three hyperedges (labelled with B and S).

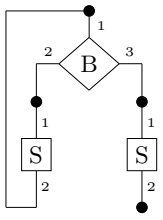


Fig. 1. A hypergraph

A hypergraph G is a *graph* if each hyperedge e is an ordinary edge, that is, if $\mathrm{att}_G(e)$ has length two. Ordinary edges may be drawn as arrows with labels written next to them.

Given hypergraphs G and H, a *hypergraph morphism* (or *morphism* for short) $g\colon G \to H$ consists of two functions $g_{\mathrm{V}}\colon \mathrm{V}_G \to \mathrm{V}_H$ and $g_{\mathrm{E}}\colon \mathrm{E}_G \to \mathrm{E}_H$ that preserve labels and attachment to nodes, that is, $\mathrm{mark}_H \circ g_{\mathrm{V}} = \mathrm{mark}_G$, $\mathrm{lab}_H \circ g_{\mathrm{E}} = \mathrm{lab}_G$ and $\mathrm{att}_H \circ g_{\mathrm{E}} = g_{\mathrm{V}}^* \circ \mathrm{att}_G$. A morphism $\mathit{incl}\colon G \to H$ is an *inclusion* if $\mathit{incl}_{\mathrm{V}}(v) = v$ and $\mathit{incl}_{\mathrm{E}}(e) = e$ for all $v \in \mathrm{V}_G$ and $e \in \mathrm{E}_G$. In this case G is a *subhypergraph* of H which is denoted by $G \subseteq H$. Morphism g is *injective* (*surjective*) if g_V and g_E are injective (surjective). If g is both injective and surjective, then it is an *isomorphism*. In this case G and H are *isomorphic*, which is denoted by $G \cong H$.

The *composition* of two morphisms $g\colon G \to H$ and $h\colon H \to M$ is the morphism $h \circ g\colon G \to M$ consisting of the composed functions $h_{\mathrm{V}} \circ g_{\mathrm{V}}$ and $h_{\mathrm{E}} \circ g_{\mathrm{E}}$. The composition is also written as $G \to H \to M$ if g and h are clear from the context.

2.2 Rules and derivations

A *rule* $r\colon \langle L \leftarrow K \to R \rangle$ consists of two hypergraph morphisms with a common domain, where $K \to L$ is an inclusion. The hypergraphs L and R are the *left-* and *right-hand side* of r, and K is the *interface*. The rule is *injective* if the morphism $K \to R$ is injective.

Let G and H be hypergraphs, $r\colon \langle L \leftarrow K \to R \rangle$ a rule and $g\colon L \to G$ an injective morphism. Then G *directly derives* H by r and g, denoted by $G \Rightarrow_{r,g} H$, if there exist two pushouts as in Figure 2. Given a set of rules $\mathcal{R}$, we write $G \Rightarrow_{\mathcal{R}}$

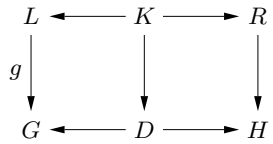


Fig. 2. A double-pushout

H to express that there exist $r \in \mathcal{R}$ and a morphism g such that $G \Rightarrow_{r,g} H$.

We refer to [20] for the definition and construction of hypergraph pushouts. Intuitively, the left pushout corresponds to the construction of D from G by removing the items in $L - K$, and the right pushout to the construction of H from D by merging items according to $K \to R$ and adding the items in R that are not in the image of K.

A double-pushout as in Figure 2 is called a *direct derivation* from G to H and may be denoted by $G \Rightarrow_{r,g} H$ or just by $G \Rightarrow_r H$ or $G \Rightarrow H$. A *derivation* from G to H is a sequence of direct derivations $G_0 \Rightarrow \ldots \Rightarrow G_n$, $n \geq 0$, such

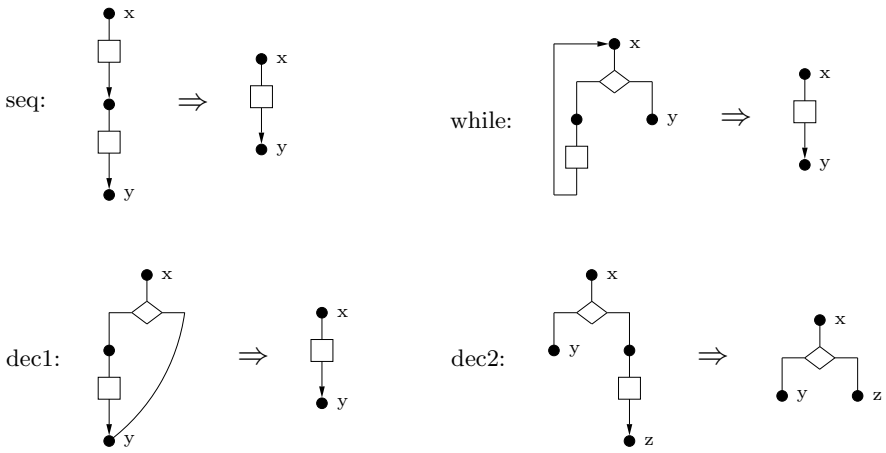


Fig. 3. Hypergraph transformation system for flow-graph reduction

that $G \cong G_0$ and $G_n \cong H$. We write $G \Rightarrow^* H$ for such a derivation or $G \Rightarrow^*_{\mathcal{R}} H$ if all rules used in the derivation are from $\mathcal{R}$.

Given a rule $r\colon \langle L \leftarrow K \rightarrow R \rangle$, an injective morphism $g\colon L \rightarrow G$ satisfies the *dangling condition* if no hyperedge in $\mathrm{E}_G - g_\mathrm{E}(\mathrm{E}_L)$ is incident to a node in $g_\mathrm{V}(\mathrm{V}_L - \mathrm{V}_K)$. It can be shown that, given r and f, a direct derivation as in Figure 2 exists if and only if g satisfies the dangling condition [11].

Definition 1 (Hypergraph transformation system). A *hypergraph transformation system* $\langle \Sigma, \mathcal{R} \rangle$ consists of a signature Σ and a finite set $\mathcal{R}$ of rules over Σ. The system is *injective* if all rules in $\mathcal{R}$ are injective. It is a *graph transformation system* if for each label l in Σ_E, all strings in $\mathrm{Type}(l)$ are of length two.

Note that since graph transformation systems are special hypergraph transformation systems, results for the latter usually apply to the former, too.

Example 1. Figure 3 shows hypergraph transformation rules for reducing control-flow graphs [20]. The associated signature contains a single node label $\bullet$ and two hyperedge labels which are graphically represented by hyperedges formed as squares and rhombs. Instead of using numbers to represent the attachment function, we use an arrow to point to the second attachment node of a square and define the order among the links of a rhomb to be "top-left-right". The rules are shown in a shorthand notation where only the left- and right-hand sides are depicted, the interface and the morphisms are implicitly given by the node names x,y,z. □

2.3 Sequential independence

Given injective rules r_1 and r_2, consecutive direct derivations $G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$ do not interfere with each other if the intersection of the right-hand side of r_1 with the left-hand side of r_2 in H consists of common interface items. In the presence of non-injective rules, however, independence needs to be defined in terms of the existence of certain morphisms. We give the general definition first and then consider a simpler condition for the case of injective rules. For $i = 1, 2$, let r_i denote a rule $\langle L_i \leftarrow K_i \rightarrow R_i \rangle$.

Definition 2 (Sequential independence). Direct derivations $G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$ as in Figure 4 are *sequentially independent* if there are morphisms $R_1 \rightarrow D_2$ and $L_2 \rightarrow D_1$ such that the following holds.

Commutativity: $R_1 \rightarrow D_2 \rightarrow H = R_1 \rightarrow H$ and $L_2 \rightarrow D_1 \rightarrow H = L_2 \rightarrow H$.
Injectivity: $R_1 \rightarrow D_2 \rightarrow M$ is injective.

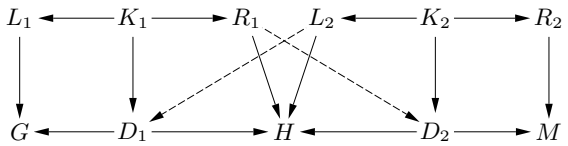


Fig. 4. Sequentially independent direct derivations

The injectivity requirement for $R_1 \rightarrow D_2 \rightarrow M$ is needed in case r_2 is non-injective. See [11] for an example that the steps $G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$ may not be interchangeable without this condition.

If r_1 and r_2 are injective, the direct derivations of Figure 4 are independent if and only if the intersection of R_1 and L_2 in H coincides with the intersection of K_1 and K_2. Moreover, the injectivity condition of Definition 2 is automatically satisfied.

Lemma 1 (Independence for injective rules). *Let rules r_1 and r_2 in Figure 4 be injective. Then the following are equivalent:*

(1) *$G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$ are sequentially independent.*
(2) *There are morphisms $R_1 \rightarrow D_2$ and $L_2 \rightarrow D_1$ such that $R_1 \rightarrow D_2 \rightarrow H = R_1 \rightarrow H$ and $L_2 \rightarrow D_1 \rightarrow H = L_2 \rightarrow H$.*
(3) *$h(R_1) \cap g(L_2) = h(b(K_1)) \cap g(K_2)$ where $h\colon R_1 \rightarrow H$, $g\colon L_2 \rightarrow H$ and $b\colon K_1 \rightarrow R_1$. (Note that $K_2 \rightarrow L_2$ is an inclusion.)*

Proof. The equivalence of (1) and (2) is an easy consequence of the fact that with injective rules, all morphisms in Figure 4 are injective. The equivalence of (2) and (3) is shown in [9,19]. □

The following theorem was originally proved in [8], in the setting of graph transformation with arbitrary, possibly non-injective matching morphisms. This proof was adapted in [19] to the setting of hypergraph transformation with injective matching.

Theorem 1 ([19,11]). *Given sequentially independent direct derivations* $G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$*, there exist direct derivations of the form* $G \Rightarrow_{r_2} H' \Rightarrow_{r_1} M$*.*

For instance, Figure 5 shows applications of the rules seq and dec2 from Example 1. The steps are independent because the occurrences of the right-hand side of seq and the left-hand side dec2 share only nodes y and z, which are interface nodes in both rules. Hence the steps can be interchanged, leading to the same result.

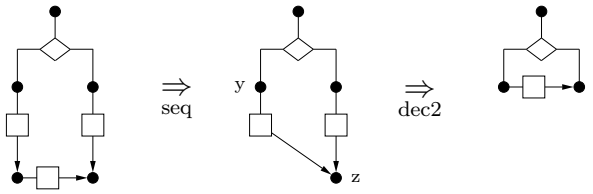


Fig. 5. Sequentially independent rule applications

3 Sequential Critical Pairs

Standard critical pairs represent conflicts between rule applications to the same graph and are used to analyse graph transformation systems for local confluence [20]. In the context of verifying termination, we adapt this concept to represent conflicts between consecutive direct derivations.

Sequential critical pairs are minimal derivations of length two whose steps are not independent. Due to the minimality, the set of critical pairs induced by two hypergraph transformation rules is finite and can be constructed. We will see that if this set is empty, then any pair of direct derivations with these rules is sequentially independent. In the proof of the main result, this will enable us to re-order the rule applications of infinite derivations.

Definition 3 (Sequential critical pair). Let $r_i \colon \langle L_i \leftarrow K_i \rightarrow R_i \rangle$ be rules, for $i = 1, 2$. A pair of direct derivations $S \Rightarrow_{r_1} T \Rightarrow_{r_2} U$ as in Figure 6 is a *sequential critical pair* if the following holds.

Minimality: $T = h_1(R_1) \cup g_2(L_2)$.
Conflict: The steps are not sequentially independent.

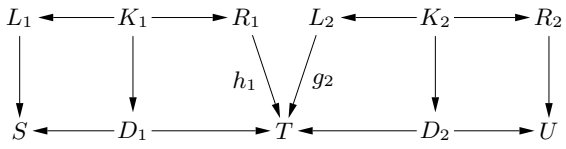


Fig. 6. A sequential critical pair

Sequential critical pairs $S \Rightarrow_{r_1,g_1} T \Rightarrow_{r_2,g_2} U$ and $S' \Rightarrow_{r_1,g_1'} T' \Rightarrow_{r_2,g_2'} U'$ are *isomorphic* if there are isomorphisms $i_1 \colon S \to S'$ and $i_2 \colon T \to T'$ such that $g_1' = i_1 \circ g_1$ and $g_2' = i_2 \circ g_2$. (This implies that U and U' are isomorphic, too.) From now on we equate isomorphic critical pairs. With the minimality condition of Definition 3 it follows that every hypergraph transformation system has only finitely many critical pairs.

Example 2. Figure 7 shows two sequential critical pairs of the rules of Figure 3.

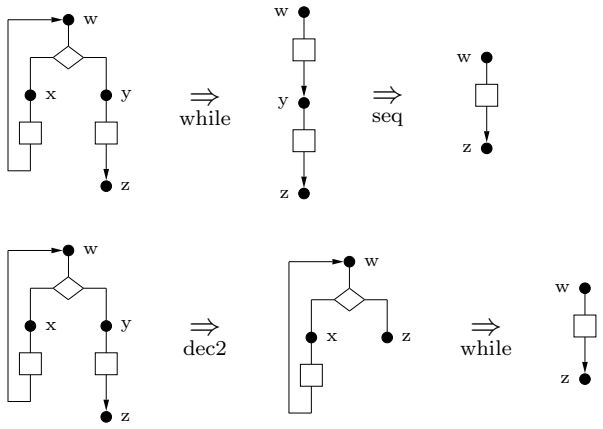


Fig. 7. Two sequential critical pairs of the system of Figure 3

Lemma 2. *For every pair of direct derivations $G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$ that are not sequentially independent, there exists a critical pair of the form $S \Rightarrow_{r_1} T \Rightarrow_{r_2} U$.*

Proof sketch. The steps $G \Rightarrow_{r_1} H \Rightarrow_{r_2} M$ can be restricted to a critical pair by removing all nodes and edges that are not used by r_1 or r_2. That is, S is the subhypergraph of G consisting of all items in the occurrence of the left-hand side of r_1 and all items that are preserved by $G \Rightarrow_{r_1} H$ and are in the occurrence of

the left-hand side of r_2. See [19] for the precise construction. It is not difficult to check that the restricted steps are minimal and in conflict. □

4 Modular Termination

A hypergraph transformation system $\langle \Sigma, \mathcal{R} \rangle$ is *terminating* if there exists no infinite derivation $G_0 \Rightarrow_{\mathcal{R}} G_1 \Rightarrow_{\mathcal{R}} G_2 \Rightarrow_{\mathcal{R}} \dots$ The problem to decide whether a system $\langle \Sigma, \mathcal{R} \rangle$ is terminating is undecidable in general, even for graph transformation systems with injective rules [18].

Theorem 2 (Modular termination). *Let $\langle \Sigma, \mathcal{R} \rangle$ and $\langle \Sigma, \mathcal{S} \rangle$ be terminating hypergraph transformation systems. If there are no sequential critical pairs of shape $S \Rightarrow_{\mathcal{R}} T \Rightarrow_{\mathcal{S}} U$, then the combined system $\langle \Sigma, \mathcal{R} \cup \mathcal{S} \rangle$ is terminating.*

Note the symmetry in the statement of Theorem 2: it is sufficient that there are no critical pairs of shape $S \Rightarrow_{\mathcal{R}} T \Rightarrow_{\mathcal{S}} U$ *or* no critical pairs of shape $S \Rightarrow_{\mathcal{S}} T \Rightarrow_{\mathcal{R}} U$.

Proof of Theorem 2. Let $\langle \Sigma, \mathcal{R} \rangle$ and $\langle \Sigma, \mathcal{S} \rangle$ be terminating hypergraph transformation systems such that there are no critical pairs of shape $S \Rightarrow_{\mathcal{R}} T \Rightarrow_{\mathcal{S}} U$. We proceed by contradiction. Suppose there exists an infinite derivation

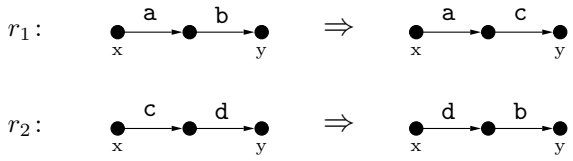
$$G_1 \underset{\mathcal{R}\cup\mathcal{S}}{\Rightarrow} G_2 \underset{\mathcal{R}\cup\mathcal{S}}{\Rightarrow} G_3 \underset{\mathcal{R}\cup\mathcal{S}}{\Rightarrow} \dots$$

Because $\Rightarrow_{\mathcal{R}}$ and $\Rightarrow_{\mathcal{S}}$ are terminating, the derivation must contain infinitely many steps of both kinds. By Lemma 2, any two steps $G_k \Rightarrow_{\mathcal{R}} G_{k+1} \Rightarrow_{\mathcal{S}} G_{k+2}$ in the sequence are sequentially independent because there are no critical pairs of shape $S \Rightarrow_{\mathcal{R}} T \Rightarrow_{\mathcal{S}} U$. By Theorem 1, the steps can be swapped such that $G_k \Rightarrow_{\mathcal{S}} G'_{k+1} \Rightarrow_{\mathcal{R}} G_{k+2}$. Thus all $\Rightarrow_{\mathcal{S}}$-steps can be pushed back to the beginning of the derivation, resulting in an infinite sequence of $\Rightarrow_{\mathcal{S}}$-steps. This contradicts the fact that $\langle \Sigma, \mathcal{S} \rangle$ is terminating. □

Figure 8 illustrates the infinite swapping process used to prove Theorem 2. Any infinite derivation with $\mathcal{R}\cup\mathcal{S}$ must contain infinitely many $\mathcal{S}$-steps and hence pushing them to the beginning *ad infinitum* will build up an infinite derivation of $\mathcal{S}$-steps.

We now present a sequence of examples which demonstrate the application of Theorem 2.

Example 3. Consider the following graph transformation system from [16]:



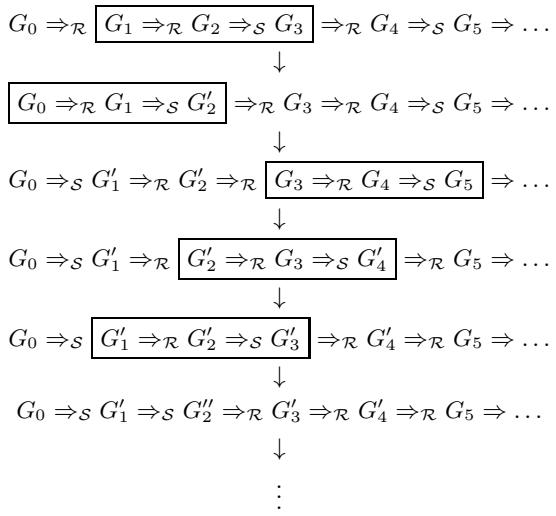
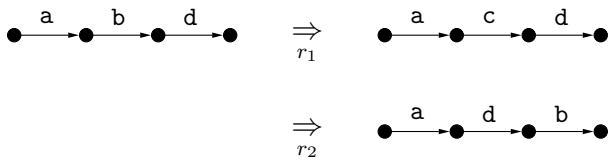


Fig. 8. Infinite swapping process to create an infinite $\mathcal{S}$-derivation

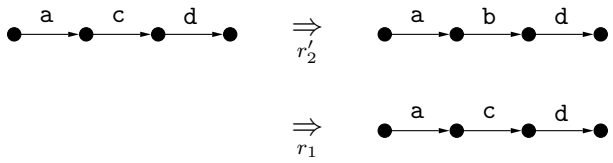
It is not obvious that this system is terminating because r_1 reduces the number of `b`-labels in a graph while r_2 increases this number. Similarly, r_2 reduces the number of `c`-labels while r_1 increases this number.

We can prove that this system is terminating by showing termination of r_1 and r_2 separately and then applying Theorem 2. Each rule individually is terminating because r_1 reduces the number of `b`'s and r_2 reduces the number of `c`'s. Moreover, there are no critical pairs of shape $S \Rightarrow_{r_2} T \Rightarrow_{r_1} U$. This is because the middle node in the right-hand side of r_2 is created by the rule and hence cannot have an incoming `a`-edge. Thus, by Theorem 2, the combined system is terminating.

Note that the system does have a critical pair of shape $S \Rightarrow_{r_1} T \Rightarrow_{r_2} U$:



Moreover, if we replace rule r_2 with r_2' by swapping labels `d` and `b` in the right-hand side, then there is also a critical pair of shape $S \Rightarrow_{r_2'} T \Rightarrow_{r_1} U$. Indeed, this change makes the system non-terminating as the new critical pair is cyclic:



□

It is worth pointing out the mechanical nature of the termination proof in Example 3. Combining a simple label counting algorithm with a generator for sequential critical pairs would automatically determine that both rules are terminating and that there are no critical pairs of shape $S \Rightarrow_{r_2} T \Rightarrow_{r_1} U$. Based on Theorem 2, this method would then report that the combined system is terminating.

Example 4. The graph transformation system in Figure 9 is shown to be terminating in [5]. The technique used is to first simplify the system by removing

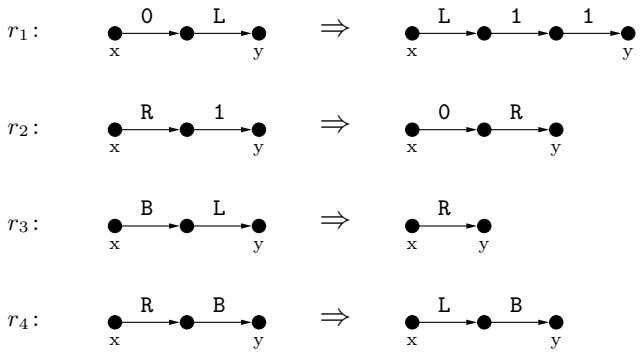


Fig. 9. A terminating graph transformation system [5]

rules r_3 and r_4, obtaining a system that is terminating if and only if the original system is terminating. The argument is as follows: rule r_3 decreases the number of B-labels while no other rule increases this number. Hence any infinite derivation contains only finitely many applications of r_3 and thus $\langle \Sigma, \{r_1, \ldots, r_4\} \rangle$ is terminating if and only if $\langle \Sigma, \{r_1, r_2, r_4\} \rangle$ is terminating. After removing r_3, one can observe that rule r_4 reduces the number of R-labels while neither r_1 nor r_2 increase this number. Hence r_4 can be removed, too.

For the simplified system $\langle \Sigma, \{r_1, r_2\} \rangle$, a so-called weighted type graph over the tropical semiring is constructed [5] which provides a decreasing measure for both rules. We can give a simpler termination argument for this system by using the approach of Example 3: r_1 is terminating as it reduces the number of 0's

and r_2 is terminating as it reduces the number of 1's. Also, it is easy to check that there are no critical pairs of shape $S \Rightarrow_{r_1} T \Rightarrow_{r_2} U$ or $S \Rightarrow_{r_2} T \Rightarrow_{r_1} U$. Hence Theorem 2 guarantees that $\langle \Sigma, \{r_1, r_2\} \rangle$ is terminating. □

Similar to Example 3, our termination proof could be found automatically by a tool that counts labels, eliminates rules that decrease label counts not increased by the other rules, and finally generates sequential critical pairs.

Example 5. The hypergraph transformation system in Figure 10 checks, when applied as long as possible, whether a host graph is 2-colourable (bipartite). If this is the case, the rules colour the graph accordingly. We assume that the initial hypergraph is a loop-free connected graph in which each node has exactly one "colour flag" attached to it (a hyperedge e with $|\mathrm{att}_G(e)| = 1$). Moreover, exactly one flag should be labelled `r` (or `b`) and all other flags should be blank. To save space, we deviate from our usual drawing convention in that all ordinary edges shown in the rules are interface edges.

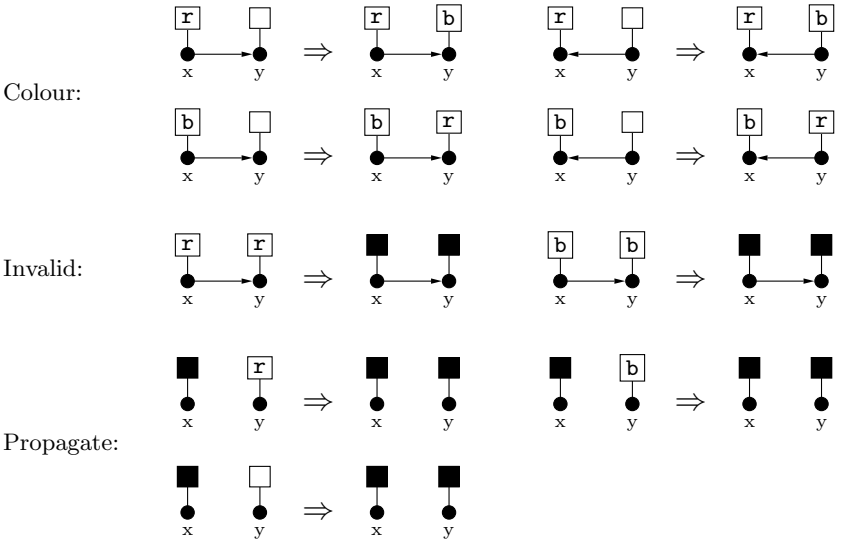


Fig. 10. Hypergraph transformation system for generating a 2-colouring

If the initial graph is 2-colourable, the system of Figure 10 will terminate with a coloured version of the graph in which each node is coloured `r` or `b`. (A graph is 2-colourable if its underlying undirected graph has no cycles of odd length.) If the graph is not 2-colourable, this will eventually be detected by the Invalid rules. The Propagate rules then make sure that all colour flags are black in the final graph.

As in the previous examples, we can prove termination of the system by label counting and Theorem 2. Subsystem Colour is terminating because all rules reduce the number of blank flags. On the other hand, Invalid ∪ Propagate is terminating as all rules decrease the number of non-black flags. It is easy to see that there are no critical pairs of shape $S \Rightarrow_{\text{Invalid}\cup\text{Propagate}} T \Rightarrow_{\text{Colour}} U$ (since all ordinary edges are interface edges) and thus the combined system is terminating. □

We remark that termination of the system of Figure 10 can alternatively be proved by removing the Colour rules and observing that the Invalid and Propagate rules reduce the number of non-black flags. This works because the Colour rules reduce the number of blank flags while the Invalid and Propagate rules do not increase this number. However, the point of Example 5 is to demonstrate that a non-trivial system can be decomposed into subsystems such that Theorem 2 is applicable, and where the components are proved terminating with different measures.

Example 6. Our final example is about *jungle evaluation*, a framework in which hypergraphs representing functional expressions are evaluated by transformation rules [12]. Figure 11 shows the non-injective evaluation rule corresponding to the term rewriting rule $\mathtt{y} + \mathtt{0} \to \mathtt{y}$, where the notation x=y means that interface nodes x and y are merged by the right-hand morphism. This rule is clearly

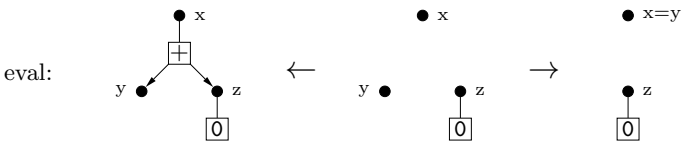


Fig. 11. Jungle evaluation rule for $\mathtt{y} + \mathtt{0} \to \mathtt{y}$

terminating as it reduces the size of any hypergraph it is applied to. Rule copy in Figure 12, on the other hand, enlarges any hypergraph it is applied to. The

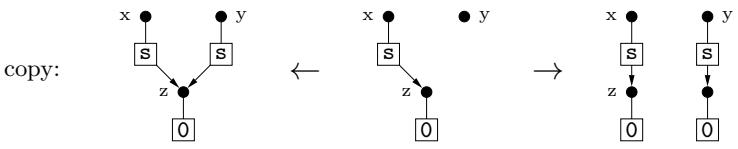


Fig. 12. Rule for copying a shared constant 0

rule copies an occurrence of the constant 0 that is shared by two s-functions.

The rule is terminating because each application reduces the measure

$$\#G = \sum_{v \in V_G} \text{indegree}(v)^2.$$

For, consider a step $G \Rightarrow_{\text{copy}} H$ and let n be the indegree of node z in G. Then $\#H = (\#G - n^2) + (n-1)^2 + 1 < \#G$ where the inequality holds because $(n-1)^2 + 1 < n^2$ for $n \geq 2$, and $n = \text{indegree}(z) \geq 2$.

It is not difficult to check that there are no sequential critical pairs of shape $S \Rightarrow_{\text{eval}} T \Rightarrow_{\text{copy}} U$, and thus the combined system {eval, copy} is terminating by Theorem 2. Note that it is not obvious how to combine graph size and the # value into a measure that decreases under rule applications of the combined rule set. This is because copy always increases graph size and eval increases the # value when applied to certain hypergraphs. To see the latter, consider a step $G \Rightarrow_{\text{eval}} H$ where $\text{indegree}(x) = 2$, $\text{indegree}(y) = 3$ and $\text{indegree}(z) = 1$. Then $\#H = \#G - (4 + 9 + 1) + (4^2 - 1) = \#G + 1$. □

5 Conclusion and Future Work

Termination is an undecidable property of graph transformation systems. We have established a criterion based on the absence of certain sequential critical pairs which guarantees that the union of two terminating systems is terminating. This allows to split systems into component systems whose termination is then verified separately, possibly using different techniques, and to conclude that the combination of the components is terminating if the critical pair-criterion is satisfied. The criterion is syntactic and can be machine-checked by generating all critical pairs between rules from different component systems. Morever, the method is a black-box approach in that the termination proofs of the component systems need not be inspected.

An obvious topic for future work is to implement a tool that given a hypergraph transformation system, generates all sequential critical pairs and calculates all possible partitions of the system into smaller components such that the condition of Theorem 2 is satisfied. For each partition, the components can then be proved to be terminating with whatever method seems suitable resp. has been implemented. The tool could be used together with a termination prover such as *Grez*, described in [5], which allows to choose different proof methods, including weighted type graphs, label counting and node counting.

Future research may also attempt to generalize Theorem 2 in various ways. It may be possible to allow critical pairs $S \Rightarrow_{\mathcal{R}} T \Rightarrow_{\mathcal{S}} U$ and formulate conditions under which arbitrary steps $G \Rightarrow_{\mathcal{R}} H \Rightarrow_{\mathcal{S}} M$ can still be swapped. A naive try is to require that there exits a graph T' such that $S \Rightarrow_{\mathcal{S}} T' \Rightarrow_{\mathcal{R}} U$, which however is insufficient as the dangling condition may prevent embedding the steps into context. The situation has some similarity with the analysis of conventional critical pairs to verify confluence: the mere joinability of all critical pairs does not guarantee local confluence of a set of rules [20].

Finally, it would be desirable to extend the approach of this paper such that rules with application conditions (of some form) can be handled. Even more challenging is an extension to attributed graph transformation on which graph programming languages such as GP 2 are based. This is because finite sets of attributed rules typically induce infinite sets of sequential critical pairs in the sense of this paper (see [13] for the corresponding problem with conventional critical pairs).

References

1. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. Marc Bezem, Jan Willem Klop, and Roel de Vrijer, editors. *Term Rewriting Systems*. Cambridge University Press, 2003.
3. H.J. Sander Bruggink. Towards a systematic method for proving termination of graph transformation systems. *Electronic Notes in Theoretical Computer Science*, 213(1):23–38, 2008.
4. H.J. Sander Bruggink, Barbara König, Dennis Nolte, and Hans Zantema. Proving termination of graph transformation systems using weighted type graphs over semirings. In *Proc. Graph Transformation (ICGT 2015)*, volume 9151 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2015.
5. H.J. Sander Bruggink, Barbara König, and Hans Zantema. Termination analysis for graph transformation systems. In *Proc. Theoretical Computer Science (TCS 2014)*, volume 8705 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2014.
6. Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 5. Elsevier, 1990.
7. Nachum Dershowitz. Termination of linear rewriting systems (preliminary version). In *Proc. Automata, Languages, and Programming (ICALP 1981)*, volume 115 of *Lecture Notes in Computer Science*, pages 448–458. Springer, 1981.
8. Hartmut Ehrig and Hans-Jörg Kreowski. Parallelism of manipulations in multidimensional information structures. In *Proc. Mathematical Foundations of Computer Science*, volume 45 of *Lecture Notes in Computer Science*, pages 284–293. Springer, 1976.
9. Hartmut Ehrig and Barry K. Rosen. Commutativity of independent transformations on complex objects. Research Report RC 6251, IBM Thomas J Watson Research Center, Yorktown Heights, 1976.
10. Annegret Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer, 1992.
11. Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.
12. Berthold Hoffmann and Detlef Plump. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 25(5):445–472, 1991.
13. Ivaylo Hristakiev and Detlef Plump. Towards critical pair analysis for the graph programming language GP 2. In *Recent Trends in Algebraic Development Techniques (WADT 2016), Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2017. To appear.

14. Madala R.K. Krishna Rao. Modular aspects of term graph rewriting. *Theoretical Computer Science*, 208(1-2):59–86, 1998.
15. Detlef Plump. Implementing term rewriting by graph reduction: Termination of combined systems. In *Proc. Conditional and Typed Rewriting Systems (CTRS'90)*, volume 516 of *Lecture Notes in Computer Science*, pages 307–317. Springer, 1991.
16. Detlef Plump. On termination of graph rewriting. In *Proc. Graph-Theoretic Concepts in Computer Science (WG'95)*, volume 1017 of *Lecture Notes in Computer Science*, pages 88–100. Springer, 1995.
17. Detlef Plump. Simplification orders for term graph rewriting. In *Proc. Mathematical Foundations of Computer Science (MFCS'97)*, volume 1295 of *Lecture Notes in Computer Science*, pages 458–467. Springer, 1997.
18. Detlef Plump. Termination of graph rewriting is undecidable. *Fundamenta Informaticae*, 33(2):201–209, 1998.
19. Detlef Plump. *Computing by Graph Rewriting.* Habilitation thesis, Universität Bremen, Fachbereich Mathematik und Informatik, 1999.
20. Detlef Plump. Confluence of graph transformation revisited. In *Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, pages 280–308. Springer, 2005.
21. David Sabel and Hans Zantema. Termination of cycle rewriting by transformation and matrix interpretation. *Logical Methods in Computer Science*, 13(1), 2017.
22. Yoshihito Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.
23. Hans Zantema, Barbara König, and H.J. Sander Bruggink. Termination of cycle rewriting. In *Proc. Rewriting and Typed Lambda Calculi (RTA-TLCA 2014)*, volume 8560 of *Lecture Notes in Computer Science*, pages 476–490. Springer, 2014.