Nat Comput
DOI 10.1007/s11047-010-9245-6

Graph multiset transformation: a new framework for massively parallel computation inspired by DNA computing

Hans-Jörg Kreowski · Sabine Kuske

Abstract In this paper, graph multiset transformation is introduced and studied as a novel type of parallel graph transformation. The basic idea is that graph transformation rules may be applied to all or at least some members of a multiset of graphs simultaneously providing a computational step with the possibility of massive parallelism in this way. As a consequence, graph problems in the class *NP* can be solved by a single computation of polynomial length for each input graph.

Keywords Graph multiset transformation · DNA computing · Transformation units · NP problems · Massive parallelism

1 Introduction

In this paper, a new type of graph transformation, called graph multiset transformation, is introduced that is inspired by the concepts of genetic algorithms and DNA computing (see, e.g., Adleman 1994; Fogel 2006; Goldberg 2002; Holland 1975; Păun et al. 1998). Adleman's seminal experiment demonstrates how combinatorial problems may be solved using DNA. Roughly speaking, a tube is filled with certain quantities of properly chosen DNA strands. Then their reactions according to the Watson–Crick complementarity produces DNA molecules, a suitable selection of which represents solutions. Similarly, a genetic algorithm transforms a "population of individuals" step by step into one of "fitter" individuals by means of "mutation," "cross-over," and "selection." If, for example, the individuals are solutions of an optimization problem that differ from the optimum, then the genetic algorithm may yield solutions that are closer to the optimum or even meet it. If one replaces tubes of molecules and populations of individuals by multisets of graphs and

H.-J. Kreowski · S. Kuske (✉)
Department of Computer Science, University of Bremen, P.O. Box 330440, 28334 Bremen, Germany
e-mail: kuske@informatik.uni-bremen.de

H.-J. Kreowski
e-mail: kreo@informatik.uni-bremen.de

Published online: 11 January 2011

chemical reactions and genetic operations by rule applications, one gets the concept of graph multiset transformation.

The basic idea is that graph transformation rules may be applied to all or at least some members of a multiset of graphs simultaneously providing a computational step with the possibility of massive parallelism in this way. As a consequence, graph problems in the class *NP* can be solved by a single computation of polynomial length for each input graph.

The paper is organized in the following way. The next section provides the preliminaries concerning graphs and rule-based graph transformation. In Sect. 3, simple graph transformation units are recalled as devices to model and compute graph algorithms and processes. Section 4 introduces a way to solve decision problems on graphs by means of terminating units. In particular, a graph-transformational variant of the class *NP* is defined. Based on simple and terminating units, graph multiset transformation is proposed as a computational framework with massive parallelism in Sects. 5 and 7. As a consequence, *NP*-problems can be solved in a polynomial number of computational steps. In Sect. 6, we simulate Adleman's experiment by means of graph multiset transformation. The Appendix recalls multisets together with some basic definitions used in this paper. Throughout the paper, the well-known *NP*-complete Hamiltonian path problem is discussed as a running example. It may be noted that the basic ideas of graph multiset transformation have been sketched in Kreowski (2002) in a draft way and that a short and less complete version of this paper appeared in Kreowski and Kuske (2008) without proofs.

2 Graphs and rule-based graph transformation

In this section, we recall the basic notions and notations of graphs and rule-based graph transformation as far as they are needed in this paper. We use directed and edge-labeled graphs with binary edges.

Let Σ be a set of labels. A *graph* over Σ is a system $G = (V, E, s, t, l)$ where V is a finite set of *nodes*, E is a finite set of *edges*, $s, t\colon E \to V$ are mappings assigning a *source* $s(e)$ and a *target* $t(e)$ to every edge in E, and $l\colon E \to \Sigma$ is a mapping assigning a label to every edge in E. An edge e with $s(e) = t(e)$ is called a *loop*. The components V, E, s, t, and l of G are also denoted by V_G, E_G, s_G, t_G, and l_G, respectively. The set of all graphs over Σ is denoted by $\mathcal{G}_\Sigma$. We reserve a specific label $*$ which is omitted in drawings of graphs. In this way, graphs where all edges are labeled with $*$ may be seen as *unlabeled graphs*. The sum of the number of nodes and the number of edges is the *size* of G, denoted by $size(G)$.

For graphs $G, H \in \mathcal{G}_\Sigma$, a *graph morphism* $g\colon G \to H$ is a pair of mappings $g_V\colon V_G \to V_H$ and $g_E\colon E_G \to E_H$ that are structure-preserving, i.e., $g_V(s_G(e)) = s_H(g_E(e)),\quad g_V(t_G(e)) = t_H(g_E(e))$, and $l_H(g_E(e)) = l_G(e)$ for all $e \in E_G$. If the mappings g_V and g_E are bijective, then g is an *isomorphism*, and G and H are called *isomorphic*. If the mappings g_V and g_E are inclusions, then G is called a *subgraph* of H, denoted by $G \subseteq H$. For a graph morphism $g\colon G \to H$, the image $g(G) \subseteq H$ of G in H is called a *match* of G in H.

Example 1 The graph G_0 in Fig. 1 has four Hamiltonian paths which are represented by the graphs H_1, H_2, H_3, and H_4.[1] A box $\square$ represents a node with an unlabeled loop. Therefore, G_0 has four nodes, four loops and five additional unlabeled edges. The other graphs are variants of G_0. We use $\odot$ to represent a *begin*-node which is a node with a loop labeled with *begin*. Analogously, $\bigcirc$ represents an *end*-node. If one starts in the *begin*-node and follows the *p*-labeled edges, one reaches the *end*-node in the graphs H_1, H_2, H_3, and

[1] A path is called Hamiltonian if it visits every node exactly once.

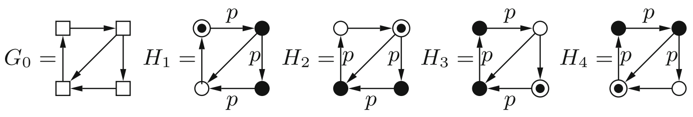


Fig. 1 G_0 with all its Hamiltonian paths

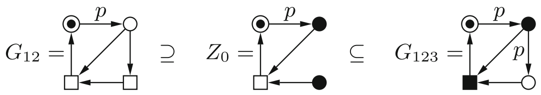


Fig. 2 Two graphs with a common subgraph

H_4. In each case, the sequence of p-edges defines a Hamiltonian path of G_0, where the intermediate nodes have no loops.

If one removes the right vertical edge and the loops at the source and the target of this edge in the graph G_{12} in Fig. 2, then one gets the subgraph Z_0. One may extend the graph Z_0 by a p-edge and an *end*-loop to get G_{123}.

There are two graph morphisms from the graph $L_{run} = \circ\!\!\longrightarrow\!\!\square$ into G_{12} which map to the subgraphs of the same form. The construction in Fig. 2 is an example of a rule application in the following sense.

A *rule* $r = (L \supseteq K \subseteq R)$ consists of three graphs $L, K, R \in \mathcal{G}_\Sigma$ such that K is a subgraph of L and R. The components L, K, and R of r are called *left-hand side*, *gluing graph*, and *right-hand side*, respectively. The application of $r = (L \supseteq K \subseteq R)$ to a graph $G = (V, E, s, t, l)$ yields a directly derived graph H and consists of the following three steps.

1. A match $g(L)$ of L in G is chosen subject to the following conditions.
   - *dangling condition*: $v \in g_V(V_L)$ with $s_G(e) = v$ or $t_G(e) = v$ for some $e \in E_G - g_E(E_L)$ implies $v \in g_V(V_K)$.
   - *identification condition*: $g_V(v) = g_V(v')$ for $v, v' \in V_L$ implies $v = v'$ or $v, v' \in V_K$ as well as $g_E(e) = g_E(e')$ for $e, e' \in E_L$ implies $e = e'$ or $e, e' \in E_K$.
2. Now the nodes of $g_V(V_L - V_K)$ and the edges of $g_E(E_L - E_K)$ are removed yielding the *intermediate graph* $Z \subseteq G$.
3. Let $d: K \to Z$ be the restriction of g to K and Z. Then H is constructed as the disjoint union of Z and $R - K$ where all edges $e \in E_Z + (E_R - E_K)$ keep their labels and their sources and targets except for $s_R(e) = v \in V_K$ or $t_R(e) = v \in V_K$ which is replaced by $d_V(v)$.

The application of a rule r to a graph G is denoted by $G \underset{r}{\Longrightarrow} H$, and called a *direct derivation*. The subscript r may be omitted if it is clear from the context.

The dangling condition requires that nodes to be removed are only incident to edges to be removed, and guarantees that the removal of $L - K$ from G yields a graph and that the restriction d of g to K and Z is a graph morphism. One possibility to construct the disjoint union of Z and $R - K$ is to rename $R - K$ to $h(R - K)$ such that $Z \cap h(R - K) = \emptyset$ and, therefore, their union is a disjoint one. In this case, we have $Z \subseteq H$, and the renaming can be extended to a graph morphism $h: R \to H$ by defining h on K by d. The identification condition concerns the match only and makes sure that G can be constructed from $d: K \to Z$ and $R - K$ as H from d and $L - K$. These constructions are known as so-called pushouts so that, altogether, a direct derivation is given by a double pushout (cf. Fig. 3).

As the pushouts and hence the rule applications are only unique up to isomorphism, it makes sense to assume from now on that $\mathcal{G}_\Sigma$ contains the corresponding equivalence

Fig. 3 Diagram of a double pushout

$$\begin{tikzcd}
L \arrow[r, phantom, "\supseteq"] \arrow[d, "g"] & K \arrow[r, phantom, "\subseteq"] \arrow[d, "d"] & R \arrow[d, "h"] \\
G \arrow[r, phantom, "\supseteq"] & Z \arrow[r, phantom, "\subseteq"] & H
\end{tikzcd}$$

classes of graphs and, in this way, abstract graphs rather than concrete ones. However, each concrete graph is a meaningful representative of its class so that we can go on to deal with graphs in a concrete way.

The sequential composition of direct derivations $d = G_0 \underset{r_1}{\Longrightarrow} G_1 \underset{r_2}{\Longrightarrow} \cdots \underset{r_n}{\Longrightarrow} G_n$ $(n \in \mathbb{N})$ is called a *derivation* from G_0 to G_n. As usual, the derivation from G_0 to G_n can also be denoted by $G_0 \overset{n}{\underset{P}{\Longrightarrow}} G_n$ where $\{r_1, \ldots, r_n\} \subseteq P$, or just by $G_0 \overset{*}{\underset{P}{\Longrightarrow}} G_n$. The subscript P may be omitted if it is clear from the context. The string $r_1, \ldots, r_n$ is the *application sequence* of the derivation d.

Example 2 Consider the following two rules

$$start \quad = \quad \square \quad \supseteq \quad \bullet \quad \subseteq \quad ◎$$

$$run \quad = \circ \longrightarrow \square \supseteq \bullet \qquad \bullet \subseteq \bullet \xrightarrow{p} \circ$$

The rule *start* describes the removal of an unlabeled loop and the addition of a *begin*-loop and an *end*-loop at the same node, which is depicted by ◎. The rule *run* replaces an unlabeled edge by a *p*-edge removing the two loops of the left-hand side and adding an *end*-loop at the target node of the right-hand side.

Figure 4 shows all derivations that start in the graph G_0 in Fig. 1 and apply the rule *start* only once in the beginning (while the rule *run* is applied repeatedly afterwards). At first,

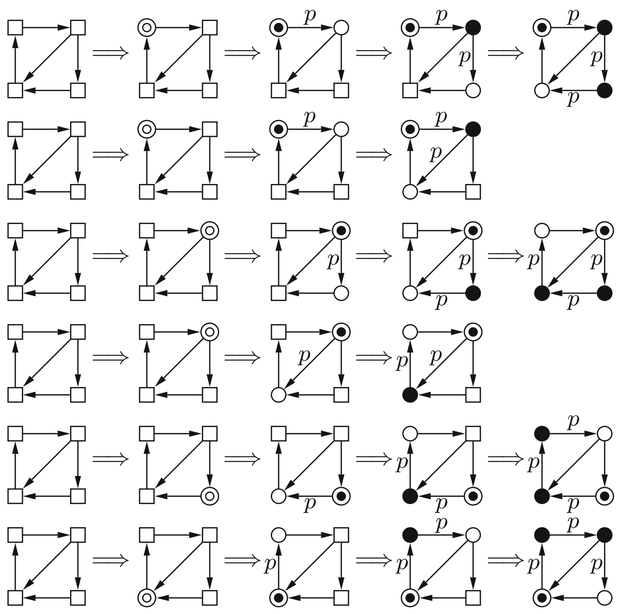


Fig. 4 The derivations starting in G_0 with one application of *start*

the rule *start* can be applied to G_0 in four ways deriving the four graphs in the second column from the left of Fig. 4. The graphs in the right-most column of Fig. 4 are H_1, H_2, H_3, and H_4 representing the Hamiltonian paths of G_0. They are characterized by the property that they do not contain any unlabeled loop.

It is not difficult to prove that the Hamiltonian paths of every unlabeled graph (with a single loop at each node) can be generated in the same way: Apply the rule *start* once and then the rule *run* repeatedly. A path in a derived graph is Hamiltonian if and only if the graph has no unlabeled loop left.

Given a finite set of rules and a graph G, the number of matches is bounded by a polynomial in the size of G because the sizes of left-hand sides of rules are bounded by a constant. Given a match, the check, whether the dangling and the identification condition hold, and the construction of the directly derived graph is linear in the size of G. Therefore, polynomial time is needed to find a match and to construct a direct derivation, and there is a polynomial number of choices at most. Moreover, the size of the resulting graph differs from the size of the host graph by a constant.

3 Simple graph transformation units

A rule yields a binary relation on graphs and a set of rules a set of derivations. The example of Hamiltonian paths shows (like many other examples would show) that more features are needed to model algorithms and processes on graphs in a proper way. In particular one needs initial graphs to start the derivation process and terminal graphs to stop it. Initial and terminal graphs may be specified by graph class expressions. Moreover, further control conditions may be helpful to regulate the derivation process. The notion of simple graph transformation units encompasses all these features to model and compute relations between initial and terminal graphs by means of regulated derivations.

3.1 Graph class expressions

A *graph class expression* may be any syntactic entity X that specifies a class of graphs $SEM(X) \subseteq \mathcal{G}_\Sigma$. A typical example is a subset $\Delta \subseteq \Sigma$ with $SEM(\Delta) = \mathcal{G}_\Delta \subseteq \mathcal{G}_\Sigma$. Requested, forbidden, and reduced structures are also frequently used. Let F be a graph. Then $SEM(requested(F))$ consists of all graphs G containing a subgraph that is isomorphic to F. And $SEM(forbidden(F))$ consists of all graphs G such that there is no subgraph in G that is isomorphic to F. Another useful type of graph class expressions is given by sets of rules P where $SEM(reduced(P))$ contains all P-*reduced* graphs, i.e., graphs to which none of the rules in P can be applied. In the examples, we use the constant expression *unlabeled graphs* denoting the set of unlabeled graphs each node of which is equipped with a single unlabeled loop.

3.2 Control conditions

A *control condition* is any syntactic entity that cuts down the non-determinism of the derivation process. A typical example is a regular expression over a set of rules (or any other string-language-defining device). Let C be a regular expression specifying the language $L(C)$. Then a derivation with application sequence s is *permitted* by C if $s \in L(C)$. In the following, we consider no other control conditions.

3.3 Simple graph transformation units

A *simple graph transformation unit* is a system $tu = (I, P, C, T)$, where I and T are graph class expressions to specify the *initial* and the *terminal* graphs respectively, P is a finite set of rules, and C is a control condition.

Each such transformation unit tu specifies a binary relation

$$SEM(tu) \subseteq SEM(I) \times SEM(T)$$

that contains a pair (G, H) of graphs if and only if there is a derivation $G \overset{*}{\underset{P}{\Longrightarrow}} H$ permitted by C.

Example 3 The considerations in Examples 1 and 2 can be summarized by the following simple graph transformation unit:

$$\begin{array}{l} HP \\ \quad \text{initial} : \mathit{unlabeled\ graphs} \\ \quad \text{rules} : \mathit{start}, \mathit{run} \\ \quad \text{control} : \mathit{start}; \mathit{run}^* \\ \quad \text{terminal} : \mathit{forbidden}(\square) \end{array}$$

The initial graphs are unlabeled graphs with a single unlabeled loop at each node. The rules *start* and *run* are given in Example 2, and the control condition is a regular expression over the set of rules with the sequential composition ; and the Kleene star $*$ (specifying that a single application of *start* can be followed by an arbitrary sequence of applications of *run*). Graphs derived in this way from initial graphs are accepted as terminal if they do not contain any unlabeled loop.

In Fig. 4, the initial graph is semantically related to the four terminal graphs—the rightmost ones. They represent the four Hamiltonian cycles of the initial graph. All other derived graphs contain some forbidden loop.

3.4 Computation and complexity

Using the effective construction of direct derivations, the relation $SEM(tu)$ of a transformation unit $tu = (I, P, C, T)$ is recursively enumerable if $SEM(I)$ is recursively enumerable and $SEM(T)$ and the control condition are decidable. $SEM(tu)$ can be computed as follows:

- Enumerate the graphs of $SEM(I)$.
- For each $G \in SEM(I)$, enumerate all derivations starting in G together with their application sequences.
- For each derived graph $\overline{G}$, check whether $\overline{G} \in SEM(T)$.
- If yes, check whether the respective application sequence belongs to $L(C)$.
- If yes, put $(G, \overline{G})$ into $SEM(tu)$.

The assumptions apply to all graph class expressions and control conditions that are explicitly introduced above.

The time to check whether a graph G belongs to $SEM(\mathit{unlabeled\ graphs})$, $SEM(\mathit{forbidden}(F))$ or $SEM(\mathit{reduced}(P))$ is polynomial in the size of G. If $G \overset{k}{\Longrightarrow} H$ and k is bounded by a polynomial in the size of G, then the size of H is also bounded by a polynomial in the size of G. Therefore, to check whether H belongs to the graph $SEM(forbidden(F))$ or $SEM(reduced(P))$ takes also time that is polynomial in the size of G. Finally, the construction of the application sequence can be done together with the derivation without extra effort and its length coincides with the length of the derivation. The membership problem of regular expressions is linear in this length so that it is polynomial in the size of G.

The notion of a transformation unit has been introduced in Kreowski and Kuske (1999a, b) and in Kreowski et al. (1997) as a modeling and structuring concept for graph transformation systems. Here the structuring component is omitted and the computational aspect is emphasized. In addition to the cited papers, one can find more about graph class expressions and control conditions in Kuske (2000a, b). Habel and Plump (2001) have shown that a similar kind of graph transformation approach is computationally complete.

4 Solving decision problems

A simple graph transformation unit is terminating if, for every initial graph, the number of derivations starting in this graph is finite. In this case, all these derivations can be constructed effectively, and it can be checked whether any of them is permitted by the control condition and derives a terminal graph. This means that a terminating unit can be re-interpreted as a solution of a decision problem on the initial graphs. If the lengths of derivations are bounded by a polynomial in addition, one gets a graph-transformational variant of the class *NP* of decision problems with nondeterministic polynomial solutions.

Definition 1 Let $tu = (I, P, C, T)$ be a transformation unit. tu is *terminating* if, for each initial graph $G \in SEM(I)$, there is an upper bound $b(G) \in \mathbb{N}$ such that $n \leq b(G)$ for each derivation $G \overset{n}{\underset{P}{\Longrightarrow}} G'$. The function $b : SEM(I) \longrightarrow \mathbb{N}$ given in this way is called *termination bound*.

A well-known sufficient condition for termination can be used in the framework of graph transformation units.

Observation 2 Let $tu = (I, P, C, T)$ be a transformation unit. Let $val : \mathcal{G}_\Sigma \longrightarrow \mathbb{N}$ be a valuation function with $val(G') > val(G'')$ for each direct derivation $G' \underset{P}{\Longrightarrow} G''$. Then tu is terminating.

Proof Consider a derivation $G_0 \underset{P}{\Longrightarrow} G_1 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} G_n$ with $G_0 \in SEM(I)$. Due to the property of val, this yields $val(G_0) > val(G_1) > \cdots > val(G_n)$ and, therefore, $n \leq val(G_0)$. In other words, val is a termination bound for tu. □

Definition 3 Let $tu = (I, P, C, T)$ be a terminating transformation unit with the termination bound $b : SEM(I) \rightarrow \mathbb{N}$.

1. A function $D : SEM(I) \rightarrow \{\text{TRUE}, \text{FALSE}\}$ is called a *decision problem*.
2. *tu solves D* if the following holds for all $G \in SEM(I)$:

$$D(G) = \text{TRUE if and only if } (G, \overline{G}) \in SEM(tu) \text{ for some } \overline{G} \in SEM(T).$$

This is denoted by $COMP(tu) = D$.

3. tu is called *polynomial* if there is a polynomial p such that, for all $G \in SEM(I), b(G) \leq p(size(G))$ and the membership problems of $SEM(I)$ as well as of $SEM(T)$ are polynomial.

4. The class of all decision problems that are solved by polynomial transformation units is denoted by NP_{GT}.

Remarks

1. If *tu* is terminating, there is only a finite number of derivations $G \overset{*}{\underset{P}{\Longrightarrow}} G'$ for each $G \in SEM(I)$. Hence, it can be checked effectively whether a terminal graph is derived by a permitted derivation or not.
2. The computational framework given by terminating and polynomial transformation units in particular is still nondeterministic because there may be a derivation $G \overset{*}{\underset{P}{\Longrightarrow}} G'$ with $G' \in SEM(reduced(P))$, but $G' \notin SEM(T)$, and also a permitted derivation $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G}$ with $\overline{G} \in SEM(T)$. In the polynomial case, it takes polynomial time to build up a single derivation and to check whether its derived graph is terminal or not (cf. 3.4). Both points together justify the denotation NP_{GT}.

The same reasoning shows that a decision problem $D : SEM(I) \to \{\text{TRUE}, \text{FALSE}\}$ which is solved by a polynomial transformation unit belongs to the class of *NP*-problems if one chooses a proper string representation of graphs. Also the converse inclusion holds because one can simulate the computational steps of a Turing machine by the application of graph transformation rules. This consideration yields the following result.

Observation 4 $NP_{GT} = NP$.

Proof Let $D \in NP_{GT}$ and *tu* be a polynomial transformation unit solving *D*. Then each derivation has polynomial length, each step is constructed in polynomial time, and the number of choices per step is polynomially bounded. Moreover, the membership problem of *SEM*(*I*), *SEM*(*T*) and of *L*(*C*) is decidable in polynomial time. All facts remain true if one chooses proper string representations of the initial graphs like the sequences of nodes, edges, sources, and targets. This yields $D \in NP$.

Conversely, let $D \in NP$ and $TM = (Z, I, d, s_0, F)$ with $d \subseteq Z \times (I \cup \{\Box\}) \times Z \times (I \cup \{\Box\}) \times \{n, l, r\}$ be a Turing machine that solves *D* in a polynomial number of steps. Then *TM* is simulated by the transformation unit *tu*(*TM*) = (*init*(*TM*), *P*(*TM*), *free*, *terminal*(*TM*)) the components of which are defined as follows.

1. The graph class expression *init*(*TM*) describes the class of graphs each member of which is a disjoint union of the two graphs $G(TM, s_0)$ and $w^\bullet$ for some $w \in I^*$. The graph $G(TM, s_0)$ is the state graph of *TM* with s_0 as current state:

$$G(TM, s_0) = (Z, d \cup \{s_0, curr\} \cup F, s, t, l)$$

with $s|_d = proj_1, t|_d = proj_3, l|_d = \langle proj_2, proj_4, proj_5\rangle, s(s_0) = t(s_0) = s_0 = s(curr) = t(curr), s(s'') = t(s'') = s''$ for all $s'' \in F$, and $l(s_0) = init, l(curr) = curr$, and $l(s'') = fin$ for all $s'' \in F$. This means that each state transition (s, x, s', y, m) is represented by an edge with source s, target s' and (x, y, m) as label. Moreover, there are some loops indicating the initial state, the current state being initially the initial one, and the terminal states. The graph $w^\bullet$ for $w = a_1 \ldots a_n \in I^*$ represents the Turing band inscribed initially with $w \neq \lambda$. Moreover, there are two loops indicating the left end and the right end of the band, and there is a further edge that plays the role of the read/write-head:

$$w^\bullet = (\{0\} \cup [n], [n] \cup \{\}, s^\bullet, t^\bullet, l^\bullet)$$

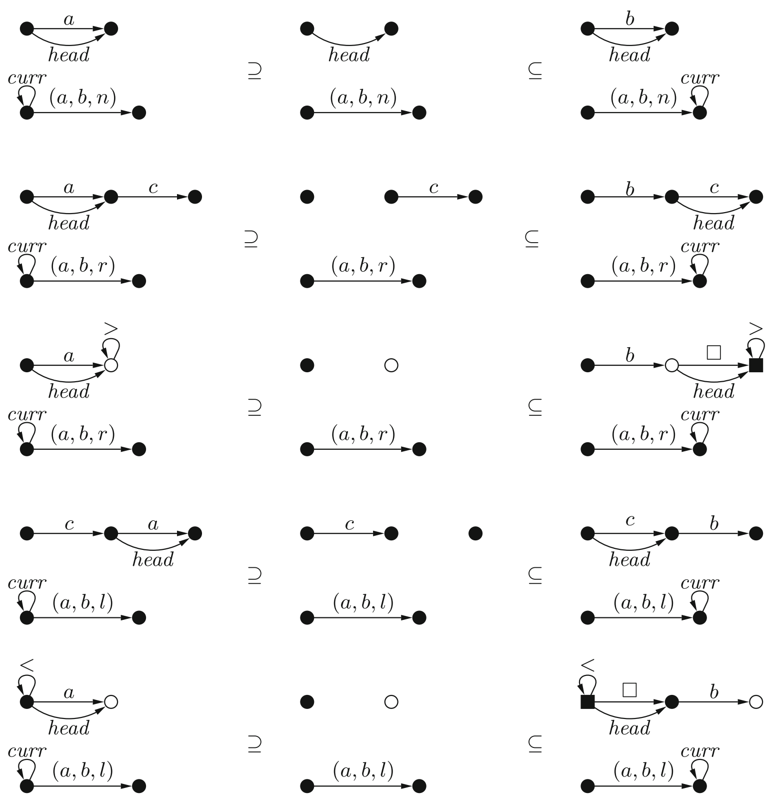


Fig. 5 The rules of *tu*(*TM*)

with

- $s^\bullet(i) = i - 1$, $t^\bullet(i) = i$ and $l^\bullet(i) = a_i$ for $i \in [n]$,
- $s^\bullet(<) = t^\bullet(<) = 0, l^\bullet(<) = \ <, s^\bullet(\ >\ ) = t^\bullet(\ >\ ) = n, l^\bullet(\ >\ ) = \ >$,
- $s^\bullet(h) = 0, t^\bullet(h) = 1$ and $l^\bullet(h) = head$.

For $w = \lambda$ we use the graph $\Box^\bullet$.

2. *P*(*TM*) contains the rules given in Fig. 5. Obviously, an application of a rule mimics a computational step of *TM* moving from the current state to a follow-up state, reading the symbol *a* and writing *b* at the head, and keeping the head or moving to the next symbol to the right or to the left. If one of the end markers is reached the band is extended by the extra symbol □.
3. *free* is a control condition that does not invoke any restriction. As a regular expression, it may be chosen as the arbitrary iteration of the alternative composition of all rules.
4. The graph class expression *terminal*(*TM*) specifies the class of all graphs with some node that is incident with a *curr*-loop as well as a *fin*-loop.

If $D(w) =$ TRUE, then *TM* runs on *w* as initial inscription of the band from the initial state to some final state in a polynomial number of steps. Starting in $G(TM, s_0) + w^\bullet$, each step of *TM* corresponds to a rule application in *tu*(*TM*) reaching a graph where the current-state loop is attached to a terminal state. Conversely, a derivation from $G(TM, s_0) + w^\bullet$ for some

$w \in I^*$ to a terminal graph corresponds to a run of *TM* on *w* with the same number of steps. As *TM* is polynomial, *tu*(*TM*) is also polynomial. Consequently, one gets $D \in NP_{GT}$ up to the representation of *w* by $G(TM, s_0) + w^\bullet$. □

Example 4 The rules *start* and *run* (cf. Example 2) decrease the number of unlabeled loops by 1 whenever one of them is applied. Therefore, the unit *HP* (cf. Example 3) is terminating due to Observation 2. Because *HP* finds all existing Hamiltonian paths of every initial graph as terminal graphs, *HP* solves the Hamiltonian path problem. Moreover the termination bound is linear so that the problem is explicitly shown to be a member of NP_{GT}.

Termination has been studied in the context of graph transformation for example by Plump (1998), Godard et al. (2002), and by Ehrig et al. (2005).

5 Graph multiset transformation

In this section, graph multiset transformation is introduced employing ordinary graph transformation as basis. The underlying data structures are finite multisets of graphs.[2] In each derivation step, some of the graphs of a given actual multiset are directly derived into graphs by applying ordinary rules, yielding a new actual multiset where the deriving graphs are replaced by the derived ones. This idea is formalized in Definition 5. A derivation of multisets of graphs corresponds to a set of simultaneous derivations of graphs (cf. Construction 6). In this sense, graph multiset transformation is a framework for massively parallel computation. In particular, one can show that *NP*-problems can be solved by graph multiset transformation in a polynomial number of steps.

Definition 5 Let *P* be a set of rules. Let $M : \mathcal{G}_\Sigma \to \mathbb{N}$ be a finite multiset of graphs and $M' \leq M$ a sub-multiset of *M*. Let $G_1 \ldots G_m \in Perm(M')$ be one of the sequential representations of M' and $G'_1 \ldots G'_m \in \mathcal{G}^*_\Sigma$ be another sequence of graphs with $G_i \underset{P}{\Longrightarrow} G'_i$ for all $i = 1, \ldots, m$. Let $M'' = [G'_1 \ldots G'_m]$ be the multiset of $G'_1 \ldots G'_m$.

Then *M directly derives* the graph multiset $\overline{M} = M - M' + M''$, denoted by $M \underset{P}{\Longrightarrow} \overline{M}$.

A sequence $M_0 \underset{P}{\Longrightarrow} M_1 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} M_n$ of direct derivations of multisets of graphs defines a *(graph multiset) derivation* from $M = M_0$ to $\overline{M} = M_n$ of length *n* in the usual way. Such derivations are shortly denoted by $M \overset{n}{\underset{P}{\Longrightarrow}} \overline{M}$ or $M \overset{*}{\underset{P}{\Longrightarrow}} \overline{M}$. The subscript *P* may be omitted if it is clear from the context.

Remarks

1. It should be noted that the derived multiset does not depend on the choice of the sequential representation of M' because each permutation of the sequence $G_1 \ldots G_n$ corresponds to the respective permutation of $G'_1 \ldots G'_n$ and the multisets of sequences are invariant with respect to permutation.
2. The sub-multiset $M' \leq M$ in a direct derivation is called *left-active* and $M'' \leq \overline{M}$ *right-active* respectively.
3. Let $Der = (M_0 \underset{P}{\Longrightarrow} M_1 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} M_n)$ be a graph multiset derivation. For $i = 1, \ldots, n-1$, let $M'_i \leq M_i$ be the left-active sub-multiset of the direct derivation $M_i \underset{P}{\Longrightarrow} M_{i+1}$ in *Der* and let $M''_i \leq M_i$ be the right-active sub-multiset of the direct

[2] The definitions concerning multisets are given in the Appendix.

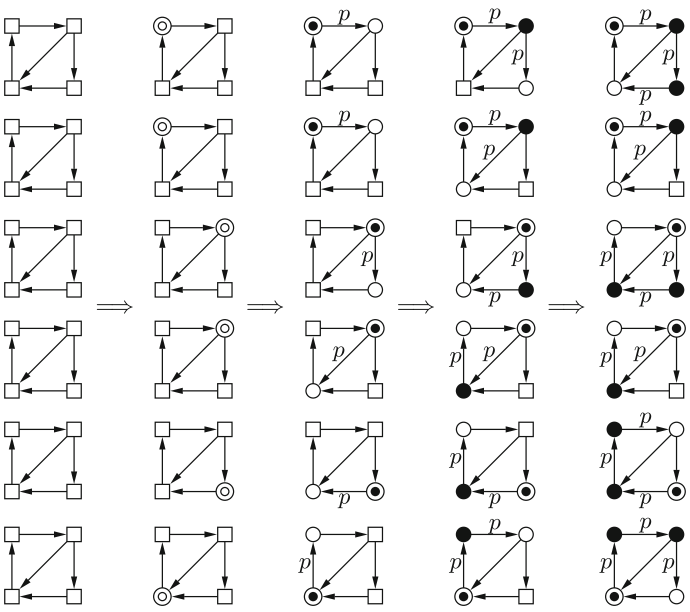


Fig. 6 A graph multiset derivation

derivation $M_{i-1} \underset{P}{\Longrightarrow} M_i$ in *Der*. Then *Der* is called *proactive* if $M''_i \leq M'_i$ for $i = 1, \ldots, n-1$. This means that a graph that is not transformed in some step is kept invariant in all further steps.

4. The left-active sub-multiset M' of a direct derivation $M \underset{P}{\Longrightarrow} \overline{M}$ may be empty so that $M'' = \mathbf{0}$ and $M = \overline{M}$. Such a direct derivation is called *inactive*. A derivation without inactive steps is called *undelayed*.

Multiset transformation as introduced is not restricted to graph transformation as underlying computational framework. It would work for any binary relation on some set of configurations where the relation provides computational steps. Nevertheless, we are interested in a graph transformational counterpart to DNA computing. Therefore, we focus on graphs as molecules and rule applications as chemical reactions.

Example 5 The six derivations in Fig. 4 can be turned into the graph multiset derivation in Fig. 6 if one considers the columns of graphs as multisets and adds copies of the second graph and the fourth graph of the last-but-one column to the last column. The derivation is proactive because no graph is kept invariant in the first three steps.

It is not difficult to see that graph multiset derivations correspond to derivations of the graphs in the multisets and that the lengths of graph multiset derivations are bounded if and only if the lengths of graph derivations are bounded. The correspondence is based on two constructions: Each graph multiset derivation induces a multiset of derivations and the other way round.

Construction 6

1. Let $M \overset{k}{\underset{P}{\Longrightarrow}} \overline{M}$ be a graph multiset derivation and $G_1 \ldots G_n$ be a sequential representation of M, i.e., $G_1 \ldots G_n \in Perm(M)$. Then there is a multiset of derivations

$MoD(M \overset{k}{\underset{P}{\Longrightarrow}} \overline{M})$: $[G_1 \overset{k_1}{\underset{P}{\Longrightarrow}} \overline{G}_1 \ldots G_n \overset{k_n}{\underset{P}{\Longrightarrow}} \overline{G}_n]$ with $k_i \leq k$ for each $i = 1, \ldots, n$ such that $[\overline{G}_1 \ldots \overline{G}_n] = \overline{M}$. Moreover, $k \leq \sum_{i=1}^{n} k_i$ if $M \overset{k}{\underset{P}{\Longrightarrow}} \overline{M}$ is undelayed. $MoD(M \overset{k}{\underset{P}{\Longrightarrow}} \overline{M})$ can be constructed inductively as follows.
Induction base: $M \overset{0}{\underset{P}{\Longrightarrow}} \overline{M}$ implies $\overline{M} = M$ yielding the derivations

$$G_i \overset{0}{\underset{P}{\Longrightarrow}} G_i = \overline{G}_i$$

for all $i = 1, \ldots, n$ with $[\overline{G}_1 \ldots \overline{G}_n] = [G_1 \ldots G_n] = M = \overline{M}$.
Induction step: Consider $M \overset{k+1}{\underset{P}{\Longrightarrow}} \overline{M}$ which decomposes into $M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M}$ and $\hat{M} \underset{P}{\Longrightarrow} \overline{M}$. By induction hypothesis, one gets derivations $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i$ with $k_i \leq k$ for all $i = 1, \ldots, n$ and $[\hat{G}_1 \ldots \hat{G}_n] = \hat{M}$. Moreover, there are sub-multisets $M' \leq \hat{M}$ and $M'' \leq \overline{M}$ with $\overline{M} = \hat{M} - M' + M''$. Without loss of generality, one can assume that M' is sequentially represented by $\hat{G}_1 \ldots \hat{G}_m$ for some $m \leq n$. Then there are direct derivations $\hat{G}_i \underset{P}{\Longrightarrow} G''_i$ for $i = 1, \ldots, m$ with $[G''_1 \ldots G''_m] = M''$. Consequently, one can construct the following derivations: $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i \underset{P}{\Longrightarrow} G''_i = \overline{G}_i$ for $i = 1, \ldots, m$ and $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i = \overline{G}_i$ for $i = m+1, \ldots, n$ with lengths not longer than $k + 1$ and

$$[\overline{G}_1 \ldots \overline{G}_n] = [G''_1 \ldots G''_m \hat{G}_{m+1} \ldots \hat{G}_n] = \hat{M} - M' + M'' = \overline{M}.$$

This completes the construction, i.e., the resulting multiset of derivations is equal to

$$[G_1 \overset{k'_1}{\underset{P}{\Longrightarrow}} \overline{G}_1 \ldots G_n \overset{k'_n}{\underset{P}{\Longrightarrow}} \overline{G}_n]$$

where $k'_i = k_i + 1$ for $i = 1, \ldots, m$ and $k'_i = k_i$ for $i = m+1, \ldots, n$.
Moreover, if $M \overset{k+1}{\Longrightarrow} \overline{M}$ is undelayed, then $m \geq 1$, $M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M}$ is undelayed, and by induction hypothesis $k \leq \sum_{i=1}^{n} k_i$. Hence, $k + 1 \leq \sum_{i=1}^{n} k_i + 1 \leq \sum_{i=1}^{n} k_i + m = \sum_{i=1}^{n} k'_i$.

2. Let *DER* be a multiset of derivations and let

$$G_1 \overset{k_1}{\underset{P}{\Longrightarrow}} \overline{G}_1 \ldots G_n \overset{k_n}{\Longrightarrow} \overline{G}_n \in Perm(DER)$$

be a sequential representation of *DER*. Then there is a graph multiset derivation $gmd(DER)$: $[G_1 \ldots G_n] \overset{k}{\underset{P}{\Longrightarrow}} [\overline{G}_1 \ldots \overline{G}_n]$ with $k = \max\{k_i \mid i = 1, \ldots, n\}$, which can be constructed as follows.
Let $G_i = G_{i0} \underset{P}{\Longrightarrow} G_{i1} \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} = G_{ik_i} = \overline{G}_i$ for $i = 1, \ldots, n$ be the given derivations. Let $G_{ij} = G_{ik_i}$ for $i = 1, \ldots, n$ and $j = k_i + 1, \ldots, k$. Then this induces

$$gmd(DER) : [G_{10} \ldots G_{n0}] \underset{P}{\Longrightarrow} [G_{11} \ldots G_{n1}] \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} [G_{1k} \ldots G_{nk}]$$

where,
- $[G_{10} \ldots G_{n0}] = [G_1 \ldots G_n]$,
- $[G_{1k} \ldots G_{nk}] = [G_{1k_1} \ldots G_{nk_n}] = [\overline{G}_1 \ldots \overline{G}_n]$, and
- for $j = 1, \ldots, k$, the direct graph multiset derivation

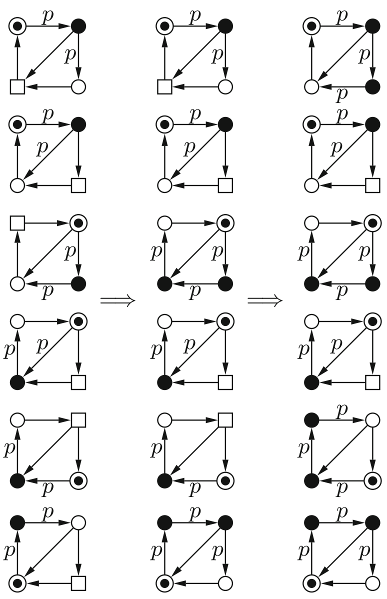


Fig. 7 A graph multiset derivation that is not proactive

$$[G_{1(j-1)} \ldots G_{n(j-1)}] \underset{P}{\Longrightarrow} [G_{1j} \ldots G_{nj}]$$

is defined by the direct derivations $G_{1(j-1)} \underset{P}{\Longrightarrow} G_{ij}$ for $i = 1, \ldots, n$ and $j \leq k_i$.

Example 6 Let Der_0 be the graph multiset derivation in Fig. 6. Then the multiset of derivations $MoD(Der_0)$ consists of the six derivations in Fig. 4. Let the (multi-)set of these derivations be denoted by DER_0. Then the induced graph multiset derivation $gmd(DER_0)$ is the one in Fig. 6. Besides Der_0, there are many other graph multiset derivations with the same multiset of derivations. For instance, one may replace the last step in Fig. 6 by the derivation in Fig. 7, which is not proactive, in particular. If the resulting derivation is denoted by Der_1, then $MoD(Der_1) = DER_0$.

Observation 7 Let *DER* be a multiset of derivations and *Der* be a graph multiset derivation. Then the following properties hold.

1. $MoD(gmd(DER)) = DER$,
2. in particular, $MoD(gmd(MoD(Der))) = MoD(Der)$,
3. $gmd(DER)$ is proactive,
4. in particular, $gmd(MoD(Der))$ is proactive,
5. if *Der* is proactive, then $Der = gmd(MoD(Der))$.

Proof

1. Let $G_1 \overset{k_1}{\underset{P}{\Longrightarrow}} \overline{G}_1 \ldots G_n \overset{k_n}{\underset{P}{\Longrightarrow}} \overline{G}_n \in Perm(DER)$. Let $\max = \max\{k_i \mid i = 1, \ldots, n\}$. The statement is proved by induction on max.
   Induction base: $\max = 0$ implies $k_i = 0$ and $G_i = \overline{G}_i$ for all $i = 1, \ldots, n$. Hence, $gmd(DER) = [G_1 \ldots G_n] \overset{0}{\underset{P}{\Longrightarrow}} [G_1 \ldots G_n] = [\overline{G}_1 \ldots \overline{G}_n]$ implying

$$MoD(gmd(DER)) = [G_1 \overset{0}{\underset{P}{\Longrightarrow}} G_1 = \overline{G}_1 \ldots G_n \overset{0}{\underset{P}{\Longrightarrow}} G_n = \overline{G}_n]$$

by definition of *gmd* and *MoD*. This proves $DER = MoD(gmd(DER))$ for $k = 0$.
Induction step: Consider $\max = k+1$. For $i = 1,\ldots,n$, let $(G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \overline{G}_i) = (G_i \overset{k}{\underset{P}{\Longrightarrow}} G'_i \underset{P}{\Longrightarrow} \overline{G}_i)$ for $k_i = k+1$ and $k'_i = k_i$ and $G'_i = \overline{G}_i$ for $k_i \leq k$. By induction hypothesis, one gets $MoD(gmd(DER')) = DER'$ for $DER' = [G_1 \overset{k'_1}{\underset{P}{\Longrightarrow}} G'_1 \ldots G_n \overset{k'_n}{\underset{P}{\Longrightarrow}} G'_n]$ with $k'_i = k$ in the case of $k_i = k+1$. The derivation $gmd(DER') = ([G_1 \ldots G_n] \overset{k}{\underset{P}{\Longrightarrow}} [G'_1 \ldots G'_n])$ can be extended by

$$[G'_1 \ldots G'_n] \underset{P}{\Longrightarrow} [\overline{G}'_1 \ldots \overline{G}'_n]$$

using the direct derivations $G'_i \underset{P}{\Longrightarrow} \overline{G}_i$ for $i = 1,\ldots,n$ with $k_i = k+1$. The resulting derivation is $gmd(DER)$ according to the construction of *gmd*, and $MoD(gmd(DER))$ contains the derivations $G_i \overset{k}{\underset{P}{\Longrightarrow}} G'_i \underset{P}{\Longrightarrow} \overline{G}_i$ for $k_i = k+1$ and $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} G'_i = \overline{G}_i$ otherwise proving the statement 1 for $k+1$ if it holds for k.

2. Statement 2 is a special case of statement 1.
3. If *DER* contains a derivation $G_{i0} \underset{P}{\Longrightarrow} G_{i1} \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} G_{ik_i}$, then $gmd(DER)$ uses k_i rule applications in the first k_i steps in this order. If k_i is not the maximum length in *DER*, then G_{ik_i} is kept invariant in all further steps. In other words, $gmd(DER)$ is proactive.
4. Statement 4 is a special case of statement 3.
5. Consider $Der = (M \overset{k}{\underset{P}{\Longrightarrow}} \overline{M})$ and $G_1 \ldots G_n \in Perm(M)$. The statement is proved by induction on k.

   Induction base: $k = 0$ implies $\overline{M} = M$. Therefore,

$$MoD(Der) = [G_1 \overset{0}{\underset{P}{\Longrightarrow}} G_1 \ldots G_n \overset{0}{\underset{P}{\Longrightarrow}} G_n]$$

and $gmd(MoD(Der)) = (M = [G_1 \ldots G_n] \overset{0}{\underset{P}{\Longrightarrow}} [G_1 \ldots G_n] = M = \overline{M})$ proving the statement for $k = 0$.
Induction step: $Der = (M \overset{k+1}{\underset{P}{\Longrightarrow}} \overline{M}) = (M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M} \underset{P}{\Longrightarrow} \overline{M})$. Because *Der* is proactive, $M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M}$ is proactive as initial section, too. By induction hypothesis, one gets $gmd(MoD(M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M})) = (M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M})$. Let $M' \leq \hat{M}$ be left-active and $M'' \leq \overline{M}$ be right-active with respect to $\hat{M} \underset{P}{\Longrightarrow} \overline{M}$. Let $G'_1 \ldots G'_m \in Perm(M')$ and $G'_j \underset{P}{\Longrightarrow} G''_j$ for $j = 1,\ldots,m$ the respective direct derivations. Let $MoD(M \overset{k}{\underset{P}{\Longrightarrow}} \hat{M}) = [G_1 \overset{k_1}{\underset{P}{\Longrightarrow}} \hat{G}_1 \ldots G_n \overset{k_n}{\underset{P}{\Longrightarrow}} \hat{G}_n]$. Without loss of generality, one may assume that the permutation is chosen in such a way that $G'_1 \ldots G'_m = \hat{G}_1 \ldots \hat{G}_m$. Then, by construction, $MoD(Der)$ contains the derivations $G_i \overset{k}{\underset{P}{\Longrightarrow}} \hat{G}_i = G'_i \underset{P}{\Longrightarrow} G''_i = \overline{G}_i$ for $i = 1,\ldots,m$ and $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i = \overline{G}_i$ for $i = m+1,\ldots,n$. Note that $k_i = k$ for $i = 1,\ldots,m$ because of

the assumed proactivity. It follows that $gmd(MoD(Der)) = (M = [G_1 \ldots G_n] \overset{k+1}{\underset{P}{\Longrightarrow}} [\overline{G}_1 \ldots \overline{G}_n] = \overline{M}) = Der$ proving the statement for $k + 1$ if it holds for k. □

The construction of a multiset of derivations from a derivation of multisets of graphs and vice versa is mainly used to compare the two kinds of computation given by the two kinds of derivations. We do not claim that the multiset of derivations has a DNA analogon of any kind. In particular, the synchronization of the derivation can be done (and is done) step by step without taking any reaction speed into account. Observation 7 means, in particular, that graph multiset transformation is a kind of parallel graph transformation that has the same termination properties as ordinary graph transformation discussed above if one considers proactive derivations. Therefore, graph multiset transformation can be used as a computational framework similarly to graph transformation. In particular, a terminating transformation unit can solve a decision problem on its initial graphs by means of graph multiset transformation. The computation starts with multiple copies of an initial graph and yields TRUE if some terminal graph occurs in one of the derived multisets. Using polynomial transformation units, the lengths of proactive graph multiset derivations are polynomially bounded and TRUE is computed in a single derivation with high probability if the multiplicity of the initial graph is chosen large enough.

Definition 8 Let $tu = (I, P, C, T)$ be a terminating transformation unit. Let $D : SEM(I) \to \{\text{TRUE}, \text{FALSE}\}$ be a decision problem. Then *tu computes D by graph multiset transformation* (GMST) if the following holds.

For each $G \in SEM(I)$, there is a proactive graph multiset derivation $Der = (n \cdot [G] \overset{*}{\underset{P}{\Longrightarrow}} \overline{M})$ with initial multiplicity n for some $n \in \mathbb{N}$ so that some underlying derivation $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G} \in MoD(Der)$ is permitted by C with $\overline{G} \in car(\overline{M}) \cap SEM(T)$ if and only if $D(G) = \text{TRUE}$.

Remarks

1. If tu computes D by graph multiset transformation, then this may be denoted by $D = COMP_{GMST}(tu)$.
2. *PGMST* denotes the set of all decision problems that are computed via graph multiset transformation by polynomial transformation units.
3. If tu is polynomial and G an initial graph, then the number of derivations starting in G is bounded by a number exponential in the size of G so that—in general—there is no feasible way to check all of them sequentially.
4. If the initial multiplicity is 1, then there is no difference between graph transformation and graph multiset transformation because the singleton x and the multiset $[x]$ with the single element x provide the same information—for example the same nondeterminism if x is a derivation.
5. If the multiplicity n of G is chosen large and the multiset derivation

$$Der = (n \cdot [G] \overset{*}{\underset{P}{\Longrightarrow}} \overline{M})$$

is long, then the probability is high that some successful derivation starting in graph G is in $MoD(Der)$. Therefore the probability is high to find the proper value of $D(G)$ in a single graph multiset derivation with a polynomial number of steps. This justifies the denotation *PGMST*.

6. To make this more precise, let N be the number of derivations $G \overset{*}{\Longrightarrow} H$ that cannot be prolonged and K the number of successful derivations. Then the probability that a single derivation out of the N ones is successful is $p_1 = \frac{K}{N}$. If *Der* cannot be prolonged, then *MoD*(*Der*) contains n test derivations that are constructed independently of each other so that one could construct them one after the other and number them from 1 to n without loss of information. Let p_i denote the probability that the derivations 1 to i contain a successful derivation. Then the derivation $i + 1$ improves the chance to get a successful one by $p_i(1 - p_i)$ yielding $p_{i+1} = p_i + p_i(1 - p_i)$. By induction, one gets for $1 \leq i \leq n,\ n \in \mathbb{N}$

$$p_i = \sum_{j=0}^{i-1} \frac{K}{N} \cdot \left(\frac{N-K}{N}\right)^j.$$

These are the finite prefixes of a geometric series with the constant ratio $\frac{N-K}{N}$ so that a well-known fact and a bit of arithmetic leads to

$$p_i = \frac{K}{N} \cdot \frac{1 - \left(\frac{N-K}{N}\right)^i}{1 - \frac{N-K}{N}} = 1 - \left(\frac{N-K}{N}\right)^i.$$

This means, in particular, that the sequence $(p_i)_{i \geq 1}$ converges against 1. In other words, a graph multiset derivation finds a successful derivation with arbitrarily high probability that is the better the larger the initial multiplicity is. The case where none of the N possible derivations is successful (i.e. $D(G) = \text{FALSE}$) can be treated similarly.

As a first result on polynomial graph multiset transformation and as the main result of this section, one can show that the classes NP_{GT} and *PGMST* coincide. Unfortunately, this is not a solution of the $P = NP$-problem because the class *PGMST* relies on massive parallelism. The equality of the classes NP_{GT} and *PGMST* is based on the following observation.

Observation 9 Let *tu* be a polynomial transformation unit. Then *COMP*(*tu*) = $COMP_{GMST}$(*tu*).

Proof Let *tu* = (*I*, *P*, *C*, *T*) be a polynomial graph transformation unit and let $G \in SEM(I)$ such that $COMP(tu)(G) = \text{TRUE}$. Then by definition there is a derivation $der = (G = G_0 \underset{P}{\Longrightarrow} G_1 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} G_n)$ with $G_n \in SEM(T)$ that is permitted by C. This induces the graph multiset derivation $Der = ([G] = [G_0] \underset{P}{\Longrightarrow} [G_1] \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} [G_n])$ with $der \in MoD(Der) = [der]$ such that $COMP_{GMST}(tu)(G) = \text{TRUE}$. If, conversely, $COMP_{GMST}(tu)(G) = \text{TRUE}$, then by definition there are a graph multiset derivation $Der = (n \cdot [G] \overset{*}{\underset{P}{\Longrightarrow}} \overline{M})$ and a derivation $der = (G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G}) \in MoD(Der)$ which is permitted by C with $\overline{G} \in car(\overline{M}) \cap SEM(T)$. The derivation *der* yields $COMP(tu)(G) = \text{TRUE}$ such that, all together, *COMP*(*tu*) = $COMP_{GMST}$(*tu*). □

From this observation it follows that the decision problems in NP_{GT} are the same as those in *PGMST*.

Corollary 10 $NP_{GT} = PGMST$.

Proof $D \in NP_{GT}$ if and only if there is a polynomial graph transformation unit *tu* = (*I*, *P*, *C*, *T*) with *D* = *COMP*(*tu*). By Observation 9 *D* = $COMP_{GMST}$(*tu*). This is in turn the case if and only if $D \in PGMST$. Hence, $NP_{GT} = PGMST$. □

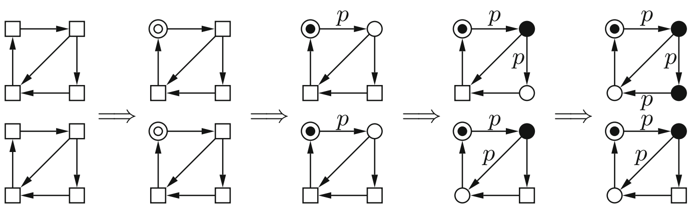
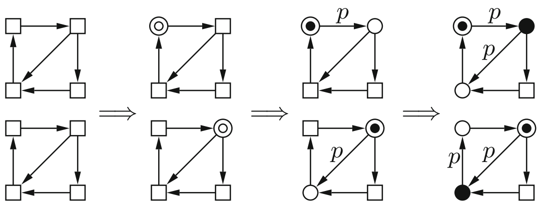


Fig. 8 A graph multiset derivation with terminal graphs

Fig. 9 A graph multiset derivation without terminal graphs



Example 7 Based on the unit *HP* in Example 3 of Sect. 3, Figure 8 shows a graph multiset derivation that starts with two copies of G_0. In the first step, the rule *start* is applied to the left upper node of both copies. There is only one possible match in each case exept for the third step where *run* is applied to the right vertical edge in the upper graph and to the diagonal edge in the lower graph. In the following steps, *run* is applied as long as possible. The horizontal rows of graphs represent the underlying derivations which are both permitted. The derived (multi-)set contains two graphs of which one graph is terminal proving that $COMP_{GMST}(HP)(G_0) = \text{TRUE}$.

The derivation in Fig. 6 yields the same result as it covers the one in Fig. 8. In contrast to that, the derivation in Fig. 9 fails because both derived graphs are not terminal.

The section is closed by a more explicit construction of the computations that solve decision problems by graph multiset transformation. To keep track of underlying derivations that are permitted by the control condition, a finite automaton is used. Moreover, we assume that terminal graphs are reduced. Therefore, the check for terminality can be postponed until a derivation cannot be prolonged.

Construction 11 Let *tu* = (*I*, *P*, *C*, *T*) be a terminating transformation unit with $SEM(T) \subseteq reduced(P)$. Let *A* = (*S*, *P*, *d*, s_0, *F*) be a finite automaton with *L*(*A*) = *L*(*C*) where *P* is the input alphabet.

As underlying data structures, configurations of the form $(M \overset{*}{\underset{P}{\Longrightarrow}} \overline{M}, W, w)$ with $W \in Perm(\overline{M})$ and $w \in S^*$ are used. In addition, one may assume *length*(*W*) = *length* (*w*) so that each copy of each graph in $\overline{M}$ is associated with a state of A. Given an initial graph *G*, a computation can be constructed inductively in the following way.

Induction base: Choose *n*, and consider $(n \cdot [G] \overset{0}{\underset{P}{\Longrightarrow}} n \cdot [G], G^n, s_0^n)$ as start configuration.
Induction hypothesis: Assume that a configuration

$$(n \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} \hat{M}, \hat{G}_1 \ldots \hat{G}_n, s_1 \ldots s_n)$$

is already constructed.

Induction step: If possible, then choose for $i = 1, \ldots, n, \hat{G}_i \Longrightarrow \overline{G}_i$ with some $\overline{s}_i \in d(s_i, r)$. Otherwise, let $\overline{G}_i = \hat{G}_i$ and $\overline{s}_i = s_i$. Then $[\hat{G}_1 \ldots \hat{G}_n] \overset{r}{\underset{P}{\Longrightarrow}} [\overline{G}_i \ldots \overline{G}_n]$ is a direct derivation giving rise to the follow-up configuration

$$(n \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} \hat{M} \underset{P}{\Longrightarrow} [\overline{G}_1 \ldots \overline{G}_n], \overline{G}_1 \ldots \overline{G}_n, \overline{s}_1 \ldots \overline{s}_n).$$

The construction can be terminated if a configuration

$$(n \cdot [G] \overset{l}{\underset{P}{\Longrightarrow}} \overline{M}, \overline{G}_1 \ldots \overline{G}_n, \overline{s}_1 \ldots \overline{s}_n)$$

is reached such that there is no $\overline{G}_i \underset{r}{\Longrightarrow} \overline{\overline{G}}_i$. Consequently, all follow-up configurations remain unchanged. Such a configuration is reached eventually because the transformation unit *tu* is terminating.

Observation 12 Let *tu* = (*I*, *P*, *C*, *T*) be a terminating transformation unit with $SEM(T) \subseteq reduced(P)$. Let *A* = (*S*, *P*, *d*, s_0, *F*) be a finite automaton with *L*(*A*) = *L*(*C*). Let *D* = $COMP_{GMST}$(*tu*). Then the following statements hold.

1. Let $(n \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} \overline{M}, \overline{G}_1 \ldots \overline{G}_n, s_1 \ldots s_n)$ be a configuration constructed inductively according to Construction 11. Let, for $i = 1, \ldots, n, u_i$ be the application sequence of the underlying derivation $G \overset{k_i}{\underset{P}{\Longrightarrow}} \overline{G}_i$. Then $s_i \in d^*(s_0, u_i)$.
2. Let $(n \cdot [G] \overset{*}{\underset{P}{\Longrightarrow}} \overline{M}, \overline{G}_1 \ldots \overline{G}_n, s_1 \ldots s_n)$ be a terminated configuration for some graph $G \in SEM\ (I)$ with $\overline{G}_i \in SEM(T)$ and $s_i \in F$ for some $i = 1, \ldots, n$. Then $D(G) =$ TRUE.
3. If $D(G) =$ TRUE, then there is a terminated configuration of the form $(1 \cdot [G] \overset{*}{\Longrightarrow} (1 \cdot [\overline{G}], \overline{G}, s)$ for some $\overline{G} \in SEM(T)$ and $s \in F$.

Proof

1. Statement 1 is proved by induction on *k*.

   Induction base: $(n \cdot [G] \overset{0}{\underset{P}{\Longrightarrow}} \overline{M}, \overline{G}_1 \ldots \overline{G}_n, s_1 \ldots s_n)$ implies

$$\overline{M} = M, \overline{G}_1 = \cdots = \overline{G}_n = G, \text{ and } s_1 = \cdots = s_n = s_0.$$

Then $(G \overset{0}{\underset{P}{\Longrightarrow}} G)$ is the only derivation in $MoD(n \cdot [G] \overset{0}{\underset{P}{\Longrightarrow}} \overline{M})$ with application sequence λ such that $s_0 \in d^*(s_0, \lambda) = \{s_0\}$.

Induction step: $(n \cdot [G] \overset{k+1}{\underset{P}{\Longrightarrow}} \overline{M}, \overline{G}_1 \ldots \overline{G}_n, \overline{s}_1 \ldots \overline{s}_n)$ is obtained from

$$(n \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} \hat{M}, \hat{G}_1 \ldots \hat{G}_n, s_1 \ldots s_n) \text{ and } \hat{M} \underset{P}{\Longrightarrow} \overline{M}$$

where $\hat{G}_i \underset{r_i}{\Longrightarrow} \overline{G}_i$ is used with $\overline{s}_i \in d(s_i, r_i)$ if possible, and $G_i = \overline{G}_i$ and $s_i = \overline{s}_i$ otherwise. By induction hypothesis, one knows that $s_i \in d^*(s_0, u_i)$ where u_i is the application sequence of the derivation $G \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i \in MoD(n \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} \hat{M})$ for each $i = 1, \ldots, n$. Then the derivation of $MoD(n \cdot [G] \overset{k+1}{\underset{P}{\Longrightarrow}} \overline{M})$ has the following form for each $i = 1, \ldots, n$: $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i \underset{r_i}{\Longrightarrow} \overline{G}_i$ with application sequence $u_i r_i$ or $G_i \overset{k_i}{\underset{P}{\Longrightarrow}} \hat{G}_i = \overline{G}_i$ with application sequence u_i. In the first case, one gets $\overline{s}_i \in d(s_i, r_i)$ and $s_i \in d^*(s_0, u_i)$ which implies $\overline{s}_i \in d^*(s_0, u_i r_i)$. And in the second case, one gets $\overline{s}_i = s_i \in d^*(s_0, u_i)$. Both together prove statement 1 for $k+1$ if it holds for k.

2. Let $G \overset{k_i}{\underset{P}{\Longrightarrow}} \overline{G}_i$ with $G \in SEM(I), \overline{G}_i \in SEM(T)$ and $s_i \in F$ be a derivation of $MoD(n \cdot [G] \overset{*}{\underset{P}{\Longrightarrow}} \overline{M})$. If the given configuration is terminated, then $\overline{G}_i \in reduced(P)$ for all $i = 1, \ldots, n$. Due to statement 1, $s_i \in d^*(s_0, u_i)$ for the application sequence of $G \overset{k_i}{\underset{P}{\Longrightarrow}} \overline{G}_i$ meaning that $u_i \in L(A) = L(C)$ and that, therefore, the derivation is permitted by C. Consequently, $D(G) = \text{TRUE}$.
3. By assumption and Corollary 10, $D \in P_{GMST} = NP_{GT}$. Hence, $D(G) = \text{TRUE}$ for some $G \in SEM(I)$ implies the existence of a derivation $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G}$ with

$$\overline{G} \in reduced(P) \cap SEM(T)$$

permitted by C. The latter means that $d^*(s_0, u) \cap F \neq \emptyset$ if u is the application sequence of $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G}$. It remains to show the following: Let $G \overset{*}{\underset{P}{\Longrightarrow}} G'$ be a derivation with application sequence u and $G \in SEM(I)$ such that $s \in d^*(s_0, u)$ for some $s \in S$. Then there is a configuration $(1 \cdot [G] \overset{*}{\underset{P}{\Longrightarrow}} 1 \cdot [G'], G', s)$. This is proved by induction on the length of $G \overset{*}{\underset{P}{\Longrightarrow}} G'$.

Induction base: $G \overset{0}{\underset{P}{\Longrightarrow}} G'$ implies $G' = G$ such that $(1 \cdot [G] \overset{0}{\underset{P}{\Longrightarrow}} 1 \cdot [G] = 1 \cdot [G'], G', s_0)$ proves the statement for this case.

Induction step: $G \overset{k+1}{\underset{P}{\Longrightarrow}} G'$ with application sequence u decomposes into

$$G \overset{k}{\underset{P}{\Longrightarrow}} G'' \underset{r}{\Longrightarrow} G'$$

with $u = u'r$ where u' is the application sequence of $G \overset{k}{\underset{P}{\Longrightarrow}} G''$. As $s \in d^*(s_0, u'r)$ for some $s \in S$ there must be a state $s' \in d^*(s_0, u')$ such that $s \in d(s', r)$. Hence, by induction hypothesis, one gets $(1 \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} 1 \cdot [G''], G'', s')$, and by Construction 11, this yields the configuration $(1 \cdot [G] \overset{k}{\underset{P}{\Longrightarrow}} 1 \cdot [G''] \underset{P}{\Longrightarrow} 1 \cdot [G'], G', s)$.

6 Adleman's experiment

Adleman's famous and seminal experiment (Adleman 1994) can be mimicked by means of graph multiset transformation illustrating how the experiment inspired the presented approach.

Consider some simple directed graph $G = (V, E)$ with a finite set of nodes V and a set of edges $E \subseteq V \times V$ as well as two distinguished nodes *begin* and *end*.

Let $s(v) = l(v)r(v) \in \{A, T, C, G\}^*$ be a single-stranded DNA sequence for each $v \in V$ so that the lengths of all $l(v)$ and $r(v)$ are equal and the left and right sections identify the nodes, i.e., $v \neq v'$ implies $l(v) \neq l(v')$ and $r(v) \neq r(v')$. Moreover, let $X(v)$ and $\overline{X(v)}$ be two extra symbols for each $v \in V$ so that all of them are pairwise different.

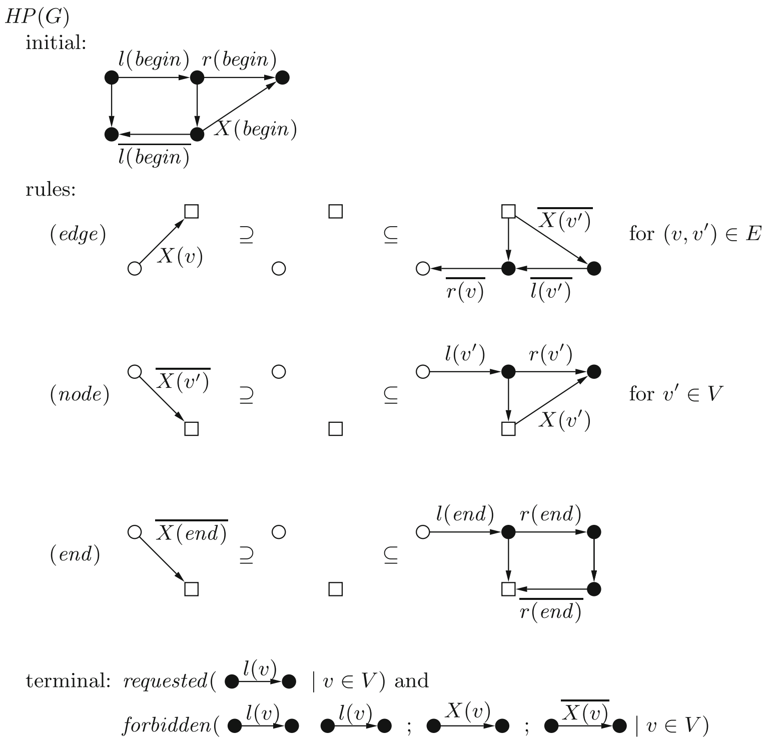
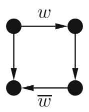


Fig. 10 The graph transformation unit *HP*(*G*)

Then G induces a graph transformation unit *HP*(G) that computes the Hamiltonian paths from *begin* to *end* in G by graph multiset transformation (see Fig. 10)

If $w \in \{A, T, C, G\}^*$, then $\overline{w}$ denotes the Watson–Crick complement. A graph of the form



represents the double strand $\left[\begin{array}{c} w \\ \overline{w} \end{array}\right]$. Starting with the initial graph, the rules of type *edge* and type *node* can be applied alternatively provided that there is a proper edge in E in the case of *edge*-rules. Every graph derived in this way, contains exactly one edge with an extra symbol which guides the choice of applicable rules. If this extra symbol is $X(end)$, then the *stop*-rule may be applied. A derivation of this kind is depicted in Fig. 11 .

The *edge*-rules are applicable if $(begin, v_1), (v_i, v_{i+1}), (v_n, end) \in E$ for $i = 1, \ldots, n-1$ meaning that the derived graph represents the path $begin v_1 \ldots v_n end$ in G. This is a Hamiltonian path if the label $l(v)$ occurs exactly once in the derived graph for each $v \in V$. In other words, the derived graph represents a Hamiltonian path from *begin* to *end* in G if and only if it is terminal.

Consequently, starting with a large number of initial graphs, one can detect Hamiltonian paths in G with high probability employing graph multiset transformation.

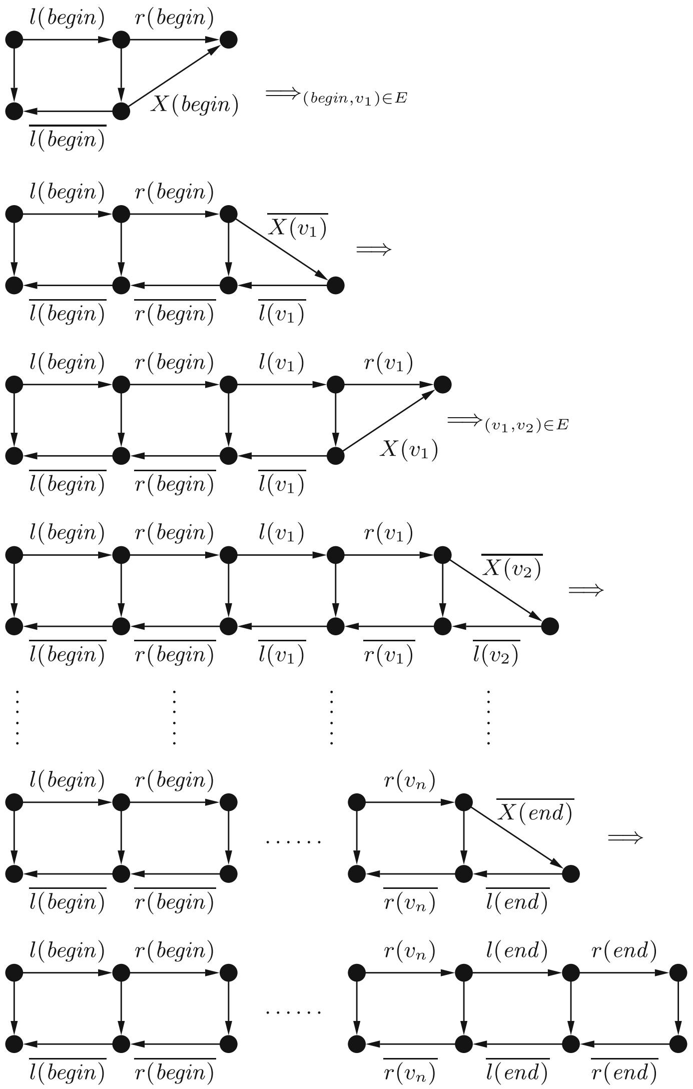
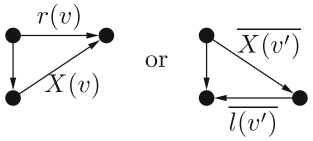


Fig. 11 A derivation of *HP*(*G*)

Each graph in Fig. 11 except the last graph contains a single edge labeled with an extra symbol which forms a triangle with an overhanging sticky end:



Each rule application removes the diagonal edge and completes the sticky end into a double strand according to the Watson–Crick-complementarity. The resulting graph corresponds to the double-stranded DNA molecule

$$\begin{bmatrix} \frac{s(begin)}{s(begin)} & \frac{s(v_1)}{s(v_1)} & \cdots & \frac{s(v_n)}{s(v_n)} & \frac{s(end)}{s(end)} \end{bmatrix}$$

In his experiment, Adleman created very similar molecules out of small pieces that correspond to right-hand sides of the rules above. Then he selected those molecules that contained the strands $s(v)$ for each $v \in V$ and had the proper lengths so that no $s(v)$ could occur twice. Therefore, the selection of the molecules that represent Hamiltonian paths in the experiment is quite similar to the termination condition above.

7 Exhaustive computations

A polynomial graph transformation unit $tu = (I, P, C, T)$ solves a decision problem by means of graph multiset transformation in a polynomial number of steps with a high probability if the multiplicity of the initial graph is large. It does not provide an exact solution because there is no guarantee that an existing permitted derivation $G \overset{*}{\Longrightarrow} \overline{G}$ with $G \in SEM(I)$ and $\overline{G} \in SEM(T)$ belongs to the derivations underlying a computation $n \cdot [G] \overset{*}{\Longrightarrow} \overline{M}$. This may be seen as a drawback which can be resolved by means of exhaustive computations that cover all derivations and all their prefixes up to a given length.

Definition 13 A graph multiset derivation $n \cdot [G] \overset{k}{\Longrightarrow} \overline{M}$ for some $k, n \in \mathbb{N}$ is *exhaustive* if each derivation $G \overset{l}{\Longrightarrow} \hat{G}$ with $l \leq k$ is an initial section of an underlying derivation, meaning that there is a derivation $\hat{G} \overset{*}{\Longrightarrow} \overline{G}$ with $G \overset{l}{\Longrightarrow} \hat{G} \overset{*}{\Longrightarrow} \overline{G} \in MoD(n \cdot [G] \overset{k}{\Longrightarrow} \overline{M})$.

As the following construction shows, exhaustive derivations exist for every length, but the initial multiplicity must be chosen large enough to cover all derivations up to the given length. In the construction, the multiplicity grows exponentially with the length if the rule application is not deterministic. If the underlying transformation unit is terminating, then the exhaustive derivation of the length of the termination bound (which can be prolonged by empty steps only) covers all derivations starting in the given initial graph including all successful ones. This is stated in Observation 14 and means that exhaustive derivations compute the corresponding decision problem exactly and not only with high probability.

Construction 14 Exhaustive derivations can be constructed inductively.

Induction base: $[G] \overset{0}{\Longrightarrow} [G]$ is an exhaustive derivation of length 0.
Induction step: Let $n \cdot [G] \overset{k}{\Longrightarrow} \overline{M}$ be an exhaustive derivation of length k which exists by induction hypothesis. Let max be the maximum number of direct derivations starting in some $\overline{G} \in car(\overline{M})$. Let $\max \cdot n \cdot [G] \overset{k}{\Longrightarrow} \max \cdot \overline{M}$ be obtained from $n \cdot [G] \overset{k}{\Longrightarrow} \overline{M}$ by copying every rule application max times. Then there are max copies of $\overline{G}$ in $\max \cdot \overline{M}$ for each $\overline{G} \in car(\overline{M})$ so that all direct derivations starting in $\overline{G}$ can be constructed. This defines an exhaustive derivation $\max \cdot n \cdot [G] \overset{k}{\Longrightarrow} \max \cdot \overline{M} \Longrightarrow \hat{M}$ of length $k + 1$.

Observation 15 Let $tu = (I, P, C, T)$ be a terminating transformation unit with termination bound $b: SEM(I) \to \mathbb{N}$ and $SEM(T) \subseteq reduced(P)$. Let $n \cdot [G] \underset{P}{\overset{b(G)}{\Longrightarrow}} \overline{M}$ for some

$n \in \mathbb{N}$ be an exhaustive graph multiset derivation with $car(\overline{M}) \subseteq reduced(P)$. Let $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G}$ with $\overline{G} \in SEM(T)$ be permitted by C. Then $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G} \in MoD(n \cdot [G] \overset{b(G)}{\underset{P}{\Longrightarrow}} \overline{M})$.

Proof Consider $G \overset{l}{\underset{P}{\Longrightarrow}} \overline{G}$ such that $\overline{G} \in SEM(T)$ and $G \overset{l}{\Longrightarrow} \overline{G}$ is permitted by C. Then $l \leq b(G)$ and by definition, there is a derivation $\overline{G} \overset{*}{\underset{P}{\Longrightarrow}} G'$ such that $G \overset{l}{\underset{P}{\Longrightarrow}} \overline{G} \overset{*}{\underset{P}{\Longrightarrow}} G' \in MoD(n \cdot [G] \overset{b(G)}{\underset{P}{\Longrightarrow}} \overline{M})$. By assumption one knows that $\overline{G} \in SEM(T) \subseteq reduced(P)$ so that $(\overline{G} \overset{*}{\underset{P}{\Longrightarrow}} G') = (\overline{G} \overset{0}{\underset{P}{\Longrightarrow}} G')$. Therefore, $G \overset{*}{\underset{P}{\Longrightarrow}} \overline{G} \in MoD(n \cdot [G] \overset{b(G)}{\underset{P}{\Longrightarrow}} \overline{M})$. □

The crucial part of Construction 14 is the copying of graphs which would be awfully inefficient if done in a sequential way. But if one assumes to have some copying mechanism for multisets of graphs that works like the polymerase chain reaction for DNA molecules, then one could duplicate a multiset of graphs in linear time and therefore multiply efficiently. A copying of this kind could be achieved by a DNA simulation of graph multiset transformation, for instance (cf. Point 6 of the conclusion).

Example 8 Figure 6 in Example 5 of Sect. 5 represents the full derivation process starting in G_0 that obeys the control condition *start*; *run**. The derivation would become exhaustive if all possibilities would be added to apply the rule *start* more that once. The derivation length is bounded by the number of nodes of the initial graph, and the reduced graphs contain a terminal graph if and only if the initial graph contains a Hamiltonian path. In other words, the exhaustive derivations of maximum lengths solve the Hamiltonian path problem in a linear number of steps.

8 Conclusion

In this paper, we have proposed graph multiset transformation as a novel framework for the modeling and computation of graph algorithms and decision problems on graphs in particular. The basic idea is to apply rules to various graphs in a multiset simultaneously in a single computational step. In particular, *NP*-problems can be solved polynomially by graph multiset transformation employing exhaustive derivations. A result like this is typical for and should be expected of a computational model with massive parallelism.

We are convinced that future investigations will emphasize the significance of this approach.

1. Graph multiset transformation may be compared with other types of parallelism within and beyond graph transformation.
2. In particular, one may relate graph multiset transformation with the underlying graph transformation because parallelism is provided in this approach in an elegant way by considering disjoint unions of rules as parallel rules. As they are also ordinary rules, they can be applied in exactly the same way as introduced in Sect. 2 Therefore, a derivation step of a multiset of graphs can be simulated by taking the disjoint union of all graphs in the multiset as host graph and applying the parallel rule of all applied rules to it. But doing this, one is faced with two problems: All known matching algorithms are exponential if the sizes of left-hand sides are not bounded so that one looses the direct correspondence to the class *NP*. Moreover, the relation is not easily conversed because a host graph does not know about its component graphs and a

parallel rule application does not automatically respect that at most one atomic rule is applied to a component graph. One would need some extra information about the component graphs that form the multiset. It may be meaningful to work this out.
3. In the introduction, we refer to genetic algorithms as one of our sources of inspiration for graph multiset transformation. A genetic algorithm transforms “populations of individuals” by means of operations like “mutation” and “crossover”. While mutation relates quite nicely to rule application in our approach, there is not yet a proper counterpart of crossover which recombines two individuals (and does not only change one). But this could be achieved by allowing rules that cut graphs into parts on one hand and merge parts together on the other hand. Such rule applications on graph multisets seems to be an interesting topic of future research. It would also yield possibilities to cover further DNA operations such as splicing.
4. Graph multiset transformation may be used like genetic algorithms as a heuristic approach. This would mean to start with a comparatively small multiplicity of initial graphs and to employ more sophisticated control conditions to improve the chances of successful computations.
5. The example of the Hamiltonian path problem indicates that simple graph transformation units and their evaluation by graph multiset transformation provides a quite natural way to model graph problems and their solutions. Further case studies can strengthen this view.
6. As pointed out in the Introduction, graph multiset transformation is inspired by Adleman’s experiment, in which he solved the Hamiltonian path problem by means of DNA computing in the proper sense using DNA molecules and their reaction with each other. Similarly, it may be possible to translate graph multiset transformation into DNA computing and implement it by a massively parallel machinery in this way.
7. Because of the close relation to genetic algorithms and DNA computing, graph multiset transformation is potentially applicable wherever these both are useful.

Acknowledgments The authors would like to acknowledge that their research ispartially supported by the Collaborative Research Centre 637(Autonomous Cooperating Logistic Processes: A Paradigm Shift and ItsLimitations) funded by the German Research Foundation (DFG). The authors are also grateful to the anonymous reviewers for their valuable comments.

Appendix

This appendix recalls the notions and notations of multisets used in the paper.

1. Let X be a set. Then a multiset (over X) is a mapping $M : X \to \mathbb{N}$, where $M(x)$ is the *multiplicity* of x in M.
2. The *carrier* of M contains all elements of X with positive multiplicity, i.e.

$$car(M) = \{x \in X \mid M(x) > 0\}.$$

3. A multiset is *finite* if its carrier is a finite set.
4. Let M and M' be multisets. Then M' is a *sub-multiset* of M, denoted by $M' \leq M$, if $M'(x) \leq M(x)$ for all $x \in X$.
5. Let M and M' be multisets. Then the *sum* (*difference*) of M and M' is the multiset defined by

$$(M \pm M')(x) = M(x) \pm M'(x) \text{ for all x} \in \text{X}.$$

Here $+$ and $-$ are the usual sum and difference of non-negative integers with $m - n = 0$ if $m \leq n$ in particular.

6. Using the sum of multisets, the multiplication of multisets with non-negative numbers can be defined inductively for all multisets M by
   (i) $0 \cdot M = \mathbf{0}$ and
   (ii) $(k + 1) \cdot M = k \cdot M + M$ for all $k \in \mathbb{N}$
   where the multiset $\mathbf{0}$ is the multiset with the constant multiplicity 0, i.e. $\mathbf{0}(x) = 0$ for all $x \in X$.
7. Each sequence $w \in X^*$ induces a multiset $[w]$ by counting the number of occurrences of each x in w, i.e., for all $x, y \in X$ and $w \in X^*$,
   - $[\lambda](x) = 0$
   - $[yw](x) = \mathit{if}\ x = y\ \mathit{then}\ [w](x) + 1\ \mathit{else}\ [w](x)$.
8. Let M be a finite multiset. Then the set of all sequences w with $[w] = M$ is denoted by *Perm*(M). An element of *Perm*(M) is called a *sequential representation* of M. Note that *Perm*(M) contains all permutations of w if $[w] = M$.
9. The set of multisets over X as well as the set of finite multisets over X give rise to a commutative monoid with the multiset $\mathbf{0}$ as null and the sum as inner composition. Moreover, the set of finite sultisets over X is generated by the singletons $[x]$ for all $x \in X$ so that the finite multisets are characterized as the free commutative monoid over X.

References

Adleman LM (1994) Molecular computation of solutions to combinatorial problems. Science 266: 1021–1024

Ehrig H, Ehrig K, Taentzer G, de Lara J, Varró D, Varró-Gyapai S (2005) Termination criteria for model transformation. In: Cerioli M (ed) Proceedings of fundamental approaches to software engineering (FASE 2005). Lecture notes in Computer science, vol 3442. Springer, Berlin, pp 49–63

Fogel DB (2006) Evolutionary computation: toward a new philosophy of machine intelligence, 3rd edn. IEEE Press, Piscataway, NJ

Godard E, Métivier Y, Mosbah M, Sellami A (2002) Termination detection of distributed algorithms by graph relabelling systems. In: Corradini A, Ehrig H, Kreowski H-J, Rozenberg G (eds) Proceedings of the first international conference on graph transformation (ICGT '02). Lecture notes in Computer science, vol 2505. Springer, Berlin, pp 106–119

Goldberg DE (2002) The design of innovation: lessons from and for competent genetic algorithms. Addison-Wesley, Reading, MA

Habel A, Plump D (2001) Computational completeness of programming languages based on graph transformation. In: Honsell F, Miculan M (eds) Proceedings of foundations of software science and computation structures (FOSSACS 2001). Lecture Notes in Computer science, vol 2030. Springer, Berlin, pp 230–245

Holland JM (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI

Kreowski H-J (2002) A sight-seeing tour of the computational landscape of graph transformation. In: Brauer W, Ehrig H, Karhumäki J, Salomaa A (eds) Formal and natural computing. Essays Dedicated to Grzegorz Rozenberg. Lecture notes in Computer science, vol 2300. Springer, Berlin, pp 119–137

Kreowski H-J, Kuske S (1999a) Graph transformation units and modules. In: Ehrig H, Engels G, Kreowski H-J, Rozenberg G (eds) Handbook of graph grammars and computing by graph transformation, vol 2: applications, languages and tools. World Scientific, Singapore, pp 607–638

Kreowski H-J, Kuske S (1999b) Graph transformation units with interleaving semantics. Form Asp Comput 11(6):690–723

Kreowski H-J, Kuske S (2008) Graph multiset transformation as a framework for massively parallel computation. In: Proceedings of 4th international conference on graph transformations (ICGT 2008). Lecture notes in Computer science, vol 5214. Springer, Heidelberg, pp 351–365

Kreowski H-J, Kuske S, Schürr A (1997) Nested graph transformation units. Int J Softw Eng Knowl Eng 7 (4):479–502

Kuske S (2000) More about control conditions for transformation units. In: Ehrig H, Engels G, Kreowski H-J, Rozenberg G (eds) Proceedings of theory and application of graph transformations. Lecture notes in Computer science, vol 1764. Springer, Berlin, pp 323–337

Kuske S (2000) Transformation units—a structuring principle for graph transformation systems. PhD thesis, University of Bremen

Păun G, Rozenberg G, Salomaa A (1998) DNA computing—new computing paradigms. Springer, Berlin

Plump D (1998) Termination of graph rewriting is undecidable. Fundam Inf 33(2):201–209