# Graph Multiset Transformation as a Framework for Massively Parallel Computation*

Hans-Jörg Kreowski and Sabine Kuske

University of Bremen, Department of Computer Science
P.O.Box 33 04 40, 28334 Bremen, Germany
{kreo,kuske}@informatik.uni-bremen.de

**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.

## 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., [1,2,3,4,5]). 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 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 authors would like to acknowledge that their research is partially supported by the Collaborative Research Centre 637 (Autonomous Cooperating Logistic Processes: A Paradigm Shift and Its Limitations) funded by the German Research Foundation (DFG).

H. Ehrig et al. (Eds.): ICGT 2008, LNCS 5214, pp. 351–365, 2008.

The paper is organized in the following way. The next section provides the prelimininaries concerning graphs and rule-based graph transformation. In Section 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 Section 5 and 6. As a consequence, *NP*-problems can be solved in a polynomial number of computational steps. 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. The proofs are omitted because of the limited space. It may be noted that the basic ideas of graph multiset transformations have been sketched in [6] in a draft way.

## 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 $\Sigma$ be a set of labels. A *graph* over $\Sigma$ 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 $\Sigma$ 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., $g_V(s_G(e)) = s_H(g_E(e))$, $g_V(t_G(e)) = t_H(g_E(e))$, 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 Figure 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 $\circledcirc$ to represent a *begin*-node which is a node with a loop labeled with *begin*. Analogously, $\circ$ represents an *end*-node. If one starts in the *begin*-node and follows the $p$-labeled

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

edges, one reaches the *end*-node in the graphs $H_1$, $H_2$, $H_3$, and $H_4$. In each case, the sequence of $p$-edges defines a Hamiltonian path of $G_0$, where the intermediate nodes have no loops.

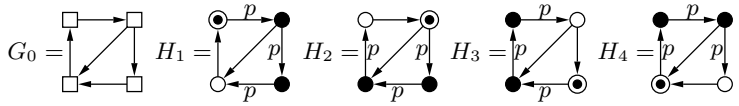



**Fig. 1.** $G_0$ with all its Hamiltonian paths

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 Figure 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 $G_{123}$.

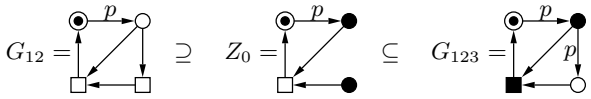



**Fig. 2.** Two graphs with a common subgraph

There are two graph morphisms from the graph $L_{run} =$  into $G_{12}$ which map to the subgraphs of the same form.

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 $G = (V, E, s, t, l)$ consists of the following three steps.

1. A match $g(L)$ of $L$ in $G$ is chosen subject to the following conditions.
   - *contact 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 $d\colon K \to Z$ be the restriction of $g$ to $K$ and $Z$, then the pushout of $d$ and the inclusion of $K$ into $R$ yields the resulting graph $H$ and graph morphisms $h\colon R \to H$ and $i\colon Z \to H$. Without loss of generality, one can assume that $i$ is the inclusion of $Z$ into $H$ and that $h$ is the identity on $R - K$. This provides an explicit construction of $H$ because $Z \cup h(R) = H$ and $Z \cap h(R) = d(K) = h(K)$.

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

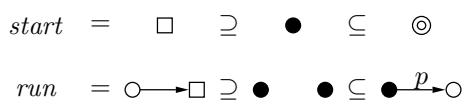
The contact condition guarantees that the removal of $L - K$ from $G$ yields a graph and that the restriction $d$ of $g$ to $K$ and $Z$ is a graph morphism. The identification condition makes sure that $G$ together with $g$ and the inclusion of $Z$ into $G$ is a pushout of $d$ and the inclusion of $K$ into $L$. Altogether, a direct derivation is given by a double pushout (cf. Figure 3).

$$\begin{array}{ccccc} L & \supseteq & K & \subseteq & R \\ \downarrow g & & \downarrow d & & \downarrow h \\ G & \supseteq & Z & \subseteq & H \end{array}$$

**Fig. 3.** Diagram of a double pushout

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 $r_1 \cdots r_n$ is the *application sequence* of the derivation $d$.

*Example 2.* Consider the following two rules




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 $G_0$ and apply the rule *start* only once in the beginning (while the rule *run* is applied repeatedly afterwards). At first, the rule *start* can be applied to $G_0$ in four ways deriving the four graphs in the second column from the left of Figure 4. The graphs in the right-most column of Figure 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 dificult 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 derived graph is Hamiltonian if and only if it 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 contact and the identification condition hold, and the construction of the directly derived graph

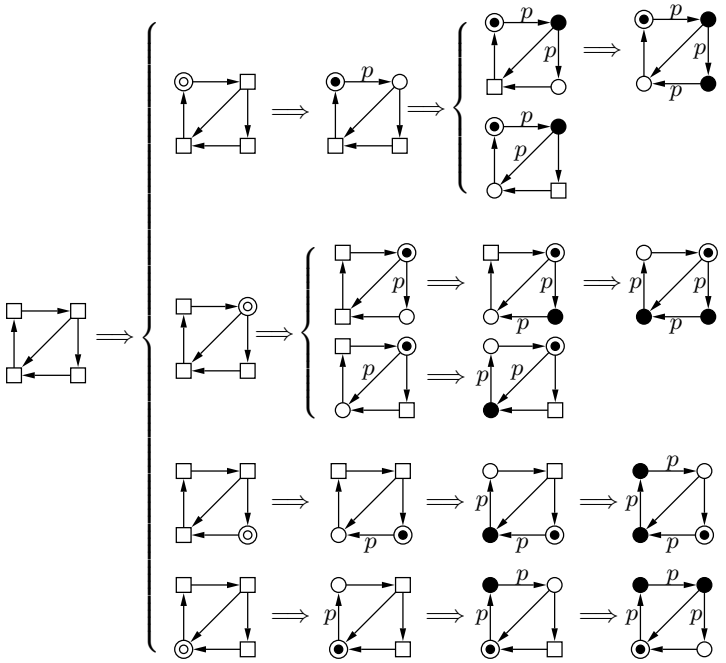


**Fig. 4.** The derivations starting in $G_0$ with one application of *start*

is linear in the size of $G$. Therefore, it needs polynomial time 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, terminal graphs to stop it, and some control conditions to regulate it. Initial and terminal graphs may be specified by graph class expressions. 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 $SEM(\Delta) = \mathcal{G}_\Delta \subseteq \mathcal{G}_\Sigma$. Forbidden and reduced structures are also frequently used. Let $F$ be

a graph, then $SEM(forbidden(F))$ contains all graphs $G$ such that there is no graph morphism $f\colon F \to G$. 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 $s \in L(C)$.

### 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 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:

*HP*
initial: *unlabeled graphs*
rules: *start, run*
control: *start ; run**
terminal: *forbidden*($\square$)

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.

### 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 $SEM(\textit{unlabeled graphs})$, $SEM(forbidden(F))$ or $SEM(reduced(P))$ is polynomial in the size of $G$. If $G \overset{k}{\Rightarrow} H$ 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 $SEM(forbidden(F))$ 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 [7,8,9] 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 [10,11]. Habel and Plump [12] have recently 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 polynom 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 $G \overset{n}{\underset{P}{\Rightarrow}} G'$. The function $b\colon SEM(I) \to \mathbb{N}$ 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 $val\colon \mathcal{G}_\Sigma \to \mathbb{N}$ be a valuation function with $val(G') > val(G'')$ for each direct derivation $G' \underset{P}{\Rightarrow} G''$. Then $tu$ is terminating.

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

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

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

   This is denoted by $COMP(tu) = D$.
3. *tu* is called *polynomial* if there is a polynom $p$ such that, for all $G \in SEM(I)$, $b(G) \leq p(size(G))$.
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 $G \overset{*}{\underset{P}{\Rightarrow}} G'$ 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 $G \overset{*}{\underset{P}{\Rightarrow}} G'$ with $G' \in SEM(reduced(P))$, but $G' \notin SEM(T)$, and also a permitted derivation $G \overset{*}{\underset{P}{\Rightarrow}} \overline{G}$ 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 $D\colon SEM(I) \to \{\text{TRUE}, \text{FALSE}\}$ 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.** $NP_{GT} = NP$.

*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 [13], Godard, Métivier, Mosbah, and Sellami [14], and by Ehrig, Ehrig, de Lara, Taentzer, Varró, and Varró-Gyapay [15].

## 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. 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. Observation 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.[2]

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

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

A sequence $M_0 \underset{P}{\Rightarrow} M_1 \underset{P}{\Rightarrow} \cdots \underset{P}{\Rightarrow} M_n$ 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 $M \overset{n}{\underset{P}{\Rightarrow}} \overline{M}$ or $M \overset{*}{\underset{P}{\Rightarrow}} \overline{M}$. The subscript $P$ may be omitted if it is clear from the context.

*Remark.* 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 $G_1 \cdots G_n$ corresponds to the respective permutation of $G'_1 \cdots G'_n$ and the multisets of sequences are invariant with respect to permutation.

It is easy 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.

**Observation 6.** 1. Let $M \overset{k}{\underset{P}{\Rightarrow}} \overline{M}$ be a graph multiset derivation of length $k$ and $G_1 \cdots G_n \in Perm(M)$ a sequential representation of $M$. Then there is a sequential representation $\overline{G_1} \cdots \overline{G_n} \in Perm(\overline{M})$ such that $G_i \overset{k_i}{\underset{P}{\Rightarrow}} \overline{G_i}$ for all $i = 1, \ldots, n$ with $k_i \leq k$.

2. Let $G_1 \cdots G_n, \overline{G_1} \cdots \overline{G_n} \in \mathcal{G}^*_\Sigma$ be sequences of graphs with $G_i \overset{k_i}{\underset{P}{\Rightarrow}} \overline{G_i}$ for all $i = 1, \ldots, n$. Then there is a graph multiset derivation $[G_1 \cdots G_n] \overset{k}{\underset{P}{\Rightarrow}} [\overline{G_1} \cdots \overline{G_n}]$ with $k = max\{k_i \mid i = 1, \ldots, n\}$.

Observation 6 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. 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

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

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 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 7.** Let $tu = (I, P, C, T)$ be a terminating transformation unit. Let $D\colon SEM(I) \to \{\text{TRUE}, \text{FALSE}\}$ 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 graph multiset derivation $n \cdot [G] \overset{*}{\underset{P}{\Rightarrow}} \overline{M}$ for some $n \in \mathbb{N}$ so that some underlying derivation $G \overset{*}{\underset{P}{\Rightarrow}} \overline{G}$ 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. $P_{GMST}$ 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 $G$. If the multiplicity $n$ of $G$ is chosen larger than this bound and the derivation $n \cdot [G] \overset{*}{\underset{P}{\Rightarrow}} \overline{M}$ is long, then the probability is high that most permitted derivations starting in $G$ are underlying $n \cdot [G] \overset{*}{\underset{P}{\Rightarrow}} \overline{M}$. 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 $P_{GMST}$.

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 $P_{GMST}$ coincide. Unfortunately, this is not a solution of the $P{=}NP$-problem because the class $P_{GMST}$ relies on massive parallelism.

**Theorem 8.** $NP_{GT} = P_{GMST}$.

*Example 5.* Based on the unit *HP* in Example 3, Figure 5 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 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.

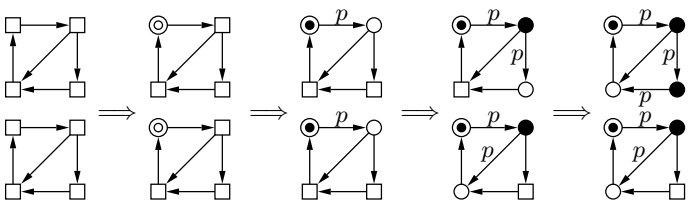



**Fig. 5.** A graph multiset derivation

Moreover, we assume that terminal graphs are reduced. Therefore, the check for terminality can be postponed until a derivation cannot be prolonged.

**Construction 9.** 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 $(M \overset{*}{\Rightarrow} \overline{M}, W, w)$ 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 $(n \cdot [G] \overset{0}{\Rightarrow} n \cdot [G], G^n, s_0^n)$ as start configuration.

*Induction hypothesis:* Assume that a configuration

$$(n \cdot [G] \overset{k}{\Rightarrow} \hat{M}, \hat{G}_1 \dots \hat{G}_n, s_1 \dots s_n)$$

is already constructed so that the following holds for $i = 1, \dots, n$:

$$s_i \in d^*(s_0, u_i)$$

where $u_i$ is the application sequence of the underlying derivation $G \overset{k_i}{\Rightarrow} \hat{G}_i$.

*Induction step:* If possible, then choose for $i = 1, \dots, n$, $\hat{G}_i \underset{r}{\Rightarrow} \overline{G}_i$ 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 $[\hat{G}_1 \dots \hat{G}_n] \Rightarrow [\overline{G}_i \dots \overline{G}_n]$ is a direct derivation giving rise to the follow-up configuration

$$(n \cdot [G] \overset{k}{\Rightarrow} \hat{M} \Rightarrow [\overline{G}_1 \dots \overline{G}_n], \overline{G}_1 \dots \overline{G}_n, \overline{s}_1 \dots \overline{s}_n).$$

The construction can be terminated if a configuration

$$(n \cdot [G] \overset{l}{\Rightarrow} \overline{M}, \overline{G}_i \dots \overline{G}_n, \overline{s}_1 \dots \overline{s}_n)$$

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

**Observation 10.** 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 $(n \cdot [G] \stackrel{k}{\Rightarrow} \overline{M}, \overline{G}_1 \dots \overline{G}_n, s_i \dots s_n)$ be a configuration constructed above. Let, for $i = 1, \dots, n$, $u_i$ be the application sequence of the underlying derivation $G \stackrel{k_i}{\Rightarrow} \overline{G}_i$. Then $s_i \in d^*(s_0, u_i)$.
2. Let $n \cdot [G] \stackrel{*}{\Rightarrow} \overline{M}, \overline{G}_1 \dots \overline{G}_n, s_i \dots s_n)$ be a terminated configuration for some $G \in SEM(I)$ with $\overline{G}_i \in SEM(T)$ and $s_i \in F$ for some $i = 1, \dots, n$. Then $D(G) = \text{TRUE}$.
3. If $D(G) = \text{TRUE}$, then there is a terminated configuration of the form $(1 \cdot [G] \stackrel{*}{\Rightarrow} (1 \cdot [\overline{G}], \overline{G}, s)$ for some $\overline{G} \in \mathcal{G}_\Sigma$ and $s \in F$.

## 6 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 a permitted derivation $G \stackrel{*}{\Rightarrow} \overline{G}$ with $G \in SEM(I)$ and $\overline{G} \in SEM(T)$ belongs to the derivations underlying a computation $M_G \stackrel{*}{\Rightarrow} \overline{M}$. This may be seen as a drawback. But the problem can be resolved by means of exhaustive computations that cover all derivations and all their prefixes up to a given length.

**Definition 11.** A computation $n \cdot [G] \stackrel{k}{\Rightarrow} \overline{M}$ for some $n \in \mathbb{N}$ is *exhaustive* if each derivation $G \stackrel{l}{\Rightarrow} \hat{G}$ with $l \leq k$ is an initial section of an underlying derivation, meaning that there is a derivation $\hat{G} \stackrel{*}{\Rightarrow} \overline{G}$ with $G \stackrel{l}{\Rightarrow} \hat{G} \stackrel{*}{\Rightarrow} \overline{G} \in der(n \cdot [G] \stackrel{*}{\Rightarrow} \overline{M})$.

**Theorem 12.** Let $tu = (I, P, C, T)$ be a transformation unit with $SEM(T) \subseteq reduced(P)$. Let $n \cdot [G] \stackrel{*}{\Rightarrow} \overline{M}$ for some $n \in \mathbb{N}$ be exhaustive with $car(\overline{M}) \subseteq reduced(P)$. Let $G \stackrel{*}{\Rightarrow} \overline{G}$ with $\overline{G} \in SEM(T)$ be permitted by $C$. Then $G \stackrel{*}{\Rightarrow} \overline{G} \in der(n \cdot [G] \stackrel{*}{\Rightarrow} \overline{M})$.

*Remark.* Exhaustive computations can be constructed inductively.

*Induction base:* $[G] \stackrel{0}{\Rightarrow} [G]$ is an exhaustive computation of length 0.

*Induction step:* Let $n \cdot [G] \stackrel{k}{\Rightarrow} \overline{M}$ be an exhaustive computation 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 $max \cdot n \cdot [G] \stackrel{k}{\Rightarrow} max \cdot \overline{M}$ be obtained from $n \cdot [G] \stackrel{k}{\Rightarrow} \overline{M}$ 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 computation $max \cdot n \cdot [G] \stackrel{k}{\Rightarrow} max \cdot \overline{M} \Rightarrow \hat{M}$ of length $k + 1$.

*Example 6.* Figure 4 in Example 2 represents the full derivation process starting in $G_0$ that obeys the control condition $start; run^*$. It can be considered as an exhaustive graph multiset derivation where the columns are the multisets and each column from right to left is filled with enough copies of the present graphs that all alternative rule applications can be applied simultaneously. Altogether,

six copies of $G_0$ are needed to derive all reduced graphs. 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.

## 7 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 prove the significance of this approach.

1. Graph multiset transformation may be compared with other types of parallelism within and beyond graph transformation.
2. 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.
3. 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.
4. 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.
5. Because of the close relation to genetic algorithms and DNA computing, graph multiset transformation is potentially applicable wherever these both are useful.

*Acknowledgement.* We are grateful to the anonymous referees for their valuable comments.

## References

1. Holland, J.M.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
2. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Addison-Wesley, Reading (2002)

3. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 3rd edn. IEEE Press, Piscataway (2006)
4. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024 (1994)
5. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing — New Computing Paradigms. Springer, Heidelberg (1998)
6. Kreowski, H.J.: 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. LNCS, vol. 2300, pp. 119–137. Springer, Heidelberg (2002)
7. Kreowski, H.J., Kuske, S., Schürr, A.: Nested graph transformation units. International Journal on Software Engineering and Knowledge Engineering 7(4), 479–502 (1997)
8. Kreowski, H.J., Kuske, S.: 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, Applications, Languages and Tools, vol. 2, pp. 607–638. World Scientific, Singapore (1999)
9. Kreowski, H.J., Kuske, S.: Graph transformation units with interleaving semantics. Formal Aspects of Computing 11(6), 690–723 (1999)
10. Kuske, S.: More about control conditions for transformation units. In: Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. (eds.) TAGT 1998. LNCS, vol. 1764, pp. 323–337. Springer, Heidelberg (2000)
11. Kuske, S.: Transformation Units—A Structuring Principle for Graph Transformation Systems. PhD thesis, University of Bremen (2000)
12. Habel, A., Plump, D.: Computational completeness of programming languages based on graph transformation. In: Honsell, F., Miculan, M. (eds.) FOSSACS 2001. LNCS, vol. 2030, pp. 230–245. Springer, Heidelberg (2001)
13. Plump, D.: Termination of graph rewriting is undecidable. Fundamenta Informaticae 33(2), 201–209 (1998)
14. Godard, E., Métivier, Y., Mosbah, M., Sellami, A.: Termination detection of distributed algorithms by graph relabelling systems. In: Corradini, A., Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2002. LNCS, vol. 2505, pp. 106–119. Springer, Heidelberg (2002)
15. Ehrig, H., Ehrig, K., Taentzer, G., de Lara, J., Varró, D., Varró-Gyapai, S.: Termination criteria for model transformation. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 49–63. Springer, Heidelberg (2005)

## 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 $M\colon X \to \mathbb{N}$, 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

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

   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 $w \in X^*$,
   - $[\lambda](x) = 0$
   - $[yw](x) =$ *if* $x = y$ *then* $[w](x) + 1$ *else* $[w](x)$.
8. Let $M$ be a 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 sigletons $[x]$ for all $x \in X$ so that the finite multisets are characterized as the free commutative monoid over $X$.