Modelling Reconfigurability using Graph Grammars and Petri Nets

M. Pietkiewicz-Koutny
Modelling Reconfigurability using Graph Grammars and Petri Nets

M. Pietkiewicz-Koutny

Abstract

We here consider non-sequential systems whose internal connectivity may change as the result of the execution of certain actions. To model such systems we first introduce a graph-grammar model in which the notion of reconfigurability can be precisely formulated. We then provide a translation from this graph-grammar model into the standard labelled Petri nets which preserves the behavioural properties of the systems. In this way we obtain a framework which allows the application of the analytical methods developed for ordinary Petri nets to deal with a class of systems with dynamically changing interconnection structure.

©1991 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
PIETKIEWICZ-KOUTNY, Marta

Modelling reconfigurability using graph grammars and Petri nets.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1991.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 356)

Abstract

We here consider non-sequential systems whose internal connectivity may change as the result of the execution of certain actions. To model such systems we first introduce a graph-grammar model in which the notion of reconfigurability can be precisely formulated. We then provide a translation from this graph-grammar model into the standard labelled Petri nets which preserves the behavioural properties of the systems. In this way we obtain a framework which allows the application of the analytical methods developed for ordinary Petri nets to deal with a class of systems with dynamically changing interconnection structure.

About the author

Mrs. Marta Pietkiewicz Koutny has been at the Computing Laboratory as a Research Associate from May, 1987 until February, 1988 and as a Demonstrator from March, 1988.

Suggested keywords

CONCURRENCY
DYNAMICALLY STRUCTURED SYSTEMS
GRAPH GRAMMARS
PETRI NETS

Suggested classmarks (primary classmark underlined)
Dewey (18th): 511.5 001.84404
U.D.C. 519.711 519.687
Modelling Reconfigurability using Graph Grammars and Petri Nets

Marta Pietkiewicz-Koutny
Computing Laboratory
The University
Newcastle upon Tyne NE1 7RU, U.K.

Abstract

We here consider non-sequential systems whose internal connectivity may change as the result of the execution of certain actions. To model such systems we first introduce a graph-grammar model in which the notion of reconfigurability can be precisely formulated. We then provide a translation from this graph-grammar model into the standard labelled Petri nets which preserves the behavioural properties of the systems. In this way we obtain a framework which allows the application of the analytical methods developed for ordinary Petri nets to deal with a class of systems with dynamically changing interconnection structure.

1. Introduction

In this paper we are concerned with formal modelling and analysis of non-sequential systems whose interconnection structure can change as a result of the execution of certain actions. More precisely, we aim at obtaining a class of Petri nets in which such systems could be represented and analysed. Ideally, we would like to define a class of Petri nets whose sets of places and transitions would remain the same during the net's evolution, and whose flow relation might change (be reconfigured) as a result of the firing of the transitions. In the standard net theory [Pet81,Rei85] the flow relation is constant; thus it is necessary to go beyond the standard model to properly specify what one understands as a reconfigurable net. There are two issues which we think should be taken into account in the development of a model of reconfigurable nets. Firstly, the model should be easy to use and the results of the analysis should allow a direct interpretation. Secondly, the model should be kept as close as possible to the standard P/T-net model. In this way one would hope to be able to apply the analytical methods which have been already developed for the standard Petri net model.

In this paper we present the first step toward the development of a model of reconfigurable nets which could meet both of the above criteria. We will use graph grammars [Ehr78, EK76] as the base model in which reconfiguration can be specified in a natural way. By using a graph rewriting technique which is based on those introduced in [Kre81] and [GJRT83] we will define the Reconfigurable Graph Grammar model (RGG) in which it will be possible to capture the intended behaviour of reconfigurable Petri net.
We envisage that the designer of a dynamically reconfigurable system would first specify the initial (static) structure of the system in terms of a suitable P/T-net. The net would then be automatically translated into an equivalent graph grammar, as described in section 3.1. After that, the transition which can carry out reconfiguration should be specified, in the form prescribed by the RGG model. For the resulting graph grammar a number of behavioural properties can be defined and verified using the existing tools of the graph grammars theory. Moreover, it turns out that it is possible to construct, for each RGG, a behaviourally equivalent labelled P/T-net without $\lambda$-transitions. Consequently, it will be possible to use the verification techniques developed for the ordinary Petri nets to analyse the reconfigurable system. Thus we will obtain two alternative methods of proving properties of reconfigurable systems: one using the tools of the graph grammars theory, the other using the tools of the standard Petri nets theory. We envisage that the choice between the two will depend on the kind of properties to be verified.

In this paper we do not introduce explicitly any new class of Petri nets. Our main aim is to clarify the notion of a reconfigurable Petri net, and we think that our RGG model does provides here a satisfactory solution. We also show an important technical result, namely we prove that each RGG can in principle be modelled by a labelled P/T-net. This establishes a link between the graph grammars and Petri nets in the context of reconfigurable systems. (Note that this also shows that approach is different from that introduced in [Val78] since there systems cannot in general be modelled by P/T-nets.) The precise formalisation of the notion of a reconfigurable Petri net will be introduced in another paper, here we only give an example of how such a Petri net might look like.

The paper is organised as follows. We first discuss two graph grammar models for standard P/T-nets, [Kre81] and [GJRT83], and identify those of their features which we think are useful in the modelling of reconfigurable nets. We then introduce the RGG graph grammar model and show how the standard P/T-net can be simulated by a suitable RGG. In the last section we show that it is possible to derive, for each RGG, an equivalent labelled Petri nets without $\lambda$-transitions.

2. Graph-Grammars and Marked Nets

In this section we discuss two approaches to modelling P/T-nets introduced by Kreowski [Kre81] and Genrich et. al [GJRT83] using an example net shown in Figure 2.1.

A net is a triple $N = (S,T;F)$, where $S$ and $T$ are two finite disjoint sets and $F \subseteq S \times T \cup T \times S$. We assume that $S \cup T \neq \emptyset$, and that for every $t \in T$ there are $x, y \in S$ such that $(t,x) \in F$ and $(y,t) \in F$. The elements of $S$ are called places and are represented by circles; the elements of $T$ are called transitions and are represented by boxes; $F$ is called the flow relation and is represented by arcs. For every $x \in$ 

![Figure 2.1](image.png)
$\cup T$ we denote $x^* = \{ y \mid (x,y) \in F \}$ and $^*x = \{ y \mid (y,x) \in F \}$. $x^*$ is called the post-set of $x$, and $^*x$ is called the pre-set of $x$.

A marking of $N$ is any mapping $M : S \rightarrow \{0,1,2,\ldots\}$. In the graphical representation, $M$ will be indicated by placing, for each place $p$, $M(p)$ tokens inside the circle representing $p$.

Taking as an example net $N = (\{p,q,r,s\}, \{a,b\}; \{(p,b),(p,a),(q,a),(a,s),(b,s),(b,r)\})$ and marking $M$ such that $M(p) = 2$, $M(q) = M(s) = 1$, $M(r) = 0$, we obtain a graphical representation as in Figure 2.1. A transition $t \in T$ is enabled at marking $M$ if $M(s) \geq 1$, for all $s \in ^*t$. A firing of an enabled transition leads to a new marking $M'$ such that for all $s \in S$, $M'(s) = M(s) - \text{card} \{s \cap \{t\}\} + \text{card} \{^*s \cap \{t\}\}$. We denote this by $M[t \rightarrow M']$. For every marking $M$, $[M]$ denotes the set of markings reachable from $M$ which is the smallest set containing $M$ and such that if $M' \in [M]$ and $M'' \in [M]'$, for some $t$, then $M'' \in [M]$.

A marked net (or P/T-net) is a quadruple $MN = (S,T;F,M_0)$ where $N = (S,T;F)$ is a net, and $M_0$ is a marking of $N$, called the initial marking.

We say that $MN$ is live if for all $M \in [M_0]$ and all $t \in T$ there are $M', M'' \in [M]$ such that $M'[t \rightarrow M'']$ and $MN$ is deadlock-free if for all $M \in [M_0]$ there is a transition enabled at $M$.

2.1 Kreowski’s Simulation of Marked Nets

Graph Grammars generalise the Chomsky’s grammars by replacing the underlying data structure of strings by graphs. A graph grammar $GG = (\Sigma,P,Z_0)$ essentially consists of a set of labels $\Sigma$, a set of productions $P$ and an initial graph over $\Sigma$, $Z_0$.

In [Kre81] each production is of the form $p = (L \Rightarrow R)$ where $L$ and $R$ are two directed labelled graphs. The application of production $p$ to graph $G$ is carried out in two steps:

1. $L$ is matched with an isomorphic subgraph of $G$.
2. $R$ replaces $L$ within $G$, i.e. $H = (G-L) + R$.

In carrying out the replacement, one needs to take care of the arcs joining the vertices from $G-L$ and $L$. This is done through an embedding rule for $R$ in $G-L$. For each production $p = (L \Rightarrow R)$ there is a common subgraph $K$ of $L$ and $R$, called gluing points, and step (2) of the above construction is only allowed if the following gluing condition holds: sources and targets of arcs of $G-L$ belong to $G-L$ or $K$. The gluing condition guarantees a proper embedding of $R$ into $G-L$. If it is satisfied we say that $p$ is applicable to $G$ and $H$ is derived from $G$ via $p$, denoted $G \Rightarrow_p H$. In [Kre81] a graph grammar is constructed in such a way that each transition firing is simulated by the application of the corresponding production. The basic idea of this approach is to replace the concept of the place being a circle holding a number of tokens inside it, by the concept of the place being a node to which a number of nodes, each representing a single token, is attached (see Figure 2.1.1).

![Figure 2.1.1](image-url)
Each transition $t \in T$ gives rise to a graph grammar production $p(t) = (L(t) \Rightarrow R(t))$, where $L(t)$ and $R(t)$ represent $t$ together with the immediate neighbourhood of $t$ and showing the local change of a marking defined by the firing rule for $t$. The construction of $p(t)$ is illustrated in Figure 2.1.2 for a transition $t$ with $\ast t = \{p_1, ..., p_m\}$ and $\ast t = \{q_1, ..., q_k\}$. Note that $p_1, ..., p_m, q_1, ..., q_k$ and the arcs joining them are the glueing points.

Finally, for net $N = (S, T; F)$ and marking $M$, a graph $GRAPH(N, M)$ is constructed by applying the transformation illustrated in Figure 2.1.1. Taking the net from Figure 2.1, we obtain $GRAPH(N, M)$ as in Figure 2.1.3(a) and the productions $p(a)$ and $p(b)$ as in Figure 2.1.3(b).

To ensure that the application of $p(t)$ to $GRAPH(N, M)$ indeed corresponds to the firing of $t$ in $N$, it is assumed that the elements of $S$ and $T$ are labels of the corresponding nodes in $GRAPH(N, M)$, $L(t)$
and \( R(t) \). That is, a marked net \( MN = (S,T,F,M_0) \) is represented by a graph-grammar
\[
GG_{Kre}(MN) = (S \cup T, \{ p(t) \mid t \in T \}, \text{GRAPH}((S,T,F), M_0)).
\]
In [Kre81] a number of properties of \( GG_{Kre}(MN) \) have been established which confirmed the soundness of the definition of \( GG_{Kre}(MN) \). In particular, liveness and deadlock-freeness of \( MN \) can be characterised in terms of the corresponding behavioural properties of \( GG_{Kre}(MN) \). The notions of conflict and concurrency in net \( MN \) turn out to be equivalent to dependency and independency understood as properties of graph grammar \( GG_{Kre}(MN) \).

### 2.2 Generalised Handle Graph Grammars

The modelling of the marked nets introduced in [GJRT83] is based on the notion of a generalised handle rewriting grammar (GH grammar).

A directed node-labelled graph (or graph) is a quadruple \( G = (V_G, A_G, \Sigma_G, \phi_G) \) where \( V_G \) is a finite non-empty set of nodes, \( A_G \subseteq V_G \times V_G \) is a non-empty set of arcs, \( \Sigma_G \) is a finite non-empty set of labels, and \( \phi_G : V_G \rightarrow \Sigma_G \) is a labelling function. For every \( v \in V_G \),
\[
\deg_G^{in}(v) = \text{card}(\{ w \mid (w,v) \in A_G \}) \quad \text{and} \quad \deg_G^{out}(v) = \text{card}(\{ w \mid (v,w) \in A_G \}).
\]

A generalised handle is a connected graph \( H \) such that \( \text{card}(V_H) \geq 2 \) and one of the following holds:

1. There is exactly one node \( v \in V_H \) such that \( \deg_H^{in}(v) = 0 \) and for every \( w \in V_H - \{ v \} \), \( \deg_H^{out}(w) = 0 \).
2. There is exactly one node \( v \in V_H \) such that \( \deg_H^{out}(v) = 0 \) and for every \( w \in V_H - \{ v \} \), \( \deg_H^{in}(w) = 0 \).

If (1) holds then \( H \) is an out-handle (see Figure 2.2.1(a)), and if (2) holds then \( H \) is an in-handle (see Figure 2.2.1(b)).

A GH grammar scheme is a pair \( GS = (\Sigma, P) \), where \( \Sigma \) is a finite non-empty set of labels and \( P \) is a finite non-empty set of productions. Each production is of the form \( \pi = (a, \beta, c^{in}, c^{out}) \) where \( a \) and \( \beta \) are generalised handles over \( \Sigma \); and \( c^{in} \) and \( c^{out} \) are subsets of \( \Sigma \times \Sigma \) called respectively the connecting in-relation and connecting out-relation.

A GH grammar is a triple \( GHG = (\Sigma, P, Z_0) \) where \( (\Sigma, P) \) is a GH grammar scheme and \( Z_0 \) is a graph over \( \Sigma \) called the axiom. Note that each production \( \pi = (a, \beta, c^{in}, c^{out}) \) essentially describes a transformation which replaces \( a \) by \( \beta \). The relations, \( c^{in} \) and \( c^{out} \), describe how to connect \( \beta \) to its immediate neighbourhood.

Let \( GS = (\Sigma, P) \) be a grammar scheme, and let \( Z \) and \( Z' \) be graphs over \( \Sigma \). \( Z \) directly derives \( Z' \) in \( GS \) (denoted \( Z \Rightarrow_{GS} Z' \)) if there is a complete subgraph \( \gamma \) of \( Z \) and a production \( (a, \beta, c^{in}, c^{out}) \) in \( P \) with \( a \) being isomorphic to \( \gamma \), such that \( Z' \) is isomorphic to the graph \( X \) constructed in the following way:

Let \( \delta \) be isomorphic to \( \beta \) with \( V_\delta \cap (V_\beta \cup V_Z) = \emptyset \). Then
\[
V_X = V_\delta \cup (V_Z - V_\beta)
\]

![Figure 2.2.1](image-url)
(b) For all $v \in V_X$, $\phi_X(v) = \phi_\delta(v)$ if $v \in V_\delta$, and $\phi_X(v) = \phi_\gamma(v)$ if $v \in V_\gamma$.

(c) $A_X = A_\delta \cap (x,y) \cap (x,y) \cap V_y = \emptyset \cup A_\delta \cup A_\in \cup A_\out$ where

\[ A_\in = \{(x,y) \in (V_X \times V_\delta) \times V_\delta \mid (\exists z \in V_y, (y,z) \in A_Z) \land (\phi_X(x), \phi_X(y)) \in c_\in \} \]

\[ A_\out = \{(x,y) \in V_\delta \times (V_X \times V_\delta) \mid (\exists z \in V_y, (z,y) \in A_Z) \land (\phi_X(x), \phi_X(y)) \in c_\out \}. \]

The GH grammar model can be used to simulate the behaviour of the marked nets. Let $MN = (S,T,F,M_\delta)$ be a marked net such that $x \cdot \in x = \emptyset$ for all $x \in S \cup T$. The construction of the corresponding GH grammar $GHG(MN)$ proceeds as follows. Below we simplify the construction of [GJRT83] without losing the essential idea behind it.

It is assumed both $S$ and $T$ are ordered sets, and that $\Sigma$ is an ordered set of labels. $(\Sigma_S, \Sigma_T)$ is a partition of $\Sigma$ into two subsets such that there is a bijective order-preserving mapping $\sigma$ from $\Sigma_S$ onto $S$, and a similar mapping $\tau$ from $\Sigma_T$ onto $T$.

Let $t \in \Sigma_T$ and $p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_n \in \Sigma_\delta$ be such that

\[ t(t) = \{\sigma(p_1), \ldots, \sigma(p_m)\} \text{ and } t(t) = \{\sigma(q_1), \ldots, \sigma(q_n)\}. \]

Then a production $(t) \in P$ takes the form shown in Figure 2.2.2, where $c_\in$ and $c_\out$ are given by:

\[ c_\in = \{(p,u) \mid p \in (q_1, \ldots, q_n) \land t \neq u \land \sigma(p) \in r(u)\} \cup \{(t,p) \mid p \in (p_1, \ldots, p_m)\} \]

\[ c_\out = \{(p,u) \mid p \in (q_1, \ldots, q_n) \land t \neq u \land \sigma(p) \in r(u)\} \cup \{(t,p) \mid p \in (p_1, \ldots, p_m)\}. \]

The axiom $Z_0$ is derived from $MN$ in the following way:

1. $\text{card}(V_{Z_0}) = \text{card}(T) + \sum_{s \in S} M_\delta(s)$.
2. For each $t \in \Sigma_T$ there is exactly one node $v \in V_{Z_0}$ with $\phi_{Z_0}(v) = t$.
3. For each $s \in \Sigma_S$, $\text{card}((v \in V_{Z_0} \mid \phi_{Z_0}(v) = s)) = M_\delta(s)$.
4. Let $v, w \in V_{Z_0}$ with $\phi_{Z_0}(v) \in \Sigma_T$ and $\phi_{Z_0}(w) \in \Sigma_S$. Then $(v, w) \in A_{Z_0} \iff \sigma(\phi_{Z_0}(w)) \in r(\phi_{Z_0}(v))$.
5. Let $v, w \in V_{Z_0}$ with $\phi_{Z_0}(v) \in \Sigma_T$ and $\phi_{Z_0}(w) \in \Sigma_S$. Then $(v, w) \in A_{Z_0} \iff \sigma(\phi_{Z_0}(v)) \in r(\phi_{Z_0}(w))$.

To illustrate the above definition, we take $MN$ from Figure 2.1, and obtain $GHG(MN) = ((a,b,p,q,r,s), p(a), p(b), Z_0)$, where

$p(a) = (a_1, \beta_1, c_1^{in}, c_1^{out})$, and $p(b) = (a_2, \beta_2, c_2^{in}, c_2^{out})$ and $Z_0, a_1, \beta_1, a_2, \beta_2$ are shown in Figure 2.2.3.

The connecting relations are defined as follows:

\[ c_1^{in} = \{(a, p), (a, q)\}, c_1^{out} = \{(a, b, a, s)\}, c_2^{in} = \{(b, p), (b, s)\} \text{ and } c_2^{out} = \{(b, r)\}. \]

Note that we assumed $r(t) = t$ for $t \in (a, b)$ and $\sigma(x) = x$ for $x \in (p, q, r, s)$. Similarly as in the case of the approach introduced in [Kre81], $GHG(MN)$ can be shown to simulate very closely the behaviour of $MN$. 

6
2.3 Comparison

The two approaches to the modelling of Petri nets introduced in [Kre81] and [GJRT83] differ in many respects. In this section we will examine the way in which the nets are represented in both models, and discuss what kind of graph grammar productions would be needed to model reconfigurability. Intuitively, we will aim at a model in which transitions and places are neither created nor destroyed, however their pre- and post-sets can change. In particular, we would like to be able to model changes similar to that shown in Figure 2.3.1, where firing of the transition changes the connectivity of places q and s. First we consider the way in which a marked net \( MN = (S,T,F,M_0) \) might be transformed into a graph \( Z_0 \) [GJRT83] models each of the tokens of the marking \( M_0 \) as a separate node, and for each such node there are arcs joining it with the pre- and post-transitions of the place the token is associated with. For the nets of Figure 2.3.1 and the tokens placed in place q this is illustrated in Figure 2.3.2. It seems that such a representation of tokens is not suitable for reconfigurable nets. The reason is that the firing of a transition, like that in Figure 2.3.1, would involve not only one token labelled by q, but (due to the change in the connectivity of place q) the pre- and post-sets of the second node labelled by q have to be adjusted. Since that second node is not involved in the transition, this leads to an non-local rewriting procedure, making the application of the productions rather awkward. On the other hand, the representation used in [Kre81] seems to
be a satisfactory one, mainly due to the fact that changes of the connectivity of nodes during reconfiguration involve only those nodes which are actually used by the production. Thus we will model the transformation from Figure 2.3.1 in the way illustrated in Figure 2.3.3.

The rewriting mechanism used in [Kre81] essentially consists in straightforward graph replacement, whereas the rewriting mechanism used in [GJRT83] uses graph replacement augmented by an embedding relation. The former mechanism is not best suited for the kind of application we are dealing with. To show this we consider the transformation from Figure 2.3.3. The arcs (s,c) and (b,s) were not present in the left-hand side graph, and would not be present in the production graph (unless we were prepared to define a large number of productions for each of the transition). We conclude that it would be rather difficult to model reconfigurating transitions using the approach of [Kre81]. The rewriting mechanism of [GJRT83], on the other hand, seems to be well suited for the reconfiguration we are trying to model. For the connecting relations cin and cout can be used to define the necessary adjustment of the endpoints of the arcs. To conclude our discussion, neither [Kre81] nor [GJRT83] provide a model which is fully satisfactory if one wants to model reconfigurable systems. However, by combining the representation of the nets used by [Kre81] with the rewriting mechanism of [GJRT83] one can obtain a graph grammar model suitable for the modelling of the systems like that of Figure 2.3.1.
3. Reconfigurable Graph Grammars

In the last section we provided an informal outline of the RGG (Reconfigurable Graph Grammar) model. Here we first discuss some of more detailed issues. The production \( pr \) of an RGG has the form \( pr = (a, \beta, c_{in}, c_{out}) \). The first two elements of \( pr \) are graphs, which in our case are quite similar to the L and R graphs of the productions in [Kre81]. Consider, for instance, the transformation in Figure 2.3.3. Suppose that \( b \) is an ordinary (i.e. causing no reconfiguration) transition of the net. For the net on the left, \( b \) would require the graphs \( a \) and \( \beta \) to be of the form shown in Figure 3.1(a). The situation, however, changes when we look at the net on the right of Figure 2.3.3. The graphs of Figure 3.1(a) are no longer sufficient, since the labelling of the post-place of \( b \) has changed. Hence we need another pair of graphs \( a \) and \( \beta \), as shown in Figure 3.1(b). This is rather worrying situation as it means that for each transition we would in general need a large set of productions, each production taking care of one potential local environment of that transition. Our solution to this problem is to use graphs \( a \) and \( \beta \) with the input and output places being labelled by label variables rather than concrete labels. We then need only one pair of graphs \( a \) and \( \beta \) for transition \( b \), as shown in Figure 3.1(c), where \( x \) and \( y \) are variables ranging over the set of labels \( \{p, q, r, s\} \). There are \( 4 \times 4 = 16 \) concrete instances of this production, and two such instances are shown in Figure 3.1(a,b).

The general form of the graphs \( a \) and \( \beta \) will be as in Figure 3.2, where \( x_1, \ldots, x_m \) and \( y_1, \ldots, y_n \) are label variables. When matched with a subgraph of the host graph to be transformed, \( a \) will provide
Figure 3.3

an assignment of values to variables $x_1, ..., x_m, y_1, ..., y_n$. The 'concrete' $a'$ and $b'$ obtained in such a way change the production from Figure 3.2 into a production similar to that in Figure 2.1.2. However, not all arcs joining the vertices of $a'$ and the host graph are to be saved. The reconfiguration specified by the connecting relations $c_{in}$ and $c_{out}$ will require that some of the arcs with the endpoints matching variables $x_1, ..., x_m$ will be affected and the new target and source nodes matching variables $y_1, ..., y_n$ will be used.

The form of the connecting relation $c_{in}$ will be $c_{in} = \{(a_1,x_{i_1},y_{j_1}), ..., (a_k,x_{i_k},y_{j_k})\}$, where $a_1, ..., a_k$ are transition labels. Each $(a_l,x_{i_l},y_{j_l})$ will be informally interpreted as follows: If there is an arc $(a_l,x_{i_l},y_{j_l})$ in the graph to be transformed, then it is to be deleted, and in its place a new arc $(a_l,y_{j_l})$ added. The form and interpretation of $c_{out}$ is similar. Taking as an example the production $p(a)$ for the graph of Figure 2.3.3, one might define it as in Figure 3.3.

3.1 The Model

Let $S$ and $T$ be two disjoint non-empty sets of labels, and let $r \in S \cup T$ be a label used for labelling the nodes representing tokens. Furthermore, let $\Sigma_{var}$ be a non-empty set of variables ranging over $S$, and $\Sigma = S \cup T \cup \{r\}$. We first define the action graph which will be used to represent the first two components of an RGG production.

**Definition 3.1.1**

An action graph $G$ is a finite directed node labelled graph $G = (M_{in}, S_{in}, t, S_{out}, M_{out}, A, \phi)$ where $V = M_{in} \cup S_{in} \cup \{t\} \cup S_{out} \cup M_{out}$ is the set of nodes, $A \subseteq V \times V$ is a set of arcs, and $\phi : V \rightarrow \Sigma \cup \Sigma_{var}$ is a labelling function such that the following are satisfied:

1. $\phi(M_{in} \cup M_{out}) = \{r\}$, $\phi(t) \in T$ and $\phi(S_{in} \cup S_{out}) \subseteq \Sigma_{var}$.
2. $\phi$ is injective on $S_{in} \cup S_{out}$.
3. $S_{in} \neq \emptyset \neq S_{out}$.  
4. Exactly one of the following holds:
   a. $M_{out} = \emptyset$ and there is a bijection $\rho : S_{in} \rightarrow M_{in}$ such that $A = \{(v, \rho(v)) \mid v \in S_{in}\} \cup (S_{in} \times \{t\}) \cup (\{t\} \times S_{out})$.
   b. $M_{in} = \emptyset$ and there is a bijection $\rho : S_{out} \rightarrow M_{out}$ such that $A = (S_{in} \times \{t\}) \cup (\{t\} \times S_{out}) \cup (\{v, \rho(v)\} \mid v \in S_{out})$.

If 4a holds then $G$ is an in-action graph (see Figure 3.1.1(a)); otherwise $G$ is an out-action graph (see Figure 3.1.1(b)). □
The next definition introduces the two remaining components of RGG production.

**Definition 3.1.2**

A **reconfiguration set** is a finite $c \subseteq T \times \Sigma_{\text{var}} \times \Sigma_{\text{var}}$ such that if $(a, x, y) \in c$ and $(a, x, z) \in c$ then $y = z$. □

**Definition 3.1.3**

An **RGG production** is a quadruple $pr = (G_{\text{in}}, G_{\text{out}}, c_{\text{in}}, c_{\text{out}})$ where

1. $G_{\text{in}} = (M_{\text{in}}, S_{\text{in}}, t, S_{\text{out}}, \emptyset, A, \phi)$ is an in-action graph and $G_{\text{out}} = (\emptyset, S_{\text{in}}, t, S_{\text{out}}, M_{\text{out}}, A', \phi')$ is an out-action graph such that $\phi|_{S_{\text{in}} \cup \{t\} \cup S_{\text{out}}} = \phi'|_{S_{\text{in}} \cup \{t\} \cup S_{\text{out}}}$.
2. $c_{\text{in}}$ and $c_{\text{out}}$ are reconfiguration sets such that if $(a, x, y) \in c_{\text{in}} \cup c_{\text{out}}$ then $x \neq \phi(t)$, $x \in \phi(S_{\text{in}})$ and $y \in \phi(S_{\text{out}})$. □

We now introduce static representation of reconfigurable system.

**Definition 3.1.4**

A **reconfiguration graph** is a finite directed node labelled graph $RG = (V_S, V_T, V_M, A, \psi)$ where $V = V_S \cup V_T \cup V_M$ is the set of vertices, $A \subseteq (V_S \times V_T) \cup (V_T \times V_S) \cup (V_S \times V_M)$ is the set of arcs, and $\psi: V \rightarrow \Sigma$ is the labelling mapping such that the following hold:

1. $\psi(V_S) \subseteq S$, $\psi(V_T) \subseteq T$ and $\psi(V_M) \subseteq \{t\}$.
2. $\deg_{RG}^{\text{in}}(v_m) = 1$ for all $v_m \in V_M$.
3. $\deg_{RG}^{\text{out}}(v_t) > 0$ and $\deg_{RG}^{\text{out}}(v_t) > 0$ for all $v_t \in V_T$.
4. $\psi$ is injective on $V_S \cup V_T$. □

The next definition captures those situations in which a production $pr$ can be applied to a reconfiguration graph $RG$. At this point we assume that the sets $\Sigma_{\text{var}}$ and $S$ are both totally ordered.

**Definition 3.1.5**

Let $pr = (G_{\text{in}}, G_{\text{out}}, c_{\text{in}}, c_{\text{out}})$ where $G_{\text{in}} = (M_{\text{in}}, S_{\text{in}}, t, S_{\text{out}}, \emptyset, A, \phi)$, be a production and let $RG = (V_S, V_T, V_M, A_{RG}, \psi)$ be an RG graph.

Then $pr$ is **applicable** to $RG$ through a mapping $\mu : M_{\text{in}} \cup S_{\text{in}} \cup \{t\} \cup S_{\text{out}} \rightarrow V_S \cup V_T \cup V_M$ if the following hold:

1. $\psi(\mu(M_{\text{in}})) = \{t\}$, $\psi(\mu(S_{\text{in}} \cup S_{\text{out}})) \subseteq S$ and $\psi(\mu(t)) = \phi(t)$.
2. $\mu$ is injective and order-preserving on $S_{\text{in}}$. 

11
3. \( \mu \) is injective and order-preserving on \( S_{\text{out}} \).
4. \( q \mid (q, \mu(t)) \in A_{\text{RG}} = \mu(S_{\text{in}}) \) and \( q \mid (\mu(t), q) \in A_{\text{RG}} = \mu(S_{\text{out}}) \).
5. If \( p \in M_{\text{in}} \) and \( q \in S_{\text{in}} \) are such that \( (q, p) \in A \) then \( (\mu(q), \mu(p)) \in A_{\text{RG}} \). □

The result of the application of a production satisfying Definition 3.1.5 is defined as follows.

Definition 3.1.6
Let \( pr, RG \) and \( \mu \) be as in Definition 3.1.5, and let \( S_{\text{out}} = \{ y_1, \ldots, y_n \} \).
Let \( V_0 = \{ v_1, \ldots, v_n \} \) be a set of \( n \) distinct elements such that \( V_0 \cap (V_S \cup V_T \cup V_M) = \emptyset \).
Define \( RG' = (V_S, V_T, V_M, A_{RG'}, \psi') \) as follows:
1. \( V_M' = V_M \cup \mu(M_{\text{in}}) \cup V_0 \).
2. \( A_{RG'} = A_{RG} \cdot \{(s, \mu(v)) | v \in M_{\text{in}} \} \cup \{(\mu(y_i), v_i) | i = 1, \ldots, n \}
   \cdot \{(v, \mu(w)) | v \in V_T \land w \in S_{\text{in}} \land \exists y, (\psi(v), \phi(w), y) \in c_{\text{in}} \}
   \cup \{(v, \mu(u)) | v \in V_T \land u \in S_{\text{out}} \land \exists w \in S_{\text{in}}, (v, \mu(w)) \in A_{RG} \land (\psi(v), \phi(w), u) \in c_{\text{in}} \}
   \cdot \{(\mu(w), v) | v \in V_T \land w \in S_{\text{in}} \land \exists y, (\psi(v), \phi(w), y) \in c_{\text{out}} \}
   \cup \{(p, v) | v \in V_T \land u \in S_{\text{out}} \land \exists w \in S_{\text{in}}, (\mu(w), v) \in A_{RG} \land (\psi(v), \phi(w), u) \in c_{\text{out}} \}
3. \( \psi' | V_S \cup V_T \cup V_M = \psi | V_S \cup V_T \cup V_M \) and \( \psi'(V_0) = \{ t \} \).

We say that \( RG' \) is obtained from \( RG \) through the application of \( pr \), denoted \( RG \Rightarrow pr RG' \) or \( \Rightarrow_{\text{pr}} \).

We then define an RGG grammar to be a pair \( \text{RGG} = (\text{RG}, P) \) where \( RG \) is a reconfiguration graph and \( P \) is a finite set of productions. For the transformation illustrated in Figure 2.3.1, a suitable RGG could be obtained by taking \( RG \) shown in Figure 2.3.3; \( p(a) \) as in Figure 3.3; and \( p(b) \) and \( p(c) \) obtained from \( p(a) \) by taking \( c_{\text{in}}' = c_{\text{out}} = \emptyset \) and changing the transition label.

For every P/T-net it is possible to find a reconfigurable graph grammar whose behaviour is equivalent to that of the net. The kind of behavioural equivalence we will consider is similar to the strong bisimulation of [Hennessy Milner 85].

Definition 3.1.7
Let \( MN = (S, T, F, M_0) \) be a P/T-net, and \( \text{RGG} = (\text{RG}_0, P) \) be a reconfigurable graph grammar. We say that \( MN \) and \( \text{RGG} \) are equivalent if there is a relation \( B \subseteq [M_0] \times R \), where \( R \) is the set of reconfiguration graphs such that the following hold:
1. \( (M_0, \text{RG}_0) \in B \)
2. If \( (M, \text{RG}) \in B \) and \( a \in T \) then
   a. \( M[a > M'] \Rightarrow \exists M' R \Rightarrow a R G' \land (M', R G') \in B \).
   b. \( R G \Rightarrow a R G' \Rightarrow 3 M'. M[a > M' \land (M', R G') \in B \). □

The above definition generalises in the obvious way to labelled nets.

Theorem 3.1.8
For every P/T-net there exists an equivalent reconfigurable graph grammar.

Proof
Let \( MN = (S, T, F, M_0) \) be a P/T-net.
For every \( M \in [M_0] \) let \( \Omega(M) \) be the set of all RG graphs \( RG = (V_S, V_T, V_M, A_{RG}, \psi) \) such that
1. \( V_S = S, V_T = T, \text{card}(V_M) = \sum s \in V_S M(s) \).
2. \( \psi \) is the identity on \( V_S \cup V_T \).
3. \( A_{RG} = P \cup A_0 \) where \( A_0 \subseteq V_S \times V_M \) is such that \( \text{card}(\{ x \mid (s,x) \in A_0 \}) = M(s) \) for all \( s \in S \).

Of course, the graphs in \( \Omega(M) \) are isomorphic.

For every \( a \in T \), let \( p(a) = (G_{in},G_{out},\emptyset,\emptyset) \) be an RGG production such that

\[
G_{in} = (M_{in},S_{in},t_{in},S_{out},\emptyset,A,\phi); \quad \text{card}(S_{in}) = \text{card}(a^*), \quad \text{card}(S_{out}) = \text{card}(a^*).
\]

Define RGG(MN) = \( (RG_0, (p(a)(a) \in T)) \) where \( RG_0 \) is an arbitrary graph in \( \Omega(M_0) \).

It can be shown, by following the definitions introduced in this section, that MN and RGG(MN) are equivalent. The relation \( B \) can be defined as \( B = \bigcup_{M \in M} M \times \Omega(M) \).

### 3.2 Reconfigurable Graph Grammars and Petri nets

In this section we show the main result of this paper proving that the languages generated by reconfigurable graph grammars can be generated by labelled Petri nets without \( \lambda \)-transitions. This means that the properties which can be expressed using the language generated by the reconfigurable system can be verified by applying a suitable translation from RGG to Petri net, and then using the methods and tools developed for ordinary nets.

Let \( RGG = ((V_S,V_T,V_M,A_{RG},\psi),P) = (RG,P) \) be an RGG graph grammar such that for every \( t \in T \) and \( k,n \geq 1 \), \( P \) contains at most one production labelled \( t \) and having exactly \( k \) input and \( n \) output places. If such a production exists, it will be denoted \( p_{t,k,n} \). We also denote \( V_T = \{ t_1, \ldots, t_m \} \) and \( Q = 2^{V_S - \{ \emptyset \}} \). The result which we want to prove can be formulated in the following way.

**Theorem 3.2.1**

There is a labelled marked net \( \Delta(RGG) \) such that the languages generated by RGG and \( \Delta(RGG) \) are identical.

To show the above theorem we first give the definition of \( \Delta(RGG) \), and then formulate two auxiliary lemmas.

**Definition 3.2.2**

We define \( \Delta(RGG) \) to be a labelled Petri net \( \Delta(RGG) = (II,\Gamma,F,M_0,\Pi) \), where \( MN = (II,\Gamma,F,M_0) \) is a marked Petri net and \( L: \Gamma \rightarrow T \) is a labelling function, in the following way.

1. \( II = V_S \cup W_1 \cup \ldots \cup W_m \cup \{ \delta_{p(t)}(C,D) \mid i < m \land C,D \in Q \} \), where \( W_i = \{ v_{p(t_i)}(i,C,D) \mid C,D \in Q \} \).
2. \( \Gamma = \{ \Gamma_{t_i,C,D} \mid i < m \land C,D \in Q \} \) where \( \Gamma_{t_i,C,D} = \{ \xi_{t_1,\ldots,t_m} \mid (t_1,\ldots,t_m) \in \{ v_{p(t_i)}(i,C,D) \times W_1 \times \ldots \times W_{t-1} \times W_{t+1} \times \ldots \times W_m \} \} \).
3. (For all \( t \in T \), C,D, L(t) = \psi(t_i).)
4. (For \( u = \xi_{t_1,\ldots,t_m} \) we define \( \pi u \) and \( u^* \) as follows (see Figure 3.2.1)).

Let \( \gamma_1 = v_{p(t_1)}(i,C,D) \) and \( \gamma_i = v_{p(t_i)}(i,C,D) \), for \( i = 2, \ldots, m \). Note that \( (u_2, \ldots, u_m) = (t_1, \ldots, t_i, t_{i+1}, \ldots, t_m) \).

If there is no production \( p_{p(t_i),k,n} \) then

\[
\pi u = C \cup \{ \delta_{p(t_i)}(i,C,D) \} \text{ and } u^* = D \cup \{ \delta_{p(t_i)}(i,C,D) \}.
\]

Otherwise we proceed as follows (below \( k = |C| \) and \( n = |D| \)):

Let \( p_{t_1,\ldots,t_n} = (G_{in},G_{out},G_{init},G_{term}) \) where \( G_{in} = (M_{in},\{ x_1, \ldots, x_k \}, t, \{ y_1, \ldots, y_n \}, A, \phi) \).

Let \( \mu_1 : \{ x_1, \ldots, x_k \} \rightarrow C \) and \( \mu_2 : \{ y_1, \ldots, y_n \} \rightarrow D \) be two order-preserving bijections.

Then \( \pi u = C \cup \{ \delta_{p(t_i)}(i,C,D) \} \text{ and } u^* = D \cup \{ \delta_{p(t_i)}(i,C,D) \} \)

and \( u^* = D \cup \{ \delta_{p(t_i)}(i,C,D) \} \).
where for $j = 2, \ldots, m$
\[
E_j = C_j \cdot \{ \mu_1(x) \mid \exists (p(u_1), x, y) \in \text{in}, \mu_1(x) \in C_j \} \cup \{ \mu_2(y) \mid \exists (p(u_2), x, y) \in \text{out}, \mu_2(y) \in C_j \}
\]
and
\[
F_j = D_j \cdot \{ \mu_1(x) \mid \exists (p(u_1), x, y) \in \text{in}, \mu_1(x) \in D_j \} \cup \{ \mu_2(y) \mid \exists (p(u_2), x, y) \in \text{out}, \mu_2(y) \in D_j \}.
\]

(5) For $s \in V_S$, $M_0(s) = \text{card}(\{x \mid x \in V_M \land (s, x) \in A_{RG}\})$. 
For $s = \psi_{(t_1, t_2)}(C, D)$, $M_0(s) = 1$ if $C = \{r \mid (t_1, r) \in A_{RG}\}$ and $D = \{r \mid (t_2, r) \in A_{RG}\}$; otherwise $M_0(s) = 0$.
For $s = \delta_{(t_1, t_2)}(C, D)$, $M_0(s) = 1$ if there is production $p_{(t_1, t_2), [C, D]}$; otherwise $M_0(s) = 0$. \(\square\)

The first lemma essentially says that at most one copy of each transition labelled $t$ is 'active' at any time. The next two lemmas show that RGG and $\Delta(RGG)$ can mutually simulate their derivations. All three lemmas follow directly from the definition of $\Delta(RGG)$ and together they imply Theorem 3.2.1.

**Lemma 3.2.3**
Let RGG and $\Delta(RGG)$ be as in Definition 3.2.2, and let $M \in [M_0 >.$

(1) For every $t \in V_T$ there is exactly one $s \in W_2$ such that $M(s) = 1$ and $M(W_1 - s) = \{0\}$.

(2) If $M[u > M'$ and $M[w > M''$ and $L(u) = L(w)$ then $u = w$. \(\square\)

---

*Figure 3.2.1*  
Shaded circles denote places in $V_S$, white circles denote places in $\Pi V_S$.  
Also $L(u) = \varphi(t_1)$ and $u = \xi_{\gamma_1, \ldots, \gamma_m}$
**Lemma 3.2.4**

Let RGG and \( \Delta(RGG) \) be as in Definition 3.2.2, and \( RG \Rightarrow_{\psi(t)} RG' \).

Then \((RG',P)\) satisfies the assumption formulated before Theorem 3.2.1, and there is \( u \in I \) such that \( M_u[u>M, L(u) = \psi(t) \) and \( \Delta(RG',P) = (II,I,F,M,L). \)

**Lemma 3.2.5**

Let RGG and \( \Delta(RGG) \) be as in Definition 3.2.2, and \( M_u[u>M. \)

Then there is \( t \in V_I \) such that \( RG \Rightarrow_{\psi(t)} RG' \) for some \( RG', \psi(t) = L(u), (RG',P) \) satisfies the assumption formulated before Theorem 3.2.1, and \( \Delta(RG',P) = (II,I,F,M,L). \)

In most cases it is possible to simplify the net \( \Delta(RGG) \) by taking into account the properties of the reconfiguring sets and the initial structure of the system. To illustrate the last definition we show in Figure 3.2.2 a simplified version of a net modelling RGG for the net in Figure 2.3.1 and defined just after Definition 3.1.6.

**Acknowledgement**

The author would like to thank Prof. G. Vidal-Naquet for his comments on the earlier versions of this paper. This research has been partially supported by the ESPRIT Basic Research Action No.3148 DEMON.

**References**


![Figure 3.2.2](image-url)


