Logist. Res. (2011) 3:159–175
DOI 10.1007/s12159-011-0054-9

ORIGINAL PAPER

Modeling production networks with discrete processes by means of communities of autonomous units

Hans-Jörg Kreowski · Sabine Kuske · Caroline von Totth

Received: 10 September 2010 / Accepted: 18 March 2011 / Published online: 19 April 2011

Abstract Communities of autonomous units are devices for the visual modeling of interactive logistic processes. The framework is founded on rule-based graph transformation and allows specifying autonomous units in such a way that they can run in parallel and make their decision about future actions independently of each other. The usefulness of the framework is demonstrated in this paper by modeling a new variant of production networks with discrete production processes. One of the main results shows that the visual model is correct with respect to a more traditional quantitative mathematical model.

Keywords Production networks · Autonomous units · Discrete production processes · Stability · Graph transformation

1 Introduction

Logistic systems and networks become more and more dynamic and structurally complex due to the fast changes in customers' demands, the wide spectrum of employed technologies, the growing globalization, and other such factors. Logistic processes with central control are not realistic and flexible enough to deal with this situation—in particular, if parties with different interests are involved. The use of interactive processes with decentralized and autonomous control is more adequate and may solve the problem, but can also cause new difficulties. How can one guarantee that the autonomous processes cooperate properly? How can interactive processes be described in a suitable way? How can one analyze their behavior and prove desired properties like conflict freeness, termination in time, and reachability of goals? To answer these questions, formal modeling methods for cooperating autonomous processes are indispensable. In logistics, there are various modeling methods like business process models, UML, Petri nets, multi-agent systems, particle swarms and systems of equations and inequalities. However, most of them do not provide both: a formal framework to prove properties on one hand and features to express decentralization, interaction, cooperation, and autonomy on the other hand. Logistic modeling of today often relies on testing and simulation which allow to track down unwanted behavior and to guarantee that a finite collection of inputs works properly, but cannot make sure that desired properties hold for the whole system and all its runs (cf., e.g., [12]).

As an alternative approach, we offer communities of autonomous units as devices for formal and visual modeling of interacting logistic processes (cf. [10, 11, 13]). In this paper, we demonstrate their usefulness by introducing and studying a new variant of production networks where the usual mathematical models are accompanied by specifications as communities of autonomous units.

Production networks are investigated in many variants in logistics and control theory mainly with continuous production processes (cf., e.g., [1–3, 6, 14–16]). A major

H.-J. Kreowski (✉) · S. Kuske · C. von Totth
Department of Mathematics and Computer Science, University of Bremen, P.O. Box 330 440, 28334 Bremen, Germany
e-mail: kreo@informatik.uni-bremen.de

S. Kuske
e-mail: kuske@informatik.uni-bremen.de

C. von Totth
e-mail: caro@informatik.uni-bremen.de

topic is the stability meaning that the quantities in a production network do not grow beyond any bound. While there are many applications for which the assumption of continuous inflow, outflow, production, and distribution is adequate, there are others for which a stepwise processing is more realistic (e.g. if production and distribution are done from time to time only).

In this paper, we model production networks with discrete production processes in two ways. We start with a quantitative mathematical model as it is quite common in logistics. Based on the mathematical model, two results can be proven that stress the basic properties of this kind of production networks. The first result states that production processes are free of loss because all quantities at all production sites that are delivered or put in during the process are stored or distributed or put out eventually. The second result concerns deterministic production networks for which we can provide a stability criterion. But this sufficient condition applies only to particular production networks. Therefore, one may wonder how the other networks run and work. For this purpose, we propose a second model of production networks by means of communities of autonomous units.

The modeling framework of communities of autonomous units offers the following features of interest:

1. It is *rule-based* so that the operational semantics provides the running processes. In other words, production processes do not have to be defined on a metalevel, but are given by the framework. Moreover, the operational semantics is independent of any particular implementation so that it is easier to understand than programing code.
2. As it is based on graphs, it is *visual* so that one can look at the running processes and see what happens—at least in principle. In practice, one needs visual simulation tools which are available for the framework in prototypical form (see Sect. 5 for more details).
3. It supports *autonomy* in form of control conditions. Each autonomous unit has a control condition to decide which rule is applied next out of all possible rule applications. But one can also employ control conditions on the community level to coordinate the interacting processes of the units and to avoid chaotic behavior in this way.
4. It provides *parallelism* so that autonomous units can run simultaneously. Parallel processing may be possible or mandatory. In the latter case, synchronization is organized by a special type of control condition. Moreover, the framework of graph transformation provides means to find out which activities can be performed in parallel without conflicts and whether required parallelism is possible at all.
5. From the point of view of logistics, our framework provides the basic elements of a language to model interactive logistic processes with autonomous control in a visual way. This allows to enhance the usual simulation of processes by a visual simulation on graph transformation tools. Furthermore, as the process semantics is formally defined, one can prove properties of the running processes like termination and correctness. There is also the future perspective of tool support because several research activities on automatic verification of graph transformation are going on.

The paper is organized in the following way. In the following section, the basic notions of production networks and their discrete processes are introduced by means of quantitative modeling. In Sect. 3, a sufficient condition for the stability of deterministic production networks is given. Section 4 treats production networks as communities of autonomous units where this rule-based and visual modeling framework is introduced, too. In Sect. 5, it is discussed shortly how the visual simulation of production networks is implemented on the graph transformation engine GrGen.NET (cf. [8]). Section 6 deals with a generalization of production networks by rules that change distribution rates, which are constant in the basic model. Section 7 concludes the paper by pointing out some topics of future research.

2 Production networks and discrete production processes

In this section, our most elementary notions of production networks and their processes are introduced. It follows the concept of supply and production networks as studied, for example, in [5, 9], but we replace continuous production processes by stepwise processing.

A production network consists of a set of production sites, which are numbered from 1 to n (without loss of generality). Some of them are also input sites and some output sites. Each site has got a maximum production rate. Each input site has a maximum input rate in addition, and each output site a maximum output rate accordingly. All of them serve as upper bounds. But for greater flexibility, each of them may be infinite so that no restriction is imposed in this case. The dynamics of a production network is given by production processes, which consist of stepwise inputs, changes of production states, and outputs. A production state provides a quantity per site. To change a current state, an input rate, a production rate, and an output rate are chosen for each site. The input rate is added to the current quantity, the output rate is part of the production rate which is subtracted from the quantity, and the difference of the production rate and the output rate is distributed. The fraction each site gets is established by the distribution matrix of the network. This yields a follow-up state so that such steps can be iterated. To start a process, an initial state is chosen. Moreover, some conditions are required. The input rates, the production rates, and the output rates should never exceed their maxima. The production rates chosen for some next step should not exceed the current quantities. And the distribution rates should make sure that the amount of site quantites that is distributed equals the sum of fractions that arive at the sites. If one wants to emphasize the network aspect, then one may consider the graph underlying a production network with the sites as nodes and an edge from site i to site j whenever the (i, j)-entry of the distribution matrix is greater than 0.

In a more concrete setting, one may think of production sites as various plants that are connected by roads or tracks so that raw material is delivered, processed material is transported from plants to other plants according to some distribution plan, and finished material is taken away by trucks or trains within a certain period of time. This establishes a production cycle, and the iteration of such steps defines a production process.

Before production networks and their processes are defined formally, some notational conventions are needed.

The set of natural numbers is denoted by $\mathbb{N}$, $\mathbb{N}\backslash\{0\}$ is denoted by $\mathbb{N}_{>0}$ and $[k]$ denotes the subset $\{1,\ldots,k\}$ of $\mathbb{N}$. The set of real numbers is denoted by $\mathbb{R}$; we use $\mathbb{R}_+$ to describe the set of non-negative real numbers with 0.

Given a set X and $n \in \mathbb{N}$, the set of all n-vectors $x = (x_1,\ldots,x_n)$ with $x_i \in X$ for $i \in [n]$ is denoted by X^n. If X is ordered by $\leq$, then the order is extended to X^n where $x \leq y$ for $x, y \in X^n$ means $x_i \leq y_i$ for $i \in [n]$. Accordingly, the set of all infinite sequences $y = (y_i)_{i\in\mathbb{N}}$ with $y_i \in X$ for $i \in \mathbb{N}$ is denoted by $X^{\mathbb{N}}$. The set of all n,n-matrices $a = (a_{ij})_{i,j\in[n]}$ with $a_{ij} \in X$ for $i,j \in [n]$ is denoted by $X^{n\times n}$. The extra symbol ∞ denotes infinity and is greater than every real number, i.e. $r < \infty$ for $r \in \mathbb{R}$.

Definition 1 (*Production network*) A *production network* *PN* consists of

- a set $[n]$ of *production sites* for some $n \in \mathbb{N}$,
- vectors $maxin, max, maxout \in (\mathbb{R}_+ \cup \{\infty\})^n$ the entries of which are called *maximum input rates*, *maximum production rates*, and *maximum output rates*, respectively, and
- a *distribution matrix* $d \in \mathbb{R}_+^{n\times n}$ with

$$\sum_{j=1}^{n} d_{ij} = 1$$

for $i \in [n]$.

Remarks

1. A production site j with $maxin_j > 0$ is called an *input site* and, accordingly, an *output site* if $maxout_j > 0$.
2. The *graph* underlying *PN* is given by $G(PN)) = ([n], E)$ with $E = \{(i,j) \in [n]^2 \mid d_{ij} > 0\}$ meaning that the sites are the nodes and there is an edge from site node i to site node j whenever the distribution rate d_{ij} is greater than 0.
3. The introduced notion is oriented on the concepts of production networks in [5, 9]. Many variations and extensions are possible, like lower bounds for input, production and output in addition to the upper bounds, or variable distribution rates rather than invariant ones. This is further discussed in Sects. 6 and 7.

Example 1 A sample production network SAMPLE is given by the components of Fig. 1. It has seven production sites: 2 and 5 are the input sites with 100 and 200, respectively, as maximum input rates and 1, 3, and 6 are the output sites with unbounded output. The distribution matrix and the vector of maximum production rates complete the quantitative model. The underlying graph is depicted in Fig. 2. It is extended in Sect. 4 where a visual representation of entire production networks is presented.

Definition 2 (*Production process*) Let *PN* be a production network.

1. A *production state* is a vector of *site quantities* $q \in \mathbb{R}_+^n$.
2. Let q be a production state, $in, p, out \in \mathbb{R}_+^n$ vectors of input rates, production rates, and output rates, respectively, with $in \leq maxin$, $p \leq max$, $out \leq maxout$ and $out \leq p \leq q$. Then, the *follow-up state* q' is defined by

$$q'_j = q_j + in_j - p_j + \sum_{i=1}^{n} d_{ij}(p_i - out_i) \qquad \text{for } j \in [n].$$

production network SAMPLE

$$d = \begin{pmatrix} 0 & .25 & .50 & .25 & 0 & 0 & 0 \\ .50 & 0 & 0 & .50 & 0 & 0 & 0 \\ 0 & 0 & .75 & 0 & 0 & .25 & 0 \\ 0 & .25 & .25 & 0 & .25 & .25 & 0 \\ 0 & 0 & 0 & .25 & 0 & 0 & .75 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & .25 & 0 & 0 & .75 & 0 \end{pmatrix}$$

$$maxin = (0, 100, 0, 0, 200, 0, 0)$$

$$maxout = (\infty, 0, \infty, 0, 0, \infty, 0)$$

$$max = (100, 150, 150, 150, 250, 200, 200)$$

Fig. 1 An example of a production network

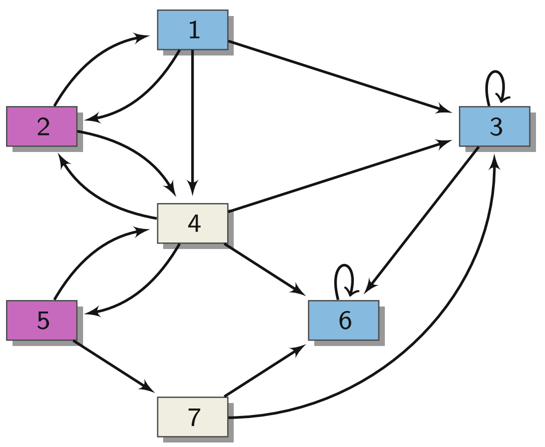


Fig. 2 The underlying graph of the production network SAMPLE

3. This construction is called a *production step* from q to q' and is denoted by

   $q \xrightarrow[in,p,out]{} q'$.

4. A *production process* is an infinite sequence of production states $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ such that, for every $k \in \mathbb{N}_{>0}$ there are vectors $in_k, p_k, out_k \in \mathbb{R}^n_+$ with $q_{k-1} \xrightarrow[in_k,p_k,out_k]{} q_k$.

Remarks

1. It should be noted that, given a production state, a production step is always defined for any choice of input rates, production rates, and output rates. Therefore, it causes no problems to assume that production processes run forever.
2. But if one is interested in finite processes, then one can consider just the prefixes $q_0, \ldots, q_k$ of a production process $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ for some $k \in \mathbb{N}$.
3. In particular, one may consider production processes $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ as finite if there is an activity bound $k \in \mathbb{N}$ such that all input, production, and output rates for $l > k$ are 0 and the quantity vectors become invariant, i.e. $q_l = q_k$. Then, only the sequence up to k matters at all and q_k can be considered as final state.
4. In a production step, the difference of the production rate and the output rate is distributed to the neighbor sites. Hence, it may be called *distribution quantity*.

Example 2 Given the production network SAMPLE of Example 1, it may start with the initial state (0, 0, 0, 0, 0, 0, 0), put in the maximum input in each step, choose always each current quantity as production rate, put this out at site 1 and 6 and half of it at site 3. Then, one gets a production process with the following first three steps:

$$\begin{aligned}(0,0,0,0,0,0,0) &\to (0,100,0,0,200,0,0)\\ &\to (50,100,0,100,200,0,150)\\ &\to (50,125,62.5,100,225,137.5,150)\end{aligned}$$

As in each step of a production process each quantity is partly kept, partly put out, and partly distributed, one may expect that the overall quantity in a production network is fully established by the initial quantities, all the inputs, and all the outputs and that there is no loss. To state this precisely, some notations are needed concerning the summation of inputs, outputs, and site quantities.

Let $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ be a production process with $q_{j-1} \xrightarrow[in_j,p_j,out_j]{} q_j$ for $j \in \mathbb{N}_{>0}$. Then, $Q_k = \sum_{j=1}^n q_{kj}$ denotes the overall quantity of state q_k for $k \in \mathbb{N}$. And, for $k \in \mathbb{N}_{>0}$, $In_k = \sum_{j=1}^k \sum_{i=1}^n in_{ji}$ and $Out_k = \sum_{j=1}^k \sum_{i=1}^n out_{ji}$ denote the *cumulated inputs* and the *cumulated outputs*, respectively, up to step k. Moreover, $In_0 = Out_0 = 0$.

Theorem 1 *Let $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ be a production process with $q_{j-1} \xrightarrow[in_j,p_j,out_j]{} q_j$ for $j \in \mathbb{N}_{>0}$. Then, the following holds:*

$Q_k = Q_0 + In_k - Out_k$ for all $k \in \mathbb{N}$.

The proof is omitted. The theorem can be shown by induction on the number of steps of a production process.

Our notion of production networks covers various special cases one encounters in the literature like networks with a single input site or a single output site (or both) and like acyclic networks where distributed quantities never come back to the distributing site. One may also get rid of the bounds and of the restrictions they impose on the free choice of input rates, production rates, and output rates by setting them to ∞. Or one may require additional properties of production processes like constant input rates or output rates proportional to the production rates, or exhaustive production rates that use up the site quantities up to the maximum production rates. These three properties together define a kind of deterministic production network that is further considered in the next section.

Instead of restrictions, one may also relax the notion of production networks. For example, the condition that the distribution rates of one site to all sites sum up to 1 may be replaced by

$$\sum_{j=1}^{n} d_{ij} \leq 1.$$

This would mean that a certain part of the distribution quantity gets lost in each step. Or the condition may be dropped completely allowing an increase in quantities while distributed. There may be even cases where negative quantities are meaningful.

There are at least two further aspects that could be subject to generalization. Instead of having only one quantity per site and step meaning that only one kind of material or goods is measured, one may consider a vector of quantities reflecting a variety of products. Moreover, it may be meaningful to replace the static distribution matrix by a dynamic one. The latter case is further discussed in Sect. 6.

3 Deterministic production networks and stability

In most applications of production networks, a site has a bounded storage capacity so that the question of stability becomes important. A production network is stable if the site quantities of each production process do not exceed a fixed bound. It will be shown in this section that deterministic production processes are stable if a certain system of linear equations has non-negative solutions.

In this context, a production network is called deterministic if its inputs are constant, its production rates are chosen exhaustively meaning that the site quantities are used up to the limit of the maximum production rates, and if its outputs are certain fractions of the production rates.

Definition 3 Let *PN* be a production network.

1. A production process $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ in *PN* is *stable* if there is an upper bound vector $m \in \mathbb{R}^n_+$ such that $q_k \leq m$ for each $k \in \mathbb{N}$.
2. *PN* is stable with respect to an upper bound vector $m \in \mathbb{R}^n_+$ if each production process $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ in *PN* with $q_0 \leq m$ is stable.
3. *PN* is *deterministic* with respect to some vector of *output factors* $a \in \mathbb{R}^n_+$ with $a_j \leq 1$ for $j \in [n]$ if every production process $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ with $q_k \xrightarrow[in_{k+1}, p_{k+1}, out_{k+1}]{} q_{k+1}$ for $k \in \mathbb{N}$ is subject to the following further conditions:
   - $in_{k+1} = maxin$,
   - $p_{(k+1)j} = \min(q_{kj}, max_j)$ for $j \in [n]$, and
   - $out_{(k+1)j} = a_j \cdot p_{(k+1)j}$ for $j \in [n]$.

Remarks

1. This means that the input is constant, the production rates are uniquely determined, and the output rates are fixed if the output factors are fixed such that there is only a single production process for each initial state.
2. As the production rates use up the site quantities up to the maxima, they are called *exhaustive*.

Theorem 2 *Let PN be a deterministic production network with the constant input vector $in = maxin$, the distribution matrix d and the vector of output factors a.* Let *$m \in \mathbb{R}^n_+$ be a solution of the system of linear equations*

$$(E - (d(a))^T) \cdot x = in$$

where E is the unit matrix, $d(a)$ is given by $d(a)_{ij} = d_{ij}(1 - a_i)$ for $i, j \in [n]$ and $(d(a))^T$ is the transposed matrix. Let $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ be a production process of PN with $q_0 \leq m \leq max$. Then PN is stable.

Again, the proof is omitted. It can be carried out by induction on the number of production steps.

Remarks

1. It is worth noting that the unique production process of the deterministic production network becomes constant if the initial state is chosen as the solution of the system of linear equations. In other words, one can show $q_k = m$ for all $k \in \mathbb{N}$.
2. Moreover, Theorem 2 still holds if one relaxes the assumption of constant inputs. Let *PN* be an arbitrary production network and $q \in (\mathbb{R}^n_+)^{\mathbb{N}}$ be a production process with the input vectors in_k for $k \in \mathbb{N}_{>0}$, exhaustive production rates and output rates that are determined by a vector *a* of output factors. Furthermore, let *m* be chosen as in the theorem. In other words, the assumptions of the theorem are fulfilled up to the constant input condition. Production networks with variable input turn out to be stable if all other assumptions are fulfilled.

Example 3 Consider the production network SAMPLE of Example 1 as a deterministic one with the constant input vector *maxin* and the vector $a = (1, 0, 0.5, 0, 0, 1, 0)$ of output factors. Then, Theorem 2 applies to SAMPLE. Its matrix $E - (d(a))^T$ and constant input vector *in* are

$$E - (d(a))^T = \begin{pmatrix} 1 & -0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.25 & 0 & 0 & 0 \\ 0 & 0 & 0.625 & -0.25 & 0 & 0 & -0.25 \\ 0 & -0.5 & 0 & 1 & -0.25 & 0 & 0 \\ 0 & 0 & 0 & -0.25 & 1 & 0 & 0 \\ 0 & 0 & -0.125 & -0.25 & 0 & 1 & -0.75 \\ 0 & 0 & 0 & 0 & -0.75 & 0 & 1 \end{pmatrix} \qquad in = \begin{pmatrix} 0 \\ 100 \\ 0 \\ 0 \\ 200 \\ 0 \\ 0 \end{pmatrix}$$

and the solution of the corresponding system of linear equations is given by:

$$m_1 = \frac{850}{13} \approx 65.38, m_2 = \frac{1700}{13} \approx 130.77,$$
$$m_3 = \frac{1540}{13} \approx 118.46, m_4 = \frac{1600}{13} \approx 123.08,$$
$$m_5 = \frac{3000}{13} \approx 230.77, m_6 = \frac{2280}{13} \approx 175.38,$$
$$m_7 = \frac{2250}{13} \approx 173.08.$$

Stability is a very important property of a production network because it makes sure that there will never be a shortage of storage capacity provided that the capacity is chosen according to the stability bound. If a network is unstable, it means that the input quantities are not distributed in such a way that all inputs are put out eventually, so that it piles up at some of the sites. To avoid this effect, the distribution rates should be adaptable to the waiting quantities at the receiving site following the principle that a site should get less input whenever its current quantity is high. The idea to readjust distribution rates and to get a more balanced distribution of quantities in this way is further considered in Sect. 6 where the production sites can decide about the quantities they deliver to neighbor sites in dependence of the quantities that are present there. For this purpose, we remodel production networks as communities of autonomous units in Sect. 4 , which allow to dynamize the distribution rates of each site by adding new rules to the site unit.

4 Production networks as communities of autonomous units

In this section, we show how production networks can be modeled as communities of autonomous units introduced in [10]. Communities of autonomous units are rule-based and graph-transformational devices to model interactive processes that run independently of each other in a common environment. An autonomous unit has a goal that it tries to reach, a set of rules the applications of which provide its actions, and a control condition which regulates the choice of actions to be performed. Each autonomous unit decides about its activities on its own depending on the state of the environment and the possibility of rule applications, but without direct influence of other ongoing processes.

The autonomous units of a community can act sequentially, in parallel, or concurrently (cf. [11, 13]). For modeling production networks by communities of autonomous units, a parallel semantics is suitable because each production site of a network can be naturally modeled by an autonomous unit that acts in parallel with all other units. More precisely, in every production step of a production process, each unit performs the following actions:

1. *Choose* Choose an input, an output, and a production rate subject to the conditions of Definition 2.
2. *Output* Subtract the output rate from the production rate.
3. *Distribute* Distribute the remaining distribution quantity to the neighbor sites according to the distribution matrix.
4. Calculate the new quantity as follows:
   (a) *Subtract* Subtract the production rate from the actual quantity q.
   (b) *Add* Add to the obtained quantity the input rate as well as all amounts obtained from the neighbors in their distribution steps.

Production networks together with their actual states can be modeled as graphs in a natural way where the production sites are represented as nodes labeled with their actual quantities and the non-zero values d_{ij} of the distribution matrix d are represented as edges from i to j labeled with d_{ij}. Consequently, the steps of production processes can be modeled as graph transformations. Since the common environments of communities are graphs and the actions of units are graph transformation rules, communities of autonomous units are well suited to specify production networks so that the described behavior of the sites can be directly modeled by the autonomous units of the community.

More precisely, the ingredients of autonomous units are taken from an underlying graph transformation approach providing a class $\mathcal{G}$ of graphs, a class $\mathcal{R}$ of graph transformation rules together with an operator $\Longrightarrow$ that specifies how to apply the rules to graphs, a class $\mathcal{C}$ of control conditions, and a class $\mathcal{X}$ of graph class expressions for specifying goals or environment properties, i.e., every expression x of $\mathcal{X}$ specifies a set $SEM(x)$ of graphs in $\mathcal{G}$. The environments that are transformed by communities belong to $\mathcal{G}$; the actions performed by the units correspond to applications of rules in $\mathcal{R}$; the decisions of the units are made according to control conditions in $\mathcal{C}$, and the goals are specified with an expression from $\mathcal{X}$. This leads to the following definition.

Definition 4 *(Autonomous units)* An *autonomous unit* is a system $aut = (g, R, c)$ where $g \in \mathcal{X}$ is the *goal*, $R \subseteq \mathcal{R}$ is a set of graph transformation rules, and $c \in \mathcal{C}$ is a control condition.

Autonomous units are meant to work within a community of autonomous units that modify the common environment together. Every community is composed of an overall goal that should be achieved, an environment specification that specifies the set of initial environments the community may start working with, a set of autonomous units, and a global control condition to restrict the possibilities of interaction among the units. The overall goal may be closely related to the goals of the autonomous units in the community. Typical examples are the goals admitting only successful semantic sequences w.r.t. one or all autonomous units in the community.

Definition 5 *(Community)* A *community* is a system $Com = (Goal, Init, Aut, Cond)$, where $Goal \in \mathcal{X}$ is a graph class expression called the *overall goal*, $Init \in \mathcal{X}$ is a graph class expression called the *initial environment specification*, *Aut* is a set of autonomous units, and $Cond \in \mathcal{C}$ is a control condition called the *global control condition*.

Communities for production networks consist of one unit per production site. The initial environment specification specifies production networks whose number of sites corresponds to the number of units in the community. The control condition requires to run all units infinitely long in parallel. In this paper, the goal specifies stability.

In the following, we present a concrete graph transformation approach that is suitable for modeling production networks.

4.1 A class of graphs for production networks

Production networks can be suitably represented as edge-labeled directed graphs consisting of nodes connected via directed labeled edges. More precisely, let Σ be a set of labels. An *edge-labeled directed graph* over Σ is a system $G = (V, E, s, t, l)$, where V is a set of nodes, E is a set of edges, s, t: $E \to V$ are the *source* and *target mappings* which assign to each edge its source and target node, respectively, and $l : E \to \Sigma$ is a mapping assigning a label to each edge in E.

For representing production networks, we must require that Σ contains the elements of $\mathbb{R}_+$ as well as the symbols ∞, *maxin*, *max*, *maxout,* and *q*. A production site j together with its maximum input rate $maxin_j$, its maximum production rate max_j, its maximum output rate $maxout_j$, and its current quantity q_j is represented by the edge-labeled graph shown on the left of Fig. 3 where $maxin_j$, max_j, $maxout_j$, and q_j are chosen as 12, 36, 15.5, and 5, respectively. The graph consists of a node equipped with a j-labeled loop, and for every $x \in \{maxin, max, maxout, q\}$ there is an x-edge pointing to a node equipped with a loop labeled with a real number (or with ∞) representing the quantity of x. Since the drawing of large production networks would lead to rather complex graphs we choose the more compact graphical representation of production sites on the right of Fig. 3. There, the site attributes are listed in the site node itself.

Accordingly, the representation of a production network *PN* with respect to a production state $q \in \mathbb{R}^n_+$ is the edge-labeled graph $env(PN)(q)$ that is constructed in the following way.

1. Take the underlying graph $G(PN)$ defined in the remarks after Definition 1 where an edge (i, j) has i as source, j as target and d_{ij} as label.
2. Extend each site node as described above.

Using the compacted representation, the example production network SAMPLE of Sect. 2 is represented by the graph in Fig. 4.

4.2 A rule class for modeling the actions of production sites

The class $\mathcal{R}$ of graph transformation rules chosen in this paper is based on the *double pushout approach*, which is

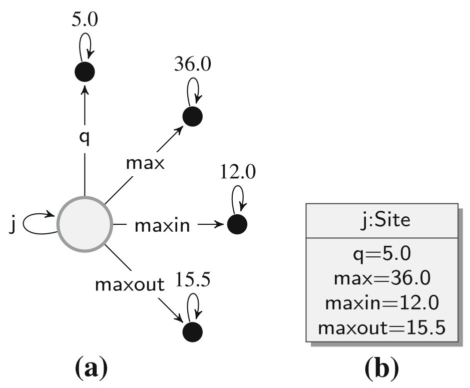


Fig. 3 A production site as a directed edge-labeled graph (*left*) and its compacted depiction (*right*)

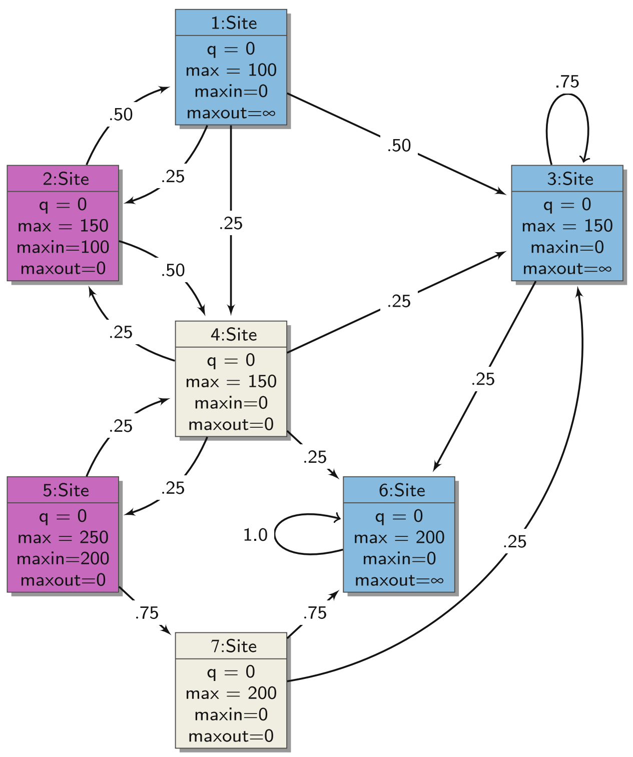


Fig. 4 The compacted graph for the example production network of Sect. 2

well studied in the literature (cf., e.g., [4, 7]). Every *graph transformation rule* of this class consists of three edge-labeled directed graphs L, K, and R such that K is a subgraph of L and R. Formally, a graph $G = (V, E, s, t, l)$ is a *subgraph* of a graph $G' = (V', E', s', t', l')$, denoted by $G \subseteq G'$, if V is a subset of V', E is a subset of E', $s(e) = s'(e)$, $t(e) = t'(e)$, and $l(e) = l'(e)$ for all edges e in E. The graphs L, K and R are called *left-hand side*, *gluing graph* and *right-hand side*, respectively. Rules are depicted in the form $L \rightarrow R$ where the nodes and edges of the gluing graph K are indicated by identical positions and node colors.

An example of a graph transformation rule is the rule *choose*(j) in Fig. 5. Its left-hand side and its gluing graph consist of the production site j. The right-hand side consists of the same site plus additional values for an input rate *in*, an output rate *out*, and a production rate *prod*. On the right of the rule, the constraints that must be satisfied by the values of *in*, *out*, and *prod* are listed.

A second example of a rule is *output*(j) given in Fig. 6. All three graphs of the rule consist of the production site j plus a chosen production rate p and a chosen output rate o. Additionally, the right-hand side contains a node, the value of which corresponds to the distribution quantity of site j because it is the difference of the production rate and the output rate.

A third example of a graph transformation rule is the rule *distribute*(j, i) given in Fig. 7. The left-hand side and the gluing graph contain the two production sites j and i that are connected with a d_{ji}-edge where d_{ji} is the corresponding entry in the distribution matrix. The distribution quantity dq of site j is given in the left-hand side, the gluing graph, and the right-hand side. The right-hand side consists of the same production sites, but site i is additionally equipped with a value g which corresponds to the fraction $d_{ji} \cdot dq$.

Graphs are transformed via applications of graph transformation rules. Roughly spoken, a rule $r = (L \supseteq K \subseteq R)$ is applied to a graph G by replacing an image of the left-hand side L with the right-hand side R such that the image of the common part K is not changed. Formally, the image of a graph L in G is the image of a graph morphism g from L to G. More precisely, for two graphs $H = (V_H, E_H, s_H, t_H, l_H)$ and $G = (V_G, E_G, s_G, t_G, l_G)$, a *graph morphism* g: $H \rightarrow G$ is a pair of structure-preserving mappings g_V: $V_G \rightarrow V_H$ and g_E: $E_G \rightarrow E_H$, 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 $e \in E_G$. The image $g(G) \subseteq H$ is also called a *match* of G in H.

Fig. 5 Rule *choose(j)*

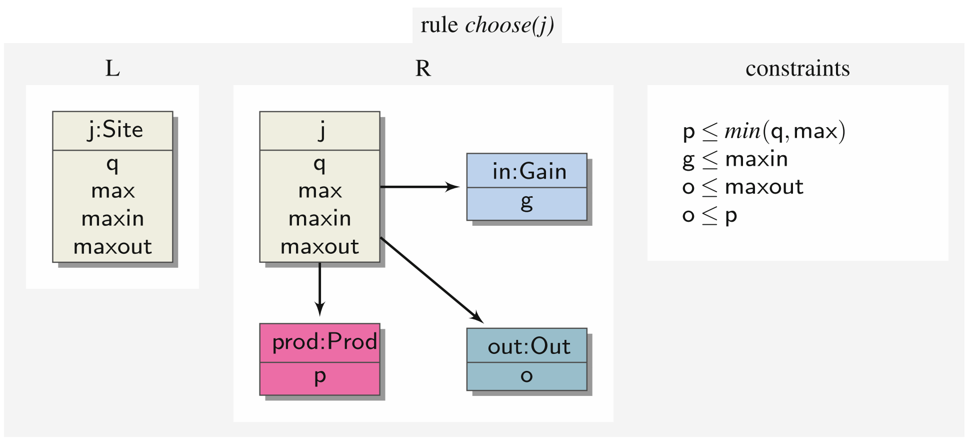
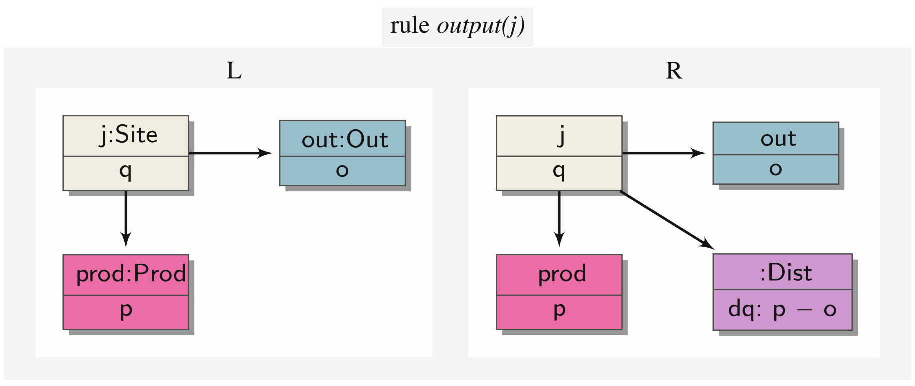


Fig. 6 Rule *output(j)*



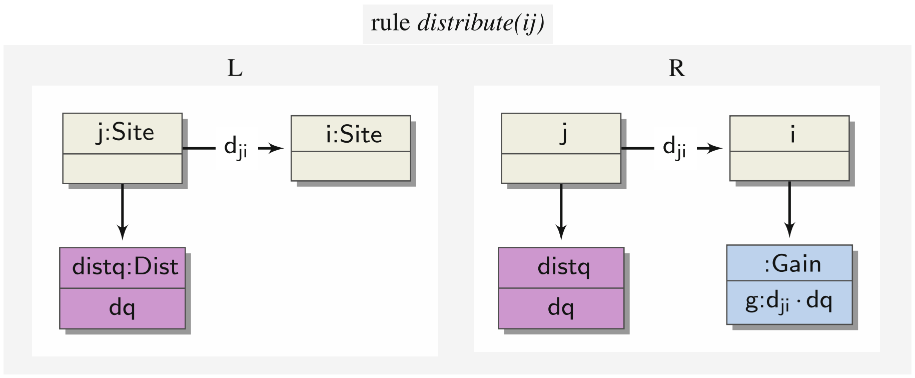


Fig. 7 Rule *distribute(ij)*

In more detail, an application of a rule $r = (L \supseteq K \subseteq R)$ to a graph G consists of the following steps:

1. A graph morphism g: $L \to G$ is selected subject to the following two *application conditions*:
   (a) the *dangling condition*: the removal of $g(L) - g(K)$ from G yields no dangling edges, and
   (b) the *identification condition*: if two nodes or two edges of L are identified (i.e., mapped to the same graph element) in the match of L, they must be in K.
2. $g(L) - g(K)$ is removed from G, yielding the graph Z.
3. R is added to Z yielding H by merging K with $g(K)$. This means that every item of R that is also in the gluing graph K is merged with its image in Z and the rest of R is added disjointly so that sources, targets, and labels are kept.

For example, for $j = 2$, the rule *choose*(j) in Fig. 5 can be applied to site 2 of the production network in Fig. 4 by selecting a value p for the production rate, a value o for the output rate, and a value g for the input rate such that the conditions of the rule are satisfied. If one chooses $p = 10.7$, $o = 0$, and $g = 100$, the resulting graph is depicted in Fig. 8. This rule application models the step *choose* mentioned before, i.e., it models the choice of an input, an output, and a production rate by site 2.

To the graph in Fig. 8, rule *output*(2) can be applied and afterward rules *distribute*(2,1) and *distribute*(2,4) (because only sites 1 and 4 are neighbors of site 2). An application of rule *distribute*(j, i) models the distribution of the amount $d_{ji} \cdot dq$ from site j to site i where dq is the distribution quantity calculated in the application of the rule *output*(j).

For modeling production processes adequately, the rule *choose* should be applied in each production step in parallel with each site. This is possible by combining rules to *parallel rules* by building the disjoint unions of their respective components. Formally, for rules $r_1, \ldots, r_n$ with $r_i = (L_i \supseteq K_i \subseteq R_i)$, their *parallel composition* $r_1 + \cdots + r_n$ yields the rule $(L_1 + \cdots + L_n \supseteq K_1 + \cdots + K_n \subseteq R_1 + \cdots + R_n)$ where $+$ denotes the disjoint union of graphs and the inclusions are the natural extensions of the inclusions in the rules $r_1, \ldots, r_n$.

For example, the parallel rule $r_1 + \cdots + r_7$ where $r_j = choose(j)$ for $j = 1, \ldots, 7$ can be applied to the production network in Fig. 4. This application models the parallel choice of the rates for each production site.

The presented rules model the above-explained steps *choose*, *output*, and *distribute* of a production step. The step *subtract* that subtracts the production rate from the current quantity is modeled with the rule *subtract*(j) given in Fig. 9. The step *add* which adds the quantities from the neighbors and the input rate to the current quantity is modeled by the rule *add*(j) given in Fig. 10.

It is worth noting that since the left-hand sides of *choose*(j), *output*(j), and *distribute*(j, i) are equal to the corresponding gluing graphs, nothing is deleted in their applications. Rules with deletion are *subtract*(j) and *add*(j).

4.3 Graph class expressions for initial production networks and goals

Graph class expressions serve to specify the set of initial environments of a community and the goals.

Typical examples of graph class expressions are concrete single graphs, sets of graphs, or sets of labels. Every graph as well as every set of graphs specifies itself, and every set Δ of labels specifies all graphs that are only labeled with symbols in Δ.

The initial environment of a community modeling the processes of a production network *PN* consists of the edge-labeled graph representing *PN* where each site has an arbitrary initial quantity. Concretely, for a production network *PN* with the components $[n]$, *maxin*, *max*, *maxout*, and d, we use the graph class expression $env(PN) = \{env(PN)(q) \mid q \in \mathbb{R}^n_+\}$ where $env(PN)(q)$ are the edge-labeled graphs introduced in subsection 4.1. For each node $i \in [n]$, its set of neighbors is defined by $N(i) = \{j \in [n] \mid d_{ij} > 0\}$.

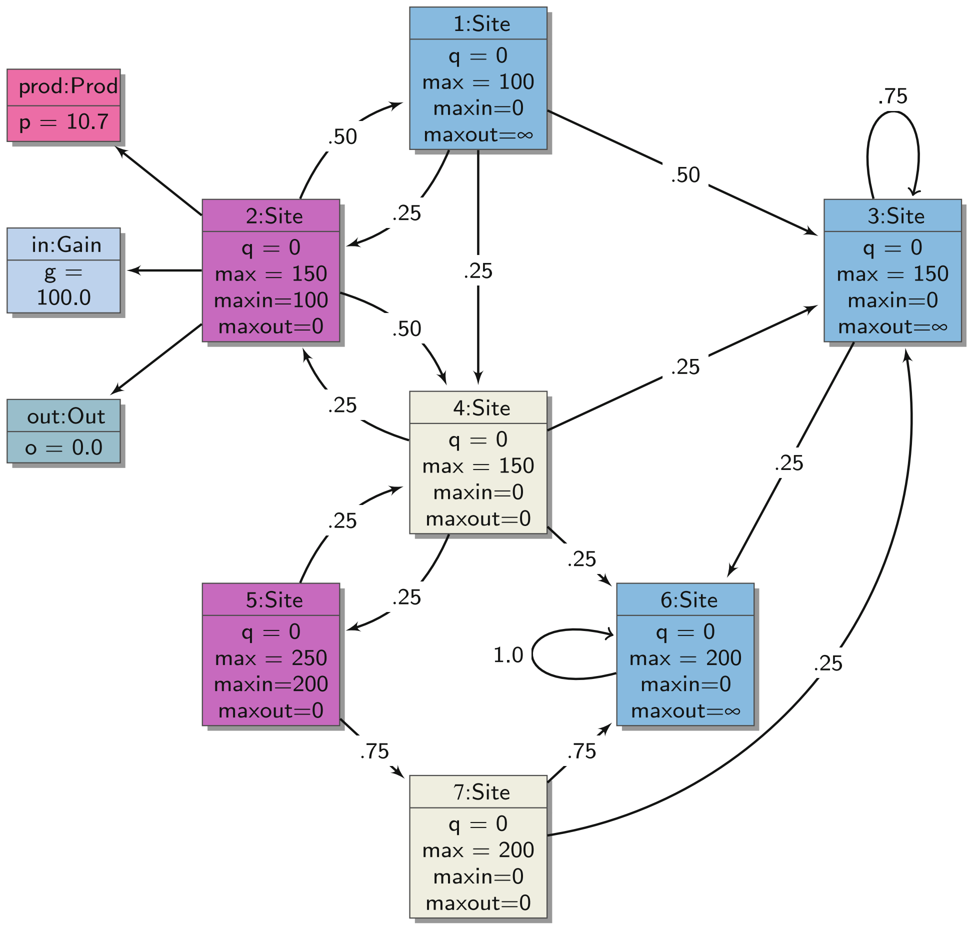


Fig. 8 A graph resulting from an application of the rule *choose(j)*

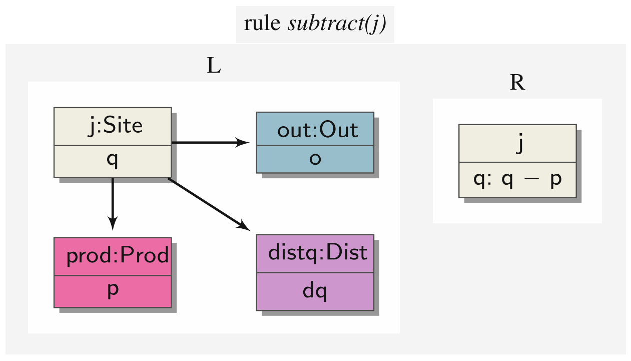


Fig. 9 Rule *subtract(j)*

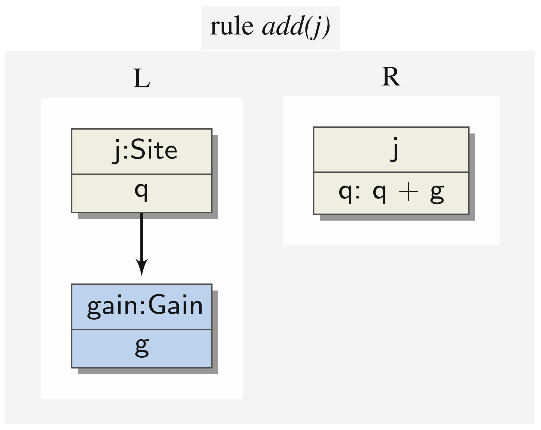


Fig. 10 Rule *add(j)*

The goal of each autonomous unit modeling a production site j can be specified with the graph class expression $bound_j$ where $bound \in \mathbb{R}^n_+$ is some fixed vector. It specifies all graphs $env(PN)(q)$ where the quantity of site j does not exceed $bound_j$, i.e., $SEM(bound_j) = \{env(PN)(q) \mid q_j \leq bound_j\}$. A transformation of the unit is said to be *successful* if the resulting graph meets the goal.

In the presented modeling of production networks, the global goal of the community expresses stability. It requires that the goal of every unit will be fulfilled after every run of the unit, i.e., after every step of a production process. This is expressed by the term *stable* with $SEM(stable) = \bigcap_{j\in[n]} SEM(bound_j)$ where n is the number of units in the community. It is worth noting that one can guarantee successful production processes if one models deterministic networks, takes a non-negative solution of the equation system in Theorem 2 as *bound*, and chooses the initial state q and the maximum production rate *max* such that $q \leq bound \leq max$.

4.4 Control conditions for a correct behavior of production networks

In many cases, rule application is highly non-deterministic—a property that is generally not desirable. On one hand, there can be several rules that are applicable to the current graph. On the other hand, there may be several matches for one and the same rule. Hence, a graph transformation approach provides a class of control conditions so that the degree of non-determinism of rule application can be reduced. Typical examples of control conditions are regular expressions over rules. It is well known that each regular expression over rules specifies a possibly infinite set of rule sequences. Hence, every regular expression *reg* over rules can be used as a control condition of an autonomous unit that allows all transformation processes in which the rules are applied in the same order as they occur in at least one of the rule sequences specified by *reg*.

For modeling the behavior of production sites, regular expressions are augmented by the condition *as_long_as_possible* and the parallel composition operator $+$. More concretely, the regular expression r (where r is some rule of an autonomous unit) means to apply r exactly once. The operator *as_long_as_possible* denoted by an exclamation mark can be applied to single rules with the effect that the rule must by applied as long as possible. The parallel composition operator $+$ is applied to a set R of rules and requires to apply all rules in this set in parallel, which corresponds to the application of the parallel rule $\Sigma_{r\in R} r$. Sequential composition of control conditions is denoted by a semicolon, i.e., the expression $c_1, \ldots, c_k$ prescribes to execute the control conditions $c_1, \ldots, c_k$ exactly in this order. Finally, non-deterministic choice is expressed by the symbol |, i.e., the expression $c_1|\cdots|c_k$ means to apply one of the conditions $c_1, \ldots, c_k$.

Concretely, the behavior of production site j can be modeled by applying the rules of the previous subsection according to the control condition

$$choose(j); output(j); \sum_{i\in N(j)} distribute(j,i); subtract(j); add(j)!$$

where $N(j) \subseteq [n]$ is the set of neighbors of site j as defined in the previous subsection. In words, this control condition prescribes to apply at first the rule $choose(j)$ and then the rule $output(j)$. Afterward the rule $distribute(j, i)$ is applied in parallel with all neighbors i of site j. In the next step, $subtract(j)$ is applied and then $add(j)$ as long as possible. It can be shown that the transformation processes that obey this control condition model the above-described steps *choose*, *output*, *distribute*, *subtract*, and *add* in the required order.

Apart from the autonomous units of a community, the community itself may be provided with a global control condition. As global control conditions we use the parallel operator ‖ and infinite sequential composition. More concretely, the global control condition $aut_1||\cdots||aut_k$ requires that the autonomous units $aut_1, \ldots, aut_k$ run in parallel each one exactly once. Moreover, the control condition c^∞ prescribes to apply the control condition infinitely often. The combination of both control conditions is used in the community presented in the following subsection.

4.5 The community for production networks

Based on the ingredients presented in the previous subsection, we can now define the community $C(PN)$ for a production network PN in a straightforward way as in Fig. 11, i.e., $C(PN) = (stable, env(PN), \{site(j) \mid j \in [n]\})$ where for $j \in [n]$ the autonomous unit $site(j) = (bound_j, \{choose(j), output(j), subtract(j), add(j)\} \cup \{distribute(j,i) \mid i \in N(j)\}, c_j)$ with $c_j = choose(j); output(j); \Sigma_{i\in N(j)} distribute(j,i); subtract(j); add(j)!$ is given in Fig. 12.

Summarizing, the following observation relates a running step in the community with a process step in the mathematical model.

Observation A running step of the community $C(PN)$ has the form

$$env(PN)(q) \Longrightarrow env(PN)(q')$$

where q' is obtained from q by $q'_j = q_j + in_j - p_j + \sum_{i=1}^n d_{ij}(p_i - out_i)$ for $j \in [n]$.

As a consequence of this observation, we get the following result.

Theorem 3 *Each production process q in PN corresponds to an infinite run of the community $C(PN)$, i.e., for every $k \in \mathbb{N}_{>0}$ there are vectors $in_k, p_k, out_k \in \mathbb{R}^n_+$ with $q_{k-1} \xrightarrow[in_k,p_k,out_k]{} q_k$ if and only if $env(PN)(q_{k-1}) \Longrightarrow env(PN)(q_k)$.*

community *C(PN)*

goal: stable
units: site(j) for j ∈ [n]
init: env(*PN*)
control: $(\text{site}(1)||\cdots||\text{site}(n))^\infty$

Fig. 11 Community *C(PN)*

Fig. 12 The autonomous unit *site*(*j*)

unit *site(j)*

goal:	bound_j
rules:	choose(j), output(j), distribute(ji), subtract(j), add(j) for $\mathsf{i} \in \mathsf{N(j)}$
control:	choose(j); output(j); $\sum_{\mathsf{i}\in\mathsf{N(j)}}$ distribute(ji); subtract(j); add(j)!

This shows that $C(PN)$ models PN correctly.

Remark The production process q is stable if for every $k \in \mathbb{N}_{>0}$ the graph $env(PN)(q_k)$ is in $SEM(stable)$.

4.6 Modeling deterministic production networks

Deterministic production networks can be modeled by replacing the constraints on the right side of rule *choose* in Fig. 5 with the properties of deterministic networks presented in Definition 3. Obviously, in this case, all modeled processes are stable if a solution of the equation system in Theorem 2 is chosen for *bound* and if this solution is between the state q_0 of the initial environment $env(PN)(q_0)$ and the maximum production rate *max* of *PN*, i.e., $q_0 \leq bound \leq max$.

5 Visual simulation

In order to simulate deterministic process runs on the sample production network in Fig. 1, we have implemented the general production community $C(PN)$ (cf. Fig. 11) using the graph transformation engine GrGen.NET (cf. [8]).

The GrGen.NET graph model is based on typed, attributed, directed multigraphs with inheritance. The base types at the core of this model are *Node* and *Edge*, and the primitive attribute data types *int, float, double, string, boolean,* and *object*, the latter denoting a .NET object.

We made use of the subpattern matching capability of GrGen.NET, using the iterated subpattern in order to simulate parallel rule application. GrGen.NET also does not provide autonomous units; however, it allows to structure rule application by embedding imperative calls to other rules into the declarative right-hand side of a rule. Furthermore, such calls may be controlled using, for example, regular expressions. We made use of this feature to emulate autonomous units very closely to our original specification.

The simulation runs very fast, with our example network SAMPLE completing 41 steps and reaching the maximal production rate at all seven sites in 156 milliseconds on an Intel Core i5 M520 CPU with 2.40 GHz and 6 GB of RAM, having found 1920 matches (many of these being parallel matches across the network) and performed 1920 graph rule applications in that time (see Fig. 13).

In order to simulate production runs on larger networks, we have written an additional graph grammar, which creates random production networks for test purposes (cf. Fig. 14). A graph with 402 nodes is generated in 655 ms; 3000 production steps are completed after another 21840 ms (i.e., some 21 seconds), with over 4 million matches found and rewrite steps executed in that time.

This type of simulation is valuable as a visual way to model and debug production networks or detect flaws in existing ones, altering them until they become stable. In particular, the simulation includes a visual debugger which allows to view in detail every step performed by the system, from the matching of rule patterns to nodes to the assignment of new values to variables. Additionally, the declarative nature of graph transformation rules makes the modeling less error prone, and the production process model easily scalable, e.g., by introducing different material types. Other possible extensions to the current model are sketched in the conclusion.

6 Production networks with variable distribution rates

In this section, the notion of production networks is extended by allowing to change the distribution rates such that the distribution matrix becomes variable. This serves two purposes. On one hand, the modeling of production networks becomes more flexible and more realistic. On the other hand, it is demonstrated how the modeling framework of communities of autonomous units can be used to specify aspects of autonomy.

An autonomous unit decides about its next action by choosing a rule application out of all possible ones. The decision depends on the control condition which may be the conjunction of several conditions of different kinds. A typical case is a condition that establishes some order or priority among the rules. For example, the control condition of the unit *site(j)* requires that first *choose(j)* must be applied then *output(j)* followed by *distribute(j,i)* and finally, *subtract(j)* followed by some *add(j)*'s. Consequently, the

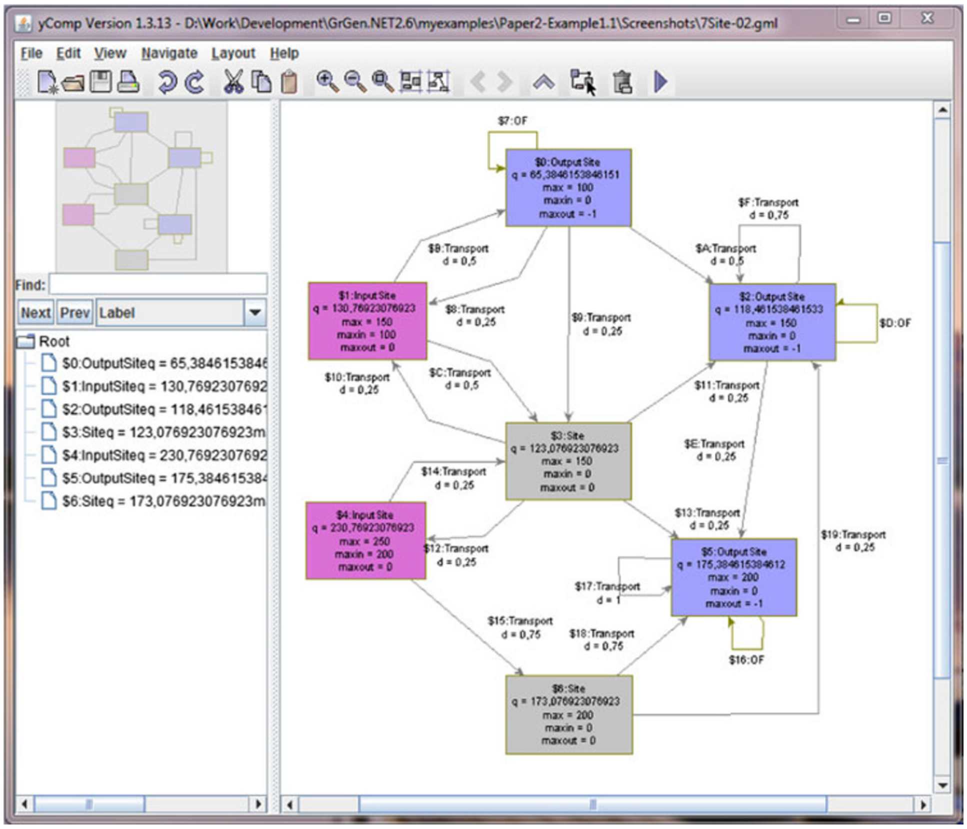

yComp Version 1.3.13 - D:\Work\Development\GrGen.NET2.6\myexamples\Paper2-Example1.1\Screenshots\7Site-02.gml
File Edit View Navigate Layout Help
Find:
Next Prev Label
Root
$0:OutputSiteq = 65,3846153846
$1:InputSiteq = 130,7692307692
$2:OutputSiteq = 118,46153846
$3:Siteq = 123,076923076923m
$4:InputSiteq = 230,7692307692
$5:OutputSiteq = 175,38461538
$6:Siteq = 173,076923076923m
$7:OF
$0:Output Site q = 65,3846153846151 max = 100 maxin = 0 maxout = -1
$F:Transport d = 0,75
$B:Transport d = 0,5
$A:Transport d = 0,5
$1:Input Site q = 130,76923076923 max = 150 maxin = 100 maxout = 0
$8:Transport d = 0,25
$9:Transport d = 0,25
$2:Output Site q = 118,461538461533 max = 150 maxin = 0 maxout = -1
$D:OF
$10:Transport d = 0,25
$C:Transport d = 0,5
$11:Transport d = 0,25
$3:Site q = 123,076923076923 max = 150 maxin = 0 maxout = 0
$E:Transport d = 0,25
$14:Transport d = 0,25
$4:Input Site q = 230,76923076923 max = 250 maxin = 200 maxout = 0
$13:Transport d = 0,25
$19:Transport d = 0,25
$12:Transport d = 0,25
$5:Output Site q = 175,384615384612 max = 200 maxin = 0 maxout = -1
$17:Transport d = 1
$15:Transport d = 0,75
$18:Transport d = 0,75
$16:OF
$6:Site q = 173,076923076923 max = 200 maxin = 0 maxout = 0

Fig. 13 Community C(SAMPLE) in GrGen.NET: all sites have reached their saturation point after 41 steps

order of rule applications is fixed by this condition. But the rules are generic since they contain variables that must be instantiated before the actual application can take place. While the actual values of the left-hand side variables are uniquely determined by the match of the rule in the environment, the unit can choose and decide about the values of the right-hand side variables. A second kind of control condition is given by constraints for these values. So far, we have used only two extreme cases: Either the choice is totally free within certain limits like the choice of p, q, g, and o in the rule *choose(j)*, or it is computed uniquely like $p - o$ in *output(j)* and $q - p$ in *subtract(j)*. How further decisions in between the two extreme cases can be designed and used is demonstrated in the following.

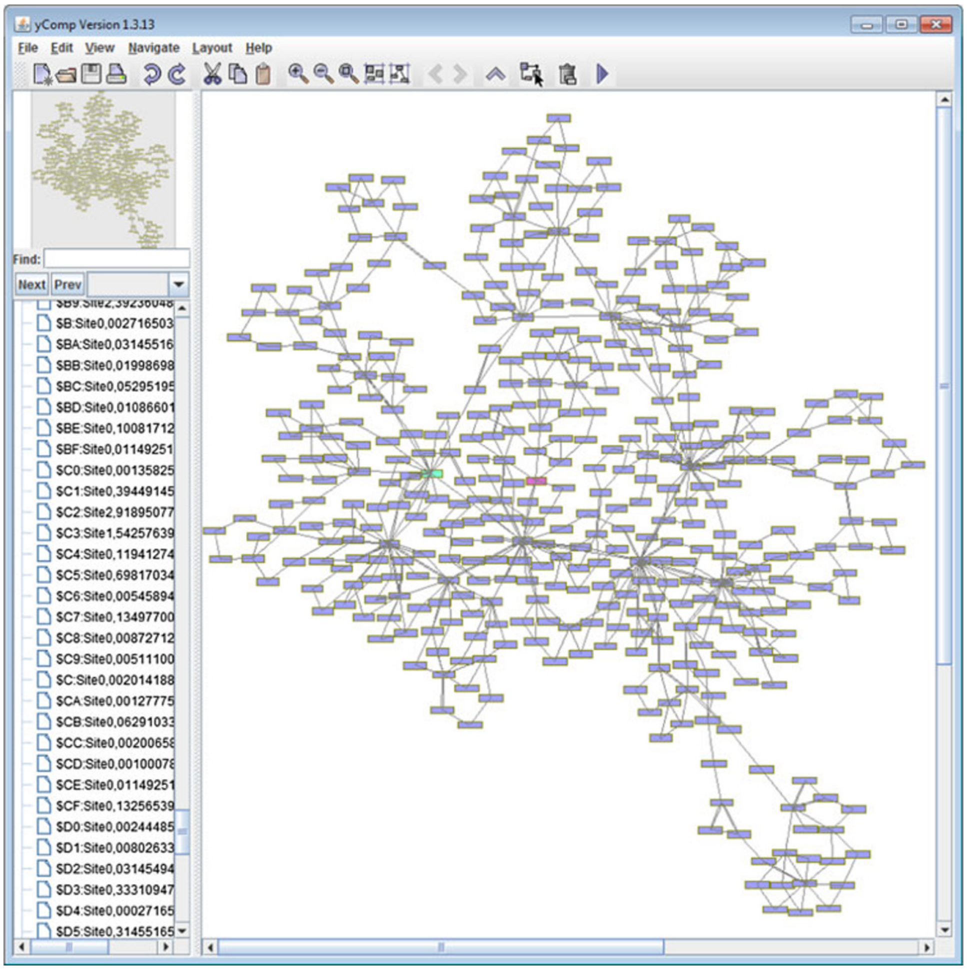
To make the distribution matrix variable, we enrich each production site by a new rule, the application of which changes the current distribution rates of the edges outgoing of the considered site. The new distribution rates are chosen due to proper constraints. To allow a variety of possibilities, the new rule is designed in a generic way. And some examples of constraints are provided. The first one is free choice. The second one reflects the quantities that wait for processing at the neighbor sites. The third one takes the maximum production rates into account additionally. In the latter two cases, the intention is to deliver the distribution quantities in such a way that the waiting time is reduced, meaning that the further processing is not delayed for too long and that the chance for stability is improved.

6.1 The rule to change the distribution rates

Let $j \in [n]$ be some production site and $i_1, \ldots, i_k$ for some $k \in \mathbb{N}$ be its neighbors that can be reached from j by a transport edge each. Then, the rule *adjust(j)* has the form given in Fig. 15. The left-hand side contains the sites $j, i_1, \ldots, i_k$ and the connecting edges. Moreover, for each neighbor i_l, there is a set of variables Var_l in the left-hand side so that the actual values are available whenever the rule is applied. Consequently, there is a unique matching for each environment graph. But to apply the rule, the new distribution rates $\widehat{d_{ji_l}}$ for $l \in [k]$ must be chosen or computed. Some possibilities are discussed in the next subsection where different sets of variables are used.

6.2 Constraints for changing the distribution rates

The simplest possibility is to allow a free choice. Then, the only constraint to be considered is that distribution rates for the site j must sum up to 1. The sets of variables may be empty in this case.



```
$B9:Site2,39236048
$B:Site0,002716503
$BA:Site0,03145516
$BB:Site0,01998698
$BC:Site0,05295195
$BD:Site0,01086601
$BE:Site0,10081712
$BF:Site0,01149251
$C0:Site0,00135825
$C1:Site0,39449145
$C2:Site2,91895077
$C3:Site1,54257639
$C4:Site0,11941274
$C5:Site0,69817034
$C6:Site0,00545894
$C7:Site0,13497700
$C8:Site0,00872712
$C9:Site0,00511100
$C:Site0,002014188
$CA:Site0,00127775
$CB:Site0,06291033
$CC:Site0,00200658
$CD:Site0,00100078
$CE:Site0,01149251
$CF:Site0,13256539
$D0:Site0,00244485
$D1:Site0,00802633
$D2:Site0,03145494
$D3:Site0,33310947
$D4:Site0,00027165
$D5:Site0,31455165
```

Fig. 14 A network with 400 nodes in GrGen.NET after 3000 production steps

$$\text{(constraint 1)} \quad \sum_{l=1}^{k} d_{ji_l} = 1$$

But this is not really a good idea because free choice may lead to quite chaotic processes.

As indicated at the end of Sect. 3 and the introduction of this section, a much better idea is to choose the new distribution rates in such a way that the chances for stability grow. A production process is unstable if there is at least one production site at which the quantities grow beyond any bound. The cause of this effect is that the site gets more delivered than it redistributes over time. To avoid the unbounded growth, one may shorten the delivered quantities by making the distribution rates smaller in inverse proportion to the amount of material piled up at the sites. The three following constraints are examples how a production site unit can autonomously control its distribution following this general principle.

The site j may consider the current quantities at the neighbor sites and assume that the smaller the quantity is, the faster the processing runs. The sets of variables must contain the quantities accordingly. This idea is reflected in the following constraint:

$$\text{(constraint 2)} \quad \widehat{d}_{ji_m} = \frac{b - q_{i_m}}{\sum_{l=1}^{k}(b - q_{i_l})}$$

for m $\in$ [k] and some $b \in \mathbb{R}_+$ with $q_{i_m} < b$

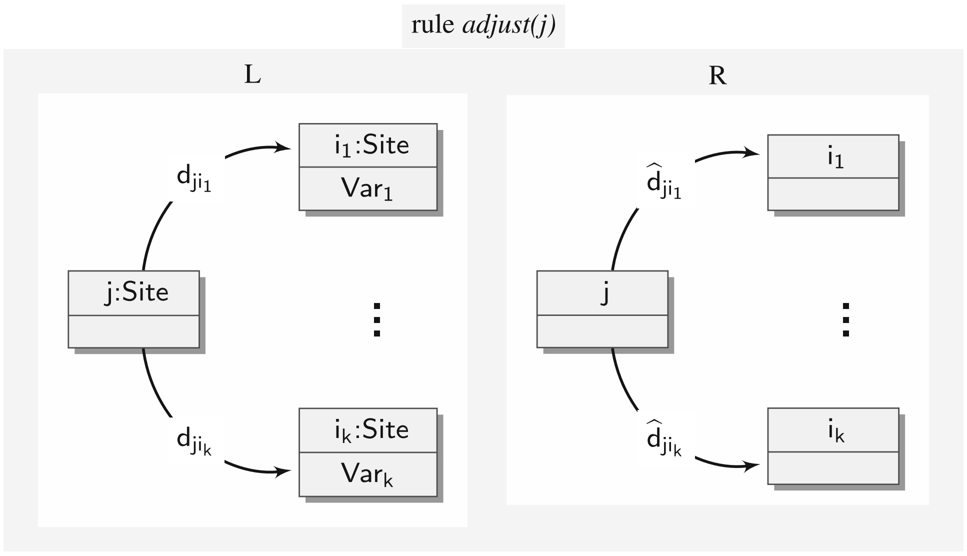
The differences $b - q_{i_m}$ are in converse order to the order of the quantities so that the larger the quantity is, the smaller the difference grows. The division by the sum makes sure that the new distribution rates sum up to 1:

$$\sum_{m=1}^{k} \widehat{d}_{ji_m} = \sum_{m=1}^{k} \frac{b - q_{i_m}}{\sum_{l=1}^{k}(b - q_{i_l})} = \frac{\sum_{m=1}^{k}(b - q_{i_m})}{\sum_{l=1}^{k}(b - q_{i_l})} = 1$$

Note that the new distribution rates reflect the differences between the current site quantities in lessened form if the upper bound b is chosen larger. We require that b is larger than $\max\{q_{i_l} \mid l \in [k]\}$ to avoid that any distribution rate becomes 0.

In the exceptional case that all quantities at neighbor sites are equal, the bound b must be greater—at least a bit—because otherwise the sum of all differences would be 0 and the quotient would not be defined.

Fig. 15 Rule *adjust(j)*



The reflection of the waiting time becomes more sophisticated if one replaces the quantities in constraint 2 by the quotients of quantities and maximum production rates.

$$\text{(constraint 3)} \quad \widehat{d}_{ji_m} = \frac{b - w_{i_m}}{\sum_{l=1}^{k}(b - w_{i_l})} \text{ with}$$
$$w_{i_m} = \frac{q_{i_m}}{\max_{i_m}} \quad \text{for } m \in [k] \text{ and some } b \in \mathbb{R}_+$$
$$\text{with } w_{i_m} < b$$

The latter quotient may be called *waiting number* because the smallest integer greater or equal is the minimum number of steps to process the current quantity. Clearly the sets of variables must contain the quantities and the maximum production rates.

The last explicit example takes into account that it may not always be reasonable to forget the old distribution totally so that one may like to mix the old rates with new ones. A weighted average will do this job:

$$\text{(constraint 4)} \quad \widehat{d}_{ji_m} = \frac{rd_{ji_m} + sd'_{ji_m}}{r + s} \quad \text{for } m \in [k]$$
$$\text{and } r,\ s \in \mathbb{R}_+ \text{ with } r + s > 0 \text{ and}$$
$$\text{some distribution rates } d'_{ji_1} \ldots d'_{ji_k}$$
$$\text{which may be chosen as one of}$$
$$\text{the three cases above}$$

Each of the four constraints (and other similar ones) can be used as control condition in the autonomous unit *site(j)* after it is enriched by the rule *adjust(j)*. As the control condition concerns only this rule, it may be placed beside the right-hand side of the rule (cf. Fig. 5).

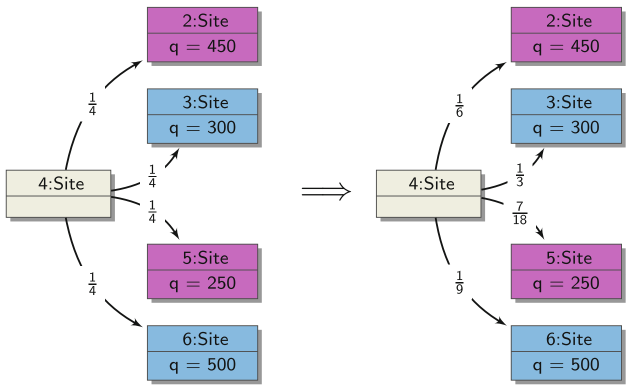


Fig. 16 Application of rule *adjust(4)* with constraint 2, with $b = 600$

6.3 Example

The production site 4 of SAMPLE has the neighbors 2, 3, 6, and 5 and distributes a quarter of the production rate to each of them due to the distribution matrix (cf. Figs. 1 and 4). Let us assume in addition the following current quantities: $q_2 = 450$, $q_3 = 300$, $q_5 = 250$, and $q_6 = 500$. Then, one can apply the rule *adjust*(4) using constraint 2 with $b = 600$. Figure 16 shows the rule application restricted to the significant part of the network. In Fig. 17, the same is depicted for constraint 3 with $b = \frac{7}{2}$. While the new distribution rates in the first case are smaller the larger the quantities are, the second case reflects the waiting numbers $w_2 = 3$, $w_3 = 2$, $w_5 = 1$, and $w_6 = \frac{5}{2}$. While, for example, site 2 gets a larger fraction from site 4 than site 6 in the first case, it is the other way round in the second case.

The considerations in this section exemplify how communities of autonomous units may be modified and extended to cover new aspects and features. Concerning the

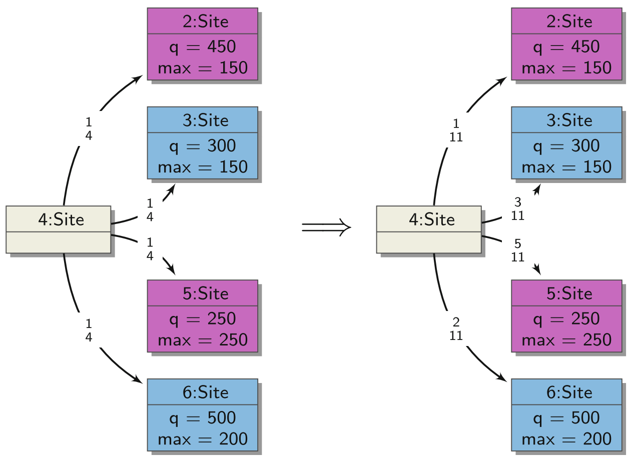


Fig. 17 Application of rule *adjust(4)* with constraint 3, with $b = \frac{7}{2}$

variable distribution rates, we have taken into account some measures that reflect something like the waiting time with a look-ahead of 1 (meaning that we access only the information provided by direct neighbors). In a similar way, one could involve larger look-aheads or criteria other than waiting time like pheromone traces (cf. [2]). We are also convinced that further principles of planning in production networks (cf. [1]) can be realized in this way.

7 Conclusion

In this paper, we have modeled and investigated a variant of production networks with discrete production processes. The usual quantitative modeling based on matrices and vectors has been supplemented by a visual modeling employing the rule-based and graph-transformational framework of communities of autonomous units. It has turned out that the community version models production networks correctly with respect to their mathematical description so that all results for one of the models apply to the other and conversely. Therefore, one gets both: On one hand, one can prove results like the stability of deterministic production networks provided that certain systems of linear equations are solvable; on the other hand, the visual simulation is supported. The attempt to bring two styles of modeling together may therefore be considered as promising. To shed more light on the significance of the approach, one may investigate the following topics:

1. The stability results may be improved by enlarging the class of production networks for which sufficient conditions yield stability. And one may also look for necessary conditions or even proper characterizations.
2. A good motivation to make the distribution rates variable is the chance for more stability. It would be nice to know whether this works and for which production network and which variability.
3. To make the model more flexible, one may enhance the notion of production networks by relaxing, modifying, or specializing various assumptions like the following:
   - There may be lower bounds of input, production, and output rates in addition to upper bounds.
   - There may be different kinds of materials and information flows through the network rather than a single homogeneous kind of quantities.
   - There may be particular time conditions for production and transportation at each site rather than the common-step assumption.
   - There may be more information about the production like costs, prices, etc. to refine the basis for the autonomous decision making and planning at the production sites, or one may also involve production goals into the consideration.
4. Another possible modification would be to assume that the produced and distributed material consists of a number of atomic items such that only integer division is possible. In this case, the graph-transformational model may be particularly suitable as the atomic items could be represented by atomic graph components explicitly.

We think that communities of autonomous units provide a suitable framework to model production networks with respect to the points 3 and 4 at least. As the framework is equipped with a well-defined syntax and semantics, it offers the perspective of further tool support beyond the visual simulation. For example, it should be possible to employ a verifier like a model checker, SAT solver, or theorem prover eventually to prove properties of production processes like stability automatically.

Acknowledgments We are grateful to the anonymous reviewers for their valuable comments. 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).

References

1. Argoneto P, Perrone G, Renna P, Lo Nigro G, Bruccoleri M, Noto La Diega S (2008) Production planning in production networks: models for medium and short-term planning. Springer, Berlin
2. Armbruster D, de Beer C, Freitag M, Jagalski T, Ringhofer C (2006) Autonomous control of production networks using a pheromone approach. Physica A 363(1):104–114
3. Armbruster D, Kaneko K, Mikhailov AS (eds) (2006) Networks of interacting machines: production organization in complex

industrial systems and biological cells. World Scientific, Singapore

4. Corradini A, Ehrig H, Heckel R, Löwe M, Montanari U, Rossi F (1997) Algebraic approaches to graph transformation part I: basic concepts and double pushout approach. In: Rozenberg G (eds) Handbook of graph grammars and computing by graph transformation, vol 1: foundations. World Scientific, Singapore, pp 163–245
5. Dashkovskiy S, Görges M, Naujok L (2009) Local input to state stability of production networks. In: Proceedings of 2nd international conference on dynamics in logistics (LDIC 2009). Springer, Bremen, pp 79–89
6. Dashkovskiy S, Rüffer BS (2010) Local ISS of large-scale interconnections and estimates for stability regions. Syst Control Lett 59(3–4):241–247
7. Ehrig H, Ehrig K, Prange U, Taentzer G (eds) (2006) Fundamentals of algebraic graph transformation. Springer, Berlin
8. Geiß R, Kroll M (2008) GrGen:NET A fast, expressive, and general purpose graph rewrite tool. In: Schürr A, Nagl M, Zündorf A (eds) Proceedings of 3rd international workshop on applications of graph transformation with industrial relevance (AGTIVE '07). Lecture Notes in Computer Science, vol 5088. Springer, pp 568–569
9. Helbing D, Lämmer S Supply and production networks: from the bullwhip effect to business cycles. In: Armbruster et al. [3], pp 33–66
10. Hölscher K, Klempien-Hinrichs R, Knirsch P, Kreowski HJ, Kuske S (2007) Autonomous units: basic concepts and semantic foundation. In: Hülsmann M, Windt K (eds) Understanding autonomous cooperation and control in logistics—the impact on management, information and communication and material flow. Springer, Berlin, pp 103–120
11. Hölscher K, Kreowski HJ, Kuske S (2009) Autonomous units to model interacting sequential and parallel processes. Fundamenta Informaticae 92(3):233–257
12. Hülsmann M, Windt K (eds) (2007) Understanding autonomous cooperation and control in logistics. Springer, Berlin
13. Kreowski HJ, Kuske S (2010) Autonomous units and their semantics—the concurrent case. In: Engels G, Lewerentz C, Schäfer W, Westfechtel B (eds) Graph transformations and model-driven engineering. Lecture notes in computer science, vol 5765, pp 102–120
14. Scholz-Reiter B, Mehrsai A, Görges M (2009) Handling the dynamics in logistics—adoption of dynamic behavior and reduction of dynamic effects. Asian Int J Sci Technol Prod Manufact Eng (AIJSTPME) 2(3):99–110
15. Sontag E (2007) Input to state stability: basic concepts and results. In: Nistri P, Stefani G (eds) Nonlinear and optimal control theory. Springer, Berlin, pp 163–220
16. Wiendahl HP, Lutz S (2002) Production in networks. Ann CIRP Manufact Technol 51(2):1–14