

Experimental Symbolic Analysis of Net Systems

Hartmann J. Genrich

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven, D-53754 Sankt Augustin, Germany
hartmann.genrich@gmd.de

1 Introduction

This note addresses certain aspects of using Petri nets in the analysis of computerized systems: engineering systems, manufacturing systems, workflow systems, communication systems – whatever Petri net models are used for in practice. Any such system when adequately presented by a Petri net I call a *net system*. And throughout this note *Petri nets* means *higher-level* ones like those represented by *PrT-nets* in theory and *Design/CPN* in practice.

Technically, the note contains little new for those familiar with Petri nets and with the notions of *S-invariants* and *T-invariants* in particular, the solutions of homogeneous linear equation systems based on the incidence matrix of a Petri net. Rather, I wish to give a twist to the understanding and usage of invariants. I hope to change our focus from S-invariants to those *state variables* or *S-quantities* that may or may not be *constant*, and from T-invariants to those *changes* that may or may not be *cyclic* and to the *process variables* or *T-quantities* that can be associated with them.

Form the very beginning when discovering the PrT-nets Kurt Lautenbach and I were intrigued by the idea of operating upon the higher-level representation of a net system *symbolically* – in the fashion of symbolic logic or computer algebra.[1] I was always convinced that this should be simple and straightforward. In recent years, in order to find out how little, or much, was actually needed I started to conduct some experiments. To support them I extended some Standard ML code I had written earlier (on top of Design/CPN's interface) to extract the incidence matrix of an executable net model.

PrT-nets introduced the notion of an *individual – value* and *variable* – into the theory of Petri nets. Hence certain basic transformations of the expressions annotating a PrT-net come in quite naturally; in particular, consistent substitution of individual variables. (For a complete set of *equivalence transformations* for PrT-nets, see [2]. Only very few of those, however, play a role in the sequel.). To do the S-invariant analysis symbolically I took a step further. I identified *S-quantities* that are connected to the markings of a Petri net and *T-quantities* that are connected to the changes and started to study their properties and relationships in a playful, computer-algebraic fashion.

Over the time this SML code grew into a kind of package whose kernel consists of two basic elements:

- the application of a combination of transformation rules to a combination of expressions and,
- the reduction of the result by simplifications some of which are built-in and others can be declared ad-hoc.

The package might be a software engineer’s nightmare but it allowed me to perform some small yet non-trivial experiments – about the results of which and some conclusions I want to talk in this note. The style of the note remains narrative rather than technical. I use a simple CPN model of a production schema as a running example and demonstrate what I can find out using my ‘analysis package’.

2 The Example

The running toy example is shown in figure 1. It presents a simple production schema consisting of several nested loops; it is denoted by Π in the sequel. It has been extracted from a more complex net system in the field of chemical engineering – a recipe for batch process control [3] – but it could as well have been derived from the transaction processing in a database system or the processing of work objects in a workflow system or, just some piece of software.

The example has been built and run by means of Design/CPN [6] and hence it follows its syntactical conventions. The main declarations of its datatypes (colour sets) and other non-logical symbols are shown in table 1. In the symbolic analysis of Petri nets that we demonstrate in this note they play a minor role only.

For the time being treat the shaded transitions as auxiliary (comment); they indicate how the system may interact with other systems or its environment. Also assume that the pairs of places s_1 and s'_1 respectively s_3 and s'_3 are fused together. Then table 2 presents the net system Π in an equivalent form, namely by its *incidence matrix* C . It has a row C_s for each place s and a column C^t for each transition t . Each entry C_s^t is the combination of arc expressions between s and t , inputs to t distinguished from outputs by a negative sign.

The incidence matrix is the basis for studying the net system Π in merely symbolical terms. Table 3 shows what the package extracts from the CPN model in simulation state – when, in general, all submodels are linked properly and it is about to be executed.

3 Combinations of Pieces

In physical systems, quantities like displacement, force, energy, charge, temperature, etc. are real-valued, continuous functions of time with sufficiently high derivatives. For net systems we do not expect that much structure. However, a minimum of numbership should be found with the quantities in net systems, too. That is, we call some observable that we associate with a net

system, a *quantity* only if we can *add* its values to each other and *multiply* them by *scalars*.¹ These scalars may be integer, rational or real numbers; they form a ring R .

Structures whose elements can be added to each other and multiplied by numbers are called *modules*. Every marking of a place, for example, is an element of a particular module; it is an integer combination of structured tokens, elements of the datatype (colour set) associated with that place.

For the rest of this paper all quantities have values that are combinations of some *pieces*. These pieces can be viewed as different units for expressing a value of the quantity – like the various coins and notes of a currency. Mere numbers themselves as values then are combinations of the unit $()$.

- Let D be a set respectively a datatype and R a ring of scalar coefficients respectively the datatype *int* or *real*.² A combination of pieces from D with coefficients from R is a mapping $\kappa : D \rightarrow R$ such that for finitely many pieces d only the coefficient $\kappa(d)$ is different from zero.
- $\mathcal{L}(D, R)$ denotes the set of combinations of pieces from D with coefficients from R .

I use the following notation:

- Listing of pieces:

$$3 \cdot p - 2 \cdot q + r := \{d \mapsto \begin{cases} 3 : d = p \\ -2 : d = q \\ 1 : d = r \\ 0 : \textit{otherwise} \end{cases} \mid d \in D\}$$

- Null:

$$\mathbf{0} := \{d \mapsto 0 \mid d \in D\}$$

- Boolean coefficients:

$$[\top] \cdot d := d \quad [\perp] \cdot d := \mathbf{0} \quad [b_1, \dots, b_k] \cdot d := [b_1] \cdot \dots \cdot [b_k] \cdot d$$

- Combination of combinations:

$$(\kappa_1 + \kappa_2)(d) := \kappa_1(d) + \kappa_2(d) \quad (\textit{addition})$$

$$(k \cdot \kappa)(d) := k \cdot \kappa(d) \quad (\textit{multiplication with scalars})$$

Let D and D' be two sets of pieces and R a set of scalars. A *linear* mapping m from $\mathcal{L}(D, R)$ into $\mathcal{L}(D', R)$ I call a *distribution*; it can be represented as a function $\delta : D \rightarrow \mathcal{L}(D', R)$ that 'distributes' every piece of D over a combination of pieces of D' . Because of the linearity of m ,

- $m(\kappa) = \sum_{d \in D} \kappa(d) \cdot \delta(d)$ (note that $\kappa(d) \neq 0$ for finitely many d only)

¹An observable is an operator that assigns to every state or every process of the net system exactly one value. A proposition is not a quantity in our sense since its truth values cannot be added or multiplied by numbers.

²Actually, neither the datatype *int* of integers nor the datatype *real* of floating point numbers form a ring.

- *Example:* Let m be defined by $\delta = \{(x, y, c) \mapsto c \cdot x - y\}$. Then
 $m(2 \cdot (A, B, 3) + 3 \cdot (A, C, -1)) = 2 \cdot (3 \cdot A - B) + 3 \cdot (-A - C) = 3 \cdot A - 2 \cdot B - 3 \cdot C$

A higher-level Petri net is full of expressions for combinations of pieces. The markings of a place s and the arc inscriptions of the adjacent arcs are expressions all denoting combinations of pieces of the *colour set* of s . The *guard* of a transition t is not a combination by itself but is a boolean coefficient that is common to all inscriptions of the arcs adjacent to t .

What about combinations of expressions. If we just view them as pieces of datatype *string*, we may combine them arbitrarily. However, in order for the addition of two expressions u and w to make sense, two conditions must be satisfied.

- All values of u and w must belong to the same $\mathcal{L}(D, R)$.
- u and w must have the same *scope* of consistent substitution of individual variables.

The first condition is satisfied for any two expressions in the reach of the same place; the second one is satisfied for any two expressions in the reach of the same transition. In general, however, two arc expressions at the same place or at the same transition do not satisfy both conditions. Before they may be combined, they must be transformed – at a place by substitutions that get rid of all individual variables (*bindings* in CPN terminology), at a transition by distributions that have the same range.

Let u be (an expression for) a combination of pieces and τ a transformation rule (substitution or distribution). The application of τ to u is denoted by $u \odot \tau$. The two conditions above for combining expressions guarantee that

$$(u + w) \odot \tau = u \odot \tau + w \odot \tau$$

What about combining two transformation rules σ and τ such that for every expression u

$$u \odot (\sigma + \tau) = u \odot \sigma + u \odot \tau$$

Here the conditions are

- Either both σ and τ are distributions of the same type $D \rightarrow \mathcal{L}(D', R)$ where D is the colour set of the place to which u belongs
- or, both σ and τ are substitutions of the individual variables (CPN variables) of the transition to which u belongs.

4 Changes

To get acquainted with the example, let us now consider what happens with one of the tokens of the initial marking of place s_0 . It is the batch $(X, 3)$, 'produce 3 units of substance X', that on place s_0 appears as an incoming order.

The processing of the batch begins with an occurrence of transition t_1 for $\{c, r, x \leftarrow 3, [D1, D3], X\}$ which takes the batch $(X, 3)$ from the *incoming*

orders, s_0 , puts a copy of the recipe for X , $r = R(x) = [D1, D3]$, on place s_6 , removes the multi-set of devices listed in r , $(dvc\ r) = 1 \cdot D1 + 1 \cdot D3$, from the *pool of available devices*, s_4 , puts an empty container for substance X on *outgoing results*, s_5 , and puts the status vector for the batch on place s_1 where the production cycle begins.

All this happens, at the chosen level of abstraction, as a single indivisible *event*. However, it may occur *only if* the binding for c and r satisfies the list of constraints stated in the *guard* of t_1 , $[c > 0, r = Rx, r \neq []]$, that is if the count c and the recipe r are meaningful and non-trivial.

The result of applying $\{c, r, x \leftrightarrow 3, [D1, D3], X\}$ to t_1 is shown in table 4(a). Note that in our symbolic approach no evaluation takes place. Rather, individual variables are replaced by terms and the resulting expressions may be transformed by some simple equivalence transformations as reduction rules.

Through transition t_1 the kernel of the processing of the batch has been entered. It consists of two loops. The inner one, determined by transitions t_2 and t_3 , follows the recipe step by step until one unit of substance x is done. A single production step is identified, for the sake of simplicity, with utilizing a particular device (resource). In the outer loop, when one unit is done, t_4 puts it on place s_3 for being accumulated at s_5 . And, as long as more units have to be produced yet, t_4 also restores the recipe and decreases the count in the status vector, and enters another round.

Transition t_5 accumulates the single units of a substance in the corresponding container that was put on place s_5 by transition t_1 . If the whole batch is done, t_5 also returns the set of devices that were reserved for the batch in the beginning and removes the copy of the recipe used for substance X as done – thus completing the processing of a single batch.

Let us now summarize the whole processing of batch $(X, 3)$ by adding up, for each transition, all the bindings for which this transition occurs. In doing so we deliberately disregard any order in which the transitions occur for the listed bindings, and whether the transition occurrences are enabled or not. The result is shown in table 4(b). It is a *T-vector* (vector whose index set is the set T of transitions) whose entries are combinations of bindings. We call such a T-vector a *change* in the sequel.

The rows C_s of the incidence matrix C (see table 2) are T-vectors, too, whose entries are combinations of arc expressions. If we apply, for each transition t , the combinations of bindings of the change in table 4(b) to the corresponding arc expressions and then add-up all transformed columns, we get the net effect that the change has on the markings of each place. Table 5 shows what the package delivers – before and after some built-in simplifications for list expression are applied.

Formally, the *effect* of a change P is the *matrix product* $C \cdot P$ between the incidence matrix C and the one-column matrix (T-vector) P . The result is exactly what we expect. The batch $(X, 3)$ is transformed from an unprocessed order into a container with 3 units of substance X . All internal places are unaffected. Their markings are restored to the initial ones. On the inner part of the system – transitions t_2, t_3, t_4 – the change is *cyclic*.

5 S-Quantities

Let Σ be a net system. A state variable is an operator that for some module $\mathcal{D} = \mathcal{L}(D, R)$, assigns to every conceivable marking of Σ an element of \mathcal{D} . If a state variable Q is linear it can be represented by an S-vector of distributions: for every place s there is a distribution $Q_s : D_s \rightarrow \mathcal{D}$ that assign to every token of the colour set of s , D_s , the corresponding value of Q . Each Q_s determines the form in which the variable Q appears on place s . I call a such linear state variable Q of a net system an *S-quantity*.

Table 6 shows three S-vectors of distributions. They denote different S-quantities of our production schema Π . The S-quantity *load*, for example, gives the work-load of Π at an arbitrary marking in terms of the number of units of each substance. It shows how the work objects (batches) are spread over the places at the various stages of the production process, i.e. the various forms in which *load* appears in Π .

$$\begin{aligned}
 \Delta load = & \Delta s_0 \odot \{(x, c) \mapsto c \cdot x\} && \text{waiting batches} \\
 & + \Delta s_1 \odot \{(x, c, -, -) \mapsto c \cdot x\} && \text{batch between stations} \\
 & + \Delta s_2 \odot \{(x, c, -, -, -) \mapsto c \cdot x\} && \text{batch at a station} \\
 & + \Delta s_3 \odot \{(x, -, -) \mapsto 1 \cdot x\} && \text{piece done} \\
 & + \Delta s_5 \odot \{(x, c', -) \mapsto c' \cdot x\} && \text{pieces stored}
 \end{aligned}$$

Each entry of the vector specifies the combination of pieces that the tokens of the corresponding place contribute to the current work-load. For the initial marking shown in figure 2, the value of *load* is

$$\begin{aligned}
 & (1 \cdot (X, 3) + 1 \cdot (Y, 2)) \odot \{(x, c) \mapsto c \cdot x\} \\
 + & \mathbf{0} \odot \{(x, c, -, -) \mapsto c \cdot x\} \\
 + & \mathbf{0} \odot \{(x, c, -, -, -) \mapsto c \cdot x\} \\
 + & \mathbf{0} \odot \{(x, -, -) \mapsto 1 \cdot x\} \\
 + & AvlDvc^0 \odot \{- \mapsto \mathbf{0}\} \\
 + & \mathbf{0} \odot \{(x, c) \mapsto c \cdot x\} \\
 + & Recipes^0 \odot \{- \mapsto \mathbf{0}\} && = 3 \cdot X + 2 \cdot Y
 \end{aligned}$$

Table 7 shows the result of applying every entry of the S-vector *load* to the entries of the corresponding row of the incidence matrix (table 2). The bottom row, *grad (gradient)*, is the combination of the rows for places s_0, \dots, s_6 . It gives for each transition t the effect that the corresponding elementary changes Δt have to the S-quantity *load*. We call it the *defect of load* at transition t .

All entries of *grad load* are equal to zero except for transition t_4 . The batch count c , however, remains greater than zero for all meaningful processes of the production schema (see below). And for $c > 0$, the defect of *load* at t_4 reduces to $\mathbf{0}$ as well. Hence *grad load* = $\mathbf{0}$ for all meaningful processes; the S-quantity *load* is constant.

By the same procedure the other two S-quantities of table 6, *recipe* and *count*, can be shown to be constant or, as the terminology is in net theory, are

S-invariants. The quantity *count* is an auxiliary S-quantity; that its defect is $\mathbf{0}$ at every transition proves that a batch count which is non-negative initially may never become negative.

The different forms of the quantity *recipe* show how a recipe is decomposed and re-composed during the production process. There is a twist, however: the coefficient of the combination on place s_6 is negative. Hence we can split *recipe* into two non-trivial quantities with non-negative coefficients, $recipe^{\geq 0}$ and $recipe^{\leq 0}$, such that

$$recipe = recipe^{\geq 0} - recipe^{\leq 0}$$

Since *recipe* is constant (see table 8) and its value for the initial marking is $\mathbf{0}$, we get

$$recipe^{\geq 0} = recipe^{\leq 0}$$

The quantity $recipe^{\leq 0}$ is a *copy* of $recipe^{\geq 0}$: every piece (x, r) on place s_6 is a copy of the recipe of a batch being processed.

The *flow* of S-quantities in net systems – between subsystems determined by disjoint sets of places – has two aspects[5]:

- *flux* or *exchange* between subsystems: *load*, for example, is exchanged between the outer part and the inner part of the system through transitions t_1 and t_4 ;
- *influence* between subsystems: $recipe^{\geq 0}$ and $recipe^{\leq 0}$ are copies maintained by transitions t_1 and t_5 .

6 T-Quantities

Not all interesting quantities that we associate with a net system may be functions of markings. There are also quantities that are functions of changes. For example, the production costs for a batch in a production schema, the total time during which a resource is idle respectively occupied, or the *work* performed by a component or subsystem during a particular process – all those quantities cannot be expressed, in general, as a function of markings but rather as functions of changes.

A linear operator that for some module $\mathcal{D} = \mathcal{L}(D, R)$ assigns an element of \mathcal{D} to every change in a net system Σ is called a *T-quantity* of Σ .

Assume we want to express the production costs for batches of our production schema Π , in terms of entire units of some currency. We assign to transition t_2 some costs involved using device d during the production of a single unit of substance x , and to transition t_5 some costs for the transport and storage of a single unit of x . Other costs are ignored. The corresponding cost functions are $K_2(x, d)$ and $K_5(x)$, respectively. The T-quantity *cost* is then represented by a T-vector as shown in table 9.

The entries of *cost* for each transition t_i are expressions for integer combinations of the unit $()$ whose free variables belong to the scope of t_i . For every binding of t_i they denote an integer value. For K_2 and K_5 as given as in table 9, the costs for processing batch $(X, 3)$, for example, are

$$3 \cdot (K_2(X, D1) \cdot ()) + K_2(X, D3) \cdot () + K_5(X) \cdot () = 27 \cdot ()$$

Also the T-vector $\text{grad } load$ of table 9 denotes a T-quantity, namely the effect that any change has on $load$. In general:

The T-vector of defects (the gradient) $\text{grad } q$ of an S-quantity q is a T-quantity.

The value of a T-quantity, k , for some change u is independent of the order in which the Δt of u may occur in an actual process. It is just the dot product between k and u .

Now assume there are two different changes, u_1 and u_2 , both leading from the same marking M^1 to the same marking M^2 . Then we may ask whether the values of k along u_1 and u_2 are the same or not. If it is the case that for any pair of markings (M^1, M^2) the value of k is the same along all changes that connect M^1 to M^2 , we call k *path-indifferent* ('path' in the reachability graph). In physical systems, work is a 'path-indifferent' quantity iff the forces are conservative – no friction, for example.

A T-quantity k is path-indifferent iff its value along any cyclic change is $\mathbf{0}$. And it is a linear-algebraical fact that

Theorem: For a T-quantity k there is an S-quantity \hat{k} such that $k = \text{grad } \hat{k}$ – a *potential function* for k – iff k is path-indifferent.

The costs for the processing of any set of batches as given by the T-quantity $cost$ are independent of the various ways in which these processes may be intertwined. And since there are no cyclic processes in the production schema Π as long it is not completed by an environment, $cost$ is path-indifferent. A potential function $COST$ is shown in table 10. The proof that

$$cost = -\text{grad } COST$$

eventually convinced me of the usefulness of that little symbolic analysis package; I could not have done without it.

7 An Interpretation

Talking about the *gradient* of an S-quantity and the *potential function* of a T-quantity suggests that I see a certain similarity with the dynamic description of physical systems. Here is the essence of what I have in mind. It seems to me that the interpretation that I give below to the incidence matrix may help to understand why net systems are so successful as discrete presentations of dynamic systems. However, for the time being I want to stress that by using some notation of calculus I do not claim more than similarity.

S-Quantities:

The state coordinates of a net system are its places. The elementary (the smallest yet not infinitesimal) changes are the transitions. The incidence matrix is the matrix of partial derivatives of the coordinates with respect to the changes:

$$\frac{\partial s}{\partial t} := C_s^t$$

A binding of the individual variables of a transition t gives an event at t . I denote the binding as Δt . Then I get the effect of that event on a place s as

$$\Delta s = C_s^t \odot \Delta t =: \frac{\partial s}{\partial t} \Delta t$$

The components (Q_1, \dots, Q_n) of the S-vector representing an S-quantity Q are the partial derivatives of Q with respect to the coordinates s_i :

$$Q_i =: \frac{\partial Q}{\partial s_i}, \quad Q_i(C_i^t) = \frac{\partial s_i}{\partial t} \odot \frac{\partial Q}{\partial s_i} =: \frac{\partial Q}{\partial s_i} \frac{\partial s_i}{\partial t}, \quad \frac{\partial Q}{\partial t} := \sum_{i=1}^n \frac{\partial Q}{\partial s_i} \frac{\partial s_i}{\partial t}$$

For the effect of an event Δt on the S-quantity Q we now get

$$\begin{aligned} \Delta Q &= \sum_{i=1}^n Q_i(C_i^t \odot \Delta t) = \sum_{i=1}^n (C_i^t \odot \Delta t) \odot Q_i \stackrel{!}{=} \sum_{i=1}^n (C_i^t \odot Q_i) \odot \Delta t \\ &= \sum_{i=1}^n \left(\frac{\partial s_i}{\partial t} \odot \frac{\partial Q}{\partial s_i} \right) \Delta t = \left(\sum_{i=1}^n \frac{\partial Q}{\partial s_i} \frac{\partial s_i}{\partial t} \right) \Delta t = \frac{\partial Q}{\partial t} \Delta t \end{aligned}$$

Note that the essential step in this derivation depends on the assumption that the distributions Q_i and the substitution Δt – as transformations of expressions – commute. This is always the case as long as the distributions do not introduce additional free individual variables (which they cannot do if they denote proper functions).

Hence the heart of the S-invariance technique, the dot product between the column C^t of the incidence matrix and the S-vector of components (Q_1, \dots, Q_n) of an S-quantity Q yields the partial derivative of Q with respect to the transition t .

$$\frac{\partial Q}{\partial t} = \sum_{i=1}^n \frac{\partial Q}{\partial s_i} \frac{\partial s_i}{\partial t} = (C^t)^\top \cdot Q$$

Result (S-Quantities):

- The matrix product of the transposed incidence matrix C^\top with the S-quantity $Q \cong (Q_1, \dots, Q_n)$ yields the *gradient* of Q .

$$\text{grad } Q = \left(\frac{\partial Q}{\partial t_1}, \dots, \frac{\partial Q}{\partial t_m} \right) = C^\top \cdot Q$$

- The components (Q_1, \dots, Q_n) are the different *forms* in which Q appears on the places s_i (like energy appears in different forms in physical systems).

$$\Delta Q = \frac{\partial Q}{\partial s_1} \Delta s_1 + \dots + \frac{\partial Q}{\partial s_n} \Delta s_n$$

T-Quantities, Fields:

Quantities like *work* or *cost* assign a value to every process rather than every state. In a net system a process P is a partially ordered set of event

occurrences. Let $\{\Delta t_i\}$ denote an enumeration of the occurrences of P that is consistent with the ordering (a 'firing sequence').

$$\Delta t_i < \Delta t_k \Rightarrow i < k$$

Let $\mathcal{F} = (\dots, \mathcal{F}_t, \dots)$ be a T-vector of expressions denoting values of the same domain $\mathcal{D} = \mathcal{L}(D, R)$. I call it a *field*; it represents a T-quantity F that assigns to every process $P \cong \{\Delta t_i\}$ a value by virtue of the 'line integral' of \mathcal{F} along P .

$$F(P) = \sum_i \mathcal{F}_t \odot \Delta t_i =: \int_P \mathcal{F} dt$$

The gradient of an S-quantity Q is such a field. Its line integral along a process P is the accumulated effect that P has on Q . If, conversely, for a field \mathcal{F} there is an S-quantity U such that

$$\mathcal{F} = -\text{grad } U,$$

U is called a *potential function* of \mathcal{F} . If it exists it is unique up to an additive constant.

Result:

For net systems as well as for physical systems the following holds:

Theorem: For a field \mathcal{F} there exists a potential function U iff for any two processes P_1 and P_2 connecting the same states the line integral of \mathcal{F} has the same value,

$$M_1 \begin{matrix} \xrightarrow{P_1} \\ \xrightarrow{P_2} \end{matrix} M_2 \Rightarrow \int_{P_1} \mathcal{F} dt = \int_{P_2} \mathcal{F} dt$$

or, equivalently, iff for any *cyclic* process (T-invariant, reproduction component) R the value of the line integral is zero.

$$C \cdot R = 0 \Rightarrow \oint_R \mathcal{F} dt = 0$$

8 Future Work, Conclusions

The package so far is a nice gadget. It helped me to test some ideas on symbolic manipulation of higher-level nets. It doesn't need much theoretical foundation but uses simple transformations and simplifications of expressions in a straightforward fashion.

I have a lot of ideas how to improve and extend it. For example, it already provides the basic ingredients to do symbolic reachability analysis. Only one more technical device is needed, namely *individual parameters* to allow transition occurrences with partial bindings. They consist of individual variables with a suffix of the form \tilde{i} for the i^{th} partial occurrence. Their scope is global.

For example, assume that t_1 is to occur for substance X and substance Y but count and recipe are left unspecified. Let the current value of a unique

counter for partial occurrences be 5. The resulting *parametrized marking* difference is shown in table 4(c).

Note that in order to add the marking difference to some given, ordinary or parametrized marking, one has to distribute the guard as a common boolean coefficient to all arc expressions of t_1 .

Before parametrized markings and other features may be added, the package has to be thoroughly and professionally re-designed and re-implemented. Its tolerance for extensions has reached its limits.

For a re-implementation there are mainly two options. Either, continue using *Standard ML* or, switch to a computer algebra system like *Mathematica*. The SML option makes it part of the Design/CPN tool but requires implementing of a good deal of features that any good computer algebra package already has. The Mathematica option makes it an independent tool that requires a good link to Design/CPN.

I don't know yet. If you have an opinion, let me know of it.

References

- [1] Genrich, H.J.; Lautenbach, K.: *System modelling with high-level Petri nets*. Theoretical Computer Science 13 (1981) 109–136
- [2] Genrich, H.J.: *Equivalence Transformations of PrT-Nets*. Advances in Petri Nets 1989 (G. Rozenberg, Ed.), LNCS 424. Berlin : Springer (1989) 179–208
- [3] Genrich, H.J.; Hanisch, H.-M.: *Modelling and Analysis of Recipes*. Workshop on Analysis and Design of Event-Driven Operations in Process Systems, Imperial College, April 1995.
- [4] Jensen, K.: *Coloured Petri Nets and the Invariant Method*. Theoretical Computer Science 14 (1981) 317–336
- [5] Petri, C.A.: *Grundsätzliches zur Beschreibung diskreter Prozesse*. 3. Colloquium über Automatentheorie (W. Händler, E. Peschl, H. Unger, Hrsg.). Basel, Stuttgart : Birkhäuser (1967) 121–140
- [6] Design/CPN reference manual. Version 3.00. Aarhus : DAIMI, Aarhus University (1996)

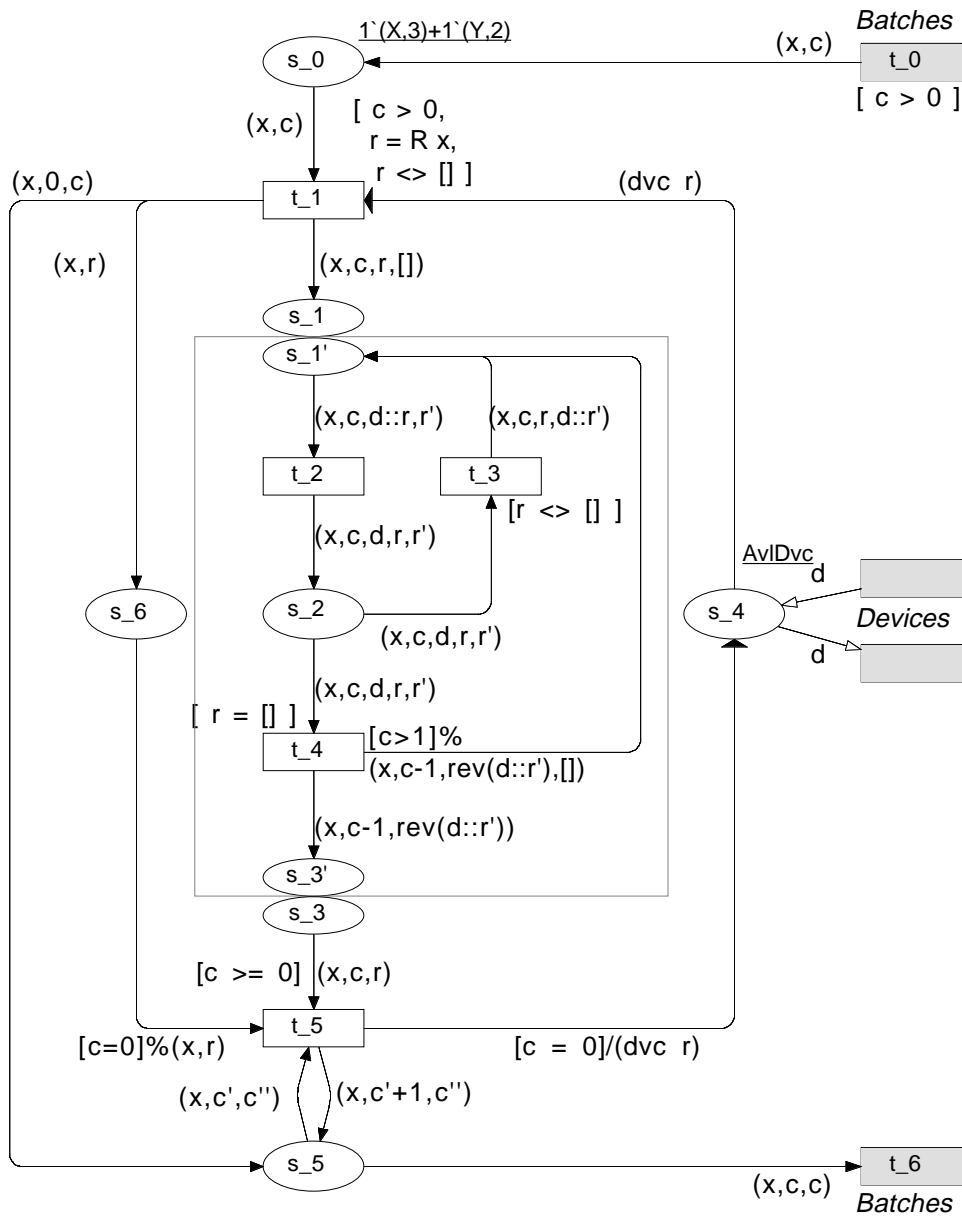


Figure 1: Net System – Production Schema

```

color Sbs = union X + Y + Z;
color Cnt = int;
color Bch  = product Sbs * Cnt;

color Dvc = union D1 + D2 + D3 declare ms;
color Stp = Dvc;
color Rcp = list Stp;

fun dvc r = (list_to_ms o remdupl) r;

val p_Rec = ref (fn X => [D1,D3] | Y => [D1,D2,D3,D1] | _ => []);
fun R d = !p_Rec d;

val AvlDvc = 2 * Dvc;

```

Table 1: Global Declarations for the Production Schema

| | t_1 | t_2 | t_3 | t_4 | t_5 |
|----------|--|---------------------|---------------------|--|---------------------------------------|
| Γ | $[c > 0,$ $r = R x$ $r \neq []]$ | | $[r \neq []]$ | $[r = []]$ | $[c \geq 0]$ |
| s_0 | $-(x, c)$ | | | | |
| s_1 | $(x, c, r, [])$ | $-(x, c, d::r, r')$ | $(x, c, r, d::r')$ | $[c > 1].$ $(x, c-1, rev(d::r'), [])$ | |
| s_2 | | (x, c, d, r, r') | $-(x, c, d, r, r')$ | $-(x, c, d, r, r')$ | |
| s_3 | | | | $(x, c-1, rev(d::r'))$ | $-(x, c, r)$ |
| s_4 | $-(dvc r)$ | | | | $[c=0].(dvc r)$ |
| s_5 | $(x, 0, c)$ | | | | $-(x, c', c'')$ $+ (x, c'+1, c'')$ |
| s_6 | (x, r) | | | | $-[c=0].(x, r)$ |

$\Gamma = \textit{Guard}$: boolean coefficient common to all entries of the respective column

Table 2: Incidence Matrix of the Production Schema with *Guards*

| | |
|---|---|
| <pre>(show_Mat_S (!IncMatrix); show_FusionSets (); show_FusionReps ());</pre> | <pre>t_1: c,r,x Guard: [c>0,r=R x,r<>[]] Pre: s_4: (dvc r) s_0: (x,c) Post: s_1: (x,c,r,[]) s_5: (x,0,c) s_6: (x,r) ----- t_2: c,d,r,r',x Guard: Pre: s_1': (x,c,d::r,r') Post: s_2: (x,c,d,r,r') ----- t_3: c,d,r,r',x Guard: [r<>[]] Pre: s_2: (x,c,d,r,r') Post: s_1': (x,c,r,d::r') ----- t_4: c,d,r,r',x Guard: [r=[]] Pre: s_2: (x,c,d,r,r') Post: s_3': (x,c-1,rev(d::r')) s_1': [c>1]'(x,c-1,rev(d::r')),[] ----- t_5: c,c',c'',r,x Guard: [c>=0] Pre: s_3: (x,c,r) s_6: [c=0]'(x,r) s_5: (x,c',c'') Post: s_5: (x,c'+1,c'') s_4: [c=0]'(dvc r) ===== FG_S1: s_1,s_1' FG_S3: s_3,s_3' ===== FG_S1: s_1 FG_S3: s_3 =====</pre> |
|---|---|

Table 3: Incidence Matrix Extracted by the Analysis Package

| | t_1 |
|----------|---|
| Γ | $[3 > 0,$ $[D1, D3] = R X,$ $[D1, D3] \neq []]$ |
| s_0 | $-(X, 3)$ |
| s_1 | $(X, 3, [D1, D3], [])$ |
| s_2 | |
| s_3 | |
| s_4 | $-(\text{dvc } [D1, D3])$ |
| s_5 | $(X, 0, 3)$ |
| s_6 | $(X, [D1, D3])$ |

$t_1 \odot \{c, r, x \leftarrow 3, [D1, D3], X\}$

(a)

| | |
|-------|--|
| t_1 | $1 \cdot \{x, c, r \leftarrow X, 3, [D1, D3]\}$ |
| t_2 | $1 \cdot \{x, c, d, r, r' \leftarrow X, 3, D1, [D3], []\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 3, D3, [], [D1]\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 2, D1, [D3], []\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 2, D3, [], [D1]\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 1, D1, [D3], []\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 1, D3, [], [D1]\}$ |
| t_3 | $1 \cdot \{x, c, d, r, r' \leftarrow X, 3, D1, [D3], []\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 2, D1, [D3], []\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 1, D1, [D3], []\}$ |
| t_4 | $1 \cdot \{x, c, d, r, r' \leftarrow X, 3, D3, [], [D1]\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 2, D3, [], [D1]\}$ + $1 \cdot \{x, c, d, r, r' \leftarrow X, 1, D3, [], [D1]\}$ |
| t_5 | $1 \cdot \{x, c, c', c'', r \leftarrow X, 2, 0, 3, [D1, D3]\}$ + $1 \cdot \{x, c, c', c'', r \leftarrow X, 1, 1, 3, [D1, D3]\}$ + $1 \cdot \{x, c, c', c'', r \leftarrow X, 0, 2, 3, [D1, D3]\}$ |

Summary of processing of batch $(X, 3)$

(b)

| | |
|-------|---|
| s_0 | $-[c^5 > 0, r^5 = R X, r^5 \neq []] \cdot (X, c^5)$ + $-[c^6 > 0, r^6 = R Y, r^6 \neq []] \cdot (Y, c^6)$ |
| s_1 | $[c^5 > 0, r^5 = R X, r^5 \neq []] \cdot (X, c^5, r^5, [])$ + $[c^6 > 0, r^6 = R Y, r^6 \neq []] \cdot (Y, c^6, r^6, [])$ |
| s_2 | |
| s_3 | |
| s_4 | $-[c^5 > 0, r^5 = R X, r^5 \neq []] \cdot (\text{dvc } r^5)$ + $-[c^6 > 0, r^6 = R Y, r^6 \neq []] \cdot (\text{dvc } r^6)$ |
| s_5 | $[c^5 > 0, r^5 = R X, r^5 \neq []] \cdot (X, 0, c^5)$ + $[c^6 > 0, r^6 = R Y, r^6 \neq []] \cdot (Y, 0, c^6)$ |
| s_6 | $[c^5 > 0, r^5 = R X, r^5 \neq []] \cdot (X, r^5)$ + $[c^6 > 0, r^6 = R Y, r^6 \neq []] \cdot (Y, r^6)$ |

Partial bindings and parameters: $\Delta M = t_1 \overset{5}{\odot} (\{x \leftarrow X\} + \{x \leftarrow Y\})$

(c)

Table 4: Changes as Combinations of Bindings

```

( val change_X_3 =
  [
    (t_1, [(1,"(x,c,r)" <-+ "(X,3,[D1,D3])" )]),
    (t_2, [(1,"(x,c,d,r,r')" <-+ "(X,3,D1,[D3],[ ])",... ]),
    ...
    (t_5, [ ..., (1,"(x,c,c',c'',r)" <-+ "(X,0,2,3,[D1,D3])" ) ] );
  EFFECT change_X_3; );

```

```

s_0: - (X,3)
s_1:   (X,3,[D1,D3],[ ])
      - (X,1,D1::[D3],[ ])
      - (X,1,D3::[ ],[D1])
      + (X,1,[D3],D1::[ ])
      - (X,2,D1::[D3],[ ])
      - (X,2,D3::[ ],[D1])
      + (X,2,[D3],D1::[ ])
      - (X,3,D1::[D3],[ ])
      - (X,3,D3::[ ],[D1])
      + (X,3,[D3],D1::[ ])
      + [1>1] '(X,1-1,rev(D3::[D1]),[ ])
      + [2>1] '(X,2-1,rev(D3::[D1]),[ ])
      + [3>1] '(X,3-1,rev(D3::[D1]),[ ])
s_3: - (X,0,[D1,D3])
      - (X,1,[D1,D3])
      - (X,2,[D1,D3])
      + (X,1-1,rev(D3::[D1]))
      + (X,2-1,rev(D3::[D1]))
      + (X,3-1,rev(D3::[D1]))
s_4: - (dvc[D1,D3])
      + [0=0] '(dvc[D1,D3])
      + [1=0] '(dvc[D1,D3])
      + [2=0] '(dvc[D1,D3])
s_5: - (X,1,3)
      - (X,2,3)
      + (X,0+1,3)
      + (X,1+1,3)
      + (X,2+1,3)
s_6:   (X,[D1,D3])
      - [0=0] '(X,[D1,D3])
      - [1=0] '(X,[D1,D3])
      - [2=0] '(X,[D1,D3])
=====

```

```

( setRed_coeff red_bool;
  setRed_piece (map_tuple [(2,red_int),(3,red_list),(4,red_list)]);
  REDUCE_IT ( ) );

```

```

s_0: - (X,3)
s_5:   (X,3,3)
=====

```

Table 5: Effect of a Change Before and After Simplification

| | <i>load</i> | <i>recipe</i> | <i>count</i> |
|-------|-------------------------------------|---|--|
| s_0 | $(x, c) \mapsto c \cdot x$ | $- \mapsto \mathbf{0}$ | $(-, c) \mapsto [c > 0] \cdot c \cdot ()$ |
| s_1 | $(x, c, -, -) \mapsto c \cdot x$ | $(x, -, r, r') \mapsto (x, (rev\ r')^{\wedge\wedge} r)$ | $(-, c, -, -) \mapsto [c > 0] \cdot c \cdot ()$ |
| s_2 | $(x, c, -, -, -) \mapsto c \cdot x$ | $(x, -, d, r, r') \mapsto (x, (rev\ d::r')^{\wedge\wedge} r)$ | $(-, c, -, -, -) \mapsto [c > 0] \cdot c \cdot ()$ |
| s_3 | $(x, -, -) \mapsto 1 \cdot x$ | $(x, c, r) \mapsto [c=0] \cdot (x, r)$ | $(-, c, -) \mapsto [c \geq 0] \cdot 1 \cdot ()$ |
| s_4 | $- \mapsto \mathbf{0}$ | $- \mapsto \mathbf{0}$ | $- \mapsto \mathbf{0}$ |
| s_5 | $(x, c) \mapsto c \cdot x$ | $- \mapsto \mathbf{0}$ | $(-, c') \mapsto c' \cdot ()$ |
| s_6 | $- \mapsto \mathbf{0}$ | $(x, r) \mapsto -(x, r)$ | $- \mapsto \mathbf{0}$ |

Table 6: Some S-Quantities of the Production Schema

| | t_1 | t_2 | t_3 | t_4 | t_5 |
|------------------|----------------------|--------------|---------------|--|--------------------------------|
| Γ | $[c > 0, r \neq []]$ | | $[r \neq []]$ | $[r = []]$ | $[c \geq 0]$ |
| s_0 | $-c \cdot x$ | | | | |
| s_1 | $c \cdot x$ | $-c \cdot x$ | $c \cdot x$ | $[c > 1] \cdot (c-1) \cdot x$ | |
| s_2 | | $c \cdot x$ | $-c \cdot x$ | $-c \cdot x$ | |
| s_3 | | | | x | $-x$ |
| s_4 | | | | | |
| s_5 | $0 \cdot x$ | | | | $-c' \cdot x + (c'+1) \cdot x$ |
| s_6 | | | | | |
| grad <i>load</i> | $\mathbf{0}$ | $\mathbf{0}$ | $\mathbf{0}$ | $x - c \cdot x$ $-[c > 1] \cdot x$ $+ [c > 1] \cdot c \cdot x$ | $\mathbf{0}$ |

Table 7: Incidence Matrix Transformed by S-Quantity *load*

```
DEFECT recipe;
```

```
t_1: - (x,r)
      + (x,(rev[])^^r)
t_2: (x,(rev d::r')^^r)
      - (x,(rev r')^^(d::r))
t_3: - (x,(rev d::r')^^r)
      + (x,(rev(d::r'))^^r)
t_4: [(c-1)=0] '(x,(rev(d::r'))
      - (x,(rev d::r')^^r)
      + [(c>1] '(x,(rev[])^^(rev(d::r'))))
=====
```

```
( setRed_coeff red_bool;
  setRed_piece (map_tuple [(2,red_list)]);
  REDUCE_IT () );
```

```
t_4: [(c=1] '(x,(rev r')^^[d])
      + [(c>1] '(x,(rev r')^^[d])
      - (x,(rev r')^^[d]^r)
=====
```

```
( Guards_to_IT [t_4];
  print_IT () );
```

```
t_4: [r=[],c=1] '(x,(rev r')^^[d])
      + [r=[],c>1] '(x,(rev r')^^[d])
      - [r=[]] '(x,(rev r')^^[d]^r)
=====
```

```
let val sub_r = subst [("r","[]")
      val rep_cf = replace [("c>1","1-[c=1]")] (* c>0! *)
in
setRed_coeff (red_bool o rep_cf o sub_r);
setRed_piece ((map_tuple [(2,red_list)]) o sub_r);
REDUCE_IT ()
end;
```

```
=====
```

Table 8: Defect of *recipe* With Simplification

| | t_1 | t_2 | t_3 | t_4 | t_5 |
|--------------|--|----------------------|----------------------|---|------------------------|
| T-Vectors | | | | | |
| type | $\{x, c, r\}$ | $\{x, c, d, r, r'\}$ | $\{x, c, d, r, r'\}$ | $\{x, c, d, r, r'\}$ | $\{x, c, c', c'', r\}$ |
| Γ | $[c > 0,$ $r = R x$ $r \neq []]$ | | $[r \neq []]$ | $[r = []]$ | $[c \geq 0]$ |
| T-Quantities | | | | | |
| cost | $\mathbf{0}$ | $K_2(x, d) \cdot ()$ | $\mathbf{0}$ | $\mathbf{0}$ | $K_5(x) \cdot ()$ |
| grad load | $\mathbf{0}$ | $\mathbf{0}$ | $\mathbf{0}$ | $\begin{matrix} x - c \cdot x \\ -[c > 1] \cdot x \\ +[c > 1] \cdot c \cdot x \end{matrix}$ | $\mathbf{0}$ |

$$K_2 = \begin{array}{c|ccc} & X & Y & Z \\ \hline D1 & 4 & 3 & 0 \\ D2 & 2 & 2 & 0 \\ D3 & 3 & 5 & 0 \\ D4 & 1 & 1 & 0 \end{array}, \quad K_5 = \begin{array}{c|ccc} & X & Y & Z \\ \hline & 2 & 1 & 0 \end{array}$$

Table 9: T-Quantities and Other T-Vectors

```

(
val COST = [
(s_0, [(1, "(x,c)" +-> "- c*(H(x, R x) + K_5(x))'(')"),
(s_1, [(1, "(x,c,r,r')" +-> "- (H(x,r)+(c-1)*H(x,(rev r')^^r))'(')"),
(s_2, [(1, "(x,c,d,r,r')" +-> "- (H(x,r)+(c-1)*H(x,(rev (d::r'))^^r))'(')"),
(s_3, [(1, "(x,c,r)" +-> "- 0'(')"),
(s_5, [(1, "(x,c',c'')" +-> "- (c''-c')*K_5(x)'(')"]);
defect COST;
Guards_to_IT [t_1,t_4];
setRed_coeff (map_arg "H" (map_tuple [(2,red_list)]));
REDUCE_IT ();
);

```

```

t_1: - [c>0,r<>[],r=R x]*H(x,r)*c'()
+ [c>0,r<>[],r=R x]*H(x,R x)*c'()
t_2: - H(x,r)'()
+ H(x,[d]^^r)'()
t_4: [r=[]]*(H(x,r))'()
+ [c>1,r=[]]*H(x,(rev r')^^[d])'()
- [r=[]]*(H(x,(rev r')^^[d]^^r))'()
+ [r=[]]*H(x,(rev r')^^[d]^^r)*c'()
- [c>1,r=[]]*H(x,(rev r')^^[d])*c'()
t_5: K_5(x)'()
=====

```

```

( setRed_coeff_at [t_1] (subst [{"r","R x"}]);
setRed_coeff_at [t_4] ((replace [{"c>1"},"1-[c=1]"]) o
subst [{"r","[]"}]);
setRed_coeff ((map_arg "H" (map_tuple [(2,red_list)])) o
red_bool);
REDUCE_IT ();
);

```

```

t_2: - H(x,r)'()
+ H(x,[d]^^r)'()
t_5: K_5(x)'()
=====

```

```

( define ("H", "(x,[]) +-> 0 |
(x,[d]) +-> K_2(x,d) |
(x,r^^r') +-> H(x,r)+H(x,r')" );
setRed_coeff (eval_defs ["H"]); REDUCE_IT ();
);

```

```

t_2: K_2(x,d)'()
t_5: K_5(x)'()
=====

```

Table 10: A Potential Function of *cost*