Parameterised General Refinement in the M-net Calculus

Raymond Devillers
Département d'Informatique, Université Libre de Bruxelles, Belgium
e-mail: rdevil@ulb.ac.be

Hanna Klaudel
Université Paris XII, IUT de Fontainebleau, France
e-mail: klaudel@lri.fr

Robert-C. Riemann
LRI, Université Paris-Sud, France and
Institut für Informatik, Universität Hildesheim, Germany,
e-mail: robert@lri.fr

Abstract

A general refinement (meta-)operation has recently been introduced into the algebra of M-nets, a high level class of labelled Petri nets. The present paper extends this operation to parameterised general refinement, such that parameters determined by the environment of refined transitions can be used in the refined part of the resulting M-net. The intended application is the modelling of procedures of parallel programming languages.

Keywords: Petri Net, Petri Box, M-Net, Refinement.

1 Introduction

The algebra of Petri boxes ([2, 1, 7, 8]) has been introduced with the aim of modelling the semantics of concurrent programming languages (and succeeded to do so up to some extent, see for instance [6]). In practical situations (and in particular when dealing with large value domains for program variables), this generally leads to huge (possibly infinite) nets, well defined mathematically but difficult to represent graphically and thus to grasp intuitively. In order to cope with this problem, higher level models have been introduced ([18, 19]), and in particular a fruitful class of so-called M-nets and M-boxes [4, 5] which nicely unfolds into low level boxes and thus allows to represent in a clear and compact way large (possibly infinite) systems.
The same operations should be defined on the M-net level than on the low level one, and in particular a refinement operation in order to model systems at different levels of abstractions. A first step in this direction has been presented in [9], where the definition of the refinement for M-nets assumed some restrictions however, on the interface of the refined transitions and on the entry/exit interface of the refining nets. A general refinement mechanism is defined in [11] for an M-net model which is slightly extended in order to reach the desired generality. The definition commutes with the unfolding operation and fulfils thus the basic property of all operations in the M-net calculus.
However, to define an M-net semantics for the parallel programming language $B(PN)^2$ [6] with procedures [20, 13], an extension of the general refinement from [11] is necessary. The present paper proposes such an extension, the parameterised general refinement, and studies some of its properties in the M-net calculus.
The paper is organised as follows. A comprehensive introduction of the M-net calculus is given in section 2. Section 3 motivates the need and describes the problems with an extension of

general refinement. Parameterised general refinement is then defined in section 4. Section 5 states some properties of the new operation. Some concluding remarks are given in section 6.

2 The M-net Model

2.1 Preliminaries

Let Val be a fixed but suitably large[1] set of *values*, Var and Par be suitably large[2] sets of *variables* and *parameters*, respectively; Val, Var and Par are assumed to be pairwise disjoint. The set of all well-formed *predicates* built from the sets Val, Var and[3] a suitable set of operators is denoted by Pr.

We assume the existence of a fixed set A of *action symbols*. Each action symbol $A \in \mathsf{A}$ is assumed to have an arity $ar(A)$ which is a natural number describing the number of its arguments. A construct $A(\nu_1, \ldots, \nu_{ar(A)})$, where A is an action symbol and $\forall j \in \{1, \ldots, ar(A)\} : \nu_j \in Var \cup Par \cup Val$, is an *action*. An action $A(\nu_1, \ldots, \nu_{ar(A)})$, where $\forall j \in \{1, \ldots, ar(A)\} : \nu_j \in Val$, is called *elementary*. An action $A(\nu_1, \ldots, \nu_{ar(A)})$, where $\forall j \in \{1, \ldots, ar(A)\} : \nu_j \in Val \cup Par$, is called *parametric*; the set of all parametric actions will be denoted as EP.

We also assume the existence of a fixed but suitably large[2] set X of *hierarchical actions*. The latter will be the key to refinements, and thus to any hierarchical presentation of a system, since they represent a kind of 'hole' to be later replaced by some corresponding (parameterised M-)net. Each hierarchical action symbol $X \in \mathsf{X}$ is as well assumed to have an arity $ar(X)$. A construct $X(\nu_1, \ldots, \nu_{ar(X)})$, where X is a hierarchical action symbol and $\forall j \in \{1, \ldots, ar(X)\} :$ $\nu_j \in Var \cup Par \cup Val$ (ν_j is an effective parameter), is a *parameterised hierarchical action*. The set of all parameterised hierarchical actions is denoted by PX. A parameterised hierarchical action $X(\nu_1, \ldots, \nu_{ar(A)})$, where $\forall j \in \{1, \ldots, ar(A)\} : \nu_j \in Val$, is called *elementary*.

Finally, we shall also use a set SA of *structured annotations*, built from the value and variable sets[3], which will denote nonempty sets of values. Their exact syntax will be specified later, in due time; at that point let us just notice that they include the sets Val and Var, a value v representing in that case the singleton set $\{v\}$ and a variable x representing the singleton set $\{v\}$ when the value of x is v.

A *substitution* is a function ϱ from a subset of parameters from Par to (a subset of) $Var \cup Par \cup Val$.

A *list of typed parameters* Ψ is a list of the form $\psi_1 : set_1, \ldots \psi_n : set_n$, where $\forall i \in \{1, \ldots, n\} :$ $\psi_i \in Par$ (with $i \neq j \Rightarrow \psi_i \neq \psi_j$) and $set_i \subseteq Val$. We denote the type set_i thus associated to the parameter ψ_i by $set(\psi_i)$, and the (formal) parameter set $\{\psi_1, ..., \psi_n\}$ associated to Ψ by $Par(\Psi)$.

2.2 Definition of M-nets

The main difference between M-nets and predicate/transition or coloured nets [14, 17] is that M-nets carry additional information in their place and transition inscriptions to support composition operations. In M-nets, besides the usual annotations on places (set of allowed tokens), arcs (multiset of structured annotations) and transitions (occurrence condition), we have an additional label on places denoting their status (entry, exit or internal) and an additional label on transitions, denoting the communication and hierarchical interface.

[1] In particular, this means that Val includes all the structured values which will be constructed through the refinement operation (see the definition of place types in section 4.1).

[2] In order to be able to rename them whenever necessary to avoid name clashes.

[3] In principle, it is possible to also allow the use of parameters from Par, but this would slightly complicate the definition of the semantics of our M-nets, so that we refrain here from the potential generality.

Definition 2.1 (Parameterised M-nets) *A* parameterised M-net *is a quadruple*[4] $(S, T, \iota; \Psi)$, *where* Ψ *is a list of typed parameters,* S *is a set of* places, T *is a set of* transitions *with* $S \cap T = \emptyset$, *and* ι *is an inscription function with domain* $S \cup (S \times T) \cup (T \times S) \cup T$ *such that:*

- *For every place* $s \in S$, $\iota(s)$ *is a pair* $\lambda(s).\alpha(s)$, *where* $\lambda(s) \in \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$, *called the* label *of* s, *and* $\alpha(s)$, *the* type *of* s, *is a nonempty set of values.*
- *For every transition* $t \in T$, $\iota(t)$ *is a triple*[5] $var(t).\lambda(t).\alpha(t)$; *where* $var(t)$, *the variables of* t, *is a finite set of variables from* Var; $\lambda(t)$, *the* label *of* t, *is a finite multiset of actions (t will then be called a* communication transition*), or a parameterised hierarchical action, i.e.,* $\lambda(t) \in \mathsf{PX}$ *(t will then be called a* hierarchical transition*); and* $\alpha(t)$, *the* guard *of* t, *is a finite set of predicates*[6]*; the variables occurring either in* $\lambda(t)$ *or* $\alpha(t)$ *are assumed to belong to* $var(t)$, *and the parameters occurring in* $\lambda(t)$ *are assumed to belong to* $Par(\Psi)$.
- *For every arc* $(s, t) \in (S \times T)$: $\iota((s, t)) \in \mathcal{M}_f(SA)$ *is a finite multiset of structured annotations*[7] *(analogously for arcs* $(t, s) \in (T \times S)$*); each structured annotation represents some nonempty set*[8] *of values absorbed or produced by the transition on the place.* $\iota((s, t))$ *will generally be abbreviated as* $\iota(s, t)$; *again, the variables occurring in* $\iota(s, t)$ *are assumed to belong to* $var(t)$.

Whenever we will need to refer to the parameters of a parameterised M-net explicitly we write $N(\Psi)$ for $N = (S, T, \iota; \Psi)$, and we will identify in the sequel the notions of M-net and parameterised M-net.
Figure 3 shows examples of parameterised M-nets[9].
Each type $\alpha(s)$ delimits the set of tokens allowed on place s, and $\lambda(s)$ describes the *status* (entry e, internal i or exit x) of s. The label of a transition t can either be a multiset of actions expressing synchronisation capabilities of t, or a parameterised hierarchical action informing about a possible future refinement of t.
Given a transition $t \in T$, the part of N which consists of the transition t and all its incident arcs will be called the *area* of t: $area(t) = (S \times \{t\}) \cup \{t\} \cup (\{t\} \times S)$. Note that areas of different transitions are always disjoint, and that $var(t)$ comprises all the variables occurring in the inscriptions of $area(t)$. Without loss of generality, it will always be possible to assume that if $t \neq t'$, then $var(t)$ and $var(t')$ are disjoint. A *binding* for t is a function $\sigma : var(t) \to Val$. If ϵ is an entity depending on the variables of $var(t)$, we shall denote by $\epsilon[\sigma]$ the evaluation of this entity under the binding σ; in general, this will be obtained by replacing[10] in ϵ each variable $a \in var(t)$ occurring in it (if any) by its value $\sigma(a)$. For instance, $\lambda(t)[\sigma] \in \mathcal{M}_f(\mathsf{EP})$, and $\alpha(t)[\sigma] \in \mathcal{M}_f(\{true, false\})$ (after the evaluation of the terms). The guard, $\alpha(t)$ plays the rôle of an *occurrence condition* in the sense that t may occur under a binding σ only if $\alpha(t)$ is

[4] This is a first difference to the previous definition of non-parameterised M-nets (see [18, 4, 10]), where no list of parameters Ψ is defined.

[5] This is a second difference w.r.t. the previous definitions of M-nets. As in [11] we define the set of variables and parameters directly, instead of an indirect definition from the set of variables and parameters occurring in the transition inscription and the variables occurring in incident arc inscriptions.

[6] Remember that parameters never occur in guards.

[7] Remember again that parameters never occur in structured annotations.

[8] Notice that it will never represent a multiset of values: the multiset aspect is coped with by the fact that $\iota((s, t))$ is itself a multiset of structured annotations, and by the fact that two distinct structured annotations may have common values in their represented sets.

[9] In order to abbreviate the annotations in the figures, we shall often omit the guard $\alpha(t)$ when it is empty, and the variable set $var(t)$ when it may easily be reconstructed from the annotations in the area of t.

[10] The evaluation rule will be slightly more complex for structured annotations; this will be clarified in definition 4.2; as mentioned earlier, each variable a occurring in $\iota(s, t)$ is evaluated in its associated singleton value $\{\sigma(a)\}$, and each value v in $\iota(s, t)$ is evaluated in its associated singleton value $\{v\}$.

true for σ, i.e., if all[11] terms from $\alpha(t)[\sigma]$ are *true*.
The arc inscriptions specify the token flow. An empty arc inscription means that no tokens may ever flow along that arc, i.e., there exists no effective connection along it; as usual, in figures, this will be noted by the absence of the arc. In figures, multiset parenthesis around arc inscriptions will sometimes be omitted for reasons of simplicity.
A binding σ of t will be said *enabling* if $\alpha(t)[\sigma] \in \mathcal{M}_f(\{true\})$, i.e., if it satisfies the guard, and if moreover $\forall s \in S:\ \iota(s,t)[\sigma] \in \mathcal{M}(\alpha(s)) \wedge \iota(t,s)[\sigma] \in \mathcal{M}(\alpha(s))$, i.e., the flow of tokens respects place types. Notice that, if we were to allow the occurrence of parameters in guards or structured annotations, the enabling character of a local binding could also rely on the choice of a *global binding* fixing the values of the net parameters; our restrictions thus amount to keeping the local aspect of the enabling condition and its insensibility to the choice of the parameter; the latter only has impact on the identification of the performed actions. The set of all enabling bindings of t will be denoted by $\beta(t)$. We shall assume that there is always at least one enabling binding for each transition (otherwise, it may be dropped; notice also that if the enablingness would depend on the parameters, it could happen that a transition has enabling bindings for some choices of the parameter values, and none for some other choices).
For a parameterised M-net $N = (S, T, \iota; \Psi)$ we will denote by ${}^\circ N$, N°, $\ddot{N} = S \setminus ({}^\circ N \cup N^\circ)$, respectively, the set of entry places, exit places, respectively the set of internal places of N.
In practical applications, we shall essentially be interested in a subclass of those M-nets, called *M-boxes*, enforcing some extra conditions, like the fact that there are entry and exit places (*ex*-restrictedness), there are no arcs to entry places nor from exit places (*ex*-directedness), and for each transition t there is at least one place s such that $\iota(s,t) \neq \emptyset$ and at least one place s' such that $\iota(t,s') \neq \emptyset$ (T-restrictedness). However, all the definitions apply to the general class of M-nets.
A *marking* of a parameterised M-net $(S, T, \iota; \Psi)$ is a mapping $M\colon S \to \mathcal{M}(Val)$ which associates to each place $s \in S$ a multiset of values from $\alpha(s)$. In particular, we shall distinguish the *entry marking*, where $M(s) = \alpha(s)$ if $\lambda(s) = \mathsf{e}$ and the empty (multi-)set otherwise, and the *exit marking*, where $M(s) = \alpha(s)$ if $\lambda(s) = \mathsf{x}$ and the empty (multi-)set otherwise.
The transition rule specifies the circumstances under which a marking M' is reachable from a marking M. A transition t is *enabled* at a marking M if there is an enabling binding σ of t such that $\forall s \in S:\ \iota(s,t)[\sigma] \leq M(s)$, i.e., there are enough tokens of each type to satisfy the required flow[12]. The effect of an occurrence of t is to remove all tokens used for the enabling binding σ of t from the input places and to add tokens according to σ to its output places.

Definition 2.2 (Transition Rule) *A transition t is* enabled *for an enabling binding σ at a marking M_1 if there is a marking M such that $\forall s \in S\colon M_1(s) = \iota(s,t)[\sigma] + M(s)$. The occurrence of t at M_1 under σ leads to a marking M_2, such that $\forall s \in S\colon M_2(s) = M(s) + \iota(t,s)[\sigma]$.*

The marking M that must exist by this definition can be understood as the part of the global marking that remains unchanged by the occurrence of t.
As usual, two (marked) M-nets N and N' are called *isomorphic*, if there are (marking-preserving, label-preserving and arc-annotation-preserving, up to locally coherent variable renamings and globally coherent parameter renamings) bijections between their places and their transitions.

[11] In other words, the multiset of predicates could be replaced by their mere conjunction; the reason why this is not done directly here is due to technical reasons explained in [19]; moreover, it could happen that the conjunction has not been included in the allowed operators.
[12] Again, from our restrictions, this is insensible to the choice of the parameter values.

2.3 Unfolding of M-nets

The unfolding operation associates a family of low level[13] Petri Boxes (see for instance [2]), $\mathcal{U}(N)$ with every parameterised M-net N, as well as a marking $\mathcal{U}(M)$ of $\mathcal{U}(N)$ with every marking M of N.

Definition 2.3 (Unfolding) *Let* $N = (S, T, \iota; (\psi_1 : set_1, \ldots, \psi_n : set_n))$, $p = (\psi_1, \ldots, \psi_n)$, $v\_set = set_1 \times \ldots \times set_n$ *and* $\rho : p \to v\_set$ *be a global binding (extending the notation introduced previously, we shall denote by* $\beta(N)$ *the set of all those global bindings); then* $\mathcal{U}(N)$ *is a mapping* $\rho \to \mathcal{U}(N, \rho)$, *where* $\mathcal{U}(N, \rho) = (\mathcal{U}(S), \mathcal{U}(T), W, \lambda_\rho)$ *is the labelled low level net defined as follows:*

- $\mathcal{U}(S) = \{s_v \mid s \in S \text{ and } v \in \alpha(s)\}$, *and for each* $s_v \in \mathcal{U}(S) : \lambda_\rho(s_v) = \lambda(s)$.
- $\mathcal{U}(T) = \{t_\sigma \mid t \in T \text{ and } \sigma \in \beta(t)\}$, *and for each* $t_\sigma \in \mathcal{U}(T) : \lambda_\rho(t_\sigma) = \lambda(t)[\sigma \cup \rho]$.
- $W(s_v, t_\sigma) = \sum_{x \in \iota(s,t)} \iota(s,t)(x) \cdot x[\sigma](v)$, *and analogously for* $W(t_\sigma, s_v)$.

Let M *be any marking of* N. *The marking* $\mathcal{U}(M)$ *of any* $\mathcal{U}(N, \rho)$ *is defined as follows: for every place* $s_v \in \mathcal{U}(S)$, $\mathcal{U}(M)(s_v) = M(s)(v)$.

Thus, each low level place $s_v \in \mathcal{U}(S)$ contains as many values v as the number of times this value occurs in the marking $M(s)$.
We refer to Figure 1 as an example for the unfolding operation.

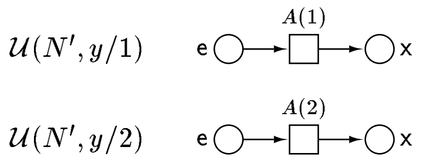


Figure 1: Unfolding of the parameterised M-net $N'(y : \{1, 2\})$ from Figure 3.

3 Describing the problem

In order to grasp the difficulty of the problem, even without explicit parameters, let us consider the example exhibited in Figure 2: a 'naive' refinement of the X-labelled transition in the M-net fragment of N as given in the first net by the parameterised M-net N' would result in the third fragment.
The problem then is that, while in N the two variables a occur in the surrounding of the same transition this is no longer the case in the third one, but variables only have a local meaning and may be changed independently around each transition. Hence it is necessary to *transport* the identity of the variables, or at least their bindings, from the entry of the refined copy to the exit (or the other way round).
A hierarchical transition t (labelled by X) of an M-net N has a set of enabling bindings, i.e., possibly more than a single one. Each enabling binding σ for t can be understood as a

[13] This is a difference with respect to the rule used in previous papers on M-nets [18, 9, 4], where the unfolding lead to a special class of M-nets, called *elementary* M-nets, from which a low level net can be immediately be obtained. Here, we preferred to go directly to a low level net because, as it will occur, elementary M-nets will no longer be stable for refinement. The same feature was also used, but for other reasons, in the A-net domain (see [18, 9, 19]). A difference with respect to the rule used in [11] is that here, due to the parametrization, the unfolding leads to a family of low level Petri Boxes instead of a single one.

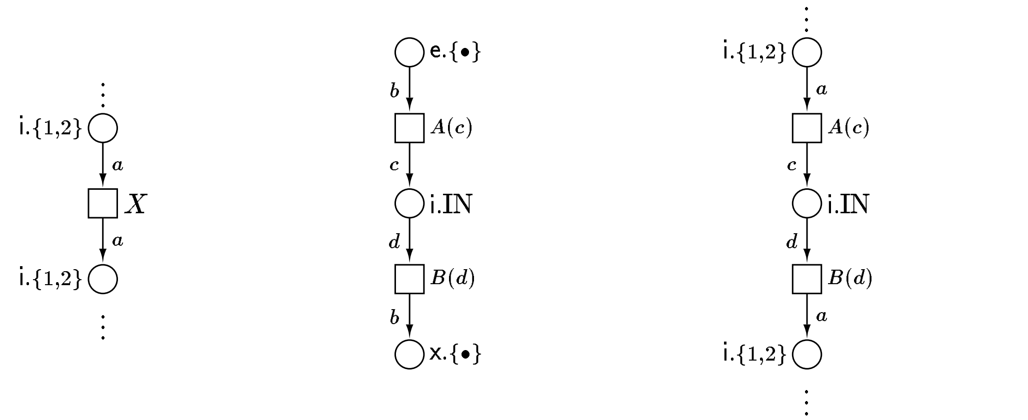


Figure 2: Illustration of the difficulty of the general refinement.

'mode' under which the refining M-net N' is executed. Once (if ever) the refining M-net has reached its exit marking, the execution of N is supposed to be continued in the state (marking) corresponding to the result of the occurrence of t under the considered mode σ. This problem is solved in [11] by extending the place types, the inscription of arcs and the evaluation of these arc inscriptions.[14]

In [20, 13] the semantics of a parallel programming language is extended by procedures and assumes therefore the existence of a refinement operation. [11] defines such an operation which possesses all the properties which are essential for a well-defined operation in the M-net calculus. However, general refinement has to be extended in order to be able to administrate parallel instantiations of a procedure and thus to be applied in the semantics of parallel programming languages with procedures as proposed in [20, 13].

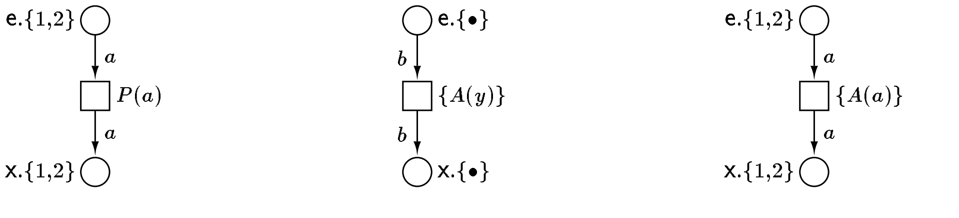


Figure 3: Illustration of the parameterised general refinement.

We explain the extension to parameterised general refinement by considering Figure 3, where we intend to model the call of a procedure P. The program semantics is basically obtained by refining the hierarchical transition in the first M-net through the second M-net. In a sequential environment, instantiations of a procedure are always ordered and might be administrated by a stack on which the procedures are stored in the order of their creation. The additional problem in a concurrent setting is, that we might have several instantiations of the same procedure generated by concurrent or recursive calls at the same level. Hence, we need to distinguish those instantiations. We may do this by using the bindings of the hierarchical transition labelled P, since each execution of the procedure is performed under a specific mode, i.e., the argument a is used to mark each instantiation. Therefore, in order to assign an action to a

[14] We do not follow the formal definition from [11] for the third M-net fragment depicted in Figure 2 in order to clarify the problem.

certain procedure instantiation, each performed action of the procedure has to be 'marked' by an additional parameter identifying the instance, thus this parameter is supposed to be bound by the chosen mode. The parameter y in the parameterised M-net $N'(y : \{1, 2\})$ of Figure 3 plays the rôle of this 'instance parameter'. Parameterised general refinement introduces now a mechanism, which allows to identify variables occurring in the area of a hierarchical transitions (such as a in N in Figure 3) into a parameter of the refining M-net (such as y in N').

4 Parameterised General Refinement

The parameterised refinement $N[X_i \leftarrow N_i \mid i \in I]$ means 'N where the X_i-labelled transitions are refined into (i.e., replaced by a copy of) N_i, for each i in the indexing set I'. By doing so, for each hierarchical transition t with a label of the form X_i(*actual parameter list*), the formal parameters of N_i will be substituted by the corresponding values from the actual parameter list. Hence, we shall assume that, for each $i \in I$, the arity of X_i is the same as the one of N_i, and that for each global binding ρ of N and each enabling binding σ of any hierarchical transition t with a label of the form X_i(*actual parameter list*), *actual parameter list*$[\sigma \cup \rho] \in v\_set_i$, i.e., the actual parameters belong to the ranges specified for the formal parameters in the corresponding refining net.
The definition is (as in [11]) slightly technically complex, due to the great generality we allow: any number of refined transitions (possibly infinitely many), any connectivity network, any arc weighting, any number of entry/exit places (possibly continuously infinitely many values, due to the cardinality explosion phenomenon [2, 1]) compatible with the M-net definition. More precisely, generalising the definition in [9, 10], the adjacent places of hierarchical transitions in N may have any arbitrary type, the connecting arcs may have any (finite) cardinalities and furthermore we impose no restriction on the types of the entry/exit places of the refining M-nets N_i.
To guarantee the commutativity of the refinement operation with the unfolding a *labelled tree* device similar to the one devised for low level nets [1] is used. This device nicely generalises the kind of multiplicative Cartesian cross product (pre/post places of transitions to be refined with entry/exit places of the refining net) commonly used in the literature [15] as the interface places. This setting has not been chosen just for the purpose of treating the general case, but also to get easily the main properties of the refinement operator. With this respect, it has been successfully reused in [7, 8, 9, 10, 11]. The main difference with the low-level case, and with the previous attempts to define refinements at the M-net level, will be that, in our context, the place types (and consequently the evaluations of the structured annotations in arc inscriptions) will be sets of labelled trees, but the interface places themselves remain unstructured.

4.1 A Tree Nursery

If $X \in \mathsf{X}$ is a hierarchical action symbol and $\mathcal{X}_I = \{X_i \mid i \in I\}$ is a set of such action symbols, let us define $T^X = \{t \in T \mid \lambda(t) = X(\nu_1, \ldots, \nu_{ar(X)})\}$ to be the set of all X-labelled hierarchical transitions, and $T^{\mathcal{X}_I} = \bigcup_{X \in \mathcal{X}_I} T^X$ to be the set of all hierarchical transitions with a label from $\mathcal{X}_I$.
The generalised Cartesian product uses a form of labelled trees. This device will be used to generate the new types of the interface places of the refined M-net, i.e., the places connected to transitions to be refined.
Like for the definition of refinements for low level-nets (see [1]), the places of refined parameterised M-net $N[X_i \leftarrow N_i]$ will be of two kinds: the interface places and the copied internal places:

$$\begin{array}{ccccc} & & v & & \\ \cdots & \sigma_t \swarrow & & \nwarrow \sigma_{t'} & \cdots \\ \cdots & (e,w) & & (x,w') & \cdots \end{array}$$

Figure 4: Illustration of the value trees for an interface place.

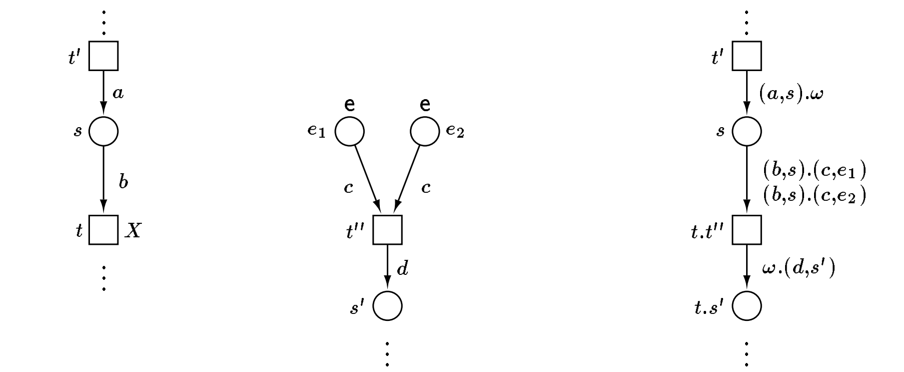
- Each place $s \in S$ of the M-net N will also[15] be a place of the refined M-net, with the same label as before. The only difference will be that its type will be a set of labelled trees constructed from the old value set and the entry/exit interface of the refining nets. The new type $\tilde{\alpha}(s)$ of s is the set of all the (isomorphic classes of) labelled trees of the following form (cf. Figure 4): the root is labelled by a value $v \in \alpha(s)$; the arcs are labelled by bindings σ_t of transitions t (in $T^{\mathcal{X}_I} \cap (s^\bullet \cup {}^\bullet s)$) and a direction (up or down) – more precisely, for each $i \in I$, for each (if any) $t \in s^\bullet$ with a label of the form $X_i(\cdot)$ and for each enabling binding σ_t of t such that $v \in \iota(s,t)[\sigma_t]$, there is an arc labelled σ_t going down to a node labelled by some arbitrarily chosen pair (e, w) where $e \in {}^\circ N_i$ and $w \in \alpha_i(e)$, and for each $i \in I$, each (if any) $t' \in {}^\bullet s$ with a label of the form X_i and for each enabling binding $\sigma_{t'}$ of t' such that $v \in \iota(t,s)[\sigma_{t'}]$, there is an arc labelled σ_t going up from a node labelled by some arbitrarily chosen pair (x, w') where $x \in N_i^\circ$ and $w' \in \alpha_i(x)$.
- Copied internal places of N_i form the set S^i of all the pairs $t.s_i$ where t is a transition of N labelled by $X_i(\cdot)$ and $s_i \in \ddot{N}_i$ is an internal place of the refining net N_i. In order to get a more uniform description, one may consider those pairs as trees reduced to their roots. The label of such a place will always be internal. The type of $t.s_i$ will be the set $\tilde{\alpha}(t.s_i)$ of all the pairs $\sigma_t.v$, where σ_t is an enabling binding of t and $v \in \alpha(s_i)$ is any value allowed on s_i.

The transitions of $\widetilde{N} = N[X_i \leftarrow N_i(\psi_{i,1} : set_{i,1}, \dots, \psi_{i,ar(X_i)} : set_{i,ar(X_i)}) \mid i \in I]$ will also be of two kinds: the untouched transitions $t \in T \setminus T^{\mathcal{X}_I}$, with the same inscription as before ($\iota(t)$ is the same in N and $\widetilde{N}$), and the copied transitions $t.t_i$, where $\lambda(t) = X_i(\nu_{i,1}, \dots, \nu_{i,ar(X_i)})$ and $t_i \in T_i$. The set of those copied transitions is denoted by T^i. As for the inscription of the latter, we shall assume (without loss of generality) that $var(t) \cap var_i(t_i) = \emptyset$; then $var(t.t_i) = var(t) \cup var_i(t_i)$, $\lambda(t.t_i) = \lambda(t_i)[\varrho_i]$ and $\alpha(t.t_i) = \alpha(t) \cup \alpha(t_i)$, where ϱ_i is the substitution $(\psi_{i,1}/\nu_{i,1}, \dots, \psi_{i,ar(X_i)}/\nu_{i,ar(X_i)})$.

In order to understand the rationale of the structured annotations occurring in the inscription of arcs of the refined M-net, let us consider the example depicted in Figure 5.

Transition t' is untouched but place s now has a type composed of labelled trees, and the occurrence of t' must produce one instance of each tree with a root labelled by a value produced through a in N; this will be represented by the notation $(a, s).\boldsymbol{\omega}$, where the 'hole' symbol $\boldsymbol{\omega}$ means that there is no constraint on the son labels. Transition $t.t''$, on the other hand, must absorb from s one instance of each tree with a root labelled by a value absorbed through b in N and a son corresponding to the selected mode (determined by the values of the variables from t) labelled by a value absorbed through c in e_1 by t'': this will be represented by the notation $(b, s).(c, e_1)$; it will also absorb trees of the same shape, but with the son labelled by a value absorbed through c in e_2 by t'', which will be represented, similarly, by the notation $(b, s).(c, e_2)$. Notice that the fact that t'' absorbs one token from each of the two entry places in N' is replaced, in the refined net, by the fact that $t.t''$ absorbs from s the tokens of two

[15] This is a major difference with respect to the solution considered in [10], where each such place was replaced by a family of tree-places of the same kind as in the low level definition, with the consequence that the commutativity with the unfolding operation was only obtained under restrictive assumptions and up to place duplication.



fragment of N fragment of N' fragment of $N[X \leftarrow N']$

Figure 5: Illustration of the structured annotations

structured annotations, like if the whole $^\circ N'$ were replaced by a single entry place; this is due to the fact that the new place s gathers all the tree values produced by the new t'. Finally, transition $t.t''$ must also produce in place $t.s'$ one instance of each value $\sigma.v$ where v is any value produced through d in N' and σ is any enabling binding for t (i.e., a mode determined by the values of the variables from t): this will be represented by the notation $\boldsymbol{\omega}.(d, s')$, where the 'hole' symbol $\boldsymbol{\omega}$ means that there is no special constraint on the first part of the value. It may be observed that some parts of these annotations are slightly redundant (the s and s'-parts may be reconstructed from the names of the connected places); they have been introduced to get a more systematic form and to simplify some proofs.

4.2 Definition of the Parameterised General Refinement

Using the concept of the labelled tree device and the notations introduced in the previous section, we are now able to give the full definition of the general parameterised refinement operation.

We will consider here general M-nets, but in practical applications we shall often restrict the treatment to cases with a finite number of hierarchical transitions; this will generally be enough to model the semantics of parallel programming languages with procedures, and is necessary to use the model checking [12] within the PEP-tool [3, 16].

Definition 4.1 (Parameterised General Refinement)
Let $N = (S, T, \iota; \Psi)$ and $N_i = (S_i, T_i, \iota_i; \Psi_i)$ be parameterised M-nets, where $i \in I$. The refinement $N[X_i \leftarrow N_i(\psi_{i,1} : set_{i,1}, \ldots, \psi_{i,ar(X_i)} : set_{i,ar(X_i)}) \mid i \in I]$ is defined as the parameterised M-net $\widetilde{N} = (\widetilde{S}, \widetilde{T}, \tilde{\iota}; \Psi)$, where

$$\tilde{S} = \bigcup_{i \in I} S^i \cup S, \qquad \tilde{T} = (T \setminus T^{\mathcal{X}_I}) \cup \bigcup_{i \in I} T^i,$$

and

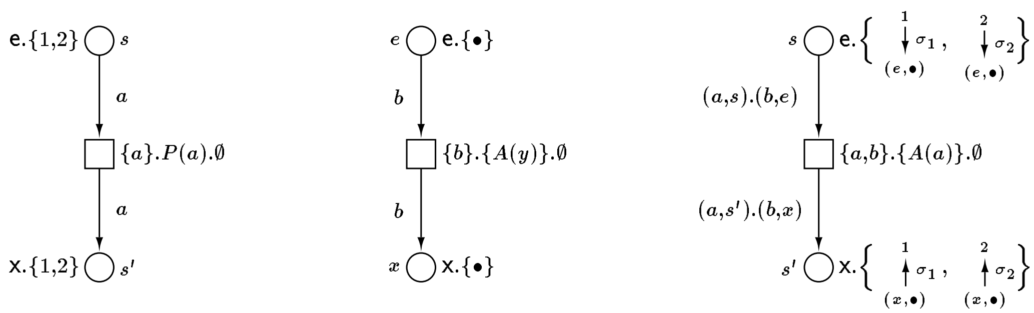
$$\tilde{\iota}(\tilde{s}) = \begin{cases} \lambda(\tilde{s}).\tilde{\alpha}(\tilde{s}) & \text{if } \tilde{s} \in S \\ \mathsf{i}.\tilde{\alpha}(\tilde{s}) & \text{if } \tilde{s} = t.s_i \in S^i \end{cases}$$

$$\tilde{\iota}(\tilde{t}) = \begin{cases} \iota(t) & \textit{if } \tilde{t} = t \in (T \setminus T^{\mathcal{X}_I}) \\ var(t) \cup var(t_i).\lambda_i(t_i)[\varrho_i].\alpha(t) \cup \alpha_i(t_i) & \textit{if } \tilde{t} = t.t_i,\ \lambda(t) = X_i(\nu_{i,1}, \ldots, \nu_{i,ar(X_i)}), \\ & t_i \in T_i, \textit{ and } \varrho_i \textit{ is the substitution} \\ & (\psi_{i,1}/\nu_{i,1}, \ldots, \psi_{i,ar(X_i)}/\nu_{i,ar(X_i)}) \end{cases}$$

$$\tilde{\iota}(\tilde{s}, \tilde{t}) = \begin{cases} \sum\limits_{a \in \iota(s,t)} \iota(s,t)(a) \cdot \{(a,s).\boldsymbol{\omega}\} & \textit{if } \tilde{t} = t \in (T \setminus T^{\mathcal{X}_I}), \textit{ and } \tilde{s} \in S \\ \sum\limits_{a \in \iota(s,t)} \sum\limits_{e_i \in {}^\circ N_i} \sum\limits_{b \in \iota_i(e_i,t_i)} \iota(s,t)(a) \cdot \iota_i(e_i,t_i)(b) \cdot \{(a,s).(b,e_i)\} & \\ & \textit{if } \tilde{t} = t.t_i \textit{ with } t \in T^{\mathcal{X}_I} \textit{ and } \tilde{s} \in S \\ \sum\limits_{b \in \iota_i(s_i,t_i)} \iota_i(s_i,t_i)(b) \cdot \{\boldsymbol{\omega}.(b,s_i)\} & \textit{if } \tilde{t} = t.t_i \textit{ and } \tilde{s} = t.s_i \in S^i, \textit{ with } t \in T^{\mathcal{X}_I} \\ \emptyset & \textit{otherwise,} \end{cases}$$

and analogously for arcs $\iota(\tilde{t}, \tilde{s})$.

Figure 6 shows the formal application of parameterised general refinement to the motivating example from Figure 3.



N $\qquad$ $N'(y : \{1,2\})$ $\qquad$ $N[P \leftarrow N']$

Figure 6: Parameterised general refinement applied to the example of Figure 3.

We still have to specify the evaluation of the structured annotations under a binding; this will lead to a set (possibly infinite) of values (labelled trees, possibly reduced to their root). In order to do that, let us first notice that, since for a hierarchical transition t of N and a transition t_i of the refining M-net N_i, we have $var(t.t_i) = var(t) \cup var(t_i)$, each binding of $t.t_i$ is the union of a binding for t and a binding for t_i, while the bindings of an untouched transition t' are the same as in N. It is not hard to see that the *enabling* bindings of t' in $\widetilde{N}$ are the enabling bindings of t' in N. The *enabling* bindings of $t.t_i$ are the unions of an enabling binding for t and an enabling binding for t_i. In other words, a refined transition $t.t_i$ in $\widetilde{N}$ may occur only under some binding (mode) σ for t in N *and* a binding σ_i for t_i in N_i, therefore $\mu = \sigma \cup \sigma_i$ denotes an enabling binding for $t.t_i$.

Definition 4.2 *The* evaluation *of a structured annotation in the inscription* $\tilde{\iota}(\tilde{s}, \tilde{t})$ *or* $\tilde{\iota}(\tilde{t}, \tilde{s})$ *of an arc in the refined net for a binding* $\mu = \sigma \cup \sigma_i$ *or* $\mu = \sigma$ *(as specified above), is defined by:*

- $(a,s).\boldsymbol{\omega}[\mu] = \{\tau \in \tilde{\alpha}(s) \mid$ *the root of* τ *belongs to* $a[\sigma]\}$, *if* $\tilde{t} = t \in (T \setminus T^{\mathcal{X}_I})$ *and* $s \in S$,
- $(a,s).(b,e)[\mu] = \{\tau \in \tilde{\alpha}(s) \mid$ *the root of* τ *belongs to* $a[\sigma]$ *and the son corresponding to the arc (down or up, depending on the inscripted arc) labelled* σ *belongs*[16] *to* $b[\sigma_i]\}$, *if* $\tilde{t} = t.t_i$ *with* $t \in T^{\mathcal{X}_I}$ *and* $s \in S$,

[16] Since parameters do not occur in arcs, we do not have to apply the substitutions ϱ_i to arc inscriptions.

- $\boldsymbol{\omega}.(b, s_i)[\mu] = \{\sigma.v \in \tilde{\alpha}(\tilde{s}) \mid v \textit{ belongs}^{15} \textit{ to } b[\sigma_i]\}$, *if* $\tilde{t} = t.t_i$ *with* $t \in T^{\mathcal{X}_I}$ *and* $\tilde{s} = t.s_i \in S^i$.

5 Some Properties of the Parameterised General Refinement

At first we notice that the definition is sound with respect to general refinement in [11], i.e., parameterised refinement for parameterised M-nets without parameters corresponds to general refinement for non-parameterised M-nets.
Our main interest lies in the relation of parameterised general refinement with respect to the unfolding operation in the presence of parameters.

Theorem 5.1 (Commutativity) *Let* $N = (S, T, \iota; \Psi)$ *and* $N_i = (S_i, T_i, \iota_i; \Psi_i)$, *be parameterised M-nets, where* $i \in I$. *Then, up to isomorphism*

$$\mathcal{U}(N[X_i \leftarrow N_i \mid i \in I]) = \mathcal{U}(N)[X_i(\rho_i) \leftarrow \mathcal{U}(N_i, \rho_i) \mid i \in I, \rho_i \in \beta(N_i)].$$

Proof (Sketch): The proof is similar to the one used in [11], and results from an identification of the components occurring on both side of the equation. ∎

6 Conclusion

We defined parameterised general refinement as an extension of the general refinement operator for M-nets proposed in [11] by introducing a substitution mechanism. It coincides with general refinement in the case where no variables are renamed. An intended application of parameterised general refinement is the semantics of a parallel programming language with procedures. By establishing some basic properties of the operation in the M-net calculus, we gained a better understanding for such applications.

References

[1] E. Best, R. Devillers, and J. Esparza. General Refinement and Recursion for the Box Calculus. *STACS'93*. Springer, LNCS Vol. 665, 130–140 (1993).

[2] E. Best, R. Devillers, and J.G. Hall. The Box Calculus: a New Causal Algebra with Multilabel Communication. *Advances in Petri Nets 92*. Springer, LNCS Vol. 609, 21–69 (1992).

[3] E. Best and H. Fleischhack, editors. *PEP: Programming Environment Based on Petri Nets*. Number 14/95 in Hildesheimer Informatik-Berichte. Universitiät Hildesheim, May 1995.

[4] E. Best, H. Fleischhack, W. Fraczak, R.P. Hopkins, H. Klaudel, and E. Pelz. A Class of Composable High Level Petri Nets. *Application and Theory of Petri Nets 1995*. Springer, LNCS Vol. 935, 103–120 (1995).

[5] E. Best, H. Fleischhack, W. Fraczak, R.P. Hopkins, H. Klaudel, and E. Pelz. An M-Net Semantics of $B(PN)^2$. *Structures in Concurrency Theory: STRICT'95* Proceedings. Springer, 85–100 (1995).

[6] E. Best, R.P. Hopkins. $B(PN)^2$ - a Basic Petri Net Programming Notation. *Proceedings of PARLE'93*. Springer, LNCS Vol. 694, 379–390 (1993).

[7] R. Devillers. The Synchronisation Operator Revisited for the Petri Box Calculus. Technical Report LIT-290, Université Libre de Bruxelles (1994).

[8] R. Devillers. S-Invariant Analysis of General Recursive Petri Boxes. *Acta Informatica*, Vol. 32, 313–345 (1995).

[9] R. Devillers and H. Klaudel. Refinement and Recursion in a High Level Petri Box Calculus. In *Structures in Concurrency Theory: STRICT'95.* Springer, 144–159 (1995).

[10] R. Devillers, H. Klaudel and R.-C. Riemann. General Refinement in the M-net Calculus. Technical Report LIT-357, Université Libre de Bruxelles (1997). http://www.lri.fr/~robert/work.html.

[11] R. Devillers, H. Klaudel and R.-C. Riemann. General Refinement for High Level Petri Nets. In *Foundation of Software Technology and Theoretical Computer Sience'97*, Springer, LNCS, (1997). http://www.lri.fr/~robert/work.html.

[12] J. Esparza. *A Partial Order Approach to Model Checking.* Habilitationsschrift, Universität Hildesheim, Februar 1993.

[13] H. Fleischhack and B. Grahlmann. A Petri Net Semantics for $B(PN)^2$ with Procedures. *Parallel and Distributed Software Engineering*, Boston Ma., 1997.

[14] H. Genrich. Predicate-Transition Nets. In *Petri Nets: Central Models and their Properties, Advances in Petri Nets 1986 Part I.* Springer, LNCS Vol. 254, 207–247 (1987).

[15] R.J. van Glabbeek and U. Goltz. Refinement of Actions in Causality Based Models. *Stepwise Refinement of Distributed Systems.* Springer, LNCS Vol. 430, 267–300 (1989).

[16] B. Grahlmann. The PEP Tool. *Computer Aided Verification*, Springer, LNCS Vol. 1254, 440–443 (1997).

[17] K. Jensen *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. EATCS Monographs on Theoretical Computer Science*, Vol. 1. Springer (1992).

[18] H. Klaudel. Modèles algébriques, basés sur les réseaux de Petri, pour la sémantique des langages de programmation concurrents. PhD Thesis, Université Paris XI Orsay (1995).

[19] H. Klaudel and E. Pelz. Communication as Unification in the Petri Box Calculus. Fundamentals of Computation Theory. Springer, LNCS Vol. 965, 303-312 (1995).

[20] J. Lilius and E. Pelz. An M-net Semantics for $B(PN)^2$ with Procedures. In *ISCIS XI*, Vol. I, 365–374, Antalya, November 1996. Middle East Technical University.

[21] R. Milner. *Communication and Concurrency.* Prentice-Hall (1989).