On Formally Refining Timed Petri Nets*

Sergio Bandinelli†
Dipartimento di Elettronica
Politecnico di Milano

Esteban Feuerstein‡
Dipartimento di Informatica e Sistemistica
Universita di Roma "La Sapienza"
on leave from ESLAI

Abstract

Petri nets —and some extended models based on them— have been recognized as an appropiate formalism for describing concurrent and time-dependent systems, because of its combination of a graphical and mathematically founded model. However, Timed Petri nets lack a formal, satisfactory and widely accepted notion of refinement. In this paper we propose a formal definition of refinement for a powerful, highly expressive and general time-oriented formalism, called ER-nets (a high level extension of the original Petri net model). We describe refinements as net morphisms that allow us to relate two net specifications at different levels of abstraction. Then we show how this definition may be used to define decomposition schemata for ER-nets, that guarantee that the decomposed net is actually a refinament of the original one. In particular, we provide nondeterministic, sequential and parallel decompositions. The use of these transformations is ilustrated through examples.

*This work has been done while both authors were at ESLAI (Escuela Superior Latino Americana de Informatica).

†Supported by Engineering S.p.A.

‡Partially supported by Fundacion Antorchas - Argentina.

1 Introduction

Petri nets are increasingly being used as a modeling language for representing concurrent systems. This is not only a consequence of their neat graphical representation, but also because the Petri net formalism [Reisig 82] is a well-known mathematical model, supported by theoretical studies. Moreover, many methods have been proposed for analyzing system properties specified with Petri nets; for example [Ghezzi 89b] proposes symbolic execution and [Berthomieu 83] presents time analysis. In addition, some extensions to the model for representing time dependent systems, i.e. systems which correctness depend on real time constraints (such as process control systems or communicating protocols) have been proposed, widening the area of application of Petri nets.

The ability of modelling a system hierarchicaly is a valuable and necessary feature of any specification formalism. But one of the main drawbacks of Petri nets is the unavailabity of a satisfactory hierarchical decomposition facility [Harel 86]. The main aim of this work is to provide a formal way to hierarchicaly decompose a net.

There have been many efforts to provide hierarchical developement for Petri nets. A very flexible way of relating system descriptions at different levels of abstraction, consisting in mapping single transitions into whole computations, is provided by [Meseguer 88] (for place transition nets) and by [Feuerstein 88] (for a simple model of Petri nets with time). A proposal towards defining refinements for TB-nets (a timed basic model described in [Ghezzi 89]) is discussed in [Felder 88] and a set of rules and restrictions to correctly expand a transition in a full process are given.

In this paper we adopt the net formalism proposed in [Morasca 89] and [Ghezzi 91] called Environment/Relation nets (ER-nets). An ER-net is a high level Petri net model, which may be particularly useful in the modelling of timming issues and has the property of being general and basic enough to include previous models, defined as ad hoc time extensions of the Petri net formalism.

We follow the ideas from [Meseguer 88] and [Feuerstein 88] to give formal notion of refinement for ER-nets. However, the definition does not provide a mechanism to correctly refine a net. Therefore we provide some topological transformations on ER-nets, based in the ones proposed in [Morasca 89], that guarantee that the net obtained after the tranformation is indeed an implementation of the original one, according to the formal definition of refinement. These transformations may be used by the designer during a hierarchical developement of the system specification.

This work is structured as follows. Section 2 presents the basic definitions of ER-nets as given in [Morasca 89]. In section 3, ER-nets are defined in terms of graphs and a notion of equivalence between nets is given. A notion of refinements for the ER-net model is formally given in section 4. Section 5 is dedicated to the description of topological transformations on ER-nets and are ilustrated through an example. The final conclusions are mentioned in section 6.

2 Basic definitions

Environment Relation nets (ER-net for short) are a high-level extension of Petri nets, similar in scope to Predicate Transition nets [Genrich 86] and Coloured nets [Jensen 86]. In this paper, we assume that the reader knows the basics of Petri nets and their intended

uses, therefore we limit ourselves to providing the basic definitions of the ER-net model and refer the reader to [Reisig 82] [Peterson 77] for basic definitions and notation relative to the Petri net formalism.

In the ER-nets model, tokens carry information in the form of environments, i.e. as a mapping from identifiers to values. The enabling of a transition depends not only on the presence of environments in the input places, but also on the number and on the information that those environments contain. The firing of a transition has the effect of destroying the environments enabling that firing and creating a new tuple of environments in the output places of the transition. The link between the removed tuple and the produced one is described by a relation associated with the transition. More than one output tuple may be related with an input tuple, giving a source of nondeterminism in the behaviour of the transition. Note that the relations also specify the number of input and output environments that participate of a firing, for each input and output place. That is, a relation links multisets of environments in the input places to multisets of environments in the output places.

2.1 Formal definition of ER-nets

Definition 2.1 (ER-net) *An ER-net is a 6-tuple* (P, T, E, F, R, M_0) *where* P, *the set of places,* T, *the set of transitions and* E, *the set of environments are arbitrary sets such that* $P \cap T = \emptyset; P \cap E = \emptyset; T \cap E = \emptyset$
$P \cup T \neq \emptyset$ *and* $E \neq \emptyset$
The flow relation F *links places to transitions and transitions to places and verifies that:* $F \subseteq (P \times T) \cup (T \times P)$ *and* $\Pi_1(F) \cup \Pi_2(F) = P \cup T$ *(i.e. there are not isolated places or transitions)*
If $E^{\oplus}$ *denotes the set of all finite multisets (also called bags) of elements of* E, *and* $A \rightarrow E^{\oplus}$, *the set of all aplications from a set* $A \subseteq P$ *to* $E^{\oplus}$, *then the relation* R *may be defined for each transition* $t \in T$ *as:*
$R(t) \subseteq ((^{\bullet}t \rightarrow E^{\oplus}) \times (t^{\bullet} \rightarrow E^{\oplus}))$
Finally, the initial marking M_0 *belongs to the set of all possible markings of the net:*

$$M_0 \in (P \rightarrow E^{\oplus})$$

During the evolution of an ER-net, the relation $R(t)$ links the tuples of environments removed from the places of the preset of t (denoted by $^{\bullet}t$) with the tuples of environments produced in the places of the postset of t (denoted by $t^{\bullet}$). The domain of the relation, $\Pi_1(R(t))$ is called the set of the enabling tuples of transition t. The range of the relation, $\Pi_2(R(t))$, is the set of tuples which can be produced by the firing of transition t.

In this work we restrict ourselves to the case where, presets, postsets and markings are finite, even in the presence of an infinite number of places and transitions, and although the set of possible markings were infinite. This kind of restriction is also present in [Degano 89] and we believe it is not a significant inconvenience.

Definition 2.2 (Set of Firings) *Let* $N = (P, T, E, F, R, M_0)$ *be an ER-net. The set of firings of* N *is defined as:*

$$Y = \{\langle t, en, pr\rangle / t \in T, \langle en, pr\rangle \in R(t)\}$$

For each firing $y \in Y$, en is called the *enabling tuple* (of bags) and pr, the *produced tuple* (of bags) for transition t. A firing $y = \langle t, en, pr \rangle$ is said to be *enabled* in the marking m iff $\forall p \in {}^\bullet t\,(en(p) \subseteq m(p))$.

The effect of the firing of a transition, called *firing occurrence* is a modification of the marking of the net in which the firing is enabled. The marking after the firing is equal to the marking before the firing except for the input places of the fired transition, where the enabling tuple is substracted, and for the output places, where the produced tuple is added.

2.2 Monotonic ER-nets

A time-dependent system is characterized by a monotonic behaviour with respect to time. This means that the output of an action may not be produced at a time which is greater than the time the inputs where created. To model this fact using ER- nets, a (partial) ordering between environments is introduced. We interpretate that an environment is less than other environment if it was produced before. An ER-net that models a monotonic behaviour must satisfy that, for each transition, the associated relation is monotonic, i.e. no environment belonging to a produced tuple is less than any environment of the corresponding removed tuple. Formaly,

Definition 2.3 (Monotonic Relation) *Let $\geq$ be a reflexive and transitive relation defined on the elements of E, i.e. let $(E, \geq)$ be a partial order, and let A and B be two arbitrary sets. A relation*

$$R \subseteq ((A \rightarrow E^{\oplus}) \times (B \rightarrow E^{\oplus}))$$

is said to be a monotonic relation w. r. t. $(E, \geq)$ iff

$$\forall \langle m, m' \rangle \in R\,(\forall a \in A\, \forall b \in B (\forall e \in m(a)\, \forall e' \in m'(b)\,(e' \geq e)))$$

Definition 2.4 (Monotonic ER-net) *An ER-net is said to be monotonic w. r. t. $(E, \geq)$ iff*

1. *$(E, \geq)$ is a partial order and*
2. *$\forall t \in T\,(R(t)$ is monotonic w. r. t. $(E, \geq)$*

Monotonic ER-nets support specification of control flow, data and functionalities, including time issues. Control flows are represented by the flow of environments; data are represented as information associated with environments and functionalities are specified as relations associated with transitions. The ordering defined on the set of environments allows also the representation of time issues. The following example shows how this can be done.

2.3 An Example

We show how a cake recipie may be specified using ER-nets. Making a cake is a real time process, since its correctness depends on time constraints (otherwise you would eat a burnt cake or a raw one!). Here follows the informal enuntiation of the recipie, as it may be found in a cooking book.

The ingredients for making a cake are: 4 eggs, one pound of flower, one pound of sugar and one ounce of butter. The preparation of the dough takes 10 or 15 minutes and the baking time is of 40 minutes. Beat the eggs for 5 minutes using an electric beater (or for 10 minutes if you use a fork). Mix the flower, the sugar and the butter and then add the beaten eggs. Put the preparation in the oven and that's all[1].

We may represent the recipie by an ER-net $\mathcal{N} = (P, T, E, F, R, M_0)$. The ingredients, the cake and the intermediate products are represented with environments having two pieces of information: the name of what they represent and the time at which it was produced. We write $e = (id, t)$ meaning that environment e represents id and was created at time t. The times associated with each environment provide us with a partial order[2] on the set E in a natual way: environment e is greater or equal than environment e' if e's time is greater or equal than e''s time.

Formally, the parameters of $\mathcal{N}$ are $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$. $T = \{$Start, Mix, BtFrk, BtBtr, Add, Bake$\}$[3]. $E = \{(egg, t_e), (flower, t_f), (butter, t_b), (sugar, t_s), (cake, t_c), (beateneggs, t_{be}), (mixture, t_m), (dough, t_d)\}$ where $t_e, t_f, t_b, t_s, t_c, t_{be}, t_m, t_d \in \Re$. F is given by Figure 1; finally, the relations associated with the transitions and the initial marking are:

$$
\begin{aligned}
R(Start) &= \{[4(p_1,(egg,t_e)) \oplus (p_1,(flower,t_f)) \oplus (p_1,(butter,t_b)) \ominus (p_1,(sugar,t_s)),\\
&\quad (p_2,(flower,t_1)) \oplus (p_2,(butter,t_1)) \ominus (p_2,(sugar,t_1)) \ominus 4(p_3,(egg,t_1))]\}\\
&\quad where\ t_1 = Max(t_e,t_f,t_b,t_s)\\
R(Mix) &= \{[(p_2,(flower,t_f)) \oplus (p_2,(butter,t_b)) \oplus (p_2,(sugar,t_s)),(p_4,(mixture,t_2))]\}\\
&\quad where\ t_2 = Max(t_f,t_b,t_s) + 5\\
R(BtFrk) &= \{[4(p_3,(egg,t_e)),(p_5,(beateneggs,t_e+10))]\}\\
R(BtBtr) &= \{[4(p_3,(egg,t_e)),(p_5,(beateneggs,t_e+5))]\}\\
R(Add) &= \{[(p_4,(mixture,t_m)) \oplus (p_5,(beateneggs,t_{be})),(p_6,(dough,Max(t_m,t_{be})))]\}\\
R(Bake) &= \{[(p_6,(dough,t_d)),(p_7,(cake,t_d+40))]\}\\
M_0 &= 4(p_1,(egg,0)) \oplus (p_1,(flower,0)) \oplus (p_1,(butter,0)) \ominus (p_1,(sugar,0))
\end{aligned}
$$

□

3 Equivalence of ER-nets

In this section we see an ER-net as a ordinary directed graph. This approach allows us to define ER-net morphisms as a certain kind of graph morphisms and give a notion of

[1] The authors shall not be liable for damages caused by the use of this recipie.

[2] Actually it is a total order.

[3] BtFrk stands for "Beat with Fork" and BtBtr stands for "Beat with (electric) Beater".

equivalence of ER-nets based on the existence of a bijective morphism between the corresponding graphs. The same definitions may be appied to monotonic ER- nets, provided that the morphism respect the monotonicity of the relations.

3.1 ER-nets as graphs

We can associate a graph with an ER-net in the following way. The nodes of the graph correspond to all the possible markings of the net (the reachable and unreachable ones) and the oriented arcs to the firings. The initial marking is a distinguished node in the graph. The functions ∂_0 and ∂_1, called *source* and *target* respectively, are defined for each firing $\langle t, en, pr \rangle \in Y$ as $\partial_0 = \Pi_2$ and $\partial_1 = \Pi_3$.

Definition 3.1 (ER-graph) *Given the ER-net* $\mathcal{N} = (P, T, E, F, R, M_0)$ *the corresponding ER-graph is defined as the graph*

$$\mathcal{G}[\mathcal{N}] = (\partial_0, \partial_1 : Y \longrightarrow (P - E^{\oplus}), M_0)$$

There is an important remark to be made with respect to the structure of the set of nodes. The set of nodes of a net is isomorphic to the free conmutative monoid generated by the pairs (p, e) with $p \in P$ and $e \in E$. In symbols [4]:

$$(P - E^{\oplus}) \simeq (P \times E)^{\oplus}$$

Therefore, there is a monoid structure defined on the nodes of the graph. A generic element $m \in (P \times E)^{\oplus}$ may be written as the formal sum: $m = \bigoplus_{i,j} k_{i,j}(p_i, e_j)$ where the order of the sumands is superfluous. This notation means that $k_{i,j}$ copies of environment e_j are present in place p_i. Addition is defined in $(P \times E)^{\oplus}$ for two elements m and m' as follows

$$m \oplus m' = \bigoplus_{i,j} k_{i,j}(p_i, e_j) \oplus \bigoplus_{i,j} k'_{i,j}(p_i, e_j) = \bigoplus_{i,j} (k_{i,j} + k'_{i,j})(p_i, e_j)$$

resulting 0 (the empty bag) as the neutral element.

The previous discusion and the fact that it is easy to translate from one notation to the other[5] allows us to use alternatively any of the two isomorphic structures.

Notation: We adopt the traditional notation from graph theory $\alpha : m \longrightarrow m'$ meaning that for the arc α, $\partial_0(\alpha) = m$ and $\partial_1(\alpha) = m'$

Observe that there is no missing information in the definition of ER-graph, since the original ER-net can be easily recovered from the graph.

Example.
Figure 2 is a graphical representation[6] of the ER-graph corresponding to the ER-net $\mathcal{N}$ of our running example. □

[4] $E^{\oplus}$ may be seen as a function from E to the natural numbers, indicating the number of instances in the bag for each environment. Then $(P - E^{\oplus}) \simeq (P - E - \mathcal{N}) \simeq (P \times E) - \mathcal{N} \simeq (P \times E)^{\oplus}$

[5] Since $(P - E^{\oplus})$ and $(P \times E)^{\oplus}$ are isomorphic, there exists a bijection $\mathcal{F} : (P - E^{\oplus}) \longrightarrow (P \times E)^{\oplus}$ defined in the following way: let $m \in (P \times E)^{\oplus}$; $m' \in (P - E^{\oplus})$. If $m = \bigoplus_{i,j} k_{i,j}(p_i, e_j)$ and $m' = \mathcal{F}(m)$

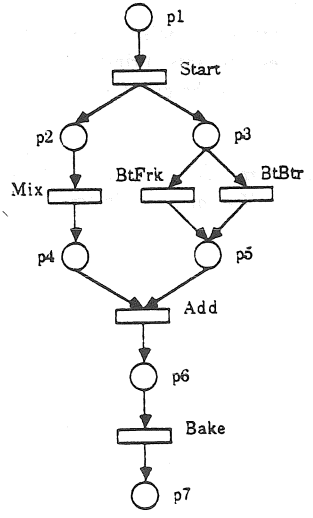

Figure 1: Graphical representation of ER-net $\mathcal{N}$.

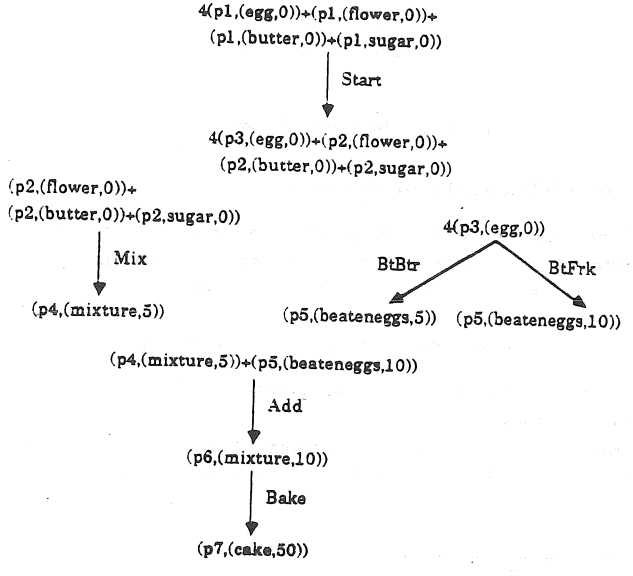

Figure 2: Graphical representation of the ER-graph $\mathcal{G}[\mathcal{N}]$.

3.2 ER-nets morphisms and equivalence

Two ER-graphs can be related through a morphism satisfying certain restrictions. The morphism respectively maps the nodes and arcs (firings and markings) of one net to the nodes and arcs of another net, and respects the graph structure and the monoid structure of the set of nodes.

Definition 3.2 (ER-graph morphism) *Let $\mathcal{N} = (P, T, E, F, R, M_0)$ and $\mathcal{N}' = (P', T', E', F', R', M_0')$ be ER-nets. An ER-graph morphism $h : \mathcal{G}[N] \longrightarrow \mathcal{G}[N']$ is a pair of functions $h = \langle f, g \rangle$ such that*
$f : Y \longrightarrow Y'$
$g : (P \times E)^{\oplus} \longrightarrow (P' \times E')^{\dot{+}}$
and
$g \circ \partial_0 = \partial_0' \circ f$
$g \circ \partial_1 = \partial_1' \circ f$
g is a monoid morphism, i.e. satisfies
$\forall m_1, m_2 \in (P' \times E')^{\oplus}\ g(m_1 \dot{+} m_2) = g(m_1) \dot{+} g(m_2)$ *and* $g(0) = 0$
g preserves the initial marking, i.e., $g(M_0) = M_0'$.

Now we are ready to provide a notion of equivalence between ER- nets.

Definition 3.3 (ER-nets equivalence) *Two ER-nets $\mathcal{N} = (P, T, E, F, R, M_0)$ and $\mathcal{N}' = (P', T', E', F', R', M_0')$ are said to be equivalent, we write $E \equiv E'$, iff there exists a ER-graph morphism $h : \mathcal{G}[N] \longrightarrow \mathcal{G}[N']$ where $h = \langle f, g \rangle$ such that*
$f : Y \longrightarrow Y'$ *is bijective*
$g : (P \times E)^{\oplus} \longrightarrow (P' \times E')^{\dot{+}}$ *is a monoid isomorphism*

Clearly '$\equiv$' is an equivalence relation between ER-nets.

3.3 Monotonic ER-graphs morphisms

Monotonic ER-nets have an associated partial order defined on the set of environments. This partial order $(E, \geq)$ may be naturally extended to the markings of an ER-net. The result is another partial order $((P \longrightarrow E^{\oplus}), \succeq)$ defined as follows. If $m, m' \in (P \longrightarrow E^{\ominus})$ then

$$m \succeq m' \Longleftrightarrow \forall p \in P(\forall e \in m(p), \forall e' \in m'(p)(e \geq e'))$$

In other words, each environment belonging to marking m has to be greater or equal than each environment of marking m'.

The monotonicity property may now be enuntiated for ER- graphs. We say that an ER-graph $\mathcal{G}[N] = (\partial_0, \partial_1 : Y \longrightarrow (P \longrightarrow E^{\oplus}), M_0)$ is monotonic iff

$$\forall y \in Y(\partial_1(y) \succeq \partial_0(y))$$

---

then $\forall i (1 \leq i \leq n) m'(p_i) = \bigoplus_j k_{i,j} e_j$.

[6] The figure is necessarily an incomplete representation, since the graph has an infinite number of nodes and arcs.

That is, we require all the environments belonging to the produced tuple to be grater or equal than the each environment of the corresponding removed tuple. Note that all the enviroments involved in a firing must be related by $\succeq$.

Definition 3.4 (Monotonic ER-graph morphism) *We call monotonic ER-graph morphism an ER-graph morphism $h = \langle f, g \rangle$ that in addition verifies that*

$$\forall m, m' \in (P - E^{\oplus})\ m \succeq m' \Rightarrow g(m) \succeq g(m')$$

4 Refinemets for ER-nets

Here we provide a formal notion of refinement for ER-nets and monotonic ER-nets. We begin by defining the ER-Computation graph, which contains as arcs all the possible computations of an ER-net. Then, a refinement of an ER-net is defined as an implementation morphism relating its ER-graph with its ER-Computation graph. In this way, a complete computation may be associated with a single transition via an implementation morphism. Finally, we observe that the initial definition of refinement as an implementation morphism appears to be too permisive: thus we propose a restricted version of the notion of refinement, that we call proper refinement.

4.1 ER-Computation graph

A computation is a single firing, a sequence of computations or a set of parallel computations. The computation graph is a graph whose nodes stand for markings and whose arcs represent all possible computations of an ER-net. For technical reasons, we include in the set of arcs an idle (identity) computation for each marking, meaning that the environments are left in their places.

Definition 4.1 (ER-Computation graph) *The ER-Computation graph corresponding to the ER-net $\mathcal{N} = (P, T, E, F, R, M_0)$ is the graph*

$$C[\mathcal{N}] = (\partial_0, \partial_1 : C \longrightarrow (P - E^{\oplus}), M_0)$$

The following rules define the arcs of the graph:

$$\frac{\langle t, en, pr \rangle : en \multimap pr \in Y}{\langle t, en, pr \rangle : en \multimap \overline{pr} \in C}$$

$$\frac{m \in (P \multimap E^{\oplus})}{m : m \multimap m \in C}$$

$$\frac{\alpha : m_1 \multimap m_1' \in C \text{ and } \beta : m_2 \multimap m_2' \in C}{\alpha \oplus \beta : m_1 \oplus m_2 \multimap m_1' \ominus m_2' \in C}$$

$$\frac{\alpha : m \multimap m' \in C \text{ and } \beta : m' \multimap m'' \in C}{\alpha ; \beta : m \multimap m'' \in C}$$

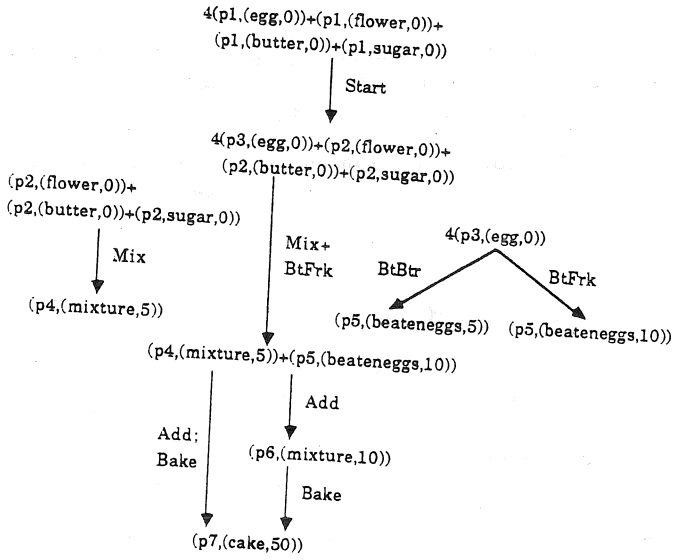

Figure 3: Graphical representation of the ER-computation graph $\mathcal{C}[N]$.

The definition introduces two operations between computations: '$\ominus$' and ';' that we call respectively parallel and sequential composition of computations[7]. With the above definition, there are some different arcs in the computation graph, that intuitively describe the same computation. Thus, we identify these arcs by completing the definition with the following axioms: Let $\alpha, \alpha', \beta, \beta', \gamma \in C$

- $(\alpha \ominus \beta) \ominus \gamma = \alpha \ominus (\beta \ominus \gamma)$
- $\alpha \ominus \beta = \beta \ominus \alpha$
- $\alpha \ominus 0 = \alpha$
- $\partial_0(\alpha); \alpha = \alpha; \partial_1(\alpha) = \alpha$
- $(\alpha;\beta);\gamma = \alpha;(\beta;\gamma)$
- $(\alpha \oplus \alpha');(\beta \oplus \beta') = (\alpha;\beta) \ominus (\alpha';\beta')$

As a consecuence of the definition we have that $(C, \oplus, 0)$is a conmutative monoid[8].

Example

Figure 3 shows the computation graph $\mathcal{C}[N]$. for net N, describing the cake recipie[9] □

[7]We use the same simbol $\oplus$ to denote the formal sum between markings and the parallel composition of computations. However this should cause no confusion, since it is always clear from the context to which operation we are making reference.

[8]Here 0 represents the empty transition that goes from the empty marking to the empty marking.

[9]Once again, the figure is necessarily an incomplete representation, since the graph has an infinite number of nodes and arcs.

4.2 Implementation morphisms and refinements

If $\mathcal{N}$ and $\mathcal{N}'$ are two ER-nets, an implementation morphism gives us the possibility of mapping a transition of $\mathcal{N}$ into an entire computation of $\mathcal{N}'$, which may contain many sequential and parallel steps. To achieve this capability it is enough to restrict our attention to the ER-graph of $\mathcal{N}$ and the ER-Computation graph of $\mathcal{N}'$, (the latter may also be seen as an ER-graph where the firings are computations of $\mathcal{N}'$).

Definition 4.2 (Implementation morphism) *Let $\mathcal{N}$ and $\mathcal{N}'$ be ER-nets. An implementation morphism from $\mathcal{N}$ to $\mathcal{N}'$ is an ER-graph morphism $h : \mathcal{G}[\mathcal{N}] \rightarrow \mathcal{C}[\mathcal{N}']$.*

Definition 4.3 (Refinement of an ER-net) *Let $\mathcal{N}$ and $\mathcal{N}'$ be ER-nets. We say that $\mathcal{N}'$ is a refinement of $\mathcal{N}$ iff there exists an implementation morphism from $\mathcal{N}$ to $\mathcal{N}'$.*

4.3 Proper Refinements for ER-nets

The definition of refinement given so far does not force the refined net to behave always as the original ER-net. The refined net may contain computations that are not image of any transition in the original net. Our conclusion is that this first general notion of refinement is too permisive. Therefore, we provide here a more restrictive notion, that we call proper refinement, requiring that every computation in the refined net corresponds to the image of a firing of the original net, or it can be completed with other computations in shuch a way that the obtained computation stisfies the previous condition.

Definition 4.4 (Proper Refinement of ER-nets) *Let $\mathcal{N}$ and $\mathcal{N}'$ be ER-nets. We say that $\mathcal{N}'$ is a proper refinement of $\mathcal{N}$ iff there exist an implementation morphism $h = \langle f, g \rangle : \mathcal{G}[\mathcal{N}] \rightarrow \mathcal{C}[\mathcal{N}']$ and for every computation η' in $\mathcal{N}'$, there exist computations α', β' and γ' such that*

$$\alpha' ; (\eta' \doteqdot \beta') ; \gamma = f^{\$}(\alpha)$$

where α is a computation of $\mathcal{N}$.

Notation. $h^{\$} = \langle f^{\$}, g^{\$} \rangle : \mathcal{C}[\mathcal{N}] \rightarrow \mathcal{C}[\mathcal{N}']$. is the homomorphic extension of $h = \langle f, g \rangle : \mathcal{G}[\mathcal{N}] \rightarrow \mathcal{C}[\mathcal{N}']$.

4.4 Monotonic computation graphs and refinements

The notion of computation graph may be adequated to monotonic ER-nets by restricting ourselves to monotonic computations, i.e. those that verify the monotonicity property as stated above (see section 3.3).

Definition 4.5 (Monotonic ER-Computation graph) *Given the monotonic ER-net $\mathcal{N} = (P, T, E, F, R, M_0)$, its monotonic ER-Computation graph is defined as the graph*

$$\mathcal{M}[N] = (\partial_0, \partial_1 : MC \longrightarrow (P - E^{\oplus}), M_0)$$

The set MC of monotonic computations is the set of arcs of the graph and is defined by the following rules. The rules remain the same of definition 4.1 except for the parallel composition rule where an additional condition is needed.

$$\frac{(\langle t, en, pr\rangle : en - pr) \in Y}{\langle t, en, pr\rangle : en - pr \in MC}$$

$$\frac{m \in (P - E)^{\oplus}}{m : m - m \in MC}$$

$$\frac{\alpha : m_1 - m_1' \in MC \text{ and } \beta : m_2 - m_2' \in MC \text{ and } m_1' \oplus n \qquad \oplus m_2}{\alpha \oplus \beta : m_1 \oplus m_2 - m_1' \oplus m_2' \in MC}$$

$$\frac{\alpha : m - m' \in MC \text{ and } \beta : m' - m'' \in MC}{\alpha ; \beta : m - m'' \in MC}$$

As before, some computations (arcs) are identified by the axioms of section 4.1.

In this case $(C, \oplus, 0)$ is not necessarily a monoid, the $\oplus$ operation is not more total since it is not always defined for every pair of elements of MC. Note that the obtained monotonic ER-Computation graph $\mathcal{M}[\mathcal{N}]$ may also be seen as a monotonic ER-graph, where the arcs are the computations of $\mathcal{N}$.

When refining monotonic ER-nets we would like to ensure that the monotonicity properties are maintained. This is achieved by requiring the implementation morphism to be a monotonic morphism.

Definition 4.6 (Monotonic Implementation morphism) *Let $\mathcal{N}$ and $\mathcal{N}'$ be monotonic ER-nets. A monotonic implementation morphism from $\mathcal{N}$ to $\mathcal{N}'$ is a monotonic ER-graph morphism $h : \mathcal{G}[\mathcal{N}] - \mathcal{M}[\mathcal{N}']$.*

Definition 4.7 (Monotonicity-preserving refinement of an ER-net) *Let $\mathcal{N}$ and $\mathcal{N}'$ be monotonic ER-nets. We say that $\mathcal{N}'$ is a monotonicity-preserving refinement of $\mathcal{N}$ iff there exist a monotonic implementation morphism from $\mathcal{N}$ to $\mathcal{N}'$.*

Definition 4.8 (Monotonicity-preserving proper refinement of ER-nets) *Let $\mathcal{N}$ and $\mathcal{N}'$ be monotonic ER-nets. We say that $\mathcal{N}'$ is a monotonicity preserving proper refinement of $\mathcal{N}$ iff it is proper refinement and, at the same time, a momotonicity preserving refinement.*

5 Topological Transformations

ER-net specifications can be hierarchicaly developed applying the definition of refinement given in the previous sections. Every step of refinement relates two nets at different levels of abstraction via an implementation morphism.

During the design of a system, it ussualy happens that some refinement schemata are repeatedly used. In these cases, it becames tedious to verify everytime that the refined net is actually related to the original net by an implementation morphism. Thus, we propose some transformation schemata for ER-nets, which impose some contraints on the transformed net, but, on the other hand, guarantee that the obtained net is a refinement of the original one.

Here we consider the following three topological transformations for ER-nets: (the first two have been already proposed in [Morasca 89]).

- Nondeterministic decomposition
- Sequential decomposition
- Parallel decomposition

We will show that after applying nondeterministic decomposition to a net, an equivalent net is obtained, while sequential and parallel decomposition lead to a net which is a proper refinement of the original one. We are not claiming that this set of transformations is enough for describing all possible refinement schemata. We rather believe in an open environment where the designer has the possibility to define or tailor his own transformation schemata.

5.1 Nondeterministic decomposition

The nondeterministic decomposition allows the designer to make explicit the nondeteminism present in the relation associated with a transition by dividing the relation in a set of more "specific" ones and by associating each of these relations to a different new transition (see figure 4). The nondeterministic decomposition is in fact similar to the split operation as defined in [Morasca 89].

Definition 5.1 (Nondeterministic Decomposition) *Let* $\mathcal{N} = (P, T, E, F, R, M_0)$, *be an ER-net; let* $t \in T$ *and* $\{\mathcal{R}_{t,i}\}\, i \in I$ *be such that*

$$\forall i \in I\,(\mathcal{R}_{t,i} \subseteq R(t))\; and \bigcup_{i \in I} \mathcal{R}_{t,i} \subseteq R(t)\; and \bigcup_{i \in I} \mathcal{R}_{t,i} = R(t)$$

i.e. it is a family of subsets of $R(t)$ *covering* $R(t)$. *The net* $\mathcal{N}' = (P', T', E', F', R', M_0')$ *is obtained from* $\mathcal{N}$ *by nondeterministic decomposition of* t *under the set* $\{\mathcal{R}_{t,i}\}$ *iff:*

$$\begin{aligned}
P' &= P \\
T' &= (T - \{t\}) \cup \{split(t,i)/i \in I\} \\
E' &= E \\
F' &= F - \{\langle a,b\rangle / \langle a,b\rangle \in F \; and \; ((a = t) \; or \; (b = t))\} \cup \\
&\quad \{\langle a, split(t,i)\rangle / a \in {}^\bullet t \; i \in I\} \cup \{\langle split(t,i), b\rangle / b \in t^\bullet \; i \in I\} \\
R' &\quad such\ that\ \forall t' \in (T - \{t\}) R'(t') = R(t') \\
&\quad and\ R'(split(t,i)) = \mathcal{R}_{t,i} \\
M_0' &= M_0
\end{aligned}$$

Theorem 5.1 *Let* $\mathcal{N}$ *be an ER-net and* $\mathcal{N}'$ *obtained from* $\mathcal{N}$ *using nondeterministic decomposition (split) of transition* $t \in T$. *Then,* $\mathcal{N} \equiv \mathcal{N}'$.

Proof: The ER-graph structure is not modified by the nondeterministic decomposition (at most there is a change of names in the firings that involve the splitted transition). Therefore, there is an obvious isomorphism between $\mathcal{G}[\mathcal{N}]$ and $\mathcal{G}[\mathcal{N}']$. □

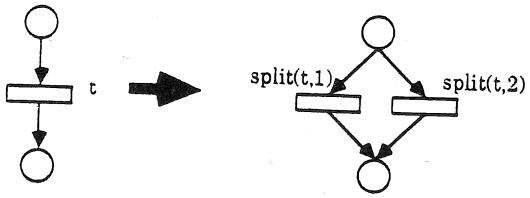

Figure 4: Nondeterministic Decomposition

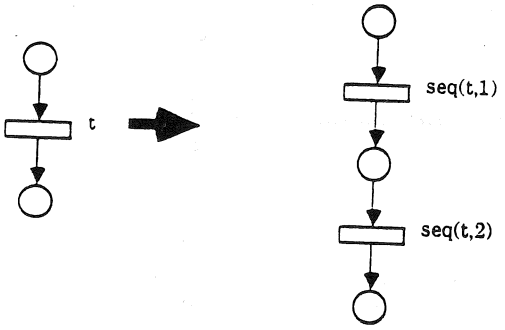

Figure 5: Sequential Decomposition

5.2 Sequential Decomposition

The intuitive idea of a sequential decomposition is to replace a transition with a sequence of two transitions (the sequence can be easily extended to more than two transitions). A set of new places is needed to connect the new transitions.

If the refined transition is $t \in T$, we will call the new ones $seq(t,1)$ and $seq(t,2)$ (see figure 5). Now the main question is what are the relations associated with these new transitions. It seems natural to request that the composition of the two new relations gives as result the relation originally associated with t. However, in order to obtain a proper refinament from a sequential decomposition, we need stronger conditions on the relations associated with the new transitions. These conditions are stated in the following restrictive version of the composition of relations.

Definition 5.2 (Restricted Composition of Relations) *Let P_i, E_i $i = 1 \ldots 3$ be ar-*

bitary sets of places and environments respectively and let $R_1 \subseteq (P_1 \times E_1)^{\ominus} \times (P_3 \times E_3)^{\ominus}$ *and* $R_2 \subseteq (P_3 \times E_3)^{\oplus} \times (P_2 \times E_2)^{\oplus}$ *be two relations. We say that* R *is the restricted composition of* R_1 *and* R_2 *and we write* $R = R_1 \odot R_2$ *iff*

1. $R = R_1 \circ R_2$
2. $\Pi_2(R_1) = \Pi_1(R_2)$. *We will call this set the set of Intermediate Markings (IM).*
3. *If the set of intermediate markings is* $IM = \{m_1, m_2, \ldots, m_i, \ldots\}$ *then* $m_j \in IM$ *cannot be written in terms of the rest of the intermediate markings* $IM - \{m_j\}$. *In other words,* $m_j = \bigoplus_i k_i m_i \Longrightarrow k_i = 0$ *for* $i \neq j$ *and* $k_j = 1$

Intuitively, the last requirement avoids the possibility of computations in the refined net that may have no counterpart in the original net. Now we are ready to define a topological transformation based on the sequential decomposition of a transition. The formal construction is the following:

Definition 5.3 (Sequential Decomposition) *Let* $N = (P, T, E, F, R, M_0)$ *be an ER-net and* $t \in T$ *a transition. The net* $N' = (P', T', E', F', R', M_0')$ *is obtained from* N *by sequential decomposition of* t *iff:*

$$
\begin{array}{lcl}
P' & = & P \cup P'' \\
T' & = & (T - \{t\}) \cup \{seq(t,1), seq(t,2)\} \\
E' & = & E \cup E'' \\
F' & = & F - \{\langle a,b\rangle / \langle a,b\rangle \in F \; and \; ((a = t) \; or \; (b = t))\} \cup \\
 & & \{\langle a, seq(t,1)\rangle / a \in {}^{\bullet}t\} \cup \{\langle seq(t,1), b\rangle / b \in P''\} \cup \\
 & & \{\langle b, seq(t,2)\rangle / b \in P''\} \cup \{\langle seq(t,2), c\rangle / c \in t^{\bullet}\} \\
R' & & such \; that \; \forall t' \in (T - \{t\}) R'(t') = R(t') \\
 & & and \; R'(seq(t,1)) = \mathcal{R}_{t,1} \; R'(seq(t,2)) = \mathcal{R}_{t,2} \; where \; \mathcal{R}_{t,1} \odot \mathcal{R}_{t,2} = R(t) \\
M_0' & & such \; that \; \forall p \in P M_0'(p) = M_0(p) and \forall p \in P'' M_0'(p) = 0
\end{array}
$$

P'' is a new set of places and E'' is a new set of environments.

Note that the net N', obtained by sequential decomposition of transition t from the ER-net N, is also an ER-net.

The following lemma is used in the proof of theorem 6.2

Lemma 5.1 *Let N' be an ER-net obtained from another ER-net N, by sequential decomposition of transition t. Let m be a reachable marking restricted to the set of new places, i.e., $m \in (P'' \times E')^{\oplus}$. Then m can be written as a linear combination of intermediate markings.*

Proof: Since the only way an environment may appear in a place $p \in P''$ is by the firing of transition $seq(t,1)$, the resulting marking is a combination of the set $\Pi_2(R(seq(t,1)))$, which is equal to the set of intermediate markings IM. In symbols:

$$m = \bigoplus_i k_i m_i$$

where the k_i are natural numbers and the $m_i \in IM$. □

Theorem 5.2 *Let $\mathcal{N}$ be an ER-net and $\mathcal{N}'$ obtained from $\mathcal{N}$ by sequentially decomposing transition $t \in T$. Then, $\mathcal{N}'$ is a proper refinement of $\mathcal{N}$. If, in addition, $\mathcal{N}$ is a monotonic ER-net and the relations $\mathcal{R}_{t,1}$ and $\mathcal{R}_{t,2}$ of the above definition are monotonic, then $\mathcal{N}'$ is a monotonicity preserving proper refinement of $\mathcal{N}$.*

Proof: We first show an (obvious) implementation morphism $h = \langle f, g \rangle$ between $\mathcal{N}$ and $\mathcal{N}'$. This morphism maps all firings of $\mathcal{N}$ to the corresponding firings of $\mathcal{N}'$, except for those where transition t is involved, which are mapped to the sequential composition of two suitable firings of $\mathcal{N}'$. Formally[10]:

$$\forall m \in (P \times E)^{\oplus}\ g(m) = m$$

$$\forall \langle t', en, pr \rangle \in Y \text{ such that } t' \in T - \{t\}\ f(\langle t', en, pr \rangle) = \langle t', en, pr \rangle$$

$$\forall \langle t, en, pr \rangle \in Y\ f(\langle t, en, pr \rangle) = \langle seq(t,1), en, m \rangle; \langle seq(t,2), m, pr \rangle \text{ where } m \in IM$$

In order to see that this is a proper refinement we have to verify that each computation of $\mathcal{N}'$ is part of a computation which is the homomorphic image of some other computation in $\mathcal{N}$. Let $\eta' : m \multimap m'$ be a computation of $\mathcal{N}'$. We consider the following cases:

1. Neither m nor m' involve new places P''.
2. The marking m of $\mathcal{N}'$ has a corresponding marking in $\mathcal{N}$, but m' does not (it involves places from P'').
3. The marking m' of $\mathcal{N}'$ has a corresponding marking in $\mathcal{N}$, but m does not (it involves places from P'').
4. Both m and m' involve places of P'', thus they have no corresponding marking in $\mathcal{N}'$ via the implementation homomorphism.

The first is the trivial case. The defined morphism assigns $g(m) = m$ and $g(m') = m'$, so the computation $\alpha : m \multimap m'$ is such that $f^{\$}(\alpha) = \eta'$. The other three cases are more interesting. Here we will only consider case 2, since the remaining two require a similar proof.

We have to show that there exist computations α', β' and γ' such that $\alpha'; \eta' \oplus \beta'; \gamma' = f^{\$}(\alpha)$. We can write m' as the union (sum) of a marking belonging to $(P \times E)^{\oplus}$ and a marking belonging to $(P'' \times E')^{\oplus}$ in the following way: $m' = m_P \oplus m_{P''}$. Now, applying lemma 6.1, $m_{P''}$ can be written as $m_{P''} = \bigoplus_i k_i m_i$.

Let r_i be a tuple in $R'(seq(t,2))$ such that $r_i = (en_i, pr_i)$ and $en_i = m_i$ (it always exists, because $IM = \Pi_1(R(seq(t,2)))$). Then the computations:

$$\begin{aligned} \alpha' &= m : m \multimap m \\ \beta' &= 0 : 0 \multimap 0 \text{ and} \\ \gamma' &= m_P; (k_1\langle seq(t,2), r_1 \rangle \oplus \ldots \oplus k_n\langle seq(t,2), r_n \rangle) : m' \multimap m'' \end{aligned}$$

where $m'' = m_P \oplus k_1 pr_1 \oplus \ldots \oplus k_n pr_n$, are such that $\alpha'; \eta' \oplus \beta'; \gamma' = f^{\$}(\alpha)$ and α is a computation of $\mathcal{N}$.□

[10] Note that if $m \in (P \times E)^{\oplus}$ then $m \in (P' \times E')^{\oplus}$

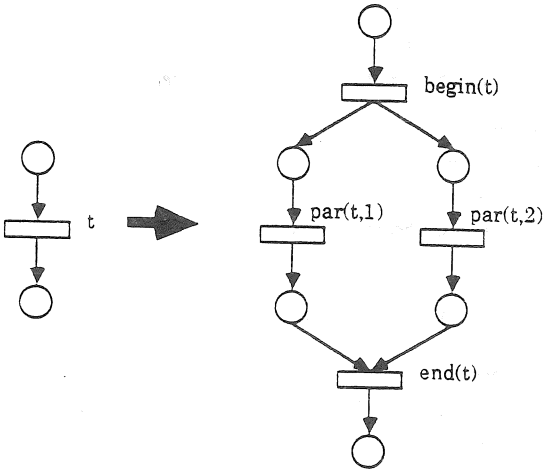

Figure 6: Parallel Decomposition

5.3 Parallel Decomposition

The intuitive idea of the parallel decomposition is to replace a transition by a set of 'more primitive' transitions that all together, carried out in parallel, produce the same effect of the original transition. In order to follow the requirements so as the decomposed net is a proper refinement of the given one, we include a transition at the beginnig and one at the end of the parallel decomposition (as shown in figure 6). Again the main question to be answered is what are the requirements that the relations associated with the new transitions have to accomplish. We begin by showing that the relations of the $par(t, i)$ transitions may be composed in one big relation:

Definition 5.4 (Parallel composition of relations) *Let P_i, E_i $i = 1 \ldots 4$ be arbitrary disjoint sets of places and environments respectively and let $R_1 \subseteq (P_1 \times E_1)^{\oplus} \times (P_2 \times E_2)^{\ominus}$ and $R_2 \subseteq (P_3 \times E_3)^{\oplus} \times (P_4 \times E_4)^{\ominus}$ be two relations. We say that R is the parallel composition of R_1 and R_2 and we write $R = R_1 \parallel R_2$ iff $R \subseteq (P_1 \cup P_3 \times E_1 \cup E_3)^{\oplus} \times (P_2 \cup P_4 \times E_2 \cup E_4)^{\oplus}$ and*

$$\begin{aligned} R \;=\; & \{(m_1 \oplus m_3, m_2 \ominus m_4)/ \\ & m_1 \in (P_1 \times E_1)^{\oplus}\, m_2 \in (P_2 \times E_2)^{\oplus}\, m_3 \in (P_3 \times E_3)^{\oplus}\, m_4 \in (P_4 \times E_4)^{\oplus} \\ & and\, (m_1, m_2) \in R_1\, (m_3, m_4) \in R_2\} \end{aligned}$$

Definition 5.5 (Parallel Decomposition) *Let N=(P, T, E, F, R, M_0) an ER-net and $t \in T$ a transition. A new net ER-net N'=(P', T', E', F', R', M_0') is obtained by parallel decomposition of N with the following contruction: Let $I = \{1 \ldots n\}$ a set of indexes and the families of sets indexed by I*

$$T_i \;=\; \{par(t, i)/i \in I\}$$

$$
\begin{aligned}
P_i &= \{inplace(t,i.j)/i \in I, j = 1,\ldots,m_i,\ m_i \in \mathcal{N}\} \cup \\
&\quad \{outplace(t,i.j)/i \in I, j = 1,\ldots,n_i,\ n_i \in \mathcal{N}\} \\
F_i &= \{\langle a,b\rangle / a = inplace(t,i,j),\ b = par(t,i),\ j = 1,\ldots,m_i\} \cup \\
&\quad \{\langle a,b\rangle / a = par(t,i),\ b = outplace(t,i,j),\ j = 1,\ldots,n_i\}
\end{aligned}
$$

Now we define $\mathcal{N}'$ in the following way:

$$
\begin{aligned}
P' &= P \cup (\bigcup_{i\in I} P_i) \\
T' &= (T - \{t\}) \cup \{begin(t), end(t)\} \cup (\bigcup_{i\in I} T_i) \\
E' &= E \cup E'' \\
F' &= F - \{\langle a,b\rangle / \langle a,b\rangle \in F \ and \ ((a = t) or (b = t))\} \cup (\bigcup_{i\in I} F_i) \cup \\
&\quad \{\langle a, begin(t)\rangle / a \in {}^\bullet t\} \cup \{\langle end(t), b\rangle / b \in t^\bullet\} \cup \\
&\quad \{\langle begin(t), b\rangle / b \in {}^\bullet par(t,i), i \in I\} \cup \{\langle a, end(t)\rangle / a \in par(t,i)^\bullet, i \in I\} \\
R' &\quad such\ that\ \forall t' \in T - \{t\}\ R'(t') = R(t')\ and \\
&\quad R'(begin(t)),\ R'(end(t)),\ R'(par(t,i))\ i \in I\ are\ such\ that \\
&\quad R'(begin(t)) \oplus (\|_{i\in I}\ R'(par(t,i))) \oplus R'(end(t)) = R(t) \\
M_0' &\quad such\ that\ \forall p \in P\ M_0'(p) = M_0(p)\ and \forall p \in \bigcup_{i\in I} P_i,\ M_0'(p) = 0
\end{aligned}
$$

E'' is a new set of environments.

Theorem 5.3 *Let $\mathcal{N}$ be an ER-net and $\mathcal{N}'$ obtained from $\mathcal{N}$ by a parallel decomposition of transition $t \in T$. Then, $\mathcal{N}'$ is a proper refinement of $\mathcal{N}$. If, in addition, $\mathcal{N}$ is a monotonic ER-net and the relations $R'(begin(t))$, $R'(end(t))$ and $R'(par(t,i))$ of the above definition are monotonic, then $\mathcal{N}'$ is a monotonicity preserving proper refinement of $\mathcal{N}$.*

Proof: We will only show the implementation morphism, since the rest of the proof is analogous to the one for sequential decomposition. The implementation morphism $h = \langle f, g\rangle$ between $\mathcal{N}$ and $\mathcal{N}'$ is given by:

$$\forall m \in (P \times E)^{\oplus}\ g(m) = m$$

$$\forall \langle t', en, pr\rangle \in Y\ such\ that\ t' \in T - \{t\}\ f(\langle t', en, pr\rangle) = \langle t', en, pr\rangle$$

$$
\begin{aligned}
&\forall \langle t, en, pr\rangle \in Y\ f(\langle t, en, pr\rangle) = \\
&\langle begin(t), en, m\rangle; (\|_i\ \langle par(t,i), m_i, m_i'\rangle); \langle end(t), m', pr\rangle
\end{aligned}
$$

□

5.4 An Example

Here we show how the cake recipie, presented in section 2.2 may be hierarchicaly developed, via a sequence of formal refinements. We begin with the ER-net of Figure 7, where the realation associated with the transition is[11]:

[11] MkCk stands for MakeCake, and PrDgh, for Prepare Dough.

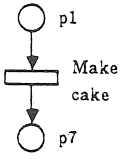

Figure 7: A high level specification of the recipie.

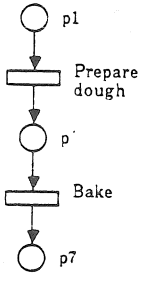

Figure 8: The recipie after sequencial decomposition.

$$R(MkCk) = \neg\{[4(p_1,(egg,t_e)) \ominus (p_1,(flower,t_f)) \ominus (p_1,(butter,t_b)) \ominus (p_1,(sugar,t_s)), (p_7,(cake,t_c))]\} \; where \; t_c = Max(t_e,t_f,t_b,t_s+\alpha \; and \; \alpha \in \{50,55\})$$

Actually, the recipie may be seen as composed of two sequencial steps. The first step involves the preparation of the dough, while the second one is baking. Therefore, it seems natural to apply sequential decomposition to obtain the net of Figure 8, with the following associated relations.

$$R(PrDgh) = \{[4(p_1,(egg,t_e)) \ominus (p_1,(flower,t_f)) \ominus (p_1,(butter,t_b)) \ominus (p_1,(sugar,t_s)), (p',(dough,t_d))]\} \; where \; t_d = Max(t_e,t_f,t_b,t_s+\alpha \; and \; \alpha \in \{10,15\})$$

$$R(Bake) = \{[(p',(dough,t_d)),(p_7,(cake,t_d+40))]\}$$

Now we concentrate on the preparation step, which consists of two independent actions that may go in parallel: mix the solid ingredients and beat the eggs. Thus, we use parallel decomposition to refine the specification (recipie). The result is Figure 9 with the associated relations as follows.

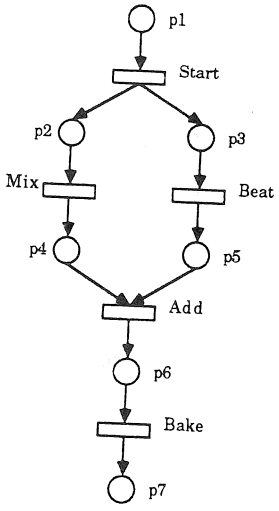

Figure 9: The recipie after parallel decomposition.

$$
\begin{aligned}
R(Start) &= \{[4(p_1,(egg,t_e)) \oplus (p_1,(flower,t_f)) \oplus (p_1,(butter,t_b)) \oplus (p_1,(sugar,t_s)),\\
&\quad (p_2,(flower,t_1)) \oplus (p_2,(butter,t_1)) \oplus (p_2,(sugar,t_1)) \oplus 4(p_3,(egg,t_1))]\}\\
&\quad where\, t_1 = Max(t_e,t_f,t_b,t_s)\\
R(Mix) &= \{[(p_2,(flower,t_f)) \oplus (p_2,(butter,t_b)) \oplus (p_2,(sugar,t_s)),(p_4,(mixture,t_2))]\}\\
&\quad where\, t_2 = Max(t_f,t_b,t_s)+5\\
R(BtEgg) &= \{[4(p_3,(egg,t_e)),(p_5,(beateneggs,t_e+\alpha))]\}\ where\, \alpha \in \{5,10\}\\
R(Add) &= \{[(p_4,(mixture,t_m)) \oplus (p_5,(beateneggs,t_{be})),(p_6,(dough,Max(t_m,t_{be})))]\}\\
R(Bake) &= \{[(p_6,(dough,t_d)),(p_7,(cake,t_d+40))]\}\\
M_0 &= 4(p_1,(egg,0)) \oplus (p_1,(flower,0)) \oplus (p_1,(butter,0)) \oplus (p_1,(sugar,0))
\end{aligned}
$$

Finally, we can apply nondeterministic decomposition to the action "BeatEggs" and obtain the net of Figure 1. Here we have only made obvious the nondeterminism of the transition, obtaining an equivalent net (not a refinament).

6 Conclusions

In this paper we proposed a formal definition of equivalence and refinement for ER-nets, a kernel formalism adequated for modelling concurrent and real-time systems. In particular we have presented different schemata of transformation for ER-nets that may be seen as a controlled way of doing refinements. Sequential and parallel decomposition may become useful mechanisms for the designer during the developement of a system specification. Certainly, some other transformations may be defined based on the given notion of refinement or directly in terms of the proposed ones.

The notion of refinement and the topological transformations, may be extended to allow also the coarsening of an ER-net by means of a higher abstract view. It would be also possible to define analysis techniques (such as symbolic excecution), that can be carried out at different abstraction levels.

Acknowledgements

We would like to thank Professor Carlo Ghezzi, who introduced us to the subject. We are also grateful to Miguel Felder, who made interesting suggestions and helped with the bibliography. The first author wish to thank his friends Marcelo Fernandez, Marcelo Fiore, Gabriela Anastasi and Sergio Yovine for their comments and fruitful conversations about this work.

References

[Berthomieu 83] Berthomieu B., Menasche M., An Enumerative Approach for Analyzing Petri Nets, *Information Processing 83*, R. E. A. Mason (Ed.), Elservier Science Pub. B.V., North Holland, 1983.

[Degano 89] Degano P. Meseguer J., Montanari U., Axiomatazing Net Computations and Processes, *4th. Annual Symposium on Logic in Computer Science*, Asilomar, CA, U.S.A., June 1989.

[Felder 88] Felder M., Some Properties when Time is Introduced in Petri Nets, Graduation Thesis, ESLAI, December 1988.

[Feuerstein 88] Feuerstein E., Timed Petri Nets are Monoids, Graduation Thesis, ESLAI, December 1988.

[Genrich 86] H. J. Genrich, Predicate Transition Nets, in *Advances in Petri Nets 1986*, LNCS 254-255, Springer Verlag 1987.

[Ghezzi 89] Ghezzi C., Mandrioli D. Morasca S., Pezzè M., On Introducing Time in Petri Nets, *IEEE 5th. International Workshop on Software Specification and Design*, Pittsburgh, U.S.A., May 1989.

[Ghezzi 89b] GhezziC., Mandrioli D., Morasca S., Pezzè M., Symbolic Excecution of Concurrent Programs Using Petri Nets, *Computer Languages*, vol. 14, No. 4, 1989.

[Ghezzi 91] GhezziC., Mandrioli D., Morasca S., Pezzè M., A Unified High Level Petri Net Formalism for Time Critical Systems, *IEEE Transactions of Software Engineering*, March 1991.

[Jensen 86] F. Jensen, Coloured Petri Nets, in *Advances in Petri Nets 1986*, LNCS 254-255, Springer Verlag 1987.

[Harel 86] Harel D., A Visual Formalism for Complex Systems, *Science of Computer Programming*, vol. 8, 1986.

[Meseguer 88] Meseguer J., Montanari H., Petri Nets are Monoids. SRI-CLS-88-3, SRI Proyect No. 1243 and 4415, January 1988.

[Morasca 89] Morasca S., Pezzè M., A Net Based Model for Time-Dependent Systems, Internal report No. 89-005, Dipartamento di Elettronica, Politecnico di Milano, Milan, Italy, 1989.

[Peterson 77] Peterson J., Petri Nets, Computing surveys, Vol. 9, No. 3, September 1977.

[Reisig 82] Reisig W., *Petri Nets, an introduction*, ETACS Monographs on Theoretical Computer Science; Springer Verlag 1985.