Direct Product Representation of Labelled Free Choice Nets

Ramchandra Phawade
The Institute of Mathematical Sciences,
CIT Campus, Chennai *600113*, India
India

ABSTRACT

Hack's theorem [7] shows that every live and bounded free choice net is covered by S-components [5, 2].
If the net is labelled, with a distribution of the alphabet into possibly overlapping subalphabets for components, the question arose whether the net can be decomposed into a product of automata, with several possible definitions of product and of equivalence [1, 13, 11, 4]. Zielonka showed [15] that there is a live and 1-bounded net which is not direct product representable.
We give a straightforward example of a live and 1-bounded labelled free choice net which is not direct product representable, we do not know of any earlier such example. We give two sufficient conditions for 1-bounded labelled free choice nets to be direct product representable. In the other direction, we give two sufficient conditions on products of automata using which we can construct labelled free choice nets. In [14] expressions corresponding to such products has been recently given.

General Terms:

Concurrency, Formal Languages

Keywords:

Petri nets, Kleene Theorem, Direct Products, Synchronization, Traces

1. INTRODUCTION

One of the early ideas connecting Petri nets to syntax was developed in an MIT Masters' thesis [7], where Hack introduced the subclass of free choice nets and showed that every live and k-bounded free choice net (for arbitrary k) has a decomposition into "state machines". (See the book [5] for a detailed treatment of free choice nets including Hack's theorem.) Since Kleene's theorem (which applies to 1-bounded state-machines) provides us with a syntactic representation of the latter, this can be seen as solving a large part of the problem.
But unlabelled Petri nets are not abstract enough to represent formal languages, even a finite language like $\{a, aa\}$ is not representable. So one has to work with labelled Petri nets. One idea is to think of the language of a distribution into several state machines as that defined by a product of these automata [11, 4]. Products are of many kinds [1], here we consider the most "direct", where the language is defined by a synchronized shuffle as in process calculi [3, 8]. Unlike process calculi we do not consider labelling as applying a global renaming operator.
A net with only two transitions with disjoint neighbourhoods and the same label a recognizes the language $\{aa\}$, but so does a net with two such transitions sequentially arranged. Mazurkiewicz introduced the theory of traces to explicitly specify concurrency [6]. Zielonka introduced [15] a more direct "location" mechanism which was explicated by Mukund and Sohoni [13]. We identify a new condition which appears to have been overlooked earlier because most work concentrated on unlabelled nets.
Zielonka's paper showed 1-bounded labelled nets which are not direct product representable.

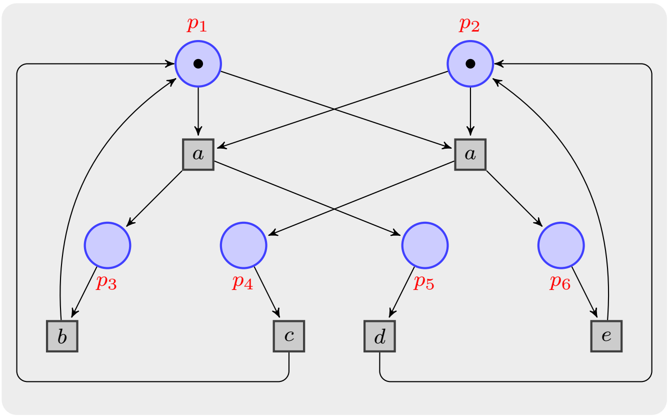

Fig. 1. Live and 1-bounded, labelled free choice net

This easy example is our first contribution in this paper. For final marking $\{1, 2\}$, the language accepted by live and 1-bounded labelled (extended) free choice net shown in Figure 1, is $\{abd, adb, ace, aec\}^*$ over the distribution $\Sigma = (\Sigma_1 = \{a, b, c\}, \Sigma_2 = \{a, d, e\})$. It is a Mazurkiewicz trace language. In Proposition 3 we formally show that this language is not accepted by any direct product over Σ.
In the rest of the paper, we proceed to identify a couple of sufficient conditions for decomposition into product automata, and con-

versely, a couple of sufficient conditions for constructing labelled free choice nets from product automata.

2. PRODUCT SYSTEMS OVER A DISTRIBUTION

Let Σ be a finite alphabet and Σ^* be the set of all words over alphabet Σ, including the empty word ϵ. A language over an alphabet Σ is a subset $L \subseteq \Sigma^*$. The projection of a word $w \in \Sigma^*$ to a set $\Delta \subseteq \Sigma$, denoted as $w{\downarrow_\Delta}$, is defined by: $\epsilon{\downarrow_\Delta} = \epsilon$ and

$$(a\sigma){\downarrow_\Delta} = \begin{cases} a(\sigma{\downarrow_\Delta}) & \text{if } a \in \Delta, \\ \sigma{\downarrow_\Delta} & \text{if } a \notin \Delta. \end{cases}$$

DEFINITION 1. *Let Loc denote the set $\{1, 2, \ldots, k\}$. A distribution of Σ over Loc is a tuple of nonempty sets $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$ with $\Sigma = \bigcup_{1 \le i \le k} \Sigma_i$. For each action $a \in \Sigma$, its locations are the set $loc(a) = \{i \mid a \in \Sigma_i\}$. Actions $a \in \Sigma$ such that $|loc(a)| = 1$ are called local, otherwise they are called global.*

A distribution induces an independence relation and thus a Mazurkiewicz trace language [6, 12]. For us the key property in terms of locations is that languages can be described in terms of the synchronized shuffle $L = L_1 \| \ldots \| L_k$ [3], defined by $w \in L$ iff for all $i \in \{1, \ldots, k\}, w{\downarrow_{\Sigma_i}} \in L_i$.
Fix a distribution $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$ of Σ. Product systems are defined over this alphabet.

DEFINITION 2. *A sequential system over a set of actions Σ_i is a tuple $A_i = \langle P_i, \to_i, G_i, p_i^0 \rangle$ where P_i are called **places**, $G_i \subseteq P_i$ are final places, $p_i^0 \in P_i$ is the initial place, and $\to_i \subseteq P_i \times \Sigma_i \times P_i$ is a set of **local moves**.*

Let $\to_a^i$ denote the set of all a-labelled moves in the sequential system A_i.
A run of the sequential system A_i on word $w = a_1 a_2 \ldots a_n$ is a sequence $p_0 a_1 p_1 a_2, \ldots, a_n p_n$, from set $(P_i \times \Sigma_i)^* P_i$, such that $p_0 = p_i^0$ and for each $j \in \{1, \ldots, n\}$, $p_{j-1} \xrightarrow{a_j} p_j$. This run is said to be accepting if $p_n \in G_i$. The sequential system A_i accepts word w, if there is at least one accepting run of A_i on w. The language $Lang(A_i)$ of sequential system A_i is defined as $Lang(A_i) = \{w \in {\Sigma_i}^* | w \text{ is accepted by } A_i\}$.

DEFINITION 3. *Let $A_i = \langle P_i, \to_i, G_i, p_i^0 \rangle$ be a sequential system over alphabet Σ_i for $1 \le i \le k$. A **product system** A over the distribution $\Sigma = (\Sigma_1, \ldots, \Sigma_k)$ is a tuple $\langle A_1, \ldots, A_k \rangle$. A is **deterministic for global actions** if for every global action a, every place is the source of at most one a-transition.*

Let $\Pi_{i \in Loc} P_i$ be the set of product states of A. We use $R[i]$ for the i'th projection of a product state R in A_i, and $R{\downarrow}I$ for the projection to $I \subseteq Loc$.
The initial product state of A is $R^0 = (p_1^0, \ldots, p_k^0)$, while $G = \Pi_{i \in Loc} G_i$ denotes the final states of A.
Let $\Rightarrow_a = \Pi_{i \in loc(a)} \to_a^i$. The set of global moves of A is $\Rightarrow = \bigcup_{a \in \Sigma} \Rightarrow_a$. Then for a global move

$$g = \langle \langle p_{l_1}, a, p'_{l_1} \rangle, \ldots \langle p_{l_m}, a, p'_{l_m} \rangle \rangle \in \Rightarrow_a, \; loc(a) = \{l_1, \ldots, l_m\},$$

we write $g[i]$ for $\langle p_i, a, p'_i \rangle$, the projection to A_i, $i \in loc(a)$, and define pre-places$(g) = \{p_i \mid i \in loc(a)\}$ and post-places$(g) = \{p'_i \mid i \in loc(a)\}$.
Please note that the set of product states as well as the global moves are not explicitly constructed when a product system is given as input to some algorithm.

2.1 Properties of Product Systems

The first property for a product system identifies synchronizations which will come together into a cluster of a free choice net. It can be checked in PTIME by counting local moves with the same label. We also define another stronger property.

DEFINITION 4. *For global $a \in \Sigma$, an a-matching is a subset of tuples $\Pi_{i \in loc(a)} P_i$ such that, each place $p \in P_i, i \in loc(a)$ having an outgoing local a-move(I.e., if $\exists \langle p, a, q \rangle \in \to_i$), appears in exactly one tuple of a-matching. When two places p and p' appear in a tuple of a-matching, then we say that they are matched on action a. We say a product state R is in an a-matching if its projection $R{\downarrow}loc(a)$ is in the matching. A product system is said to have **matching of labels** if for all global $a \in \Sigma$, there is an a-matching such that for all $i, j \in loc(a), \langle p, a, q \rangle \in \to_i$, the pre-place p is matched to a pre-place p' such that there exists a local a-move $\langle p', a, q' \rangle$ in $\to_j$. A product system A is said to have **separation of labels** if for all $i \in Loc$, whenever $\langle p, a, p' \rangle, \langle q, a, q' \rangle \in \to_i$ then $p = q$.*

A system having separation of labels property may have many a-labelled moves in each of its sequential component, but all of them are outgoing moves from an unique place in it.
The next property is necessary for product systems to represent free choice in equivalent nets. In our earlier paper [10] we used the definition of an FC-product below. The definition of FC-matching product is a generalization since conflict-equivalence is not required for all a-moves uniformly but refined into smaller equivalence classes depending on the matching.

DEFINITION 5. *In a product system, we say the local move $\langle p, a, q_1 \rangle \in \to_i$ is **conflict-equivalent** to the local move $\langle p', a, q'_1 \rangle \in \to_j$, if for every other local move $\langle p, b, q_2 \rangle \in \to_i$, there is a local move $\langle p', b, q'_2 \rangle \in \to_j$ and, conversely, for moves from p' there are moves from p. In a product system, we say that a local state $p \in P_i$ is **conflict-equivalent** to a local state $p' \in P_j$, if for some action $a \in \Sigma$, p have an outgoing local move on a i.e., $\exists \langle p, a, q_1 \rangle \in \to_i \implies \exists \langle p', a, q'_1 \rangle \in \to_j$, and, conversely, moves from p' are matched by moves from p.
An a-matching of a product system is said to be **conflict-equivalent** if any two places which are matched on action a are conflict-equivalent. We call a product system an **FC-matching product** if it has a conflict-equivalent matching for each global action a.
We call $A = \langle A_1, \ldots, A_k \rangle$ an **FC-product** if for every $a \in \Sigma$ with $|loc(a)| > 1$, every a-labelled move in A_i is conflict-equivalent to every a-labelled move in A_j, where $i, j \in loc(a)$.*

A system having a conflict-equivalent matching is a weaker condition than the system being conflict-equivalent. Checking that a system is an FC-product or an FC-matching product is in PTIME because one makes a pass through all transitions with the same locations, computing for each pre-state which partition it falls into.

PROPOSITION 1 [14]. *Let A be an FC-matching product system. For any i, if there exist local moves $\langle p, a, p' \rangle$, $\langle p, b, p'' \rangle$ in $\to_i$, then $loc(a) = loc(b)$.*

PROOF. Since p has an outgoing a-move, p belongs to some tuple of a-matching. If $j \in loc(a)$, then in this tuple there exists a state $q \in P_j$, which has an outgoing a-move. Since A is an FC-matching product, a-matching is conflict-equivalent. And, as states p and q appear in a tuple of a-matching, these states are conflict-equivalent. Therefore there exists a local move $(q, b, q') \in \to_j$. This implies that $j \in loc(b)$. $\square$

2.2 Language of a Product System

Now we describe runs of A over some word w by associating product states with prefixes of w: the empty word is assigned initial product state R^0, and for every prefix va of w, if R is the product state reached after v and Q is reached after va where, for all $j \in loc(a)$, $\langle R[j], a, Q[j]\rangle \in \rightarrow_j$ and for all $j \notin loc(a), R[j] = Q[j]$. Let $pre(a) = \{R \mid \exists Q, R \overset{a}{\Rightarrow} Q\}$.
A run is said to be accepting if the product state reached after w is in a final product state $G \subseteq G_1 \times \ldots G_k$. A direct product system has final states $G = G_1 \times \ldots G_k$. We define the language $Lang(A)$ of product system A, as the words on which the product system has an accepting run.
We use the following characterization [4, 11, 12] of direct product languages.

PROPOSITION 2. *L is a direct product language iff $L = \{w \in \Sigma^* \mid \forall i \in \{1, \ldots, k\},\ \exists u_i \in L$ such that $w{\downarrow_{\Sigma_i}} = u_i{\downarrow_{\Sigma_i}}\}$.*

The next definition is semantic, new to this paper and not easy to check (in PSPACE). If a system has separation of labels, the property obviously holds.

DEFINITION 6. *A run of A is said to be **consistent with a matching of labels** if for all global actions a and every prefix of the run $R^0 \overset{v}{\Rightarrow} R \overset{a}{\Rightarrow} Q$, the pre-places $R{\downarrow}loc(a)$ are in the matching.*

Here is our first result.

PROPOSITION 3. *Let $L = \{abd, adb, ace, aec\}^*$ be a language defined over distribution $\Sigma = (\Sigma_1 = \{a, b, c\}, \Sigma_2 = \{a, d, e\})$. Then there is no product system over Σ which represents this language.*

PROOF. Let $w = abeacd$ for which $w{\downarrow_{\Sigma_1}} = abac$ and $w{\downarrow_{\Sigma_2}} = aead$. Now consider $u_1 = abdace$ for which $u_1{\downarrow_{\Sigma_1}} = abac$ and $u_2 = aceabd$ for which $u_2{\downarrow_{\Sigma_2}} = aead$. Since both $u_1, u_2 \in L$ using characterization given in Proposition 2 we get $w \in L$, which is a contradiction. □

3. NETS

Fix a distribution $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$ of Σ. Labelled nets are defined over this alphabet.

DEFINITION 7. *A labelled net N is a tuple (S, T, F, λ), where S is a set of places, T is a set of transitions labelled by the function $\lambda : T \to \Sigma$ and $F \subseteq (T \times S) \cup (S \times T)$ is the flow relation. It will be convenient to define $loc(t) = loc(\lambda(t))$.*

Elements of $S \cup T$ are called nodes of N. Given a node z of net N, set ${}^\bullet z = \{x \mid (x, z) \in F\}$ is called pre-set of z and $z^\bullet = \{x \mid (z, x) \in F\}$ is called post-set of z. Given a set Z of nodes of N, let ${}^\bullet Z = \bigcup_{z \in Z} {}^\bullet z$ and $Z^\bullet = \bigcup_{z \in Z} z^\bullet$. We consider only those nets in which every transition has nonempty pre-set and post-set.

DEFINITION 8. *Let $N' = (S \cap X, T \cap X, F \cap (X \times X))$ be a subnet of net $N = (S, T, F)$, generated by a nonempty set X of nodes of N. N' is called a **component** of N if,*

—*For each place s of X, ${}^\bullet s, s^\bullet \subseteq X$ (the pre- and post-sets are taken in N),*
—*For all transitions $t \in T$, we have $|{}^\bullet t| = 1 = |t^\bullet|$ (N' is an S-net [5]),*
—*Under the flow relation, N' is connected.*

*A set $\mathcal{C}$ of components of net N is called **S-cover** for N, if every place of the net belongs to some component of $\mathcal{C}$. A net is covered by components if it has an S-cover.*

Note that our notion of component does not require strong connectedness and so it is different from notion of S-component in [5], and therefore our notion of S-cover also differs from theirs.
The next definition appears in several places for unlabelled nets, starting with [7].

DEFINITION 9. *A labelled net $N = (S, T, F, \lambda)$ is called **S-decomposable** if, there exists an S-cover $\mathcal{C}$ for N, such that for each $T_i = \{\lambda^{-1}(a) \mid a \in \Sigma_i\}$, there exists S_i such that the induced component (S_i, T_i, F_i) is in $\mathcal{C}$.*

There might be many S-covers, of varying sizes, for a net but when we say net is S-decomposable it means that net has an S-cover in which there are k components in it, and Σ_i is the alphabet of i-th component.

PROPOSITION 4. *If labelled net N is S-decomposable then for all a-labelled transitions $t \in T$, $|loc(a)| \geq max(|{}^\bullet t|, |t^\bullet|)$.*

PROOF. Consider a transition t labelled a with $|{}^\bullet t| > 1$. Let $\{p, q\} \subseteq {}^\bullet t$. Since N is S-decomposable, we have an S-cover for N. So there exist components $N_i = (S_i, T_i, F_i)$ and $N_j = (S_j, T_j, F_j)$ such that $p \in S_i$ and $q \in S_j$. If locations $i = j$ then p and q will belong to same component and by definition of S-cover transition t will also belong to it, which cannot be the case as components are S-nets by definition, so a transition can not have multiple pre-places in it. Therefore i and j are distinct and transition $t \in T_i \cap T_j$. Hence $|loc(a)| \geq |{}^\bullet t|$, the other case is dealt with in a similar manner. □

Now from S-decomposability we get S-cover for net N since, there exist subsets $S_1, S_2, \ldots, S_K$ of places S, such that $S = S_1 \cup S_2 \cup \ldots S_K$ and ${}^\bullet S_i \cup S_i^\bullet = T_i$, such that, subnet (S_i, T_i, F_i) generated by S_i and T_i is an S-net, where F_i is an induced flow relation from S_i and T_i.

3.1 Properties of Nets

DEFINITION 10. *Let x be a node of a net N. The cluster of x, denoted by $[x]$, is the minimal set of nodes containing x such that*

—*if a place $s \in [x]$ then $s^\bullet$ is included in $[x]$, and*
—*if a transition $t \in [x]$ then ${}^\bullet t$ is included in $[x]$.*

A cluster C is denoted by tuple (S_C, T_C), where S_C is the set of places and T_C is the set of transitions of C. A cluster C is called free choice (FC) if all transitions in C have the same pre-set. A net is called free choice if all its clusters are free choice.

The set $\{[x] \mid x$ is a node of $N\}$ is a partition of the nodes of N.
The next definition will turn out to be the analogue to the separation of labels property of product systems. It is checkable in linear time.

DEFINITION 11. *A labelled net $N = (S, T, F, \lambda)$ is said to have the **unique cluster property** (briefly, **ucp**) if for all globals $a \in \Sigma$, there exists at most one cluster in which all transitions labelled a occur. N is **deterministic for synchronization** if for every global a, every cluster contains at most one a-labelled transition.*

In a labelled N, for a cluster $C = (S_C, T_C)$ define the a-labelled transitions $C_a = \{t \in T_C \mid \lambda(t) = a\}$. If the net has an S-decomposition generated by S_i, we associate a post-product $\pi(t) = \Pi_{i \in loc(a)}(t^\bullet \cap S_i)$ with every such transition t. This is well defined since by the S-net condition every transition will have

at most one post-place in S_i. Let $post(C_a) = \bigcup_{t \in C_a} \pi(t)$. We also define the post-projection of the cluster $C_a[i] = C_a{}^{\bullet} \cap S_i$ and the **post-decomposition** $postdecomp(C_a) = \Pi_{i \in loc(a)} C_a[i]$.
Clearly $post(C_a) \subseteq postdecomp(C_a)$. The following definition appears to be new and is key to direct product representability. It says that every post-decomposition is represented in the cluster. Checking this condition is in PTIME because one can implement it by counting.

DEFINITION 12. *An S-decomposable net $N = (S, T, F, \lambda)$ is said to be* ***distributed choice*** *if, for all a in Σ and for all clusters C of N, $postdecomp(C_a) \subseteq post(C_a)$.*

We will use "distributed free choice" for nets which are distributed choice as well as free choice. Note that an unlabelled S-decomposable net can be thought of as being labelled by its set of transitions T, in which case the definition is satisfied. Our example in the introduction is not distributed choice.

3.2 Net Systems and their Languages

For our results we are only interested in 1-bounded (or condition/event) nets, where a place is either marked or not marked. Hence we define a marking as a function from the states of a net to $\{0, 1\}$.
A transition t is enabled in a marking M if all places in its preset are marked by M. In such a case, t can be fired to yield the new marking $M' = (M \setminus {}^{\bullet}t) \cup t^{\bullet}$. We write this as $M[t\rangle M'$ or $M[\lambda(t)\rangle M'$.
A firing sequence (finite or infinite) $\lambda(t_1)\lambda(t_2)\ldots$ is defined by composition, from $M_0[t_1\rangle M_1[t_2\rangle \ldots$ For every $i \leq j$, we say that M_j is reachable from M_i. A net system (N, M_0) is **live** if, for every reachable marking M and every transition t, there exists a marking M' reachable from M which enables t.

DEFINITION 13. *For a labelled net system $(N, M_0, \mathcal{G})$, its language is defined as $Lang(N, M_0, \mathcal{G}) = \{\lambda(\sigma) \in \Sigma^* \mid \sigma \in T^*$ and $M_0[\sigma\rangle M$, for some $M \in \mathcal{G}\}$.*

If a net (S, T, F, λ) is 1-bounded and S-decomposable then a marking can be written as a k-tuple from $S_1 \times S_2 \times \ldots \times S_k$. It is known [15, 12] that if we do not enforce the "direct product" condition below we get a larger subclass of languages.

DEFINITION 14. *An* ***S-decomposable labelled net system*** *$(N, M_0, \mathcal{G})$ is an S-decomposable labelled net $N = (S, T, F, \lambda)$ along with an initial marking M_0 and a set of final markings $\mathcal{G} \subseteq \wp(S)$. $\mathcal{G}$ is a set of* ***direct product*** *markings if when $\langle q_1, q_2, \ldots q_k\rangle \in \mathcal{G}$ and $\langle q'_1, q'_2, \ldots q'_k\rangle \in \mathcal{G}$ then $\{q_1, q'_1\} \times \{q_2, q'_2\} \times \ldots \times \{q_k, q'_k\} \subseteq \mathcal{G}$.*

3.3 On the Definition of Distributed Choice

As we will see, the definition of distributed choice nets is required in the proofs which go back and forth between nets and product systems. This is independent of the definition of free choice. Thus we see distributed choice as a condition which has remained hidden since all work on nets typically does not consider labellings.
One might ask whether the complicated condition of Definition 12 is really necessary for product decomposition. We first considered just the cardinality between the two sets, $post(C_a)$ and C_a i.e., $|post(C_a)| = |C_a|$. But our initial example net shown in Figure 1, satisfies this condition and in Proposition 3 we have shown that it's language is not direct product representable.

Next we considered the cardinality between the two sets, $postdecomp(C_a)$ and C_a i.e., $|postdecomp(C_a)| = |C_a|$. Unfortunately a variant of our initial example which satisfies this condition is not direct product representable. The net is S-decomposable, free choice and satisfies the unique cluster property. For $\{p_1, p_2\}$ the only final marking, the labelled net shown in the Figure 2, accepts the Mazurkiewicz trace language $L = \{abd, adb, ace, aec\}^*$ over the distribution $\Sigma = (\Sigma_1 = \{a, b, c\}, \Sigma_2 = \{a, d, e\})$.

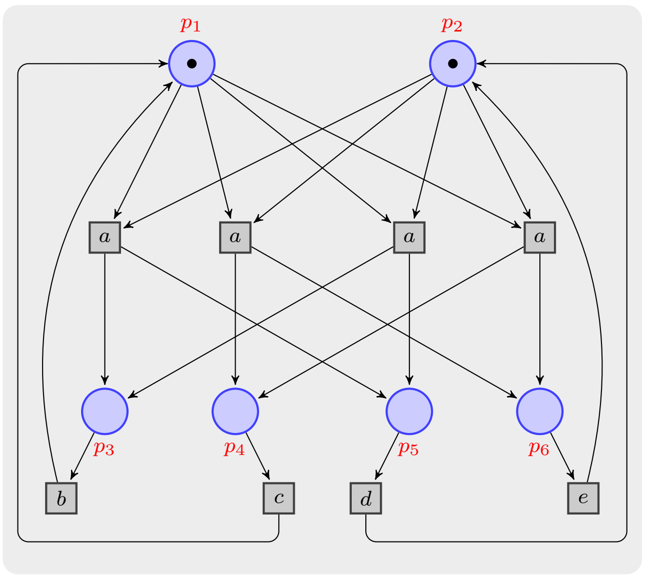

Fig. 2. Labelled free choice net, which is not direct product representable

PROPOSITION 5. *There is no direct product automaton over Σ which represents this language.*

PROOF. Let $w = abeacd$. Then $w_1 = w{\downarrow_{\Sigma_1}} = abac$ and $w_2 = w{\downarrow_{\Sigma_2}} = aead$. Since both $w_1, w_2 \in L$ But we have $u_1 = abdaec \in L$ with $u_1{\downarrow_{\Sigma_1}} = abac$ and $u_2 = aecadb \in L$ with $u_2{\downarrow_{\Sigma_2}} = aead$. So using the characterization given in Proposition 2, we get word $abeacd \in L$ which is a contradiction. □

4. NETS TO PRODUCT SYSTEMS

Even if a net is 1-bounded and S-decomposable each component need not have only one token in it, but when we say that a 1-bounded net is S-decomposable we assume that each component has one token. For live and 1-bounded free choice nets, such S-covers can be guaranteed [5].
Now we describe a simple generic construction of a product system from a net which is S-decomposable and distributed choice. By assuming more properties, we get more properties of the constructed product system. In the next section, we do another simple generic construction of a net from a product system, the properties of S-decomposability and distributed choice are obtained automatically, and again we can get more properties if desired.
Let $(N, M_0, \mathcal{G})$ be a 1-bounded and S-decomposable labelled net system, where $N = (S, T, F, \lambda)$ is the underlying net. Let $N_i = (S_i, T_i, F_i)$ denote components in the S-cover, for all i in $\{1, 2, \ldots, k\}$. We define $P_i = S_i, G = \{(M \cap P_1, \ldots, M \cap P_k) \mid M \in \mathcal{G}\}$. If $\mathcal{G}$ was a direct product set of final markings, we can

define $G_i = \{M \cap P_i \mid M \in \mathcal{G}\}$ and set G to be their product $G_1 \times \cdots \times G_k$. Let p_i^0 be the unique state in $M_0 \cap P_i$. For each $t \in T_i$, we know that, there exist places $p, p' \in S_i$ such that (p, t) and (t, p') belong to F_i. Formally we define set of local moves, $\rightarrow_i = \{\langle p, \lambda(t), p'\rangle \mid t \in T_i$ and $(p, t), (t, p') \in F_i$, for $p, p' \in P_i\}$. So we get sequential system $A_i = \langle P_i, \rightarrow_i, p_i^0\rangle$ corresponding to the component (S_i, T_i, F_i). Hence we get the product system $A = \langle A_1, A_2, \ldots, A_k\rangle$ over distributed alphabet Σ. The size of A is linear in the size of the net. If N was deterministic for synchronization then the constructed system A is deterministic for global actions.

4.1 Distributed Choice to Product Systems

THEOREM 1. *Let $(N, M_0, \mathcal{G})$ be a 1-bounded and S-decomposable labelled distributed choice net system, and A is the product system constructed as above. Then*

(1) *If net is free choice then A is a FC-matching product and*
(2) *in addition if the net system is live, then*
   (a) *all runs of A are consistent with the matching.*
   (b) *$Lang(N, M_0, \mathcal{G}) = Lang(A)$.*

PROOF. Let $N = (S, T, F, \lambda)$ be the underlying net. Let M be a reachable marking. Since A_i is a component, number of tokens in P_i remain constant, and we know that P_i had one token at M_0, so we always have one unique $r_i \in P_i$ such that $r_i = M \cap P_i$, for all i in *Loc*. So we get $R(M) = \langle r_1, r_2, \ldots, r_k\rangle$, which is a product state. In the reverse direction, for any product state R, taking union of all places in R gives us a unique set of places $M(R)$ which serves as a marking of net.

(1) Since net is free choice, we get a conflict-equivalent matching of labels by, taking tuples of pre-places of a cluster, making A an FC-matching product.
(2) Assume that the net system is live.
   (a) Suppose we have two places, say p_1 in location 1 and p_2 in location 2 which are not matched on any action a, which means that they are not in the same cluster of the free choice net N. Again by construction, let the local move $p_1 \xrightarrow{a} p'_1$ be obtained from the net transition t_1 with pre-place p_1 coming from cluster $C_1 = (S_1, T_1)$. By S-decomposability, C_1 has a matching pre-place q_2 in location 2. Similarly let $p_2 \xrightarrow{a} p'_2$ be obtained from t_2 with pre-place p_2 coming from cluster $C_2 = (S_2, T_2)$, with matching pre-place r_1 in location 1 using S-decomposability.
   Since we started with a 1-bounded S-cover, initially each component had only one token in it, and in a component number of tokens remains constant at any reachable marking. So the places q_2 and r_1 are distinct from p_1 and q_2 respectively. Again using the 1-bounded S-cover, q_2 and r_1 have to be unmarked at the net marking $M(R)$. By liveness, some transition of C_1 will get fired and by free choice, for this we need to have tokens in all places of S_1. To bring the token in to q_2 we have to fire some transition in C_2, for which by free choice we need to put a token in place r_1, which has to come from p_1. In short we have two dead places from where a transition can never be fired, contradicting liveness.
   This means that all the places $p_1, \ldots, p_l$ are in the same cluster. As a consequence the run of the product system at this marking (and inductively at all reachable markings) will be consistent with the matching of labels defined in the construction.
   (b) We will show language equivalence by showing the stronger property that the maps $R(.)$ and $M(.)$ constitute an isomorphism of reachable markings of N with reachable product states of A. Clearly this is the case for the initial marking and the initial product state.
   To prove $Lang(N, M_0, \mathcal{G}) \subseteq Lang(A)$, we show that for a transition $t \in T$ of the net system, labelled a, and $M[t\rangle M'$ in the net, then we have a global move g in the product system with label a yielding $R(M) \overset{a}{\Rightarrow} R(M')$ in the product system. We know that ${}^\bullet t \subseteq M$ and $t^\bullet \subseteq M'$. Hence for each $i \in loc(t)$ we have a place $r_i \in P_i$ such that $(r_i, t) \in F_i$. And since N_i corresponding to A_i is a component there exists another place $r'_i \in P_i$ such that $(t, r'_i) \in F_i$. But by construction we get a transition $\langle r_i, a, r'_i\rangle \in \rightarrow_i$, $\forall i \in loc(t)$. So by definition of a product system we get a global move $g = \Pi_{i \in loc(t)} \langle r_i, a, r'_i\rangle$ in constructed product system A. So we get ${}^\bullet t = \textsf{pre-places}(g)$ and $t^\bullet = \textsf{post-places}(g)$. So each $r_i \in \textsf{pre-places}(g)$ also belong to tuple $R(M)$ at i-th place. Repeating the same argument for M' we get that $r'_i \in \textsf{post-places}(g)$ also belong to tuple $R(M')$. So in the product system we have $R(M) \overset{a}{\Rightarrow} R(M')$.
   In the reverse direction, to prove $Lang(A) \subseteq Lang(N, M_0, \mathcal{G})$, we show that, if $R \overset{a}{\Rightarrow} R'$ in the product system using a global move g then we have $M(R)[a\rangle M(R')$ in the net system, when R is a reachable product state. This is the direction which uses consistency of matching which we get by using liveness of net.
   Inductively we know from the isomorphism that $M(R)$ is a reachable marking and this extends the isomorphism to $M(R')$. Let $loc(a) = \{1, \ldots, l\}$ and $g = \langle\langle p_1, a, p'_1\rangle, \ldots \langle p_l, a, p'_l\rangle\rangle$.
   We proved above that A is consistent with constructed matching. Therefore, being a reachable state of A, R is in a-matching. Hence $\textsf{pre-places}(g)$ belong to a-matching. So by construction, these places belong to same cluster of net. By S-decomposability it also means that $\textsf{post-places}(g)$ are post-places of the same cluster. Since cluster is free choice, all transitions have same pre-places. By distributed choice, the post-decomposition $\langle p'_1, \ldots, p'_l\rangle$ is represented by one of the a-labelled transitions in the cluster. Firing this transition we have $M(R)\ [a\rangle\ M(R')$ in the net system and the isomorphism is inductively extended. Since the final markings of the net get related to the final product states, we get language equivalence of net and product system.

□

If the net satisfies the unique cluster property, we get separation of labels and we do not need to use liveness in the proof.

COROLLARY 1. *Let $(N, M_0, \mathcal{G})$ be a 1-bounded, S-decomposable labelled distributed free choice net having the unique cluster property, and A the product system constructed at the beginning of this section. Then A is an FC-product with separation of labels, and $Lang(N, M_0, \mathcal{G}) = Lang(A)$.*

PROOF. Let $N = (S, T, F, \lambda)$ be the underlying net. Let M be a reachable marking. Since A_i is a component, number of tokens in P_i remain constant, and we know that P_i had one token at M_0, so we always have one unique $r_i \in P_i$ such that $r_i = M \cap P_i$, for all i in *Loc*. So we get $R(M) = \langle r_1, r_2, \ldots, r_k\rangle$, which is a product state. In the reverse direction, for any product state R, taking union

of all places in R gives us a unique set of places $M(R)$ which serves as a marking of net.
Since net is free choice, and using S-decomposability we get that A is FC-product. As N have unique cluster property and S-decomposable, constructed system A have separation of labels.
As in the proof of Theorem 1 we will show language equivalence by showing the stronger property that the maps $R(.)$ and $M(.)$ constitute an isomorphism of reachable markings of N with reachable product states of A. Clearly this is the case for the initial marking and the initial product state.
To prove $Lang(N, M_0, \mathcal{G}) \subseteq Lang(A)$, we show that for a transition $t \in T$ of the net system, labelled a, and $M[t\rangle M'$ in the net, then we have a global move g in the product system with label a yielding $R(M)\overset{a}{\Rightarrow}R(M')$ in the product system. We know that ${}^\bullet t \subseteq M$ and $t^\bullet \subseteq M'$. Hence for each $i \in loc(t)$ we have a place $r_i \in P_i$ such that $(r_i, t) \in F_i$. And since N_i corresponding to A_i is a component there exists another place $r'_i \in P_i$ such that $(t, r'_i) \in F_i$. But by construction we get a transition $\langle r_i, a, r'_i\rangle \in\rightarrow_i$, $\forall i \in loc(t)$. So by definition of a product system we get a global move $g = \Pi_{i\in loc(t)}\langle r_i, a, r'_i\rangle$ in constructed product system A. So we get ${}^\bullet t =$ pre-places(g) and $t^\bullet =$ post-places(g). So each $r_i \in$ pre-places(g) also belong to tuple $R(M)$ at i-th place. Repeating the same argument for M' we get that $r'_i \in$ post-places(g) also belong to tuple $R(M')$. So in the product system we have $R(M)\overset{a}{\Rightarrow}R(M')$.
In the reverse direction, to prove $Lang(A) \subseteq Lang(N, M_0, \mathcal{G})$, we show that, if $R\overset{a}{\Rightarrow}R'$ in the product system using a global move g then we have $M(R)[a\rangle M(R')$ in the net system, when R is a reachable product state.
Inductively we know from the isomorphism that $M(R)$ is a reachable marking and this extends the isomorphism to $M(R')$. Let $loc(a) = \{1, \ldots, l\}$ and $g = \langle\langle p_1, a, p'_1\rangle, \ldots \langle p_l, a, p'_l\rangle\rangle$.
We proved above that A have separation of labels. By S-decomposablity of net and by construction, these places belong to same cluster of net. By S-decomposability it also means that post-places(g) are post-places of the same cluster. Since cluster is free choice, all transitions have same pre-places. By distributed choice, the post-decomposition $\langle p'_1, \ldots, p'_l\rangle$ is represented by one of the a-labelled transitions in the cluster. Firing this transition we have $M(R)\ [a\rangle\ M(R')$ in the net system and the isomorphism is inductively extended. Since the final markings of the net get related to the final product states, we get language equivalence of net and product system.
In the proof of Theorem 1, we used liveness to prove consistency of matching, which in turn was used to prove that pre-places of g belonged to the same cluster. But here we use unique cluster property to prove that. □

5. PRODUCT SYSTEMS TO NETS

Given a product system $A = \langle A_1, A_2, \ldots, A_k\rangle$ over distribution Σ, we can generically produce a net system $(N = (S, T, F, \lambda), M_0, \mathcal{G})$ as follows:

—$S = \cup_i P_i$, the set of places.

—$T = \cup_a T_a$, where T_a is $\Rightarrow_a$, the set of a-labelled global moves.

—The labelling function λ labels by a the transitions in T_a.

—The flow relation $F = \{(p, g), (g, q) \mid g \in T_a, g[i] = \langle p, a, q\rangle, i \in loc(a)\}$.

—$M_0 = \{p^0_1, \ldots, p^0_k\}$, the initial product state.

—$\mathcal{G} = G$, the set of final product states.

Since a global action a can be in every component A_i of the product system and there can be an arbitrary number n_i of a-labelled choices in each component, the resulting a-cluster in the net has $n_1 \times \cdots \times n_k$ transitions which can be exponential in the size of the product system. If the product system was deterministic for global actions, then the constructed net is polynomial in size. If the final product states were a direct product $G_1 \times \cdots \times G_k$, the final markings will also be direct product.
When we construct nets from product systems with a conflict-equivalent matching of labels with respect to which all runs are consistent, we can refine the construction above to choose $T' \subseteq T$ and get a distributed free choice net. As an illustration of above construction, we give below an example.

EXAMPLE 1. *Consider the product system $A = (A_1, A_2)$ over distributed alphabet $\Sigma = (\Sigma_1 = \{a, b, d, c\}, \Sigma_2 = \{a, b, e, c\})$ given in Figure 3. Set of final states of A is $G = \{5, 6\} \times \{11, 12\}$. It's language is $Lang(A)$ is given by expression $((a + b)(de + ed)(b + c))$. Set of matchings are: a-matching$=\{(1, 7)\}$, b-matching$=\{(1, 7), (4, 10)\}$, c-matching$=\{(4, 10)\}$.*

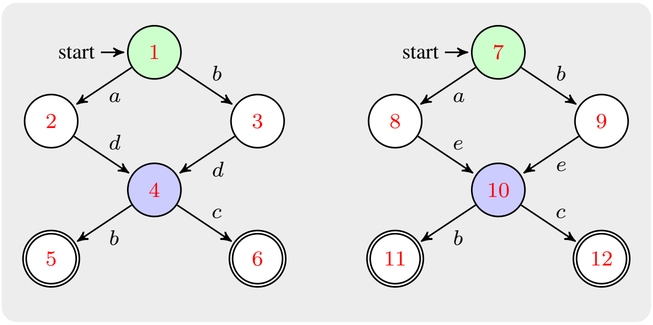

Fig. 3. FC-matching product system

From the product system given above, we get, a language equivalent, free choice net , shown in Figure 4, after pruning out transition corresponding to global move $(4, 7) \xrightarrow{b} (5, 9)$. It is easy to see that this net satisfies distributed choice property.

THEOREM 2. *Let $(N, M_0, \mathcal{G})$ be the net system constructed from product system A above. Then*

(1) N is a S-decomposable net.

(2) N satisfies distributed choice property.

(3) $Lang(N, M_0, \mathcal{G}) = Lang(A)$.

(4) Further, if A is FC-matching product and all runs of A are consistent with the given matching of labels, then we can choose $T' \subseteq T$ such that the subnet N' generated by T' is a free choice net and $(N', M_0, \mathcal{G})$ accepts the same language.

(5) Further, if the product system was deterministic for global actions, the net is deterministic for synchronization.

PROOF. (1) That N is S-decomposable follows from the fact that we can build components $N_i = (S_i, T_i, F_i)$ from product system A. Take $S_i = P_i$ and $T_i = \{\lambda^{-1}(a) \mid a \in \Sigma_i\}$ by definition. The flow relation F can be written as the union of, $F_i = \{(p, g), (g, q) \mid g[i] = \langle p, a, q\rangle\}$.

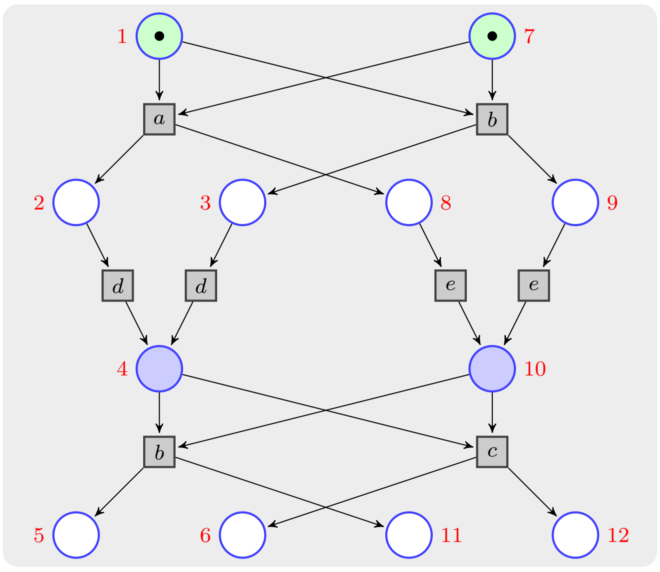

Fig. 4. Free Choice net $(N, M_0 = \{(1,7)\}, \mathcal{G} = \{(5,6) \times (11,12)\})$

(2) Now we want to prove that N satisfies distributed choice. Consider an a-labelled transition t in a cluster C of N'. Because of S-decomposability it has exactly one pre-place and one post-place from each location of a. Given a pre-place p and post-place q in A_i there can be only one local move $\langle p, a, q\rangle \in \xrightarrow{a}_i$. So for a fixed pre-place p, any local move on action a is uniquely identified by its post-place. Therefore the post-places of t uniquely identify a global move in A or transition of net.

(3) Now we prove that $Lang(N, M_0, \mathcal{G}) = Lang(A)$. In the construction, $M_0 = \{p_1^0, \ldots, p_k^0\}$ where p_i^0 is the initial place of the sequential system A_i in the product. Inductively, for any product state R of the product system we can associate a unique marking $M(R)$. The set of transitions T of the constructed net is the set of global moves of the product system, and since the places of the net are the set obtained by taking union of places of all sequential systems of the product, we have that ${}^\bullet g$ = pre-places(g) and $g^\bullet$ = post-places(g). So if there is a product state R' obtained by taking global move g having $\lambda(g) = a$ at product state R, then we get $M(R)[a\rangle M(R')$ in the net system produced.

On the other hand, from initial marking M_0 of net system, we can construct the initial product state R^0 of the product system, by taking its intersection with the places of a sequential system. So, $\{p_i^0\} = M_0 \cap P_i$, $\forall i \in Loc$. Since the result of intersection is a singleton set, we write this with abuse of notation as $p_i^0 = M_0 \cap P_i$. Inductively, we can associate a reachable product state with any given reachable marking of the net, as in the component corresponding to location i, there can be exactly one place which is marked.

Consider a transition t in the net system. if $p \in P_i \cap {}^\bullet t$ then after firing this transition t, token from place p is circulated back in the P_i for some place $q \in P_i \cap t^\bullet$. So if we have $M[a\rangle M'$ in the net system then, we get $R(M) \overset{a}{\Rightarrow} R(M')$ in the product system, where $R(M)$ and $R(M')$ are the product states corresponding to the markings M and M' respectively, of net system.

This establishes an isomorphism between the set of reachable product states of the product system and the set of reachable markings of net. Because the initial marking and the final markings correspond to the initial product state and the final product states we get language equivalence of net and product system.

(4) Now we assume that all runs of the product system are consistent with a conflict-equivalent matching of labels. Our choice of the subset of transitions $T' \subseteq T$ is to keep those whose pre-places are part of reachable product states. This does not violate S-decomposability or language equivalence.

We want to prove that N' is a free choice net. Let $C = (S_C, T_C)$ be a cluster of constructed net N'. We have to prove that it is a free choice cluster. If $|S_C| = 1$ or $|T_C| = 1$ then C is trivially a FC-cluster. So we consider the case where $|S_C| > 1$ and $|T_C| > 1$. Let $p, q \in S_C$ and $t_1, t_2 \in T_C$, such that $p \in {}^\bullet t_1 \cap {}^\bullet t_2$ and $q \in {}^\bullet t_1$. We have to prove that $q \in {}^\bullet t_2$.

Consider the case where, $\lambda(t_1) = \lambda(t_2) = a$. We know that $t_1 \in p^\bullet \cap q^\bullet$ and we have above proved that N' is S-decomposable net system, so places p and q do not belong to same component of net. Without loss of generality, assume that p is in component N_i and q is in component N_j with $i \neq j$. So transition $t_1 \in T_i \cap T_j$ implying $|loc(a)| > 1$. Since p, q were part of a reachable product state and runs of A are consistent with matching of labels, pre-place p must have the matching pre-place q. Since t_2 is in $p^\bullet$, because of consistency with matching of labels, all global moves on action a with p must use q. Thus $q \in {}^\bullet t_2$ as required.

Now consider the case where, $\lambda(t_1) = a$ and $\lambda(t_2) = b$. Since $|{}^\bullet t_1| > 1$, by Proposition 4 we have $|loc(a)| > 1$. Net N' is S-decomposable, so places p and q do not belong to same component of net. Without loss of generality, assume that p is in component N_i and q is in component N_j with $i \neq j$. By construction there exist local moves $\langle p, a, p'\rangle, \langle p, b, p''\rangle \in \rightarrow_i$ and $\langle q, a, q'\rangle \in \rightarrow_j$. As the matching is conflict-equivalent, local moves $\langle p, a, p'\rangle$ and $\langle q, a, q'\rangle$ are conflict-equivalent, implying existence of a local move $\langle q, b, q'\rangle \in \rightarrow_j$. By Proposition 1 we get $loc(a) = loc(b)$.

Since runs of product system A are consistent with the matching of labels, when p has outgoing moves on action a and b, they will match with outgoing moves from q in A_j. Since the transitions of N' come from the global moves of the product system, transition t_2, which is labelled b, has same set of pre-places as t_1 labelled a, implying $q \in {}^\bullet t_2$ as required.

(5) Since set of global moves of product system is used to construct set of transitions of the net, and reachable state spaces of both net and product system are isomorphic as proved above, then the determinism of global actions in the product system gives rise to a net which is deterministic for synchronization.

□

COROLLARY 2. *Let $(N = (S, T, F, \lambda), M_0, \mathcal{G})$ be the net system constructed from product system A, as at the beginning of this section. If A is an FC-product with the separation of labels property then N is a distributed free choice net with the unique cluster property.*

PROOF. Using separation of labels property we get the unique cluster property straightaway, and using conflict-equivalence we get that the net is distributed free choice. No transitions have to be pruned. □

6. PERSPECTIVE

In earlier work [10], it has been shown that a graph-theoretic condition called "structural cyclicity" enables us to extract syntax from a conflict-equivalent product system. In the present work we explore the connection between free choice nets and product systems which have conflict-equivalence and other properties. In particular we have a broader class of product systems, where the conflict-equivalence is not statically fixed.
More recently [14] it has been shown that a "pairing" condition in the syntax can be connected to the matching condition on product systems used here. Our expressions come in two syntactic sorts:

Regular expressions over Σ_i	$s ::= a \in \Sigma_i \mid s_1 s_2 \mid s_1 + s_2 \mid s^*$
Connected expressions	$e ::= 0 \mid fsync(s_1, s_2, \ldots, s_k)$

The s_i can be any regular expressions (of any star-height), which is different from expressions of paper [10]. Results in [14] show that for connected expressions, with suitable restrictions of a "pairing" condition, correspond to product systems with separation of labels, with matching of labels and with a matching where all runs are consistent with the matching.
Putting these theorems together with the results of this paper, we get expressions for different subclasses of labelled free choice nets. Though the present paper shares many definitions with earlier paper [14], the results in the two papers are complementary. In particular [14] did not have the definition of distributed choice. In terms of nets, this meant that we were restricted to considering analogues of labelled free choice nets with deterministic synchronization. That restriction no longer applies. Another small extension in this paper is to consider arbitrary as well as "direct product" final markings. It is known [11, 12] that arbitrary sets of final markings lead to closure of direct product languages under boolean operations, and the syntax has to be extended to allow outermost sums to match this [9].
We end with a question on which we have unfinished work. When we construct nets from product systems without using Definition 4 for matching of labels, then we might lose the free choice property in the constructed nets. We give below an example to show what happens.

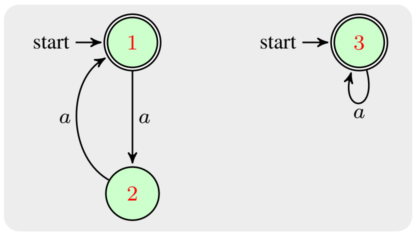

Fig. 5. Direct product system with language $L = \{aa\}^*$

For the product system given in Figure 5 we get a net shown in the Figure 6 with initial marking $M_0 = \{(1, 3)\}$, and set of final markings $\mathcal{G} = \{(1, 3)\}$. This net is not free choice although it is language equivalent to the product system from which it was constructed.
But there may be cleverer constructions: a free choice net for this example product system is obtained by unfolding the second sequential system to obtain a matching of labels.

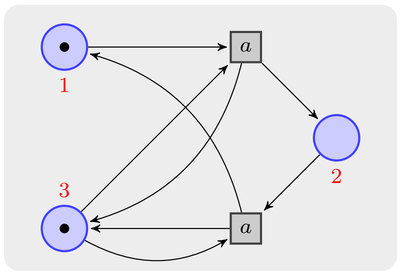

Fig. 6. Non free choice net

7. REFERENCES

[1] André Arnold (1994): *Finite transition systems*. Prentice Hall.

[2] Sandie Balaguer, Thomas Chatain & Stefan Haar (2012): *A concurrency-preserving translation from time Petri nets to networks of timed automata*. *Formal Meth. Sys. Des.* 40(3), pp. 330–355. Available at `http://dx.doi.org/10.1007/s10703-012-0146-4`.

[3] Roy H. Campbell & A. Nico Habermann (1974): *The specification of process synchronization by path expressions*. In: *Proc. Operating Systems conference, Rocquencourt*, *LNCS* 16, Springer, pp. 89–102.

[4] Ilaria Castellani, Madhavan Mukund & P.S. Thiagarajan (1999): *Synthesizing distributed transition systems from global specifications*. In C. Pandu Rangan, Venkatesh Raman & R. Ramanujam, editors: *Proc. 19th FSTTCS, Chennai*, *LNCS* 1738, Springer, pp. 219–231. Available at `http://dx.doi.org/10.1007/3-540-46691-6_17`.

[5] Jörg Desel & Javier Esparza (1995): *Free choice Petri nets*. Cambridge University Press, New York, USA.

[6] Volkert Diekert & Grzegorz Rozenberg, editors (1995): *The book of traces*. World Scientific, River Edge, NJ, USA.

[7] Michel Henri Théodore Hack (1972): *Analysis of production schemata by Petri nets*. Project Mac Report TR-94, MIT.

[8] Charles Antony Richard Hoare (1985): *Communicating sequential processes*. Prentice-Hall.

[9] Kamal Lodaya (2006): *Product automata and process algebra*. In: *Proc. 4th SEFM, Pune*, IEEE, pp. 128–136.

[10] Kamal Lodaya, Madhavan Mukund & Ramchandra Phawade (2011): *Kleene theorems for product systems*. In Markus Holzer, Martin Kutrib & Giovanni Pighizzini, editors: *Proc. 13th DCFS, Limburg*, *LNCS* 6808, pp. 235–247.

[11] Swarup Mohalik & R. Ramanujam (2002): *Distributed automata in an assumption-commitment framework*. *Sādhanā* 27, part 2, pp. 209–250.

[12] Madhavan Mukund (2011): *Automata on distributed alphabets*. In Deepak D'Souza & Priti Shankar, editors: *Modern applications of automata theory*, World Scientific, pp. 257–288.

[13] Madhavan Mukund & Milind A. Sohoni (1997): *Keeping track of the latest gossip in a distributed system*. *Distrib. Comp.* 10(3), pp. 117–127.

[14] Ramchandra Phawade & Kamal Lodaya (2014): *Kleene theorems for labelled free choice nets*. In: *Proc. 8th PNSE, Tunis*, CEUR-WS, pp. 75–89.

[15] Wiesław Zielonka (1987): *Notes on finite asynchronous automata*. *Inform. Theor. Appl.* 21(2), pp. 99–135.