Theoretical Computer Science 429 (2012) 21–29

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Vector team automata

Maurice H. ter Beek [a,*], Jetty Kleijn [b]

[a] ISTI–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

[b] LIACS, Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands

A B S T R A C T

Vector team automata are team automata with an explicit representation of synchronizations. This makes a translation possible of a subclass of vector team automata into individual token net controllers, a model of labeled Petri nets developed within the framework of vector controlled concurrent systems.

1. Introduction

Team automata consist of sequential (component) automata which interact through synchronizations on common actions. The effect of a synchronously executed global team action on the team's state is described in terms of the local state changes of the automata that take part in the synchronization. The automata not involved remain idle and their current states are unaffected. Team automata have been formally defined in [3]. Based on I/O automata, team automata usually distinguish input and output (communication) actions and (private) internal actions. Contrary to I/O automata, however, team automata impose hardly any restrictions on the role of actions in components and their composition is not based on the synchronous product or on any other a priori fixed way of synchronizing actions. Thus a variety of interaction protocols can be defined in terms of team automata [2,3]. Here we do not distinguish between possible roles of actions, as these are not relevant for our focus on making explicit the concurrency inherent to team automata through a Petri net semantics.

Petri nets [11] consist of an underlying structure (a net) and a firing rule describing the dynamics. A net is a bipartite directed graph distinguishing places (representing local states) and events (also called transitions, representing actions). The firing rule describes when (in which states) an event can occur and its effect on the current state if it occurs. It is fundamental to Petri net theory that both the conditions allowing an event to occur and the effect of its occurrence on the global state, are local in the sense that they only involve places in the immediate neighborhood of (adjacent to) the event.

A team automaton resembles a (labeled) Petri net with the local states of its components as places and synchronously executed actions as net transitions. In a team automaton, actions have a local effect, restricted to those automata actually involved in executing that action. There are, however, two important differences. Synchronizations in a team automaton depend on the global state rather than only the local states of the automata involved in the execution—so-called *state sharing* [6]. Team automata moreover lack explicit information on automata executing loops, making it sometimes impossible to determine which automata take part in a synchronization.

In this paper, we first turn to the latter issue and switch from synchronized team actions to vectors of component actions, from which the participation of automata in synchronizations can be seen immediately. This allows us to translate a subclass of vector team automata to a specific type of state machine decomposable nets with a synchronization mechanism based on vector labels, namely the *Individual Token Net Controllers* (ITNCs) introduced in [8,9]. As a result, their concurrent semantics becomes applicable to vector team automata.

* Corresponding author. Tel.: +39 050 3153471; fax: +39 050 3152040.
E-mail addresses: maurice.terbeek@isti.cnr.it (M.H. ter Beek), kleijn@liacs.nl (J. Kleijn).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.12.020

We thus relate two formalisms initiated by Grzegorz Rozenberg and his collaborators, which culminated in two Ph.D. theses under his supervision [1,7].

The formalization we present below of the connection between vector team automata and ITNCs, previously sketched in [3,10], facilitates the transfer of notions, techniques, and results. In fact, we will present two new results:

1. The behavior of non-state-sharing vector team automata equals that of the subclass of ITNCs obtained from vector team automata;
2. Subnets of ITNCs obtained from vector team automata equal the ITNCs obtained from subteams of the vector team automata.

2. Vector team automata

A *component automaton* $\mathcal{C}$ is defined as $\mathcal{C} = (Q, \Sigma, \delta, I)$, with finite sets Q of *states* and Σ of *actions* (with $Q \cap \Sigma = \varnothing$), set $\delta \subseteq Q \times \Sigma \times Q$ of *transitions* and set $I \subseteq Q$ of *initial states*. The set of (finite) *computations* of $\mathcal{C}$ is defined as $\mathbf{C}_{\mathcal{C}} = \{ q_0 a_1 q_1 a_2 \cdots a_n q_n \mid n \geq 0 \text{ and } (q_{i-1}, a_i, q_i) \in \delta \text{ for } 1 \leq i \leq n\}$ and its *behavior* as $\mathbf{B}_{\mathcal{C}} = \text{pres}_{\Sigma}(\mathbf{C}_{\mathcal{C}})$; here the homomorphism $\text{pres}_{\Gamma,\Sigma}$ preserving symbols in Σ and erasing all other symbols in Γ is defined by $\text{pres}_{\Gamma,\Sigma}(a) = \lambda$ if $a \in \Gamma \setminus \Sigma$ and $\text{pres}_{\Gamma,\Sigma}(a) = a$ if $a \in \Gamma \cap \Sigma$. Γ is omitted if clear from the context. For $a \in \Sigma$, the a-transitions in δ are $\delta_a = \{ (q, q') \mid (q, a, q') \in \delta\}$ and $(q, q) \in \delta_a$ is called a *loop*.

We fix some notation for the sequel. Let $\mathcal{I} \subseteq \mathbb{N}$ be a nonempty, finite set of indices $\{i_1, i_2, \ldots, i_n\}$ with $i_j < i_k$ if $j < k$. For a collection of sets V_i, with $i \in \mathcal{I}$, $\prod_{i\in\mathcal{I}} V_i$ is the Cartesian product consisting of elements $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ with $v_i \in V_i$ for each $i \in \mathcal{I}$. If $v_i \in V_i$, for each $i \in \mathcal{I}$, then $\prod_{i\in\mathcal{I}} v_i$ denotes element $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ of $\prod_{i\in\mathcal{I}} V_i$. For all $j \in \mathcal{I}$ and $(v_{i_1}, v_{i_2}, \ldots, v_{i_n}) \in \prod_{i\in\mathcal{I}} V_i$, define $\text{proj}_j((v_{i_1}, v_{i_2}, \ldots, v_{i_n})) = v_j$. We thus observe that if $\mathcal{I} = \{4, 5\}$, then $\text{proj}_4((a, b)) = a$. If $\varnothing \neq \mathcal{J} \subseteq \mathcal{I}$, then $\text{proj}_{\mathcal{J}}((v_{i_1}, v_{i_2}, \ldots, v_{i_n})) = \prod_{j\in\mathcal{J}} v_j$. Let $\mathcal{S} = \{\mathcal{C}_i \mid i \in \mathcal{I}\}$ be a fixed nonempty, indexed set of component automata specified as $\mathcal{C}_i = (Q_i, \Sigma_i, \delta_i, I_i)$. Let $\Sigma = \bigcup_{i\in\mathcal{I}} \Sigma_i$ and let $a \in \Sigma$.

We now directly define a vector team automaton $\mathcal{V}$ (over $\mathcal{S}$) by describing its synchronizations as transitions with vector labels chosen, for each $a \in \Sigma$, from the *complete vector transition space* of a in $\mathcal{S}$:

$$\Delta_a^v(\mathcal{S}) = \left\{ (q, \underline{a}, q') \in \prod_{i\in\mathcal{I}} Q_i \times \prod_{i\in\mathcal{I}} \{a, \lambda\} \times \prod_{i\in\mathcal{I}} Q_i \;\middle|\; (\exists i \in \mathcal{I}\colon \text{proj}_i(\underline{a}) \neq \lambda) \right.$$
$$\left. \wedge (\forall i \in \mathcal{I}\colon ([\text{proj}_i(\underline{a}) = a] \Rightarrow [(\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}]) \wedge ([\text{proj}_i(\underline{a}) = \lambda] \Rightarrow [\text{proj}_i(q) = \text{proj}_i(q')])) \right\}.$$

Definition 1. A *vector team automaton* $\mathcal{V}$ *over* $\mathcal{S}$ is defined as a construct $\mathcal{V} = (\prod_{i\in\mathcal{I}} Q_i, \Sigma, \delta^v, \prod_{i\in\mathcal{I}} I_i)$ such that $\delta^v \subseteq \bigcup_{a\in\Sigma} \Delta_a^v(\mathcal{S})$. Here δ^v is the set of *(labeled) vector transitions* of $\mathcal{V}$. The set of *(vector)* $\underline{a}$*-transitions* of $\mathcal{V}$ is defined as $\delta_{\underline{a}}^v = \{(q, q') \mid (q, \underline{a}, q') \in \delta^v\}$.

By focusing on a subset of $\mathcal{S}$, we can define a *subteam* of $\mathcal{V}$. Its vector actions/ transitions are restrictions of those of $\mathcal{V}$ to the subteam's component automata.

Definition 2. Let $\mathcal{V} = (Q, \Sigma, \delta^v, I)$ be a vector team automaton over $\mathcal{S}$ and let $\mathcal{K} \subseteq \mathcal{I}$. Then the *subteam* $SUB_{\mathcal{K}}(\mathcal{V})$ *of* $\mathcal{V}$ *determined by* $\mathcal{K}$ is defined as $SUB_{\mathcal{K}}(\mathcal{V}) = (\prod_{k\in\mathcal{K}} Q_k, \bigcup_{k\in\mathcal{K}} \Sigma_k, \delta_{\mathcal{K}}^v, \prod_{k\in\mathcal{K}} I_k)$ such that

$$\delta_{\mathcal{K}}^v = \{(\text{proj}_{\mathcal{K}}(q), \text{proj}_{\mathcal{K}}(\underline{a}), \text{proj}_{\mathcal{K}}(q')) \in \Delta_a^v(\{\mathcal{C}_k \mid k \in \mathcal{K}\}) \mid (q, \underline{a}, q') \in \delta^v\}.$$

Obviously, $SUB_{\mathcal{K}}(\mathcal{V})$ is itself a vector team automaton over $\{\mathcal{C}_k \mid k \in \mathcal{K}\}$.

Example 3. Fig. 1 depicts vector team automata $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$ over component automata $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$. Replacing transitions $(q, \underline{a}, q')$ in a vector team automaton $\mathcal{V}$ over $\mathcal{S}$ by "flat" transitions (q, a, q'), we obtain a team automaton (over $\mathcal{S}$). Fig. 1 also depicts a team automaton $\mathcal{T}$ over $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, which is the flattened version of $\mathcal{V}_1$ and $\mathcal{V}_2$. Note that explicit information on the execution of loops is lost: while for each vector transition in $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$ it is clear which component automata participate, this cannot be seen from transition $((p, q, r), a, (p, q, r'))$ in $\mathcal{T}$. In this sense, vector team automata have more modeling power than team automata.

3. Individual token net controllers

Vector Controlled Concurrent Systems (VCCSs) [7] model concurrent systems consisting of a finite number of sequential components that cooperate by synchronizing their actions. Vectors are used to control and describe the elementary synchronizations in a system as well as its behavior. Thus vector team automata fit into the VCCS framework. We first fix some notation.

Let $\mathcal{K} \subseteq \mathbb{N}$ be a finite, nonempty set of integers with cardinality $n = \#\mathcal{K}$. Let Γ_k be an alphabet, for all $k \in \mathcal{K}$. Any $v \in \prod_{k\in\mathcal{K}} \Gamma_k^*$ is an (n*-dimensional*) *word vector* (over $\{\Gamma_k \mid k \in \mathcal{K}\}$) and $\Lambda = (\lambda, \ldots, \lambda) \in \prod_{k\in\mathcal{K}} \Gamma_k^*$ is the *empty word vector*. A set of word vectors is a *vector language* (over $\{\Gamma_k \mid k \in \mathcal{K}\}$).

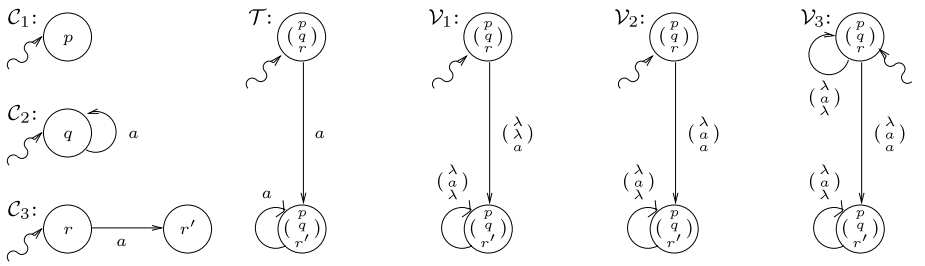

Fig. 1. Component automata $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ and (vector) team automata $\mathcal{T}$, $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$.

A vector $w \in \prod_{k\in\mathcal{K}}(\Gamma_k \cup \{\lambda\}) \setminus \{\Lambda\}$ is an (*n-dimensional*) *vector letter* (over $\{\Gamma_k \mid k \in \mathcal{K}\}$). The set of all vector letters over $\{\Gamma_k \mid k \in \mathcal{K}\}$ is denoted by $\mathrm{tot}(\{\Gamma_k \mid k \in \mathcal{K}\})$, the *total vector alphabet* over $\{\Gamma_k \mid k \in \mathcal{K}\}$. An *n-dimensional vector alphabet* (over $\{\Gamma_k \mid k \in \mathcal{K}\}$) is a subset of $\mathrm{tot}(\{\Gamma_k \mid k \in \mathcal{K}\})$. The *component-wise concatenation* of n-dimensional vector letters $v = \prod_{k\in\mathcal{K}} v_k$ and $w = \prod_{k\in\mathcal{K}} w_k$ is defined by $v \circ w = \prod_{k\in\mathcal{K}} v_k w_k$.

For a vector alphabet $\Gamma \subseteq \mathrm{tot}(\{\Gamma_k \mid k \in \mathcal{K}\})$, the homomorphism $\mathrm{coll}_\Gamma : \Gamma^* \to \prod_{k\in\mathcal{K}} \Gamma_k^*$ is defined by $\mathrm{coll}_\Gamma(v_1 v_2 \cdots v_m) = v_1 \circ v_2 \circ \cdots \circ v_m$ and yields the *collapse* of a sequence of vector letters from Γ into a word vector. Γ is omitted if clear from the context. For instance, $\mathrm{coll}\left(\binom{\lambda}{a}\binom{b}{c}\binom{d}{\lambda}\right) = \binom{\lambda}{a} \circ \binom{b}{c} \circ \binom{d}{\lambda} = \binom{bd}{ac}$.

To define computations and (vector) behavior of a vector team automaton $\mathcal{V}$, we define its *underlying automaton* $\mathrm{und}(\mathcal{V}) = (Q, \mathrm{tot}(\{\Sigma_i \mid i \in \mathcal{I}\}), \delta^v, I)$. Now $\mathbf{C}_\mathcal{V} = \mathbf{C}_{\mathrm{und}(\mathcal{V})}$ and $\mathbf{B}_\mathcal{V} = \mathbf{B}_{\mathrm{und}(\mathcal{V})} = \mathrm{pres}_{\mathrm{tot}(\{\Sigma_i \mid i\in\mathcal{I}\})}(\mathbf{C}_\mathcal{V})$. Its *vector behavior* is the collapse of the sequences of vector letters of its behavior: $\mathbf{V}_\mathcal{V} = \mathrm{coll}(\mathbf{B}_\mathcal{V})$.

In the VCCS framework, Individual Token Net Controllers (ITNCs) have been defined as a specific control mechanism with an operational motivation. They are a special type of Petri nets, designed to follow and control the progress of components of a system using individual tokens, one for each component. These tokens are distributed over the places, to indicate the local state of each component. The net's global states are vectors of places, each entry corresponding to the current local state of a component. These distributions of the individual tokens over the places are called *markings*. The events of an ITNC model synchronizations between components. To occur, an event needs certain individual tokens as input from its adjacent places and when it occurs it produces the same tokens as output in (in general) other places. In this way, the individual tokens used by an event determine which components take part in the synchronization. Events are labeled by vector letters with an entry for each component. An entry is empty iff the corresponding component does not take part in the synchronization (label-consistency). Otherwise, the corresponding component participates by executing the action mentioned.

In ITNCs, each individual token uniquely determines a sequential subnet (a state machine or automaton with labeled transitions) and each event represents a synchronization of transitions from these automata. Initial markings are any combination of initial states of each of the automata.

Formally, an (n-dimensional) ITNC (n-ITNC) consists of an underlying *(n-dimensional) Vector Labeled Individual Token Net* (n-VLITN), i.e., a labeled net with a specified set of n individual tokens, and a set of *initial markings*. For the rest of this chapter we let $n \geq 1$.

Definition 4. An *n-ITNC* is defined as a construct $\mathcal{U} = (\mathcal{N}, \mathcal{M}_0)$, in which

- $\mathcal{N} = (P, T, O, F, V, \ell)$ is its *underlying n-VLITN*, denoted by $\mathrm{und}(\mathcal{U})$, with
  - P is a finite set of *places*;
  - T is a finite set of *events* such that $P \cap T = \varnothing$;
  - $O \subseteq \mathbb{N}$ is a finite, nonempty set of n integers, called the set of *tokens*;
  - † $F : (P \times T) \cup (T \times P) \to \{o \mid o \subseteq O\}$ is a *flow function* assigning subsets of O to elements of $(P \times T) \cup (T \times P)$ such that for all $j \in O$ and all $t \in T$, $\#\{p \in P \mid j \in F(p,t)\} = \#\{p \in P \mid j \in F(t,p)\} \leq 1$;
  - $V \subseteq \mathrm{tot}(\{V_j \mid j \in O\})$, where each V_j is a finite alphabet, is an n-dimensional vector alphabet of *vector labels*;
  - ‡ $\ell : T \to V$ is an *event labeling homomorphism* such that for all $j \in O$ and for all $t \in T$, $\mathrm{proj}_j(\ell(t)) \neq \lambda$ iff $j \in \bigcup_{p\in P}(F(p,t) \cup F(t,p))$;
- $\mathcal{M}_0 \subseteq \{\mu \mid \mu : O \to P\}$ such that $\mathcal{M}_0 = \prod_{j\in O} \mathrm{proj}_j(\mathcal{M}_0)$ is a set of *initial markings*.

A VLITN is represented graphically (cf. Fig. 2) by drawing its places as circles, its events as rectangles, and an arc from place (event) x to event (place) y whenever $F(x,y) \neq \varnothing$. Events are drawn together with their label and the arcs (x,y) are labeled with the elements constituting $F(x,y)$.

To define the dynamic behavior of a VLITN, markings are used to define states by the locations of the individual tokens. These markings are (total) functions assigning a place to each token, so each token appears exactly once. A marking is graphically indicated by drawing each token in its associated place. The set of all *markings* of $\mathcal{N}$ is defined as $\mathbf{M}_\mathcal{N} = \{\mu \mid \mu : O \to P\}$. By adding a subset of $\mathbf{M}_\mathcal{N}$ as initial markings to a VLITN, an ITNC is defined. Initially, each token can be in one (of several) places. Any such combination of initial places for each of the tokens is a possible initial marking.

An event t is *enabled* (can occur) at a marking μ, denoted by $\mu[t\rangle_\mathcal{N}$, whenever each place p for which $F(p,t) \neq \varnothing$ contains at least the tokens specified in $F(p,t)$. Formally, $F(p,t) \subseteq \{j \in O \mid \mu(j) = p\}$ for all $p \in P$. If t consequently *fires* (occurs)

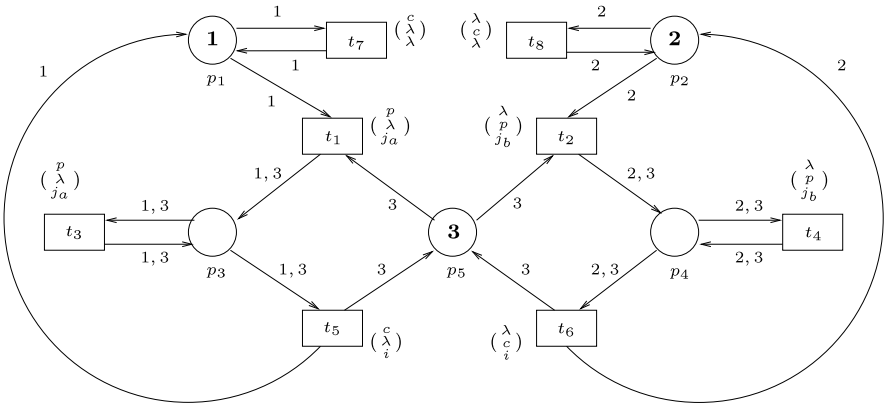

Fig. 2. 3-ITNC $\mathcal{U}$.

from marking μ to ν, denoted by $\mu[t\rangle_{\mathcal{N}}\,\nu$, all those tokens are removed and each place p for which $F(t,p) \neq \varnothing$ receives the tokens specified in $F(t,p)$, i.e., ν is defined by $\nu(j) = p$ if $j \in F(t,p)$ for $p \in P$, and else $\nu(j) = \mu(j)$. For each event t, use(t) is the set $\bigcup_{p\in P}(F(p,t) \cup F(t,p))$ of *tokens used by* t.

Condition † in Definition 4 guarantees that every VLITN is *1-throughput*: for each event t, the tokens in $\bigcup_{p\in P} F(p,t)$ are those in $\bigcup_{p\in P} F(t,p)$. Hence use$(t) = \bigcup_{p\in P} F(p,t) = \bigcup_{p\in P} F(t,p)$. This condition guarantees that after an event has fired, no individual tokens were added to or have disappeared from the VLITN, i.e., the resulting token distribution is again a marking of the VLITN.

Condition ‡ in Definition 4 guarantees that every VLITN is *label consistent*: for each event t, the nonempty entries in its vector label correspond to the tokens used by t.

It is now not difficult to imagine how by projecting on one token (in the marking and the flow function) one obtains a state machine describing the (unrestricted) behavior of one component.

If $t_1, t_2, \ldots \in T$ and $\mu_0 \in \mathbf{M}_{\mathcal{N}}$ are such that there exist markings $\mu_1, \mu_2, \ldots$ with $\mu_{i-1}[t_i\rangle_{\mathcal{N}}\,\mu_i$, for all $i \geq 1$, then $t_1 t_2 \cdots$ is a *firing sequence* of $\mathcal{N}$ starting in μ_0, denoted by $\mu_0[t_1 t_2 \cdots\rangle_{\mathcal{N}}$. $\mathcal{N}$ is omitted if it is clear from the context.

The dynamic (sequential) behavior of $\mathcal{U}$ consists of firing sequences of und$(\mathcal{U})$ that start in an initial marking of $\mathcal{U}$. The set of all *firing sequences* of $\mathcal{U}$ is defined as $\mathbf{FS}_{\mathcal{U}} = \{u \in T^* \mid \mu_0[u\rangle_{\mathcal{N}}\,\nu, \mu_0 \in \mathcal{M}_0\}$. Its set of all *reachable markings* is defined as $\mathbf{M}_{\mathcal{U}} = \{\nu \in \mathbf{M}_{\mathcal{N}} \mid \mu_0[u\rangle_{\mathcal{N}}\,\nu, \mu_0 \in \mathcal{M}_0, u \in T^*\}$, its *behavior* as $\mathbf{B}_{\mathcal{U}} = \{\ell(u) \in V^* \mid u \in \mathbf{FS}_{\mathcal{U}}\}$, and its *vector behavior* as $\mathbf{V}_{\mathcal{U}} = \text{coll}\,(\mathbf{B}_{\mathcal{U}})$.

An ITNC thus exhibits sequential behavior defined in terms of the firing sequences of its underlying VLITN. However, contrary to a finite automaton an ITNC also allows concurrent behavior, because independent events (i.e., events using disjoint sets of tokens) may be simultaneously enabled and can then fire in any order. This leads to an independence relation over the vector labels of the ITNC, similar to the independence relation used in *trace theory* [5].

Example 5. Computers A and B share a printer. A critical section prohibits them to access the printer at the same time. We model this with a 3-ITNC $\mathcal{U}$ and the following actions. A computer can c(alculate) or p(rint). The printer can be i(dle) or be printing a j(ob) for computer A (j_a) or B (j_b). Some actions are synchronized to enable the printer to indicate, while printing, which computer is printing by synchronizing p with either j_a or j_b. U is drawn in Fig. 2. Its token set is {**1**,**2**,**3**} and its set of initial markings is $\{(p_1, p_2, p_5)\}$.

From the initial marking, both computers can (concurrently) calculate by firing t_7/t_8, or one of them can print by firing t_1/t_2. In case one of them starts printing, token **3** becomes unavailable for the other. The printing computer can then either continue printing or return to calculate, in which case the printer becomes idle and token **3** becomes available for both computers again. Concurrently with the printing computer, the other can calculate but not print. These processes can be repeated and printing can be interchanged between both computers. Thus $t_8 t_7 t_1 t_3 t_8 t_5 t_7 t_2$ is a firing sequence of $\mathcal{U}$. Since $\{t_8, t_7\}$, $\{t_3, t_8\}$, and $\{t_7, t_2\}$ are pairs of events that can fire concurrently, also $u = t_7 t_8 t_1 t_8 t_3 t_5 t_2 t_7 \in \mathbf{FS}_{\mathcal{U}}$. The behavior of $\mathcal{U}$ thus includes $\ell(u) = \binom{c}{\lambda}\binom{\lambda}{c}\binom{p}{j_a}\binom{\lambda}{c}\binom{p}{j_a}\binom{c}{i}\binom{\lambda}{p}\binom{c}{\lambda}$ and its vector behavior coll $(\ell(u)) = \binom{cppcc}{j_a j_a i j_b}$.

This example moreover illustrates a main property distinguishing ITNCs from vector team automata, viz. that in an ITNC distinct actions may synchronize.

4. From team automata to ITNCs

To translate a vector team automaton $\mathcal{V}$ into an ITNC $PN(\mathcal{V})$, we use the construction sketched in Fig. 3.

The individual tokens of $PN(\mathcal{V})$ correspond to the component automata in $\mathcal{S}$. Hence the set of tokens of $PN(\mathcal{V})$ is $\mathcal{I}$. The (local) states of the component automata correspond to indexed places of $PN(\mathcal{V})$: if state q belongs to Q_i, with $i \in \mathcal{I}$, then $PN(\mathcal{V})$ thus has a place $[q,i]$. The transitions of $\mathcal{V}$ are the labeled events of $PN(\mathcal{V})$. For a transition $(q, \underline{a}, q') \in \delta^v$, $PN(\mathcal{V})$ thus has event $[q, \underline{a}, q']$ labeled by $\underline{a}$. Moreover, this event uses exactly those tokens which correspond to the

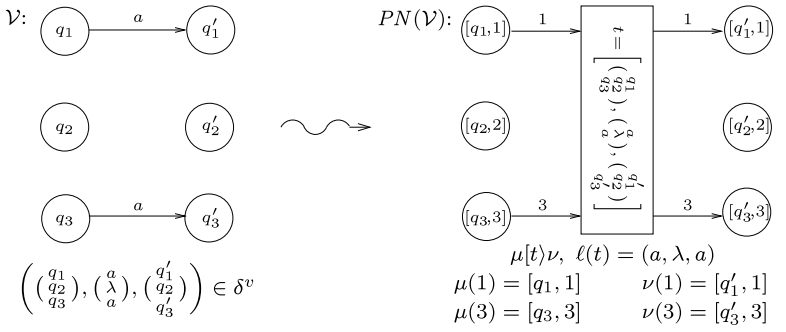

Fig. 3. Sketch of the construction of Petri net $PN(\mathcal{V})$.

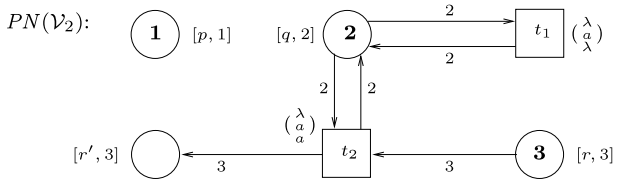

Fig. 4. Petri nets $PN(\mathcal{V}_2) = PN(\mathcal{V}_3)$.

component automata taking part in the synchronization $(q, \underline{a}, q')$. We call the set of indices of those component automata that participate in the execution of a vector action the *carrier* of that action. Hence, for a vector action $\underline{a}$, its *carrier* is defined as $\text{carrier}(\underline{a}) = \{i \in \mathcal{I} \mid \text{proj}_i(\underline{a}) \neq \lambda\}$.

The flow function of $PN(\mathcal{V})$ enforces that each event $[q, \underline{a}, q']$ uses exactly the tokens corresponding to the component automata taking part in $\underline{a}$ and, moreover, that these tokens are in the correct places (local states): whenever $\text{proj}_i(\underline{a}) \neq \lambda$, then $F([\text{proj}_i(q), i], [q, \underline{a}, q']) = F([q, \underline{a}, q'], [\text{proj}_i(q'), i]) = \{i\}$, while for all other places p of $PN(\mathcal{V})$, $i \notin F(p, [q, \underline{a}, q']) \cup F([q, \underline{a}, q'], p)$.

A marking μ of $PN(\mathcal{V})$ corresponds to a state q of $\mathcal{V}$ if μ puts token i in the place associated to the ith element of q, i.e., $\mu = \prod_{i \in \mathcal{I}} [\text{proj}_i(q), i]$. Note that for every state q of $\mathcal{V}$ there is a unique corresponding marking, which we denote by μ_q. Conversely, every marking μ of $PN(\mathcal{V})$ corresponds to a state of $\mathcal{V}$, provided that each token i is assigned to a place indexed with i.

Initial markings of $PN(\mathcal{V})$ correspond to initial states of $\mathcal{V}$: if $q \in I_i$, for $i \in \mathcal{I}$, then $[q, i]$ is an initial place for token i.

Definition 6. Let $\mathcal{V} = (Q, \Sigma, \delta^v, I)$ be a vector team automaton over $\mathcal{S}$. Then $PN(\mathcal{V}) = (\mathcal{N}, \mathcal{M}_0)$, in which

- $\mathcal{N} = (P, T, \mathcal{I}, F, V, \ell)$, with
  - $P = \bigcup_{i \in \mathcal{I}} \{[q, i] \mid q \in Q_i\}$;
  - $T = \{[q, \underline{a}, q'] \mid (q, \underline{a}, q') \in \delta^v\}$;
  - $F : (P \times T) \cup (T \times P) \to \mathcal{K}$ is defined by
    $$F([\text{proj}_i(q), i], [q, \underline{a}, q']) = F([q, \underline{a}, q'], [\text{proj}_i(q'), i]) = \{i\} \cap \text{carrier}(\underline{a});$$
  - $V = \{\underline{a} \mid (q, \underline{a}, q') \in \delta^v \text{ for some } q, q' \in Q\}$;
  - $\ell : T \to V$ is defined by $\ell([q, \underline{a}, q']) = \underline{a}$;
- $\mathcal{M}_0 = \{\mu_q \mid q \in I\}$.

It is straightforward to verify that $PN(\mathcal{V})$ satisfies the definition of an ITNC, in particular $\mathcal{N}$ is 1-throughput and label consistent and $\mathcal{M}_0$ is a set of markings.

Proposition 7. *Let $\mathcal{V}$ be a vector team automaton over $\mathcal{S}$. $PN(\mathcal{V})$ is an ITNC.*

Example 8 (*Example* 3 *Cont.*)**.** Fig. 4 depicts $PN(\mathcal{V}_2)$, obtained by applying the construction of Definition 6 to $\mathcal{V}_2$ from Fig. 1. For $\mathcal{V}_3$, $((p, q, r), (\lambda, a, \lambda), (p, q, r))$ and $((p, q, r'), (\lambda, a, \lambda), (p, q, r'))$ are mapped to the same event: $PN(\mathcal{V}_3) = PN(\mathcal{V}_2)$.

At this point, one might be inclined to conclude – incorrectly – that the computations of a vector team automaton $\mathcal{V}$ have a one-to-one correspondence with the firing sequences of the ITNC $PN(\mathcal{V})$ so that they exhibit the same behavior.

Example 9. Fig. 5 depicts vector team automaton $\mathcal{V}_4$ over component automata $\{\mathcal{C}_4, \mathcal{C}_5\}$. Fig. 6 depicts the ITNC $PN(\mathcal{V}_4)$ with $\mathcal{M}_0 = \{([q_1, 4], [q_2, 5])\}$. As $\text{use}(t_1) \cap \text{use}(t_2) = \varnothing$, events t_1 and t_2 are independent. In the initial marking both are enabled, so they can fire in any order. Indeed $\mathbf{B}_{PN(\mathcal{V}_4)} = \{\lambda, (a, \lambda), (\lambda, b), (a, \lambda)(\lambda, b), (\lambda, b)(a, \lambda)\}$. However, any nontrivial computation of $\mathcal{V}_4$ first executes (a, λ) by $((q_1, q_2), (a, \lambda), (q'_1, q_2))$ corresponding with t_1. Thus $\mathbf{B}_{\mathcal{V}_4} = \{\lambda, (a, \lambda), (a, \lambda)(\lambda, b)\} \neq \mathbf{B}_{PN(\mathcal{V}_4)}$.

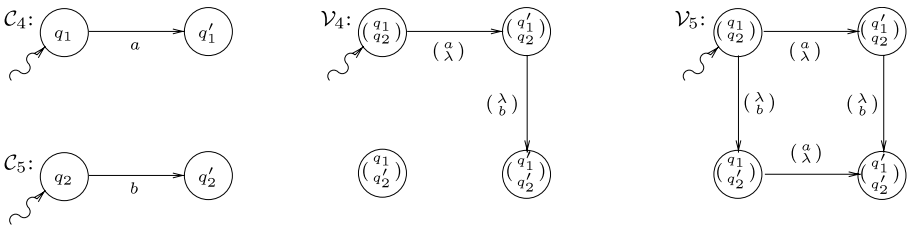

Fig. 5. Vector team automata $\mathcal{V}_4$ and $\mathcal{V}_5$ over component automata $\{\mathcal{C}_4, \mathcal{C}_5\}$.

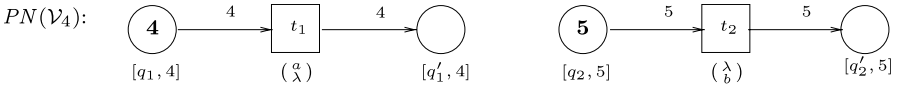

Fig. 6. ITNC $PN(\mathcal{V}_4)$.

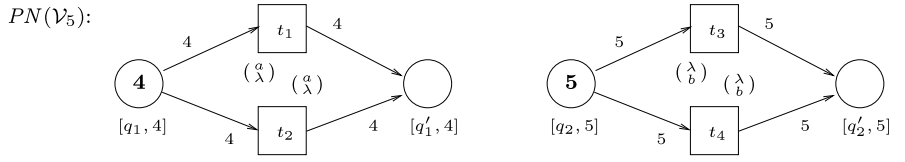

Fig. 7. ITNC $PN(\mathcal{V}_5)$ associated to vector team automaton $\mathcal{V}_5$ over $\{\mathcal{C}_4, \mathcal{C}_5\}$.

An enabled event in an ITNC can generally fire regardless of the whereabouts of tokens it does not use. Hence, as shown in Example 9, independent events enabled in an ITNC $PN(\mathcal{V})$ can occur in any order, even if the vector actions of the corresponding transitions in $\mathcal{V}$ cannot.

Example 10 (*Example* 8 *Cont.*)**.** Since token $\mathbf{2} \in \text{use}(t_1) \cap \text{use}(t_2)$, events t_1 and t_2 are not independent. Both events are enabled in the initial marking of $PN(\mathcal{V}_2)$ and can clearly be fired in any order, i.e., $\{(\lambda, a, \lambda)(\lambda, a, a), (\lambda, a, a)(\lambda, a, \lambda)\} \subseteq \mathbf{B}_{PN(\mathcal{V}_2)}$. In fact, whether or not t_1 can fire may be decided regardless of the whereabouts of **1** and **3**. In $\mathcal{V}_2$, however, (λ, a, λ) can only be executed – through the transition $((p, q, r'), (\lambda, a, \lambda), (p, q, r'))$ corresponding to t_1 – when the third component automaton is in local state r', i.e., after (λ, a, a) is executed through the transition $((p, q, r), (\lambda, a, a), (p, q, r'))$ corresponding to t_2.

Summarizing, while ITNCs allow independent events to fire in any order, vector transitions of vector team automata involving disjoint sets of component automata cannot always be executed in any order: transitions depend on combinations of local states from its component automata, so executing an action in a given local state might depend on the current states of other component automata (*state sharing*).

Definition 11. Vector team automaton $\mathcal{V} = (Q, \Sigma, \delta^v, I)$ over $\mathcal{S}$ is *non-state-sharing* if whenever $(p, \underline{a}, p') \in \delta^v$, then for all $q \in Q$ such that for all $i \in \text{carrier}(\underline{a})$, $\text{proj}_i(q) = \text{proj}_i(p)$, we have $(q, \underline{a}, q') \in \delta^v$ with $\text{proj}_i(q') = \text{proj}_i(p')$ for all $i \in \text{carrier}(\underline{a})$, and $\text{proj}_i(q') = \text{proj}_i(q)$ for all other i.

As a consequence, synchronizations involving disjoint sets of component automata are independent and can be executed concurrently. Thus ITNCs provide non-state-sharing vector team automata with a concurrent operational semantics defined through concurrent runs of Petri nets [11].

Example 12 (*Example* 9 *Cont.*)**.** Fig. 7 depicts $PN(\mathcal{V}_5)$, obtained from non-state-sharing vector team automaton $\mathcal{V}_5$ of Fig. 5. Clearly $\mathbf{B}_{\mathcal{V}_5} = \mathbf{B}_{PN(\mathcal{V}_5)} = \mathbf{B}_{PN(\mathcal{V}_4)}$.

5. Main results

As we will now show, the (vector) behavior of a non-state-sharing vector team automaton $\mathcal{V}$ equals that of the ITNC $PN(\mathcal{V})$, because every finite computation of $\mathcal{V}$ can be simulated by a firing sequence in $PN(\mathcal{V})$, and vice versa. To prove this, note that the occurrence of a transition $(p, \underline{a}, p')$ of $\mathcal{V}$ in a computation of $\mathcal{V}$ can be simulated by event $[p, \underline{a}, p']$ firing from marking μ_p to $\mu_{p'}$. For this, $\mathcal{V}$ need not be non-state-sharing. To prove the other way around, it would be convenient if $\mu[t\rangle\nu$ in $PN(\mathcal{V})$, for some $t = [p, \underline{a}, p']$, would imply $\mu = \mu_p$ and $\nu = \mu_{p'}$, with μ_p and $\mu_{p'}$ the unique markings corresponding to p and p', resp. This, however, in general is not the case. Even if $\mu = \mu_q$, for some $q \in Q$, p and q may still differ. This is due to the property of ITNCs that for an event to occur, the whereabouts of the tokens it does not use is irrelevant. If, however, $\mathcal{V}$ is non-state-sharing, then we know that $PN(\mathcal{V})$ has an event $t' = [q, \underline{a}, q']$ such that $\text{proj}_i(q') = \text{proj}_i(p')$, for all $i \in \text{carrier}(\underline{a})$. The occurrence of t can now be simulated by $\mu_q[t'\rangle\nu$, with $\nu = \mu_{q'}$ corresponding with transition $(q, \underline{a}, q')$ of $\mathcal{V}$.

A concrete example of the described situation occurs in the ITNC $PN(\mathcal{V}_5)$ of Fig. 7, where $([q_1, 4], [q_2', 5])[t_1\rangle ([q_1', 4], [q_2', 5])$ with $t_1 = [(q_1, q_2), (a, \lambda), (q_1', q_2)]$. However, $PN(\mathcal{V}_5)$ also has event $t_2 = [(q_1, q_2'), (a, \lambda), (q_1', q_2')]$ to simulate t_1 occurring at $\mu_{(q_1, q_2')} = ([q_1, 4], [q_2', 5])$, and leading to $\mu_{(q_1', q_2')} = ([q_1', 4], [q_2', 5])$. When a firing sequence of $PN(\mathcal{V}_5)$ contains $([q_1, 4], [q_2', 5])[t_1\rangle([q_1', 4], [q_2', 5])$, we may thus use t_2 in the corresponding computation of $\mathcal{V}_5$.

To avoid cumbersome descriptions we call two transitions $(p, \underline{a}, p'), (q, \underline{a}, q') \in \Delta_a^v(\mathcal{S})$ *clones* whenever $(\text{proj}_i(p), \text{proj}_i(p')) = (\text{proj}_i(q), \text{proj}_i(q'))$, for all $i \in \text{carrier}(\underline{a})$. If $\mathcal{V}$ is a non-state-sharing vector team automaton, then it follows that whenever $(p, \underline{a}, p') \in \delta^v$, all clones of $(p, \underline{a}, p')$ are transitions of $\mathcal{V}$.

Lemma 13. *Let $\mathcal{V} = (Q, \Sigma, \delta^v, I)$ be a vector team automaton over $\mathcal{S}$ and let $PN(\mathcal{V}) = (\mathcal{N}, \mathcal{M}_0)$, with $\mathcal{N} = (P, T, \mathcal{I}, F, V, \ell)$.*

1. *If $(p, \underline{a}, p') \in \delta^v$, then $\mu_p[[p, \underline{a}, p']\rangle\mu_{p'}$ in $PN(\mathcal{V})$;*
2. *If $\mu_q[[p, \underline{a}, p']\rangle\nu$ in $PN(\mathcal{V})$, for $p, p', q \in P$ and $\underline{a} \in tot(\{\Sigma_i \mid i \in \mathcal{I}\})$, then $\nu = \mu_{q'}$, where $q' \in Q$ is the unique state such that $(q, \underline{a}, q')$, $(p, \underline{a}, p')$ are clones.*

Proof. (1) Let $(p, \underline{a}, p') \in \delta^v$. Then $[p, \underline{a}, p']$ is an event of $PN(\mathcal{V})$. It is easy to see that $[p, \underline{a}, p']$ is enabled at μ_p: by the construction of $PN(\mathcal{V})$, to be able to fire, $[p, \underline{a}, p']$ needs for all $i \in \text{carrier}(\underline{a})$, token i in place $[\text{proj}_i(p), i]$. This requirement is satisfied at μ_p because by definition $\mu_p(i) = [\text{proj}_i(p), i]$, for all $i \in \text{carrier}(\underline{a})$. Let ν be the marking such that $\mu_p[[p, \underline{a}, p']\rangle\nu$ in $PN(\mathcal{V})$. Then, again by the construction, $\nu(i) = [\text{proj}_i(p'), i]$, for all $i \in \text{carrier}(\underline{a})$, and $\nu(i) = \mu(i) = [\text{proj}_i(p), i]$, for all $i \in \mathcal{I}$ for which $\text{proj}_i(\underline{a}) = \lambda$. Hence $\nu = \mu_{p'}$.

(2) Let $\mu_q[[p, \underline{a}, p']\rangle\nu$ in $PN(\mathcal{V})$, with $p, p', q \in P$ and $\underline{a} \in \text{tot}(\{\Sigma_i \mid i \in \mathcal{I}\})$. Then for all $i \in \text{carrier}(\underline{a})$, $\mu_q(i) = [\text{proj}_i(p), i]$. Since by definition $\mu_q(i) = [\text{proj}_i(q), i]$, for all $i \in \mathcal{I}$, it follows that $\text{proj}_i(p) = \text{proj}_i(q)$, for all $i \in \text{carrier}(\underline{a})$. Let $q' \in Q$ be such that $(q, \underline{a}, q')$ and (p, a, p') are clones. Thus $\text{proj}_i(q') = \text{proj}_i(p')$, for all $i \in \text{carrier}(\underline{a})$ and $\text{proj}_i(q') = \text{proj}_i(q)$, for all $i \in \mathcal{I}$ such that $\text{proj}_i(\underline{a}) = \lambda$. Given $\mu_q[[p, \underline{a}, p']\rangle\nu$ in $PN(\mathcal{V})$, the definition of F implies that $\nu(i) = [\text{proj}_i(p'), i]$, for all $i \in \text{carrier}(\underline{a})$, and $\nu(i) = \mu_q(i) = [\text{proj}_i(q), i]$, for all $i \in \mathcal{I}$ such that $\text{proj}_i(\underline{a}) = \lambda$. Consequently, $\nu = \mu_{q'}$. □

From Lemma 13(2) it immediately follows that $\mu_p[t\rangle\nu$ in $PN(\mathcal{V})$ implies that there exists a state p' in $\mathcal{V}$ such that $\nu = \mu_{p'}$. Hence, even if $\mathcal{V}$ is not non-state-sharing, each reachable marking of $PN(\mathcal{V})$ corresponds to a state of $\mathcal{V}$.

Lemma 14. *Let $\mathcal{V} = (Q, \Sigma, \delta^v, I)$ be a non-state-sharing vector team automaton over $\mathcal{S}$ and let $PN(\mathcal{V}) = (\mathcal{N}, \mathcal{M}_0)$, with $\mathcal{N} = (P, T, \mathcal{I}, F, V, \ell)$. Let $m \geq 1$ and let $(q_{j-1}, \underline{a}_j, q_j) \in \delta^v$, for all $1 \leq j \leq m$. Then $[q_0, \underline{a}_1, q_1][q_1, \underline{a}_2, q_2] \cdots [q_{m-1}, \underline{a}_m, q_m] \in \mathbf{FS}_{PN(\mathcal{V})}$ iff for all $1 \leq j \leq m$, there exists a clone $(p_{j-1}, \underline{a}_j, p_j)$ of $(q_{j-1}, \underline{a}_j, q_j)$ such that $p_0\underline{a}_1p_1\underline{a}_2p_2 \cdots p_{m-1}\underline{a}_mp_m \in \mathbf{C}_{\mathcal{V}}$.*

Proof. *(If)* Let $p_0\underline{a}_1p_1\underline{a}_2p_2 \cdots p_{m-1}\underline{a}_mp_m \in \mathbf{C}_{\mathcal{V}}$, with $(p_{j-1}, \underline{a}, p_j)$ a clone of $(q_{j-1}, \underline{a}_j, q_j)$, for all $1 \leq j \leq m$. Thus $p_0 \in I$ and $(p_{j-1}, \underline{a}_j, p_j) \in \delta^v$, for all $1 \leq j \leq m$. From Lemma 13(1), $\mu_{p_{j-1}}[[p_{j-1}, \underline{a}_j, p_j]\rangle\mu_{p_j}$ in $PN(\mathcal{V})$, for all $1 \leq j \leq m$. Since $(p_{j-1}, \underline{a}_j, p_j)$ and $(q_{j-1}, \underline{a}_j, q_j)$ are clones for all $1 \leq j \leq m$, for all places s of $PN(\mathcal{V})$, $F(s, [p_{j-1}, \underline{a}_j, p_j]) = F(s, [q_{j-1}, \underline{a}_j, q_j])$ and $F([p_{j-1}, \underline{a}_j, p_j], s) = F([q_{j-1}, \underline{a}_j, q_j], s)$, for all $1 \leq j \leq m$. Thus, for all $1 \leq j \leq m$, $\mu_{p_{j-1}}[[q_{j-1}, \underline{a}_j, q_j]\rangle\mu_{p_j}$ in $PN(\mathcal{V})$ and $\mu_{p_0}[[q_0, \underline{a}_1, q_1]\rangle\mu_{p_1}[[q_1, \underline{a}_2, q_2]\rangle\mu_{p_2} \cdots \mu_{p_{m-1}}[[q_{m-1}, \underline{a}_m, q_m]\rangle\mu_{p_m}$ in $PN(\mathcal{V})$. As $p_0 \in I$, we have $\mu_{p_0} \in \mathcal{M}_0$, and thus $[q_0, \underline{a}_1, q_1][q_1, \underline{a}_2, q_2] \cdots [q_{m-1}, \underline{a}_m, q_m] \in \mathbf{FS}_{PN(\mathcal{V})}$.

(Only if) Let $[q_0, \underline{a}_1, q_1][q_1, \underline{a}_2, q_2] \cdots [q_{m-1}, \underline{a}_m, q_m] \in \mathbf{FS}_{PN(\mathcal{V})}$ and, for $0 \leq j \leq m$, μ_j be markings such that $\mu_{j-1}[[q_{j-1}, \underline{a}_j, q_j]\rangle\mu_j$ in $PN(\mathcal{V})$, for all $1 \leq j \leq m$. Without loss of generality we may assume μ_0 is an initial marking of $PN(\mathcal{V})$. Let $p_0 \in I$ be the initial state of $\mathcal{V}$ so that $\mu_0 = \mu_{p_0}$. From Lemma 13(2) and since $\mathcal{V}$ is non-state-sharing, $\mu_{p_0}[[q_0, \underline{a}_1, q_1]\rangle\mu_{p_1}$ and $\mu_{p_0}[[p_0, \underline{a}_1, p_1]\rangle\mu_{p_1}$ in $PN(\mathcal{V})$, with $(q_0, \underline{a}_1, q_1)$, $(p_0, \underline{a}_1, p_1)$ being clones. Repeatedly using this argumentation, we know that for each $1 \leq j \leq m$, there exists $p_j \in Q$ such that $\mu_{p_{j-1}}[[q_{j-1}, \underline{a}_j, q_j]\rangle\mu_{p_j}$ and $\mu_{p_{j-1}}[[p_{j-1}, \underline{a}_j, p_j]\rangle\mu_{p_j}$ in $PN(\mathcal{V})$, with $(q_{j-1}, \underline{a}_j, q_j)$, $(p_{j-1}, \underline{a}_j, p_j)$ being clones. Hence, $\mathcal{V}$ has transitions $(p_0, \underline{a}_1, p_1)$, $(p_1, \underline{a}_2, p_2)$, …, $(p_{m-1}, \underline{a}_m, p_m)$ and since $p_0 \in I$, we obtain $p_0\underline{a}_1p_1\underline{a}_2p_2 \cdots p_{m-1}\underline{a}_mp_m \in \mathbf{C}_{\mathcal{V}}$. □

The labeling of events of $PN(\mathcal{V})$ agrees with the vector labels of corresponding transitions of $\mathcal{V}$. Thus, the behavior of $\mathcal{V}$ coincides with that of $PN(\mathcal{V})$ insofar it is based on nontrivial computations and nonempty firing sequences. Also, $\lambda \in \mathbf{FS}_{PN(\mathcal{V})}$ iff the set of initial markings of $PN(\mathcal{V}) \neq \varnothing$ iff the set of initial states of $\mathcal{V} \neq \varnothing$ iff $\mathcal{V}$ has a trivial computation. This is our first main result.

Theorem 15. *Let $\mathcal{V}$ be a non-state-sharing vector team automaton over $\mathcal{S}$. Then $\mathbf{B}_{PN(\mathcal{V})} = \mathbf{B}_{\mathcal{V}}$ and $\mathbf{V}_{PN(\mathcal{V})} = \mathbf{V}_{\mathcal{V}}$.*

We conclude with an observation relating the ITNC obtained by applying the construction of Definition 6 to the subteam determined by $\mathcal{K}$ of a vector team automaton $\mathcal{V}$ to a *subnet* of $PN(\mathcal{V})$. Mirroring the way we defined subteams of vector team automata, a subnet of an ITNC is obtained by focusing on a subset of its set of individual tokens.

Let the restriction of a function $f : A \to A'$ to a subset C of its domain A be denoted by $f \restriction C : C \to A'$ and defined by $(f \restriction C)(c) = f(c)$, for all $c \in C$.

Definition 16. Let $\mathcal{V}$ be a vector team automaton over $\mathcal{S}$, let the ITNC $PN(\mathcal{V})$ be $\mathcal{U} = (\mathcal{N}, \mathcal{M}_0)$, with $\mathcal{N} = (P, T, \mathcal{I}, F, V, \ell)$, and let $\mathcal{K} \subseteq \mathcal{I}$. Then the *subnet $SUB_{\mathcal{K}}(\mathcal{U})$ of $\mathcal{U}$ determined by $\mathcal{K}$* is defined as $SUB_{\mathcal{K}}(\mathcal{U}) = (\mathcal{N}_{\mathcal{K}}, (\mathcal{M}_0)_{\mathcal{K}})$, in which

- $\mathcal{N}_{\mathcal{K}} = (P_{\mathcal{K}}, T_{\mathcal{K}}, \mathcal{K}, F_{\mathcal{K}}, V_{\mathcal{K}}, \ell_{\mathcal{K}})$, with
  - $P_{\mathcal{K}} = \{[q, k] \mid q \in Q_k,\ k \in \mathcal{K}\}$;
  - $T_{\mathcal{K}} = \{[\text{proj}_{\mathcal{K}}(q), \text{proj}_{\mathcal{K}}(\underline{a}), \text{proj}_{\mathcal{K}}(q')] \mid [q, \underline{a}, q'] \in T \text{ for some } q, q' \in Q \text{ and } \mathcal{K} \cap \text{carrier}(\underline{a}) \neq \varnothing\}$;

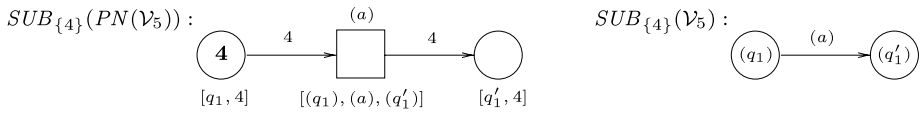

Fig. 8. ITNC $SUB_{\{4\}}(PN(\mathcal{V}_5))$ and subteam $SUB_{\{4\}}(\mathcal{V}_5)$.

- $F_{\mathcal{K}}\colon (P_{\mathcal{K}} \times T_{\mathcal{K}}) \cup (T_{\mathcal{K}} \times P_{\mathcal{K}}) \to \mathcal{K}$ is defined by $F_{\mathcal{K}}([\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')], [\mathrm{proj}_i(q'), i]) = F_{\mathcal{K}}([\mathrm{proj}_i(q), i], [\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')]) = \{i\} \cap \mathrm{carrier}(\underline{a})$;
- $V_{\mathcal{K}} = \{\underline{b} \mid [p, \underline{b}, p'] \in T_{\mathcal{K}} \text{ for some } p, p' \in \mathrm{proj}_{\mathcal{K}}(Q)\}$;
- $\ell_{\mathcal{K}}\colon T_{\mathcal{K}} \to V_{\mathcal{K}}$ is defined by $\ell_{\mathcal{K}}([p, \underline{b}, p']) = \underline{b}$;

• $(\mathcal{M}_0)_{\mathcal{K}} = \{\mu \restriction \mathcal{K} \mid \mu \in \mathcal{M}_0\}$.

Thus, a subnet $SUB_{\mathcal{K}}(PN(\mathcal{V}))$ is not simply defined by a local operation on the elements of the ITNC $PN(\mathcal{V})$, but by a (syntactical) operation that refers to the transitions of $\mathcal{V}$ underlying the events of $PN(\mathcal{V})$ and that is based on the actual participation of the component automata forming the subteam. Essentially, $SUB_{\mathcal{K}}(PN(\mathcal{V}))$ is obtained by projecting on $\mathcal{K}$, similar to the way in which the state machines underlying an ITNC can be obtained by projecting on the individual tokens [8,9]. As a result, each event t of the subnet comprises all events $[q, \underline{a}, q']$ in the full net such that $[\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')] = t$. By definition of flow function F, whenever two events $[q, \underline{a}, q'], [p, \underline{a}, p'] \in T$ are such that $[\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')] = [\mathrm{proj}_{\mathcal{K}}(p), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(p')]$, their neighborhoods when restricted to arcs with labels from $\mathcal{K}$ are the same. The definition of $F_{\mathcal{K}}$ guarantees that also the labeled arcs connecting t with places $[p, k]$ in $SUB_{\mathcal{K}}(PN(\mathcal{V}))$ correspond to the labeled arcs connecting the original events $[q, \underline{a}, q']$ with $[p, k]$ in $PN(\mathcal{V})$.

Since $P_{\mathcal{K}} \subseteq P$, $F_{\mathcal{K}}$ may be viewed as a restriction of F to $(P_{\mathcal{K}} \times T_{\mathcal{K}}) \cup (T_{\mathcal{K}} \times P_{\mathcal{K}})$ once T has been transformed into $T_{\mathcal{K}}$. Since $V_{\mathcal{K}}$ and $\ell_{\mathcal{K}}$ agree with V and ℓ after projection and since $(\mathcal{M}_0)_{\mathcal{K}}$ is the restriction of $\mathcal{M}_0$ to $\mathcal{K}$, we may refer to $SUB_{\mathcal{K}}(PN(\mathcal{V}))$ as a subnet of the ITNC $PN(\mathcal{V})$.

Example 17 (*Example* 12 *Cont.*)**.** Fig. 8 depicts the subnet determined by $\{4\}$ of $PN(\mathcal{V}_5)$. Note how $t_1 = [(q_1, q_2), (a, \lambda), (q_1', q_2)]$ and $t_2 = [(q_1, q_2'), (a, \lambda), (q_1', q_2')]$ of $PN(\mathcal{V}_5)$ resulted in one event $[\mathrm{proj}_4((q_1, q_2)), \mathrm{proj}_4((a, \lambda)), \mathrm{proj}_4((q_1', q_2))] = [\mathrm{proj}_4((q_1, q_2')), \mathrm{proj}_4((a, \lambda)), \mathrm{proj}_4((q_1', q_2'))] = [(q_1), (a), (q_1')]$ in its subnet $SUB_{\{4\}}(PN(\mathcal{V}_5))$, reflecting the fact that the dynamics of $SUB_{\{4\}}(PN(\mathcal{V}_5))$ is based on the actual participation of $\mathcal{C}_4$ – as the only component automaton forming $SUB_{\{4\}}(\mathcal{V}5)$ – in the transitions of $\mathcal{V}_5$ underlying the events of $PN(\mathcal{V}_5)$.

Analogously, $((q_1, q_2), (a, \lambda), (q_1', q_2))$ and $((q_1, q_2'), (a, \lambda), (q_1', q_2'))$ of $\mathcal{V}_5$ result in one transition $((q_1), (a), (q_1'))$ in $SUB_{\{4\}}(\mathcal{V}_5)$.

It is straightforward to verify that $SUB_{\mathcal{K}}(PN(\mathcal{V}))$ is again an ITNC. Moreover, we have as our second main result that the same ITNC results from first restricting $\mathcal{V}$ to a subteam and then constructing its net.

Theorem 18. *Let $\mathcal{V} = (Q, \Sigma, \delta^v, I)$ be a vector team automaton over $\mathcal{S}$ and let $\mathcal{K} \subseteq \mathcal{I}$. Then $SUB_{\mathcal{K}}(PN(\mathcal{V})) = PN(SUB_{\mathcal{K}}(\mathcal{V}))$.*

Proof. We inspect Definitions 6 and 16 and Definitions 2 and 6 in an element-wise way. Let $SUB_{\mathcal{K}}(PN(\mathcal{V})) = (\mathcal{N}_1, (\mathcal{M}_0)_1)$, with $\mathcal{N}_1 = (P_1, T_1, \mathcal{K}, F_1, V_1, \ell_1)$, and let $PN(SUB_{\mathcal{K}}(\mathcal{V})) = (\mathcal{N}_2, (\mathcal{M}_0)_2)$, with $\mathcal{N}_2 = (P_2, T_2, \mathcal{K}, F_2, V_2, \ell_2)$.

Obviously, the sets of places are identical: $P_1 = P_2 = \bigcup_{k \in \mathcal{K}} \{[q, k] \mid q \in Q_k\}$.

Clearly, $T_1 = \{[\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')] \mid [q, \underline{a}, q'] \in T \text{ for some } q, q' \in Q \text{ and } \mathcal{K} \cap \mathrm{carrier}(\underline{a}) \neq \varnothing\} = \{[\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')] \mid (q, \underline{a}, q') \in \delta^v \text{ and } \mathrm{proj}_{\mathcal{K}}(\underline{a}) \neq \Lambda\} = \{[\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')] \mid (\mathrm{proj}_{\mathcal{K}}(q), \mathrm{proj}_{\mathcal{K}}(\underline{a}), \mathrm{proj}_{\mathcal{K}}(q')) \in \Delta_a^v(\{\mathcal{C}_k \mid k \in \mathcal{K}\}) \text{ and } (q, \underline{a}, q') \in \delta^v\} = T_2$. So the sets of events are identical.

Let $p \in P_1 = P_2$ and $t \in T_1 = T_2$. Let $i \in K$. Then $i \in F_1(p, t)$ iff there exist $q, q' \in Q$ and $\underline{a} \in V_{\mathcal{K}}$ such that $t = [\mathrm{proj}_{\mathcal{K}}(q), \underline{a}, \mathrm{proj}_{\mathcal{K}}(q')]$ and $i \in \mathrm{carrier}(\underline{a})$, and moreover $p = [\mathrm{proj}_i(q), i]$. This is equivalent with $i \in F_2(p, t)$. So $F_1(p, t) = F_2(p, t)$. Likewise, $F_1(t, p) = F_2(t, p)$ and hence the flow functions are identical.

As $T_1 = T_2 = \{[q, \underline{a}, q'] \mid (q, \underline{a}, q') \in \delta_{\mathcal{K}}^v\}$, we have $V_1 = \{\underline{b} \mid [p, \underline{b}, p'] \in T_1 \text{ forsome } p, p' \in \mathrm{proj}_{\mathcal{K}}(Q)\} = \{\underline{b} \mid (p, \underline{b}, p') \in \delta_{\mathcal{K}}^v \text{ for some } p, p' \in Q_{\mathcal{K}}\} = V_2$ and $\ell_1([r, \underline{c}, r']) = \ell_2([r, \underline{c}, r']) = \underline{c} \in V_1 = V_2$, for all $[r, \underline{c}, r'] \in T_1 = T_2$. So the vector alphabets of vector labels and the vector labeling homomorphisms are identical.

Finally, it is immediate that $(\mathcal{M}_0)_1 = \{\mu_q \restriction K \mid q \in I\} = \{\mu_{\mathrm{proj}_{\mathcal{K}}(q)} \mid \mathrm{proj}_{\mathcal{K}}(q) \in I_{\mathcal{K}}\} = (\mathcal{M}_0)_2$, thus also the sets of initial markings are identical. □

6. Conclusion

In this paper, we have considered vector team automata in which the interactions of component automata are made explicit. This led us to a relation of vector team automata to ITNCs, a state machine decomposable net model developed in the VCCS framework. While every synchronous product of (I/O) automata can directly be seen as a Petri net, this is in general not the case for team automata [4]. In fact, a restriction to non-state-sharing vector team automata is necessary.

Though related, a number of important differences remain between vector team automata and ITNCs, especially concerning the type of synchronizations that can be modeled. Whereas synchronizations in vector team automata are uniform, i.e., with a common action for the components involved, synchronizations in ITNCs may involve different actions. In this respect, ITNCs allow the modeling of more types of synchronization than team automata do. ITNCs however are not

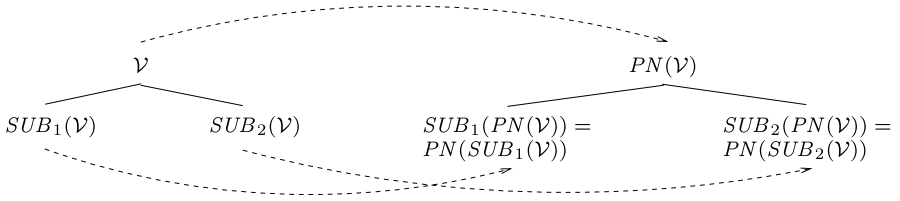

Fig. 9. Sketch of iteratively composing ITNCs.

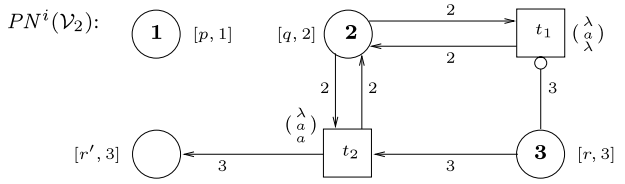

Fig. 10. Petri net $PN^i(\mathcal{V}_2)$ with an inhibitor arc.

concerned with the distinction of actions into input, output, and internal actions, which is a crucial modeling feature of team automata but outside the scope of this paper.

Finally, vector team automata, unlike ITNCs, allow the construction of hierarchical systems in a natural way, by iterated composition. Theorem 15 moreover provides a relation between non-state-sharing vector team automata and the subclass of ITNCs obtained by applying the construction of Definition 6 to vector team automata. For this particular subclass of ITNCs, the two main results of this paper (Theorems 15 and 18) hint at a way around the latter limitation of ITNCs. However, since we have no characterization of this particular subclass, in Fig. 9 indeed no more than a hint toward iteratively composing a subclass of ITNCs is sketched, in which $PN(\mathcal{V})$ might be seen as iteratively composed over $PN(SUB_1(\mathcal{V}))$ and $PN(SUB_2(\mathcal{V}))$.

To conclude, we note that the restriction to non-state-sharing team automata could be removed if we extend the construction to Petri nets with inhibitor arcs [11]. An *inhibitor arc* is a special kind of arc from a place to an event that is used to *test* for the presence of a token. Transition t_1 in Fig. 10 cannot fire in the depicted marking, as the token in $[r, 3]$ inhibits its occurrence.

Example 19 (*Example* 10 *Cont.*)**.** Fig. 10 depicts a Petri net $PN^i(\mathcal{V}_2)$ with an inhibitor arc from place $[r, 3]$ to event t_1, which could be the result from applying an adapted version of the construction of Definition 6 to $\mathcal{V}_2$ from Fig. 1. In fact, $\mathbf{B}_{\mathcal{V}_2} = \mathbf{B}_{PN^i(\mathcal{V}_2)} \neq \mathbf{B}_{PN(\mathcal{V}_2)}$ since $\{(\lambda, a, \lambda)(\lambda, a, a)\} \subseteq \mathbf{B}_{PN(\mathcal{V}_2)} \setminus \mathbf{B}_{PN^i(\mathcal{V}_2)}$. This is because t_1 has to wait until t_2 has fired and removed the token from $[r, 3]$.

We note that $\mathcal{V}_3$ ($PN(\mathcal{V}_3)$, resp.) does allow (λ, a, λ) to execute when the third automaton is in local state r (t_1 to fire regardless of the whereabouts of tokens **1** and **3**, resp.).

Acknowledgements

We thank the anonymous reviewer for careful reading and constructive remarks. We are especially grateful to Grzegorz Rozenberg, our promoter, for always having supported and encouraged our research.

References

[1] M.H. ter Beek, Team automata: a formal approach to the modeling of collaboration between system components, Ph.D. Thesis, Leiden University, 2003.
[2] M.H. ter Beek, C.A. Ellis, J. Kleijn, G. Rozenberg, Team automata for spatial access control, in: W. Prinz, M. Jarke, Y. Rogers, K. Schmidt, V. Wulf (Eds.), Proceedings European CSCW Conference, Kluwer, 2001, pp. 59–77.
[3] M.H. ter Beek, C.A. Ellis, J. Kleijn, G. Rozenberg, Synchronizations in team automata for groupware systems, Computer Supported Cooperative Work 12 (1) (2003) 21–69.
[4] J. Carmona, J. Kleijn, Interactive behaviour of multi-component systems, in: J. Cortadella, A. Yakovlev (Eds.), Proceedings ToBaCo Workshop on Token Based Computing, 2004, pp. 27–31.
[5] V. Diekert, G. Rozenberg, Book of Traces, World Scientific, 1995.
[6] G. Engels, L.P.J. Groenewegen, Towards team-automata-driven object-oriented collaborative work, in: W. Brauer, H. Ehrig, J. Karhumäki, A. Salomaa (Eds.), Formal and Natural Computing: Essays Dedicated to Grzegorz Rozenberg, in: LNCS, vol. 2300, Springer, 2002, pp. 257–276.
[7] N.W. Keesmaat, Vector controlled concurrent systems, Ph.D. Thesis, Leiden University, 1996.
[8] N.W. Keesmaat, H.C.M. Kleijn, G. Rozenberg, Vector controlled concurrent systems, Part I: basic classes, Fundamenta Informaticae 13 (1990) 275–316.
[9] N.W. Keesmaat, H.C.M. Kleijn, G. Rozenberg, Vector controlled concurrent systems, Part II: comparisons, Fundamenta Informaticae 14 (1991) 1–38.
[10] J. Kleijn, Team automata for CSCW: a survey, in: H. Ehrig, W. Reisig, G. Rozenberg, H. Weber (Eds.), Petri Net Technology for Communication-based Systems: Advances in Petri Nets, in: LNCS, vol. 2472, Springer, 2003, pp. 295–320.
[11] W. Reisig, G. Rozenberg (Eds.), Lectures on Petri Nets I (Basic Models) and II (Applications), in: LNCS, vols. 1491 and 1492, Springer, 1998.