Distributed synthesis of asynchronously communicating distributed process models

Kwantes, P.M.; Kleijn, H.C.M.; Koutny, M.; Kordon, F.; Moldt, D.

Distributed Synthesis of Asynchronously Communicating Distributed Process Models

Pieter Kwantes(✉) and Jetty Kleijn

LIACS, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands
{p.m.kwantes,h.c.m.kleijn}@liacs.leidenuniv.nl

Abstract. We investigate to what extent existing algorithms for the discovery of component models from event logs can be leveraged to a system of asynchronously communicating components. Here, Enterprise nets model local processes, while Industry nets are compositions of Enterprise nets which interact through asynchronous message passing. We investigate the relation between the behaviour of an Industry net and that of its constituting Enterprise nets and we formalise the (causal) structure of global (Industry net) behaviour in terms of a partial order derived from the message passing. Next, we specify how (existing) algorithms for the discovery of isolated processes, can be adapted to enable the discovery of Enterprise nets, and we demonstrate how to combine these Enterprise nets into an Industry net. Using the results on the structure of the global behaviour, we relate the behaviour of the Industry net thus synthesised to the behaviour of the Enterprise nets and show how fitness of the Enterprise nets (the event log provided as input is included in the behaviour of the discovered net) is preserved as fitness of the Industry net. Moreover, we discuss possible underfitting of the global model (the model exhibits more behaviour than observed in the event log) and show how it can be explained in terms of concurrency between the component models and a completeness property of the event log.

Keywords: Enterprise net · Industry net · process discovery · distributed process · asynchronous communication · partial order

1 Introduction

Industry nets have been introduced in [21] as a framework to model global communication between enterprises where the design of their internal operations is left to the local level. In this set-up, operations at the enterprise level are represented by Enterprise nets (Petri nets with input, output, and internal transitions) and Industry nets are compositions of Enterprise nets that interact by exchanging messages through channels. In [21], a method is proposed to establish global compliance of an Industry net with a reference model by local checks

M. Koutny et al. (Eds.): ToPNoC XVI, LNCS 13220, pp. 49–72, 2022.
https://doi.org/10.1007/978-3-662-65303-6_3

(of Enterprise nets) only. In this paper, we focus on the synthesis of an Industry net from the (observed) combined behaviour of a collection of collaborating enterprises.

Business process modelling and process mining are nowadays very active research areas and there are many approaches both to process discovery and conformance checking (see, eg., [6,8,12]). Here, we take advantage of this in the sense that we consider the synthesis of *distributed* processes (in the form of Industry nets) from component processes, while assuming the existence of an algorithm for the discovery of component processes (in the form of Enterprise nets).

To be precise, we assume an algorithm that discovers from an event log (a set of action sequences) a Petri net such that fitness of the discovered Petri net (the event log provided as input is included in the behaviour of the discovered net) is guaranteed. Moreover, we show how from this Petri net an Enterprise net with the same behaviour can be derived. Then, given an event log representing (observed) global behaviour of a distributed process, we first identify the behaviours of its components. Next, we show how an Industry net can be constructed by combining the Enterprise nets discovered from the respective local behaviours, such that fitness is preserved.

In order to be able to relate the given global behaviour to the behaviour of the synthesised Industry net, we first investigate, after the preliminary Sect. 2, in Sects. 3, 4, and 5, the structure of the behaviour of an Industry net in terms of the behaviours of its component Enterprise nets. In Sect. 6, working under the assumption that an algorithm for the discovery of Petri net models from isolated component behaviours is available, we explain how to derive Enterprise nets from these Petri net models. Then we show how these Enterprise nets can be used for the synthesis of Industry nets. Based on the results from the earlier sections, we then argue how fitness of the local models delivered by the original algorithm guarantees the fitness of the distributed model. Moreover, we discuss underfitting of the model (the model exhibits more behaviour than observed in the event log). In the final Sect. 7, we give an overview, compare our set-up with some approaches from the literature and briefly discuss possible future work.

This paper is a revised and extended version of the workshop paper [20]. Proofs and examples have been added. Also a notion of completeness of an event log is added as well as several results on the relationship between completeness and properties of the I-net synthesised from the event log.

2 Preliminaries

$\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers including 0. For $n \in \mathbb{N}$, we set $[n] = \{1, 2, 3, \ldots, n\}$ and if $n = 0$, then $[n] = \emptyset$. The *restriction* of a function $f : A \to B$ to a set $C \subseteq A$, is the function $f|_C : C \to B$, defined by $f|_C(b) = f(b)$ for all $b \in C$. Given a partial order $R \subseteq A \times A$, we refer to a total order $R' \subseteq A \times A$ such that $R \subseteq R'$, as a *linearisation of* R.

An *alphabet* is a finite, non-empty, set of *symbols*. Let Σ be an alphabet. A *word* over Σ is a sequence $w = a_1 \cdots a_n$, with $n \geq 0$ and $a_i \in \Sigma$, for all $i \in [n]$;

we refer to n as the *length* of w, denoted by $|w|$. If $n = 0$ then w is the *empty* word denoted by λ. The set of all words over Σ is denoted as Σ^*. Any subset of Σ^* is a *language* (over Σ). It is often convenient to consider a word $w = a_1 \cdots a_n$ with $a_i \in \Sigma$ for all $i \in [n]$, as a function $w : [n] \to \Sigma$, defined by $w(i) = a_i$ for all $i \in [n]$. The *alphabet* of w is $\mathtt{alph}(w) = \{w(i) \mid i \in [|w|]\}$. Hence $\mathtt{alph}(\lambda) = \emptyset$. For a language L, $\mathtt{Alph}(L) = \bigcup\{\mathtt{alph}(w) \mid w \in L\}$ is the set of all symbols that occur in a word of L.

If a word w is a concatenation of words v_1 and v_2, i.e., $w = v_1 v_2$, then v_1 is said to be a *prefix* of w. The set of all prefixes of w is denoted by $\mathtt{pref}(w)$. The set of all prefixes of a language L is $\mathtt{Pref}(L) = \bigcup\{\mathtt{pref}(w) \mid w \in L\}$.

The number of *occurrences of* $a \in \Sigma$ in $w \in \Sigma^*$ is defined as $\#_a(w) = |\{i \mid w(i) = a\}|$ and *the set of occurrences in* w is $\mathtt{occ}(w) = \{(a, i) \mid a \in \mathtt{alph}(w) \wedge 1 \leq i \leq \#_a(w)\}$. In addition, we allocate a position with each occurrence in w through the function $\mathtt{pos}_w : \mathtt{occ}(w) \to [|w|]$ as follows: for all $(a, i) \in \mathtt{occ}(w)$, $\mathtt{pos}_w((a, i)) = k$ if $w(k) = a$ and $\#_a(w(1) \cdots w(k)) = i$. Consider eg., $w = abbab$. Then $\mathtt{alph}(w) = \{a, b\}$ and $\mathtt{occ}(w) = \{(a, 1), (a, 2), (b, 1), (b, 2), (b, 3)\}$. Moreover, $\mathtt{pos}_w((a, 1)) = 1$, $\mathtt{pos}_w((a, 2)) = 4$, $\mathtt{pos}_w((b, 1)) = 2$, $\mathtt{pos}_w((b, 2)) = 3$ and $\mathtt{pos}_w((b, 3)) = 5$.

For a subset Δ of Σ, the projection of Σ^* on Δ^* is $\mathtt{proj}_{\Sigma,\Delta} : \Sigma^* \to \Delta^*$, defined by $\mathtt{proj}_{\Sigma,\Delta}(a) = a$ if $a \in \Delta$, $\mathtt{proj}_{\Sigma,\Delta}(a) = \lambda$ if $a \in (\Sigma \setminus \Delta) \cup \{\lambda\}$, and $\mathtt{proj}_{\Sigma,\Delta}(wa) = \mathtt{proj}_{\Sigma,\Delta}(w)\mathtt{proj}_{\Sigma,\Delta}(a)$ whenever $w \in \Sigma^*$ and $a \in \Sigma$. We omit the subscript Σ if it is clear from the context and thus write $\mathtt{proj}_\Delta(w)$ instead of $\mathtt{proj}_{\Sigma,\Delta}(w)$. The notation is extended to languages $L \subseteq \Sigma^*$ by $\mathtt{proj}_\Delta(L) = \{\mathtt{proj}_\Delta(w) \mid w \in L\}$. As an example, let $L = \{abbab, abcbdab, cbbbad\}$. Then $\mathtt{proj}_{\{a\}}(L) = \{a, aa\}$, $\mathtt{proj}_{\{a,b\}}(L) = \{abbab, bbba\}$, and $proj_{\{c,d\}}(L) = \{cd\}$.

Petri Nets. A Petri net is a triple $N = (P, T, F)$, where P is a finite set of *places*, T is a finite non-empty set of *transitions* such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs.

Let $N = (P, T, F)$ be a Petri net. If $N' = (P', T', F')$ is a Petri net such that $P \cup T$ and $P' \cup T'$ have no elements in common, then N and N' are *disjoint*. Let $x \in P \cup T$. Then ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$ are the *preset* and the *postset*, respectively, of x (in N). A *marking* μ of N is a function $\mu : P \to \mathbb{N}$. Let $t \in T$ and μ a marking of N. Then t is *enabled at* μ if $\mu(p) > 0$ for all $p \in {}^\bullet t$. If t is enabled at μ, it may *occur*, and thus lead to a new marking μ' of N, denoted $\mu \xrightarrow{t}_N \mu'$, with $\mu'(p) = \mu(p) - 1$ if $p \in {}^\bullet t \setminus t^\bullet$; $\mu'(p) = \mu(p) + 1$ if $p \in t^\bullet \setminus {}^\bullet t$; and $\mu'(p) = \mu(p)$ otherwise. We extend the notation $\mu \xrightarrow{t}_N \mu'$ to sequences $w \in T^*$ as follows[1]: $\mu \xrightarrow{\lambda}_N \mu$ for all μ; and $\mu \xrightarrow{wt}_N \mu'$ for markings μ, μ' of N, $w \in T^*$ and $t \in T$, whenever there is a marking μ'' such that $\mu \xrightarrow{w}_N \mu''$ and $\mu'' \xrightarrow{t}_N \mu'$. If $\mu \xrightarrow{w}_N \mu'$, for some $w \in T^*$, then w is a *firing sequence (in N) from* μ to μ' and μ' is said to be *reachable from* μ (in N). If N is clear from the context, we may omit the subscript N and write $\xrightarrow{w}$ rather than $\xrightarrow{w}_N$. We write $\mathcal{L}(N, \mu)$ for $\{w \in T^* \mid \mu \xrightarrow{w}_N \mu' \text{ for some marking } \mu'\}$. A place $p \in P$ is a *source*

[1] We thus view T as an alphabet.

place of N if ${}^\bullet p = \emptyset$. The marking μ of N such that, for all $p \in P$, $\mu(p) = 1$ if p is a source place and $\mu(p) = 0$ otherwise, is the *default initial marking* of N. If μ is the default initial marking of N, we also write $\mathcal{L}(N)$ to denote $\mathcal{L}(N, \mu)$ and refer to it as the behaviour or *language of* N. Note that $\mathcal{L}(N)$ is *prefix-closed*, i.e., $w \in \mathcal{L}(N)$ implies $v \in \mathcal{L}(N)$ for all prefixes v of w.

Enterprise Nets and Industry Nets. We recall the definitions of Enterprise and Industry nets from [21]. Enterprise nets (or E-nets, for short) are Petri nets equipped for asynchronous communication with other E-nets; they have transitions designated to receive input and transitions designated to produce output. An interaction between E-nets is realised by an occurrence of an output transition of one E-net and an occurrence of an input transition of another E-net that is connected to the output transition by a dedicated intermediate place and that has a matching *message type*.

Throughout this paper, we assume a fixed set $\mathcal{M}$ of message types.

Definition 1. *An Enterprise net is a tuple* $\mathcal{E} = (P, \langle T_{int}, T_{inp}, T_{out} \rangle, F, M)$ *such that* T_{int}, T_{inp}, *and* T_{out} *are mutually disjoint sets;* T_{int} *is the set of* internal transitions *of* $\mathcal{E}$, T_{inp} *its set of* input transitions, *and* T_{out} *its set of* output transitions*; furthermore, the* underlying Petri net of $\mathcal{E}$, $und(\mathcal{E}) = (P, T_{int} \cup T_{inp} \cup T_{out}, F)$, *is a Petri net with exactly one source place; finally* $M : T_{inp} \cup T_{out} \to \mathcal{M}$ *is the* communication function *of* $\mathcal{E}$. □

Given an Enterprise net $\mathcal{E}$, $\mathcal{L}(\mathcal{E}) = \mathcal{L}(und(\mathcal{E}))$ is *the language of* $\mathcal{E}$.

Composing E-nets yields an Industry-net (or I-net, for short) with multiple source places. The E-nets involved are pairwise disjoint, i.e., their underlying Petri nets are disjoint. When combining them into an I-net, output and input transitions are matched via their message types.

Definition 2. *Let* $n \geq 2$. *Let* $V = \{\mathcal{E}_i \mid i \in [n]\}$ *be a set of pairwise disjoint E-nets with* $\mathcal{E}_i = (P_i, \langle T_{i,int}, T_{i,inp}, T_{i,out} \rangle, F_i, M_i)$ *for each* $i \in [n]$.
A matching over V *is a bijection* $\varphi : \bigcup_{i \in [n]} T_{i,out} \to \bigcup_{j \in [n]} T_{j,inp}$ *such that whenever* $t \in T_{i,out}$ *and* $\varphi(t) \in T_{j,inp}$, *for some* i, j, *then* $i \neq j$ *and* $M_i(t) = M_j(\varphi(t))$. □

A set V of mutually disjoint E-nets is said to be *composable* if there exists a matching over V. To construct an I-net from a composable set V and a matching φ over V, matching output and input transitions of the E-nets in V are connected through (new) channel places using channel arcs.

Definition 3. *Let* $n \geq 2$. *Let* $V = \{\mathcal{E}_i : i \in [n]\}$ *be a composable set of E-nets with* $\mathcal{E}_i = (P_i, \langle T_{i,int}, T_{i,inp}, T_{i,out} \rangle, F_i, M_i)$ *and* $T_i = T_{i,int} \cup T_{i,inp} \cup T_{i,out}$ *for all* $i \in [n]$. *Let* φ *be a matching over* V.

Then $P(V, \varphi) = \{[t, \varphi(t)] \mid t \in T_{i,out}, i \in [n]\}$ *is the set of* channel places *of* V *and* φ, *and* $F(V, \varphi) = \{(t, [t, \varphi(t)]) \mid t \in T_{i,out}, i \in [n]\} \cup \{([t, \varphi(t)], \varphi(t)) \mid t \in T_{i,out}, i \in [n]\}$ *is the set of* channel arcs *of* V *and* φ. *The sets* $P(V, \varphi)$, $F(V, \varphi)$, *and* P_i, T_i, F_i, *where* $i \in [n]$, *are all pairwise disjoint.*

The Industry net over (V, φ) *is the Petri net* $\mathcal{I}(V, \varphi) = (P, T, F)$ *with* $P = \bigcup_{i\in[n]} P_i \cup P(V, \varphi)$, $T = \bigcup_{i\in[n]} T_i$, *and* $F = \bigcup_{i\in[n]} F_i \cup F(V, \varphi)$. □

Example 1. Consider two enterprises, an Investment Firm and an Exchange modelled by the E-nets $\mathcal{E}_1$ and $\mathcal{E}_2$ in Figs. 1(a) and 1(b), respectively. The Investment Firm sends order messages to the Exchange (modelled by output transition **so** with message type m_o). The Exchange, upon receiving a message (via input transition **ro** with message type m_o), subsequently sends an order confirmation message to the Investment Firm (output transition **sc** with message type m_c). This message can be received by the Investment Firm through input transition **rc** with message type m_c. Then $\mathcal{I}(V, \varphi)$, the I-net in Fig. 1(c), with $V = \{\mathcal{E}_1, \mathcal{E}_2\}$ and φ defined by $\varphi(\mathbf{so}) = \mathbf{ro}$ and $\varphi(\mathbf{sc}) = \mathbf{rc}$, models the collaboration between the Investment Firm and the Exchange. Note that this V allows for only one matching. □

Related models. E-nets can be considered as a generalisation of *workflow nets*. Workflow nets (see [7]) are Petri nets with a single source place. They, moreover, have a single sink place (i.e., a place p with $p^\bullet = \emptyset$) and a default final marking that assigns a single token to the sink place. Additional requirements are that all places and transitions are on a path from source to sink and that the final marking is an always reachable marking. These (so-called soundness) criteria are not a priori imposed on E-nets. All possible firing sequences reflect possible behaviour and the language of an E-net is always prefix-closed. E-nets, moreover, explicitly capture the potential interaction with other enterprises by a bilateral, asynchronous exchange of messages of a specified type (for applications, see eg., [13,18,24,28]). With the term "industry" used to loosely refer to a sector of enterprises, Industry nets model collaborating Enterprise nets. Overall, the ideas underlying the concepts of E-nets and I-nets belong to a well-established branch of research concerned with composing systems modelled as Petri nets, into larger correctly functioning (concurrent) systems [14,15,26,27]. An overview of how this line of research is concerned with combining compatible business processes (or services) in Service Oriented Architectures' can be found in [3].

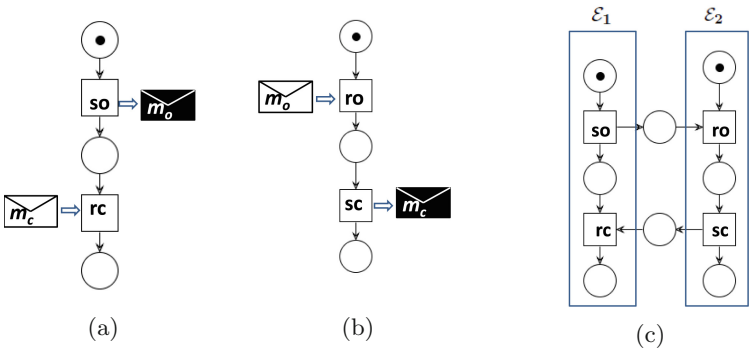


Fig. 1. (a) E-net $\mathcal{E}_1$. (b) E-net $\mathcal{E}_2$. (c) The I-net

Open Petri nets, introduced in [9], are proposed in [16] as a formalization of workflows with a capability to interact with other workflows through *open places*. The composition of open Petri nets is characterised as a pushout in the corresponding category, suitable to model both interaction through open places and synchronization of transitions. Decomposing Petri nets into *functional Petri nets* to speed up analysis is described in [30,31]. Compositions of *open workflow nets*, a model similar to functional Petri nets, are considered in [1,23,29]. Functional Petri nets have disjoint sets of input places and output places and composition is based on identifying corresponding elements. In [15], I/O-Petri nets which are similar to I-nets, are considered. These nets have transitions as interface elements that have input and output labels and communicate via places. The advantage of this approach, as argued in [15], is that it leads to a better separation of concerns: eg., the designer of a local system does not have to consider how it will be used; this is a concern for the designer of the global system. Composition of Petri nets by sharing places in [9,16,23] or by connecting transitions through channel places as in [15] and our approach, models *asynchronous communication* between the component nets.

3 E-net Languages and I-net Languages

Let the composable system V, specified as in Definition 3, and the matching φ over V be fixed for this section and Sects. 4 and 5.

Clearly, the construction of $\mathcal{I}(V,\varphi)$ does not affect the internal structure of the E-nets in V and the set of source places of $\mathcal{I}(V,\varphi)$ consists of all source places of the $\mathcal{E}_i$. Moreover, removing channel places from $\mathcal{I}(V,\varphi)$ does not restrict the behaviour of its E-nets. In other words, if $\mu \xrightarrow{w} \mu'$ in $\mathcal{I}(V,\varphi)$, then $\mu|_{P_i} \xrightarrow{proj_{T_i}(w)} \mu'|_{P_i}$ in $\mathcal{E}_i$. Consequently, the firing sequences in $\mathcal{L}(\mathcal{I}(V,\varphi))$ are combinations of firing sequences of the Enterprise nets. Actually, as formulated in the next lemma, this statement can be strengthened to include all prefixes of firing sequences in $\mathcal{L}(\mathcal{I}(V,\varphi))$, because $\mathtt{proj}_{T_i}(v) \in \mathtt{pref}(\mathtt{proj}_{T_i}(w))$, for all $v, w \in T^*$ such that $v \in \mathtt{pref}(w)$, and $\mathcal{L}(\mathcal{I}(V,\varphi))$ and $\mathcal{L}(\mathcal{E}_i)$, $i \in [n]$, are prefix-closed.

Lemma 1. *If $w \in \mathcal{L}(\mathcal{I}(V,\varphi))$, then $\mathtt{proj}_{T_i}(v) \in \mathcal{L}(\mathcal{E}_i)$ for all $v \in \mathtt{pref}(w)$ and all $i \in [n]$.* □

However, the composition of E-nets into an I-net adds channel places to the presets of input transitions. The property defined next describes how the number of occurrences of input transitions depends on the number of occurrences of corresponding output transitions.

Definition 4. *Let $w \in T^*$. Then w has the* prefix property with respect to φ *if $\#_a(v) \geq \#_{\varphi(a)}(v)$ for all $v \in \mathtt{pref}(w)$ and all $a \in T_{out}$.*
A language $L \subseteq T^*$ has the prefix property with respect to φ if all $w \in L$ have this property. □

In other words, w has the prefix property with respect to φ if $\#_b(v) \leq \#_{\varphi^{-1}(b)}(v)$ for all prefixes v of w and all $b \in T_{inp}$. Clearly, if w has the prefix property with respect to φ, then all its prefixes have this property as well.

Henceforth, we will omit the reference to φ as it is fixed.

Since channel places are not source places and consequently not marked by the default initial marking, it follows that the firing sequences of $\mathcal{I}(V, \varphi)$ have the prefix property.

Lemma 2. *If $w \in \mathcal{L}(\mathcal{I}(V, \varphi))$, then w has the prefix property.* □

Conversely, any sequence $w \in T^*$ that can be (locally) executed by all E-nets and that satisfies the prefix property, belongs to $\mathcal{L}(\mathcal{I}(V, \varphi))$.

Lemma 3. *Let $w \in T^*$ be such that $\mathtt{proj}_{T_i}(w) \in \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. If w has the prefix property, then $w \in \mathcal{L}(\mathcal{I}(V, \varphi))$.*

Proof. If $w = \lambda$ then $w \in \mathcal{L}(\mathcal{I}(V, \varphi))$. Assume now that $w = xa$ with $|x| \geq 0$ and $a \in T_k$ for some $k \in [n]$, and that w has the prefix property. Then x being a prefix of w, has the prefix property as well. Moreover, by Lemma 1, $\mathtt{proj}_{T_i}(x) \in \mathcal{L}(\mathcal{E}_i)$, for all $i \in [n]$. Hence, we may assume by an inductive argument that $x \in \mathcal{L}(\mathcal{I}(V, \varphi))$. Consequently, there exist a marking μ' of $\mathcal{I}(V, \varphi)$ such that $\mu \xrightarrow{x} \mu'$ in $\mathcal{I}(V, \varphi)$, where μ is the default initial marking of $\mathcal{I}(V, \varphi)$. This implies that $\mu|_{P_k} \xrightarrow{proj_{T_k}(x)} \mu'|_{P_k}$ in $\mathcal{E}_k$ with $\mu|_{P_k}$ the default initial marking of $\mathcal{E}_k$. By assumption $\mathtt{proj}_{T_k}(w) = \mathtt{proj}_{T_k}(xa) = \mathtt{proj}_{T_k}(x)a \in \mathcal{L}(\mathcal{E}_k)$ and so a is enabled at $\mu'|_{P_k}$ in $\mathcal{E}_k$. We distinguish two cases:

(i) If $a \notin T_{inp}$, it directly follows that a is enabled at μ' thus $w \in \mathcal{L}(\mathcal{I}(V, \varphi))$.
(ii) If $a \in T_{inp}$, then it has a corresponding output transition $t = \varphi^{-1}(a)$ in some $\mathcal{E}_i$ and a channel place $[t, a] \notin P_k$ such that $(t, [t, a]), ([t, a], a) \in F$. Note that a is otherwise only connnected to places in P_k. Since w has the prefix property, $\#_a(w) \leq \#_t(w)$. So $\#_a(x) < \#_t(x)$. Hence $\mu'([t, a]) > 0$, and a is enabled at μ'. Consequently, also in this case, $w \in \mathcal{L}(\mathcal{I}(V, \varphi))$. □

Example 2. (Ex. 1 ctd.) We have $T_1 = \{\mathbf{so}, \mathbf{rc}\}$, $T_2 = \{\mathbf{ro}, \mathbf{sc}\}$ (with T_1 and T_2 the transitions of $\mathcal{E}_1$ and $\mathcal{E}_2$ from Example 1 respectively), and $T = \{\mathbf{so}, \mathbf{rc}, \mathbf{ro}, \mathbf{sc}\}$. Consider $w = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{rc} \rangle$.[2] Now $\mathtt{proj}_{T_1}(w) = \langle \mathbf{so}, \mathbf{rc} \rangle \in \mathcal{L}(\mathcal{E}_1)$ and $\mathtt{proj}_{T_2}(w) = \langle \mathbf{ro}, \mathbf{sc} \rangle \in \mathcal{L}(\mathcal{E}_1)$. Clearly, w has the prefix property. Thus $w \in \mathcal{L}(\mathcal{I}(V, \varphi))$ by Lemma 3. For $w' = \langle \mathbf{so}, \mathbf{ro}, \mathbf{rc}, \mathbf{sc} \rangle$, we have $\mathtt{proj}_{T_1}(w') = \mathtt{proj}_{T_1}(w)$ and $\mathtt{proj}_{T_2}(w') = \mathtt{proj}_{T_2}(w)$. However, w' does not have the prefix property: in $u = \langle \mathbf{so}, \mathbf{ro}, \mathbf{rc} \rangle$, a prefix of w', we have $\#_{\mathbf{rc}}(u) = 1$, but $\#_{\mathbf{sc}}(u) = 0$. Indeed, $w' \notin \mathcal{L}(\mathcal{I}(V, \varphi))$. □

The next theorem is an immediate consequence of Lemmas 1, 2, and 3.

[2] The elements of the alphabet T are symbols consisting of two letters. We denote in this and similar examples, any sequence $a_1 \cdots a_n$ with $a_i \in T$ for each $i \in [n]$ by $\langle a_1, \cdots, a_n \rangle$.

Theorem 1. *Let $L \subseteq T^*$. Then $L \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$ if and only if L has the prefix property and $\mathtt{proj}_{T_i}(L) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$.* □

In case there exists a language L that has the prefix property and is such that $\mathtt{proj}_{T_i}(L) = \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$, the behaviour of the I-net encompasses the full unrestricted behaviour of each E-net. This is captured in Theorem 2.

Theorem 2. *Let $L \subseteq T^*$ be a language with the prefix property and let $\mathtt{proj}_{T_i}(L) = \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. Then $\mathtt{proj}_{T_i}(\mathcal{L}(\mathcal{I}(V,\varphi))) = \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$.*

Proof. $L \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$ follows from Theorem 1. Hence, for all $i \in [n]$, $\mathtt{proj}_{T_i}(L) \subseteq \mathtt{proj}_{T_i}(\mathcal{L}(\mathcal{I}(V,\varphi)))$. By Lemma 1, $\mathtt{proj}_{T_i}(\mathcal{L}(\mathcal{I}(V,\varphi))) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. By assumption $\mathtt{proj}_{T_i}(L) = \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. Thus $\mathcal{L}(\mathcal{E}_i) \subseteq \mathtt{proj}_{T_i}(\mathcal{L}(\mathcal{I}(V,\varphi))) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$ and the statement follows. □

It should however be noted, that the conditions on L in Theorem 2 do not imply that $L = \mathcal{L}(\mathcal{I}(V,\varphi))$. By Theorem 1, $L \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$, and as illustrated in the following example, this inclusion may be strict even in case L satisfies the conditions from Theorem 2.

Example 3. Consider a collaboration between three enterprises, an Investment Firm, an Exchange and a *Central Securities Depository* (modelled by E-nets $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ respectively, with transitions $T_1 = \{\mathbf{so}, \mathbf{rc}, \mathbf{rs}\}$, $T_2 = \{\mathbf{ro}, \mathbf{sc}, \mathbf{si}\}$ and $T_3 = \{\mathbf{ri}, \mathbf{ss}\}$) represented by the I-net in Fig. 2. The interaction between the Investment Firm and the Exchange is the same as in Example 2. After sending a confirmation (output transition **sc**) to the Investment Firm, the Exchange can send (output transition **si**) a settlement instruction to the Central Securities Depository to transfer ordered securities to the Investment Firm. The Central Securities Depository - after receiving (input transition **ri** matching **si**) the settlement instruction - sends (output transition **ss**) a confirmation of settlement to the Investment Firm (that receives the message via input transition **rs**). Let $w = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{rc}, \mathbf{si}, \mathbf{ri}, \mathbf{ss}, \mathbf{rs} \rangle$ and let $L = \mathtt{pref}(\{w\})$.

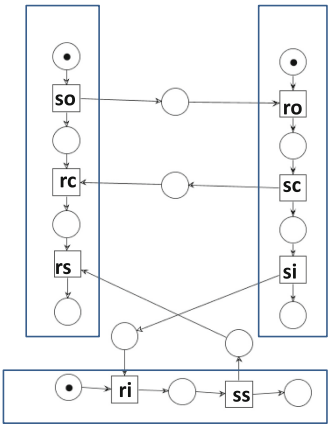


Fig. 2. The extended I-net

Clearly $proj_{T_i}(L) = \mathcal{L}(\mathcal{E}_i)$ for each $i \in [3]$. Also, L has the prefix property (w.r.t. φ displayed in Fig. 2). Hence L satisfies the conditions in Theorem 2, and, indeed, $proj_{T_i}(\mathcal{L}(\mathcal{I}(\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3\}, \varphi))) = proj_{T_i}(L)$ for each $i \in [3]$. Now, consider the sequence $v = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{si}, \mathbf{rc}, \mathbf{ri}, \mathbf{ss}, \mathbf{rs} \rangle$, obtained by exchanging the occurrences of **rc** and **si** in w. It is easy to see that v is also in the language of the I-net in Fig. 2. Hence this I-net exhibits behaviour not included in L. □

4 Structuring Words with the Prefix Property

In this section, we identify a property of I-net languages that explains why the conditions of Theorem 2 imposed on L are not sufficient to guarantee that $L = \mathcal{L}(\mathcal{I}(V, \varphi))$. First, we demonstrate how the occurrences in words with the prefix property, can be seen as partially ordered sets. The occurrences **rc** and **si** in the word w from Example 3, can be exchanged (to obtain v) because they are unrelated in this partial order (i.e., they are from different components and not connected by an input/output relation with each other, see Example 6 below). We will show that any word obtained from a word in $\mathcal{L}(\mathcal{I}(V, \varphi))$, by exchanging unrelated occurrences, is included in $\mathcal{L}(\mathcal{I}(V, \varphi))$. To define the partial order, we introduce the notion of an assignment function. Whereas the prefix property is based on a simple comparison of numbers of occurrences, assignment functions, as defined next, relate each occurrence of an input transition to a corresponding occurrence of an output transition.

Definition 5. *An* assignment function *with respect to φ for a word $w \in T^*$ is an injective function $\theta : (\mathtt{occ}(w) \cap (T_{inp} \times \mathbb{N})) \to (\mathtt{occ}(w) \cap (T_{out} \times \mathbb{N}))$ such that for every occurrence $(b, j) \in \mathtt{occ}(w)$ with $b \in T_{inp}$, $\theta(b, j) = (a, i)$ implies that $a = \varphi^{-1}(b)$ and i is such that $\mathtt{pos}_w(a, i) < \mathtt{pos}_w(b, j)$.* □

Again, since φ is fixed, we will omit the reference to φ.

Example 4. Assume $T = \{a, b\}$ and $\varphi(a) = b$. Let $w = aabb$. Then w has two assignment functions: θ defined by $\theta(b, 1) = (a, 1)$ and $\theta(b, 2) = (a, 2)$; and θ' defined by $\theta'(b, 1) = (a, 2)$ and $\theta'(b, 2) = (a, 1)$. □

The notion of an assignment function is closely related to the prefix property, as captured in Theorem 3.

Theorem 3. *Let $w \in T^*$. Then w has the prefix property if and only if there exists an assignment function for w.*

Proof. Let θ be an assignment function for w. Since θ is injective, we have that for every input transition b and every $1 \leq j \leq \#_b(w)$, the j-th occurrence (b, j) of b in w is preceded by at least j occurrences of output transition $\varphi^{-1}(b)$. In other words $\#_b(u) \leq \#_{\varphi^{-1}(b)}(u)$ for every prefix u of w as desired.

Next, assume that w has the prefix property and consider the function θ with domain $\mathtt{occ}(w) \cap (T_{inp} \times \mathbb{N})$ defined by $\theta(b, j) = (\varphi^{-1}(b), j)$ for all $(b, j) \in occ(w) \cap (T_{inp} \times \mathbb{N})$. As we argue next, θ is an assignment function for w.

Let $(b, j) \in occ(w)$ with $b \in T_{inp}$. Hence there exists a prefix xb of w such that $\#_b(xb) = j$. Since w has the prefix property, $\#_b(xb) \leq \#_{\varphi^{-1}(b)}(xb)$, thus $\#_{\varphi^{-1}(b)}(xb) \geq j$ and therefore $\mathtt{pos}_w(\varphi^{-1}(b), j) < \mathtt{pos}_w(b, j)$ as required. □

Theorem 3 implies that every word with the prefix property has an assignment function. As Example 4 shows, in general, each such word can have more than one assignment function. We will now show how each assignment function θ for a word w determines a partial order on $\mathtt{occ}(w)$. First the relation $\leq_{w,\theta}$ is defined.

Definition 6. *Let $w \in T^*$ and let θ be an assignment function for w. Let $(a, i), (b, j) \in \mathtt{occ}(w)$, with $a \in T_k$ and $b \in T_l$ for some $k, l \in [n]$.*
Then $(a, i) \leq_{w,\theta} (b, j)$ if

(1) $k = l$ and $\mathtt{pos}_w(a, i) \leq \mathtt{pos}_w(b, j)$ or
(2) $k \neq l$ and $b \in T_{inp}$ and $\theta(b, j) = (a, i)$. □

By condition (1), $\leq_{w,\theta}$ respects the ordering of occurrences of transitions in w that originate from the same E-net; by (2), it fixes the order between an occurrence of an input transition and its assigned output transition occurrence. As usual, $\leq^+_{w,\theta}$ denotes the transitive closure of $\leq_{w,\theta}$. The following lemma states that $\leq^+_{w,\theta}$ moreover respects the relative position of all occurrences in w.

Lemma 4. *Let $w \in T^*$ and let θ be an assignment function for w. Then $x \leq^+_{w,\theta} y$ implies $\mathtt{pos}_w(x) \leq \mathtt{pos}_w(y)$, for all $x, y \in \mathtt{occ}(w)$.*

Proof. Let $x, y \in \mathtt{occ}(w)$ and assume $x \leq^+_{w,\theta} y$ holds. So, there exist occurrences $o_1, \ldots, o_m$ in $\mathtt{occ}(w)$, with $m \leq |w|$, such that $o_1 = x$, $o_m = y$ and $o_i \leq_{w,\theta} o_{i+1}$, for all $i \in [m-1]$. Hence, using Definition 6 and Definition 5, it follows that $\mathtt{pos}_w(o_i) \leq \mathtt{pos}_w(o_{i+1})$, for all $i \in [m-1]$. Since $\leq$ is a total order on the positions of w, we have $\mathtt{pos}_w(x) \leq \mathtt{pos}_w(y)$. □

Using this observation, we can now prove that indeed $\leq^+_{w,\theta}$ is a partial order on the occurrences of w.

Lemma 5. *Let $w \in T^*$ and let θ be an assignment function for w. Then $\leq^+_{w,\theta}$ is a partial order on $\mathtt{occ}(w)$.*

Proof. From condition (1) in Definition 6 it follows that $\leq^+_{w,\theta}$ is reflexive, while transitivity is immediate. Let $x \leq^+_{w,\theta} y$ and $y \leq^+_{w,\theta} x$ for some $x, y \in \mathtt{occ}(w)$. By Lemma 4, $\mathtt{pos}_w(x) = \mathtt{pos}_w(y)$ and so $x = y$. Hence $\leq^+_{w,\theta}$ is anti-symmetric. □

By Lemmas 4 and 5, given an assignment function θ for w, the partial order $\leq^+_{w,\theta}$ is a subset of the total order induced by $\mathtt{pos}_w$. The linearisations of $\leq^+_{w,\theta}$, where w is a word and θ an assignment function for w, are those words that can be obtained from w by (repeatedly) interchanging the positions of occurrences not related by $\leq^+_{w,\theta}$.

Definition 7. *Let $w \in T^*$ and let θ be an assignment function for w. Then* $\mathtt{lin}_\theta(w) = \{v \in T^* \mid \mathtt{occ}(v) = \mathtt{occ}(w)$ *and for all* $x, y \in \mathtt{occ}(v), x \leq^+_{w,\theta} y$ *implies* $\mathtt{pos}_v(x) \leq \mathtt{pos}_v(y)\}$ *is the set of* θ-linearisations of w. □

The next example shows that if v is a θ-linearisation of w than w is also a θ-linearisation of v.

Example 5. (Ex. 4 ctd.) Let $v = abab$. Then $\mathtt{occ}(v) = \mathtt{occ}(w)$ and $\mathtt{lin}_\theta(w) = \{v, w\}$. Clearly, θ is an assignment function for v and $\mathtt{lin}_\theta(v) = \mathtt{lin}_\theta(w)$. Note that $\mathtt{lin}_{\theta'}(w) = \{w\}$ and θ' is not an assignment function for v. □

We now turn to the proof of the main result of this section by which for every firing sequence w of an I-net and each of the assignment functions θ of w, also all its θ-linearisations are firing sequences of the I-net. We first list some basic properties of linearisations.

Lemma 6. *Let w and θ be as in Definition 7 and let $v \in \mathtt{lin}_\theta(w)$. Then the following statements hold:*

(1) $w \in \mathtt{lin}_\theta(w)$*;*
(2) $\mathtt{proj}_{T_i}(v) = \mathtt{proj}_{T_i}(w)$ *for all* $i \in [n]$*;*
(3) θ *is an assignment function for* v *and* $\leq_{v,\theta} = \leq_{w,\theta}$*;*
(4) $\mathtt{lin}_\theta(v) = \mathtt{lin}_\theta(w)$*;*
(5) v *has the prefix property.*

Proof. (1) follows directly from Definition 7 and Lemma 4.

(2) is proved by contradiction. Assume (2) to be false and let $k \in [n]$ be such that $\mathtt{proj}_{T_k}(v) \neq \mathtt{proj}_{T_k}(w)$. Since $\mathtt{occ}(v) = \mathtt{occ}(w)$ by Definition 7, this implies that there are $x, y, z \in T^*$ and $a, b \in T_k$, such that $\mathtt{proj}_{T_k}(v) = xay$ and $\mathtt{proj}_{T_k}(w) = xbz$ with $a \neq b$. Moreover, $\mathtt{occ}(v) = \mathtt{occ}(w)$ implies that $\mathtt{occ}(\mathtt{proj}_{T_k}(v)) = \mathtt{occ}(\mathtt{proj}_{T_k}(w))$, i.e., $\mathtt{occ}(xay) = \mathtt{occ}(xbz)$. Let now $\#_a(xa) = i$ and $\#_b(xb) = j$. Then $\mathtt{pos}_{xay}((b, j)) > |xa|$ and $\mathtt{pos}_{xbz}((a, i)) > |xb|$. Thus $\mathtt{pos}_v(a, i) < \mathtt{pos}_v(b, j)$ and $\mathtt{pos}_w(b, j) < \mathtt{pos}_w(a, i)$. The latter inequality combined with condition (1) from Definition 6 shows that $(b, j) \leq^+_{w,\theta} (a, i)$. Hence the first inequality implies that $v \notin \mathtt{lin}_\theta(w)$ by Definition 7, a contradiction. Consequently, (2) must be true.

(3) Let $(b, j) \in occ(v) = occ(w)$ with $b \in T_{inp}$ and let $\theta(b, j) = (a, i)$. Since θ is an assignment function for w, it follows from condition (2) from Definition 6 that $(a, i) \leq_{w,\theta} (b, j)$. With $v \in lin_\theta(w)$ this implies that $\mathtt{pos}_v(a, i) \leq \mathtt{pos}_v(b, j)$. Hence θ is an assignment function for v.
Next we investigate $\leq_{v,\theta}$ (which is defined because θ is an assignment function for v). Let $x, y \in \mathtt{occ}(v) = \mathtt{occ}(w)$ with $x = (a, i)$ and $y = (b, j)$ where $a \in T_k$ and $b \in T_l$ with $k, l \in [n]$.
Firstly, assume $x \leq_{w,\theta} y$. Then $\mathtt{pos}_v(x) \leq \mathtt{pos}_v(y)$, because $v \in lin_\theta(w)$. In case $k = l$, $x \leq_{v,\theta} y$ by condition (1) of Definition 6. If $k \neq l$, then $x \leq_{w,\theta} y$ must be a consequence of $\theta(b, j) = (a, i)$ (condition (2) of Definition 6) and hence also $x \leq_{v,\theta} y$.
Secondly, assume $x \leq_{v,\theta} y$. In case $k = l$, $\mathtt{pos}_v(x) \leq \mathtt{pos}_v(y)$ by condition

(1) of Definition 6. We claim that also $\mathtt{pos}_w(x) \leq \mathtt{pos}_w(y)$: if not, then $\mathtt{pos}_w(y) < \mathtt{pos}_w(x)$ which would imply $x \neq y$ and $y \leq_{w,\theta} x$ by condition (1) of Definition 6 and hence $\mathtt{pos}_v(y) < \mathtt{pos}_v(x)$, because $v \in lin_\theta(w)$, a contradiction. From $\mathtt{pos}_w(x) \leq \mathtt{pos}_w(y)$ and condition (1) of Definition 6, it follows that $x \leq_{w,\theta} y$. Finally, let $k \neq l$. Then $x = \theta(y)$ follows from $x \leq_{v,\theta} y$ and condition (2) of Definition 6. This implies that also $x \leq_{w,\theta} y$.

(4) Follows from the definitions of $\mathtt{lin}_\theta(w)$ and $\mathtt{lin}_\theta(v)$ since $\mathtt{occ}(v) = \mathtt{occ}(w)$ and $\leq_{v,\theta} = \leq_{w,\theta}$ by (3).

(5) follows from (3) and Theorem 3. □

Combining Lemma 6 with Lemma 3 shows that whenever a word with the prefix property can be locally executed by all component E-nets, then, for all its assignment functions θ, each of its θ-linearisations can be executed globally by the I-net.

Theorem 4. *Let $w \in T^*$ be such that $\mathtt{proj}_{T_i}(w) \in \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. If θ is an assignment function for w, then $\mathtt{lin}_\theta(w) \subseteq \mathcal{L}(\mathcal{I}(V, \varphi))$.*

Proof. Assume θ is an assignment function for w. Let $v \in \mathtt{lin}_\theta(w)$. From Lemma 6(2) it follows that $\mathtt{proj}_{T_i}(v) = \mathtt{proj}_{T_i}(w)$ for all $i \in [n]$ and from Lemma 6(5) that v has the prefix property. Using Lemma 3, we can then conclude that $v \in \mathcal{L}(\mathcal{I}(V, \varphi))$. □

Example 6. (Ex. 3 ctd.) Consider the word w from Example 3. Define θ by $\theta(\mathbf{ro}, 1) = (\mathbf{so}, 1)$, $\theta(\mathbf{rc}, 1) = (\mathbf{sc}, 1)$, $\theta(\mathbf{ri}, 1) = (\mathbf{si}, 1)$, and $\theta(\mathbf{rs}, 1) = (\mathbf{ss}, 1)$. This θ is an (the only) assignment function for w (w.r.t. φ). Then v, obtained from w by exchanging $(\mathbf{si}, 1)$ and $(\mathbf{rc}, 1)$. These occurrences are not ordered by $\leq^+_{w,\theta}$. Hence $v \in \mathtt{lin}_\theta(w)$ and v is, like w, in the language of the I-net. □

5 Assignment Functions of Type FIFO

By Theorem 4, every assignment function θ for a word w (with the prefix property) that can be locally executed by all component E-nets, determines a set of words $\mathtt{lin}_\theta(w)$ that can be executed by the I-net. In this section, we demonstrate that considering one particular type of assignment function is sufficient to describe all possible linearisations.

Intuitively, in terms of the I-net, one could say that each assignment function describes for each occurrence of an input transition, which token to take from its channel place (namely the token deposited by the assigned occurrence of the output transition). Based on this point of view, the following two types of assignment functions represent natural policies to deal with the messages (tokens) in the channel places.

Definition 8. *Let $w \in T^*$ and let θ be an assignment function for w. Then*

(1) θ is of type FIFO *with respect to φ if $\theta(b,j) = (\varphi^{-1}(b), j)$ for every $(b,j) \in \mathtt{occ}(w)$ such that $b \in T_{inp}$;*
(2) θ is of type LIFO *with respect to φ if for every $(b,j) \in \mathtt{occ}(w)$ such that $b \in T_{inp}$ and $\theta(b,j) = (a,i)$ where $a = \varphi^{-1}(b)$, there exist $x, y, z \in T^*$ such that $w = xaybz$ with $i = \#_a(xa)$, $j = \#_b(xayb)$, and $\#_a(y) = \#_b(y)$.* □

Again, since φ is fixed, we will omit the reference to φ.

When scrutinising the proof of Theorem 3, one sees that the function θ constructed there is of type FIFO. Hence we have the following corollary.

Corollary 1. *Every word with the prefix property has an assignment function of type FIFO.* □

Actually, the argument in the proof of Theorem 3 could also have been based on the construction of an assignment function of type LIFO. In other words, every word with the prefix property has an assignment function of type LIFO. Finally, it is easily seen that each w with the prefix property has exactly one assignment function of type FIFO; also its assignment function of type LIFO is unique (the proof of which we leave to the reader).

Example 7. (Ex. 4 and Ex. 5 ctd.) Assignment function θ of w is of type FIFO and θ' is of type LIFO. For v, however, θ is an assignment function both of type FIFO and of type LIFO. □

We now investigate the assignment functions of type FIFO. It will be shown that the assignment function of type FIFO of a word w defines the least restrictive ordering on $\mathtt{occ}(w)$ in the sense that its set of linearisations is maximal (w.r.t. set inclusion) among all sets of θ-linearisations of w.

In the sequel, $\theta_w^{\mathtt{fifo}}$ denotes the assignment function of type FIFO of any word $w \in T^*$ with the prefix property. In addition, if $L \subseteq T^*$ is a language with the prefix property, then $\mathtt{lin}_{\mathtt{FIFO}}(L) = \bigcup\{\mathtt{lin}_{\theta_w^{\mathtt{fifo}}}(w) \mid w \in L\}$ consists of all *FIFO-linearisations of L*.

Lemma 7. *Let $v, w \in T^*$ be such that v and w have the prefix property and $\mathtt{occ}(v) = \mathtt{occ}(w)$. Then $\theta_v^{\mathtt{fifo}} = \theta_w^{\mathtt{fifo}}$.*

Proof. By Corollary 1, $\theta_v^{\mathtt{fifo}}$ and $\theta_w^{\mathtt{fifo}}$ exist. Let $(b,j) \in occ(v)$, with $b \in T_{inp}$ for some $j \geq 1$. Then $\theta_v^{\mathtt{fifo}}(b,j) = (\varphi^{-1}(b), j) = \theta_w^{\mathtt{fifo}}$ as required. □

A similar result does not hold for assignment functions of type LIFO (cf. Example 7).

The set of linearisations determined by an assignment function of type FIFO can be characterised as follows.

Lemma 8. *Let $w \in T^*$ have the prefix property. Then $\mathtt{lin}_{\theta_w^{\mathtt{fifo}}}(w) = \{v \in T^* \mid v$ has the prefix property and $\mathtt{proj}_{T_i}(v) = \mathtt{proj}_{T_i}(w)$ for all $i \in [n]\}$.*

Proof. The inclusion from left to right follows immediately from Lemma 6(2) and Lemma 6(5).

To prove the converse inclusion, consider a word $v \in T^*$ such that v has the prefix property and $\mathtt{proj}_{T_i}(v) = \mathtt{proj}_{T_i}(w)$ for all $i \in [n]$. Hence $\mathtt{occ}(v) = \mathtt{occ}(w)$. Let now $(a,i),(b,j) \in \mathtt{occ}(v)$ be such that $(a,i) \leq^+_{w,\theta^{\mathtt{fifo}}_w} (b,j)$. We prove that $\mathtt{pos}_v(a,i) \leq \mathtt{pos}_v(b,j)$ from which it then follows that $v \in \mathtt{lin}_{\theta^{\mathtt{fifo}}_w}(w)$ by Definition 7.

Since $(a,i) \leq^+_{w,\theta^{\mathtt{fifo}}_w} (b,j)$, we have $\mathtt{pos}_w(a,i) \leq \mathtt{pos}_w(b,j)$ by Lemma 4. If $a,b \in T_k$ for some $k \in [n]$, then also $\mathtt{pos}_v(a,i) \leq \mathtt{pos}_v(b,j)$ because $\mathtt{proj}_{T_k}(v) = \mathtt{proj}_{T_k}(w)$.

If $a \in T_k$ and $b \in T_l$, for some $k \neq l$, we first consider the case that $(a,i) \leq_{w,\theta^{\mathtt{fifo}}_w} (b,j)$. This implies that $\theta^{\mathtt{fifo}}_w(b,j) = (a,i)$ and hence $i = j$. Since v has the prefix property, $\mathtt{pos}_v(a,j) \leq \mathtt{pos}_v(b,j)$. Otherwise there exist occurrences $o_1,\ldots,o_m$ in $\mathtt{occ}(w)$, with $m \leq |w|$, such that $o_1 = (a,i)$, $o_m = (b,j)$, and $o_i \leq_{w,\theta} o_{i+1}$, for all $i \in [m-1]$. With a reasoning similar to the above, we obtain $\mathtt{pos}_v(o_i) \leq \mathtt{pos}_v(o_{i+1})$, for all $i \in [m-1]$ and so $\mathtt{pos}_v(a,i) \leq \mathtt{pos}_v(b,j)$. □

Example 8. (Ex 2. Ctd.) The Investment Firm and Exchange from Example 2 are now extended with the possibility to repeatedly send and receive orders and their sets of transitions extended to $T_1' = \{\mathbf{so}, \mathbf{rc}, \mathbf{ii1}, \mathbf{ii2}\}$ and $T_2' = \{\mathbf{ro}, \mathbf{sc}, \mathbf{ei1}, \mathbf{ei2}\}$. The composition of the thus extended E-nets leads to the I-net in Fig. 3.

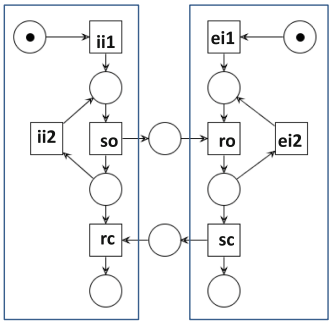


Fig. 3. I-net with option to repeat orders

Consider words $v = \langle \mathbf{ei1}, \mathbf{ii1}, \mathbf{so}, \mathbf{ii2}, \mathbf{ro}, \mathbf{ei2}, \mathbf{so}, \mathbf{ii2}, \mathbf{ro}, \mathbf{ei2}, \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{rc} \rangle$ and $w = \langle \mathbf{ei1}, \mathbf{ii1}, \mathbf{so}, \mathbf{ii2}, \mathbf{so}, \mathbf{ii2}, \mathbf{so}, \mathbf{ro}, \mathbf{ei2}, \mathbf{ro}, \mathbf{ei2}, \mathbf{ro}, \mathbf{sc}, \mathbf{rc} \rangle$. Both v and w have the prefix property and $\mathtt{proj}_{T_i'}(v) = \mathtt{proj}_{T_i'}(w)$ for $i \in \{1,2\}$. Let θ be the assignment function for w of type FIFO. From Lemma 8, we know that $v \in \mathtt{lin}_\theta(w)$. The assignment function θ' of w of type LIFO however has $\theta'(\mathbf{ro}, 1) = (\mathbf{so}, \mathbf{3})$ and so $v \in \mathtt{lin}_{\theta'}(w)$ does not hold. □

For a language L that has the prefix property, we denote by $\mathtt{lin}(L)$ the language consisting of all words that can be obtained as a θ-linearisation of

any word $w \in L$ for whatever assignment function θ for w. Thus $\mathtt{lin}(L) = \bigcup\{\mathtt{lin}_\theta(w) \mid w \in L \text{ and } \theta \text{ an assignment function for } w\}$.

Lemma 8 shows that indeed and as announced, the assignment functions of type FIFO are the most "generous" in the sense that they determine a maximal (w.r.t. set inclusion) set of linearisations. This observation is formalised in the following statement which is an extension of Theorem 4.

Theorem 5. *Let $L \subseteq T^*$ be a language with the prefix property such that $\mathtt{proj}_{T_i}(L) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. Then $L \subseteq \mathtt{lin}(L) = \mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$.*

Proof. The two inclusions are immediate: $\mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$ follows from Theorem 4 and $L \subseteq \mathtt{lin}(L)$ from Lemma 6(1). By definition $\mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathtt{lin}(L)$. So we only have to prove that $\mathtt{lin}(L) \subseteq \mathtt{lin}_{\mathtt{FIFO}}(L)$.

Let $v \in \mathtt{lin}(L)$. Hence there exists a $w \in L$ and an assignment function θ for w such that $v \in \mathtt{lin}_\theta(w)$. From Lemma 6(2) and (5), we know that $\mathtt{proj}_{T_i}(v) = \mathtt{proj}_{T_i}(w)$ and that v has the prefix property. Hence $v \in \mathtt{lin}_{\theta^{\mathtt{fifo}}_w}(w)$ by Lemma 8 (note that $\theta^{\mathtt{fifo}}_w$ exists by Corollary 1). Consequently, $v \in \mathtt{lin}_{\mathtt{FIFO}}(L)$. □

An immediate consequence of Theorem 5 is the closure of the languages of I-nets under exchanging unordered occurrences of transitions.

Corollary 2. $\mathtt{lin}(\mathcal{L}(\mathcal{I}(V,\varphi))) = \mathtt{lin}_{\mathtt{FIFO}}(\mathcal{L}(\mathcal{I}(V,\varphi))) = \mathcal{L}(\mathcal{I}(V,\varphi))$. □

The inclusion $\mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$ in Theorem 5 may be strict even if all inclusions $\mathtt{proj}_{T_i}(L) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$, are equalities (see Example 9 below). In fact, as we demonstrate next, it may be the case that not all words in $\mathcal{L}(\mathcal{I}(V,\varphi))$ are represented in L. This is expressed by means of the following notion.

Definition 9. *Let $L \subseteq T^*$ be a language with the prefix property. Then L is* complete *with respect to $\mathcal{I}(V,\varphi))$ if for all $w \in \mathcal{L}(\mathcal{I}(V,\varphi))$ there exists a $w' \in L$ such that $\mathtt{proj}_{T_i}(w) = \mathtt{proj}_{T_i}(w')$ for all $i \in [n]$.* □

By Theorem 5 we know that for a language L with the prefix property, that is locally executable by the component E-nets of $\mathcal{I}(V,\varphi)$, the FIFO-linearisation of L is included in the language of $\mathcal{I}(V,\varphi)$. The next theorem adds to this result by showing that this inclusion is an equality in case L is complete with respect to $\mathcal{I}(V,\varphi)$.

Theorem 6. *Let $L \subseteq T^*$ be a language with the prefix property such that $\mathtt{proj}_{T_i}(L) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. If $L \subseteq T^*$ is complete with respect to $\mathcal{I}(V,\varphi)$, then $\mathtt{lin}_{\mathtt{FIFO}}(L) = \mathcal{L}(\mathcal{I}(V,\varphi))$.*

Proof. Assume L is complete and let $w \in \mathcal{L}(\mathcal{I}(V,\varphi))$. Let $w' \in L$ be such that $\mathtt{proj}_{T_i}(w) = \mathtt{proj}_{T_i}(w')$ for all $i \in [n]$. Since L is complete, such w' exists. Moreover, w' has the prefix property. So, by Corollary 1, $\theta^{\mathtt{fifo}}_{w'}$ exists. Since $w \in \mathcal{L}(\mathcal{I}(V,\varphi))$, w has the prefix property by Lemma 2. Now we can apply Lemma 8 to conclude that $w \in lin_{\theta^{\mathtt{fifo}}_{w'}}(w')$, i.e., $w \in \mathtt{lin}_{\mathtt{FIFO}}(L)$. We conclude that $\mathcal{L}(\mathcal{I}(V,\varphi)) \subseteq \mathtt{lin}_{\mathtt{FIFO}}(L)$. By Theorem 5, $\mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{I}(V,\varphi))$. Hence equality follows. □

Example 9. (Ex 2. Ctd.) The E-nets $\mathcal{E}_1''$ and $\mathcal{E}_2''$ in Figs. 4(a) and 4(b), with transitions $T_1'' = \{\mathbf{so}, \mathbf{rc}, \mathbf{ii}\}$ and $T_2'' = \{\mathbf{so}, \mathbf{rc}, \mathbf{ei}\}$, are extensions of the E-nets in Fig. 1(a) and Fig. 1(b). Each has an extra internal action (**ii** and **ie**, respectively). The I-net in Fig. 3(c) is the composition of $\mathcal{E}_1''$ and $\mathcal{E}_2''$. Let $v = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{rc} \rangle$, $w = \langle \mathbf{so}, \mathbf{ro}, \mathbf{ie}, \mathbf{ii} \rangle$ and let $L = \mathtt{Pref}(\{v, w\})$. Then $\mathtt{proj}_{T_1''}(L) = \mathtt{Pref}(\{\langle \mathbf{so}, \mathbf{ii} \rangle, \langle \mathbf{so}, \mathbf{rc} \rangle\}) = \mathcal{L}(\mathcal{E}_1'')$ and $\mathtt{proj}_{T_2''}(L) = \mathtt{Pref}(\{\langle \mathbf{ro}, \mathbf{ie} \rangle, \langle \mathbf{ro}, \mathbf{sc} \rangle\}) = \mathcal{L}(\mathcal{E}_2'')$. Furthermore, we have $\mathtt{lin}_{\mathtt{FIFO}}(L) = \mathtt{Pref}(\{v, w, x\})$ where $x = \langle \mathbf{so}, \mathbf{ro}, \mathbf{ii}, \mathbf{ie} \rangle$.

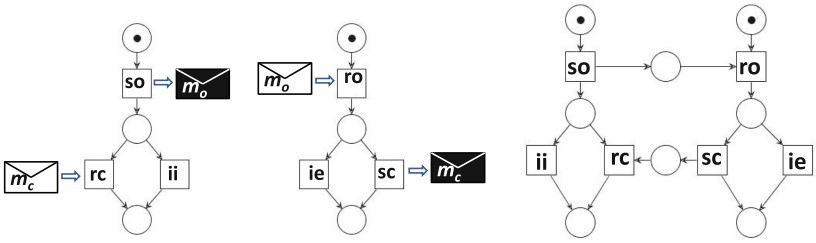


Fig. 4. (a) E-net $\mathcal{E}_1''$ (b) E-net $\mathcal{E}_2''$ (c) Their I-net

$\mathcal{L}(\mathcal{I}(V, \varphi)) = \mathtt{Pref}(\{v, w, x, y, z\})$ where $y = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{ii} \rangle$, and $z = \langle \mathbf{so}, \mathbf{ro}, \mathbf{ii}, \mathbf{sc} \rangle$. Hence $\mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{I}(V, \varphi))$. Even though $\mathtt{proj}_{T_1''}(L) = \mathcal{L}(\mathcal{E}_1'')$ and $\mathtt{proj}_{T_2''}(L) = \mathcal{L}(\mathcal{E}_2'')$, this inclusion is strict. L is not complete with respect to $\mathcal{I}(V, \varphi)$ because y does not have a representation in L. Neither v nor w (nor their prefixes) are such that their projection on T_1'' would yield $\mathtt{proj}_{T_1''}(y)$ and their projection on T_2'' would yield $\mathtt{proj}_{T_2''}(y)$: $\mathtt{proj}_{T_1''}(y) = \langle \mathbf{so}, \mathbf{ii} \rangle \neq \langle \mathbf{so}, \mathbf{rc} \rangle = \mathtt{proj}_{T_1''}(v)$ and, similarly, $\mathtt{proj}_{T_2''}(y) = \langle \mathbf{ro}, \mathbf{sc} \rangle \neq \langle \mathbf{ro}, \mathbf{ie} \rangle = \mathtt{proj}_{T_2''}(w)$. In contrast, $L' = L \cup \mathtt{pref}(y)$ is complete and we have $\mathtt{lin}_{\mathtt{FIFO}}(L') = \mathcal{L}(\mathcal{I}(V, \varphi))$. Note that $z \in \mathtt{lin}_{\mathtt{FIFO}}(L')$ as $\mathtt{proj}_{T_1''}(y) = \mathtt{proj}_{T_1''}(z)$ and $\mathtt{proj}_{T_2''}(y) = \mathtt{proj}_{T_2''}(z)$. □

Recall that $\mathcal{L}(\mathcal{I}(V, \varphi))$ has the prefix property. Hence combining Lemma 6(2) – by which $\mathtt{proj}_{T_i}(L) = \mathtt{proj}_{T_i}(\mathtt{lin}_{\mathtt{FIFO}}(L))$ whenever language L has the prefix property – with Theorem 6 leads to the observation that L and $\mathcal{L}(\mathcal{I}(V, \varphi))$ are in full agreement on their recordings of the components' behaviours.

Corollary 3. *Let $L \subseteq T^*$ be a language with the prefix property such that $\mathtt{proj}_{T_i}(L) \subseteq \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$. If $L \subseteq T^*$ is complete with respect to $\mathcal{I}(V, \varphi)$, then $\mathtt{proj}_{T_i}(L) = \mathtt{proj}_{T_i}(\mathcal{L}(\mathcal{I}(V, \varphi)))$ for all $i \in [n]$.* □

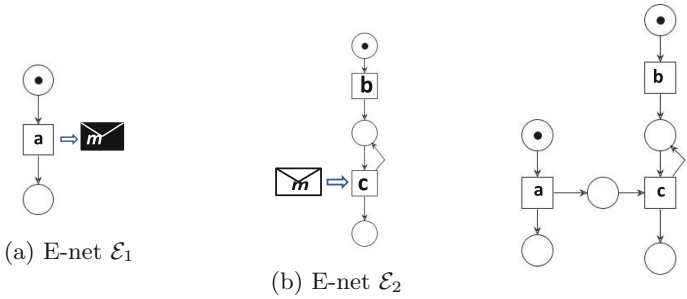
As illustrated next in Example 10, Corollary 3 does not imply that $\mathtt{proj}_{T_i}(L) = \mathcal{L}(\mathcal{E}_i)$ for all $i \in [n]$.

Example 10. Consider the I-net $\mathcal{I}(V,\varphi)$ in Fig. 5(c) composed of E-nets $\mathcal{E}_1$ in Fig. 5(a) and $\mathcal{E}_2$ in Fig. 5(b). Thus $\mathcal{L}(\mathcal{I}(V,\varphi)) = \texttt{Pref}(\{abc, bac\})$.

Consider now $L = \texttt{Pref}(\{abc\})$ which is complete with respect to $\mathcal{I}(V,\varphi)$. Moreover, L has the prefix property and $\texttt{proj}_{\{a\}}(\mathcal{L}(\mathcal{I}(V,\varphi))) = \texttt{proj}_{\{a\}}(L) = \texttt{Pref}(\{a\}) = \mathcal{L}(\mathcal{E}_1)$ as well as $\texttt{proj}_{\{b,c\}}(\mathcal{L}(\mathcal{I}(V,\varphi))) = \texttt{proj}_{\{b,c\}}(L) = \texttt{Pref}(\{bc\}) \subseteq \mathcal{L}(\mathcal{E}_2)$. However, $\texttt{proj}_{\{b,c\}}(L) = \mathcal{L}(\mathcal{E}_2)$ does not hold. □

6 Synthesising I-nets

We are now ready to address our initial question. Assume that we have an algorithm to discover Petri net models of isolated (i.e. not interacting) component processes from a description of their behaviour (in the form of action sequences). Then, when given an event log (a language), representing the observed behaviour of a given number of distinct collaborating processes, construct an I-net that generates this observed behaviour. A formal definition of the assumed process discovery algorithm, based on [6], is given next.



(a) E-net $\mathcal{E}_1$ (b) E-net $\mathcal{E}_2$ (c) The I-net $\mathcal{I}(V,\varphi)$

Fig. 5. (a) E-net $\mathcal{E}_1$ (b) E-net $\mathcal{E}_2$ (c) The I-net $\mathcal{I}(V,\varphi)$

Definition 10. *Let $\mathbb{L}$ be a family of languages. A* process discovery algorithm $\mathcal{A}$ *for $\mathbb{L}$ is an algorithm that computes for all $L \in \mathbb{L}$, a Petri net $\mathcal{A}(L) = (P,T,F)$ with a single source place such that $T = \texttt{Alph}(L)$ and $L \subseteq \mathcal{L}(\mathcal{A}(L))$.* □

Note that we require that the Petri nets discovered have a single source place (as E-nets have). Actually, such algorithms exist [6]. The Inductive Miner [22], a tool for discovering a workflow net implementing the communicating (input/output) behaviour, is one example. The similarity between E-nets and workflow nets makes it possible to leverage the large amount of research into the automated discovery of workflow nets, for the purpose of automated discovery of E-nets. A comprehensive overview of existing algorithms including an evaluation of their performance can be found in [8]. To leverage such an algorithm to a system of asynchronously communicating nets, one needs to describe the distribution of actions over components in combination with their role as internal, input, or

output action. This leads to the concept of a distributed communicating alphabet defined next.

Definition 11. *Let $n \geq 1$. An* n-dimensional distributed communicating alphabet *(or* n-DCA*, for short) is a tuple*
$\mathcal{DA} = (\langle \Sigma_1, \ldots, \Sigma_n \rangle, \langle \Sigma_{int}, \Sigma_{inp}, \Sigma_{out} \rangle, mt, cp)$ *such that*

- $\Sigma_1, \ldots, \Sigma_n$ *are non-empty, pairwise disjoint alphabets;*
- Σ_{int}, Σ_{inp}, Σ_{out} *are pairwise disjoint alphabets consisting of* internal actions, input actions *and* output actions*, respectively;*
- $\bigcup_{i \in [n]} \Sigma_i = \Sigma_{int} \cup \Sigma_{inp} \cup \Sigma_{out}$*;*
- $mt : \Sigma_{inp} \cup \Sigma_{out} \to \mathcal{M}$ *is a function that assigns message types to the input and output actions;*
- $cp : \Sigma_{inp} \cup \Sigma_{out} \to \Sigma_{inp} \cup \Sigma_{out}$ *is a (*complementing*) bijection, which is not defined ($cp = \emptyset$) if $n = 1$, and otherwise ($n \geq 2$), for all $a \in \Sigma_{inp} \cup \Sigma_{out}$: if $a \in \Sigma_{inp}$, then $cp(a) \in \Sigma_{out}$ and if $a \in \Sigma_{out}$, then $cp(a) \in \Sigma_{inp}$; if $a \in \Sigma_i$ for some $i \in [n]$, then $cp(a) \in \Sigma_j$ where $j \in [n]$ is such that $i \neq j$; $mt(cp(a)) = mt(a)$; and $cp(cp(a)) = a$.* □

The alphabet $\bigcup_{i \in [n]} \Sigma_i = \Sigma_{int} \cup \Sigma_{inp} \cup \Sigma_{out}$ in Definition 11, also referred to as the *underlying alphabet of the n-DCA* $\mathcal{DA}$, is intended to represent the actions available to n interacting enterprises. $\mathcal{DA}$ describes both their distribution across the enterprises and their interaction capacities. A 1-*DCA* consists of a single enterprise. When specifying a 1-*DCA*, we can omit its complementing bijection cp as it is not defined (and not needed). For $i \in [n]$, we refer to the 1-*DCA* $\mathcal{DA}_i = (\langle \Sigma_i \rangle, [\Sigma_{i,int}, \Sigma_{i,inp}, \Sigma_{i,out} \rangle, mt_i)$ as the *i-th component of* $\mathcal{DA}$.

For the rest of this section we assume a fixed family of languages $\mathbb{L}$ and a fixed process discovery algorithm $\mathcal{A}$ for $\mathbb{L}$. Moreover, we assume that we know the desired distribution of $\texttt{Alph}(L)$ over internal, input and output actions. The following definition describes how the discovery of E-nets depends on these assumptions.

Definition 12. *The* E-net discovery algorithm derived from $\mathcal{A}$*, is the algorithm* $\mathcal{A}_E$ *that computes, for all pairs* $(L, \mathcal{DA})$ *such that* $L \in \mathbb{L}$ *with* $\mathcal{A}(L) = (P, T, F)$*, and* $\mathcal{DA} = (\langle T \rangle, \langle T_{int}, T_{inp}, T_{out} \rangle, mt)$ *is a 1-*DCA *with* $\texttt{Alph}(L) = T$ *as its underlying alphabet, the E-net* $\mathcal{A}_E(L, \mathcal{DA}) = (P, \langle T_{int}, T_{inp}, T_{out} \rangle, F, mt)$*.* □

Thus $\mathcal{A}_E(L, \mathcal{DA})$ adds the information from $\mathcal{DA}$, about the distribution of actions, to the transitions of $\mathcal{A}(L)$. Clearly, $\mathcal{A}_E(L, \mathcal{DA})$ is an E-net.[3] Henceforth we fix an $\mathcal{A}_E$, as specified in Definition 12. The following lemma shows that the behaviour of $\mathcal{A}_E(L)$ is the same as that of $\mathcal{A}(L)$.

Lemma 9. *Let* $L \in \mathbb{L}$ *and* $\mathcal{DA}$ *a 1-*DCA *with* $\texttt{Alph}(L)$ *as its underlying alphabet. Then* $\mathcal{L}(\mathcal{A}(L)) = \mathcal{L}(\mathcal{A}_E(L, \mathcal{DA}))$*.*

Proof. The statement follows directly from Definition 12.

[3] In case $\mathcal{A}$ is an algorithm for the discovery of workflow nets, like the Inductive Miner [22], $\mathcal{A}_E(L, \mathcal{DA})$ would have the structure of a workflow net.

Now we move to the discovery of I-nets on the basis of $\mathcal{A}$.

Definition 13. *The* I-net discovery algorithm derived from $\mathcal{A}$, *is the algorithm* $\mathcal{A}_I$ *that computes, for all pairs* $(L, \mathcal{DA})$ *such that*

- $\mathcal{DA} = (\langle \Sigma_1, \ldots, \Sigma_n \rangle, \langle \Sigma_{int}, \Sigma_{inp}, \Sigma_{out} \rangle, mt, cp)$ *is an* n-DCA *for an* $n \geq 2$,
- $\mathtt{Alph}(L) = \bigcup_{i \in [n]} \Sigma_i$, *the underlying alphabet of* $\mathcal{DA}$, *and*
- $\mathtt{proj}_{\Sigma_i}(L) \in \mathbb{L}$ *for all* $i \in [n]$,

  the I-net $\mathcal{A}_I(L, \mathcal{DA}) = \mathcal{I}(V, \varphi)$ *with* $V = \{\mathcal{A}_E(\mathtt{proj}_{\Sigma_i}(L), \mathcal{DA}_i) \mid i \in [n]\}$ *and* $\varphi(a) = cp(a)$ *for all* $a \in \Sigma_{out}$. □

Note that $\mathcal{A}_I(L, \mathcal{DA}) = \mathcal{I}(V, \varphi)$ in Definition 13 is indeed an I-net, since, by Definition 12, for all $i \in [n]$, $\mathcal{A}_E(\mathtt{proj}_{\Sigma_i}(L), \mathcal{DA}_i)$ is an E-net with set of transitions Σ_i; the Σ_i are mutually disjoint; and the properties of cp described in Definition 11 guarantee that φ is a matching for V.

By Definitions 10 and 13, we can transfer Theorem 5 and Theorem 6 to the setting of discovering an I-net from a given language L.

Corollary 4. *Let* L *be a language with the prefix property and* $\mathcal{DA}$ *an* n-DCA, *both as specified in Definition 13. Then,*

(1) $\mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{A}_I(L, \mathcal{DA}))$;
(2) $\mathtt{lin}_{\mathtt{FIFO}}(L) = \mathcal{L}(\mathcal{A}_I(L, \mathcal{DA}))$ *if* L *is complete with respect to* $\mathcal{A}_I(L, \mathcal{DA})$. □

Thus, given a language L with the prefix property representing an event log of a system of n enterprises, and an n-dimensional distributed communicating alphabet, we can construct an I-net in the way described in Definition 13. This I-net has all words in L in its language together with all their linearisations (obtained by exchanging occurrences of independent actions). In fact, it is sufficient to consider only their FIFO-linearisations. Even then, though, there may in general be more firing sequences than what can be deduced from the description L of the observed behaviour. On the other hand, by Corollary 4(2), in case L is complete with respect to the I-net, i.e., each firing sequence of the constructed I-net has a representation in L, then the additional behaviour exhibited by the I-net, not observed in L (underfitting), can be fully explained by the concurrency between the E-nets.

Example 11. (Ex. 8 ctd.) Let w and v be as in Example 8 and let $L = \{w\}$. Let $\mathcal{DA} = (\langle \Sigma_1, \Sigma_2 \rangle, \langle \Sigma_{int}, \Sigma_{inp}, \Sigma_{out} \rangle, mt, cp)$ be the 2-*DCA*, with components $\mathcal{DA}_1$ and $\mathcal{DA}_2$, as given in Fig. 6, representing the actions available to the collaboration between the Investment firm and the Exchange from Example 8. Let $\mathcal{A}_E$ and $\mathcal{A}_\mathcal{I}$ be the E-net and I-net discovery algorithms, respectively, derived from process discovery algorithm $\mathcal{A}$. Furthermore, assume that the E-nets discovered from L, $\mathcal{A}_E(proj_{\Sigma_1}(L))$ and $\mathcal{A}_E(proj_{\Sigma_2}(L))$, are as depicted in Figs. 7(a) and (b), respectively. Then $\mathcal{A}_\mathcal{I}(L, \mathcal{DA})$ is the I-net in Fig. 3. By Corollary 4(1), we have $L \subseteq \mathtt{lin}_{\mathtt{FIFO}}(L) \subseteq \mathcal{L}(\mathcal{A}_\mathcal{I}(L, \mathcal{DA}))$. Hence, also $v \in \mathcal{L}(\mathcal{A}_\mathcal{I}(L, \mathcal{DA}))$ since $v \in \mathtt{lin}_{\mathtt{FIFO}}(w)$ as outlined in Example 8. □

The next example serves as an illustration of Corollary 4(2).

	Σ_{int}	Σ_{inp}	Σ_{out}
Σ_1	ii1,ii2	rc	so
Σ_2	ei1,ei2	ro	sc

$mt(\mathbf{so})=m_o$
$mt(\mathbf{ro})=m_o$
$mt(\mathbf{sc})=m_c$
$mt(\mathbf{rc})=m_c$

$cp(\mathbf{so})=\mathit{ro}$
$cp(\mathbf{ro})=\mathit{so}$
$cp(\mathbf{sc})=\mathit{rc}$
$cp(\mathbf{rc})=\mathit{sc}$

Fig. 6. The 2-DCA $\mathcal{DA}$

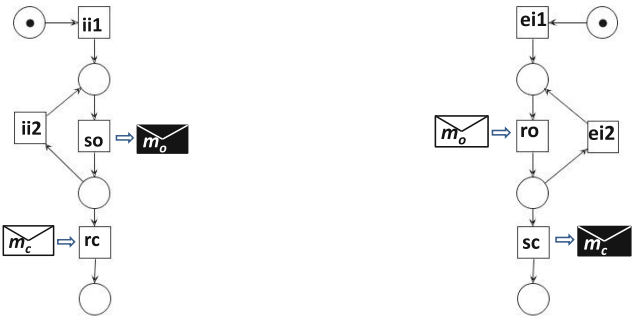


Fig. 7. (a) E-net $\mathcal{A}_E(proj_{\Sigma_1}(L))$ (b) E-net $\mathcal{A}_E(proj_{\Sigma_2}(L))$

Example 12. (Ex. 3 ctd.) Let $\mathcal{A}_E$ and $\mathcal{A}_\mathcal{I}$ be the E-net and I-net discovery algorithms respectively, derived from the process discovery algorithm $\mathcal{A}$. Let $w = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{rc}, \mathbf{si}, \mathbf{ri}, \mathbf{ss}, \mathbf{rs} \rangle$ as in Example 3 and let $L = \{w\}$. The 3-*DCA* $\mathcal{DA} = (\langle \Sigma_1, \Sigma_2, \Sigma_3 \rangle, \langle \Sigma_{int}, \Sigma_{inp}, \Sigma_{out} \rangle, mt, cp)$ is specified in Fig. 8.

It represents the actions for the collaboration between the Investment firm, the Exchange and the Central Securities Depository from Example 3 Let $\mathcal{DA}_1$, $\mathcal{DA}_2$, $\mathcal{DA}_3$ denote the first, second, and third component, resp., of $\mathcal{DA}$. Assume that $\mathcal{A}_E(proj_{\Sigma_1}(L), \mathcal{DA}_1)$, $\mathcal{A}_E(proj_{\Sigma_2}(L), \mathcal{DA}_2)$, and $\mathcal{A}_E(proj_{\Sigma_3}(L), \mathcal{DA}_3)$ are the E-nets depicted in Figs. 9(a), (b), (c) resp. Then $\mathcal{A}_\mathcal{I}(L, \mathcal{DA})$ is the I-net in Fig. 2.

	Σ_{int}	Σ_{inp}	Σ_{out}
Σ_1		rc,rs	so
Σ_2		ro	sc,si
Σ_3		ri	ss

$mt(\mathbf{so})=m_o$
$mt(\mathbf{ro})=m_o$
$mt(\mathbf{sc})=m_c$
$mt(\mathbf{rc})=m_c$
$mt(\mathbf{ss})=m_s$
$mt(\mathbf{rs})=m_s$
$mt(\mathbf{si})=m_i$
$mt(\mathbf{ri})=m_i$

$cp(\mathbf{so})=\mathit{ro}$
$cp(\mathbf{ro})=\mathit{so}$
$cp(\mathbf{sc})=\mathit{rc}$
$cp(\mathbf{rc})=\mathit{sc}$
$cp(\mathbf{ss})=\mathit{rs}$
$cp(\mathbf{rs})=\mathit{ss}$
$cp(\mathbf{si})=\mathit{ri}$
$cp(\mathbf{ri})=\mathit{si}$

Fig. 8. The 3-DCA $\mathcal{DA}$

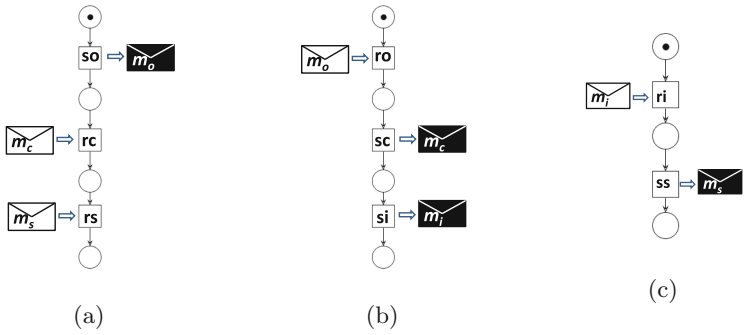


Fig. 9. (a) $\mathcal{A}_E(proj_{\Sigma_1}(L), \mathcal{DA}_1)$ (b) $\mathcal{A}_E(proj_{\Sigma_2}(L), \mathcal{DA}_2)$ (c) $\mathcal{A}_E(proj_{\Sigma_3}(L), \mathcal{DA}_3)$

Now, let $v = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{si}, \mathbf{rc}, \mathbf{ri}, \ \mathbf{ss}, \mathbf{rs} \rangle$ as in Example 3, $x = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \ \mathbf{si}, \mathbf{ri}, \ \mathbf{rc}, \mathbf{ss}, \mathbf{rs} \rangle$ and $y = \langle \mathbf{so}, \mathbf{ro}, \mathbf{sc}, \mathbf{si}, \mathbf{ri}, \mathbf{ss}, \mathbf{rc}, \mathbf{rs} \rangle$. Then $\mathcal{L}(\mathcal{A}_{\mathcal{I}}(L, \mathcal{DA})) = \mathtt{Pref}(\{w, v, x, y\})$. Note here that $\mathtt{lin}_{\mathtt{FIFO}}(\{w\}) = \{w, v, x, y\}$.

Next, we observe that L is complete with respect to $\mathtt{Pref}(\{w, v, x, y\})$ as $\mathtt{proj}_{\Sigma_i}(v) = \mathtt{proj}_{\Sigma_i}(x) = \mathtt{proj}_{\Sigma_i}(y) = \mathtt{proj}_{\Sigma_i}(w) \in \mathtt{proj}_{\Sigma_i}(L)$ for each $i \in \{1, 2, 3\}$ and similar for their prefixes. Furthermore, we know from Example 3 that L has the prefix property. And indeed $\mathcal{L}(\mathcal{A}_{\mathcal{I}}(L, \mathcal{DA})) = \mathtt{lin}_{\mathtt{FIFO}}(L)$. □

7 Discussion

In this paper we have considered the problem of the synthesis of a distributed process model (in the form of an I-net) from an event log (given in the form of a language). Also the number of participating processes (modelled as E-nets) is known as are their channels (in the form of matching input and output actions). We have shown how, given an algorithm for the discovery of Petri net models from event logs of isolated processes, the discovered models can be used for the synthesis of a new I-net. Moreover, by Theorems 1 and 5, fitness of the resulting I-net is guaranteed if and only if fitness of the component nets is guaranteed and the event log has the prefix property (a natural assumption). It is interesting to reflect on the inclusion of FIFO-linearisations in the language of the I-net (see Corollary 2 and Corollary 4.(1)) and the role of the channels. The exchange of unrelated occurrences of actions suggests a relationship to the well-known Mazurkiewicz traces (equivalence classes of words, see, eg., [25]) and their dependence graphs (defining their causal structure in the form of a labelled partial order, see [17]). There is however an important difference: independence in Mazurkiewicz' theory is between actions rather than their occurrences. The independence between occurrences considered here is determined by the history leading to these occurrences and hence to a theory of context dependent or local traces (see, eg., [11,19]). The I/O-Petri nets of [15] use communication channels modelled by places and channel properties are investigated in terms of asynchronous I/O-transition systems rather than languages. Note that, as can be

seen from our results, choosing FIFO channels or "normal" places (in which the order of arrival of tokens is not taken into account) does not change the language of the resulting I-net.

Process mining is an active research area, that has resulted in many process discovery algorithms (see, eg., [6] for an overview). Typically, these algorithms are used for the discovery of local processes in isolation, i.e., interaction between processes is not taken into account. In [5] this state of affairs is identified as an omission that should be addressed. This motivated us to investigate to what extent existing algorithms for the discovery of single (isolated) process models could be used for the synthesis of composed systems. In [5], two challenges are identified. The first one is "context dependency" which refers to the observation that composition of services might restrict their behaviour. Therefore, event logs generated by collaborating services might not show every possible local behaviour. As a consequence, the process models discovered, can be underfitting. We provide a precise specification of the conditions that determine the extent to which such underfitting can occur. The second challenge is "instance correlation" which concerns the problem of the use of different case identifiers for the same case by different services. This complicates the identification of global behaviour associated with a single case from local behaviours associated with that case. In this paper, this issue is not addressed as here event logs are seen as consisting of executed actions without explicit reference to cases. Currently we are investigating how to deal with case information associated with event sequences. Then the problem of instance correlation, mentioned above, could eg., reappear as the problem of relating observations of local behaviour to observations (as projections) of global behaviour.

In [2,4], the problem of process mining from large event logs is addressed and various options to distribute the mining problem over sublogs (with overlapping activities) are considered. In [10], similar to our approach, the event logs of Multi Agent Systems are projected onto individual agents in order to discover component models in terms of workflow nets using existing process discovery algorithms. By means of α-morphisms an abstraction of each component model is derived and the goal is to show that if the composition of these abstract models is sound, the composition of the original component models is sound.

As the focus of this paper has been on the collaboration between different parties, we did not deal with the question which algorithms from the set of available algorithms (see eg., [8]) is most suitable to serve as a precursor for the discovery of E-nets, and how an implementation of an algorithm for the discovery of E-nets can be derived from it. Since we are in particular concerned with the discovery of models of communicating behaviour this might also involve reconsidering performance criteria like fitness and precision, to accommodate this concern, for instance by making a distinction between performance with respect to internal and external behaviour. An interesting and practically relevant research question concerns the exchange of messages between processes (abstracting from their internal actions). One could adapt the approach presented here using projections on communicating actions only. The message exchanges generated from these

communicating actions, actually arise in the course of doing business between enterprises. For example, a significant volume of messages is exchanged on a daily basis, between the business processes of financial institutions connected by the computer network (SWIFTnet) that is maintained and monitored by the SWIFT organization[4]. Extending our model to include such indirect observations of communicating behaviour would open up the possibility to use these observations to support future, more empirical, research in this direction.

Acknowledgement. The authors are grateful to the anonymous reviewers for their constructive suggestions which have led to an improvement of the presentation of the results of this paper.

References

1. van der Aalst, W.M.P., Mooij, A.J., Stahl, C., Wolf, K.: Service interaction: patterns, formalization, and analysis. In: Bernardo, M., Padovani, L., Zavattaro, G. (eds.) SFM 2009. LNCS, vol. 5569, pp. 42–88. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01918-0_2
2. Aalst, W.M.P.: Distributed process discovery and conformance checking. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 1–25. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28872-2_1
3. van der Aalst, W.M.P., Weske, M.: Reflections on a decade of interorganizational workflow research. In: Seminal Contributions to Information Systems Engineering, pp. 307–313. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36926-1_24
4. van der Aalst, W.M.P.: Decomposing petri nets for process mining: a generic approach. Distr. Parallel Databases **31**(4), 471–507 (2013)
5. van der Aalst, W.M.P.: Service mining: using process mining to discover, check, and improve service behavior. IEEE Trans. Serv. Comput. **6**(4), 525–535 (2013)
6. van der Aalst, W.M.P.: Process Mining - Data Science in Action, 2nd edn. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-49851-4
7. van der Aalst, W.M.P., Stahl, C.: Modeling Business Processes - A Petri Net-Oriented Approach. MIT Press, Cambridge (2011)
8. Augusto, A., et al.: Automated discovery of process models from event logs: review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)
9. Baldan, P., Corradini, A., Ehrig, H., Heckel, R.: Compositional modeling of reactive systems using open nets. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 502–518. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44685-0_34
10. Bernardinello, L., Lomazova, I.A., Nesterov, R., Pomello, L.: Compositional discovery of workflow nets from event logs using morphisms. In: ATAED 2018. CEUR Workshop Proceedings, vol. 2115, pp. 39–55 (2018)
11. Biermann, I., Rozoy, B.: Reliable generalized and context dependent commutation relations. In: Bidoit, M., Dauchet, M. (eds.) CAAP 1997. LNCS, vol. 1214, pp. 165–176. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0030594

[4] In March 2021 this volume was in the order of 21 Mln. FIN messages per day. Source: https://www.swift.com/about-us/swift-fin-traffic-figures.

12. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer, Switzerland (2018). https://doi.org/10.1007/978-3-319-99414-7
13. EDSN: Marktfacilitering. [In Dutch] (2018).https://www.edsn.nl/
14. Gomes, L., Paulo Barros, J.: Structuring and composability issues in petri nets modeling. IEEE Trans. Indus. Inform. **1**(2), 112–123 (2005)
15. Haddad, S., Hennicker, R., Møller, M.H.: Channel properties of asynchronously composed petri nets. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 369–388. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_20
16. Heckel, R.: Open petri nets as semantic model for workflow integration. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 281–294. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40022-6_14
17. Hoogeboom, H.J., Rozenberg, G.: Dependence graphs. In: Diekert, V., Rozenberg, G. (eds.) The Book of Traces, pp. 43–67. World Scientific, Singapore (1995)
18. HL7: Health Level Seven International (2015). http://www.hl7.org/
19. Hoogers, P.W., Kleijn, H.C.M., Thiagarajan, P.S.: A trace semantics for petri nets. Inf. Comput. **117**(1), 98–114 (1995)
20. Kwantes, P.M., Kleijn, J.: On discovering distributed process models - the case of asynchronous communication. In: ATAED 2020. CEUR Workshop Proceedings, vol. 2625, pp. 49–65 (2020)
21. Kwantes, P.M., Kleijn, J.: On the synthesis of industry level process models from enterprise level process models. In: ATAED 2018 CEUR Workshop Proceedings, vol. 2115, pp. 6–22 (2018)
22. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
23. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Humboldt-Universitat zu Berlin, Mathematisch-Naturwissenschaftliche Fakultat II, Institut fur Informatik (2005)
24. S.W.I.F.T: ISO20022 Universal financial industry message scheme (2015). http://www.iso20022.org
25. Mazurkiewicz, A.: Trace theory. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) ACPN 1986. LNCS, vol. 255, pp. 278–324. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-17906-2_30
26. Reisig, W.: Towards a conceptual foundation of service composition. Comput. Sci. Res. Dev. **33**(3–4), 281–289 (2018)
27. Reisig, W.: Associative composition of components with double-sided interfaces. Acta Inf. **56**(3), 229–253 (2019)
28. GS1US: RosettaNet (2018). http://www.rosettanet.org/
29. Wolf, K.: Does my service have partners? Trans. Petri Nets Other Model. Concurr. **2**, 152–171 (2009)
30. Zaitsev, D., Sleptsov, A.: State equations and equivalent transformations for timed petri nets. Cybern. Syst. Anal. **33**, 659–672 (1997)
31. Zaitsev, D.: Decomposition of petri nets. Cybern. Syst. Anal. **4**, 131–140 (2004)