Pomsets for Local Trace Languages
— Recognizability, Logic & Petri Nets —

Dietrich Kuske and Rémi Morin*

Institut für Algebra, Technische Universität Dresden, D-01062 Dresden, Germany

Abstract. Mazurkiewicz traces can be seen as equivalence classes of words or as pomsets. Their generalisation by local traces was formalized by Hoogers, Kleijn and Thiagarajan as equivalence classes of step firing sequences. First we introduce a pomset representation for local traces. Extending Büchi’s Theorem and a previous generalisation to Mazurkiewicz traces, we show then that a local trace language is recognized by a finite step transition system if and only if its class of pomsets is bounded and definable in the Monadic Second Order logic. Finally, using Zielonka’s Theorem, we show that each recognizable local trace language is described by a finite safe labelled Petri net.

The complete version \[22\] of this paper is accessible on the web.

1 Introduction

Labelled partially ordered sets (pomsets) are widely used to model the behavior of a concurrent system \[30,15\]; in this approach, the order describes the causal dependence of the events while the labelling denotes which action is performed by an event. In particular, the incomparability of two events denotes that they can be executed simultaneously. Typical examples of this line of research are series-parallel pomsets \[23\], pomsets without autoconcurrency (also known as semiwords or partial words \[33,6\]) and dependence graphs of Mazurkiewicz traces \[24,7\]. A dependence graph is a pomset where the order relation is dictated by a static (or global) independence relation.

To any global independence relation, one can naturally associate an equivalence relation on words that identifies words if they differ only in the order in which independent actions occur. Then, a set of words is an equivalence class iff it is the set of linear extensions of some dependence graph. The rich theory of Mazurkiewicz traces is to a large extent based on these two alternative descriptions: as pomsets and as equivalence classes of words.

In the last decade, several generalizations of Mazurkiewicz traces have been proposed, among them partial traces \[6\], semi-commutations \[4\], P-traces \[1\], computations of concurrent automata \[9\] and local trace languages \[16\]. The local trace languages have two characteristic properties that distinguish them from Mazurkiewicz traces: First and similarly to concurrent automata, the independence of actions is not static, but depends on the history of a system. So here in one situation two actions might be independent while in another situation

* Supported by the German Research Foundation (DFG/Graduiertenkolleg)
they cannot be performed concurrently. The second distinguishing property is that not only pairs of actions are declared independent or dependent, but finite sets of actions. So it might happen that three actions are mutually independent and that they can be executed in any order, but the set of these three actions cannot be performed concurrently. Nevertheless, one considers the trace equivalence associated to a local trace language where this distinction is not visible anymore.

In this paper, we propose two pomset semantics for a local trace language based on the idea of several sequential observers. Any such sequential observer sees a linear execution of the events. Comparing several sequential observations, one can obtain (partial) knowledge about the concurrency in the execution which then is represented as a pomset. In the first pomset semantics (called processes) we detect only some, but not necessarily all concurrency. Differently in the second pomset semantics (called proper pomsets), all concurrency is represented. Of course, these two semantics are closely related: The processes are the order extensions of the proper pomsets and the proper pomsets are those processes whose order cannot be weakened anymore. One can describe the model of Mazurkiewicz traces, of concurrent automata as well as that of P-traces in the realm of local trace languages [17]. Our pomset semantics via proper pomsets generalize dependence graphs of Mazurkiewicz traces, dependence orders of stably concurrent automata [2] as well as CCI-sets of P-traces [1].

We finish this introduction with a description of the three results on these pomset semantics that we obtain here: Any independence alphabet from the theory of Mazurkiewicz traces defines a class of dependence graphs and there is a bijection between the equivalence classes w.r.t. the commutation of independent actions and these dependence graphs. Hence any set of dependence graphs corresponds to a closed language, namely to the language of linear extensions of one of its dependence graphs. In our context of local trace languages, there is no bijection between the equivalence classes and the pomsets. Therefore, not every set of pomsets is associated to a closed language. Our first result describes the sets of pomsets that correspond to some local trace language (Theorem 3.9).

Büchi’s paradigmatic result [3] on the relation between finite automata and monadic second order logic has been generalized into different directions, e.g. to finite and infinite trees [35,31], to dependence graphs of Mazurkiewicz traces [34,13], to dependence orders of computations of stably concurrent automata [11], to graphs [5], to series-parallel pomsets [21] etc. Here, we present two further generalizations: First, we show that a local trace language is recognizable (i.e. the language of a finite step transition system [26]) if and only if its set of processes can be defined in the monadic second order logic (Theorem 4.2). Furthermore, the set of proper pomsets of a recognizable local trace language can be defined in monadic second order logic. Conversely, there are non-recognizable local trace languages whose set of proper pomsets is definable. But we show that a local trace language is recognizable if and only if its set of proper pomsets is bounded and definable in monadic second order logic (Theorem 4.6).
Finally, we show how this work relates to the theory of Petri nets: *A local trace language is recognizable if and only if it is the language of a finite safe Petri net* (Theorem 5.3). This result relies on Zielonka’s Theorem [37] and answers a question that came up in a discussion with Thiagarajan at CONCUR’98.

## 2 Basic Notions

**Preliminaries.** We will use the following notations: for any (possibly infinite) alphabet \( \Sigma \), and any words \( u \in \Sigma^*, v \in \Sigma^* \), we write \( u \preceq v \) if \( u \) is a prefix of \( v \), i.e. there is \( z \in \Sigma^* \) such that \( u.z = v \); the empty word is denoted by \( \varepsilon \). We write \(|u|_a\) for the number of occurrences of \( a \in \Sigma \) in \( u \in \Sigma^* \) and \( \varphi_f(\Sigma) \) denotes the set of finite subsets of \( \Sigma \); for any \( p \in \varphi_f(\Sigma) \), \( \operatorname{Lin}(p) = \{ u \in p^* \mid \forall a \in p : |u|_a = 1 \} \) is the set of linearisations of \( p \). Finally, if \( \lambda : \Sigma \to \Sigma' \) is a map from \( \Sigma \) to \( \Sigma' \), we also write \( \lambda : \Sigma^* \to \Sigma'^* \) and \( \lambda : \varphi_f(\Sigma) \to \varphi_f(\Sigma') \) to denote the naturally associated monoid morphisms. For any set \( \Gamma \), \( \operatorname{Card}(\Gamma) \) denotes the cardinal of \( \Gamma \). For any positive integer \( k \), \([1, k]\) denotes the set \( \{1, 2, \ldots, k\} \).

Let \( t = (E, \preceq) \) be a finite partial order and \( x, y \in E \). Then \( y \) covers \( x \) (denoted \( x \shortrightarrow y \)) if \( x < y \) and \( x < z \preceq y \) implies \( y = z \). For all \( z \in E \), we denote by \( \downarrow z \) the set of nodes properly below \( z \), i.e. \( \downarrow z = \{ y \in E \mid y \preceq z \} \). Furthermore, for any subset \( M \) of \( E \), \( \downarrow M \) denotes \( \bigcup_{m \in M} \downarrow m \). The elements \( x \) and \( y \) are concurrent or incomparable (denoted \( x \leftrightarrow y \)) if \( \neg(x \preceq y) \land \neg(y \preceq x) \).

### 2.1 Local Independence Relations and Local Trace Languages

In this section, we introduce the model of a local trace language and define when they are recognizable. As mentioned in the introduction, in a local independence relation, we declare not only pairs of actions to be independent, but finite sets. Furthermore, the independence of such a set might depend on the history of the system, i.e. on the sequence of actions already performed:

**Definition 2.1.** A local independence relation over \( \Sigma \) is a non-empty subset \( I \) of \( \Sigma^* \times \varphi_f(\Sigma) \). The (local) trace equivalence \( \sim \) induced by \( I \) is the least equivalence on \( \Sigma^* \) such that

\[
\begin{align*}
\text{TE}_1: & \quad \forall u, u' \in \Sigma^* \forall a \in \Sigma: (u \sim u' \Rightarrow u.a \sim u'.a); \\
\text{TE}_2: & \quad \forall(u, p) \in I \forall p' \subseteq p \forall v_1, v_2 \in \operatorname{Lin}(p'): u.v_1 \sim u.v_2.
\end{align*}
\]

A (local) trace is an \( \sim \)-equivalence class \([u] \) of a word \( u \in \Sigma^* \).

Note that local trace equivalences are Parikh equivalences since trace equivalent words can be seen as permutations of each other. Usually, one considers only complete local independence relations which is e.g. justified in [19,25]:

**Definition 2.2.** A local independence relation \( I \) over \( \Sigma \) is complete if

\[
\begin{align*}
\text{Cpl}_1: & \quad (u, p) \in I \land p' \subseteq p \Rightarrow (u, p') \in I; \\
\text{Cpl}_2: & \quad (u, p) \in I \land p' \subseteq p \land v \in \operatorname{Lin}(p') \Rightarrow (u.v, p \setminus p') \in I; \\
\text{Cpl}_3: & \quad (u, \{a, b\}) \in I \land (u.ab.v, p) \in I \Rightarrow (u.ba.v, p) \in I; \\
\text{Cpl}_4: & \quad (u.a, \emptyset) \in I \Rightarrow (u, \{a\}) \in I.
\end{align*}
\]
The first axiom says that with any independent set, any of its subsets is independent. By the second axiom, once a set is independent, one can first execute part of this set and the remaining set stays independent. The third axiom asserts that trace-equivalent words lead to the same independence of steps. The last axiom requires that we list only words in the local independence relation that can be executed sequentially.

**Definition 2.3.** A local trace language is a structure $\mathcal{L} = (\Sigma, I, L)$ where $I$ is a complete local independence relation on $\Sigma$ and $L \subseteq \Sigma^*$ is such that

- $\text{LTL}_1$: $u \in L \Rightarrow (u, \emptyset) \in I$;
- $\text{LTL}_2$: $u \in L \land u \sim v \Rightarrow v \in L$.

In view of $\text{Cpl}_1$, the first axiom requires that $L$ contains only executable sequences of actions. The second axiom asserts that $L$ is closed w.r.t. the trace equivalence associated with the local independence relation.

Note here that these two requirements form a slight generalization of the local trace languages studied in [19,25,17] since the language $L$ is not necessarily prefix-closed. This extension corresponds to our aim to cope with systems provided with final states. A local trace language $(\Sigma, I, L)$ such that $L = \{ u \in \Sigma^* \mid (u, \emptyset) \in I \}$ is called saturated. Saturated local trace languages are precisely those related with event structures and Petri nets in [19,25].

The following example shows that a step might not be independent, even so any two of its linearisations are trace equivalent. This is fundamentally different from the setting of classical traces and of concurrent automata.

**Example 2.4.** We consider the alphabet $\Sigma = \{a, b, c\}$ and the local trace language $\mathcal{L} = (\Sigma, I, \Sigma^*)$ such that

$\forall u \in \Sigma^* \forall p \in \wp_f(\Sigma) : (u, p) \in I \Leftrightarrow [\text{Card}(p) \leq 2 \land (u = a \Rightarrow \text{Card}(p) \leq 1)]$.

We observe that $abc \sim bac \sim bca \sim cba \sim cab \simacb$ although $(a, \{b,c\}) \notin I$.

Actually, the trace equivalence is precisely Parikh’s equivalence.

In [17], recognizable saturated local trace languages are defined as those that can be represented by a finite step transition system [26]. Providing the latter with final states, we can extend naturally this approach to non-saturated languages. Equivalently (cf. [18]), recognizability for local trace languages is defined as follows.

**Definition 2.5.** A local trace language $\mathcal{L} = (\Sigma, I, L)$ is recognizable if and only if $\Sigma$ is finite, $L$ is recognizable in $\Sigma^*$, and for each step $p \in \wp_f(\Sigma)$, the language $L_p = \{ u \in \Sigma^* \mid (u, p) \in I \}$ is recognizable.

## 2.2 Global Independence Relations and Mazurkiewicz Traces

Local trace languages are a direct generalization of classical traces [7]. There, the independence between actions does not depend on the context of previously occurred events. Thus we consider a global independence relation over $\Sigma$ to be a binary symmetric and irreflexive relation $\parallel \subseteq \Sigma \times \Sigma$. Then a classical trace
language over \((\Sigma, ||)\) is a language \(L \subseteq \Sigma^*\) which is closed under the commutation of independent actions: \(\forall u, v \in \Sigma^* \forall a, b \in \Sigma : ((u.ab.v \in L \land a||b) \Rightarrow u.ba.v \in L)\). This leads us to introduce classical trace languages formally within the general framework of local trace languages as follows.

**Definition 2.6.** Let \(||\) be a global independence relation over \(\Sigma\). A classical trace language over \((\Sigma, ||)\) is a local trace language \(L = (\Sigma, I, L)\) such that for all \(u \in \Sigma^*\), for all \(n \in \mathbb{N}\), and for all distinct \(a_1, ..., a_n \in \Sigma\), we have \((u, \{a_1, ..., a_n\}) \in I\) if and only if \(a_i||a_j\) for all \(1 \leq i < j \leq n\).

Since \(L_p = \Sigma^*\) for any set \(p\) of pairwise independent actions and \(L_p = \emptyset\) otherwise, a classical trace language \(L_C = (\Sigma, I, L)\) over a finite alphabet \(\Sigma\) is recognizable (Def. 2.5) iff \(L\) is a recognizable language in \(\Sigma^*\). Note here that this representation of Mazurkiewicz traces within local trace languages is simpler than the approach followed in [17]; this is due to the fact that we consider in this paper non-saturated local trace languages.

### 3 Local Trace Languages Are Classes of Pomsets, Too

Classical traces admit several representations, for they can be identified with pomsets. We extend the relationship between traces and pomsets to the setting of local traces. *In this section, we fix a (possibly infinite) alphabet \(\Sigma\).*

#### 3.1 Pomsets – Basic Structures & Classical Traces

**Definition 3.1.** A pomset over \(\Sigma\) is a triple \(t = (E, \preceq, \xi)\) where \((E, \preceq)\) is a finite partial order and \(\xi\) is a mapping from \(E\) to \(\Sigma\). We denote by \(P(\Sigma)\) the class of all pomsets over \(\Sigma\). A pomset \(t = (E, \preceq, \xi)\) is without autoconcurrency if \(\xi(x) = \xi(y)\) implies \((x \preceq y \lor y \preceq x)\) for all \(x, y \in E\).

A pomset can be seen as an abstraction of an execution of a concurrent system. In this view, the elements \(e \in E\) are *events* and their label \(\xi(e)\) describes the basic action of the system that is performed by the event. Furthermore, the order describes the causal dependence between the events. In particular, if two events are concurrent, they can be executed in parallel. A pomset is without autoconcurrency if no action can be performed concurrently with itself.

A *prefix* of a pomset \(t = (E, \preceq, \xi)\) is the restriction of \(t\) to some downward closed subset of \(E\). An *order extension* of a pomset \(t = (E, \preceq, \xi)\) is a pomset \(t' = (E, \preceq', \xi)\) such that \(\preceq \subseteq \preceq'\). A *linear extension* of \(t\) is an order extension that is linearly ordered. Linear extensions of a pomset \(t = (E, \preceq, \xi)\) without autoconcurrency can naturally be identified with words over \(\Sigma\). By \(LE(t) \subseteq \Sigma^*\), we denote the set of linear extensions of a pomset \(t\) over \(\Sigma\). Since all the classes of pomsets considered (resp. defined) in this paper are assumed (resp. easily checked) to be closed under isomorphisms, we will identify isomorphic pomsets.

Let \(L_C = (\Sigma, I, L)\) be a classical trace language over \((\Sigma, ||)\) and let \(u \in \Sigma^*\). Then the trace \([u]\) is precisely the set of linear extensions \(LE(t)\) of some pomset
t = (E, ≤, ξ), i.e. [u] = LE(t). We denote by \( p_C(L_C) \) the class of all pomsets \( t \) such that \( LE(t) = [u] \) for some \( u \in L \). Then any \( t = (E, \leq, \xi) \in p_C(L_C) \) satisfies the following additional properties [24]:

MP\(_1\): for all events \( e_1, e_2 \in E \) with \( \xi(e_1) \parallel \xi(e_2) \), we have \( e_1 \leq e_2 \) or \( e_2 \leq e_1 \);

MP\(_2\): for all events \( e_1, e_2 \in E \) with \( e_1 \sim e_2 \), we have \( \xi(e_1) \parallel \xi(e_2) \).

In particular, MP\(_1\) ensures that \( t \) is without autoconcurrency. Let \( \mathbb{M}(\Sigma, ||) \) denote the class of pomsets over \( \Sigma \) satisfying MP\(_1\) and MP\(_2\). It is well-known [7] that \( p_C \) is a one-to-one correspondence between the classical trace languages over \( (\Sigma, ||) \) and the subclasses of \( \mathbb{M}(\Sigma, ||) \).

We want to have a similar presentation by classes of pomsets for arbitrary local trace languages. To this aim, in the following subsection we will define several classes of pomsets that are associated to a given local trace language.

### 3.2 Pomset Representations of Local Trace Languages

Recall that for classical traces, for any word \( u \), there exists a pomset \( t \) whose set of linear extensions equals the equivalence class containing \( u \), i.e. \( [u] = LE(t) \). The following example shows that this is not possible for local traces that we consider now (cf. also [2,1] for examples in similar settings).

**Example 3.2.** We consider the local trace language \( \mathcal{L} = (\Sigma, I, \Sigma^*) \) over the alphabet \( \Sigma = \{a, b, c\} \) with

\[
I = \{ (\varepsilon, \{a, c\}), (c, \{a, b\}), (\varepsilon, \{b, c\}) \} \cup \Sigma^* \times \{ \{a\}, \{b\}, \{c\}, \emptyset \}.
\]

The trace \( [acb] = \{acb, cab, cba, bca\} \) is easily shown not to be the set of linear extensions of any pomset. That is why we shall in the sequel associate several pomsets to this trace. Actually \( [acb] \) will be associated to the pomsets \( t_1 = (\Sigma, \text{Id}_\Sigma \cup \{ \{c\} \times \{a\} \}, \text{Id}_\Sigma) \) and \( t_2 = (\Sigma, \text{Id}_\Sigma \cup \{ \{c\} \times \{b\} \}, \text{Id}_\Sigma) \). Note here that \( [acb] = LE(t_1) \cup LE(t_2) \) and that the pomset \( t_3 = (\Sigma, \text{Id}_\Sigma \cup \{ \{c\} \times \{a, b\} \}, \text{Id}_\Sigma) \) is an order extension of \( t_1 \) and of \( t_2 \).

Pomsets associated to classical traces correspond to usual representations of concurrent executions as elementary event structures [27] or non-branching processes [28]. This interpretation can be generalized to local traces as follows.

**Definition 3.3.** Let \( \mathcal{L} = (\Sigma, I, L) \) be a local trace language. A process of \( \mathcal{L} \) is a pomset \( t = (E, \leq, \xi) \) without autoconcurrency such that for all prefixes \( t' = (E', \leq', \xi') \) of \( t \), and for all linear extensions \( u \in LE(t') \), we have

\[
(u, \xi(\min_{\leq}(E \setminus E'))) \in I.
\]

We denote by \( p^\circ(\mathcal{L}) \) the class of all processes of \( \mathcal{L} \).

Our restriction to pomsets without autoconcurrency corresponds precisely to the fact that we consider sets of concurrent actions, but not multisets (see also [11,20,19]). In Example 3.2, \( t_1, t_2 \) and \( t_3 \) are three processes of \( \mathcal{L} \).

**Example 3.4.** We consider again the local trace language \( \mathcal{L} = (\Sigma, I, \Sigma^*) \) of Example 2.4. We observe here that the pomset \( t = (\Sigma, \text{Id}_\Sigma, \text{Id}_\Sigma) \) is not a process of \( \mathcal{L} \), because \( (\varepsilon, \{a, b, c\}) \notin I \), whereas \( LE(t) = \text{Lin}\{a, b, c\} = [abc] \).
This example shows that there might be some pomsets $t$ which are not processes of $\mathcal{L}$ although $\text{LE}(t)$ consists of trace equivalent words. However, conversely, the linear extensions of a process always belong to the same trace. That is, if $t$ is a process of a local trace language $\mathcal{L}$, then $u \sim v$ for any linear extensions $u, v \in \text{LE}(t)$.

Note that the class of processes of any local trace language is closed under isomorphisms, prefixes and order extensions. Yet the class of pomsets $p_C(\mathcal{L}_C)$ of some classical trace language $\mathcal{L}_C$ is not closed under order extensions. Thus, in order to get a true extension of $p_C$, we have to focus now on proper pomsets:

**Definition 3.5.** Let $\mathcal{L}$ be a local trace language. A (proper) pomset of $\mathcal{L}$ is a process $t \in p^o(\mathcal{L})$ which is not an order extension of another process of $\mathcal{L}$. We denote by $p(\mathcal{L})$ the class of all (proper) pomsets of $\mathcal{L}$.

For instance, in Example 3.2, $t_1$ and $t_2$ are proper pomsets of $\mathcal{L}$ but not $t_3$, because $t_3$ is an order extension of $t_1$. The processes $p^o(\mathcal{L})$ are precisely the order extensions of elements of $p(\mathcal{L})$. Now, since we deal in this paper with possibly non-saturated local trace languages, $p(\mathcal{L})$ is not enough to represent $\mathcal{L}$ faithfully. Precisely we still have to specify which processes reach a final state.

**Definition 3.6.** Let $\mathcal{L} = (\Sigma, I, L)$ be a local trace language. A process of $\mathcal{L}$ is final if each of its linear extensions belongs to $L$. We denote by $p_f(\mathcal{L})$ (resp. $p^f_2(\mathcal{L})$) the class of all final proper pomsets of $\mathcal{L}$ (resp. final processes of $\mathcal{L}$).

Since linear extensions of a process are trace equivalent, a proper pomset of $\mathcal{L}$ is final if one of its linear extensions belongs to $L$. We can check that we obtain in that way a true extension of the usual representation $p_C$ of classical trace languages by classes of pomsets: Let $\mathcal{L}_C$ be a classical trace language over $(\Sigma, \parallel)$. First we observe that $p_f(\mathcal{L}_C) = p_C(\mathcal{L}_C)$, so the final proper pomsets of a classical trace language correspond to the usual dependence graphs associated to it. Second we can show that $p(\mathcal{L}_C) = \mathcal{M}(\Sigma, \parallel)$: this expresses that the proper pomsets describe all the possible concurrent behaviours enabled by the (global) independence relation, regardless of the accepting executions.

### 3.3 Expressive Power of Local Trace Languages

The aim of this subsection is to establish that the map from local trace languages to classes of pomsets is one-to-one (Th. 3.9). Also we give an order-theoretic characterization of the classes of pomsets which arise in that fashion. In this direction, the first step is to exhibit some properties of the class of processes of any local trace language.

**Example 3.7.** We consider here again the saturated local trace language $\mathcal{L} = (\Sigma, I, \Sigma^*)$ of Example 3.2. The processes $t_1$ and $t'_1 = (\{a, c\}, \text{Id}_{\{a, c\}}, \text{Id}_{\{a, c\}})$ are proper pomsets of $\mathcal{L}$. Therefore the prefix $t''_1$ of $t_1$ restricted to $\{a, c\}$ is not a proper pomset of $\mathcal{L}$ since $t''_1$ is a strict order extension of $t'_1$. Thus, in general, $p(\mathcal{L})$ is not closed under prefixes. We notice now that $t_1' = (\Sigma, \text{Id}_{\Sigma}, \text{Id}_{\Sigma})$ is not
a process of \( \mathcal{L} \) because \([acb] = \{acb, cab, cba, bca\}\). Thus, we have two processes\( t_1 = (E, \preceq, \xi) \) and \( t'_1 = (E', \preceq', \xi') \) such that \( E' \subseteq E \) and \( t'_1 = (E', \preceq' \cup \preceq_{|E \times (E \setminus E')} \cup \xi_{|E'}) \) is a prefix of \( t_1 \) and an order extension of \( t'_1 \), but \( t'_1 = (E, \preceq' \cup \preceq_{|E \times (E \setminus E')} \cup \xi) \) is not a process of \( \mathcal{L} \).

Despite of this example, we introduce now some properties of the class of processes of any local trace language (Def. 3.8 and Th. 3.9). These properties are similar to the one studied in Example 3.7, but of course they are different!

**Definition 3.8.** A class \( \mathcal{P} \) of pomsets is called consistent if the two following conditions are satisfied:

\( \text{Cons}_1: \) If two pomsets \( t = (E, \preceq, \xi) \) and \( t' = (E', \preceq', \xi') \) in \( \mathcal{P} \) are such that

\[-E' \subseteq E \text{ and } E \setminus E' \subseteq \text{max}_{\preceq} E,\]

\[-(E', \preceq' \cup \preceq_{|E \times E'}, \xi_{|E'}) \text{ is a prefix of } t \text{ and an order extension of } t',\]

then the pomset \( (E, \preceq' \cup (E' \times (E \setminus E')) \cup \text{Id}_E, \xi) \) belongs also to \( \mathcal{P} \).

\( \text{Cons}_2: \) If a pomset \( t = (E, \preceq, \xi) \) with maximal elements \( M \subseteq E \) is such that

\[-\forall m \in M : \ t_m = (E_m, \preceq_{|E_m \times E_m}, \xi_{|E_m}) \in \mathcal{P} \text{ where } E_m = E \setminus \{m\},\]

\[-(E, \preceq \cup ((E \setminus M) \times M), \xi) \in \mathcal{P},\]

then the pomset \( t \) belongs also to \( \mathcal{P} \).

Let \( \mathcal{L} \) be a local trace language and \( \mathcal{P} = \mathcal{P}^\emptyset(\mathcal{L}) \). If we take a process of \( \mathcal{L} \) and weaken its order relation, this weakening has to be compensated somewhere if we want the weakened pomset to be a process of \( \mathcal{L} \). The first axiom above states that, if we weaken the order in a downward closed set of events \( E' \), this can be compensated by putting the complement \( E \setminus E' \) on top of all of \( E' \). For an explanation of the second axiom, suppose \( t' = (E, \preceq \cup (E \setminus M) \times M), \xi \) is a process of \( \mathcal{L} \), but \( t = (E, \preceq, \xi) \) is not, i.e. we have a process \( t' \) where any maximal event is above any non maximal event and weakening this property kicks the pomset out of \( \mathcal{P} \). Then this being kicked out is already witnessed by a proper prefix \( E \setminus \{m\} \) of \( t' \) for some \( m \in M \).

This leads us to the main result of this section: each local trace language \( \mathcal{L} \) is faithfully represented by the classes of pomsets \( \mathcal{P}(\mathcal{L}) \) and \( \mathcal{P}_f(\mathcal{L}) \). In other words the mapping \( \mathcal{L} \mapsto (\mathcal{P}(\mathcal{L}), \mathcal{P}_f(\mathcal{L})) \) is one-to-one. Moreover, the pairs of classes of pomsets associated to local trace languages in that way are characterized as follows.

**Theorem 3.9.** Let \( \mathcal{P} \) and \( \mathcal{P}' \) be two classes of pomsets without autoconcurrency over \( \Sigma \) such that \( \mathcal{P} \neq \emptyset \). There exists a local trace language \( \mathcal{L} \) such that \( \mathcal{P} = \mathcal{P}(\mathcal{L}) \) and \( \mathcal{P}' = \mathcal{P}_f(\mathcal{L}) \) if and only if \( \mathcal{P} \) and \( \mathcal{P}' \) satisfy the following requirements:

1. for all pomsets \( t \in \mathcal{P} \) and \( t' \in \mathcal{P}' \), if \( t' \) is an order extension of \( t \) then \( t' = t \);
2. the class of order extensions of \( \mathcal{P} \) is consistent and closed under prefixes;
3. for all pomsets \( t \in \mathcal{P} \) and \( t' \in \mathcal{P}' \): If \( \text{LE}(t) \cap \text{LE}(t') \neq \emptyset \), then \( t \in \mathcal{P}' \).

In that case there is a unique local trace language \( \mathcal{L} \) such that \( \mathcal{P} = \mathcal{P}(\mathcal{L}) \) and \( \mathcal{P}' = \mathcal{P}_f(\mathcal{L}) \).
4 Recognizability, Logical Definability and Boundedness

We aim now at extending to the framework of local trace languages the relationship between recognizability and logical definability known for the free monoid as Büchi’s Theorem and already generalized to classical traces [7,13]: any classical trace language is recognizable if and only if its class of final proper pomsets is definable within the Monadic Second Order logic.

In this section, we fix a finite alphabet $\Sigma$. Formulas of the MSO language over $\Sigma$ that we consider involve first order variables $x, y, z$... for events and set variables $X, Y, Z$... for sets of events. They are built up from the atomic formulas $P_a(x)$ for $a \in \Sigma$ (which stands for “the event $x$ is labelled by the action $a$”), $x \preceq y$, and $x \in X$ by means of the boolean connectives $\neg, \lor, \land, \rightarrow, \leftrightarrow$ and quantifiers $\exists, \forall$ (both for first order and for set variables). Formulas without free variables are called sentences.

The satisfaction relation $\models$ between pomsets $t = (E, \preceq, \xi)$ and a sentence $\varphi$ of the monadic second order logic is defined canonically with the understanding that first order variables range over the events of $E$ and second order variables over subsets of $E$. The class of pomsets without autoconcurrency which satisfy a sentence $\varphi$ is denoted by $\text{Mod}(\varphi)$. We say that a class of pomsets without autoconcurrency $\mathcal{P}$ is MSO-definable if there exists a monadic second order sentence $\varphi$ such that $\mathcal{P} = \text{Mod}(\varphi).

Example 4.1. We consider here a Producer-Consumer system. Its alphabet is $\Sigma = \{p, c\}$ where $p$ represents a production of one item and $c$ a consumption. The language of this system describes all the possible sequences for which at each stage there are at least as many productions as consumptions. Formally, $L = \{ u \in \Sigma^* \mid \forall v \leq u : |v|_p \geq |v|_c \}$. We now want to model a possible independence between the producer and the consumer. Provided that there have already been enough items produced, the producer and the consumer can act simultaneously. This can be represented by the local independence relation $I$ defined as follows:

\begin{itemize}
  \item $\langle u, \emptyset \rangle \in I \iff u \in L \iff \langle u, \{p\} \rangle \in I$;
  \item $\langle u, \{c\} \rangle \in I \iff \langle u \in L \land |u|_p > |u|_c \rangle \iff \langle u, \{p, c\} \rangle \in I$.
\end{itemize}

Note here that $\mathcal{L} = (\Sigma, I, L)$ is not a classical trace language since $\text{ppc} \sim \text{pcp}$, but $\text{pc} \not\sim \text{cp}$. The proper pomsets of this local trace language are precisely those of the form depicted on the right. These are MSO-definable by

\begin{align*}
\forall x, y : & \quad P_p(y) \land x \preceq y \rightarrow P_p(x) \land \\
\forall y : & \quad (P_c(y) \rightarrow \exists x (P_p(x) \land x \rightarrow \xi y)) \land \\
\forall x, z : & \quad ((P_p(x) \land P_c(z) \land x \preceq z) \rightarrow \exists y (P_c(y) \land x \rightarrow \xi y))
\end{align*}

However, this saturated local trace language is not recognizable since $L$ is not recognizable.

This example shows that recognizability is not equivalent to MSO-definability in the general framework of local trace languages. Yet, Theorems 4.2 and 4.6 give
two characterizations of recognizable local trace languages in the spirit of Büchi’s Theorem.

4.1 Recognizability = Definability of Processes

By [13,11], there exists a formula \( \varphi \) with two free variables \( x \) and \( y \) such that for any pomset without autoconcurrency \( t = (E, \preceq, \xi) \), the set \( \varphi^t = \{(x,y) \in E^2 \mid t \models \varphi(x,y)\} \) is a linear extension of \( \preceq \). The proof from [8] (cf. proof of Theorem 5.1), is easily seen to work for general pomsets without autoconcurrency, even though the result is stated for Mazurkiewicz traces, only.

Now, we can state our first Büchi-type characterization of recognizable local trace languages in terms of its processes:

**Theorem 4.2.** A local trace language \( \mathcal{L} \) is recognizable if and only if its class of processes \( p^\circ(\mathcal{L}) \) and its class of final processes \( p^\circ_f(\mathcal{L}) \) are MSO-definable.

**Proof.** (sketch) Let \( \mathcal{L} \) be recognizable. By Büchi’s Theorem, we find for any \( p \subseteq \Sigma \) a formula \( \psi_p \) which defines \( \{u \in \Sigma^* \mid (u,p) \in I\} \). Hence, a pomset \( t = (E, \preceq, \xi) \) without autoconcurrency is a process of \( \mathcal{L} \) iff the following holds for any prefix \( t' = (E', \preceq|_{E'}, \xi|_{E'}) \):

- if the minimal elements of the complementary suffix of \( t \) are labeled by \( p \subseteq \Sigma \), then the word \( (E', \varphi^{t'}, \xi|_{E'}) \) satisfies \( \psi_p \)
- which can be expressed in monadic second order logic. \( \square \)

4.2 Boundedness Property & Classical Traces

Example 4.1 shows that there exist local trace languages whose class of proper pomsets is MSO-definable, but that are not recognizable. Thus, in general, the duality between recognizability and MSO-definability known for classical traces does not generalize directly to local trace languages. Therefore, now we introduce the notion of boundedness that will enable us to prove the desired generalization (cf. Th. 4.6 below).

**Definition 4.3.** [20] Let \( t = (E, \preceq, \xi) \) be a pomset and \( k \) be a positive integer. A \( k \)-chain covering of \( t \) is a family \( (C_i)_{i \in [1,k]} \) of subsets of \( E \) such that

1. each \( C_i \) is a chain in \((E, \preceq)\), i.e. \((C_i, \preceq \cap (C_i \times C_i)) \) is a linear order;
2. \( E = \bigcup_{i \in [1,k]} C_i \);
3. for all \( x, y \in E \): If \( x \sim y \), then there exists \( 1 \leq i \leq k \) such that \( \{x, y\} \subseteq C_i \).

For any \( k \in \mathbb{N} \), \( \mathbb{P}_k(\Sigma) \) denotes the class of pomsets over \( \Sigma \) which admit a \( k \)-chain covering. A class of pomsets over \( \Sigma \) is bounded if it is included in some \( \mathbb{P}_k(\Sigma) \). Note here that the class of proper pomsets of the Producer-Consumer system of Example 4.1 is not bounded.

As far as classical traces are concerned, the class \( \mathbb{M}(\Gamma, \|) \) of classical traces over \( (\Gamma, \|) \) is bounded by \( \text{Card}(\Gamma)^2 \) whenever \( \Gamma \) is finite. To see this, we simply consider for each pair of dependent actions \( a \| b \) the set of events labelled by \( a \)
or $b$. Then, by Axiom MP$_1$, this set forms a chain and by MP$_2$ the family of these chains forms a chain covering.

Conversely, any bounded class is the projection of some class of classical traces [20]: Consider the alphabet $\Gamma_k = \Sigma \times (\wp([1, k]) \setminus \{\emptyset\})$ provided with the global independence relation $|$ such that $(a, M) | (b, N)$ iff $M \cap N = \emptyset$. Let $\pi_1 : \Gamma_k \to \Sigma$ be the projection to the first component and define $\pi_1(E, \preceq, \xi) = (E, \preceq, \pi_1 \circ \xi)$ for any $(E, \preceq, \xi) \in \mathbb{M}(\Gamma_k, |)$. Then $\pi_1(\mathbb{M}(\Gamma_k, |)) = \mathbb{P}_k(\Sigma)$. Even more: Let $\mathcal{P} \subseteq \mathbb{P}(\Sigma)$ be a bounded and MSO-definable class of pomsets without autoconcurrency. Then the preimage $\pi_1^{-1}(\mathcal{P}) \subseteq \mathbb{M}(\Gamma_k, |)$ is MSO-definable, too. In other words, any MSO-definable and bounded class of pomsets without autoconcurrency is the projection of some MSO-definable class of classical traces:

**Lemma 4.4.** Let $\mathcal{P}$ be a class of pomsets without autoconcurrency over the finite alphabet $\Sigma$. If $\mathcal{P}$ is bounded by $k$ and MSO-definable, then there is an MSO-definable subclass $\mathcal{P}_C$ of $\mathbb{M}(\Gamma_k, |)$ such that $\pi_1 : \Gamma_k \to \Sigma$ induces a surjection from $\mathcal{P}_C$ onto $\mathcal{P}$.

In view of Theorem 4.2, the following corollary implies that a local trace language with a bounded class of proper pomsets is recognizable iff its classes of proper and of final proper pomsets are MSO-definable:

**Corollary 4.5.** Let $\mathcal{L}$ be a local trace language over the finite alphabet $\Sigma$ such that $p(\mathcal{L})$ is bounded by $k$. Then $p(\mathcal{L})$ (resp. $p_f(\mathcal{L})$, respectively) is MSO-definable if and only if $p^o(\mathcal{L})$ (resp. $p_f^o(\mathcal{L})$, respectively) is MSO-definable.

**Proof.** Suppose $p^o(\mathcal{L})$ is MSO-definable by $\varphi$. Note that a process $t = (E, \preceq, \xi)$ is a proper pomset iff, for any $e, f \in E$ with $e \rightarrow f$ and $\xi(f) \neq \xi(e)$, the pomset $(E, \preceq \setminus \{(e, f)\}, \xi)$ is not a process of $\mathcal{L}$. Replacing in $\varphi$ any subformula of the form $x \preceq y$ by $x \preceq y \land \neg(x = e \land y = f)$, we obtain a formula $\varphi'$. Now

$$p(\mathcal{L}) = \text{Mod}(\varphi \land \forall e, f : ((e \rightarrow f \land \xi(e) \neq \xi(f)) \rightarrow \neg \varphi')),$$

i.e. $p(\mathcal{L})$ is MSO-definable.

Conversely let $p(\mathcal{L})$ be defined by $\psi$. By Lemma 4.4 there exists an MSO-definable class of classical traces $\mathcal{P}_C \subseteq \mathbb{M}(\Gamma_k, |)$ with $\pi_1(\mathcal{P}_C) = p(\mathcal{L})$. Since the order of any pomset $t \in \mathcal{P}_C$ is dictated by the global independence relation $|$, the class $\mathcal{P}_C^t$ of order extensions of elements of $\mathcal{P}_C$ is MSO-definable, too. In addition, this class $\mathcal{P}_C^t$ is mapped by $\pi_1$ onto the class of order extensions of elements of $\pi_1(\mathcal{P}_C) = p(\mathcal{L})$, i.e. onto $p^o(\mathcal{L})$. Since the projection of an MSO-definable class is MSO-definable, the class $p^o(\mathcal{L})$ of processes of $\mathcal{L}$ is MSO-definable.

The proof for the final pomsets and processes goes through verbatim.

4.3 Recognizability = Definability and Boundedness of Pomsets

So far, we showed that a local trace language is recognizable whenever $p(\mathcal{L})$ and $p_f(\mathcal{L})$ are MSO-definable and bounded. The aim of this section is to show the inverse implication. In view of Cor. 4.5, it remains to show that $p(\mathcal{L})$ is bounded whenever $\mathcal{L}$ is recognizable. So, let $\mathcal{L}$ be recognizable. Then there exists a finite
monoid \((S, \cdot)\) and a homomorphism \(\eta : \Sigma^* \to S\) such that, for any \(u, v \in \Sigma^\star\) with \(\eta(u) = \eta(v)\) and any \(p \in \wp^\star(\Sigma)\) we have: \((u, p) \in I\) iff \((v, p) \in I\).

By contradiction assume that \(p(L)\) is unbounded. Then, for any \(n \in \mathbb{N}\), there exists \(t = (E, \preceq, \xi) \in p(L)\) and \(x_i, y_i \in E\) for \(1 \leq i \leq n\) such that

1. \(\xi(x_i) = \xi(x_j)\) and \(\xi(y_i) = \xi(y_j)\) for \(1 \leq i \leq j \leq n\),
2. \(x_i \prec y_i\) for \(1 \leq i \leq n\), and
3. \(x_i \prec x_j, y_i \prec y_j\), and \(x_j \not\preceq y_i\) for \(1 \leq i < j \leq n\).

Recall that a process \(t\) of \(L\) is a proper pomset if its order cannot be weakened without leaving the class \(p^\circ(L)\). Hence, we cannot remove any covering relation from \(t\), or, the other way round, for any edge in the covering relation of \(t\), there has to be a “reason”. In the context of local trace languages, this amounts to say that we find antichains \(M_i \subseteq E\) for \(1 \leq i \leq n\) such that \(y_i \in M_i\) and, for any linear extension \(u_i\) of the restriction of \(t\) to \(F_i = \downarrow M_i \setminus \{x\}\), we have \((u_i, \xi(M_i \cup \{x\})) \not\in I\). Since \(\Sigma\) is finite, we might assume \(\xi(M_i \setminus \downarrow x_n) = \xi(M_j \setminus \downarrow x_n)\) for \(1 \leq i < j < n\).

Next, one shows that \(\{x_j\} \cup (M_j \cap \downarrow x_n) \cup (M_i \setminus \downarrow x_n)\) consists of minimal elements of \(E \setminus ((F_j \cap \downarrow x_n) \cup (F_i \setminus \downarrow x_n))\) for \(1 \leq i < j < n\). Hence we obtain

\[
\xi(\min(E \setminus ((F_j \cap \downarrow x_n) \cup (F_i \setminus \downarrow x_n))))
\geq \xi(x_j) \cup \xi(M_j \setminus \downarrow x_n) \cup \xi(M_i \setminus \downarrow x_n)
= \xi(x_j) \cup \xi(M_j \setminus \downarrow x_n) \cup \xi(M_j \setminus \downarrow x_n) = \xi(M_j \cup \{x_j\}).
\]

So far, so good. Due to lack of space, we cannot give the details that prove the following step; the reader is invited to look at the complete and technical proof in [22]. This step uses Ramsey’s Theorem [32] and a close analysis of the relation of the sets \(M_i\) in the proper pomset \(t\). It results in the existence of \(1 \leq i < j < n\) with the following nice property: Let \(v\) be some linear extension of the restriction of \(t\) to the set \((F_j \cap \downarrow x_n) \cup (F_i \setminus \downarrow x_n)\). Recall that \(u_j\) is a linear extension of the set \(F_j = (F_j \cap \downarrow x_n) \cup (F_j \setminus \downarrow x_n)\). Then, and this is indeed the crucial point of the proof that comes out of Ramsey’s Theorem, \(\eta(v) = \eta(u_j)\).

Now we can show that this leads to a contradiction: By the choice of \(M_j\), we have \((u_j, \xi(M_j \cup \{x_j\})) \not\in I\). Hence, since \(\eta\) recognizes \(L\), we obtain \((v, \xi(M_j \cup \{x_j\})) \not\in I\) and therefore (by \(\text{Cpl}^\circ\))

\[
(v, \xi(\min(E \setminus ((F_j \cap \downarrow x_n) \cup (F_i \setminus \downarrow x_n)))) \not\in I.
\]

But this, indeed, contradicts that \(t = (E, \preceq, \xi)\) is a process of \(L\).

Hence, we indicated how to show that the class of proper pomsets \(p(L)\) is bounded for any recognizable local trace language \(L\). By Theorem 4.2 and Corollary 4.5, we now obtain our second Büchi-type result:

**Theorem 4.6.** A local trace language is recognizable if and only if its class of proper pomsets and its class of final proper pomsets are MSO-definable and bounded.

**Proof.** If \(p(L)\) is bounded and \(p(L)\) and \(p_f(L)\) are MSO-definable, then by Corollary 4.5, the classes \(p^\circ(L)\) and \(p^\circ_f(L)\) are MSO-definable, too. Hence, Theorem 4.2 asserts that \(L\) is recognizable. If, conversely, \(L\) is recognizable, by Theorem 4.2, the classes \(p^\circ(L)\) and \(p^\circ_f(L)\) are MSO-definable. Furthermore, by what we saw above, \(p(L)\) is bounded. Now Corollary 4.5 yields that the classes \(p(L)\) and \(p_f(L)\) are MSO-definable. \(\square\)
5 Finite Distributed Implementation by Petri Nets

In [26], Mukund extended the so-called synthesis problem of elementary nets [14] to the more general model of Petri nets. A similar study was achieved by Hoogers, Kleijn and Thiagarajan in [16]: Using some generalized regions, they characterized which local trace languages correspond to the behaviour of unlabelled (possibly non-safe) Petri nets. We show here that any recognizable saturated local trace language is the language of a finite safe labelled Petri net.

![Fig. 1. Petri net $\mathcal{N}$ of Ex. 5.4](image)

![Fig. 2. Traces of Petri net $\mathcal{N}$ (Ex. 5.4)](image)

**Definition 5.1.** A Petri net is a quadruple $\mathcal{N} = (S, T, W, M_{in})$ where

- $S$ is a set of places and $T$ is a set of transitions such that $S \cap T = \emptyset$;
- $W$ is a map from $(S \times T) \cup (T \times S)$ to $\mathbb{N}$, called weight function;
- $M_{in}$ is a map from $S$ to $\mathbb{N}$, called initial marking.

Given a Petri net $\mathcal{N} = (S, T, W, M_{in})$, $\text{Mar}_\mathcal{N}$ denotes the set of all markings of $\mathcal{N}$ that is to say functions $M : S \to \mathbb{N}$; a step $p \in \varphi_f(T)$ is enabled at $M \in \text{Mar}_\mathcal{N}$ if $M(s) \geq \sum_{t \in p} W(s, t)$ for all $s \in S$; in this case, we write $M[p] M'$ where $M'(s) = M(s) + \sum_{t \in p} (W(t, s) - W(s, t))$ and say that the transitions of $p$ may be fired concurrently and lead to the marking $M'$. A step firing sequence consists of a sequence of markings $M_0, ..., M_n$ and a sequence of steps $p_1, ..., p_n \in \varphi_f(T)$ such that $M_0 = M_{in}$ and $M_{k-1}[p_k] M_k$ for all $1 \leq k \leq n$.

**Definition 5.2.** A labelled Petri net is a structure $(S, T, W, M_{in}, \xi)$ where $(S, T, W, M_{in})$ is a Petri net and $\xi$ is a map from $T$ to an alphabet $\Sigma$.

The local independence relation associated to a labelled Petri net $\mathcal{N} = (S, T, W, M_{in}, \xi)$ is $I = \{(\xi(t_1...t_n), \xi(p)) \mid (t_1...t_n, p) \in T^* \times \varphi_f(T) \land M_{in} \{\{t_1\} M_1 ... \{\{t_n\} M_n [p] M_{in+1} \mid M_i \in \text{Mar}_\mathcal{N}\} \text{ and the set of sequential executions is } L = \{u \in \Sigma^* \mid (u, \emptyset) \in I\}$.

A labelled Petri net $\mathcal{N}$ is well-labelled if its associated local independence relation is complete. In that case $(\Sigma, I, L)$ is called the local trace language of $\mathcal{N}$.
Since all markings are “accepting”, the local trace language of a well-labelled Petri net is saturated.

The labelling \( \xi \) is called deterministic if for all step firing sequences \( M_{i+1} = M_i \cdot \{p_n\} M_n \) and all transitions \( t, t' \in T \):
\[
M_n (\{t\}) M_{n+1} (\{t'\}) M_{n+1} (\xi(t) = \xi(t') \Rightarrow t = t').
\]
This restriction ensures that two transitions enabled by a common reachable marking correspond to two distinct actions. This implies in particular that the net is well-labelled and the associated marking graph is a step transition system.

A Petri net is called finite if it has only a finite number of places, transitions and actions. It is called safe when there are only a finite number of reachable markings. One can easily check that the local trace language of any finite safe well-labelled Petri net is recognizable. Less obvious is the converse. The latter can be established using Theorem 4.6, Lemma 4.4 and [17, Th. 5.4] (which relies on Zielonka’s Theorem [37]).

**Theorem 5.3.** A saturated local trace language is recognizable if and only if it is the language of a finite, safe, well-labelled Petri net.

As shown by Example 5.4, Theorem 5.3 fails if one considers only Petri nets with deterministic labelling.

**Example 5.4.** Let \( \mathcal{L} = (\Sigma, I, L) \) be the saturated local trace language associated to the Petri net of Figure 1. Its traces are informally described on Figure 2. We consider now the saturated local trace language \( \mathcal{L}' \) over \( \Sigma \) associated to the local independence relation \( I' = I \setminus \{(ab, \{d\}, (abd, \emptyset)\} \). We prove here by contradiction that \( \mathcal{L}' \) is not the language of a Petri net with a deterministic labelling. For such a Petri net, all the occurrences of \( b \) correspond to the firing of the same transition because \( \{(a, \{b, c\}, (\varepsilon, \{a, c\}, (c, \{a, b\}, (\varepsilon, \{b, c\}) \subseteq I' \). Similarly for \( a \). Hence \( ab \) and \( ba \) lead to the same marking. Now one \( d \)-labelled transition is enabled after \( ba \), so the same transition is enabled after \( ab : (ab, \{d\}) \in I' \).

**References**


24. A. Mazurkiewicz: *Concurrent program schemes and their interpretations*. Aarhus University Publication (DAIMI PB-78, 1977)


