On Recognizable Stable Trace Languages

Jean-François Husson[1] and Rémi Morin[2⋆]

[1] IRIT, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse, France
[2] Institut für Algebra, Technische Universität Dresden, D-01062 Dresden, Germany

Abstract. We relate several models of concurrency introduced in the literature in order to extend classical Mazurkiewicz traces. These are mainly Droste's concurrent automata and Arnold's CCI sets of P-traces, studied in the framework of local trace languages. Also, a connection between these models and classical traces is presented in details through a natural notion of projection. These relationships enable us to use efficiently Arnold's result in two other frameworks. First, we give a finite distributed implementation for regular CCI sets of P-traces (or, equivalently, finite stably concurrent automata) by means of bounded labelled Petri nets. Second, we present a new, simple and constructive method to relate Stark's trace automata with Bednarczyk's asynchronous transition systems. This improves a recent result in Scott domain theory.

Introduction. Mazurkiewicz trace languages are a well-known and widely studied model of concurrency [4]. They were introduced in [13] to provide a partial order semantics for elementary Petri nets. In the past decade several different generalizations of classical traces have been studied in the literature. First, Droste introduced concurrent automata [5] for which the independence between actions is no longer a global independence relation, but depends on the current state of the system. These automata were shown to extend Bednarczyk's asynchronous transition systems [2] and Stark's trace automata [18]. Independently, Arnold introduced an extension of classical traces by means of labelled partial orders called P-traces [1]. In particular, a strong connection between *recognizable* classical trace languages and *regular* CCI sets of P-traces was established. More recently, local trace languages were introduced to give a trace semantics for Place/Transition nets [8,14]. There a local independence relation specifies in each configuration which subsets of actions can be executed concurrently.

At some point, it seems necessary to classify and relate the different models of concurrency arisen in the literature. For instance, the synthesis problem of Petri nets consists in characterizing which automata (or languages) correspond to the behavior of a Petri net [7,15,8]. More generally, semantical studies bring relationships between models of different levels of abstraction [20,2,16,11].

In this paper, we relate three models of concurrency which are roughly at the same level of abstraction. These are CCI sets of P-traces, stably concurrent automata and a restricted subclass of local trace languages called stable trace languages. The latter are also precisely compared to classical trace languages by

⋆ Supported by the German Research Foundation (DFG/Graduiertenkolleg)

J. Tiuryn (Ed.): FOSSACS 2000, LNCS 1784, pp. 177–191, 2000.

means of projections. We show that these relationships lead to some improvements for the theories of Petri nets, concurrent automata and dI-domains.

After some basic definitions relating recognizable local trace languages and Mukund's step transition systems [15], we introduce the subclass of stable trace languages with the help of some cube properties. The latter are actually meant to mimic the particular behaviors of stably concurrent automata. In that way, recognizable stable trace languages are easily shown to correspond to the behavior of finite stably concurrent automata. Next we focus on CCI sets of P-traces which are shown to be equivalent to some stable trace languages. Therefore they represent the behavior of stably concurrent automata. Also regular CCI sets of P-traces are associated to recognizable stable trace languages. Thus we obtain precise relationships between these three models.

These connections lead us to give a new formulation of a strong result due to Arnold [1, Th. 6.16] showing that these extensions of classical traces are closely related to the original model: *any recognizable stable trace language is the projection of a recognizable classical trace language.* This relationship holds also for non-recognizable languages over infinite alphabets. However, answering an open problem raised by Arnold, *we prove that this relationship fails in the case of non-recognizable stable trace languages over finite alphabets.* This relies on a counter-example provided by a Producer-Consumer system.

In a seminal paper [21], Zielonka proved that any recognizable classical trace language is described by an asynchronous automaton which provides a finite implementation in the form of distributed processes. In [1], Arnold introduced an extension of Zielonka's asynchronous automata, called P-asynchronous automata. However *these systems failed to describe all regular CCI sets of P-traces.* Besides, it is still an open problem to know which regular CCI sets of P-traces are described by P-asynchronous automata (obviously these are not the whole class of regular CCI sets of P-traces, see [10] for a counter-example). In order to avoid this restriction, *we present a construction of a finite distributed implementation for any recognizable stable trace language* (or any regular CCI set of P-traces) *in the form of a labelled Petri net.* This construction turns out to complete nicely a somewhat dual approach followed by Droste and Shortt [6]. There the Petri nets whose behavior corresponds to a stably concurrent automaton (or a stable trace language) are characterized by some simple conditions on the weight function.

In [17], Schmitt tackles the difficult problem to define a recognizability notion for coherent dI-domains. The basic idea is that a coherent dI-domain should be considered recognizable if it corresponds to the behavior of a finite distributed automaton. However several families of distributed automata might be considered and might give rise to different recognizability notions. The main result of [17] asserts that the coherent dI-domains obtained from either finite trace automata [18] or finite asynchronous transition systems [2] are the same. *We present here a new, simple and constructive proof of this result* — whereas Schmitt's approach is not constructive.

The proofs of our main results partly rely on technical results borrowed from [1] and [3]. A detailled study is available in [10].

1 Basic Notions

Preliminaries. We will use the following notations: for any (possibly infinite) alphabet Σ, and any words $u \in \Sigma^\star$, $v \in \Sigma^\star$, we write $u \leq v$ if u is a prefix of v, i.e. there is $z \in \Sigma^\star$ such that $u.z = v$; the empty word is denoted by ε. We write $|u|_a$ for the number of occurrences of $a \in \Sigma$ in $u \in \Sigma^\star$ and $\wp_f(\Sigma)$ denotes the set of finite subsets of Σ; for any $p \in \wp_f(\Sigma)$, $\mathrm{Lin}(p) = \{u \in p^\star \mid \forall a \in p, |u|_a = 1\}$ is the set of linearisations of p. Finally, if $\lambda : \Sigma \to \Sigma'$ is a map from Σ to Σ', we also write $\lambda : \Sigma^\star \to \Sigma'^\star$ and $\lambda : \wp_f(\Sigma) \to \wp_f(\Sigma')$ to denote the naturally associated monoid morphisms. For short, a right semi-congruence will be called right-congruence.

Local Independence Relations and Local Trace Languages. As established in [8,14], the behaviors of Petri nets are faithfully represented by local trace languages. These are a generalization of the classical Mazurkiewicz' traces [13] since they specify sets of independent actions rather than pairs.

Definition 1.1. *A* local independence relation *over Σ is a non-empty subset I of $\Sigma^\star \times \wp_f(\Sigma)$. The* (local) trace equivalence $\sim$ *induced by I is the least equivalence on $\Sigma^\star$ such that*

TE_1*:* $\forall u, u' \in \Sigma^\star, \forall a \in \Sigma, u \sim u' \Rightarrow u.a \sim u'.a$*;*

TE_2*:* $\forall (u, p) \in I, \forall p' \subseteq p, \forall v_1, v_2 \in \mathrm{Lin}(p'), u.v_1 \sim u.v_2$.

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

By TE_1 local trace equivalences are right-congruences. TE_2 asserts that for every subset of actions which are independent after a sequence u, all sequences obtained by executing first u and then in an arbitrary order the actions from this subset, are equivalent. Note also that local trace equivalences are Parikh equivalences: $u \sim u' \Rightarrow \forall a \in \Sigma, |u|_a = |u'|_a$.

These assumptions on the trace equivalence can be translated into explicit additional conditions on the local independence relation without affecting the resulting traces. A local independence relation satisfying these additional conditions is called *complete* and can be shown to be a maximal representative among local independence relations defining the same behaviors.

Definition 1.2. *A local independence relation I over Σ is* complete *if*

Cpl_1*:* $(u, p) \in I \wedge\ p' \subseteq p \Rightarrow (u, p') \in I$*;*

Cpl_2*:* $(u, p) \in I \wedge\ p' \subseteq p \wedge\ v \in \mathrm{Lin}(p') \Rightarrow (u.v, p \setminus p') \in I$*;*

Cpl_3*:* $(u, \{a, b\}) \in I \wedge\ (u.ab.v, p) \in I \Rightarrow (u.ba.v, p) \in I$*;*

Cpl_4*:* $(u.a, \emptyset) \in I \Rightarrow (u, \{a\}) \in I$.

Cpl_1 makes explicit what TE_2 from Def. 1.1 guarantees for the trace equivalence: if a set of actions p can be executed concurrently after u, then so can any subset of p; moreover, following Cpl_2, the step p can be split into a sequential execution v and a concurrent step of the remaining actions. We remark now that Cpl_3 is equivalent to the requirement that $u \sim u' \wedge\ (u, p) \in I \Rightarrow (u', p) \in I$. Thus Cpl_3 states that after two equivalent sequences the independency of actions is the

same; it corresponds to the right-congruence property TE_1 from Def. 1.1. Local independence relations satisfying Cpl_3 were called *consistent* in [16] and *durable* in [9]. Finally Cpl_4 guarantees that whenever $u.a$ is a sequential execution, then action a is allowed as a step after u.

In this paper, we study the local trace languages introduced in [11] as combinations of a complete local independence relation and a language of sequences.

Definition 1.3. *A* local trace language *over Σ is a structure $\mathcal{L} = (\Sigma, I, L)$ where I is a complete local independence relation on Σ and $L \subseteq \Sigma^\star$ is such that $u \in L \Leftrightarrow (u, \emptyset) \in I$.*

Note here that the set of sequences L is closed for the prefix relation and the trace equivalence. Moreover any local trace language is entirely determined by its associated local independence relation.

Global Independence Relations and Mazurkiewicz Traces. Local trace languages are actually a direct generalization classical traces [13,4]. There, the independence between actions does not depend on the context of previously occurred events. Thus we consider a *global independence relation* over Σ to be a binary symmetric and irreflexive relation $\| \subseteq \Sigma \times \Sigma$. Then a *classical trace language* over $(\Sigma, \|)$ consists of a language $L \subseteq \Sigma^\star$ which is closed for the commutation of independent actions: $\forall u, v \in \Sigma^\star$, $\forall a, b \in \Sigma$, $u.ab.v \in L \wedge a \| b \Rightarrow u.ba.v \in L$.

In order to connect this approach with local trace languages, *we will only consider here prefix-closed languages.* In that way any classical trace language can be formally identified with a local trace language $\mathcal{L} = (\Sigma, I, L)$ for which $(u, p) \in I$ if the actions in p are pairwise independent w.r.t. the global independence relation. This leads us to introduce formally Mazurkiewicz trace languages within the general framework of local trace languages as follows.

Definition 1.4. *Let $\|$ be a global independence relation over Σ. A* Mazurkiewicz trace language *over $(\Sigma, \|)$ is a local trace language $\mathcal{L} = (\Sigma, I, L)$ such that $\forall u \in \Sigma^\star$, $\forall n \in \mathbb{N}$, $\forall a_1, ..., a_n \in \Sigma$:*

$$(u, \{a_1, ..., a_n\}) \in I \Leftrightarrow u.a_1...a_n \in L \wedge \ \forall i, j \in [1, n] \ \textit{distinct}, a_i \| a_j .$$

Now associating any prefix-closed classical trace language L over a fixed independent alphabet $(\Sigma, \|)$ to the Mazurkiewicz trace language $\mathcal{L} = (\Sigma, I, L)$, where I is defined as in Def. 1.4, we build clearly a *one-to-one correspondence between prefix-closed classical trace languages and Mazurkiewicz trace languages.*

Despite of this nice formal connection, we should stress here that the local independence relation associated to a Mazurkiewicz trace language may have some unusual (but technically necessary) properties. In particular, if the language L is not *forward-closed* w.r.t. the global independence relation $\|$ then there are a word u and two actions a and b such that $u.a \in L$, $u.b \in L$, $a \| b$ but $u.ab \notin L$; in that case, a and b are not independent after u: $(u, \{a, b\}) \notin I$.

Recognizable Languages and Finite Step Transition Systems. The model of step transition systems was introduced by Mukund [15] in order to extend the

so-called synthesis problem of elementary Petri nets [7] to the more general model of Place/Transition nets.

Definition 1.5. *A* step transition system *over the alphabet Σ is a structure $\mathcal{A} = (Q, s, \Sigma, \longrightarrow)$ where Q is a set of states, $s \in Q$ is an initial state and $\longrightarrow \subseteq Q \times \wp_f(\Sigma) \times Q$ is a set of labelled transitions such that*

- $\forall q_1, q_2 \in Q$: $q_1 \xrightarrow{\emptyset} q_2 \Leftrightarrow q_1 = q_2$;
- $\forall q_1, q_2 \in Q$, $\forall p' \subseteq p \in \wp_f(\Sigma)$: $q_1 \xrightarrow{p} q_2 \Rightarrow \exists q_3 \in Q$, $q_1 \xrightarrow{p'} q_3 \xrightarrow{p \setminus p'} q_2$;
- $\forall q_1, q_2, q_3 \in Q$, $\forall p \in \wp_f(\Sigma)$: $q_1 \xrightarrow{p} q_2 \wedge\ q_1 \xrightarrow{p} q_3 \Rightarrow q_2 = q_3$.

The step transition system $\mathcal{A}$ is finite *if Σ and Q are finite.*

As usual, for any word $u = a_1 ... a_n \in \Sigma^\star$, we write $q \xrightarrow{u} q'$ if there are states q_0,..., q_n such that $q_0 = q$, $q_n = q'$ and for each $i \in [1, n]$, $q_{i-1} \xrightarrow{\{a_i\}} q_i$. Let us also stress here that we only consider deterministic step transition systems. This is actually meant to make sure that the local independence relations intuitively associated to them are complete — in particular, they satisfy Cpl_3.

Definition 1.6. *The local trace language associated to a step transition system $\mathcal{A} = (Q, s, \wp_f(\Sigma), \longrightarrow)$ is the structure $\mathcal{L} = (\Sigma, I, L)$ where*

- $\forall u \in \Sigma^\star$: $u \in L \Leftrightarrow \exists q \in Q, s \xrightarrow{u} q$;
- $\forall u \in \Sigma^\star$, $\forall p \in \wp_f(\Sigma)$: $(u, p) \in I \Leftrightarrow \exists q_1, q_2 \in Q, s \xrightarrow{u} q_1 \xrightarrow{p} q_2$.

Step transition systems define naturally a notion of recognizability which extends a similar notion well-known and widely studied in the case of classical language theory or classical trace languages.

Definition 1.7. *A local trace language is* recognizable *if it is the language of a* finite *step transition system.*

Note here that if $\mathcal{L} = (\Sigma, I, L)$ is recognizable then L is a recognizable language of $\Sigma^\star$, but the converse is false — except, e.g., for Mazurkiewicz trace languages over finite independent alphabets.

2 Stable Trace Languages

We introduce in this section the subclass of stable trace languages. These later generalize Mazurkiewicz traces and Nielsen, Sassone and Winskel's generalized trace languages [16].

Cube Properties in Local Trace Languages. Stable trace languages are characterized by cube properties that can be formalized as follows.

Definition 2.1. *A* stable trace language *is a local trace language $\mathcal{L} = (\Sigma, I, L)$ such that*

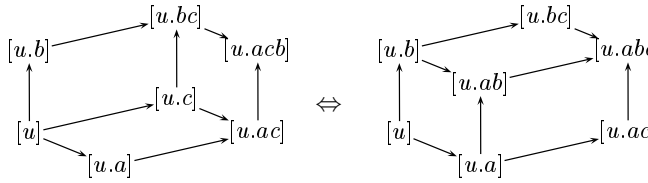
S_1: $\forall u \in \Sigma^\star$, $\forall n \geq 2$, $\forall a_1, ..., a_n \in \Sigma$ *distinct:*
$\left[\forall \sigma : [1, n] \to [1, n]$ onto $: u.a_1...a_n \sim u.a_{\sigma(1)}...a_{\sigma(n)}\right] \Rightarrow (u, \{a_1, ..., a_n\}) \in I$

S_2: $\forall u \in \Sigma^\star$, $\forall a, b, c \in \Sigma$ *distinct:*
$[(u, \{a, c\}) \in I \wedge\ (u.a, \{b, c\}) \in I] \Rightarrow [(u, \{a, b\}) \in I \Leftrightarrow (u.c, \{a, b\}) \in I]$

Condition S_1 asserts that whenever a set of actions may be executed in any order after a given sequence without affecting the resulting trace then these actions are mutually independent. Note here that the converse always holds. Therefore S_1 means simply that the independence relation I is somehow determined by its trace equivalence $\sim$. Now, the second condition S_2 requires that the concurrency between actions satisfies some local properties. As explained by the following proposition, this insures that the set of traces of a stable trace language satisfies some *cube properties* (CP) similar to those used to characterize stably concurrent automata.

Proposition 2.2. *Let $\mathcal{L} = (\Sigma, I, L)$ be a local trace language satisfying S_1. In the following diagrams, for all $u, v \in \Sigma^\star$, we note $[u] \longrightarrow [v]$ if there is $a \in \Sigma$ such that $u.a \sim v$. The language $\mathcal{L}$ is stable iff $\forall u \in \Sigma^\star$, $\forall a, b, c \in \Sigma$ distinct:*

(CP)



It is clear that any Mazurkiewicz trace language is a stable trace language. Let us also mention here that Nielsen, Sassone and Winskel's *generalized trace languages* [16] can be identified to the stable trace languages which satisfy the following additional *coherence property*: if $(u, \{a, b\}) \in I$, $(u, \{a, c\}) \in I$ and $(u, \{b, c\}) \in I$ then $(u, \{a, b, c\}) \in I$.

Stably Concurrent Automata. We present now the very natural connection between stable trace languages and stably concurrent automata.

Definition 2.3. *[3] An* automaton with concurrency relations *over the alphabet Σ is a structure $\mathcal{A} = (Q, s, \Sigma, \longrightarrow, (\|_q)_{q \in Q})$ such that*

1. *Q is a non-empty set of states, with an initial state s;*
2. *$\longrightarrow \subseteq Q \times \Sigma \times Q$ is a set of transitions assumed deterministic, i.e. whenever $p \xrightarrow{a} q$ and $p \xrightarrow{a} r$ then $q = r$;*
3. *$(\|_q)_{q \in Q}$ is a family of irreflexive, symmetric binary relations on Σ; it is required that whenever $a \|_p b$ then there exist transitions $p \xrightarrow{a} q$, $p \xrightarrow{b} q'$, $q \xrightarrow{b} r$ and $q' \xrightarrow{a} r$.*

Note that we only consider automata with concurrency relations provided with a *single initial state*. On the other hand, the set of states and the alphabet may be infinite. The language L associated to an automaton with concurrency relations is the set of finite sequences $u = a_1 ... a_n \in \Sigma^\star$ such that there are states q_0,...,q_n for which $s = q_0$ and for each $i \in [1, n]$, $q_{i-1} \xrightarrow{a_i} q_i$. For short, these conditions will be denoted by $s \xrightarrow{u} q_n$. Now the independence relations $\|_q$

provide naturally an equivalence relation over L as follows. The trace equivalence $\sim$ associated to $\mathcal{A}$ is the least equivalence over L such that $\forall u, v \in \Sigma^\star, \forall a, b \in \Sigma$: $s \xrightarrow{u} p \xrightarrow{ab} q \xrightarrow{v} r \wedge\ a\|_q b \Rightarrow u.ab.v \sim u.ba.v$.

For many different reasons, it appears that one may expect the independence relations $\|_q$ to depend locally of each other. In that way, a particular attention has been devoted to stably concurrent automata. In the following definition, for all actions a, b and c, and for all state q, we note $a\|_{q.c}b$ if there exists a state $q' \in Q$ such that $q \xrightarrow{c} q'$ and $a\|_{q'}b$.

Definition 2.4. *[3] A automaton with concurrency relations $\mathcal{A}$ is called* stably concurrent automaton *if for all $q \in Q$ and all actions a, b, $c \in \Sigma$, the following equivalence holds: $a\|_q c \wedge\ b\|_q c \wedge\ a\|_{q.c}b \Leftrightarrow a\|_q b \wedge\ b\|_{q.a}c \wedge\ a\|_{q.b}c$. We say that $\mathcal{A}$ is finite if Q and Σ are finite.*

A fundamental property of stably concurrent automata is the following correspondence between the trace equivalence $\sim$ and the family of independence relations $(\|_q)_{q\in Q}$: $\forall u \in \Sigma^\star$, $\forall a, b \in \Sigma$ distinct, $u.ab \sim u.ba \Leftrightarrow s \xrightarrow{u} q \wedge\ a\|_q b$. Therefore the assumption on $(\|_q)_{q\in Q}$ in Def. 2.4 corresponds precisely to the cube properties (CP) of Prop. 2.2. Also, the independency of actions is entirely determined by the trace equivalence. This remark lead us to represent the behavior of stably concurrent automata by stable trace languages as follows.

Definition 2.5. *Let $\mathcal{A}$ be a stably concurrent automaton over Σ, L be its language and $\sim$ be its trace equivalence. The stable trace language associated to $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = (\Sigma, I, L)$ where $\forall u \in \Sigma^\star$, $\forall n \in \mathbb{N}$, $\forall a_1, ..., a_n \in \Sigma$ distinct:*

$$(u, \{a_1, ..., a_n\}) \in I \Leftrightarrow \begin{cases} u.a_1...a_n \in L \\ \forall \sigma : [1,n] \to [1,n] \text{ onto} : u.a_1...a_n \sim u.a_{\sigma(1)}...a_{\sigma(n)} \end{cases}.$$

We easily check that $\mathcal{L}(\mathcal{A})$ is indeed a stable trace language. Moreover the restriction of the trace equivalence of $\mathcal{L}(\mathcal{A})$ to L is precisely the trace equivalence of $\mathcal{A}$. We stress that $\mathcal{L}(\mathcal{A})$ is a representation of the behavior of $\mathcal{A}$ equivalent to the *labelled dI-domain* usually considered (see e.g. [3]). Furthermore any stable trace language is the language of a stably concurrent automaton. Besides a stable trace language is recognizable if and only if it is the trace language of a *finite* stably concurrent automaton.

Full Stable Trace Languages Are Stable Right-Congruences. Although stable trace languages play a central role to relate stably concurrent automata with CCI sets of P-traces, we need to introduce first an equivalent representation in the form of particular right-congruences.

Definition 2.6. *Let $\sim$ be a right-congruence over $\Sigma^\star$. The associated* diamond relation $\sim^\diamond$ *is the least right-congruence over $\Sigma^\star$ such that*

$$\forall u \in \Sigma^\star, \forall a, b \in \Sigma,\ u.ab \sim u.ba \Rightarrow u.ab \sim^\diamond u.ba.$$

We say that the right-congruence $\sim$ is homotopic *if $\sim^\diamond = \sim$.*

It is clear that for all right-congruence $\sim$, $\sim^\diamond \subseteq \sim$. The converse inclusion holds in particular for the trace equivalence of any local trace language which is thus a homotopic right-congruence.

Definition 2.7. *A right-congruence over $\Sigma^\star$ is* stable *if it is homotopic and satisfies axiom* (CP) *of Prop. 2.2, whenever $a, b, c \in \Sigma$ are distinct and $u \in \Sigma^\star$.*

Clearly, the trace equivalence of a stable trace language is a stable right-congruence. However, different stable trace languages may determine the same trace equivalence. That is why we focus now on *full local trace languages*. The latter are defined as the local trace languages $\mathcal{L} = (\Sigma, I, L)$ such that $L = \Sigma^\star$.

Proposition 2.8. *An equivalence relation over $\Sigma^\star$ is the trace equivalence of a stable trace language if and only if it is a stable right-congruence. Moreover, in that case, it is the trace equivalence of a unique* full *stable trace language.*

3 CCI Sets of P-Traces

We show here a one-to-one correspondence between Arnold's CCI sets of P-traces [1] and full stable trace languages. Moreover, regular CCI sets of P-traces correspond to recognizable full stable trace languages.

P-Traces. In this section we consider a fixed alphabet Σ. Note here that we shall consider a slight extension of Arnold's approach since Σ may be infinite.

Definition 3.1. *[1] A* P-trace *t over Σ is a triple $(E_t, \prec_t, \xi_t)$ where $(E_t, \prec_t)$ is a finite partial order and ξ_t is a mapping from E_t to Σ such that for all $x, y \in E_t$, $\xi_t(x) = \xi_t(y) \Rightarrow (x \prec_t y \text{ or } y \prec_t x)$.*

Definition 3.2. *A* linear extension *of a P-trace $t = (E_t, \prec_t, \xi_t)$ is a total order $\prec$ over E_t such that $\prec_t \subseteq \prec$.*

Now, linear extensions of a P-trace t can easily be identified to words over Σ. Formally, let n be the cardinal of E_t. For any linear extension $\prec$ of t, there is only one way to write $E_t = \{e_1, ..., e_n\}$ with $e_i \prec e_j \Leftrightarrow i \leq j$. Then the word associated to $\prec$ is $\xi_t(e_1)...\xi_t(e_n)$. Clearly, this mapping from linear extensions of t to words is one-to-one. In the following, we shall identify any linear extension of t with its associated word.

Definition 3.3. *Let t be a P-trace over Σ. We note LE(t) the set of all the words associated to a linear extension of t.*

P-traces are naturally structured with a notion of isomorphism: two P-traces $t = (E_t, \prec_t, \xi_t)$ and $t' = (E_{t'}, \prec_{t'}, \xi_{t'})$ are isomorphic if there is a bijection σ from E_t to $E_{t'}$ such that

- $\forall x, y \in E_t : x \prec_t y \Leftrightarrow \sigma(x) \prec_{t'} \sigma(y)$;
- $\forall x \in E_t : \xi_t(x) = \xi_{t'}(\sigma(x))$.

Clearly, two isomorphic P-traces admit the same linear extensions. Noteworthy is the converse property due to Szpilrajn [19].

Proposition 3.4. *Two P-traces t and t' are isomorphic iff LE(t) = LE(t').*

CCI Sets of P-Traces Are Stable Right-Congruences Too. As in the classical case, a P-trace is meant to represent one concurrent execution of a distributed system. In order to describe all the possible behaviors of a system, one has to consider sets of P-traces.

Definition 3.5. *[1] Let $\mathcal{P}$ be a set of P-traces over Σ. We say that $\mathcal{P}$ is consistent and complete if*
- $\bigcup_{t\in\mathcal{P}} LE(t) = \Sigma^\star$ *[Complete]*
- $\forall t, t' \in \mathcal{P},\ t \neq t' \Rightarrow LE(t) \cap LE(t') = \emptyset$ *[Consistent]*

Each consistent and complete set of P-traces determines an equivalence relation $\sim$ over $\Sigma^\star$ whose equivalence classes are the linear extensions of its elements. This equivalence will be called the *trace equivalence* of $\mathcal{P}$.

However, this equivalence relation is sometimes not a right-congruence, which is admittedly still a natural assumption for traces. That is why, following Arnold, we focus on ideal sets of P-traces. These are defined according to the following partial order of P-traces.

Definition 3.6. *We say that a P-trace $t = (E_t, \prec_t, \xi_t)$ is a prefix of a P-trace $t' = (E_{t'}, \prec_{t'}, \xi_{t'})$ if the following conditions are satisfied:*
- $E_t \subseteq E_{t'}$;
- $\forall x \in E_t, \xi_t(x) = \xi_{t'}(x)$;
- $\prec_t = \prec_{t'} \cap (E_t \times E_t)$;
- $\forall x \in E_t,\ \forall y \in E_{t'}\colon y \prec_{t'} x \Rightarrow y \in E_t$.

Definition 3.7. *A set of P-traces over Σ is* ideal *if for all $t \in \mathcal{P}$, if t' is a prefix of t then $t' \in \mathcal{P}$. A complete, consistent and ideal set of P-traces will be called CCI for short.*

Useful consequence of [1, Prop. 3.1 and 3.3], our first result relates CCI sets of P-traces and stable right-congruences as follows.

Theorem 3.8. *An equivalence relation over $\Sigma^\star$ is the trace equivalence of a CCI set of P-traces if and only if it is a stable right-congruence (Def. 2.7).*

Thus any CCI set of P-traces describes a stable right-congruence and consequently it is associated to a uniquely determined full stable trace language (Prop. 2.8). Conversely, any (full) stable trace language can be associated to a CCI set of P-traces which is essentially unique up to the natural isomorphism notion defined as follows. We say that two sets of P-traces $\mathcal{P}_1$ and $\mathcal{P}_2$ are isomorphic if there is a bijection σ from $\mathcal{P}_1$ to $\mathcal{P}_2$ such that for all P-trace $t \in \mathcal{P}_1$, $\sigma(t)$ and t are isomorphic P-traces. Clearly, two CCI sets of P-traces are isomorphic iff their associated trace equivalences are equal. Thus, up to an isomorphism, Prop. 2.8 and Th. 3.8 show that *each stable trace language can be associated to the unique CCI set of P-traces which determines the same trace equivalence.*

Now the behaviors of stably concurrent automata are not full stable trace languages — except if one provide them with an additional sink state. Thus the traces of a stably concurrent automaton are described by a consistent and ideal

set of P-traces (it is complete only if its language is $\Sigma^\star$). This result completes actually some similar connections established independently in [3].

Regular CCI Sets of P-Traces vs Recognizable Languages. Theorem 3.8 and Proposition 2.8 establish a one-to-one correspondence between CCI sets of P-traces and full stable trace languages. We explain here that this relationship also holds between *recognizable* stable trace languages and *regular* CCI sets of P-traces. The latter were introduced by Arnold as follows.

Definition 3.9. *Let $\mathcal{P}$ be a CCI set of P-traces over a* finite alphabet Σ *and let $\sim$ be its associated trace equivalence. We consider the equivalence relation $\equiv$ over $\Sigma^\star$ such that $u \equiv v$ if $\forall w, w' \in \Sigma^\star, u.w \sim u.w' \Leftrightarrow v.w \sim v.w'$.*
The set $\mathcal{P}$ is called regular *if the equivalence $\equiv$ is of finite index.*

Using [3, Prop. 2.7], we can now complete Prop. 2.8 and Th. 3.8 as follows.

Proposition 3.10. *A CCI set of P-traces is regular if and only if its associated full stable trace language is recognizable.*

4 Stable Trace Languages vs Mazurkiewicz Ones

We now show how stable trace languages relate to Mazurkiewicz ones. We explain that stable trace languages form a true generalization of Mazurkiewicz trace languages through the particularly useful example of a Producer-Consumer system. However, any stable trace language may be regarded simply as a *labelled* Mazurkiewicz trace language. This will be formalized here by a notion of projections.

Projections of Local Trace Languages. We first recall the natural structure of local trace languages by morphisms introduced in [11].

Definition 4.1. *Let $\mathcal{L} = (\Sigma, I, L)$ and $\mathcal{L}' = (\Sigma', I', L')$ be two local trace languages. A* morphism λ *from $\mathcal{L}$ to $\mathcal{L}'$ is a map $\lambda : \Sigma \to \Sigma'$ such that*
- $\forall (u, p) \in I$, $(\lambda(u), \lambda(p)) \in I'$;
- $\forall (u, \{a, b\}) \in I$: $a \neq b \Rightarrow \lambda(a) \neq \lambda(b)$.

Note that if two distinct actions a and b are independent after u then their images should be independent after $\lambda(u)$ in order to respect concurrency: that is why we require that $\lambda(a) \neq \lambda(b)$. Clearly if u_1 and u_2 are trace equivalent according to I then $\lambda(u_1)$ and $\lambda(u_2)$ are trace equivalent according to I'.

In this paper, we introduce particular morphisms which insure several nice correspondences between the related local trace languages.

Definition 4.2. *A* projection *from $\mathcal{L} = (\Sigma, I, L)$ to $\mathcal{L}' = (\Sigma', I', L')$ is a morphism $\lambda : \mathcal{L} \to \mathcal{L}'$ whose underlying map $\lambda : \Sigma \to \Sigma'$ is onto and such that $\forall (u', p') \in I'$, $\exists! (u, p) \in I$, $\lambda(u) = u' \wedge \ \lambda(p) = p'$.*

We remark first that the trace equivalence is faithfully preserved and reflected by projections. Moreover there is a one-to-one correspondence between the traces of $\mathcal{L}$ and those of $\mathcal{L}'$.

LEMMA 4.3. *Let λ be a projection from $\mathcal{L} = (\Sigma, I, L)$ to $\mathcal{L}' = (\Sigma', I', L')$. Then $\lambda : \Sigma^\star \to \Sigma'^\star$ induces a bijection between L and L'. Moreover, for all u_1, u_2 in L, $u_1 \sim u_2 \Leftrightarrow \lambda(u_1) \sim \lambda(u_2)$.*

Therefore, projections of local trace languages should be regarded as simple and faithful labellings. If $\lambda : \mathcal{L} \to \mathcal{L}'$ is a projection then *we will say that $\mathcal{L}'$ is the image of $\mathcal{L}$ through the projection λ*. It is clear that the image of a recognizable local trace language through a projection is recognizable.

Projections of Mazurkiewicz Trace Languages. The connection between Mazurkiewicz trace languages and stable trace languages is first established by Theorem 4.4 below. It asserts that any stable trace language is the projection of a Mazurkiewicz trace language. This result can be established by means of known relationships between stably concurrent automata, prime event structures, and dI-domains [12,20] — at least if we assume that all alphabet is countable. However, a direct proof can be achieved without this assumption. It follows in fact the same basic idea since it relies on equivalences of prime intervals [16,17,11].

THEOREM 4.4. *A local trace language (over a possibly infinite alphabet) is stable if and only if it is the image of a Mazurkiewicz trace language through a projection.*

The connection between projections of Mazurkiewicz languages and stable languages expressed in the preceding theorem also applies to the subclasses of recognizable languages (over finite alphabets). *This very interesting result will be used in the two last sections of this paper.* It is a direct reformulation of Arnold's work [1, Th. 6.16] with the help of Prop. 3.10.

THEOREM 4.5. *A stable trace language is recognizable if and only if it is the image of a recognizable Mazurkiewicz trace language through a projection.*

The Producer-Consumer System. We are now interested by languages over *finite* alphabets. An open problem raised by Arnold [1] is to know whether each stable trace language over a finite alphabet is the image of a Mazurkiewicz trace language over a finite alphabet through a projection[1]. *We give a negative answer to this question through the example of a Producer-Consumer system.*

We consider the alphabet $\Sigma = \{p, c\}$ where p represents a production of one item and c a consumption. The language of the system describes all the possible sequences for which at each stage there may not be more consumptions than productions. Formally, $L = \{u \in \Sigma^\star \mid \forall v \leq u, |v|_p \geq |v|_c\}$. Thus p, pc, ppc and pcp are sequential executions of the system. We now want to model a possible independency between the producer and the consumer. Provided that there has been already enough items produced, the producer and the consumer can act simultaneously. For instance, $ppc \sim pcp$. This can be represented by the local independence relation I defined as follows:

[1] The question raised by Arnold dealt with CCI sets of P-traces, i.e. *full* stable trace languages. We leave it to the reader to adapt our counter-example accordingly.

– $(u, \emptyset) \in I \Leftrightarrow u \in L$;
– $(u, \{c\}) \in I \Leftrightarrow u \in L \wedge\ |u|_p \geq |u|_c + 1$;
– $(u, \{p\}) \in I \Leftrightarrow u \in L$;
– $(u, \{p, c\}) \in I \Leftrightarrow u \in L \wedge\ |u|_p \geq |u|_c + 1$.

Clearly, $\mathcal{L} = (\Sigma, I, L)$ is a stable trace language. We now prove by contradiction that $\mathcal{L}$ is *not* the image of a Mazurkiewicz trace language over a *finite* alphabet through a projection. Let us assume that $\mathcal{L}$ is the image of a Mazurkiewicz trace language $\mathcal{L}' = (\Sigma', I', L')$ over a finite independent alphabet $(\Sigma', \|')$ through a projection λ. Let n denote the size of Σ'. We consider the sequence $u = (p.c)^{n+1}$ consisting of $n+1$ productions and $n+1$ consumptions. There is a unique sequence $v \in L'$ such that $\lambda(v) = u$. Let us write $v = a_1.b_1.a_2.b_2...a_{n+1}.b_{n+1}$. Clearly, $\lambda(a_i) = p$ and $\lambda(b_i) = c$ for all $i \in [1, n+1]$. We easily check that for any $i \in [1, n]$, we have $b_i \not\| a_i$ and for all $j \in [i+1, n]$, $b_i \| a_j$. Now there are $i_1, i_2 \in [1, n]$ such that $i_1 < i_2$ and $b_{i_1} = b_{i_2}$ because $\mathrm{Card}(\Sigma') = n$. Hence $b_{i_1} \| a_{i_2} \not\| b_{i_2}$. Contradiction.

5 Distributed Implementation of Stable Trace Languages

In this section, we establish that each recognizable stable trace language admits a distributed implementation in the form of a finite bounded Petri net. According to Prop. 3.10, this result also holds for regular CCI sets of P-traces. Furthermore, this means that the labelled dI-domain of any finite stably concurrent automaton is also described by a finite bounded Petri net.

We consider here the classical model of Place/Transition nets.

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 $\mathcal{N} = (S, T, W, \mathrm{M}_{in})$, $\mathrm{Mar}_{\mathcal{N}}$ denotes the set of all markings of $\mathcal{N}$ that is to say functions $\mathrm{M} : S \to \mathbb{N}$; a step $p \in \wp_f(T)$ is *enabled* at $\mathrm{M} \in \mathrm{Mar}_{\mathcal{N}}$ if $\forall s \in S$, $\mathrm{M}(s) \geq \sum_{t \in p} W(s,t)$; in this case, we note $\mathrm{M}\,[p\rangle\,\mathrm{M}'$ where $\mathrm{M}'(s) = \mathrm{M}(s) + \sum_{t \in p}(W(t,s) - W(s,t))$ 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,..., $p_n \in \wp_f(T)$ such that $\mathrm{M}_0 = \mathrm{M}_{in}$ and $\forall k \in [1, n]$, $\mathrm{M}_{k-1}\,[p_k\rangle\,\mathrm{M}_k$. In that case, M_n is said *reachable*.

Definition 5.2. *A* labelled Petri net *is a structure* (S, T, W, M_{in}, ξ) *where* (S, T, W, M_{in}) *is a Petri net and* ξ *is a map from* T *to an alphabet* Σ *such that for all firing sequence* $M_{in} = M_0\,[p_1\rangle\,...M_{n-1}\,[p_n\rangle\,M_n$ *and all transitions* $t, t' \in T$*:*

$$M_n\,[\{t\}\rangle \wedge\ M_n\,[\{t'\}\rangle \wedge\ \xi(t) = \xi(t') \Rightarrow t = t'.$$

The restriction adopted for the labelling $\xi : T \to \Sigma$ insures that two transitions enabled by a common reachable marking correspond to two distinct actions. In other words, the labelling is deterministic.

Definition 5.3. *The local trace language associated to a labelled Petri net* $\mathcal{N} = (S, T, W, M_{in}, \xi)$ *is* $\mathfrak{t}(\mathcal{N}) = (\Sigma, I, L)$ *where* $I = \{(\xi(t_1...t_n), \xi(p)) \mid (t_1...t_n, p) \in T^\star \times \wp_f(T) \wedge\ M_{in}\,[\{t_1\}\rangle\, M_1 ...\, [\{t_n\}\rangle\, M_n\,[p\rangle\}$ *and the set of sequential executions is* $L = \{u \in \Sigma^\star \mid (u, \emptyset) \in I\}$.

Let us now focus on *finite* Petri nets — that is to say with a finite number of places and transitions — which are also *bounded*, which means that there are only a finite number of reachable markings. It is clear that local trace languages of such Petri nets are recognizable. Using Th. 4.5 and Zielonka's theorem [21] we can establish the converse property for stable trace languages. *Roughly*, the proof proceeds as follows. Given a recognizable stable trace $\mathcal{L}$, we consider a recognizable Mazurkiewicz trace language $\mathcal{L}_M = (\Sigma_M, I_M, L_M)$ over $(\Sigma_M, \|_M)$ and a projection $\lambda : \mathcal{L}_M \to \mathcal{L}$ by using Th. 4.5. Then Zielonka's theorem [21] yields an asynchronous automaton $\mathcal{A}$ over $(\Sigma_M, \|_M)$ recognizing L_M. We regard $\mathcal{A}$ as if all its states were final and describe its behavior by a (1-safe) Petri net labelled by ξ. Then the trace language of this Petri net *includes* $\mathcal{L}_M$. The technical point is then to add some places and to adapt the weight function in order to restrict the behavior of the net to L_M, *without affecting the independency of the transitions*. Finally, the labelling of the final net is changed into $\lambda \circ \xi$.

Theorem 5.4. *Any recognizable stable trace language is the local trace language of a finite bounded labelled Petri net.*

6 Asynchronous Transition Systems vs Trace Automata

Motivated by domain theoretic considerations, Schmitt established in [17] that any finite stable trace automaton is covered by a finite asynchronous transition system — which thus describes the same coherent dI-domain. We explain here how Theorem 4.5 provides a new approach to prove easily this result and yields an algorithm for the construction of such an asynchronous transition system.

Definition 6.1. *Let* $(\Sigma, \|)$ *be an independent alphabet. An* independent automaton *over* $(\Sigma, \|)$ *is a structure* $\mathcal{A} = (Q, s, \Sigma, \longrightarrow, \|)$ *where* Q *is a set of states, with initial state* $s \in Q$ *and* $\longrightarrow \subseteq Q \times \Sigma \times Q$ *is a transition relation such that* $q \xrightarrow{a} q_1 \wedge\ q \xrightarrow{a} q_2 \Rightarrow q_1 = q_2$.

A trace automaton *is an independent automaton which satisfies the* Forward Diamond *property* FD*:*

FD*:* $q \xrightarrow{a} q_1 \wedge\ q \xrightarrow{b} q_2 \wedge\ a \| b \Rightarrow \exists q_3 \in Q,\ q_2 \xrightarrow{a} q_3 \wedge\ q_1 \xrightarrow{b} q_3$.

An asynchronous transition system *over* $(\Sigma, \|)$ *is an independent automaton which satisfies* FD *and the* Independent Diamond *property* ID*:*

ID*:* $q \xrightarrow{a} q_1 \wedge\ q_1 \xrightarrow{b} q_2 \wedge\ a \| b \Rightarrow \exists q_3 \in Q,\ q_1 \xrightarrow{b} q_3 \wedge\ q_3 \xrightarrow{a} q_2$.

We shall assume in this paper that *all states of an independent automaton are reachable*[2]. We note that each trace automaton may be regarded as an automaton

[2] This means that $\forall q \in Q, \exists u \in \Sigma^\star, s \xrightarrow{u} q$.

with concurrency relation (Def. 2.3) for which $a\|_q b \Leftrightarrow a\|b \wedge\ q \xrightarrow{a} q' \wedge\ q \xrightarrow{b} q''$. It is clear that each asynchronous transition system, regarded as an automaton with concurrency relations, is in fact a stably concurrent automaton. That is not true for trace automata in general (only one implication is fulfilled). That is why Schmitt introduced *stable* trace automata as follows.

DEFINITION 6.2. *A* trace automaton $\mathcal{A} = (Q, s, \Sigma, \longrightarrow, \|)$ *is* stable *if for all states* $q, r \in Q$ *and for all actions* a, b *and* c *pairwise independent w.r.t.* $\|$*:*

$$\left[q \xrightarrow{abc} r \wedge\ q \xrightarrow{acb} r \wedge\ q \xrightarrow{bca} r\right] \Rightarrow \left[q \xrightarrow{cab} r \wedge\ q \xrightarrow{cba} r\right].$$

We remark here that a trace automaton is a stably concurrent automaton iff it is stable. Therefore any asynchronous transition system is a stable trace automaton. In order to strengthen this trivial relationship between stable trace automata and asynchronous transition systems, Schmitt used folding morphisms, which correspond somehow to projections.

DEFINITION 6.3. *Let* $\mathcal{A} = (Q, s, \Sigma, \longrightarrow)$ *and* $\mathcal{A}' = (Q', s', \Sigma', \longrightarrow')$ *be two trace automata. A* folding morphism *from* $\mathcal{A}$ *to* $\mathcal{A}'$ *is a pair of maps* $\sigma : Q \to Q'$ *and* $\lambda : \Sigma \to \Sigma'$ *such that*

- $\sigma(s) = s'$;
- $q_1 \xrightarrow{a} q_2 \Rightarrow \sigma(q_1) \xrightarrow{\lambda(a)} \sigma(q_2)$;
- $q_1 \xrightarrow{a} q_2 \wedge\ q_1 \xrightarrow{b} q_3 \wedge\ a \neq b \Rightarrow \lambda(a) \neq \lambda(b)$;
- $\sigma(q_1) \xrightarrow{a'} q_2' \Rightarrow \exists q_2 \in Q, \exists a \in \Sigma, q_1 \xrightarrow{a} q_2 \wedge\ \lambda(a) = a'$;
- $\forall q \in Q,\ q \xrightarrow{a} q' \wedge\ q \xrightarrow{b} q'' \Rightarrow [a\|b \Leftrightarrow \lambda(a)\|'\lambda(b)]$.

In that case, we say that $\mathcal{A}$ *covers* $\mathcal{A}'$.

We can now state the main result of [17].

THEOREM 6.4. *Any finite stable trace automaton is covered by some finite asynchronous transition system.*

Let us now present a new, simple and constructive proof of this result. Let $\mathcal{A} = (Q, s, \Sigma, \longrightarrow, \|)$ be a finite stable trace automaton. Viewed as a stably concurrent automaton, it describes a recognizable stable trace language $\mathcal{L} = (\Sigma, I, L)$. Applying Theorem 4.5, yields a recognizable Mazurkiewicz trace language $\mathcal{L} = (\Sigma_M, I_M, L_M)$ over an independent alphabet $(\Sigma_M, \|_M)$ and a projection $\lambda_M : \mathcal{L}_M \to \mathcal{L}$. We consider $\mathcal{A}_M = (Q_M, s_M, \Sigma_M, \longrightarrow_M, F_M)$ to be the minimal automaton of L_M, where F_M denotes the set of final states. Since L_M is recognizable and prefix-closed, $\mathcal{A}_M$ is finite and $F_M = Q_M$. We also remark that $\mathcal{A}_M$ *satisfies the Independent Diamond property* ID *w.r.t.* $\|_M$, because L_M is closed for the commutation of independent actions. We consider the synchronized product $\mathcal{A} \times \mathcal{A}_M = (Q \times Q_M, (s, s_M), \Sigma \times \Sigma_M, \longrightarrow_\times, \|_\times)$ where

$$(q, q_M) \xrightarrow{a,a_M}_\times (q', q'_M) \text{ iff } q \xrightarrow{a} q' \wedge\ q_M \xrightarrow{a_M} q'_M \wedge\ \lambda(a_M) = a$$

and $(a, a_M)\|_\times(b, b_M)$ iff $a\|b \wedge\ a_M\|_M b_M$. We easily check that $\mathcal{A} \times \mathcal{A}_M$ is a finite asynchronous transition system — once restricted to its reachable states. Moreover the pair $\sigma_1 : (q, q_M) \mapsto q$ and $\lambda_1 : (a, a_M) \mapsto a$ is a folding morphism from $\mathcal{A} \times \mathcal{A}_M$ to $\mathcal{A}$.

Let us stress finally that the construction of $\mathcal{A}_M$ from $\mathcal{A}$ is essentially provided by Arnold's proof of [1, Th. 6.16]. One can actually deduce from this proof some upper bounds for the sizes of Σ_M and Q_M (w.r.t. the sizes of Q and Σ). This is definitively impossible when following Schmitt's approach.

Acknowledgments The authors are grateful to A. Arnold and B. Rozoy for their help and their useful advices. The second author thanks M. Droste and D. Kuske for motivating discussions while preparing the last improvements of this paper.

References

1. Arnold A.: *An extension of the notion of traces and asynchronous automata.* Theoretical Informatics and Applications **25** (1991) 355–393
2. Bednarczyk M.: *Categories of asynchronous systems.* PhD thesis (University of Sussex, 1987)
3. Bracho F., Droste M., Kuske D.: *Representations of computations in concurrent automata by dependence orders.* TCS **174** (1997) 67–96
4. Diekert V., Rozenberg G.: *The Book of Traces.* (World Scientific, 1995)
5. Droste M.: *Concurrency, automata and domains.* LNCS **443** (1990) 195–208
6. Droste M., Shortt R.M.: *From Petri nets to automata with concurrency.* – Unpublished manuscript (1999) –
7. Ehrenfeucht A., Rozenberg G.: *Partial (Set) 2-structures.* Part II: State spaces of concurrent systems, Acta Informatica **27** (1990) 343–368
8. Hoogers P.W., Kleijn H.C.M., Thiagarajan P.S.: *A Trace Semantics for Petri Nets.* Information and Computation **117** (1995) 98–114
9. Husson J.-Fr.: *Modélisation de la causalité par des relations d'indépendances.* Thesis (Université Paul Sabatier de Toulouse, 1996)
10. Husson J.-Fr., Morin R.: *Relationships between Arnold's CCI sets of P-traces and Droste's stably concurrent automata.* Technical report MATH-AL-1-00 (Technische Universität Dresden, 2000)
11. Kleijn H.C.M., Morin R., Rozoy B.: *A General Categorical Connection between Local Event Structures and Local Traces.* FCT'99, LNCS **1684** (1999) 338–349
12. Kuske D.: *Nondeterministic automata with concurrency relations and domains.* CAAP'94, LNCS **787** (1994) 202–217
13. Mazurkiewicz A.: *Concurrent program schemes and their interpretations.* Aarhus University Publication (DAIMI PB-78, 1977)
14. Morin R., Rozoy B.: *On the Semantics of Place/Transition Nets.* Concur'99, LNCS **1664** (1999) 447–462
15. Mukund M.: *Petri Nets and Step Transition Systems.* International Journal of Foundations of Computer Science **3** (1992) 443–478
16. Nielsen M., Sassone V., Winskel G.: *Relationships between Models of Concurrency.* LNCS **803** (1994) 425–475
17. Schmitt V.: *Stable trace automata vs. full trace automata.* TCS **200** (1998) 45–100
18. Stark E.W.: *Connections between concrete and abstract model of concurrent systems.* LNCS **442** (1990) 53–79
19. Szpilrajn E.: *Sur l'extension de l'ordre partiel.* Fund. Math. **16** (1930) 386–389
20. Winskel G.: *Event structures.* LNCS **255** (1987) 325–392
21. Zielonka W.: *Notes on finite asynchronous automata.* Theoretical Informatics and Applications **21** (1987) 99–135