Theoretical Computer Science 135 (1994) 289–318
Elsevier

Labelled domains and automata with concurrency*

Felipe Bracho

IIMAS, Universidad Nacional Autónoma de México, A.P. 20-726, 01000 México D.F., Mexico

Manfred Droste**

Mathematik und Informatik, Universität GHS Essen, 45117 Essen, Germany

Communicated by M. Nivat
Received April 1993
Revised January 1994

Abstract

Bracho, F. and M. Droste, Labelled domains and automata with concurrency, Theoretical Computer Science 135 (1994) 289–318.

We investigate an operational model of concurrent systems, called automata with concurrency relations. These are labelled transition systems $\mathscr{A}$ in which the event set is endowed with a collection of binary concurrency relations which indicate when two events, in a particular state of the automaton, commute. This model generalizes asynchronous transition systems, and as in trace theory we obtain, through a permutation equivalence for computation sequences of $\mathscr{A}$, an induced domain $(D(\mathscr{A}), \leqslant)$. Here, we construct a categorical equivalence between a large category of ("cancellative") automata with concurrency relations and the associated domains. We show that each cancellative automaton can be reduced to a minimal cancellative automaton generating, up to isomorphism, the same domain. Furthermore, when fixing the event set, this minimal automaton is unique.

1. Introduction

In the study of programming languages like CCS [18] and CSP [12], labelled transition systems have been frequently used to give an operational semantics of concurrent processes. A labelled transition system may be defined to be a quadruple $(S, E, T, *)$ where S is a set of states, E is a set of events, $T \subseteq S \times E \times S$ is the transition

Correspondence to: F. Bracho, IIMAS, Universidad Nacional Autónoma de México, A.P. 20-726, 01000 México D.F., Mexico.

* Partially supported by UNESCO through the IIP and CREI.

** Present address: Mathematik, Technische Universität Dresden, 01062 Dresden, Germany.

SSDI 0304-3975(94)00030-M

relation, and $*\in S$ is the start state. Recently, several authors have considered labelled transition systems with an additional binary relation $\|$ on E incorporating direct information about concurrency. Then two transitions $t=(s,e,r)$, $t'=(s,e',r')$ are defined to "commute" whenever $e\|e'$. Such asynchronous transition systems, which generalize Mazurkiewicz traces [17], were investigated by Bednarczyk [2] and Shields [20]. Similar structures have been used to provide a semantics for CCS [4] and to model properties of computations in term rewriting systems, in the lambda calculus [1, 3, 13, 16] and in dataflow networks [22, 24]. For further background, we refer the reader to [26].

In the previous model, a single binary relation on E was used to represent the concurrency information for all pairs of transitions. Here, we investigate a more general model in which the concurrency information for two transitions depends not only on the two arriving events, but also on the present state of the transition system. Hence, we consider transition systems (automata) $\mathscr{A}=(S,E,T,*)$ together with a collection of binary concurrency relations $\|_s$ $(s\in S)$ on E which reflect when two events or actions commute in a particular state $s\in S$. Such *automata with concurrency relations* were introduced in [9, 10], where their domains of computation sequences were investigated, also, independently and in a slightly different form, in [14], where applications are given. They arise naturally, for instance, when considering the dynamic behaviour of place/transition nets with capacities in Petri net theory; see [11].

Similarly as for asynchronous transition systems and as in trace theory, the concurrency relations of the automata $\mathscr{A}$ induce a natural definition of permutation equivalence for (finite or infinite) computation sequences of $\mathscr{A}$; intuitively, two computation sequences are equivalent, if they represent "interleaved views" of a single computation (for the origins of this notion of equivalence, see [3, 13, 16]). Moreover, the set $D(\mathscr{A})$ of equivalence classes of computation sequences carries a nontrival partial order which is naturally induced by the prefix-ordering of computation sequences. In [9, 10], an order-theoretic characterization was given of all the partial orders $(D(\mathscr{A}), \leqslant)$ where $\mathscr{A}$ is an automaton with concurrency relations. These *weak concurrency domains* turned out to be closely related with event domains and dI-domains which arose in studies of denotational semantics of programming languages (cf. [7, 25]).

In this paper, we investigate categories of automata with concurrency relations and of weak concurrency domains. Let us say that an automaton with concurrency relations $\mathscr{A}$ *generates* a domain $(D, \leqslant)$, if $(D, \leqslant)$ is order-isomorphic to the domain $(D(\mathscr{A}), \leqslant)$. In general, a given domain $(D, \leqslant)$ may be generated by many (nonisomorphic) automata $\mathscr{A}$. This indicates that we should endow the domains $(D(\mathscr{A}), \leqslant)$ with more structure. In fact, as we will show, $\mathscr{A}$ induces on the compact elements and on the prime intervals of its domain $(D(\mathscr{A}), \leqslant)$ two labelling functions in a natural way. These take values in the state set S and the event set E of $\mathscr{A}$, respectively. This leads to a functor **D** from the category *Aut* of all automata with concurrency relations into the category *LDom* of all labelled weak concurrency domains. We show that this functor

induces, in fact, an equivalence between the large subcategory of *Aut* comprising all *cancellative* automata and the category of all *nicely labelled* weak concurrency domains. For precise definitions, see Section 2; we just note here that for cancellative automata $\mathscr{A}$ various natural notions of concurrency of events in $\mathscr{A}$ coincide.

An important tool for this result is the notion of a *reduction* of $\mathscr{A}$ to $\mathscr{A}'$; this is defined to be an epimorphism from $\mathscr{A}$ to $\mathscr{A}'$ reflecting the enabling and concurrency of events at states. We show that such a reduction of $\mathscr{A}$ to $\mathscr{A}'$ does not change (up to isomorphism) the induced domain $D(\mathscr{A})$ of computation sequences. As a consequence of our results we obtain a characterization of when a given weak concurrency domain $(D, \leqslant)$ can be generated by a finite cancellative automaton $\mathscr{A}$. We also obtain a characterization of all domains $(D, \leqslant)$ arising, in a similar way, from trace alphabets $(E, \|)$; such a characterization seemed to be open in trace theory.

Next we consider, for a given weak concurrency domain $(D, \leqslant)$, the class of all cancellative automata $\mathscr{A}$ generating $(D, \leqslant)$. We show that this class contains, with respect to reductions, a greatest automaton $\mathscr{A}_{\max}$; i.e. $\mathscr{A}_{\max}$ can be reduced to any other cancellative automaton generating $(D, \leqslant)$, and $\mathscr{A}_{\max}$ is unique up to isomorphism.

Similarly as in classical formal language theory, each cancellative automaton $\mathscr{A}$ generating $(D, \leqslant)$ can be reduced to a "minimal" such automaton $\mathscr{A}_{\min}$; here "minimal" means that any further reduction of $\mathscr{A}_{\min}$ is an isomorphism. However, here in general $\mathscr{A}_{\min}$ is not unique up to isomorphism. Somewhat surprisingly, however, if we consider only automata with (in a certain sense) a fixed event set, then this class of automata contains, with respect to state reductions, also a uniquely determined minimal automaton.

2. Automata and labelled domains

In this section, we will introduce automata with concurrency relations and their induced domains of concurrent computation sequences. These domains can be endowed, in a natural way, with two labelling functions, for events and states, respectively. Their properties will be shown to correspond to different versions of concurrency for events of the underlying automaton.

Definition 2.1. An *automaton with concurrency relations* is a quintuple $\mathscr{A} = (S, E, T, *, \|)$ where

(1) S and E are countable disjoint sets; $* \in S$ is a distinguished element;

(2) T is a subset of $S \times E \times S$ such that whenever $(s, e, s'), (s, e, s'') \in T$, then $s' = s''$; we require that for each $e \in E$ there are $s, s' \in S$ with $(s, e, s') \in T$;

(3) $\| = (\|_s)_{s \in S}$ is a family of irreflexive, symmetric binary relations on E; it is required that whenever $e_1 \|_s e_2$ $(e_1, e_2 \in E)$, there exist transitions (s, e_1, s_1), (s, e_2, s_2), (s_2, e_1, r) and (s_1, e_2, r) in T.

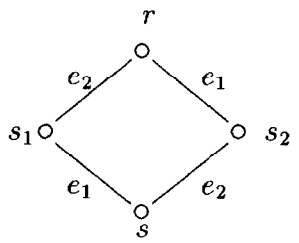

Fig. 1.

The elements of S are called *states*, the elements of E *events* and the elements of T *transitions*. Intuitively, a transition $t=(s, e, s')$ represents a potential computation step in which event e happens in state s of $\mathscr{A}$ and $\mathscr{A}$ changes from state s to s'. We write $ev(t)=e$, the event of t. The element $*$ is the *start state*. The *concurrency relations* $\|_s$ describe the concurrency information for pairs of events at state s. The last requirement can be seen as in Fig. 1.

In an automaton with concurrency relations $\mathscr{A}$ the events that concur at a state s do not have to bear upon those that concur in another state. In this general model, the concurrency relations $\|_s (s \in S)$ are thus viewed as being independent of each other. Later on we will impose additional restrictions on $\mathscr{A}$. A *finite computation sequence* in $\mathscr{A}$ is either empty (denoted by ε), or a finite sequence $u=t_1 \dots t_n$ of transitions $t_i \in T$ of the form $t_i=(s_{i-1}, e_i, s_i)$ for $i=1, \dots, n$; it can be depicted as

$$s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} s_n.$$

We call s_0 the *domain* of u, denoted $dom(u)$, and s_n the *codomain*, denoted $cod(u)$. Likewise, an infinite sequence $(t_i)_{i \in \mathbb{N}}$ of transitions $t_i=(s_{i-1}, e_i, s_i)$ $(i \in \mathbb{N})$ is called an *infinite computation sequence* of $\mathscr{A}$; its domain is s_0. A computation sequence is called *initial* if its domain is $*$, the start state. We let $CS(\mathscr{A})(CS^0(\mathscr{A}), CS_*(\mathscr{A}), CS_*^c(A))$ denote the sets, respectively, of all (all finite, all initial, all finite initial) computation sequences of $\mathscr{A}$. The *composition* uv of a finite computation sequence u and an arbitrary computation sequence v with $dom(v)=cod(u)$ is defined in the natural way by concatenating u and v. Formally, we put $u\varepsilon=\varepsilon u=u$. We call u a *prefix* of w if u is finite and $w=uv$ for some computation sequence v.

Now we want the concurrency relations of $\mathscr{A}$ to induce an equivalence relation on $CS(\mathscr{A})$, so that equivalent computation sequences are not differentiated by the order in which the concurrent events appear. For this we proceed as follows: we call two finite computation sequences $t=t_1 \dots t_n$ and $u=u_1 \dots u_n$ *strongly equivalent* if we obtain t from u by replacing for some $1 \leqslant i < n$, an occurrence $u_i u_{i+1}$ of the form $(s, a, q)(q, b, r)$ by $t_i t_{i+1}$ of the form $(s, b, p)(p, a, r)$ with $a \|_s b$ in $\mathscr{A}$. We then let $\sim$ be the reflexive and transitive closure of strong equivalence on $CS^0(\mathscr{A})$.

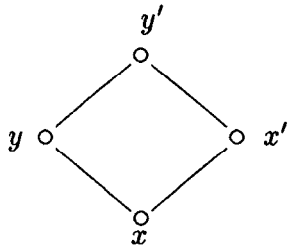

Fig. 2.

It easily follows from the above that $\sim$ is an equivalence relation and that any two equivalent sequences have the same length, domain and codomain. Also for any $a \in E$, the number of occurrences of a is the same in any two equivalent sequences.

The prefix relation together with $\sim$ induces a preorder on $CS^0(\mathscr{A})$ by letting $u \leqslant v$ iff $v \sim uw$ for some $w \in CS^0(\mathscr{A})$. This gives rise to a preorder in $CS(\mathscr{A})$ where for $u, v \in CS(\mathscr{A})$ we put $u \leqslant v$ if for every prefix u' of u there exists a prefix v' of v such that $u' \leqslant v'$. Then put $u \sim v$ iff $u \leqslant v$ and $v \leqslant u$, and let $[u]$ denote the equivalence class with respect to $\sim$. We order these classes by $[u] \leqslant [v]$ iff $u \leqslant v$. Then let $D(\mathscr{A}) = \{[u]: u \in CS_*(\mathscr{A})\}$ and call $(D(\mathscr{A}), \leqslant)$ the *domain of concurrent computation sequences associated with* $\mathscr{A}$. Given a partial order $(D, \leqslant)$, we say that $\mathscr{A}$ *generates* $(D, \leqslant)$ if $(D, \leqslant)$ is isomorphic to $(D(\mathscr{A}), \leqslant)$. It is proved in [10] that this is indeed a domain where the compact or finite elements are given by $D^0(\mathscr{A}) = \{[u]: u \in CS^0_*(\mathscr{A})\}$. We will later show that this construction induces a functor between the categories of automata and certain kinds of domains and functions. Let us now turn to the domains that are induced by the automata we have defined.

We introduce some notation. Let $(D, \leqslant)$ be a domain, i.e. an ω-algebraic complete partial order (c.p.o.). We denote the set of *compact* (=*isolated, finite*) elements of $(D, \leqslant)$ by D^0. We say that $(D, \leqslant)$ is *finitary*, if for all $x \in D^0$, the set $\{d \in D^0: d \leqslant x\}$ is finite. For $x, y \in D$ with $x < y$ we write $x \prec y$ if there is no $z \in D$ with $x < z < y$. We denote by $[x, y]$ such a pair, and we call it a *prime interval* of D.

Let Int_D denote the collection of all prime intervals of D. Also, for prime intervals $[x, y], [x', y']$ of D we put $[x, y] \frown [x', y']$ if $x' \neq y, x \prec x'$ and $y \prec y'$. This can be seen in Fig. 2.

We let $\succ\!\prec$ denote the smallest equivalence relation on Int_D containing $\frown$. For $[x, y] \in Int_D$, let $[x, y]_{\succ\!\prec}$ denote the equivalence class of $[x, y]$ with respect to $\succ\!\prec$.

Let $\mathscr{A} = (S, E, T, *, \|)$ be an automaton with concurrency relations. It is immediate from the definition of $(D(\mathscr{A}), \leqslant)$ that for $x, y \in D^0(\mathscr{A})$ we have that $x \prec y$ iff $x = [u]$ and $y = [ut]$ for some $u \in CS^0_*(\mathscr{A})$ and $t \in T$. Since the number of occurrences of any fixed event e in two equivalent finite computation sequences is the same and all $u \in x$ have the same codomain, it follows that if $x = [u'] = [u]$ and $y = [u't'] = [ut]$ then $t = t'$. So each prime interval $[x, y]$ in $D(\mathscr{A})$ determines a unique transition t. We will write $x \overset{e}{\prec} y$ when for some $u \in CS^0_*(\mathscr{A})$, $t \in T$ we have $x = [u]$, $y = [ut]$, and $ev(t) = e$.

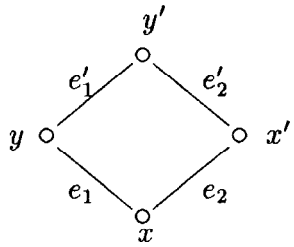

Fig. 3.

Since, conversely, t is uniquely determined by $dom(t)$ and $ev(t)$, we also obtain that $x \overset{e}{\prec} y$ and $x \overset{e}{\prec} y'$ implying $y = y'$.

Suppose now that $[x, y] \prec [x', y']$ in $(D(\mathscr{A}), \leqslant)$. Let $x = [u]$, $y = [ut_1]$ and $x' = [ut_2]$ with $t_1, t_2 \in T$. Since $y \prec y'$ and $x' \prec y'$, we have $y' = [ut_1 t_1'] = [ut_2 t_2']$ with $t_1', t_2' \in T$. Let $e_i = ev(t_i)$, $e_i' = ev(t_i')$ $(i = 1, 2)$. Then we obtain the following diagram (Fig. 3).

Since the number of occurrences of e_1 (respectively e_2) in $ut_1 t_1'$ is the same as in $ut_2 t_2'$ and $e_1 \neq e_2$ by $y \neq x'$, we obtain $e_1' = e_2$ and $e_2' = e_1$. Hence, $x \overset{e}{\prec} y$ and $[x, y] \prec [x', y']$ imply $x' \overset{e}{\prec} y'$. We summarize these observations in the following lemma which will be used subsequently very often without mentioning it explicitly again.

Lemma 2.2. *Let $\mathscr{A} = (S, E, T, *, \|)$ be an automaton with concurrency relations. Let $x, y, z, x', y' \in D^0(\mathscr{A})$ and $e, e' \in E$.*

(a) *If $x \overset{e}{\prec} y$ and $x \overset{e'}{\prec} z$, then $y = z$ iff $e = e'$.*

(b) *$[x, y] \succ\!\prec [x', y']$ and $x \overset{e}{\prec} y$ imply $x' \overset{e}{\prec} y'$.*

Let $(D, \leqslant)$ again be a finitary domain, and let $x, y \in D^\circ$ with $x < y$. A sequence $(x_i)_{i=0}^n$ in D^0 of the form $x = x_0 \prec x_1 \prec \cdots \prec x_n = y$ will be called a *covering chain* from x to y. Such a covering chain will be said to be *strongly equivalent* to any other covering chain from x to y of the form $x = x_0 \prec \cdots \prec x_i \prec z \prec x_{i+2} \prec \cdots \prec x_n = y$ with $z \neq x_{i+1}$, for some $0 \leqslant i \leqslant n-2$. We define *equivalence* on the set of all covering chains in D as the reflexive and transitive closure of strong equivalence.

Definition 2.3 ([Droste [10]]). A finitary domain $(D, \leqslant)$ is called a *weak concurrency domain*, if it satisfies the following two conditions for any $x, y, z \in D^0$:

(R) If $x \prec y$, $x \prec z$ and $[x, y] \succ\!\prec [x, z]$, then $y = z$.

(E) Any two covering chains from $\perp$ to x are equivalent.

Let $\mathscr{A}$ be an automaton with concurrency relations. Lemma 2.2 shows that $(D(\mathscr{A}), \leqslant)$ satisfies axiom (R). Besides, any finite initial computation sequence $t_1 \ldots t_n$

with $ev(t_i)=e_i$ gives rise to a covering chain $\perp = x_0 \overset{e_1}{\prec} x_1 \overset{e_2}{\prec} \cdots \overset{e_n}{\prec} x_n$ in $D(\mathscr{A})$ from $\perp$ to x_n, where $x_i=[t_1 \ldots t_i]$ $(i=0, \ldots, n)$. It is easy to see that this correspondence preserves strong equivalence, thus also equivalence; hence $(D(\mathscr{A}), \leqslant)$ satisfies axiom (E). So, $(D(\mathscr{A}), \leqslant)$ is a weak concurrency domain. Conversely, each weak concurrency domain $(D, \leqslant)$ is generated by some automaton $\mathscr{A}$ with concurrency relations [10]. Here, in general $\mathscr{A}$ is not determined uniquely by $(D, \leqslant)$. However, as will be seen, the class of all automata $\mathscr{A}$ generating a fixed weak concurrency domain $(D, \leqslant)$ can be well structured. Any domain $(D(\mathscr{A}), \leqslant)$ carries a natural labelling of its prime intervals $[x, y]$, as indicated by Lemma 2.2. This motivates the following definition.

Definition 2.4. Let $(D, \leqslant)$ be a finitary domain and E a set. A mapping $l_E: Int_D \to E$ will be called an *event-labelling function* if it is onto and satisfies the following two conditions for any $x, y, x', y', z \in D^0$:

(1) $[x, y] \succ\!\prec [x', y']$ implies $l_E([x, y])=l_E([x', y'])$.

(2) $l_E([x, y])=l_E([x, z])$ implies $y=z$.

Then $(D, \leqslant, l_E)$ will be called an *event-labelled domain.* In this case, for $x, y \in D^0$, we will write $x \overset{e}{\prec} y$ to denote that $x \prec y$ and $l_E([x, y])=e$.

Note that any event-labelled domain satisfies axiom (R). It is useful to see how event-labelling functions correspond to particular congruences of $(D, \leqslant)$ (this correspondence will be exploited in Section 4). Let $(D, \leqslant, l_E)$ be an event-labelled domain. Defining $[x, y] \equiv_E [x', y']$ iff $l_E([x, y])=l_E([x', y'])$, we obtain an equivalence relation $\equiv_E$ on Int_D satisfying

$$\succ\!\prec \subseteq \equiv_E \quad \text{and} \quad \text{if } [x, y] \equiv_E [x, z] \text{ then } y=z. \tag{$*$}$$

Any equivalence relation $\equiv$ on Int_D satisfying $(*)$ will be called an *event-congruence* on $(D, \leqslant)$. We say that $\equiv_E$ is the event-congruence corresponding to the event-labelling function l_E. Conversely, given an event-congruence $\equiv$ on $(D, \leqslant)$, let E comprise all $\equiv$-equivalence classes of Int_D, and let $l_E: Int_D \to E$ map each prime interval onto its $\equiv$-equivalence class. Then l_E is an event-labelling function whose corresponding event-congruence coincides with $\equiv$. In particular, if here $\equiv$ is equal to $\succ\!\prec$, the corresponding event-labelling function l_E (with $l_E([x, y])=[x, y]_{\succ\!\prec}$) will be called the *canonical* event-labelling function. If $(D, \leqslant)=(D(\mathscr{A}), \leqslant)$ for some automaton $\mathscr{A}$ with concurrency relations, we can also define a function $l_E: Int_{D(\mathscr{A})} \to E$ by putting $l_E([x, y])=e$ if $x=[u]$, $y=[ut]$ for some $ut \in CS^0_*(\mathscr{A})$ with $t \in T$ and $ev(t)=e$. If each transition of $\mathscr{A}$ occurs in some initial computation sequence, then l_E is onto, and by Lemma 2.2, l_E is an event-labelling function on $(D(\mathscr{A}), \leqslant)$. We call l_E the *induced* event-labelling function, and $(D(\mathscr{A}), \leqslant l_E)$ the *induced* event-labelled domain of $\mathscr{A}$.

Given an automaton $\mathscr{A}=(S, E, T, *, \|)$ with concurrency relations, the correspondence $[u] \mapsto cod(u)$ $(u \in CS^0_*(\mathscr{A}))$ defines a function from $D^0(\mathscr{A})$ into S, which we also denote, simply, by *cod*. Similarly as before for events, its properties motivate the following definition.

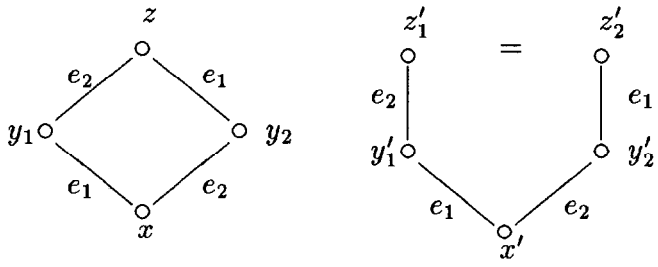

Fig. 4.

Definition 2.5. Let $(D, \leqslant, l_E)$ be an event-labelled domain and S a set. A mapping $l_S: D^0 \to S$ will be called a *state-labelling function* if it is onto and satisfies the following condition:

(1) Whenever $x, y, x' \in D^0$ with $x \overset{e}{\prec} y$ and $l_S(x) = l_S(x')$, then there is $y' \in D^0$ with $x' \overset{e}{\prec} y'$ and $l_S(y') = l_S(y)$.

Then $(D, \leqslant, l_E, l_S)$ will be called a *labelled domain*. Moreover, it will be said to be *nicely labelled* (and l_S is a *nice* state-labelling function) if the following condition is satisfied

(2) Whenever $x, y_i, z, x', y_i', z_i' \in D^0$ with $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ and $x' \overset{e_i}{\prec} y_i' \overset{e_{3-i}}{\prec} z_i'$ for $i = 1, 2$, $y_1 \neq y_2$ and $l_S(x) = l_S(x')$, then $z_1' = z_2'$.

Diagramatically this is illustrated in Fig. 4.

Clearly, for any event-labelled domain $(D, \leqslant, l_E)$, the identity mapping id_{D^0} on D^0 is a nice state-labelling of $(D, \leqslant, l_E)$, called the *trivial* state-labelling function.

Let l_S be an arbitrary state-labelling function of $(D, \leqslant, l_E)$, and let $\equiv_E$ be the event-congruence on $(D, \leqslant)$ corresponding to l_E. Then the equivalence relation $\equiv_S$ on D^0, defined by: $x \equiv_S y$ iff $l_S(x) = l_S(y)$, satisfies the following condition:

(c1) Whenever $x, y, x' \in D^0$ with $x \prec y$ and $x \equiv_S x'$, then there is $y' \in D^0$ with $[x, y] \equiv_E [x', y']$ and $y \equiv_S y'$.

Moreover, l_S is a nice state-labelling iff (c1) and (c2) hold:

(c2) Whenever $x, y_i, z, x', y_i', z_i' \in D^0$ with $x \prec y_i \prec z$, $x' \prec y_i' \prec z_i'$, $[x, y_i] \equiv_E [x', y_i']$, $[y_i, z] \equiv_E [y_i' z_i']$ for $i = 1, 2$, $y_1 \neq y_2$ and $x \equiv_S x'$, then $z_1' = z_2'$.

Any equivalence relation $\equiv$ on D^0 satisfying (c1) (resp. (c1) and (c2)) will be called a *state-congruence* (resp., *nice state-congruence*) for $(D, \leqslant, l_E)$. We say that $\equiv_S$ is the state-congruence corresponding to the state-labelling function l_S. Conversely, given a state-congruence $\equiv$ for $(D, \leqslant, l_E)$, we obtain a state-labelling function l_S by mapping each element of D^0 onto its $\equiv$-equivalence class; trivially, the state-congruence corresponding to l_S coincides with $\equiv$.

Now let $\mathscr{A}$ be an automaton with concurrency relations in which each state is reachable, and let l_E be the induced event-labelling of $(D(\mathscr{A}), \leqslant)$. Then $cod: D^0(\mathscr{A}) \to S$, as defined before, is a state-labelling of $(D(\mathscr{A}), \leqslant, l_E)$. For, if $x, y, x' \in D^0(\mathscr{A})$ with $x \overset{e}{\prec} y$ and $cod(x) = cod(x')$, then for some $u, u' \in CS^0_*(\mathscr{A})$ and $t \in T$,

we have $x=[u]$, $y=[ut]$, $ev(t)=e$, $x'=[u']$ and $cod(u)=cod(u')$; so $y'=[u't]\in D^0(\mathscr{A})$, $x' \overset{e}{\prec} y'$ and $cod(y')=cod(y)$. We put $\boldsymbol{D}(\mathscr{A})=(D(\mathscr{A}), \leqslant, l_E, cod)$, and we call $\boldsymbol{D}(\mathscr{A})$ the *induced labelled domain of* $\mathscr{A}$. In general, $\boldsymbol{D}(\mathscr{A})$ is not nicely labelled. This is closely related with two versions of concurrency of events in states, which we now introduce.

Definition 2.6. Let $(D, \leqslant, l_E, l_S)$ be a labelled domain, and let $e_1, e_2 \in E$ with $e_1 \neq e_2$, and $s \in S$.

(a) We say that e_1 and e_2 are *observably concurrent* at s, if whenever $x\in D^0$ with $l_S(x)=s$, then there exist $y_1, y_2, z\in D^0$ with $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ for $i=1, 2$.

(b) We say that e_1 and e_2 are *weakly observably concurrent* at s, if there exist x, $y_1, y_2, z\in D^0$ with $l_S(x)=s$ and $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ for $i=1, 2$.

For motivation, assume that $(D, \leqslant, l_E, l_S)=\boldsymbol{D}(\mathscr{A})$ for some automaton $\mathscr{A}$. Let $t_i=(s, e_i, r_i)$ and $t_i'=(r_i, e_{3-i}, r)\in T$ for $i=1, 2$. Then e_1 and e_2 are observably concurrent at s, if for *any* finite initial computation sequence u of $\mathscr{A}$ with codomain s, we have that ut_1t_1' and ut_2t_2' are equivalent. Also, e_1 and e_2 are weakly observably concurrent at s, if for *some* $u\in CS_*^0(\mathscr{A})$ with codomain s, ut_1t_1' and ut_2t_2' are equivalent. Trivially, if two events e_1, e_2 are observably concurrent at a reachable state s, then they are weakly observably concurrent. For the converse we note the following proposition.

Proposition 2.7. *Let $\mathscr{D}=(D, \leqslant, l_E, l_S)$ be a labelled domain. Then the following are equivalent:*

(1) *$\mathscr{D}$ is nicely labelled.*

(2) *Whenever $e_1, e_2\in E$ are weakly observably concurrent at $s\in S$, then they are observably concurrent at s.*

Proof. (1)⇒(2): Let $x'\in D^0$ with $l_S(x')=s$. There are $x, y_1, y_2, z\in D^0$ with $l_S(x)=s$ and $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ for $i=1, 2$. Hence, for $i=1, 2$, there exists $y_i'\in D^0$ with $x' \overset{e_i}{\prec} y_i'$ and $l_S(y_i')=l_S(y_i)$ and then $z_i\in D^0$ with $y_i' \overset{e_{3-i}}{\prec} z_i$. As $\mathscr{D}$ is nicely labelled, we have $z_1=z_2$. So e_1, e_2 are observably concurrent at s.

(2)⇒(1): Let $x, y_i, z, x', y_i', z_i'\in D^0$ as in condition (2) of Definition 2.5. We claim that $z_1'=z_2'$. Clearly, e_1, e_2 are weakly observably concurrent at $s=l_S(x)$, hence observably concurrent. So there are $y_i^*, z^*\in D^0$ with $x' \overset{e_i}{\prec} y_i^* \overset{e_{3-i}}{\prec} z^*$ for $i=1, 2$. Then $y_i^*=y_i'$ $(i=1, 2)$ and so $z_1'=z^*=z_2'$ as claimed. □

Let $\mathscr{A}$ be an automaton with concurrency relation. Clearly, if $e_1 \|_s e_2$, then in $\boldsymbol{D}(\mathscr{A})$, e_1 and e_2 are observably concurrent at s. Hence, in general we have the following implications for any two events at a given reachable state:

concurrency ⇒ observable concurrency ⇒ weak observable concurrency.

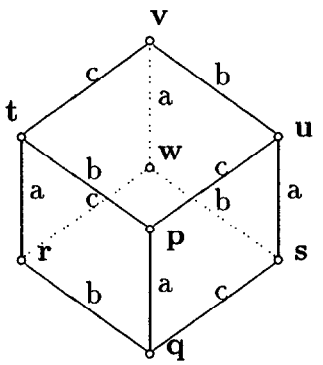

Fig. 5.

We will obtain our main results for automata in which these three notions of concurrency coincide.

Definition 2.8. Let $\mathscr{A}$ be an automaton with concurrency relations in which each state is reachable.

(a) $\mathscr{A}$ has *observable concurrency*, if in its induced labelled domain $\boldsymbol{D}(\mathscr{A})$, any two events are observably concurrent at a state iff they are weakly observably concurrent.

(b) $\mathscr{A}$ is *cancellative*, if whenever $e_1, e_2 \in E$ are weakly observably concurrent in $\boldsymbol{D}(\mathscr{A})$ at $s \in S$, then $e_1 \|_s e_2$.

(c) ([9]) $\mathscr{A}$ is *concurrent*, if whenever $(q, a, p), (q, b, r), (q, c, s) \in T$ are such that $a \|_q b$, $a \|_q c$ and $b \|_p c$, then also $b \|_q c$, $a \|_r c$ and $a \|_s b$.

Observe that by Proposition 2.7, $\mathscr{A}$ has observable concurrency iff its induced labelled domain $\boldsymbol{D}(\mathscr{A})$ is nicely labelled. Note that $\mathscr{A}$ is cancellative iff for any finite initial computation sequences ut_1t_2, $ut_1't_2'$ with $t_i, t_i' \in T$ $(i=1,2)$, $ut_1t_2 \sim ut_1't_2'$ implies $t_1t_2 \sim t_1't_2'$, which explains the name. The requirement for concurrent automata can be illustrated by Fig. 5. The black lines indicate the transitions which exist by the assumption that $a \|_q b$, $a \|_q c$, $b \|_p c$. The dotted lines indicate the transitions which are forced to exist by the requirements: $b \|_q c$, $a \|_r c$ and $a \|_s b$.

Concurrent automata were studied in detail in [10]. They correspond in a precise sense to automata with residual operations which were investigated by Stark [21], Panangaden and Stark [19] and Bachmann and Dung [1]. They occur naturally in λ-calculus, networks of communicating processes, reductions in nondeterministic term rewriting; see [16, 21, 1, 13].

Obviously, if $\mathscr{A}$ is cancellative, then it has observable concurrency. The following is less obvious.

Proposition 2.9. *Let $\mathscr{A}$ be a concurrent automaton. Then $\mathscr{A}$ is cancellative.*

Proof. By [10, (3.3)], $\mathscr{A}$ can be transformed into an "automaton with residual operation". By [23], these automata are cancellative. □

It will be shown below that any weak concurrency domain can be generated by a cancellative automaton. Hence, with respect to the class of generated domains, the assumption that the underlying automata are cancellative is no essential restriction. However, concurrent automata generate more specific domains, as shown in [9, 10]. A domain $(D, \leqslant)$ is a Scott-domain, if each nonempty subset of D which has an upper bound in D has a supremum in D. A finitary Scott-domain $(D, \leqslant)$ is called a *concurrency domain*, if it satisfies condition (R) (see Definition 2.3) and (C) for any $x, y, z \in D^0$:

(C) Whenever $x \prec\!\!\!\cdot\, y$, $x \prec\!\!\!\cdot\, z$, $y \neq z$ and $\{y, z\}$ has an upper bound in D, then $y \prec\!\!\!\cdot\, y \sqcup z$ and $z \prec\!\!\!\cdot\, y \sqcup z$.

Any such domain also satisfies condition (E) of Definition 2.3 and as shown in [10], concurrent automata generate precisely the concurrency domains.

3. Categories of automata and labelled domains

We will introduce morphisms between automata and between labelled domains and then construct a functorial equivalence between the categories of cancellative automata and of nicely labelled domains. First let us define morphisms between automata.

Definition 3.1. Let $\mathscr{A} = (S, E, T, *, \|)$ and $\mathscr{A}' = (S', E', T', *', \|')$ be two automata with concurrency relations, and let $f: S \to S'$, $g: E \to E'$ be functions. The pair (f, g) is called a *morphism* from $\mathscr{A}$ to $\mathscr{A}'$, if the following conditions are satisfied:

(1) whenever $(s, e, r) \in T$, then $(f(s), g(e), f(r)) \in T'$;

(2) whenever $e \|_s e'$ and $g(e) \neq g(e')$, then $g(e) \|'_{f(s)} g(e')$;

(3) $f(*) = *'$.

Thus, a morphism preserves states, events, transitions, the start state and maps concurrent or equal events of $\mathscr{A}$ to concurrent or equal events of $\mathscr{A}'$. A slightly restricted version of these morphisms (obtained by deleting the assumption that $g(e) \neq g(e')$ in condition (2)) and the interplay between the induced category of automata with concurrency relations and a category of Petri nets were studied, e.g., in [11].

Clearly morphisms compose and (id_S, id_E) is the identity morphism. We let *Aut* denote the category of all automata with concurrency relations in which each state is reachable, and morphisms as above as arrows, and we let *CAut* denote its full subcategory comprising all cancellative automata.

Next we define the morphisms between labelled domains.

Definition 3.2. (a) Let $(D, \leqslant, l_E)$, $(D', \leqslant, l_{E'})$ be two event-labelled domains. A pair (φ, g) of functions $\varphi: D \to D'$, $g: E \to E'$ will be called an *event-morphism* from $(D, \leqslant, l_E)$ to $(D', \leqslant, l_{E'})$, if it satisfies the following conditions:

(1) $\varphi: (D, \leqslant) \to (D', \leqslant)$ is continuous, $\varphi(D^0) \subseteq D'^0$ and $\varphi(\perp) = \perp'$;

(2) whenever $x, y \in D^0$ with $x \overset{e}{\prec} y$, then $\varphi(x) \overset{g(e)}{\prec} \varphi(y)$, i.e. $l_{E'}([\varphi(x), \varphi(y)]) = g \circ l_E([x, y])$.

(b) Let $\mathscr{D} = (D, \leqslant, l_E, l_S)$, $\mathscr{D}' = (D', \leqslant, l_{E'}, l_{S'})$ be two labelled domains, $(\varphi, g): (D, \leqslant, l_E) \to (D', \leqslant, l_{E'})$ an event-morphism and $f: S \to S'$ a function such that $l_{S'} \circ \varphi|_{D^0} = f \circ l_S$. Then the triple (φ, f, g) will be called a *morphism* from $\mathscr{D}$ to $\mathscr{D}'$. If here φ is an order-isomorphism, we call (φ, f, g) (and also (φ, g)) an *order-morphism*. If moreover, f and g (resp. g) are bijective, we call (φ, f, g) (resp. (φ, g)) an *isomorphism*.

We let *LDom* denote the category of all labelled weak concurrency domains, with morphisms (φ, f, g) as arrows. Also, let *NLDom* denote its full subcategory consisting of all nicely labelled weak concurrency domains.

It is easy to see that (φ, f, g) is a morphism from $\mathscr{D}$ to $\mathscr{D}'$ iff φ satisfies condition (1) above and preserves the event-congruences and state-congruences corresponding to the labelling functions of $\mathscr{D}$ and $\mathscr{D}'$. First we observe that for any morphism (φ, f, g), φ and g determine each other uniquely, and each of them determines f.

Proposition 3.3. *Let $\mathscr{D} = (D, \leqslant, l_E, l_S)$, $\mathscr{D}' = (D', \leqslant l_{E'}, l_{S'})$ be two labelled domains, and let $(\varphi, f, g), (\varphi', f', g'): \mathscr{D} \to \mathscr{D}'$ be two morphisms. Then $\varphi = \varphi'$ iff $g = g'$, and in this case also $f = f'$.*

Proof. Clearly, $\varphi = \varphi'$ implies $g = g'$ and $f = f'$, as l_E, l_S are onto. Now assume $g = g'$; we claim that $\varphi = \varphi'$. It suffices to show that $\varphi(x) = \varphi'(x)$ for each $x \in D^0$. We proceed by induction on the height of x, where the *height* of x is the smallest length of a covering chain from $\perp$ to x. Clearly $\varphi(\perp) = \perp'$.

Now assume that $\varphi(x) = \varphi'(x)$ for all $x \in D^0$ with height $\leqslant n$ and let $y \in D^0$ have height $n+1$. Choose $x \in D^0$ with height n and $x \overset{e}{\prec} y$. Then $\varphi(x) \overset{g(e)}{\prec} \varphi(y)$, and $\varphi(x) = \varphi'(x) \overset{g(e)}{\prec} \varphi'(y)$. Hence $\varphi(y) = \varphi'(y)$. □

Now let $\mathscr{A}$, $\mathscr{A}'$ be two automata with concurrency relations, let $\boldsymbol{D}(\mathscr{A}) = (D(\mathscr{A}), \leqslant, l_E, l_S)$, $\boldsymbol{D}(\mathscr{A}') = (D(\mathscr{A}'), \leqslant, l_{E'}, l_{S'})$ be the induced labelled domains, and let $(f, g): \mathscr{A} \to \mathscr{A}'$ be a morphism. We define a mapping $\varphi: D(\mathscr{A}) \to D(\mathscr{A}')$ as follows. If $x = [u] \in D(\mathscr{A})$ where $u = (s_0 \overset{e_1}{\to} s_1 \overset{e_2}{\to} \cdots)$ is a finite or infinite initial computation sequence of $\mathscr{A}$, let $u' = (f(s_0) \overset{g(e_1)}{\to} f(s_1) \overset{g(e_2)}{\to} \cdots)$. We say that u *induces* u' through (f, g), and we put $\varphi(x) = [u']$. Since (f, g) is a morphism, it follows easily that φ is well-defined (irrespective of the particular choice of u). We call (φ, f, g) the *induced morphism* from $\boldsymbol{D}(\mathscr{A})$ to $\boldsymbol{D}(\mathscr{A}')$, and we put $\boldsymbol{D}(f, g) := (\varphi, f, g)$. We have to check the following.

Proposition 3.4. $\boldsymbol{D}: Aut \to LDom$ is a functor. Moreover, if $\mathscr{A} \in Aut$ has observable concurrency, then $\boldsymbol{D}(\mathscr{A})$ is nicely labelled.

Proof. Let $(f, g): \mathscr{A} \to \mathscr{A}'$ be a morphism in *Aut*, and let $(\varphi, f, g): \boldsymbol{D}(\mathscr{A}) \to \mathbf{D}(\mathscr{A}')$ be the induced morphism in *LDom*. We first check that (φ, f, g) is a morphism in *LDom*. It is easy to see that φ is well-defined. If $u = (s_0 \overset{e_1}{\to} s_1 \overset{e_2}{\to} \cdots)$ is any infinite initial computation sequence of $\mathscr{A}$ and $u' = (f(s_0) \overset{g(e_1)}{\to} f(s_1) \overset{g(e_2)}{\to} \cdots)$, let $u_n = (s_0 \overset{e_1}{\to} s_1 \to \cdots \overset{e_n}{\to} s_n)$ and $u'_n = (f(s_0) \overset{g(e_1)}{\to} f(s_1) \to \cdots \overset{g(e_n)}{\to} f(s_n))$ $(n \in \omega)$. Then $[u] = \bigsqcup_{n \in \omega} [u_n]$ in $\boldsymbol{D}(\mathscr{A})$ and $[u'] = \bigsqcup_{n \in \omega} [u'_n]$, that is $\varphi([u]) = \bigsqcup_{n \in \omega} \varphi([u_n])$, in $D(\mathscr{A}')$. From this, it follows by a general domain-theoretic argument that φ is continuous.

Now if $x = [u]$, $y = [u(s, e, s')]$ for some $u \in CS^0_*(\mathscr{A})$ and $(s, e, s') \in T$, and $\varphi(x) = [u']$, then $\varphi(y) = [u'(f(s), g(e), f(s'))]$, so $\varphi(x) \overset{g(e)}{\prec} \varphi(y)$. Also, $cod(u') = f(cod(u))$. Hence (φ, f, g) is a morphism. Clearly $\boldsymbol{D}$ respects compositions and identities. The final statement is clear by Proposition 2.7. □

Next we introduce particular kinds of morphisms between automata which will be very useful in what follows. If $\mathscr{A}$ is an automaton, we say that an event $e \in E$ is *enabled* at a state $s \in S$, if there is a transition (s, e, r) in T.

Definition 3.5. Let $\mathscr{A} = (S, E, T, *, \|)$, $\mathscr{A}' = (S', E', T', *', \|')$ be two automata with concurrency relations. A morphism $(f, g): \mathscr{A} \to \mathscr{A}'$ is called a *reduction* of $\mathscr{A}$ to $\mathscr{A}'$, if the following conditions are satisfied:

(1) $f: S \to S'$ is onto;

(2) whenever $s \in S$ and $e' \in E'$ is enabled in $f(s)$, then there exists $e \in E$ which is enabled in s with $g(e) = e'$;

(3) whenever $s \in S$ and $e'_1, e'_2 \in E'$ with $e'_1 \|'_{f(s)} e'_2$, then there are $e_1, e_2 \in E$ with $e_1 \|_s e_2$ and $g(e_i) = e'_i$ $(i = 1, 2)$;

(4) g is *locally injective*, i.e. whenever $e_1, e_2 \in E$ are enabled in $s \in S$ and $e_1 \neq e_2$, then $g(e_1) \neq g(e_2)$.

Condition (2) says that g maps $\mathrm{En}_E(s) = \{e \in E : e \text{ is enabled in } s\}$ onto $En_{E'}(f(s))$ for each $s \in S$, in particular T onto T'.

We note that if $f: S \to S'$ is onto, then condition (2) implies that $g: E \to E'$ is onto. For, if $e' \in E'$, choose a transition $(s', e', r') \in T'$ and $s \in S$ with $f(s) = s'$, then $e' = g(e)$ for some $e \in E$. Condition (3) says that g maps $\|_s$ onto $\|'_{f(s)}$ $(s \in S)$, which combined with the assumption of local injectivity of g tells us that for any $e_1, e_2 \in E$ enabled in s we have that $e_1 \|_s e_2$ iff $g(e_1) \|'_{f(s)} g(e_2)$. Observe that g may be locally injective without being injective. Now we will show that if there is a reduction from $\mathscr{A}$ to $\mathscr{A}'$, then $\mathscr{A}$ and $\mathscr{A}'$ generate the same domain. More precisely, we have the following result.

Theorem 3.6. *Let $\mathscr{A}$, $\mathscr{A}'$ be two automata with concurrency relations in which all states are reachable, and let $(f, g): \mathscr{A} \to \mathscr{A}'$ be a reduction. Then the induced morphism $(\varphi, f, g): \boldsymbol{D}(\mathscr{A}) \to \boldsymbol{D}(\mathscr{A}')$ is an order-morphism.*

Proof. By Definition 3.2, it suffices to show that φ is an order isomorphism. Let x, $y \in D^0(\mathscr{A})$. Clearly, $x \leqslant y$ implies $\varphi(x) \leqslant \varphi(y)$. So assume now that $\varphi(x) \leqslant \varphi(y)$. We claim that $x \leqslant y$. We may assume that $x = [u]$, $y = [v]$ where $u, v \in CS^0_*(\mathscr{A})$ have the form

$$u = \left(s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \to \cdots \xrightarrow{e_n} s_n \right) \quad \text{and} \quad v = \left(r_0 \xrightarrow{e_1^*} r_1 \xrightarrow{e_2^*} r_2 \to \cdots \xrightarrow{e_m^*} r_m \right),$$

with $s_0 = r_0 = *$ and $m, n \in \omega$. Then $\varphi(x) = [u']$, $\varphi(y) = [v']$ with

$$u' = \left(f(s_0) \xrightarrow{g(e_1)} f(s_1) \xrightarrow{g(e_2)} f(s_2) \to \cdots \xrightarrow{g(e_n)} f(s_n) \right)$$

and

$$v' = \left(f(r_0) \xrightarrow{g(e_1^*)} f(r_1) \xrightarrow{g(e_2^*)} f(r_2) \to \cdots \xrightarrow{g(e_m^*)} f(r_m) \right).$$

We proceed in two steps. First assume $\varphi(x) = \varphi(y)$; we show that $x = y$. For this, we may suppose that the computation sequences u', v' are strongly equivalent (and different) in $CS(\mathscr{A}')$. In particular, $n = m$. Then for some $j \leqslant n-2$ we have

(1) $f(s_i) = f(r_i)$ and $g(e_i) = g(e_i^*)$ for each $i \leqslant j$ and each $i > j+2$,

(2) $f(s_j) \xrightarrow{g(e_{j+1})} f(s_{j+1}) \xrightarrow{g(e_{j+2})} f(s_{j+2}) \approx f(r_j) \xrightarrow{g(e^*_{j+1})} f(r_{j+1}) \xrightarrow{g(e^*_{j+2})} f(r_{j+2})$, and thus

(3) $g(e_{j+1}) = g(e^*_{j+2}) \neq g(e_{j+2}) = g(e^*_{j+1})$ and $g(e_{j+1}) \parallel'_{f(s_j)} g(e^*_{j+1})$.

Since g is locally injective, we obtain first $e_i = e_i^*$ and $s_i = r_i$ for each $1 \leqslant i \leqslant j$. Then we get that e_{j+1} and e^*_{j+1} are enabled in $s_j = r_j$, hence $e_{j+1} \parallel_{s_j} e^*_{j+1}$ because of (3), as noted before. This implies that e^*_{j+1} is enabled in s_{j+1}, and thus $e^*_{j+1} = e_{j+2}$ by local injectivity of g. Similarly $e_{j+1} = e^*_{j+2}$. So $e_{j+1} \parallel_{s_j} e_{j+2}$. Thus, $s_{j+2} = r_{j+2}$, and again $e_i = e_i^*$, $s_i = r_i$ also for each $j+2 \leqslant i \leqslant m$. This proves that u is strongly equivalent to v, giving $x = [u] = [v] = y$ as we needed to show.

Now we deal with the general case that $\varphi(x) \leqslant \varphi(y)$. There exists $w' \in CS^*(\mathscr{A}')$ with $u'w' \sim v'$. Let w' have the form $w' = (s'_n \xrightarrow{e'_{n+1}} s'_{n+1} \to \cdots \xrightarrow{e'_{n+k}} s'_{n+k})$ with $s'_n = f(s_n)$; then $s'_{n+k} = f(r_m)$. Since (f, g) is a reduction, inductively we obtain a computation sequence $w = (s_n \xrightarrow{e_{n+1}} s_{n+1} \to \cdots \xrightarrow{e_{n+k}} s_{n+k})$ with $g(e_{n+i}) = e'_{n+i}$ and $f(s_{n+i}) = s'_{n+i}$ for each $1 \leqslant i \leqslant k$. Clearly, $x = [u] \leqslant [uw]$ and $\varphi([uw]) = [u'w'] = [v'] = \varphi([v])$. Then $[uw] = [v] = y$ as shown above, hence $x \leqslant y$.

Similarly to the above, where we constructed the computation sequence w for w', we obtain that φ maps $D^0(\mathscr{A})$ onto $D^0(\mathscr{A}')$. Hence, φ is an order-isomorphism from $(D^0(\mathscr{A}), \leqslant)$ onto $(D^0(\mathscr{A}'), \leqslant)$ and thus also from $(D(\mathscr{A}), \leqslant)$ onto $(D(\mathscr{A}'), \leqslant)$. □

A converse of Theorem 3.6 (under the additional assumption that $\mathscr{A}$, $\mathscr{A}'$ are cancellative) will be proved later, see Corollary 3.20. We continue to establish properties of automata preserved by reductions.

Proposition 3.7. *Let $\mathscr{A}$, $\mathscr{A}'$ be two automata with concurrency relations such that each state of $\mathscr{A}$ is reachable, and let $(f, g): \mathscr{A} \to \mathscr{A}'$ be a reduction. Then*

(a) *$\mathscr{A}$ is cancellative iff $\mathscr{A}'$ is cancellative.*

(b) *$\mathscr{A}$ is concurrent iff $\mathscr{A}'$ is concurrent.*

Proof. We first show that each state of $\mathscr{A}'$ is reachable. Let $s' \in S'$. Then $s' = f(s)$ for some $s \in S$. Choose $u \in CS^0_*(\mathscr{A})$ with codomain s. Then the induced computation sequence u' of $\mathscr{A}'$ satisfies $cod(u') = s'$.

(a) Let $\mathscr{A}'$ be cancellative, and let xtu, xvw be two equivalent finite initial computation sequences of $\mathscr{A}$ ending with transitions t, u, v, w, such that $ev(t) = ev(w) = a$, $ev(u) = ev(v) = b$ and $a \neq b$. Applying the morphism (f, g) to these two computation sequences, we obtain two equivalent computation sequences $x't'u'$, $x'v'w'$ of $\mathscr{A}'$ with transitions t', u', v', w' such that $cod(x') = f(cod(x))$, $ev(t') = ev(w') = g(a)$ and $ev(u') = ev(v') = g(b)$. Since g is locally injective, we have $g(a) \neq g(b)$. As $\mathscr{A}'$ is cancellative, we get $g(a) \|'_{cod'(x')} g(b)$ and so $a \|_{cod(x)} b$. Hence $\mathscr{A}$ is cancellative.

For the converse, we argue similarly. Given two equivalent computation sequences $x't'u'$, $x'v'w'$ of $\mathscr{A}'$ as above, we obtain computation sequences xtu, xvw of $\mathscr{A}$ which induce $x't'u'$ and $x'v'w'$ respectively, through (f, g). As shown in the proof of Theorem 3.6, then xtu and xvw are equivalent. Since $\mathscr{A}$ is cancellative, we obtain $a \|_{cod(x)} b$ and then $g(a) \|'_{cod'(x')} g(b)$, as required.

(b) Let $\mathscr{A}'$ be concurrent, and let (q, a, p), (q, b, r), $(q, c, s) \in T$ such that $a \|_q b$, $a \|_q c$, $b \|_p c$. Letting $g(a) = a'$, $g(b) = b'$, etc., $f(s) = s'$, we obtain transitions (q', a', p'), (q', b', r'), (q', c', s') of $\mathscr{A}'$ with $a' \|'_{q'} b'$, $a' \|'_{q'} c'$, $b' \|'_{p'} c'$. Hence, also $b' \|'_{q'} c'$, $a' \|'_{r'} c'$, $a' \|'_{s'} b'$. Since (f, g) is a reduction, this implies $b \|_q c$, $a \|_r c$, $a \|_s b$, as required. Hence $\mathscr{A}$ is concurrent. The converse is proved similarly. □

Next we begin with the definition of the functor $A: LDom \to Aut$. First we show how to construct an automaton with concurrency relations from a given labelled domain.

Definition 3.8. Let $\mathscr{D} = (D, \leqslant, l_E, l_S)$ be a labelled weak concurrency domain. We define an *associated* automaton with concurrency relations $A(\mathscr{D}) = (S, E, T, *, \|)$ as follows:

(1) $T = \{(s, e, s') \in S \times E \times S' :$ there are $x, y \in D^0$ with $x \overset{e}{\prec} y$, $l_S(x) = s$ and $l_S(y) = s'\}$;

(2) $* = l_S(\perp)$;

(3) $\| = (\|_s)_{s \in S}$ where for all $s \in S$ and $e_1, e_2 \in E$, we put $e_1 \|_s e_2$ iff e_1 and e_2 are weakly observably concurrent in $\mathscr{D}$ at s.

Lemma 3.9. *In the above situation, $A(\mathscr{D})$ is an automaton with concurrency relations in which each state is reachable.*

Proof. Assume that $(s,e,s'), (s,e,s'')\in T$. We claim that $s'=s''$. Choose $x,x',y,y'\in D^0$ with $x\overset{e}{\prec} y$, $x'\overset{e}{\prec} y'$, $l_S(x)=l_S(x')=s, l_S(y)=s'$, and $l_S(y')=s''$. There is $y''\in D^0$ with $x'\overset{e}{\prec} y''$ and $l_S(y'')=s'$. Then $y'=y''$, so $s'=s''$. Clearly, for each $e\in E$ there are $x,y\in D^0$ with $x\overset{e}{\prec} y$; then $(l_S(x),e,l_S(y))\in T$. Now let $e_1\|_s e_2$. Then $s=l_S(x)$ for some $x\in D^0$ and there are $y_i,z\in D^0$ with $x\overset{e_i}{\prec} y_i\overset{e_{3-i}}{\longrightarrow} z$ $(i=1,2)$. Hence, we obtain transitions $(s,e_i,l_S(y_i))$, $(l_S(y_i),e_{3-i},l_S(z))$ for $i=1,2$ in $A(\mathscr{D})$ as required.

Next let $s\in S$. There is $x\in D^0$ with $l_S(x)=s$. Choose any covering chain $\perp=x_0\prec x_1\prec\cdots\prec x_n=x$ in D from $\perp$ to x. Let $t_i=(l_S(x_{i-1}), l_E([x_{i-1},x_i]), l_S(x_i))$ for $i=1,\ldots,n$. Then $t_1\ldots t_n$ is a finite initial computation sequence of $A(\mathscr{D})$ with codomain s. □

Let $\mathscr{D}$, $\mathscr{D}'$ be two labelled weak concurrency domains and $(\varphi,f,g):\mathscr{D}\to\mathscr{D}'$ a morphism. We then put $A(\varphi,f,g)=(f,g)$, and we have to show that this is a morphism from $A(\mathscr{D})$ to $A(\mathscr{D}')$. This is done in the subsequent proposition.

Proposition 3.10. *The correspondence $A: LDom\to Aut$ is a functor. Moreover, A maps order-morphisms between nicely labelled domains to reductions.*

Proof. Let $\mathscr{D}=(D,\leqslant,l_E,l_S)$, $\mathscr{D}'=(D',\leqslant,l_{E'},l_{S'})$ be two labelled weak concurrency domains and $(\varphi,f,g):\mathscr{D}\to\mathscr{D}'$ be a domain-morphism. We claim that then $(f,g):\mathbf{A}(\mathscr{D})\to A(\mathscr{D}')$ is a morphism in Aut. First let $(s,e,s')\in T$. There are $x,y\in D^0$ with $x\overset{e}{\prec} y$, $l_S(x)=s$ and $l_S(y)=s'$. Then $\varphi(x)\overset{g(e)}{\prec}\varphi(y)$ in $\mathscr{D}'$, $l_{S'}(\varphi(x))=f(s)$, and $l_{S'}(\varphi(y))=f(s')$. Hence $(f(s),g(e),f(s'))\in T'$.

Now assume that $e_1\|_s e_2$ in $A(\mathscr{D})$ and $g(e_1)\neq g(e_2)$. There are $x,y_i,z\in D^0$ with $l_S(x)=s$ and $x\overset{e_i}{\prec} y_i\overset{e_{3-i}}{\prec} z$ for $i=1,2$. Then in $\mathscr{D}'$ we have $l_{S'}(\varphi(x))=f(s)$ and $\varphi(x)\overset{g(e_i)}{\prec}\varphi(y_i)\overset{g(e_{3-i})}{\prec}\varphi(z)$ for $i=1,2$. This shows that $g(e_1)\|'_{f(s)}g(e_2)$ in $A(\mathscr{D}')$ as needed. Obviously, A preserves compositions and identities.

Suppose now that $\mathscr{D}$ and $\mathscr{D}'$ are nicely labelled and that φ is an order-isomorphism. We show that then (f,g) is a reduction. To show that f is onto, let $s'\in S'$. Choose $x'\in D'^0$ with $l_{S'}(x')=s'$. Then $x=\varphi^{-1}(x')\in D^0$ satisfies $f(l_S(x))=l_{S'}(\varphi(x))=s'$. Next let $s\in S$ and $(f(s),e',r')\in T'$. Choose $x\in D^0$ with $l_S(x)=s$. Then $f(s)=l_{S'}(\varphi(x))$, and by Definition 2.5(1) there is $y'\in D'^0$ with $\varphi(x)\overset{e'}{\prec} y'$ and $l_{S'}(y')=r'$. Choose $y\in D^0$ with $\varphi(y)=y'$. Then $x\prec y$. Putting $e=l_E([x,y])$ and $r=l_S(y)$, we get $(s,e,r)\in T$ and $g(e)=e'$. This proves condition (3.5)(2).

To check (3.5)(3), let $s\in S$ and $e'_1, e'_2\in E'$ with $e'_1\|'_{f(s)}e'_2$ in $A(\mathscr{D}')$. Choose $x\in D^0$ with $l_S(x)=s$. Since $l_{S'}(\varphi(x))=f(s)$ and $\mathscr{D}'$ is nicely labelled, there are $y'_1,y'_2,z'\in D'^0$ with

$\varphi(x) \overset{e_i'}{\prec} y_i' \overset{e_3'}{\succ} z'$ for $i=1,2$. Let $y_i=\varphi^{-1}(y_i')$ $(i=1,2)$ and $z=\varphi^{-1}(z')$. Then $x \prec y_i \prec z$, and $e_i=l_E([x,y_i])$ satisfies $g(e_i)=e_i'$ $(i=1,2)$ so $e_1 \neq e_2$. Hence we have $e_1 \|_s e_2$ in $A(\mathscr{D})$.

Finally, let $s \in S$ and $(s, e_i, s_i) \in T$ for $i=1,2$. There are $x, y_1, y_2 \in D^0$ with $l_S(x)=s$ and $x \overset{e_i}{\prec} y_i$, $l_S(y_i)=s_i$ for $i=1,2$. So $\varphi(x) \overset{g(e_i)}{\prec} \varphi(y_i)$ for $i=1,2$. Now $e_1 \neq e_2$ implies $y_1 \neq y_2$, thus $g(e_1) \neq g(e_2)$. Hence g is locally injective. □

It will be useful to consider first $A(\mathscr{D})$ where the labelled domain $\mathscr{D}$ has a particularly simple form.

Definition 3.11. Let $(D, \leqslant)$ be a weak concurrency domain. Let l be the canonical event-labelling function on $(D, \leqslant)$. As noted before, $(D, \leqslant, l, id_{D^0})$ is nicely labelled. Its associated automaton $A(D)=(S, E, T, *, \|)$ has the following form:

(1) $S=D^0$ and $*=\perp$;

(2) $E=\{[x,y]_{\prec\succ} : x, y \in D^0, x \prec y\}$;

(3) $T=\{(x, [x,y]_{\prec\succ}, y) : x, y \in D^0, x \prec y\}$;

(4) $\| = (\|_s)_{s \in S}$ where $[x, y_1] \|_x [x, y_2]$ iff $y_1 \neq y_2$ and there exists $z \in D^0$ with $y_i \prec z$ $(i=1,2)$.

This automaton will be called the *canonical* automaton with concurrency relations associated with $(D, \leqslant)$.

These automata were studied before in [10]. Next we note the close relationship between $\mathscr{D}$ and induced labelled domain of $A(\mathscr{D})$.

Lemma 3.12. Let $\mathscr{D}=(D, \leqslant, l_E, id_{D^0})$ be a weak concurrency domain with canonical event-labelling function and trivial state-labelling. Let $\mathscr{A}=A(\mathscr{D})$ be the canonical automaton associated with $(D, \leqslant)$. Then there exists an isomorphism $(\varphi, id_{D^0}, id_E)$ from $\mathscr{D}$ onto $\boldsymbol{D}(A)$, the induced labelled domain of $\mathscr{A}$.

Proof. Define $\varphi : D^0 \to D^0(\mathscr{A})$ as follows. If $x \in D^0$, choose a covering chain $\perp = x_0 \prec \cdots \prec x_n = x$ from $\perp$ to x. Let $t_i=(x_{i-1}, [x_{i-1}, x_i]_{\prec\succ}, x_i)$ $(i=1, \ldots, n)$ and put $\varphi(x)=[t_1 \ldots t_n]$ (if $x=\perp$ let $\varphi(x)=[\varepsilon]$). As shown in [10; proof of Theorem 2.4], $\varphi : (D^0, \leqslant) \to (D^0(\mathscr{A}), \leqslant)$ is an order-isomorphism and thus extends uniquely to an order-isomorphism from $(D, \leqslant)$ onto $(\mathbf{D}(\mathscr{A}), \leqslant)$. Obviously, $cod(\varphi(x))=x$ for each $x \in D^0$. Now if $x, y \in D^0$ with $x \prec y$, let $\varphi(x)$ be given as above, and put $t=(x, [x,y]_{\prec\succ}, y) \in T$. Then $\varphi(y)=[t_1 \ldots t_n t]$, so $l_{E^*}([\varphi(x), \varphi(y)])=[x,y]_{\prec\succ}=l_E([x,y])$. The result follows. □

Now we can generalize Lemma 3.12 to arbitrary nicely labelled domains.

Proposition 3.13. *Let $\mathscr{D}=(D, \leqslant, l_E, l_S)$ be a nicely labelled domain. Then there exists an isomorphism $\eta_{\mathscr{D}}=(\pi_D, id_S, id_E)$ from $\mathscr{D}$ to $\boldsymbol{D} \circ A(\mathscr{D})$.*

Proof. Let $l_{E'}$ be the canonical event-labelling function on $(D, \leqslant)$, and let $\mathscr{D}' = (D, \leqslant, l_{E'}, id_{D^0})$. Then $\mathscr{A}' = \boldsymbol{A}(\mathscr{D}')$ is the canonical automaton associated with $(D, \leqslant)$. Obviously, the triple (id_D, l_S, g) where $g([x,y]_{><}) = l_E([x,y])$ for any $[x,y] \in Int_D$, is an order-morphism from $\mathscr{D}'$ to $\mathscr{D}$. Hence, by Proposition 3.10, $(l_S, g): \mathscr{A}' \to \boldsymbol{A}(\mathscr{D})$ is a reduction. Let (φ, l_S, g) be the induced morphism from $\boldsymbol{D}(\mathscr{A}')$ to $\boldsymbol{D} \circ \boldsymbol{A}(\mathscr{D})$. By Theorem 3.6, φ is an order-isomorphism. By Lemma 3.12, there is also an isomorphism $(\psi, id_{D^0}, id_{E'})$ from $\mathscr{D}'$ to $\boldsymbol{D}(\mathscr{A}')$. Hence, $(\varphi \circ \psi, l_S, g)$ is an order-morphism from $\mathscr{D}'$ to $\boldsymbol{D} \circ \boldsymbol{A}(\mathscr{D})$. Observing that (id_D, l_S, g) is an order-morphism from $\mathscr{D}'$ to $\mathscr{D}$, we obtain easily that $(\varphi \circ \psi, id_S, id_E)$ is an isomorphism from $\mathscr{D}$ to $\boldsymbol{D} \circ \boldsymbol{A}(\mathscr{D})$.

We record an explicit description of the order-isomorphism $\pi_D = \varphi \circ \psi$ obtained in the proof of Proposition 3.13. Let $x \in D^0$ and choose any covering chain $\perp = x_0 \prec x_1 \prec \cdots \prec x_n = x$ from $\perp$ to x in $\mathscr{D}$. Then each $t_i = (l_S(x_{i-1}), l_E([x_{i-1}, x_i]), l_S(x_i))$ $(i = 1, \ldots, n)$ is a transition of $\boldsymbol{A}(\mathscr{D})$, so $u = t_1 \ldots t_n \in CS^0_*(\boldsymbol{A}(\mathscr{D}))$ (if $x = \perp$, we have $u = \varepsilon$, the empty computation sequence of $\boldsymbol{A}(\mathscr{D})$). Then $\pi_D(x) = \varphi \circ \psi(x) = [u]$, as can be easily checked.

Corollary 3.14. *Let $\mathscr{D} = (D, \leqslant, l_E, l_S)$ be a nicely labelled weak concurrency domain.*

(a) *Let $\mathscr{A}$ be the canonical automaton associated with $(D, \leqslant)$. Then there exists a reduction from $\mathscr{A}$ to $\boldsymbol{A}(\mathscr{D})$.*

(b) *$\boldsymbol{A}(\mathscr{D})$ is cancellative.*

(c) *If $(D, \leqslant)$ is a concurrency domain, then $\boldsymbol{A}(\mathscr{D})$ is concurrent.*

Proof. (a) This was shown in the course of the proof of Proposition 3.13.

(b) By (a) above and Proposition 3.7(a), it suffices to prove that $\mathscr{A}$ is cancellative. Let xtu, xvw be two equivalent finite initial computation sequences of $\mathscr{A}$ with $ev(t) = ev(w) = a$, $ev(u) = ev(v) = b$ and $a \neq b$. Then xtu and xvw have the same codomain, say z, and letting $cod(x) = x'$, we obtain $y_1, y_2 \in D^0$ with $x' \prec y_i \prec z$ for $i = 1, 2$ and $a = [x', y_1]_{><}$, $b = [x', y_2]_{><}$. Hence $a \|_{x'} b$ as required.

(c) As shown in [10], the canonical automaton associated with any concurrency domain is concurrent. Hence, the result follows from (a) and Proposition 3.7(b). □

Note that by Corollary 3.14(b), the functor $\boldsymbol{A}$ maps the objects of *NLDom* into *CAut*. We will denote the restriction of $\boldsymbol{A}$ to the subcategory *NLDom* also simply by $\boldsymbol{A}$. Now we show that there is a close relationship between the automata $\mathscr{A}$ and $\boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A})$.

Proposition 3.15. *Let $\mathscr{A}$ be an automaton with concurrency relations in which each state is reachable, and let $\mathscr{A}' = \boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A})$. Then $\varepsilon_{\mathscr{A}} = (id_S, id_E): \mathscr{A} \to \mathscr{A}'$ is a morphism. Moreover, $\varepsilon_{\mathscr{A}}$ is an isomorphism and $\mathscr{A} = \mathscr{A}'$, if and only if $\mathscr{A}$ is cancellative.*

Proof. Let $\mathscr{A} = (S, E, T, *, \|)$ and $\mathscr{A}' = (S, E, T', *, \|')$. We first show that $T = T'$. Let $(s, e, s') \in T$. Choose a finite initial computation sequence u of $\mathscr{A}$ with codomain s, and

put $u'=u(s,e,s')$. Then in $\boldsymbol{D}(\mathscr{A})$ we have $[u]\overset{e}{\prec}[u']$, $l_S([u])=s$, and $l_S([u'])=s'$, so $(s,e,s')\in T'$. Conversely, let $(s,e,s')\in T'$. Then there are $x,y\in D^0(\mathscr{A})$ with $x\overset{e}{\prec}y$, $l_S(x)=s$ and $l_S(y)=s'$. Hence, if $x=[u]$, then $y=[u(s,e,s')]$ and $(s,e,s')\in T$.

Next we show that $\|_s\subseteq\|'_s$ for each $s\in S$. Indeed, let $e_1\|_s e_2$. Choose any $u\in CS^0_*(\mathscr{A})$ with $cod(u)=s$. There are transitions $(s,e_i,p_i),(p_i,e_{3-i},q)\in T$ and $[u]\overset{e_i}{\prec}[u(s,e_i,p_i)]\overset{e_{3-i}}{\prec}[u(s,e_i,p_i)(p_i,e_{3-i},q)]=z_i$ for $i=1,2$ and $z_1=z_2$ by $e_1\|_s e_2$. Hence $e_1\|'_s e_2$ in $\mathscr{A}'$. Thus, $\varepsilon_{\mathscr{A}}:\mathscr{A}\to\mathscr{A}'$ is a morphism.

It follows that $\varepsilon_{\mathscr{A}}$ is an isomorphism iff $\mathscr{A}=\mathscr{A}'$ iff $\|'_s\subseteq\|_s$ for each $s\in S$. The latter condition means that whenever e_1 and e_2 are weakly observably concurrent in $\boldsymbol{D}(\mathscr{A})$ at $s\in S$, then $e_1\|_s e_2$, i.e., $\mathscr{A}$ is cancellative. □

Now we can derive the second main result of this section.

Theorem 3.16. *The functors $\boldsymbol{A}:NLDom\to CAut$ and $\boldsymbol{D}:CAut\to NLDom$ form an equivalence of categories; $\boldsymbol{A}\circ\boldsymbol{D}$ is the identity functor on $CAut$.*

Proof. By Propositions 3.4, 3.10 and Corollary 3.14(b), $\boldsymbol{D}$ and $\boldsymbol{A}$ are functors. By Propositions 3.15 and 3.13, $\varepsilon_{\mathscr{A}}:\mathscr{A}\to\boldsymbol{A}\circ\boldsymbol{D}(\mathscr{A})$ and $\eta_{\mathscr{D}}:\mathscr{D}\to\boldsymbol{D}\circ\boldsymbol{A}(\mathscr{D})$ are isomorphisms. Since $\mathscr{A}=\boldsymbol{A}\circ\boldsymbol{D}(\mathscr{A})$, $\boldsymbol{A}\circ\boldsymbol{D}$ is the identity functor. It only remains to prove the commutativity of the diagram:

$$\begin{array}{ccc}\mathscr{D} & \xrightarrow[\eta_{\mathscr{D}}]{} & \boldsymbol{D}\circ\boldsymbol{A}(\mathscr{D})\\ \downarrow(\varphi,f,g) & & \downarrow\boldsymbol{D}\circ\boldsymbol{A}(\varphi,f,g)\\ \mathscr{D}' & \xrightarrow[\eta_{\mathscr{D}'}]{} & \boldsymbol{D}\circ\boldsymbol{A}(\mathscr{D}')\end{array}$$

But the equality of the two morphisms $\eta_{\mathscr{D}'}\circ(\varphi,f,g)$ and $(\boldsymbol{D}\circ\boldsymbol{A}(\varphi,f,g))\circ\eta_{\mathscr{D}}$ from $\mathscr{D}$ to $\boldsymbol{D}\circ\boldsymbol{A}(\mathscr{D}')$ is immediate by Proposition 3.3. □

As a consequence, for the important class of concurrent automata, Theorem 3.16 cuts down to the following result.

Corollary 3.17. *The categories of concurrent automata and nicely labelled concurrency domains are equivalent.*

Proof. By Proposition 2.9, each concurrent automaton $\mathscr{A}$ is cancellative, and, futhermore, $(\boldsymbol{D}(\mathscr{A}),\leqslant)$ is a concurrency domain, cf. [10]. Together with Corollary 3.14(c) and Theorem 3.16, this implies the result. □

Using results of [10], similarly we also obtain equivalences between certain subcategories of concurrent automata and categories, e.g., of nicely labelled event domains, or nicely labelled dI-domains. The formulation of these is left to the reader. Let us call a morphism $(f,g):\mathscr{A}\to\mathscr{A}'$ *strong*, if whenever $e\|_s e'$ in $\mathscr{A}$, then $g(e)\|'_{f(s)}g(e')$ in

$\mathscr{A}'$. For this kind of morphisms, an adjunction between a category of automata with concurrency relations and a category of Petri nets with capacities has been recently derived in Droste and Shortt [11]. Let us call a morphism $(\varphi, f, g): \mathscr{D} \to \mathscr{D}'$ between two labelled domains *strong*, if whenever $x, y_i, z \in D^0$ with $x \prec y_i \prec z$ $(i=1,2)$ and $y_1 \neq y_2$, then also $\varphi(y_1) \neq \varphi(y_2)$. Then we have the following corollary.

Corollary 3.18. *The category of all cancellative automata with concurrency relations and strong morphisms is equivalent to the category of all nicely labelled weak concurrency domains and strong morphisms.*

Proof. Let $(f, g): \mathscr{A} \to \mathscr{A}'$ be a morphism, where $\mathscr{A}$ and $\mathscr{A}'$ are cancellative, and let $(\varphi, f, g): \boldsymbol{D}(\mathscr{A}) \to \boldsymbol{D}(\mathscr{A}')$ be the induced morphism. Because of Theorem 3.16, it suffices to show that (f, g) is strong iff (φ, f, g) is strong.

First assume (f, g) is strong, and let $x, y_i, z \in \boldsymbol{D}(\mathscr{A})^0$, $x \overset{e_i}{\prec} y_i \prec z$ $(i=1,2)$ and $y_1 \neq y_2$. Then e_1 and e_2 are weakly observably concurrent in $\boldsymbol{D}(\mathscr{A})$ art $s = l_S(x)$. Hence $e_1 \|_s e_2$, since $\mathscr{A}$ is cancellative. Then, in particular, $g(e_1) \neq g(e_2)$, as (f, g) is strong. Since $\varphi(x) \overset{g(e_i)}{\prec} \varphi(y_i)$ for $i=1,2$, we obtain $\varphi(y_1) \neq \varphi(y_2)$, proving that (φ, f, g) is strong.

For the converse, let (φ, f, g) be strong, and let $e_1 \|_s e_2$ in $\mathscr{A}$. Then e_1 and e_2 are weakly observably concurrent in $\boldsymbol{D}(\mathscr{A})$ at s, so we obtain $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ $(i=1,2)$ in $\boldsymbol{D}(\mathscr{A})$ with $y_1 \neq y_2$ and $cod(x) = s$. Then $\varphi(x) \overset{g(e_i)}{\prec} \varphi(y_i) \overset{g(e_{3-i})}{\prec} \varphi(z)$ $(i=1,2)$ in $\boldsymbol{D}(\mathscr{A}')$ and $\varphi(y_1) \neq \varphi(y_2)$. So $g(e_1) \neq g(e_2)$. As $cod(\varphi(x)) = f(cod(x)) = f(s)$ and $\mathscr{A}'$ is cancellative, we have $g(e_1) \|'_{f(s)} g(e_2)$. □

Putting Corollary 3.18 and the results of [11] together, we obtain an adjunction between a category of Petri nets with capacities and nicely labelled weak concurrency domains. We note that the computation sequences of the "intermediate" automata correspond to the possible firing sequences of events of the underlying Petri nets, starting from the initial marking. Restricting the morphisms considered further to reductions, we obtain the following result.

Corollary 3.19. *The category of all cancellative automata with reductions as arrows is equivalent to the category of all nicely labelled weak concurrency domains with order-morphisms as arrows.*

Proof. Immediate by Theorems 3.16 and 3.6 and Proposition 3.10. □

Next we obtain a partial converse of Theorem 3.6.

Corollary 3.20. *Let $\mathscr{A}$, $\mathscr{A}'$ be two cancellative automata, let $(f, g): \mathscr{A} \to \mathscr{A}'$ be a morphism, and let $(\varphi, f, g): \boldsymbol{D}(\mathscr{A}) \to \boldsymbol{D}(\mathscr{A}')$ be the induced domain-morphism. If φ is an order-isomorphism, (f, g) is a reduction.*

Proof. By Propositions 3.10 and 3.15, $(f, g) = \boldsymbol{A}(\varphi, f, g)$ is a reduction from $\mathscr{A} = \boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A})$ to $\boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A}') = \mathscr{A}'$. □

As noted in the introduction, in general a given weak concurrency domain can be generated by various automata with concurrency relations. The following is a solution of the "unique representation problem".

Corollary 3.21. *For each nicely labelled weak concurrency domain $\mathscr{D}$ there exists, up to isomorphism, a unique cancellative automaton $\mathscr{A}$ which generates $\mathscr{D}$.*

Proof. By Proposition 3.13 and Corollary 3.14(b), $\boldsymbol{A}(\mathscr{D})$ is a cancellative automaton generating $\mathscr{D}$. Let $\mathscr{A}$ be any cancellative automaton generating $\mathscr{D}$, i.e. with $\boldsymbol{D}(\mathscr{A}) \cong \mathscr{D}$. By Proposition 3.15, $\mathscr{A} = \boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A}) \cong \boldsymbol{A}(\mathscr{D})$. □

Corollary 3.21 indicates how to construct, given a weak concurrency domain $(D, \leqslant)$, a cancellative automaton $\mathscr{A}$ generating $(D, \leqslant)$: we have to find event- and state-labelling functions l_E, l_S such that $\mathscr{D} = (D, \leqslant, l_E, l_S)$ is nicely labelled; then put $\mathscr{A} = \boldsymbol{A}(\mathscr{D})$. We illustrate this by a simple example. Let $\mathscr{D} = \mathbb{N} \cup \{\infty\}$ with the natural order of $\mathbb{N}$ and such that $x < \infty$ for each $x \in \mathbb{N}$. There are the following ways to make $(D, \leqslant, l_E, l_S)$ into a labelled domain.

(1) $l_S : \mathbb{N} \to S$ is a bijection, and $Int_D \to E$ is an arbitrary surjection.

(2) S is a finite set, say $|S| = n$. Then E is also finite, $|E| \leqslant |S|$, we have $l_S(i+n) = l_S(i)$ and $l_E([i+n, i+n+1]) = l_E([i, i+1])$ for all $i \in \mathbb{N}$, and $S = \{l_S(i) : 1 \leqslant i \leqslant n\}$.

We now show that by defining labelling functions, l_E, l_S of $(D, \leqslant)$ satisfying restriction (1) or (2), the automata associated with $\mathscr{D} = (D, \leqslant, l_E, l_S)$ provide up to isomorphism *all* automata with concurrency relations and only reachable states generating $(D, \leqslant)$. Indeed, let $\mathscr{A}$ be any such automaton. Since $(D, \leqslant)$ is linearly ordered, no two events of $\mathscr{A}$ can be weakly observably concurrent in the induced domain $\boldsymbol{D}(\mathscr{A})$ at s $(s \in S)$; hence $\|_s = \phi$ for each state s of $\mathscr{A}$, and $\mathscr{A}$ is cancellative. Thus $\mathscr{A} = \boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A}) \cong \boldsymbol{A}(\mathscr{D})$, proving our claim.

An automaton with concurrency relations $\mathscr{A}$ is *finite*, if both its event set and its state set are finite. Next we characterize which weak concurrency domains can be generated by finite cancellative automata.

Corollary 3.22. *Let $(D, \leqslant)$ be a domain. The following are equivalent.*

(1) *$(D, \leqslant)$ can be generated by a finite cancellative automaton.*

(2) *$(D, \leqslant)$ is a weak concurrency domain, and there exist event- and state-labelling functions l_E, l_S for $(D, \leqslant)$ with E and S finite such that $\mathscr{D} = (D, \leqslant, l_E, l_S)$ is nicely labelled.*

Proof. Straightforward by Theorem 3.16. □

We note here that for any event-labelling function l_E for $(D, \leqslant)$ (for instance, with E finite), the proof of Theorem 4.3 below shows how to construct a state-labelling function l_S with S of minimal cardinality such that $(D, \leqslant, l_E, l_S)$ is nicely labelled.

Finally, we turn to an application of our results in trace theory.

A *trace alphabet* $\mathscr{E}=(E, \|)$ consists of a countable set E together with a symmetric irreflexive binary relation $\|$ on E. The free partially commutative monoid $M(\mathscr{E})$ is the quotient of the free monoid E^* over E modulo the congruence generated by $ab \sim ba$ whenever $a \| b$ $(a, b \in E)$, cf. [17, 8]. We denote this congruence on E^* also by $\sim$. We define a partial order on $M(\mathscr{E})$ by putting $x \leqslant z$ iff $xy=z$ for some $y \in M(\mathscr{E})$. Let $(D(\mathscr{E}), \leqslant)$ be the ideal completion of the partial order $(M(\mathscr{E}), \leqslant)$; then $(D(\mathscr{E}), \leqslant)$, the *domain of all traces* of $\mathscr{E}$, is a Scott-domain with $(D^0(\mathscr{E}), \leqslant) \cong (M(\mathscr{E}), \leqslant)$. It is known that $(D(\mathscr{E}), \leqslant)$ is, in fact, a coherent dI-domain.

Our goal is to characterize the domains of this form. Given a concurrency domain $(D, \leqslant)$, for each $x \in D^0$ we define an *associated* trace alphabet $\mathscr{E}_x=(E_x, \|_x)$ as follows.

(1) $E_x=\{[x, y]: x \prec y\}$,

(2) $[x, y] \|_x [x, z]$ iff $y \neq z$ and $\{y, z\}$ is bounded in D. (Recall that then $y \prec y \vee z$ and $z \prec y \vee z$ by axiom (C).)

The following provides a characterization of the domains $(D, \leqslant)$ isomorphic to $(D(\mathscr{E}), \leqslant)$ for some trace alphabet $\mathscr{E}$; we are thankful to an anonymous referee who provided a characterization similar to the one of (1)$\Leftrightarrow$(3) below.

Corollary 3.23. *Let $(D, \leqslant)$ be a partially ordered set. The following are equivalent.*

(1) $(D, \leqslant) \cong (D(\mathscr{E}), \leqslant)$ *for some trace alphabet* $\mathscr{E}$.

(2) $(D, \leqslant)$ *is concurrency domain, and there exists an event-labelling function l_E on $(D, \leqslant)$ with the following two properties*:

(i) *Whenever $x, y, x' \in D^0$ with $x \overset{e}{\prec} y$, then there is $y' \in D^0$ with $x' \overset{e}{\prec} y'$.*

(ii) *Whenever $x, y_i, z, x', y_i', z_i' \in D^0$ with $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ and $x' \overset{e_i}{\prec} y_i' \overset{e_{3-i}}{\prec} z_i'$ for $i=1, 2$, and $y_1 \neq y_2$, then $z_1'=z_2'$.*

(3) $(D, \leqslant)$ *is a concurrency domain, and for any $x, y \in D^0$ with $x \prec y$ there is an isomorphism $\varphi_{xy}: \mathscr{E}_x \to \mathscr{E}_y$ between the associated trace alphabets with the following two properties*:

(i) $\varphi_{xy}([x, z])=[y, y \vee z]$ *whenever $y \neq z$ and $\{y, z\}$ is bounded.*

(ii) *Whenever $x, y_i, z \in D^0$ with $x \prec y_i \prec z$ for $i=1, 2$ then $\varphi_{y_1 z} \circ \varphi_{xy_1}=\varphi_{y_2 z} \circ \varphi_{xy_2}$.*

Proof. (1)$\Rightarrow$(2): Let $\mathscr{E}=(E, \|)$. We define an automaton with concurrency relations $\mathscr{A}=(S, E, T, *, \|')$ as follows. Let $S=\{*\}$, a singleton set, $T=\{(*, e, *): e \in E\}$ and $\|'=(\|_*)$ with $\|_*=\|$. Then initial computation sequences of $\mathscr{A}$ correspond uniquely to words over E, i.e. elements of E^*, and this correspondence yields a canonical isomorphism from $(D(\mathscr{A}), \leqslant)$ to $(D(\mathscr{E}), \leqslant)$, as is easy to see. Clearly, $\mathscr{A}$ is concurrent and $\boldsymbol{D}(\mathscr{A})$ is a nicely labelled concurrency domain. Observing that $|S|=1$, properties (i) and (ii) are thus immediate for $\boldsymbol{D}(\mathscr{A})$. Since $(D, \leqslant) \cong (\boldsymbol{D}(\mathscr{A}), \leqslant)$, our claim follows.

(2)⇒(1): Let $S=\{s\}$ be a singleton set and $l_S: D^0 \to S$ the uniquely determined function onto S. Then properties (i) and (ii) say that $\mathscr{D}=(D, \leqslant, l_E, l_S)$ is nicely labelled. Put $\mathscr{A}=A(\mathscr{D})$ and $\mathscr{E}=(E, \|)$ where $\| = \|_s$ from $\mathscr{A}$. Then $\mathscr{E}$ is a trace alphabet, and as above we have $(D(\mathscr{A}), \leqslant) \cong (D(\mathscr{E}), \leqslant)$. Since $\mathscr{D} \cong \boldsymbol{D}(\mathscr{A})$, the result follows.

(2)⇒(3): Let $x, y \in D^0$, with $x \prec y$. For any $z \in D^0$ with $x \prec z$ we put $\varphi_{xy}([x,z])=[y,z']$ where z' is the uniquely determined element of D^0 with $l_E([y,z'])=l_E([x,z])$. If here $y \neq z$ and $\{y,z\}$ is bounded, we obtain $z'=y \vee z$ since $(D, \leqslant)$ is a concurrency domain. It follows that $\varphi_{xy}: \mathscr{E}_x \to \mathscr{E}_y$ is an isomorphism satisfying condition (3)(i), and condition (3)(ii) is straightforward from the definition of φ_{xy}.

(3)⇒(2): For any $x \in D^0$, we define an isomorphism $\varphi_x: \mathscr{E}_\perp \to \mathscr{E}_x$ as follows. If $x=\perp$, let $\varphi_\perp = id$. If $\perp < x$, choose a covering chain $\perp = x_0 \prec x_1 \prec \cdots \prec x_n = x$ from $\perp$ to x. Then put $\varphi_x = \varphi_{x_{n-1}x_n} \circ \cdots \circ \varphi_{x_0 x_1}$, the composition of the given trace alphabet isomorphisms $\varphi_{x_i x_{i+1}}$ $(i=n-1, \ldots, 0)$. Since any concurrency domain is a weak concurrency domain, in particular satisfies axiom (E) of Definition 2.3, by requirement (3) (ii), φ_x is well-defined and hence an isomorphism as claimed. Now put $E=\{[\perp, z]: \perp \prec z\}$, and we define a function $l_E: Int_D \to E$ by letting $l_E([x,y])=[\perp, z]$ if $\varphi_x([\perp, z])=[x,y]$ (i.e. l_E is the extension of the mappings $\varphi_x^{-1}(x \in D^0)$). It easily follows that l_E is an event-labelling function satisfying condition (2)(i). To check (2)(ii), let $x, y_i, z, x', y_i', z_i' \in D^0$ with $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ and $x' \overset{e_i}{\prec} y_i' \overset{e_{3-i}}{\prec} z_i'$ for $i=1,2$ and $y_1 \neq y_2$. Since $[x,y_1] \|_x [x,y_2]$ in $\mathscr{E}_x$ and $\varphi_{x'} \circ \varphi_x^{-1}: \mathscr{E}_x \to \mathscr{E}_{x'}$ is an isomorphism, we obtain $[x',y_1'] \|_{x'} [x',y_2']$ in $\mathscr{E}_{x'}$, so $y_i' \overset{e_{3-i}}{\prec} y_1' \vee y_2'$ for $i=1,2$, and thus $z_1'=y_1' \vee y_2'=z_2'$. □

4. Minimal automata generating given domains

Let $(D, \leqslant)$ be a given weak concurrency domain. In this section we will study the problem whether the collection of all automata with concurrency relations generating $(D, \leqslant)$ contains with respect to the preorder induced by the existence of reductions, maximal or minimal automata. The same question arises by considering only automata with a fixed set E of events; for example, can every such automaton be reduced to a minimal automaton of this kind, i.e. can the state set be made as small as possible? Dually, we may consider automata with a fixed state set and try to minimize the event set. We will show that all these questions under slight additional assumptions have positive answers.

We introduce some notation. An event-morphism $(\varphi, g): (D, \leqslant, l_E) \to (D', \leqslant, l_{E'})$ between two event-labelled domains will be called an *isomorphism*, if both φ and g are bijective. Let $\mathscr{A}$ be an automaton with concurrency relations and $\boldsymbol{D}(\mathscr{A})=(D(\mathscr{A}), \leqslant, l_E, l_S)$ its induced labelled domain. We say that $\mathscr{A}$ *generates* $(D', \leqslant, l_{E'})$, if $(D(\mathscr{A}), \leqslant, l_E)$ and $(D', \leqslant, l_{E'})$ are isomorphic. Also, $\mathscr{A}$ *generates*

$(D', \leqslant, l_{S'})$, if $\boldsymbol{D}(\mathscr{A})$ is isomorphic to $\mathscr{D}' = (D', \leqslant, l_{E^*}, l_{S'})$, for some event-labelling function l_{E^*} making $\mathscr{D}'$ a labelled domain. A reduction $(f, g): \mathscr{A} \to \mathscr{A}'$ will be called a *state-reduction*, if g is bijective, and an *event-reduction*, if f is bijective. First we study state-congruences for a given event-labelled domain.

Lemma 4.1. *Let $(D, \leqslant, l_E)$ be an event-labelled domain. Then the system $(\mathscr{S}, \subseteq)$ of all nice state-congruences for $(D, \leqslant, l_E)$, ordered under inclusion, forms a complete lattice. In particular, $(\mathscr{S}, \subseteq)$ contains a smallest and a greatest element.*

Proof. Clearly, the state-congruence corresponding to the identity function id_{D^0} is the smallest element of $(\mathscr{S}, \subseteq)$. Now let $\phi \neq \mathscr{C} \subseteq \mathscr{S}$. Let $\equiv$ be the transitive closure of $\bigcup_{C \in \mathscr{C}} C$. It is easy to see that $\equiv$ is a state-congruence for $(D, \leqslant, l_E)$. To show that $\equiv$ is nice, let $x, y_i, z, x', y_i', z_i' \in D^0$ with $[x, y_i] \equiv_E [x', y_i']$, $[y_i, z] \equiv_E [y_i', z_i']$ for $i = 1, 2$, $y_1 \neq y_2$ and $x \equiv x'$. There are $x_0, \ldots, x_n \in D^0$ such that $x = x_0$, $x' = x_n$ and $x_j \equiv_j x_{j+1}$ for some $\equiv_j \in C$ $(j = 0, \ldots, n-1)$. Since each $\equiv_j$ is a nice state-congruence, inductively we obtain elements $y_{j,1}, y_{j,2}, z_j \in D^0$ $(j = 0, \ldots, n)$ with $y_i = y_{0,i}$, $z = z_0$ and $[x_j, y_{j,i}] \equiv_E [x_{j+1}, y_{j+1,i}]$, $[y_{j,i}, z] \equiv_E [y_{j+1,i}, z]$ and $y_{j,1} \neq y_{j,2}$ for $i = 1, 2$ and $j = 0, \ldots, n-1$. It follows that $[x', y_i'] \equiv_E [x', y_{n,i}]$ and $[y_i', z_i'] \equiv_E [y_{n,i}, z_n]$, so $y_i' = y_{n,i}$ and thus $z_i' = z_n$ for $i = 1, 2$.

Clearly $\equiv$ is the smallest state-congruence for $(D, \leqslant, l_E)$ containing all congruences in $\mathscr{C}$. Hence $\equiv$ is equal to $\sup \mathscr{C}$ in $(\mathscr{S}, \subseteq)$. So $(\mathscr{S}, \subseteq)$ is a complete lattice with greatest element $\sup \mathscr{S}$. We just note that the infimum of a nonempty subset $\mathscr{C}$ of $\mathscr{S}$ in $(\mathscr{S}, \subseteq)$ is simply the intersection of $\mathscr{C}$, as is easy to see. □

Under the assumptions of Lemma 4.1, we will denote by $\equiv_{D^0}$ the smallest state-congruence for $(D, \leqslant, l_E)$. Let $l_S, l_{S'}$ be two state-labelling functions for $(D, \leqslant, l_E)$, and let $\equiv$, $\equiv'$ be the corresponding state-congruences. We note for later purposes that clearly $\equiv \subseteq \equiv'$ iff there is a function $f: S \to S'$ with $l_{S'} = f \circ l_S$. Then (id_D, f, id_E) is a morphism from $(D, \leqslant, l_E, l_S)$ to $(D, \leqslant, l_E, l_{S'})$. The following is straightforward.

Lemma 4.2. *Let $\mathscr{D} = (D, \leqslant, l_E, l_S)$ be a labelled domain and $(D', \leqslant, l_{E'})$ an event-labelled domain. Assume $(\varphi, g): (D', \leqslant, l_{E'}) \to (D, \leqslant, l_E)$ is an isomorphism, and let $l_S' = l_S \circ \varphi|_{D'^0}$. Then $\mathscr{D}' = (D', \leqslant, l_{E'}, l_S')$ is a labelled domain, and $(\varphi, id_S, g): \mathscr{D}' \to \mathscr{D}$ is an isomorphism.*

Now we can show the following result about the existence of maximal and minimal automata generating a given weak concurrency domain.

Theorem 4.3. *Let $(D, \leqslant, l_E)$ be an event-labelled weak concurrency domain. Then there exist two cancellative automata $\mathscr{A}_{\max}$ and $\mathscr{A}_{\min}$, each generating $(D, \leqslant, l_E)$, with the following two properties.*

(1) *For any cancellative automaton $\mathscr{A}$ generating $(D, \leqslant, l_E)$ there exist state-reductions from $\mathscr{A}_{\max}$ to $\mathscr{A}$ and from $\mathscr{A}$ to $\mathscr{A}_{\min}$.*

(2) *If $\mathscr{A}$ is any automaton with concurrency relations having only reachable states, then any reduction from $\mathscr{A}$ to $\mathscr{A}_{\max}$ is an event-reduction. Any state-reduction from $\mathscr{A}_{\min}$ to $\mathscr{A}$ is an isomorphism.*

In particular, $\mathscr{A}_{\max}$ and $\mathscr{A}_{\min}$ are unique up to isomorphism with property (1).

Proof. Let id_{D^0} be the trivial state-labelling function, and let $\mathscr{D}_0=(D, \leqslant, l_E, id_{D^0})$. By Lemma 4.1 there exists a greatest state-congruence $\equiv_{S_m}$ such that $\mathscr{D}_m=(D, \leqslant, l_E, l_{S_m})$ is nicely labelled, where l_{S_m} is the state-labelling function on D^0 corresponding to $\equiv_{S_m}$. Let $\mathscr{A}_{\max}=\boldsymbol{A}(\mathscr{D}_0)$ and $\mathscr{A}_{\min}=\boldsymbol{A}(\mathscr{D}_m)$. By Corollary 3.14(b) and Proposition 3.13, $\mathscr{A}_{\max}$ and $\mathscr{A}_{\min}$ are each cancellative and generate $(D, \leqslant, l_E)$.

Now let $\mathscr{A}$ be a cancellative automaton generating $(D, \leqslant, l_E)$. Let $\boldsymbol{D}(\mathscr{A})=(D(\mathscr{A}), \leqslant, l_{E'}, l'_S)$ be its induced (nicely labelled) domain. Since $(D(\mathscr{A}) \leqslant, l_{E'})$ is isomorphic to $(D, \leqslant, l_E)$, by Lemma 4.2 there is a state-labelling function $l_S: D^0 \to S$ such that $\boldsymbol{D}(\mathscr{A})$ is isomorphic to $\mathscr{D}=(D, \leqslant, l_E, l_S)$ which is hence nicely labelled. If $\equiv_S$ is the state congruence corresponding to l_S, clearly we have $\equiv_{D^0} \subseteq \equiv_S \subseteq \equiv_{S_m}$. So (id_D, l_S, id_E) is an order-morphism from $\mathscr{D}_0$ to $\mathscr{D}$, and there is also an order-morphism (id_D, f, id_E) from $\mathscr{D}$ to $\mathscr{D}_m$. Applying the functor $\boldsymbol{A}$, by Proposition 3.10 we get state-reductions (l_S, id_E) from $\mathscr{A}_{\max}$ to $\boldsymbol{A}(\mathscr{D})$ and (f, id_E) from $\boldsymbol{A}(\mathscr{D})$ to $\mathscr{A}_{\min}$. But by Proposition 3.15, $\mathscr{A}=\boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A})$, which is isomorphic to $\boldsymbol{A}(\mathscr{D})$. Hence property (1) follows.

To check property (2), let $\mathscr{A}$ be any automaton with concurrency relations having only reachable states and let (f, g) be either (a) a reduction from $\mathscr{A}$ to $\mathrm{A}_{\max}$; or (b) a state reduction from $\mathrm{A}_{\min}$ to $\mathscr{A}$. Then $\mathscr{A}$ is cancellative by Proposition 3.7, and $\boldsymbol{D}(\mathscr{A})=(D(\mathscr{A}), \leqslant, l'_{E'}, l'_S)$ is a nicely labelled domain. Applying the functor $\boldsymbol{D}$, in case (a) we get an order-morphism (φ, f, g) from $\boldsymbol{D}(\mathscr{A})$ to $\boldsymbol{D}(\mathscr{A}_{\max})$, and by Proposition 3.13 there is an isomorphism $(\pi_{\mathscr{D}}^{-1}, id_{D^0}, id_E)$ from $\boldsymbol{D}(\mathscr{A}_{\max})$ to $\mathscr{D}_0$. Thus, $\pi_{\mathscr{D}}^{-1} \circ \varphi|_{D^0(\mathscr{A})}=f \circ l'_S$, so f (and l'_S) must be injective. Hence (f, g) is an event-reduction. In case (b), similarly we obtain an order-morphism (ψ, f, g) from $\mathscr{D}_m$ to $\boldsymbol{D}(\mathscr{A})$. Then, by Lemma 4.2, $l_S^*=l'_S \circ \psi|_{D^0}$ is a state-labelling function with which $(D, \leqslant, l_E, l_S^*)$ is nicely labelled. Since $l_S^*=f \circ l_{S_m}$, we obtain that the congruence corresponding to l_S^* contains, and hence equals $\equiv_{S_m}$, showing that f is injective. So, (f, g) is an isomorphism.

For the final statement, let $\mathscr{A}^*$ be a cancellative automaton with property (1) as, say, for $\mathscr{A}_{\max}$ (for $\mathscr{A}_{\min}$, the argument is analogous). Then there exists a state-reduction from $\mathscr{A}^*$ to $\mathscr{A}_{\max}$ which, by property (2), must be an event-reduction and thus an isomorphism. So $\mathscr{A}^* \cong \mathscr{A}_{\max}$. □

Theorem 4.3 shows that the class of all cancellative automata $\mathscr{A}$ having a fixed (up to bijective relabelling) event set E and generating a given weak concurrency domain $(D, \leqslant)$ contains, with respect to state-reductions, a greatest automaton $\mathscr{A}_{\max}$ and a smallest automaton $\mathscr{A}_{\min}$. Moreover, these two automata are unique up to isomorphism. In combination with Lemma 4.1, the proof also gives us an explicit description of these automata.

In this context we note that if $(f, g): \mathscr{A} \to \mathscr{A}'$ and $(f', g'): \mathscr{A}' \to \mathscr{A}$ are state-reductions between cancellative automata $\mathscr{A}$, $\mathscr{A}'$, it does *not* follow that (f, g) and (f', g') are isomorphisms. Indeed, let $\mathscr{A} = (S, E, T, *, \|)$ where

(1) $S = \{*, a_n, b_m : n, m \in \mathbb{Z}, m \leqslant 0\}$,

(2) $E = \{e_n : n \in \mathbb{Z}\}$,

(3) $T = \{(*, e_n, a_n), (a_n, e_n, b_n), (b_n, e_n, b_n) : n \leqslant 0\} \cup \{(*, e_n, a_n), (a_n, e_n, a_n) : n > 0\}$,

(4) $\| = (\|_s)_{s \in S}$ with $\|_s = \emptyset$ for each $s \in S$.

Define $f: S \to S$ by $f(*) = *$, $f(a_n) = a_{n+1}$ for each $n \in \mathbb{Z}$, $f(b_m) = b_{m+1}$ for each $m < 0$, and $f(b_0) = a_1$. Let $g(e_n) = e_{n+1}$ for each $n \in \mathbb{Z}$. Then $(f, g): \mathscr{A} \to \mathscr{A}$ is a state-reduction, but not an isomorphism.

However, if $(f, g): \mathscr{A} \to \mathscr{A}'$ and $(f', g'): \mathscr{A}' \to \mathscr{A}$ are state-reductions and $\mathscr{A}$ is finite then $|\mathscr{A}| = |\mathscr{A}'|$ since f, f' are onto, and thus f, f' are bijective. From this it easily follows that (f, g) and (f', g') are isomorphisms.

Next we wish to derive a similar result as Theorem 4.3 for cancellative automata $\mathscr{A}$ with a fixed state set S and generating a given weak concurrency domain $(D, \leqslant)$. We first show that the canonical automaton associated with $(D, \leqslant)$ is the greatest element of this class, with respect to event-reductions.

Theorem 4.4. *Let $\mathscr{D} = (D, \leqslant, l_E, l_S)$ be a nicely labelled domain, where l_E denotes the canonical event-labelling function on $(D, \leqslant)$. Then $\boldsymbol{A}(\mathscr{D})$ is cancellative, generates $(D, \leqslant, l_S)$ and has the following properties.*

(1) *For any cancellative automaton $\mathscr{A}$ generating $(D, \leqslant, l_S)$ there exists an event-reduction from $\boldsymbol{A}(\mathscr{D})$ to $\mathscr{A}$.*

(2) *If $\mathscr{A}$ is any cancellative automaton, then any reduction from $\mathscr{A}$ to $\boldsymbol{A}(\mathscr{D})$ is a state-reduction.*

In particular, $\boldsymbol{A}(\mathscr{D})$ is unique up to isomorphism with property (1).

Proof. By Corollary 3.14(b), $\boldsymbol{A}(\mathscr{D})$ is cancellative, and by Proposition 3.13, it generates $\mathscr{D}$. Now let $\mathscr{A}$ be any cancellative automaton generating $(D, \leqslant, l_S)$. Then there exists an event-labelling function $l_{E'}$ on $(D, \leqslant)$ such that $\boldsymbol{D}(\mathscr{A})$ is isomorphic to $\mathscr{D}' = (D, \leqslant, l_{E'}, l_S)$, which is hence nicely labelled. If $g: E \to E'$ is defined by $g([x, y]_{\prec\succ}) = l_{E'}([x, y])$ (for $[x, y] \in Int_D$), we obtain an order-morphism $(id_D, id_S, g): \mathscr{D} \to \mathscr{D}'$. By Proposition 3.10, $(id_S, g): \boldsymbol{A}(\mathscr{D}) \to \boldsymbol{A}(\mathscr{D}')$ is a reduction. Since $\boldsymbol{A}(\mathscr{D}') \cong \boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A}) = \mathscr{A}$ by Proposition 3.15, we have thus established property (1).

To check property (2), let $\mathscr{A}$ be a cancellative automaton and $(f, g): \mathscr{A} \to \boldsymbol{A}(\mathscr{D})$ a reduction. By Theorem 3.6 and Proposition 3.13, there is an order-morphism (φ, f, g) from $\boldsymbol{D}(\mathscr{A}) = (D(\mathscr{A}), \leqslant, l_{E'}, l_{S'})$ to $\mathscr{D}$. In particular, for any $x, y \in D^0(\mathscr{A})$ with $x \prec\!\!\!\cdot\, y$ we have $g \circ l_{E'}([x, y]) = l_E([\varphi(x), \varphi(y)]) = [\varphi(x), \varphi(y)]_{\prec\succ}$. Hence, if $g(e_1') = g(e_2')$ and, say, $e_i' = l_{E'}([x_i, y_i])$ for $i = 1, 2$ then $[\varphi(x_1), \varphi(y_1)] \succ\!\!\prec [\varphi(x_2), \varphi(y_2)]$ in $(D, \leqslant)$, so $[x_1, y_1] \succ\!\!\prec [x_2, y_2]$ in $(D(\mathscr{A}), \leqslant)$, showing $e_1' = e_2'$. Thus, g is bijective, and so (f, g) is a state-reduction.

The uniqueness of $\boldsymbol{A}(\mathscr{D})$ can be checked similarly as for Theorem 4.3. □

Next we turn to the existence of minimal cancellative automata $\mathscr{A}$ with a fixed state set and generating a given weak concurrency domain. We first need some preparation.

Lemma 4.5. *Let $(D, \leqslant, l_E, l_S)$ be a nicely labelled domain, $l_{E'}$ an event-labelling of $(D, \leqslant)$, and $(id_D, g):(D, \leqslant, l_E)\to(D, \leqslant, l_{E'})$ an event-morphism. Then $(D, \leqslant, l_{E'}, l_S)$ is nicely labelled.*

Proof. Let $x, y_i, z, x', y_i', z_i' \in D^0$ with $x \overset{e_i'}{\prec} y_i \overset{e_{3-i}'}{\prec} z$ and $x' \overset{e_i'}{\prec} y_i' \overset{e_{3-i}'}{\prec} z_i'$ where $e_i' \in E'$ for $i=1,2$, $y_1 \neq y_2$ and $l_S(x)=l_S(x')$. Then in $(D, \leqslant, l_E)$ we have $x \overset{e_i}{\prec} y_i \overset{e_{3-i}}{\prec} z$ for some $e_i \in E$ $(i=1,2)$. Since (id_D, g) is an event-morphism, for $i=1,2$ we have $g(e_i)=e_i'$; also, there is $y_i^* \in D^0$ with $x' \overset{e_i}{\prec} y_i^*$ and $l_S(y_i)=l_S(y_i^*)$. Then $x' \overset{e_i'}{\prec} y_i^*$, so $y_i^*=y_i'$. Applying the same argument again, we get $y_i' \overset{e_{3-i}}{\prec} z_i'$. So $x' \overset{e_i}{\prec} y_i' \overset{e_{3-i}}{\prec} z_i'$ for $i=1,2$. Since $(D, \leqslant, l_E, l_S)$ is nicely labelled, we get $z_1'=z_2'$ as needed. □

Now we can show the following theorem.

Theorem 4.6. *Let $\mathscr{A}$ be any cancellative automaton and $\boldsymbol{D}(\mathscr{A})=(D, \leqslant, l_E, l_S)$. Then there exists a cancellative automaton $\mathscr{A}_{\min}$ generating $(D, \leqslant, l_S)$ with the following properties.*

(1) *There exists an event-reduction from $\mathscr{A}$ to $\mathscr{A}_{\min}$.*

(2) *Any reduction from $\mathscr{A}_{\min}$ to an automaton with concurrency relations is a state-reduction.*

Proof. Let $\equiv_E$ be the event-congruence corresponding to l_E. Since the union of any chain of event-congruences of $(D, \leqslant)$ is again an event-congruence, by Zorn's lemma there is a maximal event-congruence $\equiv$ on $(D, \leqslant)$ containing $\equiv_E$. Let M be the set of all $\equiv$-equivalence classes of Int_D, and let $l_M: Int_D \to M$ map each prime interval onto its $\equiv$-equivalence class. Since $\equiv_E \subseteq \equiv$, putting $g(e)=l_M([x, y])$ if $e=l_E([x, y])$ we obtain a well-defined function $g: E\to M$ with $l_M=g\circ l_E$. Then $\mathscr{D}=(D, \leqslant, l_M, l_S)$ is nicely labelled by Lemma 4.5. Put $\mathscr{A}_{\min}=\boldsymbol{A}(\mathscr{D})$. Clearly, $(id_D, id_S, g): \boldsymbol{D}(\mathscr{A})\to\mathscr{D}$ is an order-morphism. Hence, by Propositions 3.10 and 3.15, (id_S, g) is an event-reduction from $\mathscr{A}=\boldsymbol{A}\circ\boldsymbol{D}(\mathscr{A})$ to $\mathscr{A}_{\min}$, establishing (1).

To check (2), let $\mathscr{A}^*$ be any automaton with concurrency relations, and let $(f, g): \mathscr{A}_{\min}\to\mathscr{A}^*$ be a reduction. By Proposition 3.7, $\mathscr{A}^*$ is cancellative.By Proposition 3.13 and Theorem 3.6, there are order-morphisms $(\pi_{\mathscr{D}}, id_S, id_E): \mathscr{D}\to\boldsymbol{D}(\mathscr{A}_{\min})$ and $(\varphi, f, g): \boldsymbol{D}(\mathscr{A}_{\min})\to\boldsymbol{D}(\mathscr{A}^*)$. Let $\boldsymbol{D}(\mathscr{A}^*)=(D(\mathscr{A}^*), \leqslant, l_{E^*}, l_{S^*})$. Put $\psi=\varphi\circ\pi_{\mathscr{D}}$, an order-isomorphism from $(D, \leqslant)$ to $(D(\mathscr{A}^*), \leqslant)$. Then for any $x, y\in D^0$ with $x\prec y$ we have $l_{E^*}([\psi(x), \psi(y)])=g\circ l_M([x, y])$. Hence, defining $l'_{E^*}([x, y]):=l_{E^*}([\psi(x), \psi(y)])$ for $[x, y]\in Int_{\mathscr{D}}$, we obtain an event-labelling function l'_{E^*} on $(D, \leqslant)$ with $l'_{E^*}=g\circ l_M$. Hence, the event congruence $\equiv_{E^*}$ corresponding to l'_{E^*} contains $\equiv$ and thus equals $\equiv$, by maximality of $\equiv$. So, g is bijective. □

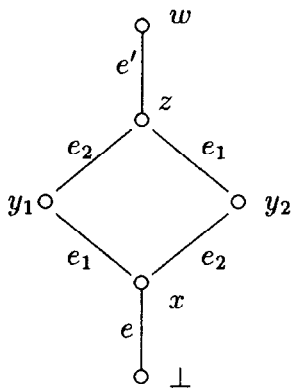

Fig. 6.

Note that in Theorem 4.3 we obtained a smallest (unique minimal) automaton in the considered class of automata with fixed event set, whereas in Theorem 4.6 we derived the existence (not uniqueness) of minimal automata with fixed state set and given domain. The following example shows that in general these minimal automata are *not* unique up to isomorphism. It is due to D. Kuske and a slight simplification of a similar example obtained previously by the authors.

Example 4.7. Let $(D, \leqslant, l_E)$ be the following event-labelled domain where $E=\{e_1, e_2, e, e'\}$ (see Fig. 6).

Let $l_S = id_{D^0}$ be the trivial state-labelling. Let $\mathscr{A}$ be any cancellative automaton with $\boldsymbol{D}(\mathscr{A}) \cong (D, \leqslant, l_E, l_S)$. Consider the following two event-labellings l_1, l_2 of $(D, \leqslant)$: $l_1([\perp, x]) = l_2([\perp, x]) = l_1([z, w]) = e_1, l_2([z, w]) = e_2, l_j([x, y_i]) = l_j([y_{3-i}, z]) = e_i$ for $i, j \in \{1, 2\}$. The corresponding event-congruences are clearly maximal event-congruences on $(D, \leqslant)$. Let $\mathscr{D}_j = (D, \leqslant, l_j, l_S)$ and $\mathscr{A}_j = \boldsymbol{A}(\mathscr{D}_j)$ for $j = 1, 2$. Then $\mathscr{A}_1$ and $\mathscr{A}_2$ are non-isomorphic and each satisfy the assertions of Theorem 4.6. (Moreover, it can be shown that any cancellative automaton satisfying these assertions is isomorphic to $\mathscr{A}_1$ or $\mathscr{A}_2$).

Now we will deal with the class of all cancellative automata generating a given weak concurrency domain. First we show that this class contains, with respect to reductions, a greatest automaton.

Theorem 4.8. *Let $(D, \leqslant)$ be a weak concurrency domain and $\mathscr{A}_D$ the canonical automaton associated with $(D, \leqslant)$. Let $\mathscr{A}$ be any cancellative automaton. Then*

(a) *if $\mathscr{A}$ generates $(D, \leqslant)$, then there exists a reduction from $\mathscr{A}_D$ to $\mathscr{A}$,*

(b) *any reduction $(f, g): \mathscr{A} \to \mathscr{A}_D$ is an isomorphism.*

Proof. (a) We can choose labelling functions l_E, l_S for $(D, \leqslant)$ such that $\mathscr{D}=(D, \leqslant, l_E, l_S)$ is isomorphic to the nicely labelled domain $\boldsymbol{D}(\mathscr{A})$. By Corollary 3.14(a) and Proposition 3.15, there is a reduction from $\mathscr{A}_D$ to $\boldsymbol{A}(\mathscr{D})$, and $\boldsymbol{A}(\mathscr{D}) \cong \boldsymbol{A} \circ \boldsymbol{D}(\mathscr{A}) = \mathscr{A}$. For an alternative argument (and for the proof of (b)), let l denote the canonical event-labelling function on $(D, \leqslant)$, and $\mathscr{D}'=(D, \leqslant, l, id_{D^0})$. Then $\mathscr{A}_D=\boldsymbol{A}(\mathscr{D}')$. The automaton $\mathscr{A}_{\max}$ constructed in the proof of Theorem 4.3 for $(D, \leqslant, l_E)$ generates $(D, \leqslant, id_{D^0})$, so by Theorem 4.4 there exists an (event-) reduction from $\mathscr{A}_D$ to $\mathscr{A}_{\max}$. Since $\mathscr{A}$ generates $(D, \leqslant, l_E)$, by Theorem 4.3 there is a (state-) reduction from $\mathscr{A}_{\max}$ to $\mathscr{A}$ and the result follows by composing these two reductions.

(b) Let $(f, g): \mathscr{A} \to \mathscr{A}_{\mathscr{D}}$ be a reduction. By Theorem 4.3, (f, g) is an event-reduction, and by Theorem 4.4 a state-reduction, hence an isomorphism. □

Finally, we consider the existence of minimal cancellative automata generating a given weak concurrency domain. For this, we employ our previous corresponding results (Theorems 4.3 and 4.6).

Theorem 4.9. *Let $(D, \leqslant)$ be a weak concurrency domain, and let $\mathscr{A}$ be any cancellative automaton generating $(D, \leqslant)$. Then there exists a cancellative automaton $\mathscr{A}_{\min}$ generating $(D, \leqslant)$ with the following properties.*

(1) *There exists a reduction from $\mathscr{A}$ to $\mathscr{A}_{\min}$.*

(2) *Any reduction from $\mathscr{A}_{\min}$ to an automaton with concurrency relations is an isomorphism.*

Proof. Let $\mathscr{A}_m$ be an automaton given by Theorem 4.6 for $\mathscr{A}$, and let $\mathscr{A}_{\min}$ be the automaton given by Theorem 4.3 for $(\boldsymbol{D}(\mathscr{A}_m), \leqslant, l_{E_m})$, the induced event-labelled domain of $\mathscr{A}_m$. We obtain, correspondingly, reductions from $\mathscr{A}$ to $\mathscr{A}_m$ and, say (f', g'), from $\mathscr{A}_m$ to $\mathscr{A}_{\min}$, proving (1). To check (2), let $\mathscr{A}^*$ be an automaton with concurrency relations and $(f, g): \mathscr{A}_{\min} \to \mathscr{A}^*$ a reduction. By Proposition 3.7, $\mathscr{A}^*$ is cancellative. By Theorem 4.6, the reduction $(f \circ f', g \circ g'): \mathscr{A}_m \to \mathscr{A}^*$ is a state-reduction, so g is bijective. Hence, (f, g) is a state-reduction and so, by Theorem 4.3, an isomorphism. □

As Example 4.7 shows, again the minimal automaton of Theorem 4.9 is in general not unique up to isomorphism.

Finally, we note that further results on the relationship between domains and event structures, and, respectively, between domains, monoids and nondeterministic automata with concurrency relations will be given in [6, 15].

Acknowledgment

This paper is an extended version of [5]. Most of this work was done while M. Droste was a visitor at the Universidad Nacional Autónoma de México in spring and summer 1992. He would like to thank his colleagues for their hospitality and a wonderful time.

References

[1] P. Bachmann and P.M. Dung, Nondeterministic computations – structure and axioms, *Elektron. Inf. verarb. Kybern. EIK* **22** (1986) 243–261.

[2] M. Bednarczyk, Categories of asynchronous systems, Ph.D. Thesis, University of Sussex, 1987.

[3] G. Berry and J.-J. Levy, Minimal and optimal computations of recursive programs, *J. ACM* **26** (1979) 148–175.

[4] G. Boudol and I. Castellani, A non-interleaving semantics for CCS based on proved transitions, *Fundam. Inform.* **11** (1988) 433–452.

[5] F. Bracho and M. Droste, From domains to automata with concurrency, in: 20th *ICALP*, Lecture Notes in Computer Science, Vol. 700 (Springer, Berlin 1993) 669–681.

[6] F. Bracho and M. Droste, From domains to event structures, to appear.

[7] P.L. Curien, *Categorical Combinators, Sequential Algorithms and Functional Programming*, Progress in Theoretical Computer Science (Birkhäuser, Boston, MA, 1993).

[8] V. Diekert: *Combinatorics on Traces*, Lecture Notes in Computer Science, Vol. 454 (Springer, Berlin, 1990).

[9] M. Droste, Concurrency, automata and domains, in: *17th ICALP*, Lecture Notes in Computer Science, Vol. 443 (Springer, Berlin, 1990) 195–208.

[10] M. Droste, Concurrent automata and domains, *Internat. J. Found. Comput. Sci.* **3** (1992) 389–418.

[11] M. Droste and R.M. Shortt, Petri nets and automata with concurrency relations – an adjunction, in: M. Droste and Y. Gurevich, eds. *Semantics of Programming Languages and Model Theory* (Gordon and Breach, OPA, Amsterdam, 1993) 69–87.

[12] C.A.R. Hoare, Communicating sequential processes, *Comm. ACM* **21** (1978) 666–676.

[13] G. Huet and J.-J. Levy, Call-by-need computations in non-ambiguous linear term rewriting systems, IRIA-LABORIA Report, Vol. 359, 1979.

[14] S. Katz and D. Peled, Defining conditional independence using collapses, in: M.Z. Kwiatkowska, M.W. Shields and R.M. Thomas, eds., *Semantics for Concurrency Proc. Internat. BCS-FACS Workshop at Leicester*, 1990 (Springer, Berlin, 1990) 262–280.

[15] D. Kuske, Modelle nebenläufiger Prozesse – Monoide, Residuensysteme und Automaten, Dissertation, Universität Essen, 1994.

[16] J.-J. Levy, Optimal reductions in the lambda calculus, in: J.P. Seldin and J.R. Hindley, eds., *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism* (Academic Press, New York, 1980) 159–191.

[17] A. Mazurkiewicz, Concurrent program schemes and their interpretations, DAIMI Report PB-78, Aarhus University, Aarhus, 1977.

[18] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin, 1980).

[19] P. Panangaden and E.W. Stark, Computations, residuals and the power of indeterminacy, in: *15th ICALP*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988, 439–454.

[20] M.W. Shields, Concurrent machines, *Comput. J.* **28** (1985) 449–465.

[21] E.W. Stark, Concurrent transition systems, *Theoret. Comput. Sci.* **64** (1989) 221–269.

[22] E.W. Stark, Compositional relational semantics for indeterminate dataflow networks, in: *Proc. Category Theory and Computer Science*, Lecture Notes in Computer Science, Vol. 389 (Springer, Berlin, 1989) 52–74.

[23] E.W. Stark, Connections between a concrete and an abstract model of concurrent systems, in: *5th Conf. on the Mathematical Foundations of Programming Semantics*, Lecture Notes in Computer Science, Vol. 442 (Springer, Berlin, 1989) 53–79.

[24] E.W. Stark, Dataflow networks are fibrations, in: *Category Theory and Computer Science Proc.*, Paris, Lecture Notes in Computer Science, Vol. 530 (Springer, Berlin, 1991) 261–281.

[25] G. Winskel, Event structures, in: *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science, Vol. 255 (Springer, Berlin, 1987) 325–392.

[26] G. Winskel and M. Nielsen, Models for concurrency, draft copy of Oct. 1991, in: S. Abramsky, D.M. Gabbay and T.S.E. Maibaum, eds., *Handbook of Logic in Computer Science*, to appear.