A noninterleaving semantics for communicating sequential processes: a fixed-point approach

D. V. HUNG, E. KNUTH

Abstract

The paper presents a noninterleaving semantics for Communicating Sequential Processes introduced by Hoare and studied in many works. Concurrency is expressed explicity in the introduced model. Furthermore, semantics of CSP-programs can be obtained by equations in the model. By relating the model to labelled event structures and Petri nets the relationship between CPS and the mentioned models is pointed out.

Key words: CPS programs, concurrency, traces, semiwords, event structures, synchronization.

1. Introduction

In 1978 C. A. R. Hoare introduced in [6] a language for distributed programming called Communicating Sequential Processes — in short CSP. Subsequently, this language has received a great deal of attention. As mentioned in [4], both ADA and OCCAM are based upon CSP.

Many models of semantics for CSP have been proposed. Among them we should mention Hoare's interleaving of strings [7, 8], Gostz's and Reisig's Petri nets with individual tokens [4], Janicki's semitraces [11], Hennessy's (et al.) operational model [5]. In the model of interleaving semantics concurrency is represented by the possibility of shuffling sequences of operations, and thus is not expressed in an explicit form. Furthermore, concurrency is not distinguishable from nondeterminism in this model. Janicki [11] has used Mazuskiewicz's traces to give semantics for CSP, in which concurrency can be distinguished from nondeterminism. The possibility of handling with traces as words makes analyzing properties of CSP-programs in Janicki's model as easy as in Hoare's model. However, as shown by him, traces cannot be used to give semantics for all CPS-programs. The notion of so-called semitraces was introduced by Janicki in order to describe the behaviour of all CPS-programs. Although his semitraces are powerful enough to describe the behaviour of CSP, they have a disadvantage that they cannot be represented by single words.

In our paper [8] we have developed the notion of Mazurkiewicz's traces to a new one based upon the notion of Starke's semiwords. This is the notion of labelled

traces. As pointed out in [8], labelled traces have the same advantage as and are more powerful than traces. They can describe the behaviour of concurrent systems modelled by bounded Petri nets, e.g. producer-consumer systems, while traces cannot.

In the present paper, with the same point of view as in Gorts and Reisig's, Janiki's papers [4], [11], we construct a model of semantics for concurrent systems, more specifically, CSP by using labelled traces. The advantage of Mazurkiewicz's model [14] is taken to this model. By relating the model to Starke's one the relationship between CSP and finite event structures is pointed out. Combining with the results presented in [8], CSP are related to Petri nets as well.

In this paper the basic notion of Mazurkiewicz's traces is used and understood as follows.

For a finite alphabet X, X^* denotes the set of all finite strings over X, ε denotes the empty word, a subset of X^* is called a language over X; a reflexive and symmetric relation on X is called a dependency on X. For a given dependency D on X, let $\equiv_D$ be the least congruence on X^* w.r.t. the concatenation of strings such that $ab \equiv_D ba$ for all $a, b \in X$ with $(a, b) \notin D$. Each equivalence class of $\equiv_D$ is called a trace over D, and a set of traces over D is called a trace language over D.

The paper is organized as follows.

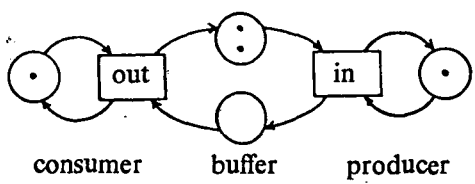
The second section presents a model of semantics for concurrent systems by introducing the notion of labelled traces. The third section is devoted to a study on labelled trace languages. The introduced model is related to other models in the fourth section. A noninterleaving semantics for CSP based upon labelled traces is presented in the fifth one.

2. Labelled trace languages

Let us consider the following problem (bounded buffer [6]):

Construct a buffering process X to smooth variations in the speed of output of portions by a producer process and input by a consumer process. The buffer should contain up to two portions.

A solution of the problem is represented by a 2-bounded Petri net as follows.



Suppose that, at the begining, the buffer is empty so that only the action "in" of the producer can be executed for the first time. After that, the action "out" can occur the first time and the action "in" can occur the second time concurrently. However the first occurrence of "out" depends causally on the first occurrence of "in". Thus, actions "in" and "out" have different cases, and we should take them for atomic actions. For the sake of simplicity sets of atomic actions is assumed to be finite. From the 2-boundedness of the buffer process, every occurrence of one action is concurrent

with not more than one occurrence of the other. Hence, we can use a dependency on a set of four elements to construct a set of labelled partial orderings for describing the behaviour of the above solution.

Basing on the above notice and the theory of Mazurkiewicz's traces we introduce a new description of concurrent systems presented bellow.

All the notions introduced in this section have been presented in our paper [8] in detail. Here, for the aim of the paper we present them in a different form for the sake of convenience.

Let X be a finite alphabet. A finite symmetric relation $D \subseteq (X \times \{1, 2, \ldots\}) \times (X \times \{1, 2, \ldots\})$ such that

a) $(a, i) \in \mathrm{dom}(D) \Rightarrow (a, j) \in \mathrm{dom}(D)$ for all $j \leqq i$, and

b) $((a, j), (a, i)) \in D$ for all $(a, i), (a, j) \in \mathrm{dom}(D)$ will be called a *labelled dependency* over X.

Let D be a labelled dependency over X. Define $\equiv_D$ as the least congruence over $(\mathrm{dom}(D))^*$ (w.r.t. the concatenation) such that $(a, j)(b, i) \equiv_D (b, i)(a, j)$ for all $(a, j), (b, i) \in \mathrm{dom}(D)$ and $((a, j), (b, i)) \notin D$. Each equivalence class of $\equiv_D$ will be called a *labelled trace* over the labelled dependency D, and a set of labelled traces over D will be called a *labelled trace language* over D. Like in the case of traces $[w]_D$ will denote the labelled trace generated by a string $w \in (\mathrm{dom}(D))^*$, and $[L]_D$ will denote the labelled trace language generated by a language $L \subseteq (\mathrm{dom}(D))^*$, i.e.

$$[w]_D := \{v\colon v \in (\mathrm{dom}(D))^* \,\&\, v \equiv_D w\},$$

$$[L]_D := \{[v]_D\colon v \in L\}.$$

For $(a, i) \in \mathrm{dom}(D)$, (a, i) will be called a case of a in D and we denote by $\#(a, D)$ the number $\max\{i\colon (a, i)$ is a case of a in $D\}$. Throughout this paper l always denotes the projection from cases to their first component.

Remark 1. If we identify X with $X \times \{1\}$, each dependency over X (defined by Mazurkiewicz) is a labelled dependency over X. On the other hand each labelled dependency is a special dependency on $X \times \{1, 2, \ldots\}$. Thus, all the notions and results obtained in the theory of traces [1], [13], [14] can be applied to labelled traces and labelled trace languages. This means that we can handle with labelled traces as traces, and the advantage of traces and trace languages is taken to labelled traces and labelled trace languages. The only difference between traces and labelled traces is how the atomic actions are considered.

To show the difference between our notion and Mazurkiewicz's one we consider the dependency graphs of labelled traces.

Definition 1. Let D be a labelled dependency over X, $w \in (\mathrm{dom}(D))^*$. A dependency graph of w (over D) abbreviated a dep-graph of w (over D) and denoted by $D(w)$, is a graph isomorphic to the node-labelled graph (V, E, X, β), defined by:

$$V = \{1, 2, \ldots, n\} \quad \text{if} \quad w = x_1 x_2 \ldots x_n, \quad \beta(i) = l(x_i), \quad \text{and}$$

$$E \subseteq V \times V \quad \text{is such that, for all } 1 \leqq i, j \leqq n, \quad (i, j) \in E$$

if and only if $i < j$ and $(x_i, x_j) \in D$.

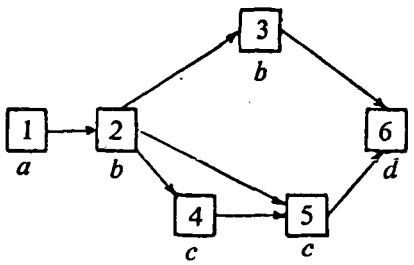
For the sake of convenience by syre (R) we denote the symmetrical and reflexive closure of a binary relation R on dom (R) throughout the paper.

Example 1. Let $X=\{a, b, c, d\}$,

$$D = \text{syre}\big(\{((a, 1), (b, 1)), ((b, 1), (b, 2))((b, 1), (c, 1)), ((b, 2), (d, 1)), ((c, 1), (d, 1))\}\big),$$

$$w = (a, 1)(b,1)(b, 2)(c, 1)(c, 1)(d, 1).$$

$D(w)$ has the following form:



Notice that this node-labelled graph can not be a dep-graph of any trace over any dependency on X. The reason is that the occurrences of c depend on the first occurrence of b, but are concurrent with the second occurrence of b.

It can be seen from the theory of traces that

$$w \equiv_D v \Rightarrow D(w) \cong D(v).$$

(Unlike in the case of dep-graphs of traces, the converse direction is not true!)

Hence, it is reasonable to define a *dep-graph of a labelled trace* t over D as $D(w)$ for any $w \in t$. Not distinguishing labelled traces, dep-graphs of which are isomorphic, we define that labelled traces t and t' over D are *isomorphic* iff $D(t) \cong D(t')$, where $D(u)$ denotes a dep-graph of a labelled trace u over D.

Clearly, each dep-graph of a labelled trace is acyclic, and its transitive closure is a labelled partial ordering over X, which will be called a *labelled partial ordering generated* or *induced* by a labelled trace, and in which any pair of nodes with the same label is ordered. Hence, our notion is related to Starke's one of semiwords (see the section 4).

As mentioned in Remark 1, all the notions of traces are applied to labelled traces. Here we remention some of them, which are needed in the sequel.

Let D be a labelled dependency over X. The *trace concatenation* and *trace iteration* of labelled traces and labelled trace languages are defined by

$[x]_D[y]_D := [xy]_D$ for $x, y \in (\text{dom}\,(D))^*$,

$UV := \{uv \mid u \in U\ v \in V\}$ for labelled trace languages U and V over D.

$U^* := \bigcup_{i=0}^{\infty} U^i$, $U^0 = [\varepsilon]_D$, $U^{i+1} = U^i U$, $i \geqq 0$, for a labelled trace language U over D.

The following lemma is needed for a purpose of constructing operations on labelled trace languages.

Lemma 1. Let D and D' be labelled dependencies on X, $h: (\mathrm{dom}\,(D'))^* \to (\mathrm{dom}\,(D))^*$ be a homomorphism satisfying:

a) $\forall x \in \mathrm{dom}\,(D')$: $h(x) \in \mathrm{dom}\,(D) \cup \{\varepsilon\}$,

b) $\forall x, y \in \mathrm{dom}\,(D')$: $((x, y) \notin D' \,\&\, h(x) \neq \varepsilon, h(y) \neq \varepsilon) \Rightarrow (h(x), h(y)) \notin D$.

Then, for any labelled trace u over D', $w \in u$ we have

$$h(u) \subseteq [h(w)]_D.$$

Proof. Let u be a labelled trace over D' and $w \in u$. Since h is a homomorphism, we have only to prove that if $w = w_1 xyw_2$, $w' = w_1 yxw_2$, $(x, y) \notin D'$, then $h(w) \equiv_D h(w')$. But this is obvious from the specified property of h.

Hence, each mapping $h: (\mathrm{dom}\,(D'))^* \to (\mathrm{dom}\,(D))^*$ satisfying the condition of Lemma 1 can be considered as a homomorphism from a labelled trace language over D' to a labelled trace language over D as well.

We shall adopt Mazurkiewicz's denotation. Let D be a labelled dependency over X.

$$A(D) := \{[a]_D : a \in \mathrm{dom}\,(D)\},$$

$$T(D) := \{[w]_D : w \in (\mathrm{dom}\,(D))^*\}, \quad \text{and}$$

$$P(D) := 2^{T(D)}.$$

Having in mind our intended interpretation, elements of $A(D)$ will be called *actioncases* over D, those of $T(D)$ *processes* over D and those of $P(D)$ *activities* over D. Actioncases a and b occur *concurrently* in a process t if $t = t'[ab]t''$ where $(a, b) \notin D$.

From Remark 1 and the definition of dep-graphs, if we identify X with $X \times \{1\}$, and a word over X with a trace over the dependency $D = (X \times \{1\}) \times (X \times \{1\})$ in the obvious way, we have that each word, each trace, and each labelled trace induce a labelled partial ordering over X. Let $W(X)$, $T(X)$ and $LT(X)$ denote classes of labelled partial orderings induced by words, traces, and labelled traces, respectively, on X. Clearly,

$$W(X) \subsetneq T(X) \subsetneq LT(X).$$

We have introduced cases of actions in order to expand the power of our model comparing to Mazurkiewicz's model of traces. Thus, from the intended meaning, we should not distinguish cases having the same effect in a labelled dependency. Therefore, only reduced labelled dependencies are considered. Formally, we introduce the following notions.

Definition 2. Let D and D' be labelled dependencies on X, T and T' labelled trace languages over D and D', resp.

(i) D and D' are said to be isomorphic, denoted by $D \cong D'$, if there exists a mapping φ from dom (D) onto dom (D') satisfying:

(i) φ preserves cases of actions, i.e. if x is a case of an action a, so is $\varphi(x)$,

(ii) φ preserves the dependence, i.e. $(x, y) \in D$ iff $(\varphi(x), \varphi(y)) \in D'$.

(ii) T and T' are said to be isomorphic, denoted by $T \cong T'$, iff $\forall t \in T$, $\exists t' \in T'$ such that $D(t) \cong D(t')$ and vice versa.

Definition 3. Let D be a labelled dependency on X.

(i) Cases (a, i) and (a, j) are said to be equivalent iff $\forall (b, k) \in \mathrm{dom}\,(D)$: $((b, k), (a, i)) \in D \Leftrightarrow ((b, k), (a, j)) \in D$.

(ii) A labelled dependency D' on X is said to be a reduced version of D iff D and D' are isomorphic and D' is reduced, i.e. for $(a, i), (a, j) \in \mathrm{dom}(D')$ with $i \neq j$ (a, i) and (a, j) are not equivalent.

Proposition 1. Every labelled dependency on X has its reduced version.

Proposition 2. Let D be a labelled dependency on X and T a labelled trace language over D. Assume that D is isomorphic to D' by an isomorphism φ. Then T and $\varphi(T)$ are isomorphic.

The above propositions follow immediately from the definitions 2 and 3.

3. Operations on labelled trace languages (on activities)

In the previous section we have defined some operations on labelled trace languages over a given labelled dependency. Those operations have restricted applications, as pointed out by Janicki [11], since concurrency relations are fixed. To improve upon this shortcomings we define our operations corresponding to ones on concurrent processes from Milner's and Hoare's works [7], [15].

In the sequel, let X be an alphabet, D_i a labelled dependency over X, and let t_i and U_i, respectively, be labelled traces and labelled trace languages over D_i, $i = 1, 2$.

a) *Sequential concatenation and concurrent composition.*

We intend to use the sequential concatenation to represent the fact that a process in U_2 starts only when a process in U_1 has terminated. By the concurrent composition, we shall represent a synchronization of processes corresponding to the synchronization mechanism introduced by Hoare [2], [7], Mazurkiewicz [14], and in our papers [8], [9], [10]. By this operation we want to construct a process t from t_1 and t_2, which behaves like t_1 and t_2, progressing in parallel and simultaneously participating in actions having cases in D_1 and D_2.

Having in mind our attention, we define some operations on labelled dependencies as follows.

Sequential composition of D_1 and D_2, denoted by $S(D_1, D_2)$, is the labelled dependency:

$$S(D_1, D_2) := D_1 \cup \{((a, i + \#(a, D_1)), (b, j + \#(b, D_1))) \mid ((a, i), (b, j)) \in D_2\} \cup$$

$$\cup \mathrm{syre}(\{((a, i), (b, j + \#(b, D_1))) \mid (a, i) \in \mathrm{dom}(D_1), (b, j) \in \mathrm{dom}(D_2)\}).$$

Together with $S(D_1, D_2)$ a mapping s from $T(D_2)$ to $T(S(D_1, D_2))$ is defined by

$$s((a, i)) = (a, i \# (a, D_1)).$$

The definition of s is reasonable by Lemma 1,

Concurrent composition of D_1 and D_2, denoted by $C(D_1, D_2)$ is a labelled dependency defined as follows. Let $Y = l(\mathrm{dom}(D_1)) \cap l(\mathrm{dom}(D_2))$ be the set of actions having cases in both D_1 and D_2. Then,

$$\mathrm{dom}(C(D_1, D_2)) := \{x \mid x \in \mathrm{dom}(D_1) \cup \mathrm{dom}(D_2) \,\&\, l(x) \notin Y\} \cup$$

$$\cup \{(a, i) \mid a \in Y \,\&\, i \leqq \#(a, D_1) \cdot \#(a, D_2)\}.$$

For two positive integers m, n let en (n, m) and rem (n, m) stand for the quotient and remainder of dividing n by m. For $i=1, 2$, define mappings

$$h_i\colon \operatorname{dom}(C(D_1, D_2)) \to \operatorname{dom}(D_i)\cup\{\varepsilon\} \quad \text{as follows.}$$

For $a\notin Y$, $i=1, 2$,

$$h_i((a, j)) = \begin{cases}(a, j) & \text{if } (a, j)\in\operatorname{dom}(D_i),\\ \varepsilon & \text{otherwise;}\end{cases}$$

for $a\in Y$

$$h_1((a, j)) = (a, \operatorname{rem}(j-1, \#(a, D_1))+1),$$

$$h_2((a, j)) = (a, \operatorname{en}(j-1, \#(a, D_1))+1).$$

Now, $C(D_1, D_2)$ is defined by

$C(D_1, D_2)=\{(x, y)|$ there exists an i in $\{1, 2\}$ such that $(h_i(x), h_i(y))\in D_i\}$.

Since h_1 and h_2 satisfy the condition of Lemma 1, h_1 and h_2 can be extended to homomorphisms from $T(C(D_1, D_2))$ to $T(D_1)$ and $T(D_2)$ respectively. h_1 and h_2 will be called the projections associated with $C(D_1, D_2)$.

Now, we are ready to define our operations on labelled trace languages.

Definition 4.

(i) The sequential concatenation $U_1\circ U_2$ of U_1 and U_2 is a labelled trace over $S(D_1, D_2)$ defined by $U_1\circ U_2=U_1 s(U_2)$, where U_1 is considered as a labelled trace language over $S(D_1, D_2)$ and the trace concatenation in the right side is for labelled trace languages over $S(D_1, D_2)$.

(ii) The concurrent composition $U_1\| U_2$ of U_1 and U_2 is a labelled trace language over $C(D_1, D_2)$ defined by:

$$U_1\| U_2 = \{t\in T(C(D_1, D_2))\,|\, h_1(t)\in U_1, h_2(t)\in U_2\}.$$

(iii) Sequential iteration (iteration for short) of U_1, denoted by $U_1^{\otimes}$, is defined by

$$U_1^{\otimes} = ((U_1-\{[\varepsilon]_{D_1}\})\circ(U_1-\{[\varepsilon]_{D_1}\}))^*(U_1\cup\{[\varepsilon]_{S(D_1, D_1)}\}),$$

where $U_1\cup\{[\varepsilon]_{S(D_1, D_2)}\}$ is considered as a labelled trace language over $S(D_1, D_1)$, and the trace iteration and trace concatenation are for labelled trace languages over $S(D_1, D_1)$.

When U_1 and U_2 contain a single element, say $U_1=\{t_1\}$, $U_2=\{t_2\}$, we write $t_1\circ t_2$, $t_1\| t_2$ instead of $\{t_1\}\circ\{t_2\}$, $\{t_1\}\|\{t_2\}$ resp.

Example 2.

(i) Let $D_1=\operatorname{syre}(\{((a, 1), (b, 1))\})$,

$$U_1 = [\operatorname{pref}((b, 1)(a, 1))^*]_{D_1},$$

$$D_2 = \operatorname{syre}(\{((b, 1), (c, 1)), ((b, 2), (c, 2)), ((b, 1), (b, 2)), ((c, 1), (c, 2))\})$$

$$U_2 = [\operatorname{pref}((c, 1)(b, 1)+(c, 2)(b, 2))^*]_{D_2}.$$

Then,

$$C(D_1, D_2) = \text{syre}(\{((a, 1), (b, 1)), ((a, 1), (b, 2))\}) \cup D_2,$$

$$h_1((a, 1)) = (a, 1), \quad h_1((b, 1)) = h_1((b, 2)) = (b, 1),$$

$$h_1((c, 1)) = h_1((c, 2)) = \varepsilon$$

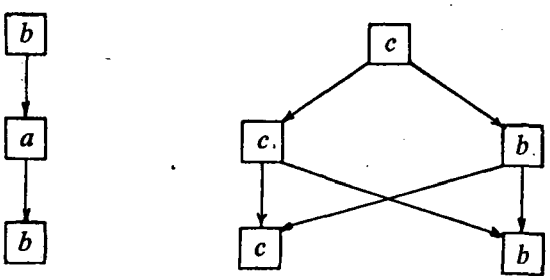
$$h_2(a, 1) = \varepsilon, \quad h_2((b, 1)) = (b, 1), h_2((b, 2)) = (b, 2), h_2((c, 1)) = (c, 1),$$

$$h_2((c, 2)) = (c, 2).$$

Let $t_1 = [(b, 1)(a, 1)(b, 1)]_{D_1} \in U_1$,

$$t_2 = [(c, 1)(b, 1)(c, 2)(b, 2)(c, 1)]_{D_2} \in U_2.$$

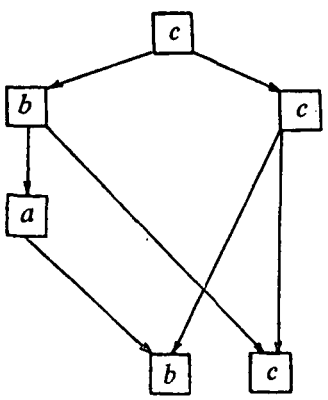
Reduced versions of dep-graphs of t_1 and t_2, resp., are of the form (i.e. transitive arcs are omitted):



By the definition of the concurrent composition, $t_1 \| t_2$ is a labelled trace over $C(D_1, D_2)$

$$t_1 \| t_2 = [(c, 1)(b, 1)(a, 1)(c, 2)(b, 2)(c, 1)]_{C(D_1, D_2)}$$

A reduced version of a dep-graph of $t_1 \| t_2$ is of the form:



It can be seen that

$$U_1 \| U_2 = [\text{pref}((c, 1)(b, 1)(a, 1) + (c, 2)(b, 2)(a, 1))^*]_{C(D_1, D_2)}.$$

We propose in the example that U_1 is the activity of the single portion-buffer, and U_2 is the activity of the two-portion buffer with a and b corresponding to "out" and "in" repectively, in the former, b and c corresponding to "out" and "in" in the latter. Then $U_1 \| U_2$ corresponds to a composition of the two buffers: the two-portion buffer inputs from its producer, then outputs to the single-portion buffer, and in turn, the single-portion buffer outputs to its consumer.

(ii) Let $D = \mathrm{syre}(\{((a, 1), (e, 1)), ((b, 1), (e, 1)), ((e, 1), (c, 1))\})$,

$$U = \{[(a, 1)(b, 1)(e, 1)(a, 1)(c, 1)]_D\}.$$

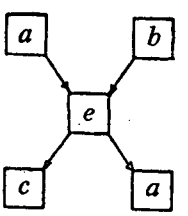
Then

$$S(D, D) = D \cup \{((x, 2), (y, 2)) \mid ((x,1), (y, 1)) \in D\} \cup$$
$$\cup \mathrm{syre}(\{((x,1), (y, 2)) \mid x, y \in l(\mathrm{dom}(D))\}).$$

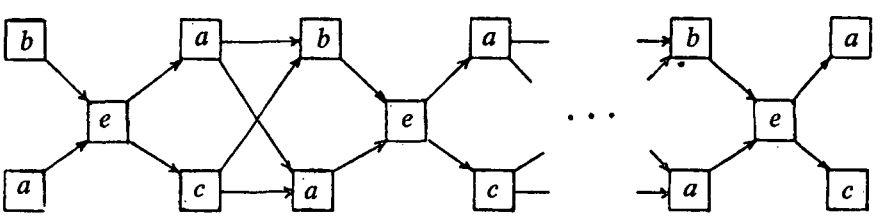
By the definition of the iteration

$$U^{\otimes} = [((a, 1)(b, 1)(e, 1)(a, 1)(c, 1)(a, 2)(b, 2)(e, 2)(a, 2)(c, 2))^*$$
$$(\varepsilon + (a, 1)(b, 1)(e, 1)(a, 1)(c, 1))]_{S(D, D)}.$$

A reduced version of a dep-graph of $t \in U$ has the form:



and reduced versions of dep-graphs of elements of $U^{\otimes}$ are of the form:



In the sequel, for $u \in X^*$, $Y \subseteq X$, by $u|_Y$ we denote the projection of u on Y, i.e. the image of u by the erasing homomorphism from X^* to Y^*. For $u, v \in X^*$ we also write $u|_v$ instead of $u|_{\mathrm{alph}(v)}$ without fear of confusion, where alph (v) denotes the set of symbols forming v.

Proposition 3. $t_1 \| t_2$ contains not more than one element.

Proof. By trivial induction on the length of elements of $T(C(D_1, D_2))$ we can show that if for $t, t' \in T(C(D_1, D_2))$ $h_1(t) = h_1(t')$ and $h_2(t) = h_2(t')$, then $t = t'$.

Proposition 4.

$$t_1 \| t_2 \neq \emptyset \quad \text{if and only if}$$

$$l(t_1)|_{l(\mathrm{dom}(D_2))} \cap l(t_2)|_{l(\mathrm{dom}(D_1))} \neq \emptyset.$$

Proof. The "only if" part is obvious and we prove the "if" part. Suppose that there exists $w \in l(t_1)|_{l(\mathrm{dom}(D_2))} \cap l(t_2)|_{l(\mathrm{dom}(D_1))} \subseteq X^*$. Then there exist $u_1 \in t_1$, $u_2 \in t_2 : l(u_1)|_{l(\mathrm{dom}(D_2))} = l(u_2)|_{l(\mathrm{dom}(D_1))} = u$. Let

$$u_1 = a_1 a_2 \ldots a_n \in (\mathrm{dom}(D_1))^*,$$

$$u_2 = b_1 b_2 \ldots b_m \in (\mathrm{dom}(D_2))^*,$$

$$w = c_1 c_2 \ldots c_k \in Y^* = (l(\mathrm{dom}(D_1)) \cap l(\mathrm{dom}(D_2)))^*.$$

Then, there exist monotonic functions $f_1 : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, n\}$ and $f_2 : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, m\}$ satisfying:

a_j is a case of an element in Y if and only if $j = f_1(i)$ for some $i \leqq k$, $j \leqq n$, and b_j is a case of an element in Y if and only if $j = f_2(i)$ for some $i \leqq k$, $j \leqq m$.

Let $g : \{c_1, c_2, \ldots, c_k\} \to \mathrm{dom}(C(D_1, D_2))$ be defined as follows.

Let $a_{f_1(i)} = (c_i, p)$, $b_{f_2(i)} = (c_i, q)$. Then $g(c_i) = (c_i, s)$, where s is determined from the equation system:

$$p = \mathrm{rem}(s-1, \#(c_i, D_1)) + 1.$$

$$q = \mathrm{en}(s-1, \#(c_i, D_1)) + 1.$$

Let $u_1' = a_1' a_2' \ldots a_n'$, $u_2' = b_1' b_2' \ldots b_m'$ be defined by

$$a_j' = \begin{cases} a_j & \text{if } j \notin f_1(\{1, 2, \ldots, k\}), \\ g(c_j) & \text{if } j = f_1(i), \text{ for } j \leqq n, \end{cases}$$

$$b_j = \begin{cases} b_j & \text{if } j \notin f_2(\{1, 2, \ldots, k\}), \\ g(c_i) & \text{if } j = f_2(i) \text{ for } j \leqq m. \end{cases}$$

Clearly, $u_1'|_{u_2'} = u_2'|_{u_1'} = g(c_1 c_2 \ldots c_k)$.

Hence, by Theorem 2 ([12], pp. 205) there exists $w' \in (\mathrm{dom}(C(D_1, D_2))^*$ such that $w'|_{u_1'} = u_1'$, $w'|_{u_2'} = u_2'$. It is obvious from the definition of g that

$$[w']_{C(D_1, D_2)} = t_1 \| t_2.$$

Proposition 5. Let D_3, D_1, D_2 be labelled dependencies on X, U, U_1, $U_2 \in P(D_1)$, V, V_1, $V_2 \in P(D_2)$, $Z \in P(D_3)$, $W \in P(C(D_1, D_2))$, $t \in T(C(D_1, D_2))$, and h_1, h_2 the homomorphisms associated with $C(D_1, D_2)$. Then

a) $C(D_1, D_2) \cong C(D_2, D_1)$ and $U \| V \cong V \| U$;
b) $C(D_1, C(D_2, D_3)) \cong C(C(D_1, D_2), D_3)$ and $(U \| V) \| Z \cong U \| (V \| Z)$;
c) $U \| \emptyset = \emptyset$;
d) $[\varepsilon]_{D_1} \| [\varepsilon]_{D_2} = [\varepsilon]_{C(D_1, D_2)}$;
e) $(U_1 \cup U_2) \| V = (U_1 \| V) \cup (U_2 \| V)$;
f) $U \| (V_1 \cup V_2) = (U \| V_1) \cup (U \| V_2)$;
g) $(h_1(t)U) \| (h_2(t)V) = t(U_1 \| U_2)$;
h) $W \subseteq h_1(W) \| h_2(W)$.

Proof.

a) For $a\in X$, $0<j\leqq \#(a, D_1)\cdot\#(a, D_2)$ the numbers $i=$ $=\mathrm{rem}\,(j-1,\ \#(a, D_1))+1$, $k=\mathrm{en}\,(j-1,\ \#(a, D_1))+1$ are defined uniquely, and for a pair (i, k) with $i\leqq \#(a, D_1)$ and $k\leqq \#(a, D_2)$ the integer $j'=\#(a, D_2)\times$ $\times(i-1)+k$ is determined uniquely. Thus the correspondence $f_a(j)=j'$ is an one-to-one mapping from $\{1, \ldots, \#(a, D_1)\cdot(a, D_2)\}$ to $\{1, \ldots, \#(a, D_1)\cdot\#(a, D_2)\}$.

By the definitions of $C(D_1, D_2)$, $C(D_2, D_1)$ and h_1, h_2, the mapping h: dom $(C(D_1, D_2))\to$dom $C(D_2, D_1))$ defined by

$$h((a, j)) = \begin{cases}(a, j) & \text{if} \quad \#(a, D_1)\cdot\#(a, D_2) = 0,\\ (a, f_a(j)) & \text{if} \quad \#(a, D_1)\cdot\#(a, D_2) > 0\end{cases}$$

is an one-to-one isomorphism. Furthermore, let h_1' and h_2' be the projections associated with $C(D_2, D_1)$, we have:

$$h_1((a, j)) = h_2'(h(a, j)),\ h_2((a, j)) = h_1'(h(a, j)).$$

Consequently, it follows a).

b) For $a\in X$ with $\#(a, C(D_1, C(D_2, D_3)))>0$ and for $i=1, 2, 3$ let the mappings g_{ia} and g'_{ia} from $\{1, 2, \ldots, \#(a, C(D_1, C(D_2, D_3)))\}$ to $\{1, 2, \ldots, \#(a,D_i)\}$ be defined as follows (g_{ia} and g'_{ia} are undefined if $\#(a, D_i)=0$).

If $\#(a, D)\cdot\#(a, D_2)\cdot\#(a, D_3)>0$, for $j\leqq \#(a, D_1)\cdot\#(a, D_2)\cdot\#(a, D_3)$, let

$$g_{1a}(j) = \mathrm{rem}\,(j-1,\ \#(a, D_1))+1,$$

$$g_{2a}(j) = \mathrm{rem}\,(\mathrm{en}\,(j-1,\ \#(a, D_1)),\ \#(a, D_2))+1,$$

$$g_{3a}(j) = \mathrm{en}(\mathrm{en}\,(j-1,\ \#(a, D_1)),\ \#(a, D_2))+1,$$

$$g'_{1a}(j) = \mathrm{rem}\,(\mathrm{rem}\,(j-1,\ \#(a, D_1)\cdot\#(a, D_2)),\ \#(a, D_1))+1,$$

$$g'_{2a}(j) = \mathrm{en}\,(\mathrm{rem}\,(j-1,\ \#(a, D_1)\cdot\#(a, D_2)),\ \#(a, D_1))+1$$

$$g'_{3a}(j) = \mathrm{en}\,(j'-1,\ \#(a, D_1)\cdot\#(a, D_2))+1.$$

If $\#(a, D_1)\cdot\#(a, D_2)\cdot\#(a, D_3)=0$, for $j\leqq \#(a, C(D_1, C(D_2, D_3)))$, then let

$$g_{1a}(j) = g'_{1a}(j) = \begin{cases}\text{undefined} & \text{if} \quad \#(a, D_1) = 0,\\ \mathrm{rem}\,(j-1,\ \#(a, D_1))+1 & \text{if} \quad \#(a, D_1) > 0,\end{cases}$$

$$g_{2a}(j) = g'_{2a}(j) = \begin{cases}\text{undefined} & \text{if} \quad \#(a, D_2) = 0,\\ \mathrm{en}\,(j-1,\ \#(a, D_1))+1 & \text{if} \quad \#(a, D_1)\cdot\#(a, D_2) > 0,\\ \mathrm{rem}\,(j-1,\ \#(a, D_2))+1 & \text{if} \quad \#(a, D_1) = 0\,\&\,\#(a, D_2) > 0,\end{cases}$$

$$g_{3a}(j) = g'_{3a}(j) = \begin{cases}\text{undefined} & \text{if} \quad \#(a, D_3) = 0,\\ \mathrm{en}\,(j-1,\ \#(a, D_1)+\#(a, D_2)) & \text{if} \quad \#(a, D_3) > 0.\end{cases}$$

Let $f_a(j)=j'$ iff for $i=1, 2, 3$ $g_{ia}(j)=g'_{ia}(j')$.

It can be seen easily that $((a, j), (b, j'))\in C(D_1, C(D_2, D_3))$ ($C(C(D_1, D_2), D_3)$, resp.) if and only if there exists i in $\{1, 2, 3\}$ such that $((a, g_{ia}(j)), (b, g_{ib}(j'))\in$

$\in D_i((a, g'_{ia}(j)), (b, g'_{ib}(j')))\in D_i$, resp.). Hence, the mapping f: dom $(C(D_1, C(D_2, D_3)))\to$ dom $(C(C(D_1, D_2), D_3))$ defined by

$$f((a, j)) = (a, f_a(j))$$

is an isomorphism between $C(D_1, C(D_2, D_3))$ and $C(C(D_1, D_2), D_3)$.

For $i=1, 2, 3$ mappings G_i, G'_i from $T(C(D_1, C(D_2, D_3)))$, $T(C(C(D_1, D_2), D_3))$ resp., to $T(D_i)$ defined by

$$G_i((a, j)) = (a, g_{ia}(j)),$$

$$G'_i((a, j)) = (a, g'_{ia}(j))$$

are homomorphisms by Lemma 1. Furthermore, for $t\in T(C(D_1, C(D_2, D_3)))$ $(T(C(C(D_1, D_2), D_3)),$ resp.) $t\in U\|(V\|Z)$ $((U\|V)\|Z,$ resp.) if and only if $G_1(t)\in U$, $G_2(t)\in V$, $G_3(t)\in Z$ $(G'_1(t)\in U, G'_2(t)\in V, G'_3(t)\in Z,$ resp.). Hence, by Lemma 1 f can be extended to an isomorphism from $T(C(D_1, C(D_2, D_3)))$ to $T(C(C(D_1, D_2), D_3))$ and $f(U\|(V\|Z))=(U\|V)\|Z$. Thus, by Proposition 2, $U\|(V\|Z)\cong(U\|V)\|Z$.

The properties (c)—(h) are obvious.

The following theorem has been formulated by Mazurkiewicz for the case of trace languages. Fortunately, it is still true for labelled trace languages, although our operation of synchronization is more powerful and general than his one.

Theorem 1. The concurrent composition $\|$ is the least function from $P(D_1)\times$ $\times P(D_2)$ to $P(C(D_1, D_2))$ (w.r.t. the inclusion ordering of its values) meeting the following conditions:

(a) $$(h_1(x)U)\|(h_2(x))V) = x(U\|V),$$

(b) $$(U_1\cup U_2)\|V = (U_1\|V)\cup(U_2\|V),$$

(c) $$U\|(V_1\cup V_2) = (U\|V_1)\cup(U\|V_2),$$

(d) $$[\varepsilon]_{D_1}\|[\varepsilon]_{D_2} = [\varepsilon]_{C(D_1, D_2)},$$

for each actioncase x in $C(D_1, D_2)$, U, U_1, $U_2\in P(D_1)$, $V, V_1, V_2\in P(D_2)$.

The proof of the theorem is similar to the proof of Theorem 1, [14], pp. 352 and is omitted here.

b) *Union and intersection.*

We deal with the construction of activities from activities over different labelled dependencies. The union is intended for the nondeterministic choice and the intersection is intended to represent the tied synchronization.

Let $N(D_1, D_2)$ be the labelled dependency defined by

$$N(D_1, D_2) = D_1\cup\{((a, i+\#(a, D_1)), (b, j+\#(b, D_1)))\,|\,((a, i), (b, j))\in D_2\}\cup$$

$$\cup \text{syre}(\{(a, j), (a, i)+\#(a, D_1)\,|\,(a, j)\in\text{dom}(D_1), (a, i)\in\text{dom}(D_2)\}).$$

A mapping s: $T(D_2)\to T(N(D_1, D_2))$ associated to $N(D_1, D_2)$ is defined by

$$s((a, i)) = (a, i+\#(a, D_1)), \quad (a, i)\in\text{dom}(D_2).$$

Let $I(D_1, D_2)$ be the labelled dependency defined by $I(D_1, D_2) = C(D_1, D_2) \cap \cap (Y \times \{1, 2, \ldots\})^2$, where $Y = l(D_1) \cap l(D_2)$. Since $C(D_1, D_2) \cap (\mathrm{dom}\,(I(D_1, D_2)))^2 = I(D_1, D_2)$ each labelled trace over $I(D_1, D_2)$ is a labelled trace over $C(D_1, D_2)$. Let h_1, h_2 be homomorphisms associated with $C(D_1, D_2)$.

Definition 5. (i) A nondeterminic composition $U_1 \square U_2$ of U_1 and U_2 is a labelled trace language over $N(D_1, D_2)$ defined by

$U_1 \square U_2 = U_1 \cup s(U_2)$, where the operation $\cup$ on the right hand side is for labelled trace languages over $N(D_1, D_2)$ with considering U_1 as a labelled trace language over $N(D_1, D_2)$.

(ii) The intersection $U_1 \sqcap U_2$ of U_1 and U_2 is a labelled trace language over $I(D_1, D_2)$ defined by

$$U_1 \sqcap U_2 = \{t \in T(I(D_1, D_2)) \mid h_1(t) \in U_1, h_2(t) \in U_2\}.$$

Proposition 6. For $U', U'' \in P(D_1)$, $V' \in P(D_2)$,

a) $U' \square U'' \cong U' \cup U''$; $N(D_1, D_1) \cong D_1$,

b) $U' \| V' = U' \sqcap V'$ if $D_1 = D_2$.

This follows immediately from Definition 3.

Proposition 7. Let D_1, D_2, D_3, D be labelled dependenceis on X, $U \in P(D)$, $V \in P(D_1)$, $W \in P(D_2)$, $Z \in P(D_3)$, $t_1 \in T(D_1)$, $t_2 \in T(D_2)$. Then

a) $[\varepsilon]_{D_1} \circ U \cong U \circ [\varepsilon]_{D_1} \cong U$,

b) $(U \circ V) \circ W = U \circ (V \circ W)$,

c) $U \circ (V \square W) \cong (U \circ V) \square (U \circ W)$, $(V \square W) \circ U \cong (V \circ U) \square (W \circ U)$,

d) $t_1 \circ U \circ t_2 \cong t_1 \circ V \circ t_2 \Rightarrow U \cong V$.

Proof. a), b) and d) are obvious. To prove c) consider a mapping h from $\mathrm{dom}\,(N(S(D, D_1), S(D, D_2)))$ onto $\mathrm{dom}\,(S(D, N(D_1, D_2)))$ defined by:

$$h((a, j)) = \begin{cases} (a, j), & \text{if } j \leqq \#(a, D) + \#(a, D_1), \\ (a, j - \#(a, D)), & \text{otherwise.} \end{cases}$$

By the definition of the operations S, N on labelled dependencies, $((a, i), (b, j)) \in N(S(D, D_1), S(D, D_2))$ iff $(h((a, i)), h((b, j)) \in S(D, N(D_1, D_2))$. Hence, h can be extended to a homomorphism from $T(N(S(D, D_1), S(D, D_2)))$ to $T(S(D, N(D_1, D_2)))$ in the obvious way (by Lemma 1). Furthermore, it can be seen easily that $h((U \circ V) \square (U \circ W)) = U \circ (V \square W)$. By Proposition 2, $U \circ (V \square W) \cong (U \circ V) \square (U \circ W)$. The remaining case of c) is proved similarly.

4. Relations to other models

As mentioned in the section 2 each labelled trace induces a lebelled partial ordering over X, and each labelled partial ordering over X is a finite labelled event structure over X ([16]). Thus, a labelled trace language over a labelled dependency on X is a set of labelled event structures having a very simple representative: a (finite) labelled dependency and a word language (may be represented by a regular expression). In our paper [8] we have related labelled trace languages to Petri nets and some

interesting results have been obtained. In this section, we relate labelled trace languages to semilanguages introduced by Starke [18], [19].

Definition 6 ([19] pp. 337).

(i) A *labelled partial ordering* (lpo for short) over X is a triple (A, S, β), where (A, S) is a irreflexive partial ordering, $\beta: A \to X$ is a labelling mapping.

(ii) Two lpo's (A, S, β) and (A', S', β') are said to be isomorphic iff there exists a bijection b from A onto A' which preserves the labelling and the ordering:

$$aSc \Leftrightarrow b(a)S'b(c) \& \beta(a) = \beta'(b(a)).$$

The isomorphy class $[(A, S, \beta)]$ of a finite lpo (A, S, β), i.e. the class of all lpo's which are isomorphic with (A, S, β) is called a *partial word* over X. A partial word $[(A, S, \beta)]$ over X such that for all a, b from A

$$\beta(b) = \beta(a) \Rightarrow aSb \vee bSa \vee a = b, \tag{1}$$

i.e. where all the sets $\beta^{-1}(x)$ (for $x \in X$) are chains w.r.t. S is called a *semiword* over X.

Let $p(t)$ denote a partial word over X induced by a labelled trace t over a labelled dependency on X (see section 2) i.e. $p(t) = [(A, S, \beta)]$ where (A, S, β) is the labelled partial ordering induced by t.

Theorem 2. For each labelled trace t over a labelled dependency on X, $p(t)$ is a semiword over X.

Proof. Let $D(t)$ be a dep-graph of t, where t is a labelled trace over a labelled dependency D on X. By the definition of D, if x, y are cases of an action $a \in X$, $(x, y) \in D$. Thus, the labelled partial ordering over X induced by t satisfies (1). Consequently, $p(t)$ is a semiword over X.

It follows from Theorem 2 that every labelled trace language over a labelled dependency on X generates a semilanguage over X in the natural way.

For $U \in P(D)$, denote by

$$SL(U) = \{p(t) | t \in U\}.$$

$SL(U)$ is called *semilanguage generated* by U.

Theorem 3. Let $U \in P(D_1)$, $V \in P(D_2)$, where D_1, D_2 are labelled dependencies on X. Then

$$SL(U^{\otimes} \circ V) = SL(U \circ U^{\otimes} \circ Y \square Y).$$

Proof.

By (iii) of Definition 4

$$U^{\otimes} = ((U - \{[\varepsilon]_{D_1}\}) \circ (U - \{[\varepsilon]_{D_1}\}))^* (U \cup \{[\varepsilon]_{S(D_1, D_1)}\}) \in P(S(D_1, D_1)),$$

where $U \cup \{[\varepsilon]_{S(D_1, D_1)}\}$ is considered as a labelled trace language over $S(D_1, D_1)$. It follows from the definition of $S(D_1, D_1)$ and of the sequential concatenation that for $t \in T(S(D_1, D_1))$, $p(t) \in U^{\otimes}$ if and only if there exist $t_1, t_2, \ldots, t_n \in U - \{[\varepsilon]_{D_1}\}$ such that

(i) $D(t) = \emptyset$ iff $n = 0$, and

(ii) Let $D(t_1), \ldots, D(t_n)$ be dep-graphs of $t_1, \ldots, t_n$ over D_1, $D(t_i) = (V_i, E_i, X_i, \beta_i)$, $i = 1, 2, \ldots, n$, $V_i \cap V_j = \emptyset$ for $i \neq j$, $i, j \leqq n$.

Then $D(t) \cong (\bigcup_{i=1}^{n} V_i, E, X, \beta)$, where

$$E = (\bigcup_{i=1}^{n} E_i) \cup \{(a, b) \mid a \in V_i, b \in V_{i+1}, i \leqq n-1\}, \beta|_{V_i} = \beta_i.$$

Hence, since $V_i \neq \emptyset$ for $i \leqq n$, for $i < j \leqq n$, $a \in V_i$, $b \in V_j$, (a, b) is an arc of the transitive closure of $D(t)$. From the definition of the sequential concatenation of labelled trace languages it follows that

$$SL(U \circ U^{\circledast}) \cup \{p([\varepsilon]_{D_1})\} = SL(U^{\circledast}),$$

$$SL(U^{\circledast} \circ V) = SL(U \circ U^{\circledast} \circ V) \cup SL(V).$$

By the definition of the operation $\square$ our theorem is obtained from the last equality.

To relate labelled trace languages to interleavings of strings we recall the synchronization mechanism introduced by Hoare [7], E. Knuth [12].

Definition 7 ([7]). Let $L_1, L_2 \subseteq X^*$, $Y \subseteq X$. The synchronized parallel composition $L_1 \|_Y L_2$ is the set $\bigcup_{\substack{w_1 \in L_1 \\ w_2 \in L_2}} w_1 \|_Y w_2$, where $w_1 \|_Y w_2$ denotes the set of all successful interleavings of w_1 and w_2 with synchronising communications in Y and is defined inductively as follows:

(i) $\varepsilon \|_Y \varepsilon = \{\varepsilon\}$

(ii) $aw \|_Y \varepsilon = \varepsilon \|_Y aw = \begin{cases} \emptyset & \text{if } a \in Y \\ a(w \|_Y \varepsilon) & \text{if } a \notin Y, \end{cases}$

(iii) $aw \|_Y bw' = bw' \|_Y aw = \begin{cases} a(w \|_Y w') & \text{if } a = b \in Y \\ \emptyset & \text{if } a \neq b \wedge a, b \in Y \\ a(w \|_Y bw') & \text{if } a \notin Y \wedge b \in Y \\ a(w \|_Y bw') \cup b(aw \|_Y w') & \text{if } a \notin Y, b \notin Y. \end{cases}$

Theorem 4. For a labelled trace language $U \in P(D)$ let $\text{inter}(U) = \bigcup_{t \in U} l(t)$. Then, for $U \in P(D_1)$, $V \in P(D_2)$, (where D_1, D_2 are labelled dependencies) and $Y = l(\text{dom } D_1) \cap l(\text{dom } (D_2))$,

a) $\text{inter } (U \circ V) = \text{inter } (U) \text{ inter } (V)$,

b) $\text{inter } (U \| V) = \text{inter } (U) \|_Y \text{ inter } (V)$,

c) $\text{inter } (U^{\circledast}) = (\text{inter } (U))^*$,

d) $\text{inter } (U \square V) = \text{inter } (U) \cup \text{inter } (V)$,

e) $\text{inter } (U \sqcap V) = \text{inter } (U) \cap \text{inter } (V)$.

Proof. a) and c), d) are obvious. e) follows from b).

b) is proved as follows.

Let h_1 and h_2 be the projections associated with $C(D_1, D_2)$. Clearly, for $t \in U \| V$ $h_1(t) \in U$, $h_2(t) \in V$, and $l(t)|_{l(\text{dom } (D_2))} = l(h_2(t)) \subseteq \text{inter } (V)$, $l(t)|_{l(\text{dom } (D_1))} = l(h_1(t)) \subseteq \subseteq \text{inter } (U)$. Thus, $\text{inter } (U \| V) \subseteq \text{inter } (U) \|_Y \text{ inter } (V)$.

Let $y \in \text{inter}(U) \|_Y \text{inter}(V)$. By Definition 7, $y|_{l(\text{dom}(D_1))} \in \text{inter}(U)$, $y|_{l(\text{dom}(D_2))} \in \text{inter}(V)$. There exist $u \in U$, $v \in V$ such that $y|_{l(\text{dom}(D_1))} \in l(u)$, $y|_{l(\text{dom}(D_2))} \in l(v)$. By Proposition 4, $t = u \| v$ is defined.

It follows easily from the definition of h_1 and h_2 that $y \in l(t)$. This completes the proof of the theorem.

In the sequel, for simplicity of denotation, if L_1 and L_2 are considered over fixed alphabets, say Σ_1 and Σ_2, and $Y = \Sigma_1 \cap \Sigma_2$, we shall writte $L_1 \| L_2$ instead of $L_1 \|_Y L_2$.

Proposition 8. Let L_1, L_2, L_3 are languages over $\Sigma_1, \Sigma_2. \Sigma_3$ respectively. Then

$$L_1 \| L_2 = L_2 \| L_1, \quad \text{and} \quad (L_1 \| L_2) \| L_3 = L_1 \| (L_2 \| L_3).$$

Proof. Straightforward from the definition of the operation $\|$.

5. Labelled trace languages as a noninterleaving semantics for CSP

The notion of CSP presented in this paper is at an abstract level necessary for our purpose.

Let *Comm* be a finite set of actions. A process P over *Comm* is in one of the following forms:

$$P := P_1; P_2; \ldots; P_n,$$

$$P := [P_1 \| P_2 \| \ldots \| P_n],$$

$$P := \underline{\otimes} P_1,$$

$$P := [P_1 \square P_2 \square \ldots \square P_n],$$

$$P := a \to P_1, a \in Comm,$$

$$P := \textbf{skip}, (\textbf{skip} \notin Comm),$$

$$P := P_1 \setminus \{b_1, b_2, \ldots, b_n\},$$

where $P_1, P_2, \ldots, P_n$ are processes over *Comm*.

The meaning of the above constructions of processes is given informally as follows.

$P_1; P_2; \ldots; P_n$ specifies sequential excution of $P_1, P_2, \ldots, P_n$ in the order written (process by process, P_{i+1} starts only P_i has terminated, $1 \leqq i \leqq n-1$), and starts with the start of P_1, terminates with the termination of P_k.

$[P_1 \| P_2 \| \ldots \| P_n]$ specifies concurrent excution of its constituent processes. They all start simultaneously and the process $P = [P_1 \| \ldots \| P_n]$ terminates successfully only if and when they have all successfully terminated. The relative speed with which they are excuted is arbitrary. The set of actions excuted by each of them is required to be disjoint from those executed by the rest. $P_1, P_2, \ldots, P_n$ are synchronized by the actions intended. P_i excutes an action intended to synchronize with P_j (in the construction) if and only if P_j excutes a corresponding action (intended to synchronize P_j with P_i) simultaneously (see [6], [7], [11]).

$\underline{\otimes} P$ specifies as many iterations as necessary of P sequentially.

$[P_1 \square P_2 \square \ldots \square P_n]$ specifies excution of exactly one of its consituent processes and the choice between them is fully nondeterministic, cannot be influenced by the environment.

$a \rightarrow P_1$ specifies excution of the action a followed by excution of P_1.

Skip specifies the process having no effect and never fails.

Now, we identify the action intended to synchronize P_i with P_j with a corresponding action intended to synchronize P_j with P_i in a construct $[P_1 \| P_2 \| \ldots \| P_n]$. We can suppose that the set of actions excuted by P_i may not be disjoint from the one by P_j and the actions in their intersection require that P_i and P_j must excute each of them simultaneously. (This abstraction has been made by Hoare in [7], [2], Janicki in [11]).

The interleaving semantics for CSP given by Hoare [2], [7] is a follows.

Each process over *Comm* is identified with a subset of *Comm** called its interleaving semantics:

$$\mathbf{skip} := \{\varepsilon\},$$

$$a \rightarrow P := aP,$$

$$P_1;\ P_2;\ \ldots;\ P_n := P_1 P_2 \ldots P_n,$$

$$[P_1 \square P_2 \square \ldots \square P_n] := P_1 \cup P_2 \cup \ldots \cup P_n,$$

$$[P_1 \| P_2 \| \ldots \| P_n] := P_1 \| P_2 \| \ldots \| P_n, \quad \text{where}$$

the operation $\|$ on languages is defined in the previous section and $P_1, \ldots, P_n$ are considered as languages over alph $(P_1), \ldots,$ alph (P_n) respectively. (Here for a language L, alph (L) denotes the smallest alphabet, over which L is a language).

$$\underline{\otimes} P := P^*,$$

$$P \setminus \{b_1, \ldots, b_n\} := P|_{\text{alph}(P_1) \setminus \{b_1, \ldots, b_n\}},$$

where $P|_A$ denotes the projection of P on A^*.

Because of the presence of the hidding operation in CSP, to relate our model to CSP we have to extend the notion of labelled trace languages.

An ε-labelled dependency on X is a symmetric relation $D_\varepsilon \subseteq ((X \cup \{\varepsilon\}) \times \{1, 2, \ldots\})^2$ satisfying:

(i) $(a, i) \in \text{dom}(D_\varepsilon) \Rightarrow (a, j) \in \text{dom}(D_\varepsilon)$ for $j \leqq i$,

(ii) $((a, i), (a, j)) \in D_\varepsilon$ for $(a, i), (a, j) \in \text{dom}(D_\varepsilon)$ and $a \neq \varepsilon$.

An ε-labelled dependency on X may not be reflexive in its domain. However, this has no effect in the definition of trace languages and the notion of trace languages is extended to this case. Then, a trace language over D_ε is called an ε-labelled trace language over D_ε. All the notions and the results presented in the previous sections are valid for ε-labelled trace languages as well with the only exception that the set Y in the definition of the operation $\|$ of labelled trace languages is modified as

$$Y = (l(\text{dom}(D_1)) \cap l(\text{dom}(D_2))) \setminus \{\varepsilon\}.$$

ε-labelled trace semantics proposed for CSP is presented bellow. Each process over *Comm* is identified with an ε-labelled trace language over an ε-labelled dependency on *Comm* as follows:

$$a \to P := \{[(a, 1)]_{\{((a,1),(a,1))\}} \circ P,$$

$$P_1; P_2; \ldots; P_n := P_1 \circ P_2 \circ \ldots \circ P_n;$$

$$[P_1 \| P_2 \| \ldots \| P_n] := P_1 \| P_2 \| \ldots \| P_n;$$

$$[P_1 \square \ldots \square P_n] := P_1 \square P_2 \square \ldots \square P_n;$$

$$\underline{\otimes} P := P^{\otimes}$$

$$P \setminus \{b_1. \ldots, b_n\} := h_{\{b_1, \ldots, b_n\}}(P),$$

where $h_{\{b_1, \ldots, b_n\}}$ is defined as follows: For an ε-labelled dependency D_ε on X let

$$h_{\{b_1, \ldots, b_n\}}((a, i)) = \begin{cases} (a, i) & \text{if} \quad a \notin \{b_1, \ldots, b_n\} \wedge (a, i) \in \mathrm{dom}\,(D_\varepsilon) \\ (\varepsilon, i) & \text{if} \quad a \in \{b_1, \ldots, b_n\} \wedge (a, i)\, \mathrm{dom}\,(D_\varepsilon), \end{cases}$$

$$D_\varepsilon\{b_1, \ldots, b_n\} = \{(h_{\{b_1, \ldots, b_n\}}(x), h_{\{b_1, \ldots, b_n\}}(y)) | (x, y) \in D_\varepsilon\}.$$

By Lemma 1, $h_{\{b_1, \ldots, b_n\}}$ is considered as a homomorphism from $T(D_\varepsilon)$ to $T(D_\varepsilon | \{b_1, \ldots, b_n\})$.

The correspondence between ε-labelled trace semantics and interleaving semantics for CSP is stated by the following theorem, which follows immediately from Theorem 4.

Theorem 5. For a process P over *Comm*. Let $LT(P)$, $INTER\ (P)$ denote the ε-labelled trace semantics, interleaving semantics, respectively, for P. Then

$$\mathrm{inter}\,(LT(P)) = INTER\,(P).$$

Proposition 9. If a process P over *Comm* does not contain a construction $[P_1 \square \ldots \square P_n]$, $LT(P)$ contains, at most, one element.

The Proposition follows from Proposition 3.

6. Conclusion

We have presented an extention of the theory of traces as an attempt to provide a mathematical description for the behaviour of concurrent systems, more specifically, CSP. Labelled trace languages have been shown to be more powerful than trace languages and to have a simple representation.

However, the construction of the theory of CSP based upon labelled trace languages requires a deeper study on labelled trace languages concluding a construction of domains of the operations on processes so that the operations are continuous and the representation of the properties of processes in its semantics in the model. This will be presented in our future work.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACADEMY OF SCIENCES
VICTOR HUGO U. 18—22
1123 BUDAPEST
HUNGARY

References

[1] I. J. AALBERSBERG and G. ROZENBERG, Theory of traces, Institute of Applied Mathematics and Computer Science, Univ. of Leiden, The Netherlands, Rep. 85—16, August 1985.

[2] S. D. BROOKES, C. A. R. HOARE, A. W. ROSCOE, A theory of communicating sequential processes, J. of ACM, Vol. 31, N. 3, July 1984, pp. 560—599.

[3] P. DEGANO and U. MONTANARI, Distributed Systems, partial orderings of events, and event structures, Proc. of the International Summer School "Control Flow and Data Flow: Concepts of Distributed Programming", NATO ASI Series, M. Broy, Ed., Vol. 38F14, Springer-Verlag, 1985, pp. 7—106.

[4] U. GOLTZ and W. REISIG, CSP-programs as nets with individual tokens, LNCS, Springer-Verlag, Vol. 188, 1985, pp. 169—196.

[5] M. HENNESSY, W. LI, G. PLOTKIN, "A first attempt at translating CSP into CCS", Proc. of the 2nd International Conference on Distributed Computing, *IEEE*, N. 81, CH 1591—7, Paris 1981.

[6] C. A. R. HOARE, Communicationg sequential processes, *Comm. of ACM*, Vol. 21, N. 8, 1978, pp. 666—677.

[7] C. A. R. HOARE, Specification-oriented semantics for communicating processes, Acta Informatica, Vol. 23 Springer-Verlag, 1986, pp. 9—66.

[8] D. V. HUNG and E. KNUTH, Labelled trace languages and Petri nets, Working Paper, MTA-SZTAKI.

[9] D. V. HUNG, Notes on trace languages, Projections and synthesized computation systems, Közlemények, MTA-SZTAKI, Vol. 32, 1985, pp. 87—104.

[10] D. V. HUNG and M. SZIJJÁRTÓ, Synchronized parallel composition of languages, Proc. of Conference of Automata, Languages and Programming Systems, Salgótarján, Hung. May 1986.

[11] R. JANICKI, Trace Semantics for Communicating Sequential Processes, Institute for Elektroniske Systemer, Danmark, R—85—12, 1985.

[12] E. KNUTH, GY. GYŐRY and L. RÓNYAI, *A Study of the projection operation*, Application and Theory of Petri Nets, W. Reisig, Ed. Springer-Verlag, Vol. 52, 1982.

[13] A. MAZURKIEWICZ, Concurrent Program Schemes and Their Interpretations, DAIMI PB—78, Aarhus Univ., Press, 1977.

[14] A. MAZURKIEWICZ, *Semantics of concurrent systems: a modular fixed-point trace approach*, LNCS, Vol. 188, 1985, pp. 353—375.

[15] R. MILNER, Calculi for synchrony and asynchrony, *TCS*, Vol. 25, 1983, pp. 267—310.

[16] M. NIELSEN, G. PLOTKIN, and F. WINSKEL, Petri nets, event structures and domains, Part I, *TCS*, Vol. 13, 1981, pp. 85—108.

[17] C. A. PETRI, Non-sequential processes, GMD—ISF Report, ISF—77—05, 1977, pp. 1—24.

[18] P. H. STARKE, Processes in Petri nets, *Informationverarb u. Kybernet.* EIK, Vol. 17, 1981, pp. 389—416.

[19] P. H. STARKE, *Traces and semiwords*, LNCS, Computation Theory, Andrezej, Ed., Fifth. Symposium, Vol. 208, 1985, pp. 332—350.

(Received June 20, 1987)