MTA SZTAKI Közlemények 32/1985 87-104

NOTES ON TRACE LANGUAGES, PROJECTION AND SYNTHESIZED COMPUTATION SYSTEMS

DANG VAN HUNG

Computer and Automation Institute,
Hungarian Academy of Sciences

ABSTRACT

Connection between trace languages and projective products is pointed out, the parallel product is defined. The way how it can be used in analyzing synthesized computation systems is presented. Relation of the trace languages to safe Petri-nets is considered, too.

1. INTRODUCTION

Trace languages and projective products are used to represent behaviour of parallel computation systems. Their relation to Petri-nets have been studied by A.Mazukiewicz [8] and E.Knuth [1]. The application of trace languages in representing projective products is presented in [2].

It is worth pointing out the connection between that concepts, how the projective products can be used to represent trace languages. In this paper we shall be concerned with those problems and their applications in studying behaviour of systems synthesized from component systems, which has attracted a great deal of attention to our knowledge.

The next section is devoted to considering the connection between trace languages and projective products. In the third section we attempt to apply this connection to study the behaviour of synthesized computation systems. The way of verification of such systems is discussed in the last one.

2. REPRESENTING TRACE LANGUAGES VIA PROJECTIVE PRODUCTS

The main objects concerned to in this section are trace languages and projective products. For their details we refer to [1,2,8] . Here we recall only the basic definitions.

Let Σ be a finite alphabet. A binary symmetric and irreflexive relation I over Σ is said to be an "independency" one. Now define "~" as the least equivalence relation on Σ^* satisfying the condition:

$$a,b \in I \Rightarrow w'abW'' \sim w'baw''$$

for all strings w', $w'' \in \Sigma^*$ and all symbols $a,b \in \Sigma$. Traces (with respect to I) are defined as equivalence classes of the relation $\sim$. The trace containing the word w (with respect to I) is denoted by $[w]_I$, and the set of words belonging to trace T is denoted by $\{T\}$.

Suppose that $w_1, w_2, \ldots, w_m \in \Sigma^*$. The projective product of $w_1, w_2, \ldots, w_m$ (denoted by $\bigotimes_{i=1}^{n} w_i$) is the set $\{w \in \Sigma^* \mid \forall i=1,2,\ldots,m,\ w|_{w_i} = w_i\}$, where $w|_v$ denotes the projection of w into the set of symbols constituting v.

The constructing of the independency relation, with respect to which the given projective products is a trace, has been presented in [2]. Now we consider the reverse problem. In doing so, we need the following concepts, which has been presented detailly in [6].

Every independency relation is called sir-relation. Let I be sir-relation, families of subsets $\overline{\text{ken}}$ (I) and ken (I) of Σ are defined as follows:

$\mathrm{ken}(I) = \{A \mid \forall a,b \in A, (a,b) \in I \cup id \,\&\, \forall c \notin A, \exists a \in A, (a,c) \notin I\}$,

$\overline{\mathrm{ken}}(I) = \{A \mid \forall a,b \in A, (a,b) \notin I \,\&\, \forall c \notin A, \exists a \in A, (a,c) \in I\}$,

where id denotes identify relation.

$\mathrm{ken}(I)$, $\overline{\mathrm{ken}}(I)$ are coverings of Σ. Reversely, from any covering $\mathcal{G}$ of Σ, we construct a sir-relation $\mathrm{sir}(\mathcal{G})$ as the following:

$$\mathrm{sir}(\mathcal{A}) = \{(a,b) \mid a \neq b \,\&\, \forall A \in \mathcal{A},\ a \notin A \text{ or } b \notin A\}.$$

Corollary 1:

For every covering $\mathcal{A}$ of Σ

a/ $\forall B \in \mathcal{G}$, $\exists A \in \overline{\mathrm{ken}}(\mathrm{sir}(\mathcal{A}))$ such that $B \subseteq A$,

b/ $\mathrm{Sir}(\mathcal{G}) = \mathrm{sir}(\overline{\mathrm{ken}}(\mathrm{sir}(\mathcal{G})))$.

Now, let T be a trace language over Σ under I, $T = [L]_I$ (L is a word-language), $\mathcal{A}$ be any covering of Σ such that $\mathrm{sir}(\mathcal{G}) = I$, $\mathcal{G} = \{A_1, A_2, \ldots, A_n\}$. Denote by h_i, $i=1,2,\ldots,n$ the projections from Σ to A_i, $i=1,2,\ldots,n$;

$$h_i(a) = \underline{\text{if}}\ a \in A_i\ \ \underline{\text{then}}\ a\ \underline{\text{else}}\ e,$$

where e is the empty word. For every $i=\overline{1,n}$, h_i can be extended to a homomorphism from Σ^* to A_i^* by the usual way.

Theorem 1: $\forall t \in T$, there exist uniquely $w_1, w_2, \ldots, w_n$, $w_i \in A_i^*$, $i=1,2,\ldots,n$ such that

$$t = \bigotimes_{i=1}^{n} w_i .$$

Proof: Take $w \in t$ and put $w_i = h_i(w)$, $i=1,2,\ldots,n$. Let $w' \sim w$. By definition of relation $\sim$, there exist $w^{(1)}, w^{(2)}, \ldots, w^{(m)}$ such that $w^{(1)} = w$, $w^{(m)} = w'$ and $\forall j=1,2,\ldots,m-1$

$$w^{(j)} = w_1^{(j)} ab\, w_2^{(j)}$$
$$w^{(j+1)} = w_1^{(j)} ba\, w_2^{(j)},\ (a,b) \in I .$$

We show by induction on j that:

$$\forall i=1,2,\ldots,n, \quad h_i(w)=h_i(w^{(j)}).$$

Because $w^{(1)}=w$, the case of $j=1$ is trivial. If $(a,b)\in I=\mathrm{sir}(\mathcal{A})$ then $\forall i=1,2,\ldots,n$, $a\notin A_i$ or $b\notin A_i$. Hence $h_i(ab)=h_i(ba)$ and $h_i(w^{(j)})=h_i(w^{(j+1)})$.

The inductive hypothese gives $h_i(w^{(j)}) = h(w)$. So $h_i(w')=h_i(w)$, $\forall i = 1,2,\ldots,n$, $\forall\ w'\sim w$. This implies that $w_1, w_2,\ldots, w_n$ are defined and by the definition of projective product,

$$w'\in \bigotimes_{i=1}^{n} w_i \ .$$

We have shown that $\forall t, \forall w\in t, \ t \subseteq \bigotimes_{i=1}^{n} h_i(w)$.

Now, by induction on the length $|w|$ of w, we show that

$$\bigotimes_{i=1}^{n} h_i(w)\subseteq t \ \forall t\in T\ , \ t = [w]_I \ .$$

When $|w|=1$, it is obvious since t and $\bigotimes_{i=1}^{n} h_i(w)$ contains exactly one element.

Suppose that $\bigotimes_{i=1}^{n} h_i(w) \subseteq t$, $\forall w, |w|\leq k$, $k\geq 1$. Let $w''=wa$ and $w'\in \bigotimes_{i=1}^{n} h_i(w'')$. Then

$$h_i(w'') = \begin{cases} h_i(w) & \text{if } a\notin A_i, \\ h_i(w)a & \text{if } a\in A_i \ . \end{cases}$$

Because a has an occurence in w', w' can be written in the form

$$w' = y_1\ a\ y_2\ ,$$

where y_2 does not contain an occurence of letters in A_i containing a by $\mathrm{sir}(\mathcal{A})=I$. Hence:

$$h_i(w')=h_i(y_1\,a\,y_2)=\begin{cases} h_i(y_1y_2), & a\notin A_i \\ h_i(y_1a), & a\in A_i \end{cases} .$$

Of course, $h_i(y_1y_2)= h_i(w)$, $h_i(y_1a)= h_i(w)a$. Therefore $h_i(y_1y_2)= h_i(w)$, $\forall i=1,2,\dots,n$ and since that, $y_1y_2\in[w]_I$ by inductive hypotheses. For every b having an occurence in y_2, $(a,b)\in I=\mathrm{sir}(\mathcal{A})$. This implies that $y_1ay_2 \sim y_1y_2a \sim wa$.

To complete the proof of theorem 1, we show that the representation $t=\bigotimes_{i=1}^{n} w_i$ is unique. But this fact is obvious by definition of projective product.

Combining theorem 1 and theorem 5 (in [2]) gives that a set of words in Σ^* is a trace if and only if it is a projective product of some words.

For the given independency relation I, we prefer to use this representation in the case of $\mathcal{A}=\overline{\mathrm{ken}}(I)$ and for the given trace language we prefer to consider the case when I is the smallest relation, with respect to which T is trace language.

From the representation of traces, we can define the parallel concatenation of trace languages, which is useful for researching the concurrency of combination of parallel computation systems.

Let $\Sigma_1,\Sigma_2,\dots,\Sigma_m$ be alphabets (not necessarily disjoint), $I_1,I_2,\dots,I_m$ be sir-relations on $\Sigma_1,\Sigma_2,\dots,\Sigma_m$ respectively. Suppose that

$$\overline{\mathrm{ken}}(I_i)=\{A_{n_{i-1}+1},\ A_{n_{i-1}+2},\dots,\ A_{n_i}\},\quad i=1,2,\dots,m,$$

$n_o=0$.

Let $t_1,t_2,\dots,t_m$ be traces on $\Sigma_1,\dots,\Sigma_m$ with respect to $I_1,I_2,\dots,I_m$ respectively. By theorem 1, there exist

$w_1,w_2,\ldots,w_{n_m}$ such that:

$$t_i=\bigotimes_{j=n_{i-1}+1}^{n_i} w_j,\ i=1,2,\ldots,m,\ w_j\in A_j^*,\ j=1,2,\ldots,n_m\ .$$

It follows from theorem 1 that if

$$\bigotimes_{j=1}^{n_m} w_j\neq\emptyset,\ t=\bigotimes_{j=1}^{n_m} w_j$$

is a trace over $\Sigma=\Sigma_1\cup\Sigma_2\cup\ldots\cup\Sigma_m$ (with respect to $\mathrm{sir}(\bigcup_{i=1}^{m}\overline{\mathrm{ken}}(I_i))$).

Definition 1: The trace t defined as above is called parallel concatenation of $t_1,t_2,\ldots,t_m$ and is denoted by $t_1\times t_2\times\ldots\times t_n$.

The following preposition shows the relation of $\mathrm{sir}(\bigcup_{i=1}^{m}\overline{\mathrm{ken}}(I_i))$ to $I_1,I_2,\ldots,I_m$ (we restrict our attention to the case $m=2$).

Preposition 1: The parallel concatenation t of two traces t_1 and t_2 is a trace on $\Sigma_1\cup\Sigma_2$ with respect to

$$R=((I_1\cup I_2)\setminus(\Sigma_1\cap\Sigma_2)^2\cup(I_1\cap I_2)\cup((\Sigma_1\setminus\Sigma_2)\times(\Sigma_2\setminus\Sigma_1)\cup$$
$$\cup((\Sigma_2\setminus\Sigma_1)\times(\Sigma_1\setminus\Sigma_2)).$$

Proof:

It is because of $\mathrm{sir}(\overline{\mathrm{ken}}(I_1)\cup\overline{\mathrm{ken}}(I_2)=R$. Now, with $\Sigma_1,\Sigma_2,\ldots,\Sigma_m$, $I_1,I_2,\ldots,I_m$ as above, let $T_1,T_2,\ldots,T_m$ be trace languages on $\Sigma_1,\Sigma_2,\ldots,\Sigma_m$ under $I_1,I_2,\ldots,I_m$ and $L_1,L_2,\ldots,L_m$ be languages on $\Sigma_1,\ldots,\Sigma_m$ respectively.

Definition 2: The parallel concatenation of $T_1,T_2,\ldots,T_m$ and the projective product of $L_1,L_2,\ldots,L_m$ (denoted by $T_1\times T_2\times\ldots\times T_m$ and $L_1\otimes L_2\otimes\ldots\otimes L_m$ respectively) are defined as follows:

$$T_1 \times T_2 \times \ldots \times T_m = \{t \mid t = t_1 \times t_2 \times \ldots \times t_m,\ t_i \in T_i,\ i = \overline{1,m}\},$$

$$L_1 \otimes L_2 \otimes \ldots \otimes L_m = \cup \{w_1 \otimes w_2 \otimes \ldots \otimes w_m \mid w_i \in L_i,\ i = \overline{1,m}\}.$$

It follows from theorem 1. that if $T = [L]_I$ is trace language on Σ under I and $\overline{\text{ken}}(I) = \{A_1, A_2, \ldots, A_n\}$ then

$$\{T\} \subseteq \bigotimes_{i=1}^{n} h_i(L). \quad (*)$$

The trace language equating (*) plays an important role in studying systems decomposable into sequential components. So we refer to "decomposable condition" as:

$$\{T\} = \bigotimes_{i=1}^{n} h_i(L).$$

This condition shall be concerned to in the next sections.

3. SYNTHESIZED COMPUTATION SYSTEMS AND THEIR BEHAVIOUR

In this section, computation systems take the general form presented in [3].

Definition 3: A computation system consists of:

(i) a set D (states),
(ii) an element x of D (the initial state),
(iii) a finite set Σ of operations,
(iv) a function "-" from Σ to the set of partial functions from D to D. The function - is extended to Σ^* in the usual way. We sometimes write $S = (D, \Sigma, x)$ instead of $S = (D, \Sigma, x, \bar{\ })$.

The set C_S of all computation sequences (from x) and the reachability set R_S of reachable states of computation system S are defined as

$$C_S = \{\alpha\ \Sigma^* | \bar{\alpha}(x) \text{ is defined}\},$$
$$R_S = \{ y \in D | \exists \alpha \in \Sigma^*,\ \bar{\alpha}(x) = y\}.$$

By a synthesized computation system, we think of computation one comming from these being concurrently active with some synchronization conditions. We shall confine our attention to the case when synchronization conditions come from the fact that some actions must take place at the same time. By constructing homomorphisms, we shall reduce that case to the one when the "contemporary" actions are common ones of some component systems. The following definition is consistent in that case:

Definition 4: Let $S_i = (D_i, \Sigma_i, x_i, {}^{-i})$, $i=1,2,\dots,n$ be computation systems. The synthesized computation system of $S_1, S_2, \dots, S_n$ (denoted by $S_1 \times S_2 \times \dots \times S_n$) is the following:

$$S = (D, \Sigma,\ , \bar{\ })$$

where

$$D = D_1 \times D_2 \times \dots \times D_n\ ,$$
$$\Sigma = \Sigma_1 \cup \Sigma_2 \times \dots \times \Sigma_n\ ,$$
$$x = (x_1, x_2, \dots, x_n), \text{ and}$$
$$\bar{\ } : \Sigma \longrightarrow (D \longrightarrow D)$$

is defined as follows:

$$\forall a \in \Sigma,\ \bar{a}(y_1, y_2, \dots, y_n) = (z_1, z_2, \dots, z_n) \text{ iff:}$$

$$z_i = \bar{a}^i(y_i),\ a \in \Sigma_i$$

$$z_i = y_i$$

in the other cases.

Now let h_i, $i=1,2,\dots,n$ be projection from Σ into Σ_i, $i=1,2,\dots,n$. When $\bar{\alpha}(x)=y$ we write $x \xrightarrow{\alpha} y$ for convenience. The connection between the behaviour of S and the behaviour of $S_1, S_2, \dots, S_n$ is showed by the following theorem:

Theorem 2:

$$C_S = C_{S_1} \otimes C_{S_2} \otimes \dots \otimes C_{S_n} .$$

Proof:

$$w \in C_{S_1} \otimes C_{S_2} \otimes \dots \otimes C_{S_n} \iff \forall i=1,2,\dots,n,$$

$$h_i(w) \in C_{S_i} \iff \forall i=1,2,\dots,n,$$

$\exists y_i \in D_i$ such that $\quad i \xrightarrow{h_i(w)} y_i \iff w \in C_S$

by the definition of "-" in S.

Remark: We sometimes deal with the set of computation sequences of a computation system, which lead the system to the state in the given set of states.

Denoting

$$C_S(>Q) = \{\alpha \in \Sigma^* \mid x \xrightarrow{\alpha} y,\ y \in Q \subseteq D\}$$

we have also: (by modifying consistently the proof of theorem 4):

$$C_S(>Q_1 \times Q_2 \times \dots \times Q_n) = C_{S_1}(>Q_1) \otimes \dots \otimes C_{S_n}(>Q_n),$$

where

$$Q_i \subseteq D_i,\ i=1,2,\dots,n.$$

Now we consider computation systems realized by Petri-nets [3]. We shall combine Petri-nets with one to another in the way presented in [5].

A Petri-net $\underline{P} = (\Pi,\Sigma,\Delta,x)$ consists of:

(i) a finite set Π of places,

(ii) a finite set Σ of transitions,

(iii) an incidence function $\Delta : \Pi\times\Sigma \cup \Sigma\times\Pi \longrightarrow \{0,1\}$,

(iv) an initial marking $x : \Pi \longrightarrow N$.

A function $y:\Pi \rightarrow N$ is called a marking. When $\Pi=\{p_1,p_2,\dots,p_k\}$, we sometimes regard a marking y as an n-dimensional vector $<y(p_2),y(p_2),\dots,y(p_k)>$.

Let $D_{\underline{P}}$ be a set of markings of P. Fór each $a \in \Sigma$ a partial function $\bar{a}: D_{\underline{P}} \rightarrow D_{\underline{P}}$ is defined as follows:

Let $y \in D_{\underline{P}}$. Then $\bar{a}(y)$ is defined if and only if $y(p_i) \geqslant \Delta(p_i,a)$ for all $p_i \in \Pi$. Suppose that $\bar{a}(y)$ is defined, then

$$\bar{a}(y)(p_i)= y(p_i) - \Delta(p_i,a) + \Delta(a,p_i),\ p \in \Pi.$$

The computation system $S_{\underline{P}}=(\Sigma,D_{\underline{P}},x)$ is said to be realized by P.

The Petri-net $\underline{P}$ is said to be safe if $R_{S_{\underline{P}}} \subseteq \{0,1\}\times\{0,1\}\times\dots\times\{0,1\}$. When $\underline{P}$ is a safe Petri-net, each marking y of $R_{S_{\underline{P}}}$ can be written as a subset M of Π, (namely, $y(p_i)=1$ iff $p_i \in M$).

Definition 4: Let $\underline{P}$ be a safe Petri-net. A relation I on Σ is said to be an independency relation generated by $\underline{P}$ iff:

$$\forall a,b \in \Sigma, (a,b) \in I \Longleftrightarrow \exists \text{ marking } M \in R_{S_{\underline{P}}}$$

such that both a and b are enabled at M and $\Delta(p,a)\,.\,\Delta(p,b)=0$, $\forall p \in \Pi$ (a is called to be enabled at M if $\forall p \in \Pi$, $(\Delta(p,a)=1) \Rightarrow y(p) \geq 1$).

Definition 5: Trace language T is said to be realized by safe Petri-net $\underline{P}$ if T is a trace language on Σ under the independency relation generated by $\underline{P}$ and $\{T\}=C_{S_{\underline{P}}}$.

Let $\underline{P}_i=(\Pi_i,\Sigma_i,\Delta_i,x_i)$, i=1,2 be Petri-nets, S_1 and S_2 be the computation systems realized by $\underline{P}_1$ and P_2 respectively. Assuming $\Pi_1=\{p_2,\dots p_k\}$, $\Pi_2=\{p_{k+1},\dots,p_m\}$, $\Pi_1 \cap \Pi_2=\emptyset$. We consider the Petri-net $P_1 \times P_2$ received from P_1 and P_2 in the following way [see 5]:

$$\underline{P} = \underline{P}_1 \times P_2 = (\Pi, \Sigma, \Delta, x),$$

where

$$\Pi = \Pi_1 \cup \Pi_2, \; \Sigma = \Sigma_1 \cup \Sigma_2, \; \Delta(p_i, a) = \begin{cases} \Delta_1(p_i, a) \text{ if } a \in \Sigma_1, \\ 0 \quad \text{otherwise} \end{cases}$$

for i=1,2,...,k and

$$\Delta(p_i, a) = \begin{cases} \Delta_2(p_i, a) \text{ if } a \in \Sigma_2, \\ 0 \quad \text{otherwise} \end{cases}$$

for i=k+1,k+2,...,m,

$$x = (x_1, x_2).$$

Let S be the computation system realized by P.

Theorem 3:

$$S = S_1 \times S_2 .$$

Proof: Each marking of P can be written as

$$(y_1, y_2), \; y_1 \in N^k, \; y_2 \in N^{m-k} .$$

By definition of Δ, $\forall i=1,2,...,k$,

$y(p_i) \geq \Delta(p_i, a) \forall a \in \Sigma \iff y_1(p_i) \geqslant \Delta_1(p_i, a) \forall a \in \Sigma_1$,

$\forall i=k+1,k+2,...,m$,

$y(p_i) \geq \Delta(p_i, a) \forall a \in \Sigma \iff y_2(p_i) \geq \Delta_2(p_i, a) \forall a \in \Sigma_2$.

That is, a is enabled in marking y of P if and only if a is enabled in y_j when $a \in \Sigma_j$, j=1,2. Furthermore

$$\forall i = 1,2,...,k$$

$$y(p_i) - \Delta(p_i,a) + \Delta(a,p_i) = \begin{cases} y_1(p_i) - \Delta_1(p_i,a) + \Delta_1(a,p_i) & \text{if } a\in\Sigma_1, \\ y_1(p_i) & \text{if } a\notin\Sigma_1. \end{cases}$$

$$\forall_i = k+1, k+2, \ldots, m.$$

$$y(p_i) - \Delta(p_i,a) + \Delta(a,p_i) = \begin{cases} y_2(p_i) - \Delta_2(p_i,a) + \Delta_2(a,p_i) & \text{if } a\in\Sigma_2, \\ y_2(p_i) & \text{if } a\notin\Sigma_2. \end{cases}$$

Since that $y \xrightarrow{a} y' \Longleftrightarrow y_j \xrightarrow{a} y'_j$ when $a\in\Sigma_j\,(j=1,2)$ and $y_j=y'_j$ when $a\in\Sigma_j$, $(j=1,2)$. That means, $S=S_1\times S_2$

<u>Corollary 2:</u> If P_1 and P_2 are safe nets then P is a safe one.

<u>Proof:</u> Since $R_S \subseteq R_{S_1} \times R_{S_2}$.

<u>Theorem 4:</u> If T_1 and T_2 are trace languages realized by safe Petri-nets then so is $T_1\times T_2$.

<u>Proof:</u> Let safe Petri-nets P_1, P_2 be realisations of T_1, T_2 respectively. By theorem 3 and corollary 2, $P_1\times P_2$ is safe Petri-net and $C_{S_{P_1\times P_2}} = \{T_1\}\times\{T_2\}$. Of course, $\{T_1\}\times\{T_2\} = \{T_1\times T_2\}$.

$$\forall t=t_1\times t_2\in T_1\times T_2,\ \forall w\in t \Longrightarrow [w]_I \subseteq t,$$

where I is the independency relation generated by $P_1\times P_2$. This is followed from the fact that $I \subseteq sir(\overline{ken}(I_1)\cup\overline{ken}(I_2))$, where I_1 and I_2 are the independency relations generated by P_1 and P_2 respectively.

For every $w'\in t$, $w'\in[w]_{sir(\overline{ken}(I_1)\cup\overline{ken}(I_2))}$, there exist $w_1, w_2, \ldots, w_n$, $w_1=w$, $w_n=w'$, $w_i=w_i^1abw_i^2$, $w_{i+1}=w_i^1baw_i^2$, $(b,a)\in sir(\overline{ken}(I_1)\cup\overline{ken}(I_2))$, $w_1=w\in[w]_I$. If $w_i\in[w]_I$, since

$w_i, w_{i+1} \in \{T_1\} \times \{T_2\}$, there exists $m \in R_{S_{P_1 \times P_2}}$ such that both a and b are enabled at m. On the other hand, from definition of $P_1 \times P_2$ and preposition 1 it follows that if $(a,b) \in sir(\overline{ken}(I_1) \cup \overline{ken}(I_2))$ then $\forall p \in \Pi, \Delta(p,a) \cdot \Delta(p,b) = 0$. This implies that, in our case, $(a,b) \in I$ which means $w_{i+1} \in [w]_I$. The inductive principle gives $w' \in [w]_I$ and this completes the proof of theorem 4.

This theorem states the closure property of the family of trace languages realized by safe Petri-nets under parallel concatenation.

Corollary 3: A trace language $T=[L]_I$ satisfying the decomposable condition (in the 2 section) is realized by safe Petri-net. if for every $i=1,2,\ldots,n$, $h_i(L)$ is realized by safe Petri-net.

(The analogous and stronger result has been stated by E.Knuth in [2].)

Now we conclude this section by a small remark. Namely, synthesized computation systems defined in this paper, to our knowledge, are general enough to research the synchronization of asynchronised processes. In the definition of it, if $\Sigma_1, \Sigma_2, \ldots, \Sigma_{n-1}$ are disjoint pairwise and $\Sigma_n = \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_{n-1} = \Sigma$ then the systems S can be considered as the synchronization of asynchronised systems $S_1, S_2, \ldots, S_{n-1}$ by S_n. When S_n is realized by Petri-net, S turns to a multiprocessor system defined and studied detailly by P.H.Starke [7]. When S_n is a unshared producer-consumer system [3], S turns to a system synchronized by P-V operations. The same method used in studying those systems can be used to study our system also.

4. ANALYSING SYNTHESIZED SYSTEMS VIA THEIR COMPONENTS

Many properties of synthesized systems can be received through the properties of their components. Unfortunately, those properties have been based on the regularity and it is not very useful to analyze synthesized systems by analyzing their components as the regularity is preserved by synthesizing.

As for us, we think that the most useful thing of this way is in verifying systems and proving the correctness of translating from one to another.

This section is devoted to the application of the above concept in the verification of synthesized systems. We shall take the method presented in [4].

Let

$$S = S_1 \times S_2 \times \ldots \times S_n \ .$$

By [4], our task is construct an assertion system for S.

Assume that $AS_1, AS_2, \ldots, AS_n$ are assertion systems for $S_1, S_2, \ldots, S_n$ respectively, $AS_i = (V_i, E_i, M_i)$, $i = 1, 2, \ldots, n$. We construct AS for S as follows:

$$AS = (V, E, M),$$

where

$$V = V_1 \times V_2 \times \ldots \times V_n,$$

$$E = \{((v_1, v_2, \ldots, v_n), t, (v_1', v_2', \ldots, v_n')) \mid \text{if } t \in \Sigma_i,\ (v_i, t, v_i') \in E_i,\ \text{if } t \notin \Sigma_i,\ v_i = v_i'\},$$

$$M = M_1 \times M_2 \times \ldots \times M_n : V_1 \times V_2 \times \ldots \times V_n \longrightarrow 2^D,$$

$$M(v_1, v_2, \ldots, v_n) = M_1(v_1) \times M_2(v_2) \times \ldots \times M_n(v_n) \subseteq$$
$$\subseteq D_1 \times D_2 \times \ldots \times D_n \ .$$

Preposition 2: If AS_i are correct assertion systems for S_i, complete for V_i', $i= 1,2,\ldots,n$ than AS is a correct assection system for S and complete for $V'=V_1' \times V_2' \times \ldots \times V_n'$.

Proof: Denote

$$Xt = \{y \mid \exists x \in X,\ x \xrightarrow{t} y\} \quad \text{for any } X \subseteq D.$$

We have

$$\forall((v_1,v_2,\ldots,v_n),t,(v_1',v_2',\ldots,v_n')) \in E$$

$$M((v_1,v_2,\ldots,v_n)) = M_1(v_1) \times M_2(v_2) \times \ldots \times M_n(v_n) \neq \emptyset.$$

by $M_i(v_i) \neq \emptyset \ \ \forall i=1,2,\ldots,n$.

If $t \notin \Sigma_i$ then $v_i'=v_i$ and $M_i(v_i) = M_i(v_i')$. Since that

$$M((v_1,v_2,\ldots,v_n)) \xrightarrow{t} Q = (Q_1',Q_2',\ldots,Q_n')$$

where $Q_i' = Q_i$ if $t \in \Sigma_i$ and $Q_i' = M_i(v_i)$ in the otherwise. It means that AS is correct for S.

To show that AS is complete for $V'=V_1' \times V_2' \times \ldots \times V_n'$, we should note that $\forall\ v' \in V'$, $\forall x,y \in D$, $t \in \Sigma$, if $x \xrightarrow{t} y$, $x \in M(v')$ then $x_i \xrightarrow{t} y_i$ in S_i when $t \in \Sigma_i$ and $x_i=y_i$ when $t \notin \Sigma_i$. Furthermore $\forall i=1,2,\ldots,n$, $x_i \in M_i(v_i')$. Since $\forall i=1,2,\ldots,n$, AS_i is complete for V_i', there exist $v_i \in V_i$, $(v_i',t,v_i) \in E_i$ for $t \in \Sigma_i$. Hence, putting $v= \delta_1,\delta_2,\ldots,\delta_n$, $\delta_i=v_i'$ if $t \notin \Sigma_i$ and $\delta_i=v_i$ if $t \in \Sigma_i$ we have $(v',t,v) \in E$.

5. CONCLUSION

We have shown certain connections between trace languages and projective products. Mathematically, they are different from each to other, but both are introduced to for the purpose of studying the behaviour of concurrent computation systems, especially in representing their concurrency.

The approach presented in this paper can be used in studying concrete systems (such as distributed systems, multiprocessor systems) and the concurrency measure of synthesized systems.

6. REFERENCES

|1| E.Knuth: Petri-nets and regular trace languages. April, 1978, The University of Newcastle upon Tyne, Computing Laboratory.

|2| E.Knuth, Gy.Győry, L.Rónyai: A study of the projection operation. Application and theory of Petri-nets. Springer-Verlag, 1982.Vol.52.

|3| Takumi Kasai and R.E.Miller: Homomorphisms between models of parallel computation. J. of Comp. and Syst.Sciences, 25, (1982).

|4| Horst Müller: Inductive assertions for analysing reachability sets (in 2).

|5| C.André: Behaviour of a place-transition net on a subset of transitions (in 2).

|6| Ryszard Janicki: Nets, sequential components and concurrency relations. Theoret.Computer Science 29, (1984) 87-121.

|7| Peter H.Starke: Multiprocessor systems and their concurrency. J. of Information Processing and Cybernetics - EIK - 20, (1984), 4.

|8| A.Mazurkiewicz: Concurrent program schemes and their interpretations. DAIMI PB-78, Aarhus Univ. Press, 1977.

Megyjegyzések a nyomnyelvekről, projekciós és szintetizált számitási rendszerekről

DANG VAN HUNG

Összefoglaló

A szerző definiálja a párhuzamos szorzat fogalmát és rámutat bizonyos összefüggésekre a nyom-nyelvek és a projektiv-szorzatok között. Megmutatja, hogyan lehet ezeket felhasználni a szintetizált számitási rendszerek elemzéséhez. A biztonságos Petri-hálók és noym-nyelvek egymáshoz való viszonyát is megvizsgálja.

Замечания о языках-следах, проекционных и синтетизированных вычислительных системах

Данг Ван Хунг

Резюме

Вводится понятие параллельного продукта и показывается связь между языками-следами и проективными продуктами. Показывается как могут использоваться эти понятия для анализа синтетизированных вычислительных систем. Рассматривается также связь между безопасными сетями Петри и языками-следами.