PRACE IPI PAN • ICS PAS REPORTS

Andrea Maggiolo-Schettini,
Józef Winkowski

On equivalence of behaviour expressions

669

October 1989
WARSZAWA

INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK
INSTITUTE OF COMPUTER SCIENCE POLISH ACADEMY OF SCIENCES
00-901 WARSAW, P.O. Box 22, POLAND

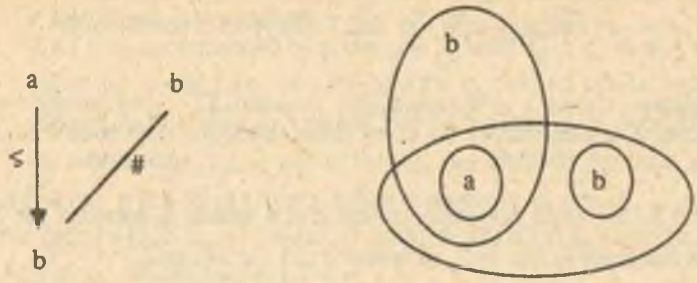
In this paper we are interested in the behavioural equivalence with respect to distributed environments. Because of difficulties with the axiomatic approach, we follow rather the approach in which concrete structures and operations are used. We represent behaviours and operations on behaviours by labelled event structures and operations on labelled event structures as in [13]. We do it with the aid of expressions, called behaviour expressions after [9]. These are constructed by means of symbols of operations, constants, and variables, where the latter ones may appear either free or bound. The constants represent given labelled event structures, whereas the free variables stand for labelled event structures which are not yet specified, and they can be replaced by arbitrary labelled event structures. This allows a stepwise specification of a behaviour. The initial state of such a specification may be a context for behaviours which will be specified in further steps.

The equivalence of behaviours is represented by a congruence of labelled event structures as in [12]. Behaviour expressions with free variables are defined to be equivalent if the results of substituting the same labelled event structures for their free variables are equivalent. Actually, the equivalences corresponding to different substitutions must fit each other in the sense which is explained in the paper.

The main result of the paper says that the equivalence thus defined of behaviour expressions is preserved under substitutions of equivalent behaviour expressions for free variables.

Usually, labelled event structures are considered in the form $E = (E, \leq, \#, \lambda)$, where E is a set of events, $\leq$ and # are a causal order and a conflict relation between events, and λ is a labelling which associates actions with events (cf. [10] and [13]). In this paper we find technically convenient exploiting rather a modified version of this concept, as in [12]. The corresponding structures, called configuration systems, are defined in section 2, following the ideas of [12] and [13]. Roughly speaking, a configuration system consists of finite conflict-free prefixes, called configurations, of a standard labelled event structure. This is illustrated in the example of a labelled event structure and the respective configuration system in fig.1.1 (where the events of the configuration system are represented by occurrences of the respective action symbols in the figure).

The paper is organized as follows. In section 2 we recall concepts and results of [12] concerning configuration systems and operations on such systems. In section 3 we recall the concepts and results of [12] about a congruence of configuration systems and about a corresponding equivalence of operations on such systems. In section 4 we formulate the concept of behaviour expressions and we give the main result.



A labelled event structure E A configuration system for E

Fig. 1.1

2. Configuration systems and operation on such systems

In this section we recall the concept of a configuration system and operations on configuration systems.

As we said, configuration systems correspond to the families of finite configurations of labelled event structures, where a configuration is (the set of events of) a conflict-free prefix of the respective labelled event structure. Operations on configuration systems are defined with the aid of a structure in the set of possible actions, called a synchronization structure (cf. [7]), which is a generalization of the synchronization algebras like in [13].

2.1 Definition. A *synchronization structure* is $\Sigma = (D_\Sigma, I_\Sigma, \otimes_\Sigma)$, where D_Σ is a set of *action symbols*, $I_\Sigma \subseteq D_\Sigma$ is a subset of symbols of *internal actions*, and $\otimes_\Sigma$ is a partial binary operation in $D_\Sigma - I_\Sigma$ which is strongly commutative ($d \otimes_\Sigma e = e \otimes_\Sigma d$ whenever either side is defined) and strongly associative ($(d \otimes_\Sigma e) \otimes_\Sigma f = d \otimes_\Sigma (e \otimes_\Sigma f)$ whenever either side is defined). All $d,e \in D_\Sigma$ such that $d \otimes_\Sigma e$ is defined are said to be *complementary*. ♦

For example, we may have $D_\Sigma = -A \cup +A \cup \{\tau\}$, $I_\Sigma = \{\tau\}$, and $\otimes_\Sigma$ defined by $(-a) \otimes_\Sigma (+a) = \tau$ for some $a \in A$, and by $(-a) \otimes_\Sigma (+a) = -a$, $(+a) \otimes_\Sigma (+a) = +a$, $(-a) \otimes_\Sigma (-a)$ undefined, for some other $a \in A$. We can interpret A as a set of symbols

of data transfers with -a and +a standing for the respective send and receive action in the mode of handshaking (if $(-a)\otimes_\Sigma(+a)=\tau$) or broadcasting (if $(-a)\otimes_\Sigma(+a)=-a$, $(+a)\otimes_\Sigma(+a)=+a$, and $(-a)\otimes_\Sigma(-a)$ is undefined).

2.2. Definition. Given a synchronization structure Σ, a *configuration system* (*c-system*) over Σ is a nonempty set S of finite functions with values in D_Σ such that:

(1) $\bigcap Z \in S$ for each nonempty subset $Z \subseteq S$ which is *bounded* in S in the sense that there exists $s \in S$ such that $z \subseteq s$ for all $z \in Z$,

(2) $\bigcup Z \in S$ for each subset $Z \subseteq S$ which is bounded in S.

We call all $s \in S$ *configurations* of S and, for each configuration $s \in S$ and each $(x,v) \in s$, we call (x,v) an *event* with the *execution symbol* x and the *action symbol* v. Given a configuration $s \in S$ and two events $e,f \in S$, we say that f *follows* e in s (resp.: f is *coincident* with e in s) iff for all configurations $t \in S$ with $t \subseteq s$ the condition $f \in t$ implies (resp.: is equivalent to) the condition $e \in t$.
By **Nil** we denote the c-system $\{\varnothing\}$. By $cs(\Sigma)$ we denote the universe of c-systems over Σ. ♦

For c-systems we have the following results.

2.3. Proposition. For all c-systems $P, S_0, S_1 \in cs(\Sigma)$, each $K \subseteq D_\Sigma$ with $I_\Sigma \subseteq K$, and each automorphism b of Σ, where an automorphism is a bijection $b: D_\Sigma \to D_\Sigma$ such that b and b^{-1} preserve I_Σ and $\otimes_\Sigma$, we have the following c-systems:

(1) the result $P|K$ of *restricting* P to K, where
$s \in P|K$ iff $s \in P$ and for all $(x,v) \in s$ we have $v \in K$,

(2) the result Pb of *relabelling* P according to b, where
$s \in Pb$ iff $s = \{(x,b(v)) : (x,v) \in p\}$ for some $p \in P$,

(3) the result $S_0;P$ of *prefixing* S_0 to P,
where $s \in S_0;P$ iff $s = \{((0,x),v): (x,v) \in s_0\}$ for some $s_0 \in S_0$
or $s = \{((0,x),v): (x,v) \in s_0\} \cup \{((1,y),w): (y,w) \in q\}$
for a maximal $s_0 \in S_0$ and some $q \in P$,

(4) the *sum* $S_0 + S_1$, where $s \in S_0 + S_1$ iff $s = \{((0,x),v): (x,v) \in s_0\}$ for some $s_0 \in S_0$ or $s = \{((1,y),w): (y,w) \in s_1\}$ for some $s_1 \in S_1$,

(5) the *parallel composition* $S_0 \| S_1$, where $s \in S_0 \| S_1$ iff s *consists of* some $s_0 \in S_0$ and $s_1 \in S_1$ in the sense that
$s = \{((0,x),v): (x,v) \in s_0 - \alpha^{-1}(s_1)\} \cup \{((1,y),w): (y,w) \in s_1 - \alpha(s_0)\}$
$\cup \{(((0,x,v),(1,y,w)), v \otimes_\Sigma w): ((x,v),(y,w)) \in \alpha\}$

for a one-to-one partial function α from s_0 to s_1 such that:

(5.1) v and w are complementary for all $((x,v),(y,w))\in\alpha$,

(5.2) if $e,f \in s_0$ are separated by some $t_0 \in S_0$ with $t_0 \subseteq s_0$, in the sense that either $e \in t_0$ and $f \notin t_0$ or $e \notin t_0$ and $f \in t_0$, then e,f can be separated by some $u_0 \in S_0$ with $u_0 \subseteq s_0$ such that there exists $u_1 \in S_1$ with $u_1 \subseteq s_1$ for which $\alpha(u_0) \subseteq u_1$ and $\alpha^{-1}(u_1) \subseteq u_0$; similarly for $e,f \in s_1$.

Moreover, f follows e in s iff in s there exists a sequence $e_1 = e, e_2, \ldots, e_n, e_{n+1} = f$ such that, for each $i\in\{1,2,\ldots,n\}$, $pr_0(e_{i+1})$ follows $pr_0(e_i)$ in s_0 or $pr_1(e_{i+1})$ follows $pr_1(e_i)$ in s_1, where, for each $e \in s$,

$pr_0(e)=(x,v)$ for e of the form $((0,x,v)$,

$pr_0(e)=(x,v)$ and $pr_1(e)=(y,w)$ for e of the form $(((0,x,v),(1,y,w))$, $v \otimes_\Sigma w)$,

$pr_1(e)=(y,w)$ for e of the form $((1,y),w)$.

Proof outline.

For (1)-(4) proofs are trivial. For (5) we proceed as follows.

Consider a nonempty $Z\subseteq S$ with an upper bound $s\in S$. Then $Z_0=\{pr_0(z):z\in Z\}$ and $Z_1=\{pr_1(z):z\in Z\}$ are nonempty bounded subsets of S_0 and S_1 respectively, and $z\in Z$ iff z consists of some $z_0\in Z_0$ and $z_1\in Z_1$ with the correspondence $\alpha_z = \alpha\cap(z_0\times z_1)$. From the properties of S_0 and S_1 we obtain that $\cap Z_0\in S_0$ and $\cap Z_1\in S_1$. On the other hand, $\cap Z$ consists of $\cap Z_0$ and $\cap Z_1$ with the correspondence $\cap(\alpha_z : z\in Z)$. Indeed given any $e,f\in\cap Z_0$ and t_0 with $t_0\subseteq s_0$ which separates e,f, for each $z\in Z$ we take $u_0(z)$ and $u_1(z)$ satisfying the requirements of (5.2) and obtain $u_0=\cap(u_0(z):z\in Z)\in S_0$ and $u_1=\cap(u_1(z):z\in Z)\in S_1$ with u_0 separating e,f. Finally, $\cap Z\in S$, as required. Similarly, $\cup Z\in S$ for each bounded $Z\subseteq S$.

The characterization of the relation f follows e in s is a direct consequence of the fact that (5.2) is equivalent to the lack in s of a non trivial cycle $e_1,e_2,\ldots,e_n, e_{n+1}=e_1$ such that $pr_0(e_{i+1})$ follows $pr_0(e_i)$ in s_0 or $pr_1(e_{i+1})$ follows $pr_1(e_i)$ in s_1 for all $1\in\{1,\ldots,n\}$. ♦

Proposition 2.3 defines the following operations on c-systems: $P \mapsto PlK$, $P \mapsto Pb$, $P \mapsto S_0;P$, $(S_0,S_1) \mapsto S_0+S_1$, $(S_0,S_1) \mapsto S_0\|S_1$. Note that the prefixing is regarded as an operation with respect to the second argument only, the first one playing the role of a parameter. Note also that prefixing an action $d\in D_\Sigma$

to a c-system P is a particular case of such an operation with d represented by a one-event c-system with an event with the label d.

For c-systems we have a natural relation of being a prefix.

2.4. Definition. Given two c-systems $P,Q \in cs(\Sigma)$, we say that P is a *prefix* of Q, written as $P << Q$, iff $P \subseteq Q$ and, for each $q \in Q$, $q \subseteq \cup P$ implies $q \in Q$. ♦

The following property is a simple consequence of the definition.

2.5. Proposition. The relation $<<$ is a chain complete partial order on $cs(\Sigma)$ with **Nil** playing the role of the least element, and the supremum of each countable chain $P_0 << P_1 << \ldots$ given by $\cup\{P_i: i \in \omega\}$, where $\omega = \{0,1,2,\ldots\}$. The operations defined in 2.3 are continuous with respect to this order, that is they preserve the suprema of countable chains in the respective cartesian powers of $cs(\Sigma)$. ♦

From the known properties of complete partial order we obtain the following result.

2.6. Proposition. Let $F: (cs(\Sigma))^{m+n} \to (cs(\Sigma))^m$ be a continuous mapping which transforms each pair (P,Q) with $P \in (cs(\Sigma))^m$ and $Q \in (cs(\Sigma))^n$ into some $R = F(P,Q) \in (cs(\Sigma))^m$. Then we have:

(1) the fixed-point equation $P = F(P,Q)$ has a least solution $h(Q)$,

(2) this solution is given by $\cup\{P_i: i \in \omega\}$, where $P_0 = \mathbf{Nil}^m$ and $P_{i+1} = F(P_i, Q)$ for $i \in \omega$,

(3) the correspondence $Q \mapsto h(Q)$ is a continuous mapping from $(cs(\Sigma))^n$ to $(cs(\Sigma))^m$.

We write $h(Q)$ as $\mathbf{fix}_P F(P,Q)$ and call the correspondence $F \mapsto h$ a *fixed-point operator*. ♦

Thanks to this result we may define a large variety of operations on c-systems.

2.7. Definition. The operations defined in 2.4 are called *basic operations* on

c-systems. The operations which can be obtained by combining basic operations with the aid of superpositions and fixed-point operators are called *definable operations*. ♦

3. Equivalence

In this section we recall the concepts and results of [12] about an equivalence of configuration systems and about a corresponding equivalence of operations on such systems.

The equivalence of configuration systems can be defined with the aid of suitable morphisms of configuration systems. These morphisms, called simulations, generalize the concept of a rooted bisimulation of processes modulo invisible events as in [1]. They are similar to the mentioned bisimulations in preserving the transitional structure of behaviours, but more subtle in preserving the possible concurrency. The equivalence we introduce corresponds to the so called "pomset bisimulation equivalence".

We start with the following simple observations.

3.1. Proposition. For each c-system $P \in \mathbf{cs}(\Sigma)$ which has a greatest configuration we have a unique c-system $\mathbf{visible}(P) \in \mathbf{cs}(\Sigma)$, called the *visible kernel* of P, where $s \in \mathbf{visible}(P)$ iff $s=\{(x,v) \in p : v \notin I_\Sigma\}$ for some $p \in P$. ♦

3.2. Proposition. For each c-system $P \in \mathbf{cs}(\Sigma)$ and each configuration $p \in P$ we have a unique c-system $P\text{-}p \in \mathbf{cs}(\Sigma)$, called the *continuation* of P from p, where $s \in P\text{-}p$ iff $s=q\text{-}p$ for some $q \in P$ with $p \subseteq q$. ♦

We want now to define a concept of simulation for c-systems.

3.3. Definition. A *transition* from one system c-system to another is a triple $P \Rightarrow_A Q$, where P is a c-system, Q is a continuation of P from a configuration $p \in P$, and $A=\mathbf{visible}(\downarrow p)$ for the prefix $\downarrow p=\{q \in P: q \subseteq p\}$. ♦

3.4. Definition. An *isomorphism* from a c-system P to a c-system Q is a bijection $b: \cup P \to \cup Q$ such that, for all $(x,v),(y,w),p,q$, $(y,w)=b((x,v))$ implies $v=w$, $p \in P$ implies $b(p) \in Q$, and $q \in Q$ implies $b^{-1}(q) \in P$. ♦

3.5. Definition. By a *simulation* of a c-system P in a c-system Q we mean a triple $\rho: P \to Q$, where:

(1) $\rho \subseteq P \times Q$,

(2) $(\varnothing, \varnothing) \in \rho$,

(3) for each $(p,q) \in \rho$ and each transition $P\text{-}p \Rightarrow_A (P\text{-}p)\text{-}p'$ there exists

a transition $Q\text{-}q \Rightarrow_B (Q\text{-}q)\text{-}q'$ such that $(p \cup p', q \cup q') \in \rho$ and B is isomorphic to A.

If $(\varnothing, q) \in \rho$ only for $q = \varnothing$, then we call $\rho: P \to Q$ a *rooted* simulation. If $\rho^{op}: Q \to P$, where $\rho^{op} = \{(q,p): (p,q) \in \rho\}$, is also a simulation then we call $\rho: P \to Q$ a *bisimulation.* ♦

For example, $\mathbf{id}_P: P \to Q$ with $P << Q$ and $\mathbf{id}_P$ denoting the identity relation in P, written also as $P \overset{<<}{\to} Q$, is a simulation. Similarly, $B_P : P \to P+P$ with

$$(p,q) \in B_P \text{ iff } q = \{((0,x),v): (x,v) \in p\} \text{ or } q = \{((1,x),v): (x,v) \in p\}$$

is a rooted bisimulation.

From the definition we have the following results and concepts.

3.6. Proposition. If $\rho : P \to Q$ and $\sigma : Q \to R$ are simulations (resp: rooted simulations, bisimulations) then $\rho \circ \sigma : P \to R$ with

$$\rho \circ \sigma = \{(p,r): (p,q) \in \rho \text{ and } (q,r) \in \sigma \text{ for some } q \in Q\}$$

is also a simulation (resp: a rooted simulation, a bisimulation). ♦

3.7. Proposition. The binary relation in $\mathbf{cs}(\Sigma)$ defined by

$$P \approx Q \text{ iff there exists a rooted bisimulation } \rho: P \to Q$$

is an equivalence relation. We call it *behavioural equivalence*, and if $P \approx Q$ then we say that P and Q are *behaviourally equivalent.* ♦

For example, for $\tau \in I_\Sigma$, $d \in D_\Sigma$, and arbitrary c-systems P,Q,R, we have:

$P+Nil \approx P$	$P+Q \approx Q+P$	$(P+Q)+R \approx P+(Q+R)$
$P+P \approx P$	$P\|\|Q \approx Q\|\|P$	$(P\|\|Q)\|\|R \approx P\|\|(Q\|\|R)$
$d;(\tau;P) \approx d;P$	$P+(\tau;P) \approx \tau;P$	$(d;(P+(\tau;Q)))+(d;Q) \approx d;(P+(\tau;Q))$.

3.8. Proposition. The c-systems over a synchronization structure Σ and their simulations constitute a category which we denote by $\mathbf{cs}(\Sigma)$. For each cardinal m we have the cartesian power $(\mathbf{cs}(\Sigma))^m$ of this category. ♦

3.9. Proposition. The category $\mathbf{cs}(\Sigma)$ has colimits of countable chains. The colimit of each chain $P_0 << P_1 << \ldots$ coincides with the supremum $P = \cup\{P_i: i \in \omega\}$. Similarly for the cartesian powers of $\mathbf{cs}(\Sigma)$. Moreover,

$\rho = \cup\{\rho_i : i \in \omega\}$ for each commutative diagram as in fig. 3.1 with the unique $\rho: P \to Q$ resulting from the universal properties of colimits.

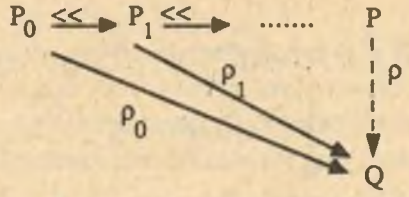


Fig.3.1

Proof outline. It suffices to notice that the commutativity of the diagram in fig.3.1 means that each ρ_i with $i \leq j$ is the restriction of ρ_j to P_i and that $\rho = \cup\{\rho_i : i \in \omega\}$ is the unique relation such that the diagram in fig.3.1 commutes. ♦

3.10. Definition. Given a functor $F : cs(\Sigma)^m \to cs(\Sigma)^n$, we say that F is *continuous* iff it preserves colimits and the prefix order, where the preservation of the prefix order means that $F(P) \overset{F(<<)}{\to} F(Q)$ is $F(P) \overset{<<}{\to} F(Q)$ for each $P << Q$, and we say that F *preserves rooted bisimulations* iff for each rooted bisimulation $\rho : P \to Q$, the simulation $\rho: F(P) \to F(Q)$ is a rooted bisimulation. ♦

3.11. Proposition. Each definable operation on c-systems can be extended in a canonical way to a continuos functor which preserves rooted bisimulations.

Proof outline.

For the basic operations the proof is straightforward. For example, for $\rho_0 : S_0 \to S_0'$ and $\rho_1 : S_1 \to S_1'$ we define $\rho_0 || \rho_1 : S_0 || S_0' \to S_1 || S_1'$ by $\rho_0 || \rho_1 : \{(s,s') : s$ consists of s_0 and s_1, s' consists of s_0' and s_1', $(s_0, s_0') \in \rho_0$, $(s_1, s_1') \in \rho_1\}$, and the continuity follows from 2.5 and 3.9.

In order to prove that the property holds for each definable operation, it suffices to consider a continuous functor $F: (cs(\Sigma))^{m+n} \to (cs(\Sigma))^m$ which preserves rooted bisimulations and to prove that $T \mapsto fix_S F(S,T)$ extends to a continuous functor which preserves rooted bisimulations.

Suppose that $\rho : T \to T'$ is a bisimulation and consider the least solutions f(T) and f(T') of the respective fixed point equations S=F(S,T) and S=F(S,T'). As F is continuous, we obtain the commutative diagram in fig.3.2 with a unique simulation $\sigma : f(T) \to f(T')$. From the uniqueness of σ it follows that the correspondence

$\rho \mapsto \sigma$, where $f(\rho) = \cup\{\rho_i : i \in \omega\}$ with $\rho_0 = \varnothing^m$ and $\rho_{i+1} = F(\rho_i, \rho)$ for $i \in \omega$, is a functor, and that this functor is continuous. It is easy to see that this functor preserves rooted bisimulations.

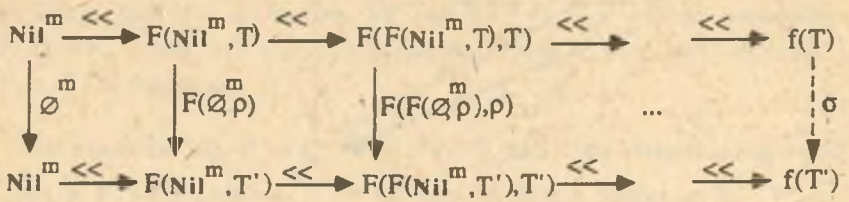


Fig. 3.2

From the definition of the behavioural equivalence of c-system and 3.11 we obtain the following result.

3.12. Proposition. The behavioural equivalence of c-systems is a congruence for all definable operations on c-systems. ♦

The concept of behavioural equivalence of c-systems extends on definable operations on c-systems.

3.13. Definition. Given two functors $F: (cs(\Sigma))^m \to (cs(\Sigma))^n$ and $G: cs(\Sigma)^m \to cs(\Sigma)^n$, we say that such functors are *equivalent*, written as $F \approx G$, iff there exists a natural transformation $\rho : F \to G$ which consists of rooted bisimulations, that is a family

$$\rho = (\rho(P) : F(P) \to G(P) : P \in (cs(\Sigma))^m)$$

of rooted bisimulations such that, for each simulation $\sigma: P \to Q$, the diagram in fig. 3.3 commutes. ♦

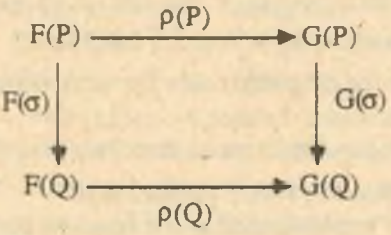


Fig. 3.3

For example, for $P \mapsto P$ and $P \mapsto P+P$, the canonical extensions of these operations to continuous, rooting and bisimulation preserving functors, are equivalent with the equivalence given by $(B_P: P \mapsto P+P:\ P \in cs(\Sigma))$ with B_P as in the example to 3.6.

So the operations themselves can be regarded to be equivalent.

Definable operations on c-systems are constructed by combining basic operations with the aid of superpositions and fixed-point operators. We shall be able to show that operations constructed in the same manner from equivalent ones are equivalent as well.

For example, the operations $(Q,R) \mapsto \mathbf{fix}_P(P \| Q)+R$ and $(Q,R) \mapsto R+\mathbf{fix}_P(Q \| P)$ are constructed in the same manner from the following equivalent operations : $(P,Q) \mapsto P \| Q$ and $(P,Q) \mapsto Q \| P$, $Q \mapsto \mathbf{fix}_P(P \| Q)$ and $Q \mapsto \mathbf{fix}_P(Q \| P)$, $(S,R) \mapsto S+R$ and $(S,R) \mapsto R+S$, and so they will be equivalent.

The general result is as follows.

3.14. Proposition. If two definable operations on c-systems are constructed in the same manner from equivalent definable operations (that is from definable operations whose canonical extensions to functors are equivalent) then they are equivalent (that is their canonical extensions to functors are equivalent).

Proof outline.

The only nontrivial part of the proof is about operations defined by fixed-point operators. Thus it suffices to consider two equivalent, continuous, rooted bisimulations preserving functors $F: (cs(\Sigma))^{m+n} \to (cs(\Sigma))^m$ and $G: (cs(\Sigma))^{m+n} \to (cs(\Sigma))^m$ with the equivalence given by a family ρ of rooted bisimulations $\rho(P,Q) : F(P,Q) \to G(P,Q)$, and find a suitable family $\sigma(Q) : f(Q) \to g(Q)$ for $f(Q) = \mathbf{fix}_P F(P,Q)$ and $g(Q) = \mathbf{fix}_P G(P,Q)$. To this end we consider the diagram in fig.3.4 which is commutative due to the continuity of F and G and due to the fact that ρ is a natural transformation from F to G. By 3.9 we obtain a unique rooted bisimulation $\sigma(Q) : f(Q) \to g(Q)$ which completes this diagram to a commutative one. From the uniqueness of $\sigma(Q)$ we obtain that the family $\sigma = (\sigma(Q) : f(Q) \to g(Q) : Q \in cs(\Sigma))^n)$ is a natural transformation, as required. ♦

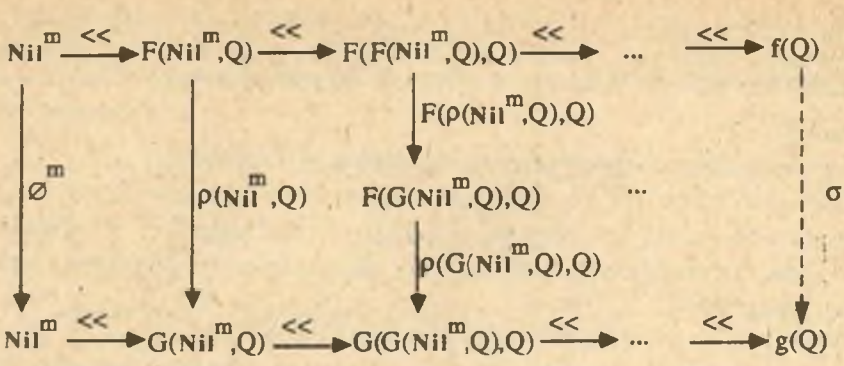


Fig. 3.4

4. Behaviour expressions and their equivalence

We represent configuration systems with the aid of expressions, called behaviour expressions. These expressions are constructed by means of constants, variables, symbols of basic operations on configuration systems, and fixed-point operators. For constants and variables denoting configuration systems, and for operations as in section 2, such expressions can be interpreted as definable operations on configuration systems. The required equivalence of behaviour expressions is defined in terms of the equivalence of operations on configuration systems.

4.1. Definition. Let Σ be a synchronization structure, X an infinite set of variables, and C a nonempty set of constants such that $D_\Sigma \subseteq C$. A *behaviour expression* over Σ,X,C is an expression σ with a set $fv(\sigma) \subseteq X$, called the set of *free variables* of σ, where:

(1) each constant $c \in C$ is a behaviour expression with $fv(c) = \emptyset$,
(2) each variable $x \in X$ is a behaviour expression with $fv(x) = \{x\}$,
(3) for any behaviour expression σ, and each $K \subseteq D_\Sigma$ as in 2.3, $\sigma|K$ is a behaviour expression with $fv(\sigma|K) = fv(\sigma)$,
(4) for any behaviour expression σ, and each automorphism b of Σ as in 2.3, σb is a behaviour expression with $fv(\sigma b) = fv(\sigma)$,
(5) for each constant $c \in C$, and any behaviour expression σ, $c;\sigma$ is a behaviour expression with $fv(c;\sigma) = fv(\sigma)$,
(6) for any behaviour expressions σ_0 and σ_1, and for $\$ \in \{+, ||\}$, $(\sigma_0 \$ \sigma_1)$ is a behaviour expression with $fv((\sigma_0 \$ \sigma_1)) = fv(\sigma_0) \cup fv(\sigma_1)$,

(7) for any behaviour expression σ with $fv(\sigma) = \{x_1,...,x_n\}$, and for any behaviour expressions $\sigma_1,...,\sigma_n$, the result of substituting $\sigma_1,...,\sigma_n$ for $x_1,...,x_n$, written as $[\sigma_1/x_1,...,\sigma_n/x_n](\sigma)$, is a behaviour expression with $fv([\sigma_1/x_1,...,\sigma_n/x_n](\sigma)) = fv(\sigma_1) \cup ... \cup fv(\sigma_n)$,

(8) for any behaviour expression σ, and any variable $x \in X$, $fix_x\ \sigma$ is a behaviour expression with $fv(fix_x\ \sigma) = fv(\sigma) - \{x\}$.

Following a common custom, we write a behaviour expression σ with $fv(\sigma) = \{x_1,...,x_n\}$ as $\sigma(x_1,...,x_n)$. By $bex(\Sigma,X,C)$ we denote the set of all behaviour expressions over Σ,X,C. ♦

For example, $(P \parallel Q)$, $fix_P\ (P \parallel Q)$, $(fix_P\ (P \parallel Q)) + R$, where $P,Q,R \in X$, are behaviour expressions with $fv((P \parallel Q)) = \{P,Q\}$, $fv(fix_P\ (P \parallel Q)) = \{Q\}$, and $fv((fix_P(P \parallel Q)) + R) = \{Q,R\}$, respectively. As usual, we may assume a precedence of symbols of operations and avoid unnecessary brackets. We may also use the standard concept of binding variables.

4.2. Definition. Given Σ,X,C as in 4.1, we define an *interpretation* of behaviour expressions over Σ,X,C as a function J assigning an operation on $cs(\Sigma)$ to each behaviour expression in $bex(\Sigma,X,C)$ such that:

(1) for each behaviour expression $\sigma(x_1,...,x_n)$, $J(\sigma(x_1,...,x_n))$ is an n-ary operation on $cs(\Sigma)$, written as $(S_1,...,S_n) \mapsto \sigma(S_1,...,S_n)$,

(2) for each $c \in D_\Sigma$, $J(c)$ is a nullary operation giving a one-element c-system with an event with the label c,

(3) for each variable $x \in X$, $J(x)$ is the identity operation $S \mapsto S$,

(4) for each behaviour expression $\sigma(x_1,...,x_n)$ and each $K \subseteq D_\Sigma$ as in 2.4, $J(\sigma(x_1,...,x_n)|K)$ is the operation $(S_1,...,S_n) \mapsto \sigma(S_1,...,S_n)|K$,

(5) for each behaviour expression $\sigma(x_1,...,x_n)$, and each automorphism b of Σ as in 2.4, $J(\sigma(x_1,...,x_n)\ b)$ is the operation $(S_1,...,S_n) \mapsto \sigma(S_1,...,S_n)\ b$,

(6) for each constant $c \in C$, and each behaviour expression $\sigma(x_1,...,x_n)$, $J(c;\sigma(x_1,...,x_n))$ is the operation $(S_1,...,S_n) \mapsto J(c);\sigma(S_1,...,S_n)$,

(7) for each behaviour expression $\sigma(x_1,...,x_n)$ of the form $\sigma_0(x_{i_1},...,x_{i_p})\ \$\ \sigma_1(x_{j_1},...,x_{j_q})$, where $\$ \in \{+, \parallel\}$, $J(\sigma(x_1,...,x_n))$ is the operation $(S_1,...,S_n) \mapsto \sigma_0(S_{i_1},...,S_{i_p})\ \$\ \sigma_1(S_{j_1},...,S_{j_q})$,

(8) for each behaviour expression $\sigma(x_1,...,x_n)$, and arbitrary behaviour expressions $\sigma_1(x_{11},...,x_{1m_1}),...,\sigma_n(x_{n1},...,x_{nm_n})$, $J([\sigma_1(x_{11},...,x_{1m_1})/x_1,...,\sigma_n(x_{n1},...,x_{nm_n})/x_n](\sigma(x_1,...,x_n)))$ is the operation

$(S_{11},...,S_{1m_1},...,S_{n1},...,S_{nm_n}) \mapsto \sigma(\sigma_1(S_{11},...,S_{1m_1}),...,\sigma_n(S_{n1},...,S_{nm_n}))$,

(9) for each behaviour expression $\sigma(x_1,...,x_n)$ and $x_j \in \{x_1,...,x_n\}$, $J(\mathbf{fix}_{x_j}\sigma(x_1,...,x_n))$ is the operation
$(S_1,...,S_{j-1},S_{j+1},...,S_n) \mapsto \mathbf{fix}_{S_j}\sigma(S_1,...,S_{j-1},S_j,S_{j+1},...,S_n)$;
for a variable $x \notin \{x_1,...,x_n\}$, $J(\mathbf{fix}_{x_j}\sigma(x_1,...,x_n))$ is defined as $J(\sigma(x_1,...,x_n))$. ♦

From the definition of an interpretation we obtain the following property.

4.3. Proposition. Each interpretation of behaviour expressions over Σ,X,C is determined uniquely by its values for all constants $c \in C$. ♦

4.4. Definition. Behaviour expressions $\sigma(x_1,...,x_n) \in \mathbf{bex}(\Sigma,X,C)$ and $\sigma'(x_1,....x_n) \in \mathbf{bex}(\Sigma,X,C)$ are said to be *equivalent* for an interpretation J, written as $\sigma(x_1,...,x_n) =_J \sigma'(x_1,...,x_n)$, iff the canonical functorial extensions of the operations $J(\sigma(x_1,...,x_n))$ and $J(\sigma'(x_1,...,x_n))$ are equivalent in the sense of 3.13. ♦

From 3.14 we obtain the following result.

4.5. Theorem. Let J be an interpretation of behaviour expressions from $\mathbf{bex}(\Sigma,X,C)$. Then we have:

(1) for each $K \subseteq D_\Sigma$ as in 2.4,
$\sigma(x_1,...,x_n) =_J \sigma'(x_1,...,x_n)$ implies $\sigma(x_1,...,x_n)|K =_J \sigma'(x_1,...,x_n)|K$,

(2) for each automorphism b of Σ as in 2.4,
$\sigma(x_1,...,x_n) =_J \sigma'(x_1,...,x_n)$ implies $\sigma(x_1,...,x_n)b =_J \sigma'(x_1,...,x_n)b$,

(3) for each constant $c \in C$,
$\sigma(x_1,...,x_n) =_J \sigma'(x_1,...,x_n)$ implies $c;\sigma(x_1,...,x_n) =_J c;\sigma'(x_1,...,x_n)$,

(4) for each behaviour expression $\sigma(x_1,...,x_n)$ of the form $\sigma_0(x_{i_1},...,x_{i_p}) \$ \sigma_1(x_{j_1},...,x_{j_q})$, and each specification $\sigma'(x_1,...,x_n)$ of the form $\sigma'_0(x_{i_1},...,x_{i_p}) \$ \sigma'_1(x_{j_1},...,x_{j_q})$, where $\$ \in \{+, ||\}$, if $\sigma_0(x_{i_1},...,x_{i_p}) =_J \sigma'_0(x_{i_1},...,x_{i_p})$ and $\sigma_1(x_{j_1},...,x_{j_q}) =_J \sigma'_1(x_{j_1},...,x_{j_q})$ then $\sigma(x_1,...,x_n) =_J \sigma'(x_1,...,x_n)$,

(5) for all behaviour expressions $\sigma(x_1,...,x_n),\sigma'(x_1,...,x_n)$, $\sigma_1(x_{11},...,x_{1m_1}),...,\sigma_n(x_{n1},...,x_{nm_n}),\sigma'_1(x_{11},...,x_{1m_1}),...,\sigma'_n(x_{n1},...,x_{nm_n})$ such that $\sigma(x_1,...,x_n) =_J \sigma'(x_1,...,x_n)$,

$\sigma_1(x_{11},...,x_{1m_1}) \sim_J \sigma'_1(x_{11},...,x_{1m_1}),...,\sigma_n(x_{n1},...,x_{nm_n}) \sim_J \sigma'_n(x_{n1},...,x_{nm_n})$

we have

$[\sigma_1(x_{11},...,x_{1m_1})/x_1,...,\sigma_n(x_{n1},...,x_{nm_n})/x_n](\sigma(x_1,...,x_n))$

$\sim_J [\sigma'_1(x_{11},...,x_{1m_1})/x_1,...,\sigma'_n(x_{n1},...,x_{nm_n})/x_n](\sigma'(x_1,...,x_n))$,

(6) $\sigma(x_1,...,x_n) \sim_J \sigma'(x_1,...,x_n)$ implies $\mathbf{fix}_{X_j}\sigma(x_1,...,x_n) \sim_J \mathbf{fix}_{X_j}\sigma'(x_1,...,x_n)$. ♦

For example, $(\mathbf{fix}_X(x \parallel y)) + z \sim_J z + (\mathbf{fix}_X(y \parallel x))$ for all interpretations J (cf. the example to 3.14).

5. Conclusions

We have considered the equivalence problem for behaviour expressions with free variables. These expressions represent behaviours given by configuration systems. For such expressions we have equivalences which are preserved by substitutions. These equivalences are defined in terms of a bisimulation concept for configuration systems. This allows a semantic verification of equivalence of behaviour expressions, which is essential since we do not offer any axiomatic characterization of the equivalence. On the other hand, finding such a characterization would also be important as a step towards a formal procedure for equivalence verification.

References

1. J. A. Bergstra and J. W. Klop, Algebra of Communicating Processes with Abstraction, Theoret. Comp. Sci. **37** (1988), 77-121.
2. G. Boudol and I. Castellani, On the Semantics of Concurrency: Partial Orders and Transition Systems, Springer LNCS **259**, 1987,123 -137.
3. P. Degano and U. Montanari, Concurrent Histories - A Basis for Observing Distributed Systems , J.Comput.System Sci. **34** (1987), 422 - 461.
4. P. Degano, R. De Nicola and U. Montanari, Partial Order Descriptions and Observations of Nondeterministic Concurrent Processes, to appear in Springer LNCS.
5. R. Devillers, On the Definition of a Bisimulation Based on Partial Words, Petri Net Newsletters **29** (1988).
6. R. van Glabbeek and F. Vaandrager, Petri Net Models for AlgebraicTheories of Concurrency, Springer LNCS **259**, 1987, 224 - 242.
7. A. Maggiolo-Schettini and J. Winkowski, A Compositional Semantics for Timed Petri Nets, to appear in Fundamenta Informaticae.
8. R. Milner, A Calculus of Communicating Systems, Springer LNCS **92**, 1980.

9. R. Milner, A Complete Axiomatization for Observational Congruence of Finite State Behaviours, LFCS Report Series, ECS-LFCS-86-8, Dept. of Comp. Sc., University of Edinburgh, August 1986.
10. M. Nielsen, G. D. Plotkin and G. Winskel, Petri Nets, Event Structures and Domains, Part I, Theoret.Comp. Sci. **13** (1981), 85 - 108.
11. J. Staples and V. L. Nguyen: A Fixpoint Semantics for Nondeterministic Data Flow, Journal of the ACM **32** (1985), 411- 444.
12. J. Winkowski, A Equivalence of Communicating Processes in Distributed Environments, Fundamenta Informaticae **XII** (1989), 97-128.
13. G. Winskel, Event Structure Semantics for CCS and Related Languages, Springer LNCS **140**, 1982, 561-576.