Reihe Informatik
4/1990
The consistency of a noninterleaving and an interleaving model for full TCSP

Christel Baier
Mila E. Majster-Cederbaum

Extended abstract
November 1990
1. Introduction

Various formalisms have been proposed in the past for the description of nondeterministic concurrent systems, the most well-known of which are CCS [14,15], ACP [2] and TCSP [6,13,17]. These languages or calculi have been given a variety of semantical descriptions [1,2,3,4,5,6,7,8,10,11,12,18,19,20,21]. A first classification of this semantics distinguishes between interleaving and noninterleaving models.

In noninterleaving models as [3,4,8,10,11,12,18,21,23] an attempt is made to capture 'true parallelism' whereas interleaving models as [1,2,5,6,7,19] somehow reduce concurrency to nondeterministic sequential behaviour by arbitrary interleaving of atomic actions, e.g. the process $\alpha.\text{stop} \parallel \beta.\text{stop}$ 'behaves' like $\alpha.\beta.\text{stop} \uplus \beta.\alpha.\text{stop}$, if $\alpha, \beta \neq \tau$.

In this paper we compare an interleaving semantics of full TCSP based on a transition system with a noninterleaving model based on labelled event structures [16,22,23,24].

In an earlier paper [11] have shown for finite TCSP processes without recursion and $\text{div}$ that the interleaving transition system based description and the respective event structure semantics are consistent. As recursion is a very powerful tool to build concurrent systems, it is an interesting question if this result carries over to full TCSP. We show here that this question has a positive answer. The result is in particular interesting, as it not only relates an interleaving specification with a noninterleaving but also relates at the same time an operational specification with a compositional one, that provides semantic operators for all syntactical constructs including $\text{fix}$.

2. The syntax of guarded TCSP

Let $\text{Comm}$ be the set of possible communications. A special action $\tau$, as in CCS, is introduced to describe internal actions which may not communicate. For notational convenience, we allow $\tau$ to occur syntactically in expressions denoting processes.

So let the set $\text{Act of actions}$ be defined as

$$\text{Act} := \text{Comm} \cup \{ \tau \}.$$ 

Let $\text{Idf}$ be a set of identifiers which will serve as variables for programs. The set $\text{TCSP}$ of $\text{TCSP terms}$ is defined by the following production system:

$$P := \text{stop} | \alpha.P | \text{div} | P \text{ or } Q | P \text{ or } Q | P \| A.Q | P \setminus \beta | z | \text{fix } z.P,$$

where $\alpha \in \text{Act}$, $\beta \in \text{Comm}$, $A \subseteq \text{Comm}$, $z \in \text{Idf}$.

2.1 Definition:

An occurrence of an identifier $z$ is called free in a term $P \in \text{TCSP}$ iff it does not occur within a subterm of the form $\text{fix } z.Q$. A TCSP term $P$ is said to be closed iff it does not contain identifiers which occur free in $P$.

An identifier $x$ is guarded in a term $P \in \text{TCSP}$ iff each free occurrence of $x$ in $P$ is in the scope of a prefixing operation $Q \leftarrow \alpha.Q$.

A term $P \in \text{TCSP}$ is called guarded iff in each subterm $\text{fix } z.Q$ of $P$ the identifier $z$ is guarded in $Q$.

Let $\text{GTCS}$ be the set of all guarded TCSP terms.
A GTCSP process is a closed, guarded TCSP term.

2.2 Definition:
Let $P, A_1, \ldots, A_n \in GTCSP$ and $z_1, \ldots, z_n \in Idf$ pairwise distinguished identifiers. The GTCSP term

$$P[A_1/z_1, \ldots, A_n/z_n]$$

or shortly $P[\hat{A}/\hat{z}]$

arises from $P$ by substituting each free occurrence of the identifiers $z_1, \ldots, z_n$ in $P$ simultaneously by the GTCSP terms $A_1, \ldots, A_n$.

3. Transition systems

3.1 Definition:
$A = (S, L, \rightarrow, q_0)$ is called a (labelled) transition system iff

(a) $S$ is a set of states.
(b) $L$ is a set of labels.
(c) $\rightarrow \subseteq S \times L \times S$, where we will write $p \xrightarrow{\alpha} q$ instead of $(p, \alpha, q) \in \rightarrow$.
(d) $q_0 \in S$, $q_0$ is called the initial state of $A$.

3.2 Definition:
Two equally labelled transition systems $A_i = (S_i, L, \rightarrow_i, q_i), i = 1, 2$, are bisimilar ($A_1 \approx A_2$) if there exists a bisimulation $R$ between $A_1$ and $A_2$, i.e. a relation $R \subseteq S_1 \times S_2$ with $(q_1, q_2) \in R$ and, for all $(p, q) \in R$:

1. Whenever $p \xrightarrow{\alpha} p'$ for some $p' \in S_1$ then there exists some $q' \in S_2$ with $(p', q') \in R$ and $q \xrightarrow{\alpha} q'$

and symmetrically

2. whenever $q \xrightarrow{\alpha} q'$ for some $q' \in S_2$ then there exists some $p' \in S_1$ with $(p', q') \in R$ and $p \xrightarrow{\alpha} p'$.

4. An interleaving transition system based description for guarded TCSP

4.1 Definition:

Let $\rightarrow$ be the binary relation on TCSP that is defined as follows:

(a) Prefixing
$$\alpha.P \xrightarrow{\alpha} P$$

(b) Internal nondeterminism
$$P \xrightarrow{\alpha} P', \quad P \text{ or } Q \xrightarrow{\alpha} Q$$

(c) External nondeterminism
External choice:
$$\frac{P \xrightarrow{\alpha} P'}{P \sqcup Q \xrightarrow{\alpha} P'} \quad \frac{Q \xrightarrow{\alpha} Q'}{P \sqcup Q \xrightarrow{\alpha} Q'},$$

where $\alpha \neq \tau$.  

2
Internal choice: \[ \frac{P \xrightarrow{\tau} P'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q}, \quad \frac{Q \xrightarrow{\tau} Q'}{P \parallel Q \xrightarrow{\tau} P \parallel Q'} \]

(d) Parallel composition

Synchronisation case: \[ \frac{P \xrightarrow{\alpha} P', Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q'}, \text{ where } \alpha \in A. \]

Independent execution (modelled by interleaving):

\[ \frac{P \xrightarrow{\alpha} P'}{P \parallel A Q \xrightarrow{\alpha} P' \parallel A Q'}, \quad \frac{Q \xrightarrow{\alpha} Q'}{P \parallel A Q \xrightarrow{\alpha} P \parallel A Q'}, \text{ where } \alpha \notin A. \]

(e) Hiding

\[ \frac{P \xrightarrow{\alpha} P'}{P \setminus \alpha \xrightarrow{\tau} P' \setminus \alpha}, \quad \frac{P \xrightarrow{\alpha} P'}{P \setminus \alpha \xrightarrow{\tau} P' \setminus \alpha}, \text{ where } \alpha \neq \beta. \]

(f) Recursion

\[ \frac{P[f z x P[z] \xrightarrow{\alpha} Q}{f z x P \xrightarrow{\alpha} Q}. \]

(g) Divergence

\[ \text{div} \xrightarrow{\tau} \text{div}. \]

An interleaving model of a closed GTCSP term \( P \) is the transition system

\[ A(P) = (\text{GTCSP}, \text{Act}, \xrightarrow{\tau}, P). \]

4.2 Definition:

For \( P, Q \in \text{GTCSP} \) and \( \omega \in \text{Comm}^* \), we define:

\[ P \xrightarrow{\omega} Q, \text{ iff there exists a sequence } \]

\[ P = P_1 \xrightarrow{\alpha_1} P_2 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} P_{n+1} = Q \]

where \( n \geq 0 \) and \( \omega \) results from \( \alpha_1 \ldots \alpha_n \in \text{Act}^* \) by skipping all occurrences of \( \tau \).

We call \( Q \) a derivative of \( P \).

Let \( P \) be a closed GTCSP term. Then, the transition system

\[ O(P) = (\text{GTCSP}, \text{Comm}^*, \Rightarrow, P) \]

gives an operational semantics for \( P \) that specifies only the observable behaviour of the process \( P \).

5. Labelled event structures

5.1 Definition:

\( \epsilon = (E, \leq, \#) \) is called a (labelled) event structure iff

(a) \( E \) is a set (of events),

(b) \( \leq \) is a partial order on \( E \),

(c) \( \# \) is an irreflexive, symmetric relation on \( E \), called conflict relation, with:

\[ \forall e_1, e_2, e_3 \in E : (e_1 \leq e_2 \text{ and } e_1 \neq e_3) \Rightarrow e_2 \neq e_3, \]

(d) \( l : E \rightarrow \text{Act} \), where \( \text{Act} \) is the alphabet of actions (labelling functions).
5.2 Definition:
Let \( \varepsilon = (E, \leq, \#, I) \) be an event structure, \( E' \subseteq E, \ e \in E \).

(a) \( \#(e) := \{ e' \in E : e' \# e \} \).

(b) \( \#(E') := \bigcup_{e \in E'} \#(e) \).

(c) \( \downarrow e := \{ e' \in E : e' \leq e \text{ and } e' \neq e \} \) is called the preset of \( e \).

5.3 Definition:
Let \( \varepsilon = (E, \leq, \#, I) \) be an event structure, \( e \in E \).

\[
深度(e) = \begin{cases} 
1 & \text{if } \downarrow e = \emptyset \\
\max \{\text{深度}(e') : e' \in \downarrow e\} + 1 & \text{if } \downarrow e \text{ is finite} \\
\infty & \text{otherwise} 
\end{cases}
\]

5.4 Definition:
An event structure \( \varepsilon = (E, \leq, \#, I) \) is called (finitely) approximable iff

(a) for each \( e \in E \), depth\( (e) \) is finite and

(b) for each \( n \in \mathbb{N} \), \( \{ e \in E : \text{depth}(e) = n \} \) is finite.

\( Ev \) denotes the set of all finitely approximable event structures where we abstract from the names of the events, i.e., we will not distinguish isomorphic event structures. Two event structures \( \varepsilon_i = (E_i, \leq_i, \#, i_i), i = 1, 2 \) are isomorphic if there exists a bijective mapping \( f : E_1 \to E_2 \) so that

1. \( e_1 \leq_1 e_2 \iff f(e_1) \leq_2 f(e_2) \ \forall e_1, e_2 \in E_1 \)
2. \( e_1 \#_1 e_2 \iff f(e_1) \#_2 f(e_2) \ \forall e_1, e_2 \in E_1 \) and
3. \( l_i(f(e)) = l_i(e) \ \forall e \in E_i \).

Event structures can be depicted graphically by representing events as boxes (inscribed with the event label) and connecting them with their direct predecessors and successors. A conflict between two events is a direct conflict if no predecessors of the events are in conflict. Direct conflicts are depicted graphically by a broken line.

Example:
The event structure \( \varepsilon = (E, \leq, \#, I) \) with
\( E = \{e_1, e_2, e_3\} \), \( e_1 \leq e_2 \), \( e_1 \# e_3 \), \( e_2 \# e_3 \) and
\( l(e_1) = \alpha, l(e_2) = \beta, l(e_3) = \gamma \) is shown as

\[
\begin{array}{c}
\alpha \\
\downarrow \\
\beta \\
\downarrow \\
\gamma
\end{array}
\]
6. Composition operations for event structures

The event structure semantics for GTCSP to be defined is compositional, which means that composition operators corresponding to the syntactical operators prefix, or , $\Box$, $\|$, $\setminus$, and $\text{fix}$ have to be defined. This has been done in [11], we will here explain examples only and refer for the precise definitions to the appendix.

6.1 Example: Prefixing

$\alpha . \varepsilon$ describes a process that first performs $\alpha$ and then behaves like $\varepsilon$. If $\varepsilon$ is

$\alpha \rightarrow \beta$

then $\alpha . \varepsilon$ is

$\alpha \rightarrow \beta$

\[\text{\bigg\rightharpoonup}\]

$\alpha \rightarrow \beta$

6.2 Example: $\Box$ - choice

Let $\varepsilon_1$ be $\tau \rightarrow \alpha$ and $\varepsilon_2$ be $\beta$. Then $\varepsilon_1 \Box \varepsilon_2$ is given by

$\tau \rightarrow \alpha$

\[\text{\bigg\rightharpoonup}\]

$\beta$

which describes that $\varepsilon_1$ may perform its $\tau$-actions independently and that a decision has to take place as soon as communications are involved.

6.3 Example: $\Box$ - choice

Let $\varepsilon_1$ be $\alpha \rightarrow \beta$ and $\varepsilon_2$ be $\tau \rightarrow \delta$, then $\varepsilon_1 \Box \varepsilon_2$ is

$\alpha \rightarrow \beta$

\[\text{\bigg\rightharpoonup}\]

$\tau \rightarrow \delta$

describing external choice.
6.4 Example: or-choice
The or-choice reflects internal nondeterminism.

Let $\varepsilon_1$ be $\tau \rightarrow \alpha$ and $\varepsilon_2$ be $\beta$.

Then $\varepsilon_1 \lor \varepsilon_2$ is given by

\[
\begin{array}{c}
\tau \\
\downarrow \\
\tau \\
\downarrow \\
\beta
\end{array}
\]

The internal character of the or-choice is modelled by prefixing the respective event structures with internal actions and by imposing a conflict between these internal actions.

6.5 Example: Parallel composition $\parallel A$

Let $\varepsilon_1$ be $\beta \rightarrow \alpha \rightarrow \tau$ and $\varepsilon_2$ be $\delta \rightarrow \alpha \rightarrow \rho$, then $\varepsilon_1 \parallel (\alpha) \varepsilon_2$ is given by

\[
\begin{array}{c}
\beta \\
\delta
\end{array}
\leftarrow
\begin{array}{c}
\alpha \\
\rho
\end{array}
\rightarrow
\begin{array}{c}
\tau
\end{array}
\]

6.6 Example: Hiding

Let $\varepsilon$ be

\[
\begin{array}{c}
\alpha \\
\beta
\end{array}
\]

then $\varepsilon \setminus \beta$ is

\[
\begin{array}{c}
\alpha \\
\tau
\end{array}
\]

i.e. hiding transforms actions labelled by $\beta$ into actions labelled by $\tau$. 
7. The metric space of finite approximable event structures

In this section we will define a metric \( d \) on finite approximable event structures. [15] have shown that \((\text{Ev}, d)\) is a complete ultrametric space. Thus, every Banach-contractive mapping \( \Phi : \text{Ev} \rightarrow \text{Ev} \) has a unique fixpoint in \( \text{Ev} \).

7.1 Definition:

Let \( \varepsilon, \varepsilon' \in \text{Ev}, n \in N, \varepsilon = (E, \leq, \#) \).

(a) The truncation of \( \varepsilon \) (of the depth \( n \)) is defined as follows:
\[
\varepsilon^n := (E^n, \leq |E^n, \#, |E^n, l|E^n)
\]
where \( E^n := \{ e \in E : \text{depth}(e) \leq n \} \).

(b) The distance between the event structures \( \varepsilon, \varepsilon' \) is defined by
\[
d(\varepsilon, \varepsilon') = 0 \iff \varepsilon = \varepsilon' \\
d(\varepsilon, \varepsilon') = \frac{1}{2^n} \iff \varepsilon \neq \varepsilon' \text{ and } n = \max\{ i : e^i = e'^i \}.
\]

We recall that we deal with isomorphism class of event structures, i.e. we abstract of the names of the events \( e \in E \). It is clear that the distance \( d(\varepsilon, \varepsilon') \) is independent of chosen representatives.

7.2 Definition:

Let \( \text{Env} := \{ \sigma : \sigma : \text{Idf} \rightarrow \text{Ev} \} \) the set of environments. These are mappings which assign a meaning to free identifiers of a term.

For \( \varepsilon_1, \ldots, \varepsilon_n \in \text{Ev} \), we define \( \sigma[\varepsilon_1/x_1, \ldots, \varepsilon_n/x_n] : \text{Idf} \rightarrow \text{Ev} \) by
\[
z_i \mapsto \varepsilon_i, \quad i = 1, \ldots, n, \\
y \mapsto \sigma(y) \quad \text{if } y \notin \{x_1, \ldots, x_n\}.
\]

Let \( \Phi : \text{GTCSP} \times \text{Env} \times \text{Id} \rightarrow (\text{Ev} \rightarrow \text{Ev}) \) be given by
\[
\Phi(P, \sigma, x)(\varepsilon) := M[P[\sigma/x]],
\]
where \( M \) is the meaning function
\[
M : \text{GTCSP} \times \text{Env} \rightarrow \text{Ev}
\]
given by:

Let \( \sigma \in \text{Env}, \alpha \in \text{Act}, \beta \in \text{Comm}, A \subseteq \text{Comm}, P, P_1, P_2 \in \text{GTCSP} \).

(a) \( M[x] \sigma := \sigma(x) \) where \( x \in \text{Idf} \).
(b) \( M[\alpha.P] \sigma := \alpha.M[P] \sigma \).
(c) \( M[P \setminus \beta] \sigma := M[P] \sigma \setminus \beta \).
(d) \( M[P_1 \sqcup P_2] \sigma := M[P_1] \sigma \sqcup M[P_2] \sigma \).
(e) \( M[P_1 \circ P_2] \sigma := M[P_1] \sigma \circ M[P_2] \sigma \).
(f) \( M[P_2 \parallel_A P_2] \sigma := M[P_1] \sigma \parallel_A M[P_2] \sigma \).
(g) \( M[\text{fix } x.P] \sigma := \text{fix } \Phi(P, \sigma, x) \)

where \( \text{fix } \Phi(P, \sigma, x) \) denotes the unique fixpoint of the Banach - contractive mapping \( \Phi(P, \sigma, x) \). See [11], where it has been shown that \( \Phi(P, \sigma, x) \) is Banach - contractive.
Lemma 1:
Let $z \in Idf$ be guarded in $P \in GTCS$. 

(a) $\sigma_1, \sigma_2 \in Env, \sigma_1(y) = \sigma_2(y) \forall y \in Idf \setminus \{z\} \Rightarrow fix\Phi(P, \sigma_1, z) = fix\Phi(P, \sigma_2, z)$. 

(b) If $P$ is closed then $fix\Phi(P, \sigma, z)$ is independent of the environment $\sigma$. 

(c) Let $x_1, \ldots, x_n$ be pairwise different identifiers, $A_1, \ldots, A_n \in GTCS$, then:

$$M[P[A_1/x_1, \ldots, A_n/x_n], \sigma] = M[P, \sigma] = M[A_1/\sigma, \ldots, M[A_n/\sigma, x_n].$$

Proof:
(a) follows immediately from the definition of $\Phi$. 
(b) is clear. 
(c) By structural induction on the syntax of $P$.

Lemma 2:
Let $P, B, A_1, \ldots, A_n \in GTCS$ and let $x_1, \ldots, x_n, y \in Idf$ be pairwise different identifiers, so that $y$ does not occur free in $A_1, \ldots, A_n$. Then,

$$P[A_1/x_1, \ldots, A_n/x_n, B[y/x\ldots], y = P[B[y/x\ldots], y].$$

Proof:
By induction on the syntax of $P$.

Lemma 3:
Let $P \in GTCS$. Then, for all $x_1, \ldots, x_n \in Idf$ pairwise different identifiers, which are guarded in $P$, and for all $A_1, \ldots, A_n \in GTCS$:

If $P[A_1/x_1, \ldots, A_n/x_n] \alpha Q$, then there exists $P' \in GTCS$ with

1. $P \xrightarrow{\sigma} P'$ and 
2. $P'[A_1/x_1, \ldots, A_n/x_n] = Q$.

Proof:
By induction on the syntax of $P$.

Remark:
Let $P, Q, A_1, \ldots, A_n \in GTCS$ and $x_1, \ldots, x_n \in Idf$ be pairwise different identifiers which are guarded in $P$ so that $P[A/x] \alpha Q$.

Then, there exists $P' \in GTCS$ with

1. $P \xrightarrow{\sigma} P'$ and 

It is easy to see that for all terms $B_1, \ldots, B_n \in GTCS$:

$$P[B/x] \alpha P'[B/x].$$

Remark:
If $A \in GTCS$ is closed then

$$M[A] = M[A] \quad \forall \sigma_1, \sigma_2 \in Env.$$
So, we can define

\[ M[A] := M[A] \sigma \quad \text{where} \quad \sigma \in \text{Env}. \]

7.3 Definition:

For \( \omega = \alpha_1 \ldots \alpha_n \in \text{Act}^* \), we define \( \hat{\omega} \) to be the word in \( \text{Comm}^* \) which arises from \( \omega \) by eliminating all actions labelled by \( \tau \).

I.e., \( \hat{\omega} = \alpha_{i_1} \ldots \alpha_{i_k} \) where \( 1 \leq i_1 < \ldots < i_k \leq n \) are the indices \( i \in \{1, \ldots, n\} \) with \( \alpha_i \in \text{Comm} \).

7.4 Definition:

(a) Let \( \mu \in \text{Act}, \varepsilon, \varepsilon' \in \text{Ev}, \varepsilon = (E, \leq, \#, I) \). The transition relation 
\[ \varepsilon \leftrightarrow \varepsilon' \quad \text{iff} \quad \text{there exists some event} \ e \in E \quad \text{with} \quad \text{depth}(e) = 1, \ l(e) = \mu \quad \text{and} \quad \varepsilon' = (E', \leq, \#, I, [\varepsilon(e)]. \]

(b) When we abstract from \( \tau \)-events we get the transition relation 
\[ \Rightarrow \subset \text{Ev} \times \text{Comm}^* \times \text{Ev} : \]
\[ \varepsilon \leftrightarrow \varepsilon' \quad \text{iff} \quad \text{there exists a sequence} \]
\[ \varepsilon = \varepsilon_1 \mu_1 \varepsilon_2 \mu_2 \ldots \mu_{n+1} = \varepsilon' \]
\[ \text{where} \ n \geq 0, \mu_1, \ldots, \mu_n \in \text{Act} \quad \text{and} \ \omega \in \text{Comm}^* \ \text{results from} \ \mu_1 \mu_2 \ldots \mu_n \ \text{by removing all} \ \mu_i = \tau. \]

(c) The (observable) interleaving semantics of \( \varepsilon \in \text{Ev} \) is defined as the transition system

\[ O(\varepsilon) = (\text{Ev}, \text{Comm}^*, \Rightarrow, \varepsilon). \]

7.5 Definition:

The event structures \( \varepsilon_1, \varepsilon_2 \) are called \( \tau \)-equivalent, written \( \varepsilon_1 \approx_\tau \varepsilon_2 \), iff there exists event structures \( \varepsilon, \text{Int}_1, \text{Int}_2 \), where all events in \( \text{Int}_1, \text{Int}_2 \) are labelled by \( \tau \), with \( \varepsilon_i = \varepsilon \parallel \text{Int}_i, \ i = 1, 2 \).

It is easy to see that \( \tau \)-equivalence is an equivalence relation on \( \text{Ev} \). [11] have shown that if \( \varepsilon_1 \approx_\tau \varepsilon_2 \) and \( \varepsilon_1 \leftrightarrow \varepsilon'_1 \) then there exists \( \varepsilon'_2 \in \text{Ev} \) with \( \varepsilon'_1 \approx_\tau \varepsilon'_2 \) and \( \varepsilon_2 \leftrightarrow \varepsilon'_2 \).

Lemma 4:

Let \( P \in \text{GTCSP}, \alpha \in \text{Act}, \sigma \in \text{Env}. \)

(a) If \( P \approx P' \) then \( M[P] \sigma \approx M[P'] \sigma \).

(b) If \( x_1, \ldots, x_n \) be the pairwise different identifiers that occur free in \( P \) and if \( \sigma(x_i) = M[A_i] \) where \( A_i \) is a closed GTCSP term, \( i = 1, \ldots, n \), then, for all event structures \( \varepsilon' \in \text{Ev} \) with \( M[P] \sigma \approx \varepsilon' \), there exists a term \( P' \in \text{GTCSP} \) with

1. \( P[A_1/x_1, \ldots, A_n/x_n] \approx P' \)

2. \( M[P'] \sigma \approx \varepsilon' \).

Proof: We will prove the statements by induction on the structure of \( P \).

(a) We assume that \( P \approx P' \). Basis of induction:

1. \( P = \text{stop} \) has no derivatives.
2. \( P = \text{div} \), then \( \alpha = \tau \) and \( P' = \text{div} \), \( M[P] \sigma = M[P'] \sigma \).
3. \( P = z \in \text{Idf} \), then \( P \) has no derivatives.
Induction step:

The most interesting operator is the \textit{fix} - operator.
\[ P = \text{fix} \, z. \, Q, \quad \text{then} \quad Q[\text{fix} \, z. \, Q/\!z] \xrightarrow{\alpha} P'. \]

By Lemma 3, there exists \( Q' \in \text{GTCS}P \) with \( Q \xrightarrow{\alpha} Q' \) and \( Q'[\text{fix} \, z. \, Q/\!z] = P' \).

By induction hypothesis:

\[ M[Q] \sigma[ M[P] \sigma/\!x ] \xrightarrow{\alpha} M[Q'] \sigma[ M[P] \sigma/\!x ] \]

On the other side, we have:

\[ M[P] \sigma = \text{fix} \, \Phi(Q, \sigma, z) = M[Q] \sigma[ M[P] \sigma/\!x ] \]

and

\[ M[P'] \sigma = M[Q'[\text{fix} \, z. \, Q/\!z]] \sigma = M[Q'[P/\!z]] \sigma = M[Q'] \sigma[ M[P] \sigma/\!x ] \]

(Lemma 1c). Then, \( M[P] \sigma \xrightarrow{\alpha} M[P'] \sigma \).

(b) Induction step:

Again, we only consider the \textit{fix} - operator: \( P = \text{fix} \, z. \, Q \).

The identifiers occurring free in \( Q \) are \( z_1, \ldots, z_n \) and \( z \). We get:

\[ M[P] \sigma = \text{fix} \, \Phi(Q, \sigma, z) = M[Q] \sigma[ M[P] \sigma/\!x ] \]

and

\[ M[P] \sigma = M[ P[A_1/z_1, \ldots, A_n/z_n] ] \]

(by Lemma 1c).

Hence

\[ \sigma[ M[P] \sigma/\!x ](x) = M[P] \sigma = M[ P[A/\!z] ] \]

and

\[ \sigma[ M[P] \sigma/\!x ](z_i) = \sigma(z_i) = M[A_i], i = 1, \ldots, n. \]

Since \( M[P] \sigma \xrightarrow{\alpha} \epsilon' \), we get by Lemma 1c and Lemma 2:

\[ M[Q] \sigma[ M[P] \sigma/\!x ] \xrightarrow{\alpha} \epsilon'. \]

By induction hypothesis, there exists \( P' \in \text{GTCS}P \) with

\[ Q[ A_1/z_1, \ldots, A_n/z_n, P[A/\!z]/z ] \xrightarrow{\alpha} P' \quad \text{and} \quad M[P] \sigma \xrightarrow{\alpha} \epsilon'. \]

Since the terms \( A_1, \ldots, A_n \) are closed, we get:

\[ Q[A/\!z][ P[A/\!z]/z ] = Q[ A_1/z_1, \ldots, A_n/z_n, P[A/\!z]/z ]. \]

\[ \Rightarrow \quad Q[A/\!z][ \text{fix} \, z. \, Q[A/\!z]/z ] \xrightarrow{\alpha} P' \]

\[ \Rightarrow \quad P[A/\!z] = \text{fix} \, z. \, Q[A/\!z] \xrightarrow{\alpha} P'. \]
Corollary:
Let \( R := \{ (P, e) : P \in \text{GTCSP}, P \text{ closed}, e \in Ev, e \sim, M[P] \} \).
Then \( R \) is a bisimulation.

Proof:
Let \((P, e) \in R\).
1. When \( P \rightarrow P' \), so we have by Lemma 4a:
\[ M[P] \triangleleft M[P'] \]
Since \( e \sim, M[P] \), there exists \( e' \in Ev \) with
\[ e \triangleleft e' \text{ and } e' \sim, M[P'] \]
Then \((P', e') \in R\).
2. When \( e \rightarrow e' \), then there exists \( e'' \in Ev \) with
\[ M[P] \triangleleft e'' \text{ and } e' \sim, e'' \]
By Lemma 4b, it is easy to show that there exists \( P' \in \text{GTCSP}, P' \text{ closed}, \) with
\[ P \triangleleft P', M[P'] \sim, e'' \]
Then, \( M[P'] \sim, e' \) and \((P', e') \in R\).

Theorem:
For every closed \( P \in \text{GTCSP} \) (i.e. every guarded process), the transition systems \( O(P) \) and \( O(M[P]) \) are bisimilar.

8. Conclusion

We have shown that an interleaving specification of a GTCSP process \( P \) and a noninterleaving meaning of \( P \) are 'bisimular'. One difficulty in establishing such a result, in particular when including recursion via the \textit{fix}-operator, is, that a compositional semantics that provides semantic operators for the syntactical constructs, is compared with an operational semantics using a transition system. Hence, in order to establish a relation between the two meanings of a process \( P \) we may not simply perform an induction on the structure of \( P \). In particular, in the case of recursion, we have no operator that determines the 'meaning' of \( \text{fix } z.Q \) from the 'meaning' of \( Q \) in the transition system case.

Our proof works by obtaining information on the behaviour of a process \( P \) from the knowledge of the behaviour of \( P[A/z] \), see lemma 3 and lemma 4.

The obtained theorem may be interpreted as a consistency result.
Consistency problems concerning noninterleaving and interleaving models are discussed in [9,18,20]. These investigations differ from the present work in particular in the noninterleaving model (petri nets, prime event structures) and/or in the language studied and in the proof method.
Appendix

This section gives the operations for finite approximable event structures modelling the operations of GT CSP as defined in [11].

A.1 Definition: Let \( \text{stop} \in \mathcal{E} \) be defined as
\[
\text{stop} := (0,0,0,0).
\]

A.2 Definition: Let \( \varepsilon = (E, \leq, \#, l) \in \mathcal{E} \), \( \alpha \in \mathcal{A} \), \( e_0 \notin E \). Then, the event structure \( \alpha \cdot \varepsilon \) will describe a process which first performs \( \alpha \) and then behaves like \( \varepsilon \).
\[
\alpha \cdot \varepsilon = (E', \leq', \#', l')
\]

where
1. \( E' = E \cup \{e_0\} \),
2. \( e_1 \leq' e_2 \iff e_1 = e_0 \) or \( (e_1, e_2) \in E \land e_1 \leq e_2 \),
3. \( e_1 \#' e_2 \iff e_1, e_2 \in E \land e_1 \# e_2 \),
4. \( l' : E' \rightarrow \mathcal{A} \) is defined by \( l'(e) = l(e) \), if \( e \in E \), and \( l'(e_0) = \alpha \).

A.3 Definition: For \( \varepsilon = (E, \leq, \#, l) \in \mathcal{E} \), we define the set of initial internal events by
\[
\mathcal{I}n(\varepsilon) := \{e \in E : \forall e' \in E, e' \leq e : l(e') = \tau\}
\]

A.4 Definition: Let \( \varepsilon_i = (E_i, \leq_i, \#_i, l_i) \in \mathcal{E} \), \( i = 1, 2 \), w.l.o.g. \( E_1 \cap E_2 = \emptyset \). The conditional composition of \( \varepsilon_1 \) and \( \varepsilon_2 \) is defined by
\[
\varepsilon_1 \sqcap \varepsilon_2 := (E, \leq, \#, l)
\]

where
1. \( E = E_1 \cup E_2 \),
2. \( \leq = \leq_1 \cup \leq_2 \),
3. \( e_1 \# e_2 \iff (e_1, e_2) \in E_1 \land e_1 \#_1 e_2 \) or \( (e_1, e_2) \in E_2 \land e_1 \#_2 e_2 \)
   or \( (e_1 \in \mathcal{I}n(e_1) \land e_2 \in \mathcal{I}n(e_2)) \) or \( (e_1 \in \mathcal{I}n(e_2) \land e_2 \in \mathcal{I}n(e_1)) \),
4. \( l : E \rightarrow \mathcal{A} \), \( l(e) = l_i(e) \) if \( e \in E_i \), \( i = 1, 2 \).

\( \varepsilon_1 \sqcap \varepsilon_2 \) describes the process which behaves like one of the event structures \( \varepsilon_1 \) or \( \varepsilon_2 \) where the decision which alternative is left open as long as only internal actions are being performed.

A.5 Definition: Let \( \varepsilon_i = (E_i, \leq_i, \#_i, l_i) \in \mathcal{E} \), \( i = 1, 2 \), w.l.o.g. \( E_1 \cap E_2 = \emptyset \). The nondeterministic combination of \( \varepsilon_1 \) and \( \varepsilon_2 \) is defined by
\[
\varepsilon_1 \lor \varepsilon_2 := (E, \leq, \#, l)
\]

where
1. \( E = E_1 \cup E_2 \cup \{ f_1, f_2 \} \), \( f_1, f_2 \notin E_1 \cup E_2 \)

2. \( e_1 \leq e_2 \iff (e_1, e_2) \in E_i \& e_1 \leq_i e_2, i = 1 \text{ or } i = 2 \) or
   \((e_i = f_i \& e_2 \in E_i, \ i = 1 \text{ or } i = 2) \) or \( e_1 = e_2 \)

3. \# is the symmetric closure of \( \#_1 \cup \#_2 \cup ((E_1 \cup \{ f_1 \}) \times (E_2 \cup \{ f_2 \})) \)

4. \( I : E \rightarrow \text{Act} \), \( I(e) = I_1(e) \) if \( e \in E_1 \) and \( I(f_i) = \tau, i = 1, 2 \).

The nondeterministic combination \( \varepsilon_1 \) or \( \varepsilon_2 \) behaves like \( \varepsilon_1 \) or like \( \varepsilon_2 \) where an internal decision choose the alternative.

A.6 Definition:
Let \( \varepsilon_i = (E_i, \leq_i, \#_i, I_i) \in Ev, i = 1, 2 \) and \( A \subseteq \text{Comm} \).

1. The syntactical communication of \( \varepsilon_1 \) and \( \varepsilon_2 \) on \( A \) is defined by
   \[ \text{Comm}_A(\varepsilon_1, \varepsilon_2) := \{ (e, \ast) : e \in E_1 \& I_1(e) \notin A \]
   \[ \text{or } e \in E_2 \& I_2(e) \notin A \} \]
   \[ \cup \{ (e_1, e_2) \in E_1 \times E_2 : I_1(e_1) = I_2(e_2) \in A \} \].

   There \( \ast \) is an auxiliary symbol, \( \ast \notin E_1 \cup E_2 \).

2. Two communications \((e_1, e_2), (e_1', e_2') \in \text{Comm}_A(\varepsilon_1, \varepsilon_2)\) are in conflict iff they contain conflicting events, i.e. \( e_1 \#_1 e_1' \) or \( e_2 \#_2 e_2' \), or one event communicates with two distinct events, i.e. \( e_1 = e_1' \land e_2 \neq e_2' \) or \( e_2 = e_2' \land e_1 \neq e_1' \).

3. A subset \( C \) of \( \text{Comm}_A(\varepsilon_1, \varepsilon_2) \) is conflict-free iff no two communications in \( C \) are in conflict.

4. Let \( C \subseteq \text{Comm}_A(\varepsilon_1, \varepsilon_2) \) be conflict-free, \((e_1, e_2), (f_1, f_2) \in C\).

   (a) The relation \( \prec \) is defined by
   \[ (e_1, e_2) \prec (f_1, f_2) \iff ((e_1 \leq f_1) \land \neg (e_2 > f_2)) \text{ or } ((e_2 \leq f_2) \land \neg (e_1 > f_1)). \]

   We say \( (e_1, e_2) \) precedes \( (f_1, f_2) \) if \( (e_1, e_2) \prec (f_1, f_2) \).

   (b) \( C \) is called complete iff
   \[ \forall (e_1, e_2) \in C, \forall f_1 \in E_1 \text{ with } f_1 \leq e_1 \text{ there exists } (e_2, f_2) \in C \text{ with } \]
   \[ (f_1, f_2) \prec (e_1, e_2) \]

   and symmetrically
   \[ \forall (e_1, e_2) \in C, \forall f_2 \in E_2 \text{ with } f_2 \leq e_2 \text{ there exists } (f_1, f_2) \in C \text{ with } \]
   \[ (f_1, f_2) \prec (e_1, e_2) \]

   (c) \( C \) is called cycle-free iff the transitive closure of \( \prec \) is antisymmetric.

5. The parallel composition of \( \varepsilon_1 \) and \( \varepsilon_2 \) with communication on \( A \subseteq \text{Comm} \) is given by
   \[ \varepsilon_1 \parallel_A \varepsilon_2 := (E, \leq, \#, I) \]
   where
   (a) \( E = \{ C(e_1, e_2) : C(e_1, e_2) \subseteq \text{Comm}_A(\varepsilon_1, \varepsilon_2) \text{ is conflict-free, cycle-free, complete } \}
   \[ \text{and } (e_1, e_2) \in C(e_1, e_2) \text{ is the only maximal element (with respect to } \prec) \} \}

   (b) \( \leq = \subseteq \)

   (c) \# = \{ (C_1, C_2) \in E \times E : \exists (e_1, e_2) \in C_1, (f_1, f_2) \in C_2 \text{ with } \]
   \[ (e_1, e_2), (f_1, f_2) \text{ in conflict } \} \]

13
(d) \( l : E \rightarrow \text{Act} \), \( l(C(e_1, e_2)) = \text{label}(e_1, e_2) \)
where \( \text{label}(e_1, e_2) = l_1(e_1) \) if \( e_1 \in E_1 \) and \( \text{label}(e_1, e_2) = l_2(e_2) \) if \( e_2 \in E_2 \).

The parallel composition \( e_1 \parallel e_2 \) describes the independent execution of \( e_1 \) and \( e_2 \) where the actions of \( A \) may only be executed as joint actions by both processes together. In particular, \( \parallel_\text{if} \) stand for fully independent execution (without synchronisation), and on the other extrem, \( \parallel_\text{comm} \) only allows actions which are performed in common.

A.7 Definition:
Let \( e = (E, \leq, \#, l) \in E^v, \beta \in \text{Comm} \).

\( e \setminus \beta := (E, \leq, \#, l') \)
where \( l' : E \rightarrow \text{Act}, l'(e) = l(e) \) if \( l(e) \neq \beta \) and \( l'(e) = \tau \) otherwise.

The hiding operator transforms the actions labelled by \( \beta \) into internal actions, i.e. \( \tau \)-events.
References

1. J.W. de Bakker, J.L. Zucker:
   Processes and the Denotational Semantics of Concurrency,
   Information and Control, Vol.54, No 1/2, pp 70-120, 1982.

2. J.A. Bergstra, J.W. Klop:
   Process Algebra for Synchronous Communication,

3. G. Boudol, I. Castellani:
   On the Semantics of Concurrency: Partial Orders and Transition Systems,
   Proc. TAPSOFT 87, Vol 1, Lecture Notes in Computer Science 249, Springer - Verlag,

4. G. Boudol, I. Castellani:
   Permutation of transitions: An event structure semantics for CCS and SCCS,
   Proc. School/Workshop on Linear Time, Branching Time and Partial Order in Logics and
   Models for Concurrency,

5. S.D. Brookes:
   A Model for Communicating Sequential Processes,

6. S.D. Brookes, C.A.R. Hoare, A.W. Roscoe:
   A Theory of Communicating Sequential Processes,

7. S.D. Brookes, A.W. Roscoe:
   An improved Failure Model for Communicating Processes,

8. P. Degano, R. De Nicola, U. Montanari:
   A Distributed Operational Semantics for CCS Based on Condition/Event Systems,

9. P. Degano, R. De Nicola, U. Montanari:
   On the Consistency of 'Truly Concurrent' Operational and Denotational Semantics,

10. U. Goltz:
    On Representing CCS Programs as Finite Petri Nets,

11. U. Goltz, R. Loogman:
    Modelling Nondeterministic Concurrent Processes with Event Structures,
    to appear in Fundamentae Informaticae;
    see also: Schriften zur Informatik und angewandten Mathematik,

12. U. Goltz, A. Mycroft:
    On the Relationship of CCS and Petri Nets,

13. C.A.R. Hoare:
    Communication Sequential Processes,

14. R. Milner:
    A Calculus of Communication Systems,
    Lecture Notes in Computer Science 92, Springer - Verlag, 1980.
15. R. Milner:
Lectures on a Calculus of Communicating Systems,

16. M. Nielsen, G. Plotkin, G. Winskel:
Petri - Nets, Event Structures and Domains,

17. E.R. Olderog:
TCSP : Theory of Communicating Sequential Processes,
Advances in Petri - Nets 1986,

18. E.R. Olderog:
Operational Petri - Net Semantics for CCSP,
Advances in Petri - Nets 1987,

19. G.D. Plotkin:
An Operational Semantics for CSP,

20. W. Reisig:
Partial Order Semantics versus Interleaving Semantics for CSP - like languages and its Impact
on Fairness,

21. D. Taubner, W. Vogler:
The Step Failure Semantics,

22. G. Winskel:
Events in Computation,

23. G. Winskel:
Event Structure Semantics for CCS and Related Languages,

24. G. Winskel:
Event Structures,
Petri - Nets : Applications and Relationships to Other Models of Concurrency,