Modelling and Verification of Communicating Processes in the Event of Interface Difference

Jonathan Burton¹, Maciej Koutny¹, and Giuseppe Pappalardo²

¹ Department of Computing Science, University of Newcastle, Newcastle upon Tyne NE1 7RU, U.K.
² Dipartimento di Matematica e Informatica, Università degli Studi di Catania, Viale A.Doria 6, 95125 Catania, Italy

Abstract. We extend our investigation of the notion that a system built of communicating processes is an acceptable implementation of another base or target system, in the case that respective specification and implementation processes have different interfaces and we combine into a single scheme implementation relations previously presented. We also relax significantly the restrictions placed upon target processes. Using this implementation relation scheme, two basic kinds of results are obtained: realizability and compositionality. The former ensures that implementations may be put to good use; in practice, this means that plugging an implementation into an appropriate environment should yield a conventional implementation of the target. The latter requires that a target composed of several connected systems may be implemented by connecting their respective implementations.

We then give graph-based representations of the formal structures which we use, develop graph theoretic statements of the implementation relations and finally present algorithms for their automatic verification.

Keywords: Theory of parallel and distributed computation, behaviour abstraction, communicating sequential processes, compositionality, verification.

1 Introduction

The approach presented in [8–12] aimed at formalising the notion that a system is an acceptable implementation of another base or target system, in the case that the two systems (respective specification and implementation processes) have different interfaces. We developed and expounded the notion of extraction pattern to deal with the fact of this interface difference.

Extraction patterns interpret the behaviour of a system at the level of communication traces by relating behaviour on a set of channels in the implementation to behaviour on a channel or channels in the specification. The set of extraction patterns defined for all channels in an implementation system may act as a formal parameter in a generic implementation relation scheme. The scheme given here unifies into a single presentation the notions of weak and strong implementability defined in [10, 11], with a set of relations of increasing discriminative power. We now also relax the behavioural restrictions on the target processes allowed, thus significantly widening the applicability of our treatment.

The implementation relation scheme given satisfies two light but very natural and useful requirements. The first, accessibility or realizability, ensures that the abstraction built into the implementation relation may be put to good use; in practice, this means that plugging an implementation into an appropriate environment¹ should yield a con-

¹ In our treatment of NMR [9], this environment is made up of disturbers, feeding faulty but sufficiently redundant input, and extractors, which interpret the implementation's output.
ventional implementation of the target. Distributivity or compositionality, the other constraint on the implementation relation, requires it to distribute over system composition; thus, a target composed of two connected systems may be implemented by connecting two of their respective implementations.

As before, both implementation and specification systems are represented using the FD (failure-divergence) model of CSP [3, 6]. To facilitate proceeding to the development of algorithms for automatic verification of the implementation relations, we give representations for extraction patterns and CSP processes, amenable to computer implementation and also formulate the implementation relations in a manner directly related to our chosen means of representation.

The implementation relations as stated imply the use of state space methods for (automatic) verification. This suggests the use of graph-based structures to represent both extraction patterns and CSP processes. It is then a simple enough matter to convert our implementation relations — presented in the denotational semantics of CSP — into graph-theoretic terms, using the equivalences proven between our CSP and graphical presentations of extraction patterns and processes respectively.

That the implementation relations have the property of compositionality has an important consequence when we approach automatic verification. It allows us to verify each component of the implementation system explicitly in terms of its specification component and we avoid one of the great sources of the state explosion problem in concurrency, namely the generation of a state space which is a subset (not necessarily proper) of the product of all the state spaces of a set of component processes composed in parallel.

As a result, we avoid in most cases the need to explicitly generate a state space: the state spaces with which we need to deal are mostly implicit in the structure of the labelled transition systems with which we represent CSP processes. In those cases where we have to explicitly generate a state space, namely when testing for trace inclusion, only two processes are involved and the testing is done on-the-fly anyway. This leads to generally favourable complexity characteristics. Also of importance in this respect is the fact that we need only deal with maximal (in a sense to be defined) failures when verifying the implementation relations.

We present here implementation relations and verification algorithms for the purpose of verifying that a system is a valid implementation of a target system, in the event that the two systems have different interfaces. It may be the case that the target system must be expressed at too low a level of abstraction to function as a conventional specification. In this event, it may be necessary to verify that the target conforms to another (higher level) specification, also expressed in the FD model of CSP. For this purpose, a tool such as FDR [15] may be used, since any interface difference between the target and the specification may be dealt with using the conventional means of renaming and hiding. Though this (higher level) process of verification will not be able to take advantage of the property of compositionality as described in our treatment, the (highest level) specification process will generally be smaller than the target, and the target process will generally be smaller than the implementation process. It is thus clear that our verification methods with their attendant gains in efficiency will be used at those points in the verification process where the systems to be dealt with are of the greatest size.

The paper is organised as follows. In the next section we introduce some basic notions used throughout the paper. In section 3 we first introduce extraction patterns — a central notion to defining the interface of an implementation. This is followed by the definition of successively stronger implementation relations, and the proofs of their suitability. Section 4 deals with computer representations of CSP processes and extraction patterns, in both cases employing a variant of a labelled transition system. In section 5 we make the
necessary technical steps to relate the labelled transition systems representing processes and the extract patterns, and in sections 6 and 7 we show how the defining conditions for implementation relations can be verified algorithmically.

2 Preliminaries

In this section, we first recall those parts of the CSP theory which are needed throughout the paper. We then introduce a class of base processes.

2.1 Actions and traces

Communicating Sequential Processes (CSP) [2, 3, 6, 15] is a formal model for the description of concurrent computing systems. A CSP process can be regarded as a black box which may engage in interaction with its environment. Atomic instances of this interaction are called actions and must be elements of the alphabet of the process. A trace of the process is a finite sequence of actions that a process can be observed to engage in. In this paper, structured actions of the form \( b!v \) will be used, where \( v \) is a message and \( b \) is a communication channel. \( b!v \) is said to occur at \( b \) and to cause \( v \) to be exchanged between processes communicating over \( b \). For every channel \( b \), \( \mu b \) is the message set of \( b \) - the set of all \( v \) such that \( b!v \) is a valid action. We define \( ab = \{ b!v \mid v \in \mu b \} \) to be the alphabet of channel \( b \). It is assumed that \( \mu b \) is always finite and non-empty. For a set of channels \( B \), \( \alpha B = \bigcup_{b \in B} \mu b \).

The following notation is similar to that of [6] (below \( t, u, t_1, t_2, \ldots \) are traces; \( b, b', b'' \) are channels; \( B_1, \ldots, B_n \) are disjoint sets of channels; \( a \) is an action; \( A \) is a set of actions; and \( T, T' \) are non-empty sets of traces):

- \( t = (a_1, \ldots, a_n) \) is the trace whose \( i \)-th element is \( a_i \), and whose length, \( |t| \), is \( n \).
- \( t \circ u \) is the trace obtained by appending \( u \) to \( t \).
- \( A^* \) is the set of all traces of actions from \( A \), including the empty trace, \( \langle \rangle \).
- \( T^* \) is the set of all traces \( t = t_1 \circ \cdots \circ t_n \) (\( n \geq 0 \)) such that \( t_1, \ldots, t_n \in T \).
- \( \preceq \) denotes the prefix relation on traces, and \( t \prec u \) if \( t \leq u \) and \( t \neq u \).
- \( \text{Pref}(T) = \{ u \mid \exists t \in T : t \leq u \} \) if the prefix-closure of \( T \); \( T \) is prefix-closed if \( T = \text{Pref}(T) \).
- \( f[b/u] \) is a trace obtained from \( f \) by replacing each action \( b!v \) by \( b'/v \).
- \( f[b/U] \) is obtained by deleting from \( t \) all the actions that do not occur on the channels in \( B \); for example, \( \langle b!1, b!3, b''!2, b''!3, b''!/6, b''!/2 \rangle = \{ b, b, b' \} = \{ b!1, b!3, b''!/6, b''!/2 \} \).
- \( \chi a \) gives the channel on which the event \( a \) occurred; for example, \( \chi b!1 = b \). Moreover, \( \chi A = \{ a \mid a \in A \} \).
- An infinite sequence \( t_1, t_2, \ldots \) is \( \omega \)-monotonic if \( t_1 \leq t_2 \leq \ldots \) and \( \lim_{n \to \infty} |t_n| = \infty \).
- A mapping \( f : T \to T' \) is monotonic if \( t, u \in T \) and \( t \leq u \) implies \( f(t) \leq f(u) \); \( f \) is strict if \( \langle \rangle \in T \) and \( f(\langle \rangle) = \langle \rangle \); and \( f \) is a homomorphism if \( t, u, t \circ u \in T \) implies \( f(t \circ u) = f(t) \circ f(u) \).
- A family of sets \( X \) is subset-closed if \( Y \subseteq X \in X \) implies \( Y \in X \).

2.2 Processes

We use the divergence model of CSP [3, 6, 15] in which a process \( P \) is a triple \((aP, \phi P, \delta P)\) where \( aP \) — alphabet — is a non-empty finite set of actions, \( \phi P \) — failures — is a subset of \( aP^* \times 2\phi P \), and \( \delta P \) — divergences — is a subset of \( aP^* \). The conditions imposed on the three components are given below, where \( \tau P = \{ t \mid (t, R) \in \phi P \} \) denotes the traces of \( P \).
CSP1. \( \tau P \) is a non-empty and prefix-closed set.
CSP2. If \((t, R) \in \phi P \) and \( S \subseteq R \) then \((t, S) \in \phi P \).
CSP3. If \((t, R) \in \phi P \) and \( a \in \alpha P \) satisfy \( t \circ \{a\} \notin \tau P \) then \((t, R \cup \{a\}) \in \phi P \).
CSP4. If \( t \in \delta P \) then \((t \circ v, R) \in \phi P \), for all \( v \in \alpha P^* \) and all \( R \subseteq \alpha P \).

If \((t, R) \in \phi P \) then \( P \) is said to refuse \( R \) after \( t \). Intuitively, this means that if the environment offers \( R \) as a set of possible events to be executed after \( t \), then \( P \) can deadlock. If \( t \in \delta P \) then \( P \) is said to diverge after \( t \). In the CSP model this means the process behaves in a totally uncontrollable way. Such a semantical treatment is based on what is often referred to as 'catastrophic' divergence whereby the process in a diverging state is modelled as being able to accept any trace and generate any refusal. We will also consider maximal failures defined as those belonging to the set

\[
\text{max} \phi P = \{(t, R) \in \phi P \mid (t, S) \in \phi P \wedge R \subseteq S \Rightarrow R = S\}.
\]

We will associate with \( P \) a set of channels, \( \chi P \), and stipulate that the alphabet of \( P \) is that of \( \chi P \). Thus, we shall be able to identify \( P \) with the triple \((\chi P, \phi P, \delta P)\) in lieu of \((\alpha P, \phi P, \delta P)\).

2.3 CSP operators

For our purposes neither the syntax nor the semantics of the whole standard CSP is needed. Essential are only the parallel composition of processes, hiding of the communication over a set of channels and renaming of channels. In the examples we also use deterministic choice, \( P \parallel Q \), non-deterministic choice \( P \cap Q \), and prefixing, \( a \rightarrow P \). All these operators are formally defined in the appendix.

Parallel composition \( P || \parallel Q \) models synchronous communication between processes in such a way that each of them is free to engage independently in any action that is not in the other's alphabet, but they have to engage simultaneously in all actions that are in the intersection of their alphabet. Parallel composition is commutative and associative; we will use \( P_1 \parallel \cdots \parallel P_n \) to denote the parallel composition of processes \( P_1, \ldots, P_n \).

Let \( P \) be a process and \( B \) be a set of channels of \( P \); then \( P \setminus B \) is a process that behaves like \( P \) with the actions occurring at the channels in \( B \) made invisible. Hiding is associative in that \( (P \setminus B) \setminus B' = P \setminus (B \cup B') \).

Let \( P \) be a process with a channel \( b \in \chi P \), and \( b' \) be a channel not in \( \chi P \) such that \( \mu b = \mu b' \). Then \( P[\mu b/\mu b'] \) is a process that behaves like \( P \) except that each action \( b'v \) is replaced by \( b'v \).

A crucial property [3] involving the parallel composition and hiding operators states that if \( P \) and \( Q \) are two processes then

\[
B \subseteq \chi P - \chi Q \Rightarrow (P \setminus B) \parallel Q = (P || Q) \setminus B.
\] (1)

Its relevance follows from an application to modelling of networks of processes.

Processes \( P_1, \ldots, P_n \) form a network if no channel is shared by more than two \( P_i \)'s. We define \( P_1 \parallel \cdots \parallel P_n \) to be the process obtained by taking the parallel composition of the processes and then hiding all interprocess communication, i.e., the process \( (P_1 \parallel \cdots \parallel P_n) \setminus B \), where \( B \) is the set of channels shared by at least two different processes \( P_i \).

Theorem 1. Let \( P_1, \ldots, P_n \) be a network of processes.

1. \( P_1 \parallel P_2 \parallel P_3 \parallel \cdots \parallel P_n = (P_1 \parallel P_2) \parallel P_3 \parallel \cdots \parallel P_n \), if \( n \geq 3 \).
2. \( P_1 \parallel \cdots \parallel P_n = P_{i_1} \parallel \cdots \parallel P_{i_n} \), for any permutation \( i_1, \ldots, i_n \) of \( 1, \ldots, n \).
Proof. (1) Let \( B = \chi((P_1 \odot P_2) || P_3 || \ldots || P_n) - \chi((P_1 \odot P_2) \odot P_3 \odot \ldots \odot P_n) \) and \( B' = \chi P_1 \cap \chi P_2 \). Then
\[
(P_1 \odot P_2) \odot P_3 \odot \ldots \odot P_n = (((P_1 \| P_2) \setminus B') || P_3 || \ldots || P_n) \setminus B
\]
\[
= (((P_1 || P_2) || P_3 || \ldots || P_n) \setminus B') \setminus B \quad \text{(by (1))}
\]
\[
= (P_1 || P_2 || P_3 || \ldots || P_n) \setminus (B \cup B')
\]
\[
= P_1 \odot \ldots \odot P_n.
\]

(2) Follows immediately from \( P_1 || \ldots || P_n = P_1 || \ldots || P_n \). \(\square\)

As a result, a network can be obtained by first composing some of the processes into a subnetwork, and then composing the result with the remaining processes. Moreover, the order in which processes are composed does not matter. In the failure model of CSP, where a process \( P \) is identified with the pair \((\alpha P, \phi P)\), the former property does not hold [3], whence the need for the more complicated divergence model.

To relate failures of a process network with those of the constituent processes, we will use an auxiliary notation. Let \( P \) and \( Q \) be two processes forming a network, \( Z = P \odot Q \) and \((t, R) \in \alpha Z^* \times 2^{\alpha Z}\). Then \( \mathcal{R}_{P, Q}(t, R) \) will denote the set of all triples
\[
(w, R_P, R_Q) \in (\alpha P \cup \alpha Q)^* \times 2^{\alpha P} \times 2^{\alpha Q}
\]
such that \( t = w[\chi Z, \ (w[\chi P, R_P]) \in \phi P, \ (w[\chi Q, R_Q]) \in \phi Q \) and \( R_P \cup R_Q = R \cup (\alpha P \cap \alpha Q) \).

**Proposition 2** The following hold.

1. If \( \mathcal{R}_{P, Q}(t, R) \neq \emptyset \) then \((t, R) \in \phi Z\).
2. If \((t, R) \in \phi Z \) and \( t \notin \delta Z \), then \( \mathcal{R}_{P, Q}(t, R) \neq \emptyset \).

Proof. Follows directly from the definitions of parallel composition and hiding. \(\square\)

Intuitively, \( \mathcal{R}_{P, Q}(t, R) \) comprises realisations of a failure \((t, R)\) of \( P \odot Q \) in terms of failures of the underlying processes, \( P \) and \( Q \). In general, there may be more than one realisation of a given \((t, R)\).

We can partition the channels of a process \( P \) into the input channels, \( \text{in } P \), and output channels, \( \text{out } P \). It is assumed that no two processes in a network have a common input channel or a common output channel and
\[
\text{in } (P_1 \odot \ldots \odot P_n) = \bigcup_{i=1}^{n} \text{in } P_i - \bigcup_{i=1}^{n} \text{out } P_i
\]
\[
\text{out } (P_1 \odot \ldots \odot P_n) = \bigcup_{i=1}^{n} \text{out } P_i - \bigcup_{i=1}^{n} \text{in } P_i.
\]

In the diagrams representing base processes, an outgoing arrow indicates an output channel, and an incoming arrow indicates an input channel. In the diagrams representing implementation processes (other than figure 5), arrowheads are not used.

### 2.4 Basic processes

A class of base processes considered in [9] was that of general input/output processes (GIO). These comprise \( P \) which: (i) are input-guarded, i.e., for every infinite set \( T \) of traces of \( P \), \( T |\text{in } P \) is also infinite; (ii) never refuse any input, i.e., \( \phi P \subseteq \alpha P^* \times 2^{\text{out } P} \).
and (iii) have at least one output channel. In this paper, we relax the restrictions placed on base processes, in the following way. A channel \( c \) of a process \( P \) is \textit{value independent}, denoted \( c \in \text{vind} \ P \), if

\[
\forall (t, R) \in \phi P : \ c \in \chi R \implies (t, R \cup ac) \in \phi P .
\]

We then define an \textit{input-output process} to be a non-diverging process \( P \) such that in \( P \subseteq \text{vind} \ P \), and denote \( P \in IO \).

Since \( GIO \) processes never refuse any input, all their input channels are value independent; moreover, \( GIO \) processes are divergence-free, and so \( GIO \subseteq IO \). The reverse inclusion does not hold. For example, a deterministic buffer of capacity one is an \( IO \) but not \( GIO \) process.

Intuitively, in an \( IO \) process the data component of a message arriving on an input channel \( c \) is irrelevant as far as its receiving is concerned; if one such message can be refused then so can any other message. This is not a restrictive property and, in particular, the standard programming receive constructs like \( c?x \) give rise to value independent input channels.

**Theorem 3.** The class of base \( IO \) processes is compositional, \textit{i.e.}, a non-diverging network of \( IO \) processes is an \( IO \) process.

**Proof.** In view of theorem 1, it suffices to show the result for two base processes forming a network.

Let \( K, L \in IO \) be as in figure 1 and \( \delta(K \odot L) = \emptyset \). We need to show that \( C \cup F \subseteq \text{vind} \ K \odot L \). Without loss of generality, suppose that \( c \in C \) and \( (t, R) \in \phi (K \odot L) \) are such that \( R \cap ac \neq \emptyset \). Since \( K \odot L \) is non-diverging, by proposition 2(2), there is \((w, R_K, R_L) \in \mathcal{R}_{K,L}(t, R)\). Now, since \( R_K \cap ac \neq \emptyset \) and \( K \in IO \), we have \((w \mid \chi K, R_K \cup ac) \in \phi K \), and so \((w, R_K \cup ac, R_L) \in \mathcal{R}_{K,L}(t, R \cup ac)\). Hence, by proposition 2(1), \((t, R \cup ac) \in \phi (K \odot L) \).

Thus \( c \in \text{vind} \ K \odot L \). \( \Box \)

**Fig.1.** Processes in the proof of theorem 3

---

3 Implementation of base processes

In this section, we shall first recall the notion of an extraction pattern. We then define three implementation relations and show that they all satisfy the compositionality and realizability properties.

3.1 An example

Consider a pair of \( IO \) processes, \( Snd \) and \( Buf \), shown in figure 2(a). The former generates an infinite sequence of 0s or an infinite sequence of 1s, depending on the value (0 or 1)
received on its input channel, \( c \), at the very beginning of its execution. The latter is a buffer process of capacity one, forwarding values received on its input channel, \( d \). In terms of CSP, we have:

\[
\begin{align*}
Snd &= \{i \in \{0,1\} | d \mapsto Snd_i \\
Buf &= \{i \in \{0,1\} | d \mapsto B_i \\
\end{align*}
\]

where \( Snd_i \) and \( B_i \ (i = 0, 1) \) are auxiliary processes defined thus:

\[
\begin{align*}
Snd_i &= d \mapsto Snd_i \\
B_i &= e \mapsto Buf.
\end{align*}
\]

![Diagram](image.png)

**Fig. 2.** Two base IO processes and their implementations

Suppose now that the communication on \( d \) has actually been implemented using two channels, \( r \) and \( s \), where \( r \) is a data channel, and \( s \) is a feedback channel used to pass acknowledgements. It is, moreover, assumed that a given message is sent at most twice since a re-transmission always succeeds. This leads to a simple protocol which is incorporated into suitably modified original processes. The resulting implementation processes shown in figure 2(b), \( Snd'_i \) and \( Buf'_i \), are given by:

\[
\begin{align*}
Snd'_i &= \{i \in \{0,1\} | r \mapsto Snd'_i \\
Buf'_i &= \{i \in \{0,1\} | (\text{slack} \rightarrow B'_i \cap s!\text{nak} \rightarrow B) \\
\end{align*}
\]

where \( B, Snd'_i \) and \( B'_i \ (i = 0, 1) \) are auxiliary processes defined thus:

\[
\begin{align*}
Snd'_i &= r \mapsto (\text{slack} \rightarrow Snd'_i \cap s!\text{nak} \rightarrow r!i \mapsto Snd'_i) \\
B &= \{i \in \{0,1\} | r \mapsto B'_i \\
B'_i &= e \mapsto Buf'.
\end{align*}
\]

It may be observed that \( Snd' \circ Buf' = Snd \circ Buf = Snd[e/d] \). One way of showing this would be to compose the two pairs of processes and prove their equality using, e.g., CSP laws [6]. This would be straightforward for \( Snd \circ Buf \), but less so for \( Snd' \circ Buf' \), at least by hand. Alternatively, the compositional way in which we intend to proceed is to show that \( Snd' \) and \( Buf' \) are implementations of the respective base processes according to suitable extraction patterns, and then derive the desired relationship using general results developed later in this section.

### 3.2 Extraction patterns

The notion of extraction pattern (introduced in [9–11]) relates behaviour on a set of channels in an implementation process to that on a channel or channels in a target process. It has two main functions: that of interpretation of behaviour necessitated by interface difference and the encoding of some correctness requirements.
An extraction pattern\(^2\) is a tuple \(ep = (B, b, dom, extr, ref, inv)\) satisfying the following conditions:

**EP0** \(B\) is a non-empty set of channels, called sources and \(b\) is a channel, called target.

**EP1** \(dom\) is a non-empty set of traces over the sources; its prefix-closure will be denoted by \(Dom\).

**EP2** \(extr\) is a strict monotonic mapping defined for traces in \(Dom\); for every \(t\), \(extr(t)\) is a trace over the target.

**EP3** \(ref\) is a mapping defined for traces in \(Dom\); for every \(t\), \(ref(t)\) is a non-empty subset-closed family of subsets of \(aB\) such that \(aB \notin ref(t)\). It is assumed that if \(a \in aB\) and \(t \circ (a) \notin Dom\) then \(R \cup \{a\} \in ref(t)\), for all \(R \in ref(t)\).

**EP4** \(inv\) is a homomorphism from traces over the target to traces in \(Dom\); for every trace \(w\) over the target, \(extr(inv(w)) = w\).

The mapping \(extr\) interprets a trace over the source channels in the implementation process in terms of the interface of the target and defines functionally correct (i.e., in terms of traces) behaviour over those source channels by way of its domain. The mapping \(ref\) is used to define correct behaviour in terms of failures as it gives bounds on refusals after execution of a particular trace sequence over the source channels.

The extraction mapping is monotonic as receiving more information cannot decrease the current knowledge about the transmission. \(aB \notin ref(t)\) means that for an unfinished communication \(t\) we do not allow the sender to refuse all possible transmission. The second condition in EP3 is a rendering of CSP3 in terms of extraction patterns. Note that since \(inv\) is a trace homomorphism, it suffices to define it for single actions over the target only.

To demonstrate that \(Snd\) and \(Buf\) are implementations of respectively \(Snd\) and \(Buf\), we will need to define two kinds of extraction patterns, \(id_c\) and \(ep_{\text{twice}}\).

An identity extraction pattern for a channel \(c\), \(id_c\), is one for which \(B = \{c\}\), \(b = c\), \(dom = Dom = \{c\}\), \(extr(t) = inv(t) = t\) and \(ref(t) = 2\{c\} - \{c\}\).

For the \(ep_{\text{twice}}\) extraction pattern, \(B = \{s, r\}\) are the source channels and \(b = d\) is the target channel; moreover \(\mu d = \mu r = \{0, 1\}\) and \(\mu s = \{ack, nak\}\). The remaining components of \(ep_{\text{twice}}\) are defined in the following way, where \(t \in dom\) and \(t \circ u \in Dom\):

- \(dom = \{r!0, s\text{ack}, r!1, s\text{ack}, r!0, s\text{ack}, r!1, s\text{nak}, r!1, s\text{nak}, r!1\}\)\(^*\)
- \(extr(t \circ u) = \begin{cases} \{\}\text{ if } t \circ u = \{\} \\
extr(t) \circ \langle d!v \rangle \text{ if } u = \langle r!v, s\text{ack} \rangle \text{ or } u = \langle r!v, s\text{nak}, r!v \rangle \\
extr(t) \text{ if } u = \langle r!v \rangle \text{ or } u = \langle r!v, s\text{ack} \rangle 
\end{cases}\)
- \(ref(t \circ u) = \begin{cases} \langle r!v \rangle \text{ if } u = \{\} \\
\{R \in 2^{ar^2 s} | \alpha r \notin R\} \text{ if } u = \{\} \\
\{R \in 2^{ar^2 s} | r!v \notin R\} \text{ if } u = \langle r!v, s\text{ack} \rangle 
\end{cases}\)
- \(inv(d!v) = \{\langle r!v, s\text{ack} \rangle \} .\)

The various components of the extraction patterns can be annotated (e.g., subscripted) to avoid ambiguity. Unless explicitly stated, different extraction patterns will have disjoint sources and distinct targets.

---

\(^2\) What we define here is a basic extraction pattern in the terminology of [9]; [11] also allowed sets of target channels.
Sets of extraction patterns. For notational convenience, we lift some of the notions to finite sets of extraction patterns. Let \( \mathcal{EP} = \{ ep_1, \ldots, ep_n \} \) be a non-empty set of extraction patterns \( ep_i = \{ B_i, k_i, \text{dom}_i, \text{extr}_i, \text{ref}_i, \text{inv}_i \} \); moreover, let \( B = B_1 \cup \ldots \cup B_n \) and \( C = \{ b_1, \ldots, b_n \} \). Then:

- **EP5** \( \text{dom}_{ep} = \{ t \in \alpha B^* \mid \forall i \leq n : t[B_i \in \text{dom}_i] \} \).
- **EP6** \( \text{Dom}_{ep} = \{ t \in \alpha B^* \mid \forall i \leq n : t[B_i \in \text{Dom}_i] \} \).
- **EP7** \( \text{extr}_{ep}(\langle \rangle) = \langle \rangle \) and, for every \( t \circ \langle a \rangle \in \text{Dom}_{ep} \) with \( a \in \alpha B_i \),
  \[
  \text{extr}_{ep}(t \circ \langle a \rangle) = \text{extr}_{ep}(t) \circ u
  \]
  where \( u \) is a (possibly empty) trace such that
  \[
  \text{extr}_{i}(t[B_i \circ \langle a \rangle]) = \text{extr}_{i}(t[B_i] \circ u).
  \]
- **EP8** \( \text{inv}_{ep} \) is a homomorphism from traces over \( C \) to traces over \( B \) such that \( \text{inv}_{ep}(a) = \text{inv}_i(a) \), for all \( i \leq n \) and \( a \in \alpha B_i \).

Note that in EP7 is well defined since \( \text{extr}_i \) is monotonic, and that \( \text{inv}_{ep} \) is well defined since \( b_i \neq b_j \) for \( i \neq j \).

**Proposition 4**

1. \( \text{Dom}_{ep} \) is the prefix-closure of \( \text{dom}_{ep} \).
2. \( \text{extr}_{ep} \) is monotonic and strict.

**Proof.** (1) To show the prefix-closure of \( \text{Dom}_{ep} \), let \( t \circ \langle a \rangle \in \text{Dom}_{ep} \). Then, for every \( i \leq n \), \( (t \circ \langle a \rangle)[B_i \in \text{Dom}_i] \). Since, by EP1, each \( \text{Dom}_i \) is prefix-closed and \( t[B_i \leq (t \circ \langle a \rangle)[B_i] \) then, for every \( i \leq n \), \( t[B_i \in \text{Dom}_i] \). Hence \( t \in \text{Dom}_{ep} \).

We next show that, for every \( t \in \text{Dom}_{ep} \), there is \( u \in \text{dom}_{ep} \) such that \( t \leq u \). Let \( t \in \text{Dom}_{ep} \). By EP6, for every \( i \leq n \), \( t[B_i \in \text{Dom}_i] \). By EP1, for every \( i \leq n \), there are \( u_i \in \text{dom}_i \) and \( v_i \) such that \( t[B_i u_i = v_i] \). Then it follows that \( u = t[v_1 \circ \cdots \circ v_n \in \text{dom}_{ep}] \) is such that \( t \leq u \) and \( u[B_i = v_i] \) for every \( i \leq n \) (note that the \( B_i \)'s are disjoint sets).

To conclude the proof, we observe that \( t \in \text{Dom}_{ep} \), for every \( t \in \text{dom}_{ep} \). Indeed, by EP5, for every \( i \leq n \), \( t[B_i \in \text{Dom}_i] \) and so \( t[B_i \in \text{Dom}_i] \), meaning that \( t \in \text{Dom}_{ep} \), by EP6.

(2) \( \text{extr}_{ep} \) is strict by definition. Moreover, monotonicity follows from \( \text{extr}_{ep}(t \circ \langle a \rangle) = \text{extr}_{ep}(t) \circ u \), in EP7. \( \Box \)

**Proposition 5** Let \( B' = B_{i_1} \cup \cdots \cup B_{i_k} \), \( C' = \{ b_{i_1}, \ldots, b_{i_k} \} \) and \( ep' = \{ ep_{i_1}, \ldots, ep_{i_k} \} \), for some distinct \( i_1, \ldots, i_k \) \( (k \geq 1) \).

1. \( \text{extr}_{ep'}(t[B']) = \text{extr}_{ep}(t)[C'], \) for every \( t \in \text{Dom}_{ep} \).
2. \( \text{inv}_{ep'}(w[C']) = \text{inv}_{ep}(w)[B'], \) for every \( w \in C' \).

**Proof.** (1) The proof proceeds by induction on the length of \( t \). In the base case, \( t = \langle \rangle \), we have \( \text{extr}_{ep}(\langle \rangle[B']) = \langle \rangle = \text{extr}_{ep}(\langle \rangle)[C'] \) since \( \text{extr}_{ep} \) and \( \text{extr}_{ep'} \) are strict by proposition 4(2). In the induction step, we consider \( t' = t \circ \langle a \rangle \). By EP7,

\[
\text{extr}_{ep}(t'[B']) = \text{extr}_{ep}(t[B'] \circ \langle a \rangle)[B'] = \text{extr}_{ep}(t[B'] \circ r)
\]

where \( r \) is (possibly empty) trace and \( u \) is such that \( \text{extr}_{ep}(t \circ \langle a \rangle) = \text{extr}_{ep}(t) \circ u \). By the induction hypothesis, it suffices to show that \( r = u[C'] \). We consider two cases.
Case 1: \( a \notin \alpha B' \). Then \( \langle a \rangle [B'] = \emptyset \) and so \( \text{extr}_{ep}(t[B']) = \text{extr}_{ep}(t[B']) \). Moreover, since the target channels of the \( ep_i \)'s are distinct, \( u[\widehat{C}'] = \emptyset \). We thus have \( r = \emptyset = u[\widehat{C}'] \).

Case 2: \( a \in \alpha B' \). Then \( \langle a \rangle [B'] = \langle a \rangle \) and \( \text{extr}_{ep}(t[B']) = \text{extr}_{ep}(t[B'] \circ \langle a \rangle) = \text{extr}_{ep}(t[B'] \circ \langle a \rangle) \circ u \) (note that \( t[B'] \circ B_i = t[B_i] \) for any \( i \) satisfying \( b_i \in B' \)). Moreover, \( u[\widehat{C}'] = u. \) We thus have \( r = u = u[\widehat{C}'] \).

(2) Since, by EP8, \( \text{inv}_{ep} \) and \( \text{extr}_{ep} \) are homomorphisms, for \( w = \langle a_1 \circ \ldots \circ a_m \rangle \), we have

\[
\begin{align*}
\text{inv}_{ep}(w)[B'] &= (\text{inv}_{ep}(a_1) \circ \ldots \circ \text{inv}_{ep}(a_m))[B'] \\
&= \text{inv}_{ep}(a_1)[B'] \circ \ldots \circ \text{inv}_{ep}(a_m)[B'] \\
\text{inv}_{ep}(w[C']) &= \text{inv}_{ep}(\langle a_1 \rangle)[C'] \circ \ldots \circ \text{inv}_{ep}(\langle a_m \rangle)[C'] \\
&= \text{inv}_{ep}(\langle a_1 \rangle)[C'] \circ \ldots \circ \text{inv}_{ep}(\langle a_m \rangle)[C'] .
\end{align*}
\]

Thus it suffices to show that \( \text{inv}_{ep}(\langle a \rangle)[B'] = \text{inv}_{ep}(\langle a \rangle)[C'] \), for all \( a \in \alpha C \). We first observe that, by EP8, \( \text{inv}_{ep}(\langle a \rangle)[C'] = \text{inv}_{ep}(\langle a \rangle)[C'] \), and then consider two cases.

Case 1: \( a \notin \alpha C' \). Then \( \langle a \rangle [C'] = \emptyset \) and so \( \text{inv}_{ep}(\langle a \rangle)[C'] = \text{inv}_{ep}(\emptyset) = \emptyset \). Moreover, since the source channels of the \( ep_i \)'s are disjoint, \( \text{inv}_{ep}(\langle a \rangle)[B'] = \emptyset \).

Case 2: \( a \in \alpha C' \). Then \( \langle a \rangle [C'] = \langle a \rangle \) and so \( \text{inv}_{ep}(\langle a \rangle)[C'] = \text{inv}_{ep}(\langle a \rangle) \). Moreover, \( \text{inv}_{ep}(\langle a \rangle)[B'] = \text{inv}_{ep}(\langle a \rangle) \).

\[ \square \]

3.3 Implementation relations

We are now ready to introduce three implementation relations which will provide a means to relate the base and implementation processes.

Let \( P \) be a base IO process as in figure 3 and, for every \( i \leq m + n \),

\[ ep_i = (B_i, b_i, dom_i, ext_{ri}, ref_{ri}, inv_i) \]

be an extraction pattern. We assume that the \( B_i \)'s are mutually disjoint channel sets, and denote \( ep = \{ ep_1, \ldots, ep_m \} \) and \( ep' = \{ ep_{m+1}, \ldots, ep_{m+n} \} \). We then take a process \( Q \) such that \( in Q = B_1 \cup \ldots \cup B_m \) and \( out Q = B_{m+1} \cup \ldots \cup B_{m+n} \), as shown in figure 3.

![Fig.3. Base IO process P and its implementation Q](image)

The three implementation relations are defined thus. For \( i = 1, 2, 3 \), we denote \( Q \in \text{Impl}_i (P, ep, ep') \) if respectively IR1–IR3, IR1–IR4 and IR1–IR3&IR5 below hold.

**IR1** If \( t \in \tau Q \) and \( t[\text{in } Q] \in \text{Dom}_{ep} \) then:
(a) \( t \in \text{Dom}_{ep_{up}} \).
(b) \( t \notin \delta Q \).
(c) \( \text{extr}_{pe{up}}(t) \in \tau P \).

**IR2** If \( t_1, t_2, \ldots \) is an \( \omega \)-monotonic sequence of traces in \( \tau Q \cap \text{Dom}_{ep_{up}} \), then the following sequence is also \( \omega \)-monotonic:

\[ \text{extr}_{pe{up}}(t_1), \text{extr}_{pe{up}}(t_2), \ldots \]
The implementation relations defined above form a hierarchy.

**Proposition 6** \( \text{Impl}_3(P, ep, ep') \subseteq \text{Impl}_2(P, ep, ep') \subseteq \text{Impl}_1(P, ep, ep') \).

**Proof.** It suffices to observe that IR5 with \( B = \emptyset \) is equivalent to IR4. \( \square \)

Crucially, all three implementation relations are compositional in the sense that they are preserved by the network composition operation.

**Theorem 7.** Let \( K \) and \( L \) be two base processes whose composition is non-diverging, as in figure 1, and let \( c, d, e, f, g \) and \( h \) be sets of extraction patterns whose targets are respectively the channel sets \( C, D, E, F, G \) and \( H \). Moreover, let \( i \in \{1, 2, 3\} \). If \( M \in \text{Impl}_i(K, c \cup h, d \cup e) \) and \( N \in \text{Impl}_i(L, d \cup f, g \cup h) \) then

\[
M \oplus N \in \text{Impl}_i(K \ominus L, c \cup f, e \cup g)
\]

**Proof.** In the proof, we use \( X \) to denote the sources, and \( Y \) to denote the targets, of an extraction pattern set \( X \), for each \( x \in \{c, d, e, f, g, h\} \). We also use \( Z \) to denote the channel set of a process \( Z \), for \( Z \in \{I, J, K, L, M, N, O, S\} \), where \( I = K \| L, J = K \ominus L, S = M \| N \) and \( O = M \ominus N \). The union of sets of channels or extraction patterns, such as \( C \cup D \) or \( c \cup f \cup g \), is simply denoted as \( C \cup D \) or \( c \cup f \cup g \), respectively. For a channel \( z \) in \( K \cup L \), we denote by \( ep_z \) that extraction pattern which has the target \( z \), its components being indexed by \( z \). Figure 4 may be useful in following the proof details.

\[\]

**Fig. 4.** Processes in the proof of theorem 7; moreover, \( I = K \| L \) and \( S = M \| N \)

Let \( W = \{t \in \tau S \mid t[C \mathcal{F} \in \text{Dom}_{e,f}\} \) and \( W' = \{t \in \tau O \mid t[C \mathcal{F} \in \text{Dom}_{e,f}\} \). Note that both \( W \) and \( W' \) are prefix-closed sets of traces which follows from CSP1 and proposition 4(1). Our next observation is that

\[
t \in W \land t[M \in \tau M \land t[N \in \tau N \implies t \in \text{Dom}_{e,f,g,h}\]
\]
which can be shown by a straightforward induction on the length of the prefixes of \( t \),
using the prefix-closure of the Dom’s and \( IR1(a) \) for \( M \) and \( N \). We now observe that
\[
W \cap \delta S = \emptyset .
\] (4)

For suppose that \( W \cap \delta S \neq \emptyset \). Then, by \( \text{Pref}(W) = W \) and without loss of generality,
there is \( t \in W \cap \delta S \) such that \( t[\mathcal{M} \in \delta M \subseteq \tau M \) and \( t[\mathcal{N} \in \tau N \). By (3), \( t \in \text{Dom}_{\text{defg}} \)
which implies that \( t[\mathcal{C}\mathcal{H} \in \text{Dom}_{\text{cb}} \). Thus \( t[\mathcal{M} \in \delta M \) and \( t[\mathcal{C}\mathcal{H} \in \text{Dom}_{\text{cb}} \), producing
a contradiction with \( IR1(b) \) for \( M \). Thus (4) holds. We then note that from (3,4) it follows that
\[
W \subseteq \text{Dom}_{\text{defg}} .
\] (5)

Suppose now that \( W' \cap \delta O \neq \emptyset \). Then, by (4,5), there is an \( \omega \)-monotonic sequence
of traces \( \{t_1, t_2, \ldots \} \in \tau S \cap \text{Dom}_{\text{defg}} \) such that \( t_i[O = t_j[O, \) for all \( i, j \geq 1 \). Let \( w_i = \text{extr}_{\text{defg}}(t_i) \) for all \( i \geq 1 \). By \( IR1(c) \) for \( M \) and \( N \) and proposition (5,1), \( w_i[K \in \tau K \)
and \( w_i[L \in \tau L, \) for all \( i \geq 1 \). Moreover, by \( IR2 \) for \( M \) and \( N \), both \( w_1[K, w_2[K, \ldots \)
and \( w_1[L, w_2[L, \ldots \) are \( \omega \)-monotonic sequences of traces satisfying \( w_i[K = w_j[K \) and \( w_i[L = w_j[L, \)
for all \( i, j \geq 1 \) (which follows from proposition 5(1) and \( t_i[O = t_j[O, \) for all \( i, j \geq 1 \).
Hence \( w_1[J \in \delta J, \) producing a contradiction with \( J = K \cap L \) being non-diverging. Thus
\[
W' \cap \delta O = \emptyset \quad \text{and} \quad W' = W'[O .
\] (6)

We now proceed with the proof proper. That \( IR1 \) holds for \( O \) follows from (5,6),
proposition 5(1) and the assumption that \( IR1(c) \) holds for \( M \) and \( N \).

To show \( IR2 \) suppose that \( t_1, t_2, \ldots \) is an \( \omega \)-monotonic sequence of traces in \( \tau O \cap \text{Dom}_{\text{defg}} \).
Then, by (5,6), there is a sequence of traces \( w_1, w_2, \ldots \) in \( \tau S \cap \text{Dom}_{\text{defg}} \)
such that \( t_i = w_i[O, \) for all \( i \geq 1 \). Thus, by König’s Lemma, there is an \( \omega \)-monotonic
sequence \( w_i_1, w_i_2, \ldots \) such that \( w_1 < w_2 < \ldots \) which means, by \( IR2 \) for \( M \) and \( N \),
that \( \text{extr}_{\text{defg}}(w_i_1), \text{extr}_{\text{defg}}(w_i_2), \ldots \) is an \( \omega \)-monotonic sequence of traces. Hence,
by proposition 5(1), \( \text{extr}_{\text{defg}}(t_1), \text{extr}_{\text{defg}}(t_2), \ldots \) is an \( \omega \)-monotonic sequence of traces.
Thus, by proposition 4(2), \( \text{extr}_{\text{defg}}(t_1), \text{extr}_{\text{defg}}(t_2), \ldots \) is \( \omega \)-monotonic.

To show \( IR3 \), let \( (t, R) \in \phi O \) be such that \( t \in \text{Dom}_{\text{defg}} \). Moreover, let \( Z_X \subseteq X, \)
for \( X \in \{C, E, F, G\}, \) be (possibly empty) sets of channels of base processes such that,
for every \( z \in Z_E \cup Z_G, \) \( \alpha_B \cap R \notin \text{ref}_z(w[B_x]) \) and, for every \( z \in Z_C \cup Z_F \), \( \alpha_B \cap R \in \text{ref}_z(w[B_x]) \). By (6) there is \( \{w, \mathcal{R}_M, \mathcal{R}_N\} \in \mathcal{R}_{M,N}(t, R) \). Thus, it follows directly from \( IR3(a) \) for \( M \) and \( N \) that \( IR3(a) \) holds for \( O \) as well. To show that \( IR3(b) \) also holds,
we assume additionally that, for every channel \( z \in J, w[B_x \in \text{dom}_{\alpha_z} \) and proceed thus.

Let \( Z_D \subseteq D \) and \( Z_H \subseteq H \) be the sets of all channels \( z \) such that:
(i) \( \alpha_B \cap R \notin \text{ref}_z(w[B_x]) \), for every \( z \in Z_H; \) and (ii) \( \alpha_B \cap R \notin \text{ref}_z(w[B_x]) \), for every \( z \in Z_D \). We
then observe that, from \( \alpha_D \cup \alpha_H \subseteq R_M \cup R_N \) and \( \text{EP3} \), it follows that:
(iii) \( \alpha_Y \cap R \notin \text{ref}_z(w[B_x]) \), for every \( z \in H - Z_H \); and (iv) \( \alpha_B \cap R \in \text{ref}_z(w[B_x]) \), for every \( z \in Z_D \). Thus, by (iv) and \( IR3(a) \) for \( M \) and \( N \) it follows that \( w[B_x \in \text{dom}_{\alpha_z} \) for every channel \( z \in D \cup H \). We now can use \( IR3(b) \) for \( M \) and \( N \) to conclude that
\[
(\text{extr}_{\text{defg}}(w), \alpha Z) \in \phi I
\]
where \( Z = Z_C \cup Z_D \cup Z_E \cup Z_F \cup Z_G \cup Z_H \cup (D - Z_D) \cup (H - Z_H) \). Thus
\[
(\text{extr}_{\text{defg}}(w), \alpha Z') \in \phi J
\]
where \( Z' = Z_C \cup Z_E \cup Z_F \cup Z_G \). This completes the proof for \( i = 1 \) (i.e., for \( \text{Impl}_1 \)).

It is straightforward to show that the result holds also for \( i = 2 \), using proposition 5(2). To show it holds for \( i = 3 \), let \( Y \subseteq \mathcal{J} \) and \( (t, \alpha Y) \in \phi J \). Denote \( Y_K = Y \cap K \) and \( Y_L = Y \cap L \). Since \( \delta J = \emptyset \), there is \( (w, \mathcal{R}_K, \mathcal{R}_L) \in \mathcal{R}_{K,L}(t, \alpha Y) \).
Let $Y_D \subseteq D$ and $Y_H \subseteq H$ be the sets of all channels $z$ such that: (i) $R_L \cap \alpha z \neq \emptyset$, for $z \in D$; and (ii) $R_K \cap \alpha z \neq \emptyset$, for $z \in H$. Then, by $\alpha D \cup \alpha H \subseteq R_K \cup R_L$, we have:

(iii) $\alpha Y_D' \subseteq R_K$ where $Y_D' = D - Y_D$; and (iv) $\alpha Y_H' \subseteq R_L$ where $Y_H' = H - Y_H$. Since both $K$ and $L$ are base processes and so their input channels are value independent, $(w[K, R_K \cup \alpha Y_H]) \in \delta K$ and $(w[L, R_L \cup \alpha Y_D]) \in \delta L$. Thus, by IR5 for $M$ and $N$, as well as by IR1($a$) for $M$ and $N$ and CSP3, we have

\[
\begin{align*}
\{a \in \bigcup_{z \in \alpha D} \alpha B_z \mid inv_{cdhb}(w[K]) \circ \langle a \rangle \in Dom_{cdhb}\} & \in \delta M \\
\{a \in \bigcup_{z \in \alpha D} \alpha B_z \mid inv_{cdhb}(w[K]) \circ \langle a \rangle \notin Dom_{cdhb}\} & \in \delta M \\
\{a \in \bigcup_{z \in \alpha H} \alpha B_z \mid inv_{fgzh}(w[L]) \circ \langle a \rangle \in Dom_{fgzh}\} & \in \delta N \\
\{a \in \bigcup_{z \in \alpha H} \alpha B_z \mid inv_{fgzh}(w[L]) \circ \langle a \rangle \notin Dom_{fgzh}\} & \in \delta N.
\end{align*}
\]

One can then easily see that

\[
(inv_{cfg}(t), \{a \in \bigcup_{z \in \alpha} \alpha B_z \mid inv_{cfg}(t) \circ \langle a \rangle \in Dom_{cfg}\}) \in \delta O
\]

which completes the proof of IR5 for $O$. \hfill \square

### 3.4 Realisability relations

In this section, we assume that $P$ is a (specification) base IO process and $Q$ is its implementation which can be used in place of $P$ in an environment $T$ providing inputs and accepting outputs, as shown in figure 5. We will define three implementation relations, of increasing complexity, but also guaranteeing progressively better approximation of the behaviour of the specification process.

![Fig.5. Relating a base process and its implementation](image)

In the definitions of the three realisability relations, we assume that $P$, $T$ and $Q$ are non-diverging processes such that $in P = in Q \subseteq \alpha out T$ and $out P = out Q \subseteq in T$. Moreover, $P$ and $T$ are IO processes whose composition is non-diverging, i.e., $\delta(P \odot T) = \emptyset$. The realisability relations are defined thus.

For $i = 1, 2, 3$, we denote $Q \preceq_i P$ if respectively RR1, RR1&RR2 and RR1&RR3 below hold.

**RR1** If $C \subseteq in \ P$, $D \subseteq \alpha out P$ and $(t, \ R \cup \alpha D) \in \delta Q$ are such that $C \subseteq \chi R$, then $(t, \alpha C \cup \alpha D) \in \delta P$.
RR2 \( \tau P \subseteq \tau Q \).
RR3 If \( B \subseteq \chi P \) and \( (t, aB) \in \phi \), then \( (t, aB) \in \phi Q \).

We then obtain a result characterising the degree to which each of the realisability relations approximates the behaviours of the base processes.

**Theorem 8.** Let \( k \in \{1, 2, 3\} \) and \( Q \preceq_i P \).

1. If \( k = 1 \) then \( \delta(Q \otimes T) = \emptyset \) and \( \phi(Q \otimes T) \subseteq \phi(P \otimes T) \) and \( \tau(Q \otimes T) \subseteq \tau(P \otimes T) \).
2. If \( k = 2 \) then \( \delta(Q \otimes T) = \emptyset \) and \( \phi(Q \otimes T) \subseteq \phi(P \otimes T) \) and \( \tau(Q \otimes T) = \tau(P \otimes T) \).
3. If \( k = 3 \) then \( Q \otimes T = P \otimes T \).

**Proof.** (1) We first observe that \( \tau Q \subseteq \tau P \) since it suffices to apply RR1 with \( C = D = \emptyset \) and \( (t, \emptyset) \in \phi Q \). Hence, since \( P \otimes T \) and \( Q \) are non-diverging processes, \( \delta(Q \otimes T) = \emptyset \).

Let \((t, R) \in \phi(Q \otimes T)\). Then, since \( \delta(Q \otimes T) = \emptyset \), there is \((w, R_Q, R_T) \in \mathbb{R}_{Q,T}(t, R)\). Let \( C \) be the set of all the channels \( c \in \chi \) such that \( \cap c \neq \emptyset \), and \( D \) be the set of all the channels \( d \in \text{out} Q \) such that \( \alpha d \subseteq \cap R_Q \). Then, by RR1, \((w | \alpha Q, \alpha C \cup \alpha D) \in \phi P \).

We now observe that, for every channel \( c \in \text{in} P - C \subseteq \text{out} T \), it is the case that \( \alpha c \subseteq R_T \) (since \( \alpha c \subseteq \cap R_T \) and \( \alpha c \cap \cap c = \emptyset \)). Moreover, for every channel \( d \in \text{out} P - D \subseteq \text{in} T \), it is the case that \( \cap T \cap \alpha d \neq \emptyset \) (since \( \alpha d \subseteq \cap R_T \) and \( \alpha d \subseteq \cap R_Q \)). Hence, since \( T \) is a base process, \((w | \chi T, \cap T \cup \alpha(\text{out} P - D)) \in \phi T \). As a result, \((w, \cap T \cup \alpha P) \in \phi(\text{in} T)\). Hence \((t, R) \in \phi(P \otimes T)\). Thus \( \phi(Q \otimes T) \subseteq \phi(P \otimes T) \) and so \( \tau(Q \otimes T) \subseteq \tau(P \otimes T) \).

(2) To show \( \tau(P \otimes T) \subseteq \tau(Q \otimes T) \), let \( t \in \tau(P \otimes T) \). By \( \delta(P \otimes T) = \emptyset \), there is \( w \in \tau(P ||T) \) such that \( t = w[\chi(P \otimes T)] \). Hence, by RR2, \( w \in \tau(Q ||T) \) and so \( t = w[\chi(Q \otimes T)] \in \tau(Q \otimes T) \).

(3) In view of part (1), it suffices to show that \( \phi(P \otimes T) \subseteq \phi(Q \otimes T) \). Let \((t, R) \in \phi(P \otimes T)\). Then, since \( \delta(P \otimes T) = \emptyset \), there is \((w, R_P, R_T) \in \mathbb{R}_{P,T}(t, R)\).

Let \( C \) be the set of all the channels \( c \in \chi \) such that \( \cap P \cap \alpha c \neq \emptyset \), and \( D \) be the set of all the channels \( d \in \text{out} P \) such that \( \alpha d \subseteq \cap P \). Since \( P \) is a base process, we have \((w | \chi P, \cap P \cup \alpha C) \in \phi P \). Thus, by RR3, \((w | \alpha P, \alpha C \cup \alpha D) \in \phi Q \). The rest of the proof is similar to the argument made in part (1). \( \Box \)

We have defined three realisability relations and demonstrated how they can be used to approximate the behaviour of a base process in an environment provided by another IO process. We now show that these realisability notions correspond to the implementation relations developed in the previous section under the proviso that only identity extraction patterns are involved. Referring to the notation used in section 3.3, we first observe that if \( ep \) and \( ep' \) are sets of identity extraction patterns, then RR2 is vacuously true and that IR1\&IR3\&IR5 reduce to the following.

IR1' \( \delta Q = \emptyset \) and \( \tau Q \subseteq \tau P \).
IR3' If \( (t, R) \in \phi Q \) and \( B \subseteq \chi P \) are such that

\[
\begin{align*}
\text{for every } b_i \in B, \text{ then } (t, aB) &\in \phi P, \\
\text{then } (t, aB) &\in \phi P.
\end{align*}
\]

IR4' \( \tau P \subseteq \tau Q \).
IR5' If \( B \subseteq \chi P \) and \( (t, aB) \in \phi P \), then \( (t, aB) \in \phi Q \).

**Theorem 9.** Let \( i \in \{1, 2, 3\} \) and \( ep \) and \( ep' \) be sets of identity extraction patterns. Then \( Q \in \text{Impl}(P, ep, ep') \) if and only if \( Q \preceq_i P \).
Proof. Notice that RR1, RR2, RR3 are nothing but IR3', IR4', and IR5', respectively. Moreover, from RR1 with \( C = D = \emptyset \), we obtain \( \tau Q \subseteq \tau P \). \( \square \)

The realizability results can be strengthened after assuming that the base process \( P \) is deterministic by which we mean that:

- **D1** If \( (t, \{a\}) \in \delta P \) then \( t \circ \langle a \rangle \notin \tau P \).
- **D2** If \( t \circ \langle a \rangle, t \circ \langle b \rangle \in \tau P \) and \( \chi a = \chi b \in \text{out} P \), then \( a = b \).

Note that D1 is the usual defining condition for deterministic CSP processes [6]. We have added D2 which essentially means that \( P \) is not allowed to produce different results on any of its output channels for a given set of input messages.

**Proposition 10** If \( P \) is a deterministic IO process and \( Q \preceq_{3} P \) then \( Q = P \).

**Proof.** From the definitions of an IO process and \( \preceq_{3} \), it follows that \( \tau Q = \tau P \) and \( \delta Q = \delta P = \emptyset \). Thus, by D1 and CSP3, \( \delta P \subseteq \delta Q \). Hence it suffices to show that \( \delta P \subseteq \delta Q \). This, in turn, will follow from \( \tau Q = \tau P \) and

\[
(t, \langle a \rangle) \in \delta Q \Rightarrow t \circ \langle a \rangle \notin \tau Q . \tag{7}
\]

Suppose \( (t, \langle a \rangle) \in \delta Q \) and \( a \in \alpha k \). We consider two cases.

- **Case 1:** \( b_i \in \text{in} Q \). Then, by RR1, \( (t, ab_i) \in \delta P \). So, by D1, \( t \circ \langle a \rangle \notin \tau P \Rightarrow t \circ \langle b_i \rangle \notin \tau P \).

- **Case 2:** \( b_i \in \text{out} Q \). Suppose that \( t \circ \langle a \rangle \in \tau Q \). Then, by D2 and \( \tau Q = \tau P \), we have \( t \circ \langle b_i \rangle \notin \tau Q \), for all \( b \in \alpha b_i - \{a\} \). Hence, by CSP3, \( (t, ab_i) \in \delta Q \) and so, by RR1, \( (t, ab_i) \in \delta P \). This, however, contradicts D1 and \( t \circ \langle a \rangle \in \tau P \). \( \square \)

### 3.5 Example revisited

It can be shown that

\[
\text{Snd}^d \in \impl_3(\text{Snd}, \{id_c\}, \{ep\}_\text{twice}) \quad \text{and} \quad \text{Buf}^d \in \impl_3(\text{Buf}, \{ep\}_\text{twice}, \{id_c\}) .
\]

Hence, by theorems 7 and 9, and \( \text{Snd} \oplus \text{Buf} = \text{Snd}[e/d] \), we have:

\[
\text{Snd}^d \oplus \text{Buf}^d \preceq_3 \text{Snd} \oplus \text{Buf} = \text{Snd}[e/d] .
\]

Moreover, \( \text{Snd} \) is a deterministic IO process and so, by proposition 10,

\[
\text{Snd}^d \oplus \text{Buf}^d = \text{Snd} \oplus \text{Buf} = \text{Snd}[e/d] .
\]

### 4 Representing extraction patterns and CSP processes

Extraction patterns and CSP processes are potentially infinite objects. We therefore need a means to represent them in a finite way in order to allow a computer implementation. To deal with CSP processes we will use the standard device of a transition system, while extraction patterns will be represented by the novel notion of an extraction graph.
4.1 Communicating transition systems

In order to represent processes in a manner amenable to computer representation, we take advantage of the operational semantics defined for CSP in [15]. This uses a labelled transition system (LTS) to represent a CSP process. An LTS may be derived from an algebraic representation of a CSP process using the inference system detailed in [15]. For the purposes of this paper, however, we can simply assume that a process is given in the form of an LTS, without having to worry about how such a representation has been obtained.

We shall represent a process in terms of a communicating transition system (CTS), which is essentially an LTS incorporating additional information about channels. A communicating transition system is a tuple

$$CTS = (V, C, D, A, v_0)$$

such that: $V$ is a set of states (nodes); $v_0 \in V$ is the initial state; $C$ and $D$ are finite disjoint sets of channels ($C$ will represent input and $D$ output channels); and $A \subseteq V \times (\alpha C \cup \alpha D \cup \{\tau\}) \times V$ is the set of labelled directed arcs, called transitions, where $\tau$ is a distinguished symbol denoting an internal action. We will use the following notation:

- If $(v, a, w) \in A$, we denote $v \xrightarrow{a} w$.
- If $v_1 \xrightarrow{a_1} v_2 \xrightarrow{a_2} \cdots \xrightarrow{a_n} v_{n+1}$, we denote $v_1 \xrightarrow{\{a_1\} \cdots \{a_n\}} v_{n+1}$ where it is assumed that $\langle \tau \rangle = \langle \rangle$; moreover, $v \xrightarrow{\emptyset} v$, for every $v \in V$.
- If $v \xrightarrow{a} w$, we denote $a \in \text{en}(v)$ and call $a$ enabled at $v$.
- A state $v \in V$ is stable if $\tau \notin \text{en}(v)$; the set of stable states will be denoted by $V_{stb}$.
- If $v \xrightarrow{\emptyset} w$, we denote $v \xrightarrow{\emptyset} w$ or $v \xrightarrow{\emptyset}$.

We shall assume that a transition system is finite, i.e., both $V$ and $A$ are finite. Figure 6 shows the graph of a communicating transition system such that $C = \{d\}$ and $D = \{e\}$, where $\mu d = \mu e = \{0, 1\}$. Note that the initial state is indicated by the node with a white centre.

4.2 Traces and failures information

The implementation relations which we want to verify algorithmically are all expressed in the denotational semantics, and so we must know how to derive information on divergences, traces and failures from a given CTS.
Let $CTS = (V, C, D, A, v_0)$ be a communicating transition system. Then $PCTS = (C, D, \Phi, \Delta)$ is a tuple such that the following hold (below $\alpha CTS = \alpha C \cup \alpha D$):

$$\Delta = \{ t \circ u \in \alpha CTS \mid \exists k \geq 1 \exists v_1, \ldots, v_k \in V : v_0 \xrightarrow{t} v_1 \xrightarrow{r} \cdots \xrightarrow{r} v_k \xrightarrow{r} v \}$$

$$\Phi = \{ (t, R) \in \alpha CTS \times 2^{\alpha CTS} \mid \exists v \in V_{th} : v_0 \xrightarrow{t} v \land R \cap en(v) = \emptyset \} \cup \Delta \times 2^{\alpha CTS}.$$ 

Note that a divergence is represented in a CTS by a cycle composed only of $\tau$-labelled transitions which is reachable from the initial state.

It is not difficult to check that the communicating transition system in figure 6 models the buffer of capacity one defined in section 3.1; i.e., $PCTS_{buf} = \text{Buf}$.

**Proposition 11** $PCTS$ is a CSP process. Moreover, if $\Delta = \emptyset$, then

$$\tau PCTS = \{ t \in \alpha CTS \mid v_0 \xrightarrow{t} \}.$$ 

**Proof.** As required by the definition of a process, $C$ and $D$ are finite, disjoint sets of channels. Below, we denote $\tau PCTS = \{ t \mid (t, R) \in \Phi \}$. We now show that $PCTS$ satisfies CSP1-4.

Proving CSP1: We first show that $\tau PCTS$ is prefix-closed. Let $(t, R) \in \Phi$. We consider two cases.

Case 1: $t \notin \Delta$. Then, by definition, there are $v_1, \ldots, v_n \in V$ such that $v_0 \in V_{th}$ and $v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} v_n$ and $t = \langle a_1 \rangle \circ \cdots \circ \langle a_n \rangle$. Let $u$ be a trace such that $u \leq t$. Then there is $0 \leq i \leq n$ such that $u = \langle a_1 \rangle \circ \cdots \circ \langle a_i \rangle$.

If $v_i$ is a stable node then $R$ exists (e.g., $R = \emptyset$) such that $R \cap en(v_i) = \emptyset$. In the case that $v_i \notin V_{th}$, since $t \notin \Delta$ and so no prefix of $t$ may be a divergent trace, it follows that there exists a stable node $v'$ such that $v_i \xrightarrow{0} v'$. Again, an $R$ exists such that $R \cap en(v') = \emptyset$. Thus, in either case, there is $R$ such that $(u, R) \in \Phi$ and so $u \in \tau PCTS$.

Case 2: $t \in \Delta$. If there is no prefix of $t$ which is divergent, then there exists a node $v \in V$ such that $v_0 \xrightarrow{t} v$. We then follow similar reasoning as for Case 1 above to show that if $u \leq t$ then there is $R$ such that $(u, R) \in \Phi$. If $t$ has a divergent prefix, let $u$ be the trace such that $u \leq t$, $u \in \Delta$ and where, for every $r$ such that $r \leq u$, $r \notin \Delta$. By definition of $\Delta$, for every trace $s$ such that $u \leq s \leq t$, $s \in \Delta$. It therefore follows that $(s, R) \in \Phi$, where $R \in 2^{\alpha CTS}$. We then follow similar reasoning as in the first part of Case 2, to show that for every $v$ such that $v \leq u$, there exists $R$ such that $(v, R) \in \Phi$.

We next show that $\tau PCTS$ is non-empty. If $v_0$ is a stable node, then $\emptyset \in \tau PCTS$, and so $(\emptyset, \emptyset) \in \Phi$. If $v_0 \notin V_{th}$, either there is a node $v \in V_{th}$ such that $v_0 \xrightarrow{0} v$, or $\emptyset \notin \Delta$. In the former case, $\emptyset \in \tau PCTS$, and, therefore, $(\emptyset, \emptyset) \in \Phi$. In the latter case, $(\emptyset, R) \in \Phi$, for every $R \in 2^{\alpha CTS}$.

Proving CSP2: Let $(t, R) \in \Phi$ and $S \subseteq R$. We consider two cases.

Case 1: $t \notin \Delta$. Then there exists a stable node $v$ such that $v_0 \xrightarrow{t} v$ and $R \cap en(v) = \emptyset$. Thus, since $S \subseteq R$, we have $S \cap en(v) = \emptyset$ and so $(t, S) \notin \Phi$.

Case 2: $t \in \Delta$. Then $(t, S) \in \Phi$, for every $S \in 2^{\alpha CTS}$; in particular, for every $S \subseteq R$.

Proving CSP3: Let $(t, R) \in \Phi$ and $a \in \alpha CTS$ be such that $t \circ (a) \notin \tau PCTS$. We consider two cases.

Case 1: $t \notin \Delta$. Then there exists a stable node $v \in V$ such that $R \cap en(v) = \emptyset$. Since $t \notin \Delta$, $en(v)$ and $t \circ (a) \notin \tau PCTS$, $a \notin en(v)$ and so $(R \cup \{a\}) \cap en(v) = \emptyset$. It follows that $(t, R \cup \{a\}) \in \Phi$. 

Case 2: $t \in \Delta$. Then $t \circ (a) \in \Delta$ and, by definition of $\Delta$, there exists $R$ such that $(t \circ (a), R) \in \Phi$. Hence $t \circ (a) \in \tau \Phi_{CTS}$, yielding a contradiction with $t \circ (a) \notin \tau \Phi_{CTS}$.

Proving CSP: If $t \in \Delta$, then $t \circ u \in \Delta$, for all $u \in \alpha CTS^*$. Moreover, by definition, $(t \circ u, R) \in \Phi$, for every $R \in \alpha CTS$.

We have shown that $\Phi_{CTS}$ meets CSP 1, i.e., that it is a CSP process. To show the second part of the proposition, we observe that if $\Delta = \emptyset$ then

$$\Phi = \{ (t, R) \in \alpha CTS^* \times \alpha CTS | \exists v \in V_{stb} : v \xrightarrow{t} v' \land \emptyset \cap en(v) = \emptyset \}$$

and so $\tau \Phi_{CTS} = \{ t \in \alpha CTS^* | \exists v \in V_{stb} : v \xrightarrow{t} v' \}$. Moreover, since $\Delta = \emptyset$, for every node $v \in V$ there is a stable node $v'$ such that $v \xrightarrow{t} v'$. Then, by an argument similar to that used in Case 1 for CSP1, $\tau \Phi_{CTS} = \{ t \in \alpha CTS^* | v \xrightarrow{t} \}$.

Let $Buf$, $Snd$, $Buf'$ and $Snd'$ be processes defined in section 3.1. As already mentioned, $\Phi_{CTS_{Buf}} = Buf$ where $CTS_{buf}$ is as in figure 6. Moreover, $\Phi_{CTS_{Snd}} = Snd$, $\Phi_{CTS_{Buf'}} = Buf'$ and $\Phi_{CTS_{Snd'}} = Snd'$, where $CTS_{snd}$, $CTS_{buf'}$ and $CTS_{snd'}$ are as in figure 7.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig7.png}
\caption{Communicating transition systems}
\end{figure}

Calculating failures On the basis of the previous definition, we proceed to elaborate on how failures information may be explicitly derived from a CTS.

The set of maximal refusals $R$, for any particular non-diverging $t$ such that $(t, R) \in \phi \Phi_{CTS}$ may be generated simply by considering the events not on offer at all stable
nodes \(v\) reachable from the initial state, \(v_0 \xrightarrow{t} v\). All other refusals can be derived from the maximal ones.

In order that all sets of nodes representing the same trace may be grouped together, a normalisation process is used, as detailed in [15] specifically with respect to the operational semantics of CSP. This produces a CTS with two fundamental properties: (i) there are no \(\tau\) transitions; and (ii) each node has a unique successor on each visible action it can perform. This normalisation serves two main purposes.

- It creates a deterministic CTS in order that trace inclusion properties may be easily tested for.
- Every node \(p\) in the new CTS such that \(p_0 \xrightarrow{t} p\) is mapped to a set of stable nodes in the original CTS.

The algorithm used for this normalisation process is adapted from [15]. Given a finite CTS \((V, C, D, A, v_0)\) such that \(\Delta = \emptyset\), we form a labelled transition system \(CTS_{det}\) whose nodes \(V_{CTS_{det}}\) are members of the powerset of \(V\), as follows:

1. The initial node is the set \(p_0 = \mathcal{T}(v_0)\) where, for any node \(v \in V\), \(\mathcal{T}(v) = \{w \in V \mid v \xrightarrow{\tau} w\}\) are the nodes reachable under some sequence of \(\tau\)'s from \(v\).
2. For each node \(p\) generated, we determine the set of non-\(\tau\) actions possible for \(v \in p\), that is all actions in \(\bigcup_{v \in p} \text{en}(v) - \{\tau\}\). For each such action \(a\), we form a new node, \(p' = \{\mathcal{T}(w) \mid \exists v \in p : v \xrightarrow{a} w\}\), the set of all nodes reachable after action \(a\) and any number of \(\tau\)'s from members of \(p\). We also add a transition \(p \xrightarrow{a} p'\).

We also denote, for every node \(p\) of \(CTS_{det}\),

\[
\kappa_{CTS}(p) = \{\text{en}(v) \mid v \in p \land \forall w \in p : \text{en}(w) \subseteq \text{en}(v) \Rightarrow \text{en}(w) = \text{en}(v)\}.
\tag{8}
\]

In other words, \(\kappa_{CTS}(p)\) is the set of minimal sets of actions enabled at stable nodes corresponding to \(p\). Such a set can then be used to calculate the maximal failures.

**Proposition 12** Let CTS be a communicating transition system such that \(\Delta = \emptyset\), and \(CTS_{det}\) be the determinised version of CTS, with the initial state \(p_0\). Then \((t, R) \in \phi P_{CTS}\) if and only if there exist \(p_0 \xrightarrow{t} p\) and \(A \in \kappa_{CTS}(p)\) such that \(R \subseteq \alpha P_{CTS} - A\).

**Proof.** Follows from proposition 11, \(\Delta = \emptyset\), and the definitions of \(\Phi\) and \(\kappa_{CTS}\). \(\Box\)

**Corollary 13** \((t, R) \in \text{max}\phi P_{CTS}\) if and only if there exist \(p_0 \xrightarrow{t} p\) and \(A \in \kappa_{CTS}(p)\) such that \(R = \alpha P - A\).

### 4.3 Testing value independence

We now investigate how one may check whether the input channels of a process represented by a CTS are value independent. We first show that in the definition of value independence (2) it is necessary to consider only maximal failures.

**Proposition 14** The definition of a value independent channel (2) can be replaced by:

\[
\forall (t, R) \in \text{max}\phi P : \exists c \in \chi R \Rightarrow \alpha c \subseteq R.\tag{9}
\]

\[\text{We use only the first part of that normalisation process and do not deal with nodes which are bisimilar.}\]
Proof. Let \((t, R') \in \delta P\) and \(c \in \chi R'.\) Then there is \((t, R) \in \max \delta P\) such that \(R' \subseteq R\) and \(c \in \chi R.\) The latter and (9) means that \(\alpha c \subseteq R.\) Hence, by CSP2, \((t, R' \cup \alpha c) \in \delta P.\) \(\square\)

The rendering of value independence in terms of a communicating transition system CTS is as follows. Let \(c \in C \cup D.\) We denote \(c \in \text{vind} CTS\) if, for all \(p \in V_{CTS_{\Delta}}\) and \(A \in k_{CTS}(p),\) if \(\alpha c - A \neq \emptyset\) then \(\alpha c \cap A = \emptyset.\)

**Proposition 15** \[\text{vind} CTS = \text{vind} P_{CTS},\] provided that \(\Delta = \emptyset.\)

**Proof.** By corollary 13, \((t, R) \in \max \delta P_{CTS}\) if and only if there exist \(p_0 \overset{t}{\Rightarrow} p\) and \(A \in k_{CTS}(p)\) such that \(R = \alpha P - A.\) Moreover, for \(c \in C \cup D,\) \(c \in \chi R \iff \alpha c - A \neq \emptyset\) and \(\alpha c \subseteq R \iff \alpha c \cap A = \emptyset.\) Thus the result follows from (9) and the definition of \text{vind} CTS. \(\square\)

The above considerations give the following algorithm, used to test for the value independence of input channels.

**Algorithm 1.** Let \(CTS = (V, C, D, A, v_{\emptyset})\) be a communicating transition system such that \(\Delta = \emptyset.\) Moreover, let \(C = \{1, \ldots, k\}\) and, for simplicity, \(\mu c = \{1, 2, \ldots, l\},\) for every \(c \in C.\) The pseudo-code of the algorithm is given in figure 8.

```
// To indicate whether a particular event was offered
bool array [1..k, 1..l] chanEvents

// To indicate the number of events refused on a channel
int array [1..k] counters

function valIndep()
for every \(p \in V_{CTS_{\Delta}}\)
for every \(A \in k_{CTS}(p)\)
counters \gets \emptyset
chanEvents \gets false
for every \(c \in A\)
if \(\neg \text{chanEvents}[c][e]\)
then
chanevents[c][e] \gets true
counters[c] \gets counters[c] - 1
for every \(c \in C\)
if \(0 \neq \text{counters}[c] \neq l\) then return failure
return success
```

Fig. 8. Testing for value independence of input channels

4.4 Extraction graphs

We now turn to the representation of extraction patterns. An extraction graph is a tuple

\[EG = (B, b, V, A, v_{\emptyset}, g, \delta, l)\]
such that: $B$ is a non-empty finite set of channels; $b$ is a channel; $V$ is a set of nodes; $A \subseteq V \times (aB \times aB^*) \times V$ is a set of labelled arcs; $v_0 \in V$ is the initial node; $g$ is a mapping returning for every node in $V$ a non-empty family of proper subsets of $aB$; $\delta : V \to \{d, D\}$; and $i : aB \to aB^*$.

Intuitively, $g$ corresponds to $\text{ref}$, $d$ indicates traces in $\text{dom}$, $D$ indicates traces in $\text{Dom} - \text{dom}$, and $i$ corresponds to $\text{inv}$ (see section 3.2). We will use the following notation:

**EN1** If $(v, a, t, w) \in A$, we denote $v \xrightarrow{a, t} w$.

**EN2** If $v_1 \xrightarrow{a_1, t_1} v_2 \xrightarrow{a_2, t_2} \cdots \xrightarrow{a_n, t_n} v_{n+1}$, we denote $v_1 \xrightarrow{\{a_1, a_2, \ldots, a_n\}; t_1, t_2, \ldots, t_n} v_{n+1}$; moreover, $v \xrightarrow{t} v$, for every node $v$.

**EN3** If $v \xrightarrow{w} w$, we denote $v \xrightarrow{w} w$.

We impose the following restrictions on an extraction graph $EG$, where $v \in V$:

**EG0** If $R, R' \in g(v)$ and $R \subseteq R'$, then $R = R'$; moreover, if $a \in aB$ and there are no $w$ and $t$ such that $v \xrightarrow{a, t} w$, then $a \notin R$.

**EG1** $v_0 \xrightarrow{\tau} v$.

**EG2** If $v \xrightarrow{\tau} w$ and $v \xrightarrow{a, t} w'$ then $t = t'$ and $w = w'$.

**EG3** If $\delta(v) = D$, then there is $w \in V$ such that $v \xrightarrow{\tau} w$ and $\delta(w) = d$.

**EG4** If $t = \langle a_1, \ldots, a_k \rangle \in aB^*$, then there is $w$ such that $v_0 \xrightarrow{\langle a_1, \ldots, a_k \rangle \cdot \tau} w$.

When representing an extraction pattern $ep = (B, b, dom, extr, ref, inv)$, it is straightforward to deal with its last component, $\text{inv}$, which can be represented by the mapping $i$, giving for every $a \in ab$ the trace $inv(a)$. The representation of $\text{dom}$, $\text{extr}$ and $\text{ref}$ can be provided by other components of an extraction graph.

Let $EG = (B, b, V, A, v_0, g, \delta, i)$ be an extraction graph. Then $\text{Dom}_EG$ is a set of traces, and

$$ep_{EG} = (B, b, \text{dom}_EG, \text{extr}_EG, \text{ref}_EG, \text{inv}_EG)$$

is a tuple, defined in the following way.

**EG5** $\text{dom}_EG = \{ u \in aB^* \mid \exists v, t : v_0 \xrightarrow{\tau} v \}$.

**EG6** $\text{dom}_EG = \{ u \in aB^* \mid \exists v, t : v_0 \stackrel{\tau}{\rightarrow} v \land \delta(v) = d \}$.

**EG7** For every $t = \langle a_1, \ldots, a_k \rangle \in ab^*$, $\text{inv}_EG(t) = i(a_1) \circ \cdots \circ i(a_k)$.

**EG8** By EG2, for every $u \in \text{Dom}_EG$, there are unique $t$ and $v$ such that $v_0 \xrightarrow{\tau} v$. We then define $\text{extr}_EG(u) = t$ and $\text{ref}_EG(u) = \{ R \mid \exists R' \in g(v) : R : R' \subseteq R' \}$.

**Proposition 16** $ep_{EG}$ is an extraction pattern.

**Proof.** We take advantage of the intuition that each component of $ep_{EG}$ corresponds to the similarly named component of a generic extraction pattern $ep$. We proceed to show that each condition $EP_i$, for $0 \leq i \leq 4$, is met by the relevant component(s) of $ep_{EG}$.

**EP0:** Clearly, $B$ is a non-empty finite set of channels and $b$ is a channel.

**EP1:** By EG5, $\text{dom}_EG \subseteq aB^*$. Moreover, $\text{dom}_EG \neq \emptyset$ and $\text{Dom}_EG = \text{Pref}(\text{dom}_EG)$ follow from EG3.

**EP2:** Strictness of $\text{extr}_EG$ follows from $v_0 \xrightarrow{\tau} v_0$ in EN2, and monotonicity follows from EN2 and EG8. If $\text{extr}_EG(u) = t$, then $u$ is a trace in $\text{Dom}_EG$ and $t$ is a trace over $b$, by $A \subseteq V \times (aB \times ab^*) \times V$, EN1, EN2 and EG8.
EP3: \( \text{ref}_{\text{EG}} \) is defined for traces in \( \text{Dom}_{\text{EG}} \) by \( A \subseteq V \times (\alpha B \times \alpha^*) \times V \), \( \text{EN1, EN2} \) and \( \text{EG8} \). Let \( u \in \text{Dom}_{\text{EG}} \). \( \text{ref}_{\text{EG}}(u) \) is subset-closed by \( \text{EG8} \). Moreover, \( \text{ref}_{\text{EG}}(u) \) is non-empty and contains only proper subsets of \( \alpha B \), by definition of \( \varrho \) and \( \text{EG8} \). The second part of EP3 follows from \( \text{EG8} \) and the second part of \( \text{EG0} \).

EP4: This follows from \( \text{EG4} \) and \( \text{EG7} \).

Conversely, one can easily see that for every extraction pattern \( \text{ep} \) there is an extraction graph \( \text{EG} \) such that \( \text{ep} = \text{ep}_{\text{EG}} \). From the point of view of practical implementation, however, we will be interested only in those extraction graphs which are finite, i.e., have a finite number of nodes \( V \).

**Proposition 17** If \( V \) is finite then \( (V, A) \) is a finite graph.

*Proof.* Follows from \( V \) being finite, \( \alpha B \) being finite and \( \text{EG2} \).

In the rest of the paper, we will assume that we can extract at most one event out of any event over the source channels, i.e., for every extraction mapping and a trace \( t \circ \langle a \rangle \) in its domain,

\[
|\text{extr}(t \circ \langle a \rangle)| \leq 1 + |\text{extr}(t)|.
\]

(10)

In terms of extraction graphs, this means that \( |t| \leq 1 \), for every labelled arc \( v \xrightarrow{a} w \). The above assumption has been introduced in order to simplify the presentation; it could be omitted at the cost of slightly complicating (but not losing) the subsequent results.

### 4.5 Examples of extraction graphs

The identity extraction pattern \( \text{id}_c \) defined in section 3.2 for a channel \( c \) with \( \mu c = \{0, 1\} \), can be represented by the identity extraction graph \( \text{EG}_c \), shown in figure 9. The extraction pattern \( \text{ep}_{\text{twice}} \), also defined in section 3.2, can be represented by the extraction graph \( \text{EG}_{\text{twice}} \), shown in figure 10. It is easy to check that \( \text{ep}_{\text{EG}_c} = \text{id}_c \) and \( \text{ep}_{\text{EG}_{\text{twice}}} = \text{ep}_{\text{twice}} \).

![Fig.9. Extraction graph \( \text{EG}_c \)](image)

### 5 Unambiguous CTS

Extraction patterns and so extraction graphs are defined for a channel in a base process \( P \) and a channel or channels in an implementation process \( Q \). As a result, more than one EG will usually be required to interpret the behaviour of the implementation process \( Q \) as a whole. Moreover, it is possible that the CTS representing \( Q \) will be ambiguous (in the sense explained below) with respect to interpretation in terms of the EGs.
Modelling and Verification of Processes in the Event of Interface Difference

We now discuss how to verify trace-based implementation conditions, such as IR1. Let us consider a base process $Buf$ modelling a buffer of capacity one, with input channel $d$ and output channel $e$, defined in section 3.1. Recall that it can be modelled by the communicating transition system $CTS_{buf}$ shown in figure 6. We also consider two extraction patterns, $ep_1$ and $ep_2$, given by the extraction graphs $EG_1$ and $EG_2$, i.e., $ep_i = ep_{EG_i}$ (for $i = 1, 2$). The first extraction graph, $EG_1 = EG_{d_r}$, is the identity extraction graph defined similarly as in section 9. The second one, over the sources $\{r, s\}$ and target $e$, is given in figure 11.

We would like to verify the implementation conditions, with respect to $ep_1$ and $ep_2$, for a process $Q_0$ such that $in Q_0 = \{d\}$ and $out Q_0 = \{r, s\}$, and whose behaviour is described by the communicating transition system $CTS_{Q}$ shown in figure 12, i.e., $Q_0 = P_{CTS_Q}$. Although it is not difficult to see that $Q_0 \in Impl_3 (Buf, ep_1, ep_2)$, it may not be clear what needs to be done to verify this using only the representations of $Q_0$, $Buf$, $ep_1$ and $ep_2$, given in the form of the appropriate communicating transition systems and extraction graphs. In particular, suppose that we want to verify that IR1(c) holds,
A possible attempt would be to replace each of the arc annotations in $CTS_Q$ by the 'extracted' string given by the corresponding extraction pattern. This could be done for all the actions except $slack$ from which we can extract either $\langle e!0 \rangle$ or $\langle e!1 \rangle$, depending on the previous actions executed by the process. Thus $CTS_Q$ is an ambiguous representation of $Q_0$ given the extraction patterns $ep_1$ and $ep_2$. A solution we propose is to remove this ambiguity, by suitably modifying $CTS_Q$.

![Diagram of CTS_Q](image)

**Fig. 12.** An implementation of a buffer of capacity one

We split the node $x_3$ of $CTS_Q$ and separate the two arcs incoming to it, obtaining $CTS'_Q$ shown in figure 13(a). We can now unambiguously interpret each of the arc annotations, which leads to the graph $G$ shown in figure 13(b). To verify that IR1(c) holds, it now suffices to check that the traces generated by $G$ are also generated by $CTS_{buf}$.

![Diagram of CTS_Q and G](image)

**Fig. 13.** Disambiguating $CTS_Q$

The following algorithm generates an equivalent unambiguous CTS, from a given CTS and a set of extraction graphs.

**Algorithm 2.** For $i = 1, \ldots, m + n$, let

$$EG_i = (B_i, b_i, V_i, A_i, v_{0i}, q_i, \delta_i, \iota_i)$$

be extraction graphs such that the $B_i$'s are mutually disjoint and the $b_i$'s distinct. Moreover, let $CTS = (V, C, D, A, v_0)$ be a communicating transition system such that

$$C = B_1 \cup \ldots \cup B_m \quad \text{and} \quad D = B_{m+1} \cup \ldots \cup B_{m+n}.$$
The algorithm generates a communicating transition system $CTS^u$ in two steps.

**Step 1:** We generate a labelled directed graph $G$, with the nodes $V \times V_1 \times \cdots \times V_p$, as follows. Let $q = (v, v_1, \ldots, v_n)$ be a node in $G$. The arcs outgoing from $q$ are derived from those outgoing from $v$; for each arc $v \rightarrow w$ in $CTS$ we proceed according to exactly one of the following four cases.

1. $a = \tau$. Then we add a transition $q \rightarrow (w, v_1, \ldots, v_n)$.
2. $a \neq \tau$ and there is an arc $v_i \rightarrow w_i$ in $EG_i$, for some $i \geq 1$.\(^4\) Then we add a transition $q \rightarrow (w, v_1, \ldots, v_n)$ where $w_j = v_i$, for all $j \neq i$. Moreover, we denote $\text{extr}(q, a) = \tau$ if $l = \langle \rangle$, and $\text{extr}(q, a) = b$ if $l = \langle b \rangle$. Note that $\text{extr}(q, a)$ is well defined by $EG_2$.
3. $a \in \alpha C$ and there is no arc $v_i \rightarrow w_i$, for any $i \geq 1$. Then we do nothing.
4. $a \in \alpha D$ and there is no arc $v_i \rightarrow w_i$, for any $i \geq 1$. Then we mark permanently $q$ as an unfinished node (all the nodes are assumed to be finished at the beginning).

**Step 2:** From the graph $G$ we obtain a communicating transition system $CTS^u$ with the same channels as $CTS$, by taking $q_0 = (v_0, v_{01}, \ldots, v_{0n})$ as the initial node, and then adding all the nodes reachable from $q_0$, together with all the interconnecting arcs. If any of the reachable nodes is marked as unfinished, we reject $CTS^u$ (since this means that the traces generated by $Q = P_{CTS^u}$ do not satisfy the condition IR1(a), where each $ep_i$ is generated by $EG_i$, see the proof of proposition 24).

The above algorithm will be executed on the CTS representation of the implementation process $Q$. The result is denoted $CTS^u_Q$; its main characteristic is that the definition of the nodes allows the unambiguous interpretation of the arc labels through the extraction mappings (see proposition 19). In addition, $\tau P_{CTS^u_Q} = \tau Q \cap \text{Dom}_{ep^u}$ (see proposition 25).

In practice, one can avoid generating the whole graph $G$, by performing a depth first search starting from the initial node $q_0$. Then only the nodes of $CTS^u$ will be visited. Later, for a node $q = (w_0, w_1, \ldots, w_n)$ of $CTS^u$ and $0 \leq i \leq m + n$, we will denote $q^{(i)} = w_i$.

The graph $G$ for the example in figure 11 is shown in figure 14(a), where the * indicates an unfinished node. After restricting ourselves only to the relevant subgraph (comprising nodes reachable from the initial one), we obtain the graph, shown in figure 14(b), which is isomorphic to $CTS^u_Q$ obtained informally before. Note that $\text{extr}((x_3, v_0, w_1), \text{stack}) = e^{10}$ and $\text{extr}((x_1, v_0, w_0), \times 0) = \langle \rangle$.

Let $ep_i = (B_i, b_i, dom_i, \text{extr}_i, ref_i, inv_i)$ be the extraction pattern generated by $EG_i$; i.e., $ep_i = ep_{EG_i}$. Moreover, $ep = \{ep_1, \ldots, ep_n\}$ and $ep' = \{ep_{m+1}, \ldots, ep_{m+n}\}$. We now provide some basic properties of $CTS^u$.

**Proposition 18** If $q_0 \xrightarrow{t} q$ then $v_{hi} \xrightarrow{\{B_i, \text{extr}_i(B_i)\}} q^{(i)}$, for every $i \geq 1$.

**Proof.** We proceed by induction on the length of a path along which $t$ is generated. In the base case, we have

$$q_0 \rightarrow q = q_0$$ and $$v_{hi} \xrightarrow{\langle \rangle} q^{(i)} = v_{hi}.$$ Hence the result holds since $\text{extr}_i$ is strict. In the induction step, suppose that

$$q_0 \xrightarrow{t} q \rightarrow q'$$ and $$v_{hi} \xrightarrow{\{B_i, \text{extr}_i(B_i)\}} q^{(i)}.$$\(^4\) There can only be one such $EG_i$, since the $B_i$’s are mutually disjoint.
Then immediately. In the induction step, assume that the result holds for \( v \).

\[ \text{We then consider two cases.} \]

Case 1: \( a \notin \alpha_B \). Then, by algorithm 2(1,2), \( q^{(i)} = q^{(i)} \). Moreover, \( \langle a \rangle |B_i = \emptyset \), and so we have \( v_i \xrightarrow{(t \circ \langle a \rangle)[B_i]} q^{(i)} \).

Case 2: \( a \in \alpha_B \). Then, by algorithm 2(2), \( q^{(i)} \xrightarrow{a^* \tau'} q^{(i)} \) and \( \langle a \rangle |B_i = \langle a \rangle \). Hence \( v_i \xrightarrow{(t \circ \langle a \rangle)[B_i]} q^{(i)} \), by \( \text{extr}_i(t \circ \langle a \rangle)[B_i] = \text{extr}_i(t[B_i] \circ \langle a \rangle) = \text{extr}_i(t[B_i] \circ \tau') \).

\[ \square \]

We may now state precisely why \( CTS^u \) can be regarded as an unambiguous transition system.

**Proposition 19** Let \( q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots \xrightarrow{a_k} q_k \) in \( CTS^u \) and \( t = \langle a_1 \rangle \circ \ldots \circ \langle a_k \rangle \). Then \( t \in \text{Dom}_{ePu} \) and \( \text{extr}_{ePu}(t) = u_1 \circ \ldots \circ u_k \) where \( u_i = \emptyset \) if \( a_i = \tau \) and \( u_i = \langle \text{extr}(q_{i-1}, a_i) \rangle \) if \( a_i \neq \tau \), for every \( i \leq k \).

**Proof.** We proceed by induction on \( k \). In the base case, \( k = 0 \), the result follows immediately. In the induction step, assume that the result holds for \( k \) and \( q_k \xrightarrow{a_k+1} q_{k+1} \). We then consider two cases.

Case 1: \( a_{k+1} = \tau \). Then \( t \circ \langle a_{k+1} \rangle = t \) and so the result still holds.

Case 2: \( a_{k+1} \notin \alpha_B \). Then, by algorithm 2(2), there is an arc \( q_k \xrightarrow{\tau' \tau'} q_{k+1} \) in \( EG_i \).

By proposition 18, \( v_i \xrightarrow{\text{extr}_i(t[B_i])} q_k \) and so, by EG5 and EP6, we have \( t \circ \langle a_{k+1} \rangle \in \text{Dom}_{ePu} \). Moreover, \( \langle \text{extr}(q_k, a_{k+1}) \rangle = u \) and so the second part also holds. \[ \square \]

We can also directly relate various paths in \( CTS \) and \( CTS^u \).
Proposition 20 If \( q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} q_k \) in \( CTS^u \) then \( q_0^{(i)} \xrightarrow{a_1^{(i)}} q_1^{(i)} \xrightarrow{a_2^{(i)}} \cdots \xrightarrow{a_k^{(i)}} q_k^{(i)} \) in \( CTS \).

Proof. Follows by induction on \( k \), directly from algorithm 2(1,2).

Proposition 21 If \( v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} v_k \) in \( CTS \) and \( \langle q_1 \rangle \circ \cdots \circ \langle q_k \rangle \in Dom_{epu}ep \), then there is exactly one derivation \( q_0^{(i)} \xrightarrow{a_1^{(i)}} q_1^{(i)} \xrightarrow{a_2^{(i)}} \cdots \xrightarrow{a_k^{(i)}} q_k^{(i)} \) in \( CTS^u \) such that \( q_i^{(i)} = v_i \), for every \( i \leq k \).

Proof. We proceed by induction on \( k \). In the base case \( k = 0 \), and the result follows immediately. In the inductive step, assume that the result holds for \( k \), and that \( v_k \xrightarrow{a_k^{(i)}} v_{k+1} \) is such that \( \langle a_1 \rangle \circ \cdots \circ \langle a_k \rangle \circ \langle a_{k+1} \rangle \in Dom_{epu}ep \). We consider two cases.

Case 1: \( a_{k+1} = \tau \). Then, by algorithm 2(1), \( q_k \xrightarrow{a} q_{k+1} \) where \( q_{k+1}^{(i)} = v_{k+1} \), for exactly one \( q_{k+1} \) (note that \( q_{k+1}^{(i)} = q_{k+1}^{(i)} \), for \( i \geq 1 \)).

Case 2: \( a_{k+1} \in \alpha B_k \). From \( \langle a_1 \rangle \circ \cdots \circ \langle a_k \rangle \circ \langle a_{k+1} \rangle \in Dom_{epu}ep \) and proposition 18, it follows that we can apply algorithm 2(2). Then we proceed similarly as in Case 1.

Proposition 22 If \( q \) is a node in \( CTS^u \) then

\[
\text{en}(q) = \text{en}(q^{(i)}) - \{ a \in \alpha CTS \mid \text{extr}(q, a) \text{ is undefined} \}.
\]

Moreover, \( CTS^u \) is rejected if and only if there exists a node \( q \) in \( CTS^u \) such that \( \{ \text{en}(q^{(i)}) = \text{en}(q) \} \cap \alpha D \neq \emptyset \).

Proof. The first part follows directly from Step 1 of algorithm 2. The second part follows from the rejection of \( CTS^u \) depends on marking at least one node as un\( \emptyset \)inished. The latter is equivalent, by algorithm 2(4), to the existence of a node \( q \) in \( CTS^u \) such that \( \{ \text{en}(q^{(i)}) = \text{en}(q) \} \cap \alpha D \neq \emptyset \).

6 Graph representation of implementation relations

In this section, we will transfer the implementation conditions IR1–IR5 formulated in terms of the denotational semantics of CSP, into equivalent conditions expressed in terms of communicating transition systems and extraction graphs. The latter will provide, in section 7, suitable basis for verification algorithms. Below we list some general assumptions which will be used throughout this and the next section.

- \( P, Q, ep_i \) (for \( i = 1, \ldots, m + n \)) \( ep \) and \( ep' \) are as in section 3.3.
- \( CTS_P \) and \( CTS_Q \) are communicating transition systems representing \( P \) and \( Q \) respectively; i.e., \( P = P_{CTS_P} \) and \( Q = P_{CTS_Q} \).
- For \( i = 1, \ldots, m + n \), \( EG_i \) is an extraction graph with the initial node \( v_0 \) representing \( ep_i \); i.e., \( ep_{EG_i} = ep_i \).
- \( P_{det} \) is the normalised version of \( CTS_P \). We will use \( \kappa_P \) to denote the mapping defined as in (8) for the nodes of \( P_{det} \) and denote the initial state of \( P_{det} \) by \( p_0 \).
- \( CTS^u_Q \) is a disambiguated version of \( CTS_Q \) w.r.t. extraction graphs \( EG_i \) (see algorithm 2). We will denote the initial state of \( CTS^u_Q \) by \( q_0^u \).
- The process generated by \( CTS^u_Q \) will be denoted by \( \hat{Q} \), i.e., \( \hat{Q} = P_{CTS^u_Q} \).
- \( Q_{det} \) is the normalised version of \( CTS^u_Q \). We will use \( \kappa_Q \) to denote the mapping defined as in (8) for the nodes of \( Q_{det} \) and denote the initial state of \( Q_{det} \) by \( q_0^u \).

It may be observed (see proposition 23 below) that if \( v \) is a node of \( Q_{det} \) and \( q, r \in v \) then, for all \( 1 \leq i \leq m + n, q_i = r_i \). We can therefore use \( r_i \) to denote \( q_i \), and \( \text{extr}(v, a) = \text{extr}(q, a) \) whenever the latter is defined.
Proposition 23 If \( v \in V_{Q_{det}} \) and \( q, r \in v \) then, for all \( 1 \leq i \leq m + n, q^{(i)} = r^{(i)} \)

Note: Thus \( extr(v, a) \) is well-defined with respect to \( Q_{det} \).

Proof. To the contrary, suppose that \( q^{(i)} \neq r^{(i)} \). Then, by proposition 18, for every derivation \( q^{(i)}_i \xrightarrow{[P_i, extr_i(P_i)]} q \) (and there is at least one such derivation) it is the case that \( v_{[P_i, extr_i(P_i)]}^{(i)} q^{(i)} \in EG_i \). Moreover, again by proposition 18 and \( EG_i \) being deterministic and without \( \tau \)-labels, there is no derivation \( q^{(i)}_i \xrightarrow{r} r \). It then follows, from the definition of normalisation algorithm, that \( q \) and \( r \) will not be put in the same node of \( Q_{det} \). \( \square \)

We now proceed with a systematic re-evaluation of the implementation conditions \( IR1-IR5 \). We first obtain that testing for \( IR1(b) \) amounts to checking for the presence of \( \tau \)-loops in the graph of \( CTS_Q^{\mu} \), and if there is no such loop, then testing for \( IR1(a) \) is done while generating \( CTS_Q^{\nu} \).

Proposition 24 \( Q \) satisfies \( IR1(a,b) \) if and only if \( CTS_Q^{\mu} \) has been successfully generated and there are no nodes \( v_1, \ldots, v_k (k \geq 2) \) in \( CTS_Q^{\mu} \) such that \( v_1 \xrightarrow{\tau} v_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} v_k = v_1 \).

Proof. By \( Q = P_{CTS_Q} \) and proposition 11, and also by proceeding similarly as in the proof of proposition 11, one can show that \( Q \) satisfies \( IR1(a,b) \) if and only if, in \( CTS_Q \), there is neither a derivation:

(i) \( r_0 \xrightarrow{a_1} r_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} r_{k+1} \xrightarrow{\tau} \cdots \xrightarrow{\tau} r_{k+l} = r_k \) \((l \geq 1)\)

such that \( \langle a_1 \rangle \circ \cdots \circ \langle a_k \rangle \in Dom_{rep} \), nor a derivation:

(ii) \( s_0 \xrightarrow{b_1} s_1 \xrightarrow{b_2} \cdots \xrightarrow{b_n} s_b \xrightarrow{\tau} s \)

such that \( \langle b_1 \rangle \circ \cdots \circ \langle b_n \rangle \in Dom_{rep} \) and \( \langle b_1 \rangle \circ \cdots \circ \langle b_n \rangle \circ \langle b \rangle \notin Dom_{rep} \) and \( v \in out Q \).

\((\Rightarrow)\) The absence of \( \tau \)-loops follows from the absence of derivations (i), and propositions 19 and 20. That \( CTS_Q^{\mu} \) is successfully generated follows from the absence of derivations (ii), and propositions 18, 19, 20 and 22, and finally from algorithm 2(4).

\((\Leftarrow)\) By propositions 18, 21 and 22, the existence of a derivation (ii) means that \( CTS_Q^{\mu} \) is rejected in Step 2 of algorithm 2. Moreover, by propositions 21, the existence of a derivation (i) implies that \( CTS_Q^{\mu} \) contains a \( \tau \)-loop. \( \square \)

From now on, we will assume that \( CTS_Q^{\mu} \) has been successfully generated and does not contain any \( \tau \)-loops, and so \( IR1(a,b) \) hold. In such a case, as the next result shows, \( CTS_Q^{\mu} \) generates a process which can be used to test for all implementation relations in place of \( Q \).

Proposition 25 For each verification condition \( IR1-IR5 \) (other than \( IR1(a,b) \)), it is the case that \( Q \) satisfies the condition if and only if the same is true of \( Q \). Moreover, \( \tau Q = \tau Q \cap Dom_{rep} \).

Note: Thus, for \( i = 1, 2, 3, Q \in Impl_i(P, ep, ep') \) if and only if \( Q \in Impl_i(P, ep, ep') \).

Proof. Under the assumption that \( IR1(a,b) \) hold and by proposition 24, \( \tau Q = \tau Q \cap Dom_{rep} \) follows from propositions 19, 20 and 21. In turn, \( \tau Q = \tau Q \cap Dom_{rep} \) implies the result w.r.t. the conditions \( IR1(c), IR2 \) and \( IR4 \).

There are two other issues which need to be discussed. The first is the satisfaction of the condition related to an input channel \( b_i \) in the formulation of \( IR3 \). Let \( v \) be a node
in \( CTS_Q \) reachable by a trace \( t \in Dom_{epuwp} \). By proposition 21, there is a node \( q \) in \( CTS_Q \) reachable by \( t \) such that \( q^{(0)} = v \) and so, by proposition 22, \( en(v) = \kappa_Q(q) \subseteq Z \) (where \( \oplus \) denotes disjoint union), for some \( Z \subseteq Q \). Moreover, \( t \circ \langle a \rangle \notin Dom_{epuwp} \), for every \( a \in Z \). Hence, by the second part of EP3, if \( en(q) \cap \alpha_B = \emptyset \), then \( \kappa_Q(q) \subseteq \alpha_B \cup \emptyset \subseteq Z \cap \alpha_B \equiv \emptyset \). As we have already seen, IR1(a,b) can be checked directly using \( CTS_Q \). In dealing with the remaining implementation conditions, we assume that IR1(a,b) hold, and use \( Q_{det} \), which is a normalised CTS derived from \( CTS_Q \). Note that, by propositions 24 and 25, we may assume that \( \hat{Q} \) is the implementation process such that

\[
\delta \hat{Q} = \emptyset \quad \text{and} \quad \tau \hat{Q} \subseteq Dom_{epuwp}.
\]

(12)

Proposition 26 \((t, R) \in \partial \hat{Q} \) if and only if there exist \( q_0 = q \) and \( \hat{A} \in \kappa_Q(q) \) such that \( R \subseteq \alpha \hat{Q} - A \).

Proof. Follows from proposition 11, (12), and the definitions of \( \partial \) and \( \kappa_Q \). \( \Box \)

Corollary 27 \((t, R) \in \max \partial \hat{Q} \) if and only if there exist \( q_0 = q \) and \( \hat{A} \in \kappa_Q(q) \) such that \( R \subseteq \alpha \hat{Q} - A \).

A relation \( sim_{extr} \subseteq V_{Q_{det}} \times V_{P_{det}} \) is an extr-simulation for \( Q_{det} \) and \( P_{det} \) if \( (q_0, p_0) \in sim_{extr} \) and, for every \( (q, p) \in sim_{extr} \),

\[
q \xrightarrow{a} q' \quad \Rightarrow \quad \exists (q', p') \in sim_{extr} : p \xrightarrow{(extr(q, a))} p'.
\]

(13)

Proposition 28 \( \hat{Q} \) (and so \( Q \)) satisfies IR1(c) if and only if there exists an extr-simulation for \( Q_{det} \) and \( P_{det} \).

Proof. \((\Rightarrow)\) By propositions 12 and 26, IR1(c) amounts to saying that for every derivation \( q_0 = q \) in \( Q_{det} \) there exists a derivation \( p_0 \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p \) in \( P_{det} \). Let \( tr_{\hat{Q}, a} \subseteq V_{Q_{det}} \times V_{P_{det}} \) be a relation such that \((q, p) \in tr_{\hat{Q}, a} \) if \( q_0 = q \) and \( p_0 \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p \), for some \( t \). We may now write IR1(c) as \( \forall q \in V_{Q_{det}} \exists p : (q, p) \in tr_{\hat{Q}, a} \).

We will show that \( tr_{\hat{Q}, a} \) is an extr-simulation for \( Q_{det} \) and \( P_{det} \). We first note that \((q_0, p_0) \in tr_{\hat{Q}, a} \) by the strictness of extr. If \( q_0 = q \) and \( (q, p) \in tr_{\hat{Q}, a} \), then \( (q', p') \in tr_{\hat{Q}, a} \), where \( p' \) is such that \( p_0 \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p' \), by the monotonicity of extr, \( extr_{\hat{Q}, a} \) is an extr-simulation for \( Q_{det} \) and \( P_{det} \).

\((\Leftarrow)\) Let \( sim_{extr} \) be an extr-simulation for \( Q_{det} \) and \( P_{det} \). We proceed by induction on the length of traces, showing that if \( q_0 = q \) then there is a derivation \( p_0 \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p \) such that \((q, p) \in sim_{extr} \).

In the base case \( t = \emptyset \), so \( q_0 = q \) and \( p_0 \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p_0 \) and \((q_0, p_0) \in sim_{extr} \). In the inductive step, we assume \( q_0 = q \) and \( p_0 \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p \) and \((q, p) \in sim_{extr} \). Then there is \( (q', p') \in sim_{extr} \) such that \( p \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p' \), and so we have \( q_0 = q' \) and \( p \xrightarrow{(extr_{\hat{Q}, a}(\langle a \rangle))} p' \) by EN2 and the definition of extr \((q, a)\). \( \Box \)
We also denote from a finite input, preserves traces, and finally from the finiteness of if a finite CTS follows from (11) above and EP5. Moreover, IR3(b) follows from (2) above and CSP2 for communicating transition systems and extraction graphs. For all J. Burton, M. Koutny and G. Pappalardo there exists \( b \) if and only if for every derivation \( w \rightarrow_{\tau} \ldots \rightarrow_{\tau} v_k = v_1 \) and \( \text{extr}(v_i, a_i) = \tau \), for all \( i \leq k \).

**Proposition 29** \( \hat{Q} \) (and so \( Q \)) satisfies IR2 if and only if there are no nodes \( v_1, \ldots, v_k \) (\( k \geq 2 \)) in \( Q_{\text{det}} \) such that \( v_1 \overset{b_1}{\rightarrow} v_2 \overset{b_2}{\rightarrow} \ldots \overset{b_{l-1}}{\rightarrow} v_k = v_1 \) and \( \text{extr}(v_i, a_i) = \tau \), for all \( i \leq l \).

**Proof.** We first observe that, by (12) and the finiteness of \( CTS_Q^w \), we may state that IR2 is met if and only if there are no nodes \( w_1, \ldots, w_l \) (\( l \geq 2 \)) in \( CTS_Q^w \) such that \( w_1 \overset{b_1}{\rightarrow} w_2 \overset{b_2}{\rightarrow} \ldots \overset{b_{l-1}}{\rightarrow} w_k = w_1 \) and \( \text{extr}(w_i, b_i) = \tau \), for all \( i \leq l \). Thus IR2 is met if and only if for every derivation \( w_1 \overset{b_1}{\rightarrow} w_2 \overset{b_2}{\rightarrow} \ldots \overset{b_{l-1}}{\rightarrow} w_k = w_1 \) in \( CTS_Q^w \), there exists \( b_i \neq \tau \) such that \( \text{extr}(w_i, b_i) \neq \tau \). The proof then follows from the fact that \( CTS_Q \) contains no \( \tau \)-loops, and that the process of normalisation returns a finite result from a finite input, preserves traces, and finally from the finiteness of \( CTS_Q \), note that if a finite \( CTS \) generates an infinite trace \( t \circ w \circ w \circ \ldots \) then there is a cycle in \( CTS \) generating a trace of the form \( w \circ w \circ w \circ \ldots \). \( \square \)

To prepare the ground for testing of IR3, we re-phrase it in terms of maximal failures of \( \hat{Q} \). For every \( (t, R) \in \hat{Q} \), we denote

\[ C_{t, R} = \{ b_k \in \text{in} \mid \alpha B_k - R \in \text{ref}_{\text{epu}}(t[B_i]) \} \cup \{ b_k \in \text{out} \mid \alpha B_k \cap R \notin \text{ref}_{\text{epu}}(t[B_i]) \}. \]

We also denote \( C_t = \{ C_{t, R} \mid (t, R) \in \max \hat{Q} \} \) and \( \overline{C}_t = \bigcup \{ B \mid B \in C_t \} \).

**Proposition 30** \( \hat{Q} \) (and so \( Q \)) satisfies IR3 if and only if for every \( t \in \tau \hat{Q} \), the following hold.

1. If \( b_k \in \overline{C}_t \) then \( t[B_k] \in dom_t \).
2. If \( t \in \text{dom}_{\text{epu}} \) then for every \( B \in C_t \) there exists \( R \) such that \( (\text{extr}_{\text{epu}}(t), R) \in \max \hat{Q} \) and \( \alpha B \subseteq R \).

**Proof.** (\( \Leftarrow \)) We first observe that, by EP3, if \( R' \subseteq R \) then \( C_{t, R'} \subseteq C_{t, R} \). Hence IR3(a) follows from (1) above and EP5. Moreover, IR3(b) follows from (2) above and CSP2 for \( P \).

(\( \Rightarrow \)) The proof follows by EP5 and the fact that (1) and (2) above deal with a special case of the original condition. \( \square \)

We now introduce notions corresponding to \( C_{t, R}, C_t \) and \( \overline{C}_t \) in the domain of communicating transition systems and extraction graphs. For all \( q \in V_{\hat{Q}_{\text{det}}} \) and \( A \in \kappa Q(q) \), we denote

\[ C_{q, A} = \{ b_k \in \text{in} \mid \exists R' \in \mathcal{B}(q^{(i)}) : \alpha B_k \cap A \subseteq R' \} \cup \{ b_k \in \text{out} \mid \exists R' \in \mathcal{B}(q^{(i)}) : \alpha B_k - A \subseteq R' \}. \]

We also denote \( C_q = \{ C_{q, A} \mid A \in \kappa Q(q) \} \) and \( \overline{C}_q = \bigcup \{ B \mid B \in C_q \} \).

**Proposition 31** If \( \hat{q} \overset{\tau}{\Rightarrow} q \) then \( C_q = C_t \). Moreover.

1. \( C_{q, A} = C_{t, \alpha Q - A} \) for every \( A \in \kappa Q(q) \).
2. \( C_{t, R} = C_{q, \alpha Q - R} \) for every \( (t, R) \in \max \hat{Q} \).
Proof. The first part follows immediately from (1) and (2), so we only show these.

(1) Let \( R' \in g_i(q^{(i)}) \) is such that \( a_B \cap A \subseteq R' \) then by EG8 and propositions 16 and 18, we have \( a_B \cap A \in ref_i(t[B]) \). By corollary 27, \((t, R) \in \max \phi \) where \( R \models aQ - A \). For such an \( R \), we have \( a_B \cap A \notin \ref_i(t[B]) \).

By similar reasoning, we infer that if there does not exist \( R' \in g_i(q^{(i)}) \) such that \( a_B - A \subseteq R' \) then \( a_B \cap A \notin \ref_i(t[B]) \), where \( (t, R) \in \max \phi \) and \( R \models aQ - A \).

(2) Let \((t, R) \in \max \phi \). By corollary 27, there exists \( A \in \kappa Q \) such that \( R \models aQ - A \). Thus, by EG8 and proposition 18, \( a_B - R \in \ref_i(t[B]) \) implies \( a_B \cap A \subseteq R' \) where \( R' \in g_i(q^{(i)}) \). By similar reasoning, \( a_B \cap R \notin \ref_i(t[B]) \) implies that there does not exist an \( R' \) such that \( R' \in g_i(q^{(i)}) \) and \( a_B - A \subseteq R' \).

Proposition 32 \( \hat{Q} \) (and so \( Q \)) satisfies IR3 if and only if, for every \( q \in V_{Q_{aut}} \), the following hold.

1. If \( b_i \in \bar{C}_q \) then \( \delta_i(q^{(i)}) = d \).
2. If \( (q, p) \in \text{sim}_{extr} \) and \( \delta_i(q^{(i)}) = \ldots = \delta_i(q^{(m+n)}) = d \) then, for every \( B \in C_q \), there is \( A \in \kappa P(p) \) satisfying \( A B \cap A = \emptyset \).

Proof. (\( \Rightarrow \)) Let \( q \in V_{Q_{aut}} \). By proposition 26, \( q_0 \overset{t}{\rightarrow} q \), for some \( t \in \tau \hat{Q} \).

(1) By proposition 31, \( C_q = C_t \) and so \( \bar{C}_q = \bar{C}_t \). The proof follows by EG6 and propositions 18 and 30(1).

(2) Again by proposition 31, \( C_q = C_t \). By propositions 16 and 18, EG6, and EP5, we have that \( \delta_i(q^{(i)}) = d \) (for \( 1 \leq i \leq m + n \)) implies \( t \in \text{dom}_{pQ} \). By proposition 28 and the fact that \( \sim_{min} \) is the smallest extr-simulation, \( (q, p) \in \sim_{min} \) implies \( p_0 \overset{\text{extr}_Q(t)}{\Rightarrow} p \). Then we apply corollary 13 and proposition 30(2).

(\( \Leftarrow \)) It suffices to show that the two conditions in proposition 30 are satisfied. Let \( t \in \tau \hat{Q} \). By proposition 26, there exists a derivation \( q_0 \overset{t}{\Rightarrow} q \). Then proposition 30(1) follows from (1) above, proposition 18, \( \bar{C}_q = \bar{C}_t \) (follows from proposition 31), and EG6.

Proposition 30(2) can be shown thus. By EP5, EG6 and proposition 18, we have that \( t \in \text{dom}_{pQ} \) implies \( \delta_i(q^{(i)}) = d \) (for \( 1 \leq i \leq m + n \)). By corollary 13 and proposition 28, there exist \( p \in V_{P_{det}} \) such that \( p_0 \overset{\text{extr}_Q(t)}{\Rightarrow} p \) and \( (q, p) \in \sim_{min} \).

Let \( B \in C_t \). By \( C_q = C_t \), there is \( A \in \kappa P(p) \) such that \( A B \cap A = \emptyset \). Thus, by corollary 13, \( (\text{extr}_Q(t), R) \in \max \phi P \) and \( A \subseteq R \), where \( R = aP - A \).

We now turn to the two remaining implementation conditions. Since the \( \text{inv}_i's \) are homomorphisms, they can interpret the arc labels directly, without taking into account how a particular node has been reached. However, the situation is complicated by the fact that \( \text{inv}_i(a) \) will usually be a non-singleton trace.

A relation \( \sim_{inv} \subseteq V_{P_{aut}} \times V_{Q_{aut}} \) is an inv-simulation for \( P_{det} \) and \( Q_{det} \) if \( (p, q) \in \sim_{inv} \) and, for every pair \( (p, q) \in \sim_{inv} \),

\[
p \overset{a}{\Rightarrow} p' \Rightarrow \exists (p', q') \in \sim_{inv} : q = \overset{\text{inv}_{Qaut}(t)}{\Rightarrow} q'.
\]

(14)

Proposition 33 \( \hat{Q} \) (and so \( Q \)) satisfies IR4 if and only if there exists an inv-simulation.

Proof. IR4 holds if and only if \( \text{inv}_{pQ}(\tau P) \subseteq \tau \hat{Q} \). Thus, by preservation of traces by the process of normalisation, we may work with \( Q_{det} \) and \( P_{det} \).

(\( \Rightarrow \)) By propositions 12 and 26, we may write IR4 as \( p_0 \overset{t}{\Rightarrow} p \) implies \( q_0 \overset{\text{inv}_{Qaut}(t)}{\Rightarrow} q \), for some \( q \in V_{Q_{aut}} \). Let \( tr_{incl} \subseteq V_{P_{aut}} \times V_{Q_{aut}} \) be a relation such that \( (p, q) \in tr_{incl} \)
if \( p_1 \overset{t}{\Rightarrow} p \) and \( q_0 \overset{\text{inv}_{\mu \nu}(t)}{\Rightarrow} q \), for some \( t \in \tau P \). We will show that \( \tau_{\text{incl}} \) is an inv-simulation.

We first note that \((p_0, q_0) \in \tau_{\text{incl}} \) since \( \text{inv}(\emptyset) = \emptyset \). Suppose now that \( p_0 \overset{t}{\Rightarrow} p \overset{a}{\Rightarrow} p' \) and \( q_0 \overset{\text{inv}_{\mu \nu}(t)}{\Rightarrow} q \) and \((p, q) \in \tau_{\text{incl}}\). Then, since IR4 is met, \( \text{inv} \) is a homomorphism and by the determinism of \( Q_{\text{det}} \), there is \( q' \) such that \( q_0 \overset{\text{inv}_{\mu \nu}(t)}{\Rightarrow} q \overset{\text{inv}_{\mu \nu}(a)}{\Rightarrow} q' \). Thus \((p', q') \in \tau_{\text{incl}}\).

\((\Leftarrow) \) Let \( \text{sim}_{\text{inv}} \) be an inv-simulation for \( P_{\text{det}} \) and \( Q_{\text{det}} \). We proceed by induction on the length of traces, showing that if \( p_0 \overset{t}{\Rightarrow} p \) then there is a derivation \( q_0 \overset{\text{inv}_{\mu \nu}(t)}{\Rightarrow} q \) such that \((p, q) \in \text{sim}_{\text{inv}}\).

In the base case \( t = \emptyset \), and so \( p_0 \overset{\emptyset}{\Rightarrow} p_0 \) and \( q_0 \overset{\text{inv}_{\mu \nu}(\emptyset)}{\Rightarrow} q_0 \) and \((p_0, q_0) \in \text{sim}_{\text{inv}}\).

In the inductive step, we assume that \( p_0 \overset{t}{\Rightarrow} p \overset{a}{\Rightarrow} p' \) and \( q_0 \overset{\text{inv}_{\mu \nu}(t)}{\Rightarrow} q \) and \((p, q) \in \text{sim}_{\text{inv}}\). Then there is \((p', q') \in \text{sim}_{\text{inv}}\) such that \( q_0 \overset{\text{inv}_{\mu \nu}(t)}{\Rightarrow} q' \) and so we have \( p_0 \overset{t (a)}{\Rightarrow} p' \) and \( q_0 \overset{\text{inv}_{\mu \nu}(t (a))}{\Rightarrow} q' \) since \( \text{inv} \) is a homomorphism.

In the last part of this section dealing with IR5, we will assume that \( \hat{Q} \) (and so \( Q \)) satisfies IR4. This does not result in a loss of generality as IR4 is implied by IR5. Note that, since \( Q_{\text{det}} \) is deterministic and contains no \( \tau \)-transitions, if there is at least one inv-simulation, then there exists the smallest one, \( \text{sim}_{\text{inv}} \).

To test for IR5, we first observe that it can be equivalently expressed in terms of maximal failures. For every \((t, R) \in \phi P\), we denote \( \mathcal{D}_{t, R} = \{ b_i \in \chi P \mid ab_i \subseteq R \}; \) moreover, \( \mathcal{D}_{t} = \{ \mathcal{D}_{t, R} \mid (t, R) \in \phi \mu P \} \).

**Proposition 34** \( \hat{Q} \) (and so \( Q \)) satisfies IR5 if and only if for every \( t \in \tau P \) and \( B \in \mathcal{D}_{t} \), there is \( \text{inv}_{\mu \nu}(t), R \in \max \hat{Q} \) such that \( \bigcup_{b_i \in B} \alpha B_i \subseteq R \).

Proof. \((\Rightarrow) \) Let \( B' \subseteq \chi P \) and \((t, aB') \in \phi P \). Then there is \((t', R') \in \max \mu P \) such that \( B' \subseteq B = \mathcal{D}_{t, R'} \). Hence there is \( \text{inv}_{\mu \nu}(t), R \in \max \hat{Q} \) such that \( \bigcup_{b_i \in B} \alpha B_i \subseteq R \).

Thus IR5 holds by \( B' \subseteq B \) and CSP2 for \( \hat{Q} \).

\((\Leftarrow) \) Let \( t \in \tau P \) and \( B \in \mathcal{D}_{t} \). Then, by IR5, we have that

\( \text{inv}_{\mu \nu}(t), \{ a \in \bigcup_{b_i \in B} \alpha B_i \mid \text{inv}_{\mu \nu}(t) \circ a \in \text{Dom}_{\mu \nu} \} \in \phi \hat{Q} \).

Thus, by CSP3 for \( \hat{Q} \) and \( (12), (\text{inv}_{\mu \nu}(t), \bigcup_{b_i \in B} \alpha B_i ) \in \phi \hat{Q} \). Hence there is a maximal failure \( \text{inv}_{\mu \nu}(t), R \in \max \hat{Q} \) such that \( \bigcup_{b_i \in B} \alpha B_i \subseteq R \).

We now introduce notions corresponding to \( \mathcal{D}_{t, R} \) and \( \mathcal{D}_{t} \) in the domain of communicating transition systems. For every \( p \in V_{\mu P} \) and \( A \in \chi P \), we denote \( \mathcal{D}_{p, A} = \{ b_i \in \chi P \mid ab_i \cap A = \emptyset \}; \) moreover \( \mathcal{D}_{p} = \{ \mathcal{D}_{p, A} \mid A \in \chi P \} \).

**Proposition 35** If \( p_0 \overset{t}{\Rightarrow} p \) then \( \mathcal{D}_{p} = \mathcal{D}_{t} \). Moreover,

1. \( \mathcal{D}_{p, A} = \mathcal{D}_{p, A \cap p} \) for every \( A \in \chi P(p) \).
2. \( \mathcal{D}_{p, A} = \mathcal{D}_{p, A \cap p} \) for every \((t, R) \in \max \mu P \).

Proof. The first part follows immediately from (1) and (2), so we only show these.

1. Let \( A \in \chi P(p) \). By corollary 13, \((t, R) \in \max \mu P \) where \( R = aP - A \). Moreover, \( ab_i \cap A = \emptyset \) if and only if \( ab_i \subseteq R \).

2. Let \((t, R) \in \max \mu P \). By corollary 13, there exists \( A \in \chi P(p) \) such that that \( R = aP - A \). Moreover, \( ab_i \subseteq R \) if and only if \( ab_i \cap A = \emptyset \).
Proposition 36. \( \hat{Q} \) (and so \( Q \)) satisfies IR5 if and only if for every \( (p,q) \in \text{sim}_{\text{inv}}^{\min} \), if \( B \in \mathcal{D}_p \), then there is \( A' \in \kappa_Q(q) \) such that \( \bigcup_{b_i \in B} \alpha B_i \cap A' = \emptyset \).

Proof. (\( \Rightarrow \)) Let \( (p,q) \in \text{sim}_{\text{inv}}^{\min} \) and \( B \in \mathcal{D}_p \). Then \( p \xrightarrow{\tau} p \) and \( q \xrightarrow{\text{inv}_{\text{exp}}(t)} \), by proposition 33 and since \( \text{sim}_{\text{inv}}^{\min} \) is the smallest inv-simulation. By propositions 12 and 35, \( t \in \tau P \) and \( \mathcal{D}_p = \mathcal{D}_q \). By proposition 34, there is \( (\text{inv}_{\text{exp}}(t), R) \in \text{max}_Q \) such that \( \bigcup_{b_i \in B} \alpha B_i \subseteq R \). Then, by corollary 27, \( A' = \alpha Q - R \in \kappa_Q(q) \). And, for such an \( A' \), \( \bigcup_{b_i \in B} \alpha B_i \cap A' = \emptyset \).

(\( \Leftarrow \)) Let \( t \in \tau P \) and \( B \in \mathcal{D}_p \). By propositions 12 and 35, there exists \( p \in \text{V}_{\text{det}} \) such that \( p \xrightarrow{\tau} p \) and \( B = \mathcal{D}_p \). By corollary 27, there exists \( q \in \text{V}_{\text{det}} \) such that \( q \xrightarrow{\text{inv}_{\text{exp}}(t)} \) and so \( (p,q) \in \text{sim}_{\text{inv}}^{\min} \), by proposition 28. Hence there is \( A' \in \kappa_Q(q) \) such that \( \bigcup_{b_i \in B} \alpha B_i \cap A' = \emptyset \). Thus, by by corollary 27, \( (\text{inv}_{\text{exp}}(t), R) \in \text{max}_Q \) where \( R = \alpha Q - A' \). Moreover, for such an \( R \), \( \bigcup_{b_i \in B} \alpha B_i \subseteq R \). \( \square \)

7 Algorithms

In this section, we outline algorithms for checking the implementation relations IR1–IR5 except for IR1(a) which is implicitly tested during the generation of \( CTS^u_Q \) provided that \( CTS^u_Q \) does not contain any \( \tau \)-loop which is checked in order to establish that IR1(b) holds (see proposition 24).

To test for IR1(b), we use a modified version of the depth-first search algorithm given in [17] to test for strong connectivity in directed graphs. The (original) algorithm returns the nodes in each strongly connected component of the graph, thus indicating all those nodes on cycles. We make two modifications to the original algorithm. Firstly, we explore only \( \tau \)-transitions and so return only those strongly connected components in which all nodes are mutually accessible by such transitions. Secondly, we ignore components consisting of only one node, unless they admit a self-\( \tau \)-loop. Thus, due to proposition 24, IR1(b) is met if and only if the modified algorithm finds no \( \tau \)-traversable strongly connected components.

Algorithm 3. The outline for the algorithm is shown in figure 15. The \( \text{visit} \) function executes the recursive depth-first search which searches for (\( \tau \)-traversable) strongly connected components reachable (by \( \tau \)-only transitions) from the node \( v \) for which it is originally called. If and when such a component is encountered, the nodes within it are recorded in the global variable \( \text{cyclicNodes} \). The function \( IR1b() \) calls \( \text{visit} \) only for those nodes which have not already been considered and at least one \( \tau \)-transition leaving them.

The algorithm to test IR1(c) is based on proposition 28. We aim to construct the minimal extr-simulation \( \text{sim}_{\text{inv}}^{\min} \), by traversing the product \( V_{\text{det}} \times P_{\text{det}} \). We first map the initial nodes to each other, \( (q_0, p_0) \in \text{sim}_{\text{extr}} \). We then perform a depth-first search, beginning at \( (q_0, p_0) \). If the construction is successful, the set of all pairs of nodes reachable from \( (q_0, p_0) \) gives the minimal extr-simulation.

Algorithm 4. The pseudo-code of the algorithm is shown in figure 16. In the function \( IR1c() \) itself, the initial nodes in \( Q_{\text{det}} \) and \( P_{\text{det}} \) are paired. The \( \text{visit} \) function is then called for this initial pair of nodes. The data structure used to indicate if the paired nodes have been seen is:

- array [..\text{V}_{\text{det}}] [..\text{V}_{\text{det}}] \text{jointNodes}
function IR1b()
for every \(v \in V_{CT_Q}\) such that \(\tau \in \text{en}(v)\)
if \(v\) has not already been seen then \(\text{visit}(v)\)
if cyclicNodes \(\neq \emptyset\)
then return failure
else return success

function IR2()
for every \(q \in V_{Q_{det}}\)
if \(q\) has not already been seen then \(\text{visit}(q)\)
if cyclicNodes \(\neq \emptyset\)
then return failure
else return success

Fig. 15. Testing for IR1(b) and IR2

The array represents \(V_{Q_{det}} \times V_{P_{det}}\): if a combination in jointNodes has been seen by the execution of the depth-first search, then it constitutes a pair in \(sim_{\text{min}}^{\text{extr}}\) (this relation is then used in testing for IR3).

function IR1c()
jointNodes \(\leftarrow\) unseen; outcome \(\leftarrow\) success
visit(qo, po)
return outcome

void visit(q, p)
jointNodes[q][p] \(\leftarrow\) seen
for every \(q \rightarrow q'\)
if \(\text{extr}(q, a) = \tau\)
then
if jointNodes[q'][p] = unseen then visit(q', p)
else
if \(\text{extr}(q, a) \notin \text{en}(p)\)
then outcome \(\leftarrow\) failure
else if jointNodes[q'][p'] = unseen where \(p \rightarrow p'\) then visit(q', p')
return

Fig. 16. Testing for IR1(c)

The algorithm to test for IR2 is based on proposition 29. We again use a modified version of the algorithm testing for strong connectivity. This time, however, we wish only to find those strongly connected components where all nodes are mutually accessible by \(\tau\)-transitions after extraction. In addition, we wish to ignore components consisting of only one node \(q\), unless \(q \rightarrow q\) and \(\text{extr}(q, a) = \tau\), as this does not constitute a cycle for our purposes.
Algorithm 5. The pseudo-code for the algorithm is shown in figure 15. The visit function executes the recursive depth-first search which searches for (τ-traversable, after extraction given by $\text{extr}(q, a)$) strongly connected components reachable (by empty traces, after applying extraction given by $\text{extr}(q, a)$) from the node $q$ for which it is originally called. If and when such a component is encountered, the nodes within it are recorded in cyclicNodes. The function $IR_2()$ calls visit for every node $q \in V_{Q_{\alpha_1}}$.

The algorithm to test for IR3 is based on proposition 32 and uses the relation $sim_{\text{extr}}^{\text{min}}$ calculated during the execution of algorithm 4.

Algorithm 6. The pseudo-code for the algorithm to test for IR3 is shown in figure 17. It uses three auxiliary functions:

- $IR3a()$ to test for proposition 32(1) (which captures IR3(a)) for $q$ and $C_q$.
- $IR3b()$ to test for proposition 32(2) (which captures IR3(b)) for $q$ and $C_q$.
- getC() to calculate the set $C_q$ for a given $q$.

To test for IR4 we use proposition 33, aiming to construct the minimal inv-simulation $sim_{\text{inv}}^{\text{min}}$ by traversing the product $V_{P_{\alpha_1}} \times V_{Q_{\alpha_1}}$. We first map the initial nodes to each other, $(p_0, q_0) \in sim_{\text{inv}}$. We then perform a depth-first search, beginning at $(p_0, q_0)$. If the construction is successful, the set of all pairs of nodes reachable from $(p_0, q_0)$ gives the minimal inv-simulation.

If the depth-first search function has been called for $(p, q) \in sim_{\text{inv}}^{\text{min}}$, then we need to consider every transition $a$ out of $p$, and generate $\text{inv}(a)$ before considering if the first event in $\text{inv}(a)$ is offered as a transition out of $q$.

Algorithm 7. The pseudo-code is shown in figure 18. The function $IR_4()$ calls visit for $(p_0, q_0)$, as a result of which all pairs in $sim_{\text{inv}}^{\text{min}}$ are reached. The algorithm uses the following global array:

- array [1...$|V_{P_{\alpha_1}}|$$][1...|V_{Q_{\alpha_1}}|]jointNodes

The array represents $V_{P_{\alpha_1}} \times V_{Q_{\alpha_1}}$; if a combination in jointNodes has been seen by the execution of the depth-first search, then it constitutes a pair in $sim_{\text{inv}}^{\text{min}}$ (this relation is then used in testing for IR5).

The algorithm to test for IR5 is based on proposition 36, and uses the relation $sim_{\text{inv}}^{\text{min}}$ calculated during the execution of algorithm 7.

Algorithm 8. The pseudo-code for the algorithm to test for IR5 is shown in figure 18.

8 Concluding remarks

We have presented an implementation relation scheme which formalises the notion that a system built of communicating processes is an implementation of another base or target system in the event that the respective specification and implementation processes of which the systems are built have different interfaces. An important compositionality result was obtained which allows for automatic verification of the implementation relations in terms of constituent processes of the systems, so avoiding a major cause of the state explosion problem. We then presented algorithms for automatic verification of the implementation relations which take advantage of this compositionality result.

The algorithms presented here have been derived almost directly from the implementation relations themselves and little effort has yet been put into optimisation. Future work will explore possibilities for optimisation, as well as including a case study to evaluate the performance of the algorithms in practice and to establish in which context the implementation relations may prove to be most applicable.
function IR3[]
    for every \( q \in V_{\text{qf}} \)
        \( C_q \leftarrow \text{getC}(q) \)
        if \( IR3a(q, C_q) = \text{failure} \) or \( IR3b(q, C_q) = \text{failure} \) then return \( \text{failure} \)
    return \( \text{success} \)

function IR3a(q, C_q)
    \( B \leftarrow \bigcup_{b \in C_q} B \)
    for every \( b \in B \)
        if \( \delta(q^{[1]}) = D \) then return \( \text{failure} \)
    return \( \text{success} \)

function IR3b(q, C_q)
    if \( \delta(q^{[1]}) = \cdots = \delta(q^{m+n}) = d \)
        then
            for every \( B \in C_q \)
                for every \( p \) such that \( (q, p) \in \text{sim}_{\text{min}} \)
                    \( \text{successful} \leftarrow \text{false} \)
                    for every \( A \in \kappa_{\text{p}}(p) \)
                        if \( A \cap A = \emptyset \) then \( \text{successful} \leftarrow \text{true} \); break
                    if \( \text{successful} = \text{false} \) then return \( \text{failure} \)
            return \( \text{success} \)

function getC(q)
    \( C_q \leftarrow \emptyset \)
    for every \( A \in \kappa_{\text{q}}(q) \)
        \( B \leftarrow \emptyset \)
        for every \( b \in P \)
            if \( \exists R : q^{[1]} \in R \) then \( B \leftarrow B \cup \{b\} \)
        for every \( b \in \text{out} P \)
            if \( \forall R : q^{[1]} \notin R \) then \( B \leftarrow B \cup \{b\} \)
        \( C_q \leftarrow C_q \cup \{B\} \)
    return \( C_q \)

Fig. 17. Testing for IR3

References

function \text{IR4}()
  \text{jointNodes} \gets \text{unseen}; \text{outcome} \gets \text{success}
  \text{visit}(p_0, q_0, ()
  \text{return} \text{outcome}

\text{void visit}(p, q, \text{invEvents})
  \text{if} () \neq \text{invEvents} = \langle \alpha \rangle \circ \text{invEvents}'
    \text{then}
      \text{if} \alpha \not\in \text{en}(q)
        \text{then} \text{outcome} = \text{failure}
      \text{else} \text{visit}(p, q', \text{invEvents}') \text{ where } q \xrightarrow{\alpha} q'
    \text{else}
      \text{if} \text{jointNodes}[p][q] = \text{unseen}
        \text{then}
          \text{jointNodes}[p][q] \gets \text{seen}
          \text{for every } p \xrightarrow{\alpha} p'
            \text{visit}(p', q, \text{inv}_{\alpha_{p',q'}}(\alpha'))
      \text{return}

function \text{IR5}()
  \text{for every } p \in V_{P_{P_0}}
    \text{for every } A \in \kappa_{P}(p)
      B \gets \chi P - \chi A
      \text{for every } q \text{ such that } (p, q) \in \text{sim}^{\min}_{\text{inv}}
        \text{matchFound} \gets \text{false}
        \text{for every } A' \in \kappa_{Q}(q)
          \text{if } \bigcup_{B \in B} a B, \subseteq \chi Q - \chi A' \text{ then } \text{matchFound} = \text{true}; \text{break}
          \text{if } \text{matchFound} = \text{false} \text{ then return } \text{failure}
  \text{return } \text{success}

\text{Fig. 18. Testing for IR4 and IR5}
A Appendix

\( \chi(P||Q) = \chi P \cup \chi Q \)
\( \delta(P||Q) = \{ t : u \in \{ t[\chi P, t[\chi Q] \in (\tau P \times \delta Q) \cup (\delta P \times \tau Q) \} \} \)
\( \phi(P||Q) = \{ (t, R \cup S) \in (\phi P) \land (t[\chi Q, S] \in (\phi Q) \cup \delta(P||Q) \times 2^{\alpha(P||Q)} \}
\)

\( \chi(P\mid B) = \chi P - B \)
\( \delta(P\mid B) = \{ t[\chi P\mid B, B] \mid t \in \delta P \lor \exists a_1, a_2, \ldots \in aB \forall n \geq 1 : t \circ \{ a_1, \ldots, a_n \} \in \tau P \}
\)
\( \phi(P\mid B) = \{ (t[\chi P\mid B, B], R) \mid (t, R \cup \alpha B) \in \phi P \cup \delta(P\mid B) \times 2^{\alpha(P\mid B)} \}
\)

\( \chi(a \rightarrow P) = \chi P \)
\( \delta(a \rightarrow P) = \{ (a) \circ t \mid t \in \delta P \}
\)
\( \phi(a \rightarrow P) = \{ ((a) \circ t, R) \mid (t, R) \in \phi P \cup \{ () \} \times 2^{\alpha P - [a]} \}
\)

\( \chi(P \cap Q) = \chi P \)
\( \delta(P \cap Q) = \delta P \cup \delta Q 
\)
\( \phi(P \cap Q) = \{ () \mid () \in \phi P \cap \phi Q \cup \{ (t, R) \mid t \neq () \land (t, R) \in \phi P \cup \phi Q \}
\)

\( \chi(P \cap Q) = \chi P \)
\( \delta(P \cap Q) = \delta P \cup \delta Q 
\)
\( \phi(P \cap Q) = \phi P \cup \phi Q 
\).

In the above, \( B \) is a proper subset of \( \chi P \); \( b \notin \chi P \) and \( b \notin \chi P \) are channels with the same message sets; \( \delta[b/b'] \) is \( R \) with each \( b' \) changed to \( b \); \( a \) is an action in \( a P \); in the last two definitions \( \chi P = \chi Q \).