COMPUTING LABORATORY

Distributable nets

R.P. Hopkins

TECHNICAL REPORT SERIES
No 339 July, 1991
Abstract

Using Petri nets to represent systems, a framework is proposed for formulating the notion of one system being a distributed implementation of another, with a classification of whether an implementation introduces any additional activity and whether it introduces an unbounded amount of communication activity. We give net structure characterisations of those systems for which there is always a distributed implementation within a particular classification; and general constructions for those implementations.

©1991 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne,
Computing Laboratory, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

HOPKINS, Richard Pinder
Distributable nets.
[By] R.P. Hopkins

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1991.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 339)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.

Abstract

Using Petri nets to represent systems, a framework is proposed for formulating the notion of one system being a distributed implementation of another, with a classification of whether an implementation introduces any additional activity and whether it introduces an unbounded amount of communication activity. We give net structure characterisations of those systems for which there is always a distributed implementation within a particular classification; and general constructions for those implementations.

About the author

Dr. R.P. Hopkins has been a lecturer in the Computing Laboratory since October, 1983, prior to which he was a Research Associate.

Suggested keywords

DISTRIBUTED SYSTEMS
PETRI NETS
PROCESSOR PLACEMENT

Suggested classmarks (primary classmark underlined)
Dewey (18th): 511.5 0001.64404
U.D.C. 519.711 519.687
DISTRIBUTABLE NETS

R.P. Hopkins

Computing Laboratory
The University of Newcastle upon Tyne
Newcastle upon Tyne, NE1 7RU, U.K.

ABSTRACT Using Petri nets to represent systems, a framework is proposed for formulating the notion of one system being a distributed implementation of another, with a classification of whether an implementation introduces any additional activity and whether it introduces an unbounded amount of communication activity. We give net structure characterisations of those systems for which there is always a distributed implementation within a particular classification; and general constructions for those implementations.

Keywords Petri Nets, Distributed Systems, Processor Placement.

CONTENTS

0. Introduction
1. Basic Definitions and Notations
2. Framework for Expressing Distributability
3. Characterisations of Distributable Nets
4. Discussion
APPENDIX - Programming Notation
Acknowledgements
References

† This work was supported by ESPRIT Basic Research project - DEMON 3148.
0. Introduction

This paper addresses the common perception that certain structures of inter-process communication within a concurrent system do not have a "good" distributed implementation; and in particular the structures that can arise in concurrent programming languages which allow both input and output actions as guards. The framework we adopt is to take the system as being specified by a Petri net (specifically, a Place/Transition net [3] with unitary arc weights and unbounded place capacities [1]). To this specification net is added a locality function (giving a "located net") which assigns to each transition of the specification net a set of machines within which that transition must be implemented. Rationales for these distribution constraints might be: a particular transition being an external action which must take place on a particular hardware device; or a performance decision to exploit the concurrency by a particular actual parallelism structure.

A "distributed implementation" of a specification net is again a located net, with a certain behavioural correspondence to the specification net, and with its own locality function. This implementation locality must satisfy two conditions: firstly it must assign to each transition and place a single machine on which that element is implemented (executed/stored); secondly a place and the transitions which depend on it as input-must all be assigned to the same machine. These conditions (together with taking a net model where a transition's output places do not affect its firing) mean that every decision taken (in whether a transition fires) is local to one machine. Thus we assume a hardware model in which the communication mechanism supports just the unilateral transmission of a signal from one (source) "machine" to another (destination), corresponding to a token moving from a transition located in the source machine to a place located in the destination (we refer to a such a transition as a "communication" transition). This corresponds directly to communication architectures such as that supported by Transputer networks; and we assume that more sophisticated communication is ultimately realised in such a hardware model, either by protocols introduced by the translation from specification net to implementation net, or by the architecture's communication mechanism actually being in our terms one or more machines.

The required behavioural correspondence between specification net and implementation net is based on that of "simulation" from [1]. This requires that for each transition of the specification system there be a corresponding "goal" transition of the implementation system, and allows the implementation to introduce additional "tau" transitions; such that each transition firing of the specification system is simulated by a sequence of transition firings in the implementation with one being the goal and the rest being taus (and a converse condition.) To this we add the requirement that for a transition firing in the specification system, there must always exist an implementing sequence which is contained in the set of machines for that transition. Without some such locality-respecting requirement, every system would have the trivial "distributed implementation" in which everything is implemented on the same machine.

We will identify two classes of implementation: "tau-less", in which the implementation has the desirable property that no tau transitions are introduced (the translation from specification being just the introduction of more places, which effectively give replication of state over multiple machines); "busy", in which the implementation has the undesirable property that it can exhibit an unbounded number of communication transition
firings without any goal transition being fired, i.e. there is some busy-wait loop of tau transitions involving interaction between at least two machines.

We will use the structural notion of a net being (tau-less/busy-)distributable which means that, for any initial marking and any locality function, it has a (tau-less/busy-)distributed implementation. (This approach of dealing with net structure independently of particular initial marking and of particular locality function is discussed subsequently.) The main result we present is to characterise two classes of nets, ID ("interference dominated") and ICD ("interference closure dominated"), which are generalisations of Asymmetric Choice nets, such that: ID is exactly those nets (as specifications) which are distributable; ICD is exactly those nets (as specifications) which are tau-less-distributable; also, if a net is ID but not ICD, there will be some locality function and some initial state for which it has only busy implementations. Part of the proof of the above involves defining general constructions for the appropriate class of implementation of an ICD/ID net, and the derivation of its localities (and initial marking) from any that may be given for the net it is implementing.

The above development relates to nets in general. However to motivate and illustrate the development, and to make connection with work on translation of programming languages to nets, we consider program structures which generate the net structures of interest. Also, this work is towards algorithmic generation of a distributed implementation of a net, and nets generated from programs is an important special case of this. For this purpose, we introduce a programming notation which corresponds to an extended subset of OCCAM - this is close to the notation discussed in [2, 5] and related to e.g. [4]. To make the range of translation a located net, we include "placement" clauses, as in the PLACED PAR of OCCAM but generalised to allow explicit placement of any component process (not just those composed at the top-level PAR). Also, in order to be able to generate nets which are in ID but not ICD, we include an extension, and relax the variable sharing rules, such that communication via shared variables can be used similarly to message communication in OCCAM.

1. Basic Definitions and Notations

1.1. Strings and Sets

For any set-valued function $f$ (e.g. $f(x)$ being $\star x$ below), we extend $f$ to sets of arguments by $f(s) = \bigcup_{x \in s} f(x)$. Also, we use $\Theta$ for asymmetric set difference.

We use the following notations for strings $\lambda, \eta$ of elements $y$ from some set $Y$:

- $\varepsilon$ is the empty string.
- $Y^*$ is the set of strings over $Y$ (including $\varepsilon$.)
- $\lambda \cdot \eta, y \cdot \lambda$ and $\lambda \cdot y$ denote string concatenation and element prefixing and postfixing.
- $|\lambda|$ denotes the length of $\lambda$, i.e.: $|\lambda \cdot y| = |\lambda| + 1$; $|\varepsilon| = 0$.
- $\lambda^0$ and $\lambda^\circ$ denote the first and last element of $\lambda$, for $|\lambda| \geq 1$;
- $\lambda \downarrow Z$ denotes the projection of $\lambda$ onto $Z$, i.e.: $\varepsilon \downarrow Z = \varepsilon$;
  - if $y \in Z$ then $(y \cdot \lambda) \downarrow Z = y \cdot (\lambda \downarrow Z)$; if $y \not\in Z$ then $(y \cdot \lambda) \downarrow Z = \lambda \downarrow Z$.
- $[\lambda]$ denotes the set of elements that occur in $\lambda$, i.e. $[\varepsilon] = \emptyset$;
- $[\lambda \cdot x] = [\lambda] \cup \{x\}$.
For $f:Y \to Y'$, we extend $f$ to strings, $f:Y^* \to Y'^*$, as: $f(\lambda \cdot y) = f(\lambda) \cdot f(y)$; $f(\varepsilon) = \varepsilon$. Also we assume the corresponding extension of $f^{-1}$ if $f$ is bijective.

1.2. Nets

A net, $N$, is a triple $(S,T,F)$, $S$ being the set of places, $T$ being the set of transitions, and $F \subseteq (S \times T) \cup (T \times S)$ being the flow relation. We assume the set $X$ of elements, $X = S \cup T$, is finite, and that there are no isolated elements:

$$\forall x \in X: \exists x' \in X: (x,x') \in F \cup F^{-1}.$$ 

For elements $x,y \in X$: $x = \{ y \mid (y, x) \in F \}$; $x^* = \{ y \mid (x, y) \in F \}$; and $x^* = x \cap x^*$. We also use the notation: $x$ → $y$ or $y$ ← $x$ for $y \in x^*$; $x$ ↔ $y$ for $y \in x^*$; and $x$ → $y$, $y$ ← $x$ and $x$ ↔ $y$ for the negation of those conditions. In pictures of nets we use a bi-directional arrow to represent $x$ ↔ $y$, as in $t_2$ ↔ $t_3$ in Fig.3(i).

1.3. Systems

A system, $\Sigma$, is a quadruple $(S,T,F;M_0)$, where $N = (S,T,F)$ is a net and $M_0$ is the initial marking, a marking for $N$ being a function from $S$ to the non-negative integers.

A transition $t \in T$ is enabled at a marking $M_1$, denoted $M_1[t]$, iff $\forall s \in T: M_1(s) \geq 1$. If $M_1[t]$, then the new marking obtained by firing $t$, denoted $t::M_1$, is given by:

$$\forall s \in S: t::M_1(s) = M_1(s) + \delta_{t,s}, \text{ where } \delta_{t,s} = |\{s \cap t\} - |\{s \cap t\}|$$

The notation $M_1(t)M_2$ means $M_1(t)$ and $t::M_1 = M_2$

(We take $t::$ to be defined not just on markings, but on functions from $S$ to integers (including negative integers), in order to unconditionally use the property $t::(u::M) = u::(t::M)$)

The set of markings reachable from a marking $M_1$, denoted $[M_1]$, is the smallest set of markings such that:

$$M_1 \in [M_1]; \text{ and } M_2 \in [M_1] \land \exists t:M_2(t)M_3 \Rightarrow M_3 \in [M_1]$$

1.4. Occurrence/Firing Sequences

For a net $N = (S,T,F)$, and a marking $M_1$:

A finite string of alternating markings and transitions, $\sigma = M_1 \cdot t_2 \cdot M_2 \cdot \ldots \cdot t_n \cdot M_n$, (n ≥ 1), is an occurrence-sequence (from $M_1$ to $M_n$), denoted $M_1(\sigma)M_n$ or $M_1(\sigma)$ if: either $n = 1$, or $\forall i, 2 \leq i \leq n: M_{i-1}(t_i)M_i$.

We define the sequential composition of two occurrence sequences, $\sigma_1 = \lambda_1 \cdot M$ and $\sigma_2 = \lambda_2 \cdot M$, as $\sigma_1; \sigma_2 = \lambda_1 \cdot M; \lambda_2$.

A finite string of transitions, $\omega = t_2 \cdot t_3 \cdot \ldots \cdot t_n$, is a firing-sequence (from $M_1$ to $M_n$), denoted $M_1(\omega)M_n$ or $M_1(\omega)$, if there is an occurrence sequence $\sigma$, from $M_1$ to $M_n$, such that $\sigma \upharpoonright T = \omega$.
1.5. Additional Notions

Defn. 1.1

For a net \( N = (S, T, F) \), and a place \( s \in S \), we define:

(a) \( \text{dec}(s) = \{ t \in T \mid s \rightarrow t \} \), the transitions which decrease its marking;
(b) \( \text{inc}(s) = \{ t \in T \mid t \rightarrow s \} \), the transitions which increase its marking; and
(c) \( \text{rest}(s) = \{ t \in T \mid s \leftrightarrow t \} \), the transitions which decrease and immediately restore its marking.

(d) We say two transitions, \( t_1, t_2 \in T \), interfere at \( s \), denoted \( t_1 \ldots t_2 \) or \( t_1 \leftrightarrow t_2 \), if
   \[ t_1 \neq t_2 \land t_1 \leftarrow s \rightarrow t_2 \land \neg (t_1 \rightarrow s \leftrightarrow t_2) \]
   This means that given a suitable marking, at least one can disable the other.

(e) We define the predicate \( \text{sticky}(s) = s \subseteq s \). (A sticky place is a singleton trap, once marked it remains marked; and no transitions interfere at a sticky place.)

2. Framework for Expressing Distributability

2.1. Located Nets

We now introduce a located net or system by adding a locality function which gives the locality of each transition as the set of machines within which it must be implemented, and identify the locality of a place as the set of machines common to all its post-transitions. Locality functions are subject to the restriction that all transitions having a common pre-place have also some common machine. Without some such restriction (discussed further subsequently), we would find that only interference-free nets would be distributable.

Defn./Prop. 2.1

(a) We assume \( MC \) as a finite set of "machines", and a selection function
   \( \text{sel} : 2^{MC} \land \emptyset \rightarrow MC \), such that \( \forall mcs \in 2^{MC} \land \emptyset : \text{sel}(mcs) \in mcs. \)

(b) A located net is a quadruple \( (S, T, F, L) \) where \( N = (S, T, F) \) is a net and \( L \) is a locality function for \( N \), which is a function \( L : T \rightarrow 2^{MC} \) that satisfies the property:
   \[ \forall s \in S, \forall t \in T : L(t) \neq \emptyset \land s \neq \emptyset \Rightarrow \forall t' \in s.s, L(t') = \emptyset \]
   (That property could be weakened, without loss of the results obtained, to:
   \[ \forall s \in S, \forall t \in T : L(t) \neq \emptyset \land \neg \text{sticky}(s) \Rightarrow \forall t' \in s.s, L(t') = \emptyset \]

(c) We define the extension of \( L \) to places as:
   \[ \forall s \in S : L(s) = \cap_{t \in s \cdot t} L(t) \]

(d) Obviously, \( s \in t \Rightarrow L(s) \subseteq L(t) \).
   Also, by the restriction on \( L \), we have \( \forall s \in S : s \neq \emptyset \Leftrightarrow L(s) \neq \emptyset \)

In Figs.1(i)-4(i) we show some located nets (with in each case (ii) being a distributed implementation of (i) if such exists). We take the set of machines as being \( \{ A, B, \ldots \} \), and indicate the localities by labeling transitions and place, and/or by labeling segments of a horizontal bar such that a net element's locality (if non-empty) is that labeling the segment below it - see Fig.1. where both notations are used. In Fig.3(i) and Fig.4(i) the
Fig. 1. (i) An example with no distribution

(i) Specification

(ii) Distribution

Fig. 2 - An example with a simple distribution

C:: ( x=T ; ( ( A:: ( T [] x ) ) || B:: ( t3 [ ] T ) ) )

(i) Specification

(ii) Distribution

Fig. 3 - An example which requires place replication for its distribution
Fig. 4 - An example requiring busy distribution
initial marking would be a single token on each place which has no input transitions; the marking shown being reachable from that. The solid arcs highlight the structures of significance for distributability, the remainder being a result of producing the net from a particular program structure.

In [2,5] is given a scheme for the translation of program structures to net structures. Here we use the ideas of that translation to give for each specification net in Figs.1-4 an example program fragment which would generate such a net structure as (part of) its translation. These use a process notation which is adapted from that proposed in [5], and very similar to that of [4]. The programming notation and the scheme for obtaining a located net from a program are described in the Appendix. In the figures we give for some program constructs their correspondences with particular places/transitions in the net.

2.2. Distributed Nets and Communication

We now define the special class of located nets (or systems) which are "distributed" in that each transition is located on a single machine together with all the places which affect its firing. For such we introduce the notion of a "communication" transition as one which communicates information to a different machine.

Defn./Prop. 2.2

(a) A distributed net is a located net \( (S, T, F; L) \) with \( \forall t \in T: |L(t)| = 1 \).

(b) For such a distributed net it is obvious that \( \forall s \in S: s \neq \emptyset \leftrightarrow |L(s)| = 1 \); and \( \forall t \in s^*: L(s) = L(t) \).

(c) A located/distributed system, \( \Sigma \), is a quintuple \( (S, T, F; L, M_0) \) where \( (S, T, F; L) \) is a located/distributed net and \( (S, T, F; M_0) \) is a system. We say \( \Sigma \) is "built-on" net \( (S, T, F) \).

(d) For a distributed system \( \Sigma \), a transition \( t \) is a communication if it is a member of the set of communications, \( \Xi_\Sigma \), where

\[
\Xi_\Sigma = \{ t \mid \exists t' \in T: L(t') \neq L(t) \}.
\]

In each of Figs.2-4, (ii) is a distributed net, and e.g. in Fig.2(ii) the communication transitions are \( t_0 \)' and \( t_2 \)'.

2.3. Simulation and Distribution

We now consider requirements for a locality-respecting behavioural correspondence (I-simulation) between an implementation \( \Sigma' \) and a specification \( \Sigma \).

Defn. 2.3. I-simulation

Let \( \Sigma = (S, T, F; L, M_0) \) and \( \Sigma' = (S', T', F'; L', M_0') \), be two located systems with \( T' = T \cup G \), and \( T \cap G = \emptyset \), and \( f: T \rightarrow G \) a bijection. We say \( \Sigma' \) is a locality-respecting simulation, or I-simulation of \( \Sigma \) with respect to \( f \), denoted \( \Sigma \vdash f \Sigma' \), iff there is a surjection \( \beta': [M_0'] \rightarrow [M_0] \), with \( \beta = \beta'^{-1} \), such that:
(i) \( M_0 = \beta'(M_0') \)

(ii) Suppose \( M_1 = \beta'(M_1') \), \( M_1' \in \{M_0'\} \), \( M_1 \in \{M_0\} \), then

(a) Whenever \( M_1(t)M_2 \) with \( t \in T \)
then \( \exists M_2' \in \beta(M_2) \), \( \exists \omega \in T^*: M_1'[\omega]M_2' \land f^{-1}(\omega \downarrow G) = t \land L'([\omega]) \subseteq L(t) \)
(b) Whenever \( M_1'[\omega]M_2' \) with \( \omega \in T^* \), then \( M_1[f^{-1}(\omega \downarrow G)] \beta'(M_2') \).

(iii) \( \forall M \in \{M_0\} \colon |\beta\{M\}| \) is finite.

The notion of behavioural correspondence defined above is that of "simulation" from [1], but with an additional locality-respecting requirement (as the final conjunct of clause (ii)(a)). Simulation requires that each transition \( t \) of \( \Sigma \) be equated (by injection \( f \)) with a corresponding transition \( t' \) of \( \Sigma' \). Each such \( t' \) is a "Goal" (G) transition which must fire once in simulating a firing of its \( t \). \( \Sigma' \) may also have some additional "tau" transitions (T) which may be fired in achieving a goal firing. The behavioural correspondence is formulated in terms of each state \( M' \) of \( \Sigma' \) having a corresponding state of \( \Sigma \) (surjection \( \beta' \)) such that the initial states correspond and all pairs of corresponding states have corresponding possible futures, as follows. (Clause (ii)(a)) If \( t \) can fire at \( M \), then a firing of the corresponding goal \( t' \) must be achievable from \( M' \), with the resulting states again corresponding, and (for locality-respect) it must be achievable by transitions with localities within those of \( t \). (Clause (ii)(b)) If a sequence \( \omega \) of firings from \( M' \) is possible, then from \( M \) it must be possible to have the sequence of firings which corresponds to the goal sequence achieved by \( \omega \), and both sequences must again lead to corresponding states. (Note that in clause (ii)(b) there is no locality-respecting requirement; we do not require that every sequence of \( \Sigma' \) which achieves a \( t \) of \( \Sigma \) do so within the locality of \( t \).) The remaining clause (iii) is to ensure that if \( \Sigma \) is safe and \( \Sigma' \) implements \( \Sigma \), then \( \Sigma' \) also is safe.

Prop. 2.4.

With \( \Sigma \overset{f}{\rightarrow} \Sigma', \beta', \beta, f, T, G, M, M_1, M_1' \), as above:

(a) If \( M_1'(t')M_2', t' \in T, M_1' \in \{M_0'\} \) then \( \beta'(M_2') = \beta'(M_1') \).

This follows by having \( \omega = t' \) in (ii)(b) of Defn.2.3, whereby \( f^{-1}(\omega \downarrow G) = \epsilon \), and thus \( \beta'(M_1')(\epsilon) \beta'(M_2') \), which gives \( \beta'(M_1') \beta'(M_2') \).

(b) Whenever \( M_1(t)M_2 \) with \( t \in T \), \( M_1 \in \{M_0\} \)
then \( \exists M_2' \in \beta(M_2), \exists \omega \in T^* \colon M_1'[\omega \cdot f(t)]M_2' \land L'([\omega \cdot f(t)]) \subseteq L(t) \).

(follows directly from (ii)(a) of Defn.2.3.)

(c) If \( M_1(t) \), with \( t \in T \), then \( L'(f(t)) \subseteq L(t) \)
(follows directly from (ii)(a) of Defn.2.3.)

Defn. 2.5.

\( \Sigma' \) is an distribution of \( \Sigma \), denoted \( \Sigma_{d1} \Gamma_\Sigma' \), if \( \Sigma' \) is distributed and \( \Sigma_{d1} \Gamma_\Sigma' \).

We now illustrate the notions of simulation and distribution using the emphasised structures in the example of Fig.4, where (ii) is a distribution of (i). For Fig.4 the goal transitions are \( t_1' - t_4' \), corresponding respectively to \( t_1 - t_4 \) (and we always use that convention of \( t_i' \) being the goal corresponding to \( t_i \)). The \( \tau_1 - \tau_4 \) are the additional tau transitions.
The place \( x_P \) of (i) is split into places \( x_1P - x_3P \), and similarly for \( y_P \). The state correspondence is that in all corresponding pairs of states: the marking of \( x_P \) equals the marking of \( x_1P \) and equals the sum of the markings of \( x_2P \) and \( x_3P \); and the similar condition for \( y_P \). It is easy to check that the simulation conditions are met for any markings of the \( x \) and \( y \) places which conforms to this state correspondence condition. For instance, at the marking shown, \( t_1 \) can fire which requires that there be an implementing sequence which achieves \( t'_1 \). Three such sequences are \( t'_1, \tau_4 \cdot t'_1, \tau_4 \cdot \tau_3 \cdot t'_1 \). The first two of these sequences satisfy the locality-respecting requirement in that both its transitions are on machine \( A \) which is the only machine for \( t_1 \). (However the third of these involves machine \( B \) for \( \tau_3 \) and thus does not meet that requirement.) Also, for the firing of \( t_2 \), we have e.g. any sequence of the form \( \tau_1 \cdot \tau_2 \cdot t_1 \cdots \tau_3 \cdot \tau_1 \cdot t_2' \), which respects the \( \{ A, B \} \) locality of \( t_2 \). Note that the tau sequence \( \tau_1 \cdot \tau_2 \) comprises communication transitions and can repeat indefinitely; and thus the distribution is "busy", as defined below. In the Fig.2 and Fig.3, the distribution relationship between (i) and (ii) is simpler in that it is a "tau-less" distribution, and can be easily checked.

**Defn. 2.6 - Classification of Distributions**

For \( \Sigma \xrightarrow{\tau} \Sigma' \), with \( T \) and \( M_0' \) as in Defn.2.3, the distribution is

(a) **tau-less**: iff \( T = \emptyset \)

(b) **busy**: if there can be an unbounded number of communication transition firings (transitions in \( \Sigma \)) without a goal transition firing; i.e., there is no integer \( N \) such that: whenever \( M_1'(\omega) \) with \( \omega \in T^* \), \( M_1 \in (M_0') \) then \( |\omega \downarrow \Sigma| < N \)

(One could strengthen the definition of busy by excluding those implementations for which an unbounded communication sequence can only occur from a marking \( M_1 \) for which some transition of \( \Sigma \) is enabled at \( \beta'(M_1') \).)

**Defn. 2.7 - Distributability**

A Net \( N \) is (busy-/tau-less-)distributable if for every \( M_0 \) and \( L \) such that \( \Sigma = (N; L, M_0) \) is a located system, there is some \( \Sigma' \) which is a (busy - /tau-less-)distribution of \( \Sigma \).

**2.4. A Theorem giving Non-distributability Results.**

**Theorem 1**

Given located systems \( \Sigma = (S, T, F; L, M_0) \) and \( \Sigma' = (S', T', F'; L', M_0') \), an injection \( f: T \rightarrow T' \), and a surjection \( \beta': (M_0') \rightarrow (M_0) \), with transitions \( t_b, t_c \in T \), a place \( s_a \in S \), and a marking \( M_A \in (M_0) \) such that the following premises hold (see Fig.5(i)):

(i) \( t_b \rightarrow s_a \rightarrow t_c \); (Either \( t_b \rightarrow s_a \) or \( t_c \rightarrow s_a \); without loss of generality assume the former.)

(ii) \( M_A(t_b), M_A(t_c), M_A(s_a) = 1 \)

(iii) \( \Sigma' \) is a distribution of \( \Sigma \) with respect to \( f \), with \( \beta' \) being the markings surjection.

Then the following consequences hold, where \( t_b' = f(t_b) \) and \( t_c' = f(t_c) \)
(a) \( L'(t_b') \subseteq L(t_c) \)

(b) \( L'(t_c') \subseteq L(t_b) \)

(c) \( L'(t_b') \neq L'(t_c') \Rightarrow \) the distribution is busy.

The use of this theorem can be seen in showing that the located system of Fig.1 has no distribution; and that the located system of Fig.4 has at best a busy distribution (and thus the corresponding distributability properties of the underlying nets).

\[ \Sigma = \text{Fig.1(i)} \]
Assume some distribution \( \Sigma' \), with the usual ' notation for correspondences with \( \Sigma \). Take the reachable marking of single tokens on \( c_c \) and \( c_d \). The (a) or (b) of the theorem applies to the interference \( t_1 x_c t_2 \), and thus \( L'(t_2') \subseteq L(t_1) = \{ A \} \), i.e. that interference forces \( L'(t_2') = \{ A \} \).
Similarly \( t_2 x_c t_3 \) forces \( L'(t_2') = \{ B \} \). Which is a contradiction.

\[ \Sigma = \text{Fig.4(i)} \]
Assume some distribution \( \Sigma' \), with the usual ' notation for correspondences with \( \Sigma \). Take the marking shown. Interference \( t_1 x_y t_2 \) forces \( L'(t_2') = \{ A \} \); and interference \( t_3 x_y t_4 \) forces \( L'(t_3') = \{ B \} \). That is, \( L'(t_2') = L'(t_3') \), and so applying (c) of the Theorem at interference \( t_2 + y t_3 \) we have that the distribution is busy.

Note, the highlighted structure of Fig.4(i) is the smallest structure which can force busy distribution. If say the flow \( y_F t_2 \) were deleted, then \( t_2 \) could have locality just \( \{ A \} \), in which case interference structure \( t_2 x_F t_3 y_F t_4 \) means there is no distribution, as for the similar structure in Fig.1. On the other hand, if say \( x_F t_1 \) were deleted, then there would be a tau-less distribution (with \( L'(t_2') = L'(t_3') = L'(t_4') = \{ B \} \).

---

Fig. 5 - Structures used in the Proof of Theorem 1

Fig.5 shows the structures used in the theorem and its proof: (i) is the interference at \( s_a \); (ii) is the relevant part of the reachability tree of \( \Sigma \), with the notation being that a dashed arrow means that the transition sequence labeling it is a firing sequence from its source (to destination) marking, and a crossed arrow means it is not; (iii) and (iv) are
part of the reachability tree of $\Sigma'$, with the convention that marking $X_i$ corresponds to marking $M_X$, and the additional notation of a dotted arrow, $x$ or $y$, from one reachability arc to another, which means that a transition in the label of the former is disabled by a transition in the label of the latter, by having a singly-marked common pre-place; (v) and (vi) show the disabling structures identified in (iii) and (iv).

Proof of Theo.1

1. Let $A_1$ be any marking of $\Sigma'$ such that $A_1 \in (M_0')$ and $\beta'(A_1) = M_A$.

2. We have the (non-)reachability of the markings and firing sequences shown in Fig.5(ii)-(iv.), namely -

2.1. $\exists M_C, M_B \in (M'_0): M_A[t_c] M_C \land M_A[t_b] M_B \land \neg M_B[t_c]$  
(By premises (i) and (ii).)

2.2. $\exists \mu \in T^*, \exists A_2, B_1 \in (M'_0)$ such that:
$A_1[\mu] A_2 \land A_2[t_b] B_1 \land \beta'(B_1) = M_B$  
(By Prop.2.4(b) for $M_A(t_b) M_B$, $\beta'(A_1) = M_A$.)

and $\beta'(A_2) = M_A$  
(By Prop.2.4(a) repeatedly along $A_1[\mu] A_2$.)

2.3. $\exists \omega \in T^*, \exists A_3, C_1 \in (M'_0)$ such that:
$A_2[\omega] A_3 \land A_3[t_c] C_1 \land \beta'(A_3) = M_A \land L'[(\omega \cdot t_c')] \subseteq L(t_c)$  
(As for 2.2, Prop.2.4(b) and  
and $\beta'(A_2) = M_A$.)  
(By Defn.2.3(ii)(b), $B_1[\omega \cdot t_c']$ gives $\beta'(B_1)[t_c]$, i.e. $M_B(t_c)$, contradicting 2.1.)

Furthermore, we take $\omega$ as a minimal sequence such that $A_2[\omega \cdot t_c']$

2.4. $\exists \nu \in T^*, \exists A_4, B_2 \in (M'_0)$ such that:
$A_3[\nu] A_4 \land \beta'(A_4) = M_A \land A_4[t_b] B_2 \land \beta'(B_2) = M_B \land L'[(\nu \cdot t_b')] \subseteq L(t_b) \land \neg B_2[t_c']$  
(As for 2.2/2.3, using $\beta'(A_3) = M_A$ and $M_A(t_b) M_B$.)

3. We now deal with the existence of transitions and places $t_d', s_d', t_e'$ and $s_e'$, with the structure shown in Fig.5(v,v).

3.1. $\exists t_d' \in (\omega \cdot t_c') \land \exists s_d' \in S'': t_b' \leftarrow s_d' \rightarrow t_d'$

Assume the contrary, which implies there is not a transition in $\omega \cdot t_c'$ which is disabled by $t_b'$, and not such a transition which disables $t_b'$.

From 2.3 we have $A_2[\omega \cdot t_c'] C_1$ which means that there is an occurrence sequence $S_0 \cdot \omega_1 \cdot S_1 \ldots \omega_n \cdot S_n$, where $S_0 = A_2$, $S_n = C_1$ and $\omega_1 \ldots \omega_n = \omega \cdot t_c'$; and $t_b'$ is enabled at every $S_i$, by the assumption and it being enabled at $S_0 = A_2$.

Thus (by firing rule definition) we have $S_i[\omega_i] (\omega_i::S_i)$ with $(\omega_i::S_i) = S_{i+1}$.

This implies $(t_b'::S_i)(\omega_i)(\omega_i::(t_b'::S_i))$, since: $t_b'::S_i$ is the marking produced by firing $t_b'$ at $S_i$, which leaves $\omega_i$ enabled, by the assumption.

This can be re-written as $(t_b'::S_i)(\omega_i)(t_b'::(\omega_i::S_i))$, and thus as $(t_b'::S_i)(\omega_i)(t_b'::S_{i+1})$

Thus there is the occurrence sequence $S_0\cdot t_b'\cdot(t_b'::S_0)\cdot\omega_1\cdot(t_b'::S_1)\ldots \omega_n \cdot(t_b'::S_n)$ which contradicts the $\neg B_1[\omega \cdot t_c']$ of 2.3 (since $B_1 = t_b'::A_2 = t_b'::S_0$, and $\omega_1 \ldots \omega_n = \omega \cdot t_c'$); and thus the assumption is false.
3.2. \( \exists \epsilon \in [\nu \cdot t_b'] \), \( \exists \epsilon \in S': \ t_c' \leftarrow \epsilon \rightarrow t_e' \)
Since otherwise \( t_c' \) being enabled, which holds at \( A_3 \), would hold at every marking reached in the occurrence sequence for \( A_3(\nu \cdot t_b')B_2 \); and thus \( t_c' \) would be enabled at \( B_2 \), which contradicts 2.4.

4. To prove clause (a)
From \( t_b' \leftarrow \epsilon \rightarrow t_d' \), we have \( L'(t_b') = L'(t_d') \), by Prop.2.2(b) since \( S' \) is a distributed net.
Now (by 2.3) \( L'([\omega \cdot t_c']) \subseteq L(t_c) \) and (by 3.1) \( t_d' \in [\omega \cdot t_c'] \), and thus \( L'(t_d') \subseteq L(t_c) \)
Therefore \( L'(t_b') \subseteq L(t_c) \).

5. To prove clause (b)
As for proof of clause (a), with \( t_b, s_d', t_d, \omega \) and \( t_c \) all respectively replaced by \( t_c, s', t_e, v \) and \( t_b \).

6. To prove clause (c)
Take \( \omega = t_0 \cdot t_1 \cdot \ldots \cdot t_n' \), and let \( t_n+1' = t_c' \).
Assume: \( [\omega \downarrow \Xi_S] = \emptyset \), i.e. \( \forall t' \in [\omega] : L'(t') = L'(t_{**}) \)
Now we have \( \forall i \in \{1, \ldots, n+1\} : \exists j \in \{i+1, \ldots, n+1\}, \exists s' \in S' : t_i' \rightarrow s' \rightarrow t_j' \)
since otherwise: \( A_2[t_1' \cdot \ldots \cdot t_i-1' \cdot t_{i+1}' \cdot \ldots \cdot t_n' \cdot t_{c'}] \), which contradicts minimality of \( \omega \).
(I.e., if \( t_t \) does not mark a place on which some subsequent transition of the sequence is dependent, then we can eliminate \( t_t \) from the sequence and still fire the remainder).
Thus, \( L'(t_i') = L'(t_j') \), by \( t_j \in t_{**} \), and the assumption.
Hence, by an induction with base case \( L'(t_{n+1}') = L'(t_c') \), and induction step for \( t_j' \):
\[ L'(t_{j+1}') = L'(t_{j+2}') = \ldots = L'(t_{n+1}') = L'(t_c') \Rightarrow L'(t_j') = L'(t_c') \]
we have \( \forall i \in \{1, \ldots, n+1\} : L'(t_i') = L'(t_c') \); i.e. \( t' \in [\omega \cdot t_c'] \Rightarrow L'(t') = L'(t_c') \)
Now, \( t_d' \in [\omega \cdot t_c'] \), and so \( L'(t_d') = L'(t_c') \)
But (from step 4.), \( L'(t_b') = L'(t_d') \), and so \( L'(t_b') = L'(t_c') \)
Thus, \( L'(t_b') \neq L'(t_c') \) implies the assumption is false, i.e. implies \( \exists t' \in [\omega] : t' \in \Xi_S \).
Thus from an arbitrarily chosen marking \( A_1 \) in \( \beta^{-1}(M_A) \) there is a transition sequence, \( \lambda = \mu \cdot \omega \), to a marking, \( A_3 \), again in \( \beta^{-1}(M_A) \), with \( [\lambda \downarrow \Xi_S] = \emptyset \land [\lambda] \subseteq T \.
Hence there can be a transition sequence comprising any number of such \( \lambda \)s, and so \( S \) is busy.

3. Characterisations of Distributable Nets
We define two classes of nets and then deal with their distributability properties.

3.1. Interference (Closure) Dominated Nets

Defn. 3.1
(a) We define the interference set, \( IS(t) \) of \( t \in T \), as itself and the set of transitions with which it interferes (as in Defn.1.1(d)): \( IS(t) = \{ t' \in T | t' \rightarrow t \} \cup \{ t \} \).
(b) We define \(\rightarrow_*\) as the transitive reflexive closure of \(\rightarrow\), this being an equivalence relation with its equivalence classes being referred to as interference classes.

(c) The interference class, \(IC(T)\), of \(t \in T\) is \(IC(t) = \{t' \in T | t' \rightarrow_* t\}\).

(d) For a set of transitions \(ts \subseteq T\), we denote their common pre-places, or dominating places, as \(\Uparrow ts\); i.e. \(\Uparrow ts = \{s \in S | ts \subseteq s\}\).

A net \(N = (S, T, F)\) is:

(e) interference dominated, ID, if \(\forall t \in T: \bullet t = \emptyset \lor \Uparrow IS(t) = \emptyset\).

(f) interference closure dominated, ICD, if \(\forall t \in T: \bullet t = \emptyset \lor \Uparrow IC(t) = \emptyset\).

For Fig.1, the net is not ID since \(IS(t_2) = \{t_1, t_2, t_3\}\) has no dominating place. For Fig.2(i), the net is ICD; within the emphasised structure, the interference class \(\{t_1, t_2\}\) is dominated by \(c_c\). For Fig.3(i), the net is ICD; the significant interference classes are \(\{t_1, t_2\}\) and \(\{t_3, t_4\}\), which are dominated by \(c_a\) and \(c_b\) respectively. In Fig.4(i), the net is ID; e.g. the interference sets of \(t_2\) (i.e. \(\{t_1, t_2, t_3\}\)) and of \(t_3\) (i.e. \(\{t_4, t_2, t_3\}\)), are dominated by \(x_F\) and \(y_F\) respectively. However, it is not ICD; the interference class \(\{t_1, t_2, t_3, t_4\}\) has no dominating place.

3.2. Equality of I(C)D and (tau-less-)distributable

The result of this paper is that a net is distributable iff it is ID, and is tau-less-distributable iff it is ICD. The following theorems imply this result, but are somewhat stronger.

Theo. 2- If a net \(N\) is not ICD, then there is some located system built on \(N\) for which any distribution must be busy.

Theo. 3- If a net \(N\) is not ID, then there is some located system built on \(N\) for which there is no distribution.

Theo. 4- If a net \(N\) is ICD then it is possible to construct a net \(N'\), such that any located system built on \(N\) has a tau-less distribution built on \(N'\).

Theo. 5- If a net \(N\) is ID then it is possible to construct a net \(N'\), such that any located system built on \(N\) has a distribution built on \(N'\).

Theo. 4 and particularly Theo. 5 are rather tedious, and are only outlined, in the next two sections. In the sections following that we give the proofs of Theo. 2 and 3, which are simple applications of Theo. 1.

3.3. Outlines of the constructions and proofs for ICD nets (Theorem 4)

The basis for the construction needed for this theorem is the place replication used in Fig.3. Fig.6 shows a more general example. The following steps are necessary.

3.3.1. Net

Construct a new net \(N'(=\text{Fig.6(ii)})\) from the given net \(N(=\text{Fig.6(i)})\).

For each transition \(p\) of \(N\) there is a corresponding transition \(p'\) of \(N'\) - these are the goal transitions and there are no tau transitions. In \(N\) there may be sticky places, e.g. \(a\), having post-transitions, \(t, u, v\), all of which are restoring transitions. Each such place becomes in \(N'\) a set of places, \(\langle a, t\rangle, \langle a, u\rangle, \langle a, v\rangle\), each connected out to one of the
correspondents \((t', u', v')\), of those post-transitions and also in from every transition \((x'\) and \(y')\) which corresponds to a purely pre-transition \((x\) and \(y\)) of \(a\). Note that in the case of place \(b\) which has no pre-transitions, we obtain in \(N'\) a disconnected part of the net, \(w' \leftrightarrow (b, w)\), even though the original is connected. Apart from this replication of sticky places, the structure of \(N\) is carried forward to \(N'\), e.g. \(u \leftrightarrow c \rightarrow v\) becomes \(u' \leftrightarrow c' \rightarrow v'\).

### 3.3.2. Marking

Construct an initial marking \(M_0'\) for \(N'\) from a given initial marking \(M_0\) of \(N\).

Every place in \(N'\) has the initial marking of the place in \(N\) from which it is derived. E.g., \(M_0'((a, u)) = M_0(a)\) and \(M_0'(c') = M_0(c)\).

### 3.3.3. Locality

Construct a locality function, \(L'\) for \(N'\) from a given locality function \(L\) for \(N\).

For this we make use of the selection function \(sel\) of Defn.2.1(a) which provides a single machine from a locality. If a transition of \(N'\) has no pre-place, e.g. \(y'\), we select its machine from the corresponding transition of \(N\), \(L'(y') = \{sel(L(y))\} = \{C\}\). Otherwise, e.g. \(u'\), we select one \((C)\) from those \((A, B, C)\) of the places \((a, c, b)\) which dominate the interference class \((u, v)\) to which \(u\) belongs; i.e. \(L'(u') = \{sel(L(\uparrow IC(u)))\}\). This is the
part of the construction which depends on $N$ being ICD.)

This construction for $L'$ satisfies the requirement for $L'$ being a locality function, and $(N';L')$ being a distributed net, namely: (i) All localities are singleton which is obvious; (ii) All the post-transitions of a place have the same locality. For (ii): for all post-transitions of a carried-forward place in $N'$, e.g. $u'$ and $v'$ of $c'$, the corresponding transitions, $u$ and $v$, in $N$ must be in the same interference class, and thus the constructed $L'$ gives them all the same locality; for other places, e.g. $(a,u)$, there is only one post transition. The construction also ensures that for all $p'$ of $N'$, $L'(p')\subseteq L(p)$ which is needed for the locality-respecting part of the next step.

3.3.4. Behaviour

Show that $\Sigma = (N', L', M_0')$ satisfies the requirements of Defn.2.3 for being a locality-respecting simulation of $\Sigma = (N; L, M_0)$ under the transition correspondence $f(p) = p'$.

The required state correspondence $\beta'$ is that $(M', M)$ is in $\beta'$ if $M'(s') = M(r)$ where $r$ is the place of $N$ from which is derived place $s'$ of $N'$, e.g. $M'(a, u) = M(a)$ and $M'(c') = M(c)$. It is straightforward to show that $\beta'$ is a surjection from reachable markings of $\Sigma'$ to reachable markings of $\Sigma$, and that it satisfies the requirements of Defn.2.3.

3.4. Outlines of the constructions and proofs for ID nets (Theorem 5)

The basis for the construction needed for this theorem is the example of Fig.4 where e.g. the place $x_F$ of (i) becomes in (ii) the place $x_1$ and the cycle $x_2 \rightarrow \tau_1 \rightarrow x_3 \rightarrow x_2$. Fig.7 shows a more general example. The following steps are required, as for the previous section.

3.4.1. Net

Construct a new net $N' = \text{Fig.7(ii)}$ from the given net $N = \text{Fig.7(i)}$.

For each transition $p$ of $N$ there is a corresponding transition $p'$ of $N'$ - these are goal transitions.

A place in $N$ having no post-transitions, such as $e$, is carried forward as a corresponding place, $e'$, with corresponding connectivity, $e' \leftrightarrow t'$ due to $e \leftrightarrow t$.

For each place such as $a$ which has restoring transitions there is a set of cycles, one for each restoring transition, $t$ and $u$. These cycles are characterised as $(a, t \cdot u \cdot v)$ and $(a, u \cdot u \cdot v)$, i.e. the place $a$ and a sequence comprising one of the restoring transitions and all the decreasing transitions.

For a place such as $c$ which has decreasing transitions, $t, u, v$, but not restoring transitions, there is a single cycle characterised as $(c, t \cdot u \cdot v)$, i.e. the place and a sequence comprising all its post-transitions. Note that the transitions comprising a cycle constitute a set of transitions which all mutually interfere at its place; e.g. for cycle $(a, t \cdot u \cdot v)$, the transitions $t, u, v$ all interfere at $a$.

Each such cycle, e.g. $(a, t \cdot u \cdot v)$, derived from $N$ generates in $N'$ a cyclically connected set of places and tans, comprising a place $(a, t, p)$ for each $p$ in the sequence $t \cdot u \cdot v$, with each such place being connected to $p'$ in the same way as $a$ is connected to $p$ in $N$; E.g. (for $p = t$), $t' \leftrightarrow (a, t, t)$ since $t \leftrightarrow a$, and (for $p = u$), $u' \leftrightarrow (a, t, u)$ since $u \leftrightarrow a$. Also one of the places
Fig. 7 - The Construction for ID nets
generated from the cycle, \( \langle a,t,t \rangle \), is connected by an incoming arc to each transition \( (x' \text{ or } y') \) for which the corresponding transition \( (x \text{ or } y) \) has \( a \) as a post- but not pre-place.

In the case of a place, e.g. \( b \), with only restoring transitions, \( w \) and \( x \), this construct gives single-tau cycles, \( \langle b,w \rangle \) and \( \langle b,x \rangle \), and may introduce a disconnection in the net as occurs in this example around transition \( x' \).

3.4.2. Marking

Construct an initial marking \( M_0' \) for \( N' \) from a given initial marking \( M_0 \) of \( N \).

For each carried forward place, e.g. \( e' \), there is the same initial marking, \( M_0'(e') = M_0(e) \).

For each cycle, e.g. \( \langle a,t,u,v \rangle \), one of the generated places, \( \langle a,t,t \rangle \) has the marking of the place from which the cycle is derived, \( M_0'(\langle a,t,t \rangle) = M_0(a) \), and the remaining generated places have marking 0.

3.4.3. Locality

Construct a locality function, \( L' \) for \( N' \) from a given locality function \( L \) for \( N \).

If a transition of \( N' \) has no pre-place, e.g. \( y' \), we select its machine from the corresponding transition of \( N \), \( L'(y') = \{ \text{sel}(L(y)) \} \), as for the ICD construction.

The remaining transitions of \( N' \) are partitioned into clusters comprising a goal transition, say \( t' \), and all tau transitions, \( \langle t,t,a \rangle \) and \( \langle t,t,c \rangle \), which share a pre-place with that goal transition. All transitions in a cluster with a particular goal transition, \( \langle t' \rangle \), have the same locality for which we select a machine \( \langle C \rangle \) from those \( \langle A,C \rangle \) of the places \( \langle a,c \rangle \) which dominate the interference set \( \langle t,u,v \rangle \) of \( t \); i.e. \( L'(t') = \{ \text{sel}(L(\uparrow IS(t))) \} \). (This is the part of the construction which depends on \( N \) being ID.)

Clearly all localities so constructed are singleton. Also all post transitions of a place have the same locality since they are in the same cluster. Thus \( L' \) is a proper locality function and \( \langle N';L' \rangle \) is distributed.

3.4.4. Behaviour

Show that \( \Sigma = (N',M_0',L') \) satisfies the requirements of Defn.2.2 for being a locality-respecting simulation of \( \Sigma = (N,M_0,L) \) under the transition correspondence \( f(p) = p' \).

The required state correspondence \( \beta' \) uses the notion of the marking of a cycle, say \( \langle a,t,u,v \rangle \), as the sum of the markings of all the places generated from it, i.e. \( M'((a,t,t)) + M'((a,t,u)) + M'((a,t,v)) \). We define \( \beta' \) as \( (M',M) \) is in \( \beta' \) if (i) for a place \( s \) of \( N \) which defines a set of cycles, the marking of each such cycle equals \( M(s) \); (ii) for other places, e.g. \( s = e \), \( M'(s') = M(s) \).

It is straightforward to show that \( \beta' \) is a surjection from reachable markings of \( \Sigma' \) to reachable markings of \( \Sigma \) and that it satisfies the requirements of Defn.2.3. for \( \Sigma' \) to be a locality-respecting simulation of \( \Sigma \). The important case for the latter is a firing of a transition of \( \Sigma \) when the corresponding goal transition of \( \Sigma' \) is not enabled. E.g., for the markings shown for Fig.7, \( u \) can fire in \( \Sigma \) (\( \text{(ii)} \)), but \( u' \) is not enabled in \( \Sigma' \) (\( \text{(iii)} \)). The required firing of the goal transition, \( u' \), can be obtained by the firing of tau transitions within those generated by cycles derived from the pre-places of the corresponding transition, \( u \), in \( N \). For this case, the required tau firings are \( \langle t,t,c \rangle \), \( \langle t,t,a \rangle \) and \( \langle w,w,a \rangle \). That
firing sequence of taus, and then the $u'$, must be locality-respecting, i.e., the localities of those transitions for $\Sigma'$ must be contained in the locality of $u$ ($\{A, C\}$). For the goal transition, $u'$, $L'(u')$ is defined to be from a pre-place of $u$ which is necessarily in $L(u)$ (Prop.2.1(d)). For any of the tau transitions involved, say $(t,t,c)$, we have the property that its locality ($C$) is: from a set of places ($a$ and $c$) each of which is necessarily a pre-place of $u$; and thus also within $L(u)$.

3.5. Theorem 2

If a net $N = (S,T,F)$ is non-IDC then there is a located system, $\Sigma$, built on $N$ such that any distribution $\Sigma'$, of $\Sigma$ is busy.

Proof

Let $\Sigma = (N;L,M_0)$ with $L$ and $M_0$ satisfying:
(a) $\forall s \in S : M_0(s) = 1$
(b) $|MC| = |S|$ where $MC = L(T)$
(c) There is a bijection $l:S \rightarrow MC$
(d) $\forall t \in T : L(t) = \bigcup_{r \in t} l(r)$; and thus
(e) $\forall s \in S : L(s) = \{l(s)\}$
(f) $\forall t \in T$, $s \in S : s \in L(t) \Rightarrow s \in t$

Assume $\Sigma' = (S',T',F',L',M_0')$ is a non-busy distribution of $\Sigma$ with respect to some $f:T \rightarrow T'$.

Take any $t_b \in T$ such that $t_b \neq \emptyset \land \uparrow IC(t_b) = \emptyset$ (negation of IDC, Defn.3.1(f))
This implies $IC(t_b) \cap \{t_b\} = \emptyset$
Let $L'(f(t_b)) = \{l(s)\}$ for some $s \in S$
Take any $t_c \in IC(t_b) \cap \{t_b\}$
By the definition of $IC(.)$, there are transitions $t_0,t_1,...,t_n \in T$, and places $s_1,s_2,...,s_n \in S$, such that $t_0 = t_b$, $t_n = t_c$, and $\forall i \in \{1...n\} : t_i = t_{i-1} \cdot s_i \cdot t_i$
Also, by property (a) of $\Sigma$, $\forall i \in \{1...n\} : M_0(s_i) = 1 \land M_0(t_{i-1}) = \land M_0(t_i)$

Thus, by non-busy-ness of $\Sigma'$, in Theo. 1(c), $\forall i \in \{1...n\} : L'(f(t_{i-1})) = L'(f(t_i))$
Thus, $L'(f(t_0)) = L'(f(t_n))$, that is $L'(f(t_0)) = L'(f(t_b)) = \{l(s)\}$
But $L'(f(t_c)) \subseteq L(t_c)$, by Prop.2.4(c), and thus $l(s) \subseteq L(t_c)$;
\text{hence} $s \in \uparrow IC(t_c)$, by (f) above
\text{But also similarly,} $s \in \uparrow IC(t_b)$
Thus, $\forall t \in IC(t_b) : s \in \uparrow IC(t_b)$
I.e. $s \in \uparrow IC(t_b)$, which contradicts $\uparrow IC(t_b) = \emptyset$, and thus no such $\Sigma$ exists.

3.5.1. Theorem 3

If a net $N = (S,T,F)$ is non-ID then there is a located system, $\Sigma$, built on $N$ such that there is no distribution of $\Sigma$.

Proof

Let $\Sigma = (N;L,M_0)$ with (a) - (f) as in Theo. 2
Assume $\Sigma' = (S', T', F', L', M_0')$ is a distribution of $\Sigma$ with respect to some $f: T \rightarrow T'$.

Take any $t_b \in T$ such that $\new{t}_b = \emptyset \land \uparrow IS(t_b) = \emptyset$ (negation of ID, Defn.3.1(e))

This implies $IS(t_b) \varnothing \{t_b\} = \emptyset$

Let $L'(f(t_b)) = \{l(s)\}$ for some $s \in S$

Take any $t_c \in IS(t_b) \varnothing \{t_b\}$, then we have, by the definition of $IS(C)$$$
\exists r \in S: t_c \leadsto r \leadsto \cdots \leadsto t_b$

Now by $M_0(s_a) = 1 \land M_0(t_b) \land M_0(t_c)$, in Theo. 1(a)/(b), we have

$L'(f(t_b)) \subseteq L(t_b)$; and thus $l(s) \in L(t_b)$.

Also, by Prop.2.4(c), we have $L''(t_b) \subseteq L(t_b)$; and thus $l(s) \in L(t_b)$.

That is $\forall t \in IS(t_b): l(s) \in L(t)$; and thus, by (f), $s \notin t$

That is, $s \in \uparrow IS(t_b)$, which contradicts $\uparrow IS(t_b) = \emptyset$, and thus no such $\Sigma$ exists.

4. Discussion

4.1. Interleaving Semantics

In Fig.3, the replication of $x_T$ of (i) as two separate places in (ii), which is crucial for obtaining the distribution, is only valid because we are using an interleaving semantics, i.e. the required simulation relation between (i) and (ii) is in terms of sequences of single transitions. In a true concurrency semantics (ii) does not have the same behavior as (i) in that: for (ii) $t_2'$ and $t_3'$ are concurrent, but the corresponding $t_2$ and $t_3$ of (i) are not.

4.2. Relationship to Free/Asymmetric Choice

The central notion of interference used in characterising distributable nets plays a similar role for interleaving semantics as does the notion of choice for true concurrency semantics. This observation motivates making some comments on the relationship between the I(C)D nets identified here and the (extended) free choice, (E)FC, and asymmetric choice, AC, classes.

\hspace{1cm}

![Diagram](attachment:image.png)

Fig. 8 - For Proof that AC is in ICD

Firstly, it is simple to show that AC is contained in ICD (which is contained in ID); i.e. any AC net $N$ is ICD: (see Fig.8)

For any interference class $I$ of $N$, with $|I| > 1$, take a proper subset $T_1$ of $I$ which has a dominating place $s_1$. By the definition of interference class, there must be a $t_2$ in $I \cup T_1$ which has a common pre-place $s_2$ with some $t_1$ in $T_1$. By the AC property: either the
post-set of \( s_2 \) contains the post-set of \( s_1 \), which includes \( T_1 \), and thus \( s_2 \) dominates the extended subset, \( T_1 \cup \{ t_2 \} \), of \( I \); or the post-set of \( s_1 \) contains the post-set of \( s_2 \) and thus \( s_1 \) dominates that extended subset of \( I \). This constitutes the induction step in showing that \( I \) has a dominating place; the base case, singleton \( T_1 \), being obvious. (For the remaining interference classes, \( |I|=1 \), the ICD requirement is trivially satisfied.)

The extension of AC to ICD involves two relaxations:

(a) In both ICD and AC, if two places, \( d \) and \( e \), have intersecting post-sets, the union of their post-sets must be dominated - for AC the dominating place must be one of \( d \) or \( e \); whereas for ID it may be a third place, \( c \).

(b) ICD uses interference as the significant relationship between transitions rather than the conflict of AC.

Point (a) is very particular to the notion of distributability, and the extension of AC in this way seems unlikely to have any other significance. However point (b) is perhaps slightly more interesting.

It is possible to extend both EFC and AC by replacing the notion of conflict by that of interference. One way to do this is to define the interference set of a place \( s \), \( IS(s) \), as the set comprising every transition which interferes with some other transition at \( s \). \( IS(s) \) is the post-set of \( s \), unless \( s \) is sticky, in which case \( IS(s) \) is empty.) We can then characterise free-interference, FI, analogously to extended-free-choice, as: \( IS(s_1) \) and \( IS(s_2) \) having non-empty intersection implies \( IS(s_1) = IS(s_2) \). Similarly, asymmetric-interference, AI, would be: \( IS(s_1) \) and \( IS(s_2) \) having non-empty intersection implies \( IS(s_1) \) contains \( IS(s_2) \) or vice-versa.

For some properties of AC there are similar properties for AI. This arises from general properties of the (sticky place replication) construction shown in Fig.6, and described in Section 4.3. For two systems \( \Sigma \) (e.g. Fig.6(i)) and \( \Sigma' \) (e.g. Fig.6(ii)) related by this construction, and built on nets \( N \) and \( N' \):

(i) \( \Sigma \) and \( \Sigma' \) have exactly the same behavioural properties (other than in their concurrency).

(ii) If \( N \) is AI then \( N' \) is AC:
If \( N \) is AI then deleting all sticky places in \( N \) (\( a \) and \( b \) for Fig.6(i)) gives a net \( N'' \) which is AC (because every place \( s \) of \( N'' \) is non-sticky for which \( IS(s) = s^* \), and thus the AI property between a pair of places is the AC property between them). Now \( N' \) is obtained from \( N'' \) by adding in places \( (a,t), (a,u), \) etc.) and their connectivity to transitions of \( N'' \), such that each such additional place has only a single post transition, and therefore cannot introduce a violation of the AC property of \( N'' \).

From the above two observations it is easily shown that: for a system built on an AI net, liveness and place-liveness are equivalent; due to the corresponding property of AC nets [1].

4.3. Locality Assignment Properties

The notions of distributability have been locality-independent in requiring a distribution for any locality assignment satisfying the requirement of Defn.2.1(b) (that any set of transitions with a common pre-place also have a common machine). It will often be the
case that the distribution constraints given by a locality function are a second-order con-
sideration - they may be added after the behaviour has been finalised, and different
operating environments may require different distributions. Thus we consider it
significant to have considered locality-independent distributability.

The results we have presented can be seen as dealing with one point in the following
hypothesised hierarchy of increasingly restrictive properties of locality assignments, and
decreasingly restrictive properties of net structure, needed to guarantee distributability
of any such net with respect to any such locality function.
(a) For no locality assignment restrictions (i.e. dropping the requirement of Defn.2.1(b)) -
a net is distributable (and tau-less distributable) with respect to any locality assignment
iff it is interference-free, i.e. no transition can disable any other transition.
(b) The locality assignment restriction of Defn.2.1(b) - giving the results of this paper for
nets with the ID/ICD properties. (This restriction is simple to satisfy by unrestrictedly
assigning a locality to each place, and taking the locality of a transition as the union of
the localities of its pre-places.)
(c) For no restrictions on net structure - any net is distributable with respect to any
locality assignment with the property that transitions of an interference set have a common
machine; any net is tau-less-distributable with respect to any locality assignment
with the property that transitions of an interference class have a common machine.

4.4. Behavioural Distributability

Up to now we have considered "structural" distributability, in that distributability is
considered a property of a net, which pertains to any marking; rather than a
"behavioural" distributability which would be a property of a system and only pertains to
states reachable from its specific initial marking. However, there are systems which are
behaviourally distributable, despite having an underlying net which is not structurally
distributable. Trivially, any system with a deadlocked initial marking is behaviourally
distributable regardless of the structure of its underlying net. A more significant exam-
ple is that shown in Fig.9(i).

In this case the net itself is not (structurally) distributable because: interferences of $t_4$
with $t_2$, and $t_4$ with $t_5$ mean that $t_4$ can be implemented neither on B nor on A. How-
ever with the given initial marking the former structural interference never actually
occurs as a behavioural interference. A distribution is possible for that initial marking,
as shown in Fig.9(ii).

In terms of program structure, this is a simple instance of general situation where part
of the program on one machine ($A::(\ldots)$) assigns to a variable ($x$) and is followed by
perhaps multiple parallel parts on different machines ($B::(\ldots)$ being one such) which
only read that variable, and this sequence generally being within a loop. Such a struc-
ture is not structurally distributable but is behaviouraly distributable.

We make the following observations

(i) We can define BID and BICD for a system in the same way as ID and ICD for a net,
but with a behavioural notion of interference between two transitions which depends on
a reachable marking at which both transitions are enabled.
(ii) We hypothesize that: if a system is not BID then there are locality functions for which it has no distribution; if a system is not BICD then there are locality functions for which it has no tau-less distribution.

(iii) We also hypothesise that for an IB(C)D system it is possible to construct a (tau-less) distribution, by appropriate replication of places (such as $x_F$ of Fig.9(i)) at which there is no behavioural interference. The practical difficulty is that showing the system is IB(C)D and identifying the places which can be replicated, requires analysis of the behaviour of the system rather than its net structure. Where the system is generated from a program, as in Fig.9, it should be straightforward to recognise all situations where the control structure ensures that the structural interference at a data place ($x_F$) cannot be a behavioural interference. However, there can be situations (more difficult to identify) where details of the data accessing structure happen to make a structural interference not a behavioural interference.

4.5. Distributed Implementations of Input and Output Guards

We can apply our results to OCCAM, by producing a Petri net semantics in a similar way as in the Appendix for the programming notation used for examples (and this has been done in [2]). We find that any net from an OCCAM program has a tau-less distribution, due to only input guards being allowed, i.e. a transition which is the synchronisation of an input and an output, such as $t_2$ of Fig.1(i) or Fig.2(i), can be involved in one choice (Fig.2(i)), but not two choices (Fig.2(ii)). Allowing both input and output guards would allow the construction of examples such as in Fig.1(i), which by our analysis has no distributed implementation.
In [6] there is a survey of schemes for distributedly implementing communication structures with both input and output guards, and the presentation of what is claimed to be a satisfactory distributed implementation scheme for such structures. That survey gives a list of criteria for a distributed implementation to be satisfactory, the criterion relevant here being that "there should only be a small number of processes involved in synchronising two processes with matching communication commands". In terms of our formalism, this criteria is that the "locality respecting" part of Defn 2.3(ii)(a) be as follows, where \( \omega \) is a sequence of transitions implementing a communication transition \( t \):
\[ |L'(\omega)| < D, \text{ where } D \text{ is "a small number".} \]

In contrast, we have a precise requirement:
\[ L'(\omega) \subseteq L(t) \]
which appears to be violated by every implementation scheme mentioned in [6].

**APPENDIX - Programming Notation**

The elements of the programming notation used in the examples, and their translation to nets, are described below.

**Boolean Expressions and Variables**

\( T, F, \land, \lor, \neg \) - The boolean constants and operators.

Every variable \( x \) is of type boolean, and is represented as two places ("data places") \( x_T \) and \( x_F \), at most one of which is marked depending on the value of \( x \) (if any.)

**Primitive Processes**

Each primitive process (e.g. an assignment) has an "entry" place and an "exit" place (generically "control places") connected as pre- and post-places of a number of possible transitions for the behaviour of the process. There is one such transition for every combination of variable values for which the process can execute, each transition being appropriately connected to data places for variables which it uses. For example, in Fig.9(i), the assignment \( x := \neg x \) produces two transitions: \( t_2 \) for the case \( x = F \); and \( t_3 \) for the case \( x = T \). Transition \( t_2 \) requires a token on \( x_F \) which it moves to \( x_T \), and vice-versa for \( t_3 \). Both \( t_2 \) and \( t_3 \) have \( c_0 \) as entry place and \( c_1 \) as exit place.

\( x = T \) or \( x = F \) - Declaration of \( x \) with an initial value. E.g., in Fig.3(i), \( x = T \) giving transition \( t_0 \). Note that prior to this transition \( x \) is without a value, and thus (unlike assignment) there is only a single transition.

\( e \) - Condition, being an expression which must evaluate to \( T \) for the process to execute. E.g., in Fig.4(i), the \( \neg x \) produces transition \( t_1 \) which depends on \( x_F \) being marked, and maintains that marking. The condition \( T \) acts as a skip, e.g. \( t_1 \) in Fig.1(i).

\( e \& ch! \) or \( e \& ch? \) - Communication(output or input), the sending or receiving of a control signal on channel \( ch \) which as in CSP only executes if \( e \) evaluates to \( T \) and a matching communication simultaneously executes. (Omitting the \( e \& \) implies \( T \& \).)

\( e \& x := f \) - Assignment, for which the assignment of the value of expression \( f \) to variable \( x \) can occur if \( e \) evaluates to \( T \). (Omitting the \( e \& \) implies \( T \& \).) E.g., in
Fig. 4(i), the \( \neg y \land \neg x \land x := T \) gives the transition \( t_2 \) which requires both \( x \) and \( y \) to be false, and negates \( x \) (placing a token on place \( x_T \), which is not shown.)

**Constructors**

The process constructors give compositions on the nets representing the component processes \((p\text{ and }q)\). The essentials of those compositions are the manipulation of entry and exit places (i.e. the control flow), although there are also manipulations of data places and transitions as described after introducing the constructors. These manipulations involve the "merging" of two elements (places or transitions), producing a result element with the connectivity of both merges.

\[ p ; q \text{ - Sequencing} \] in which the exit place of \( p \) is merged with the entry place of \( q \). E.g., in Fig. 2(i), the \( T;ch! \) is the composition of the net \( c_b \rightarrow t_b \rightarrow c_f \) (the \( T \)) with \( c_d \rightarrow t_2 \rightarrow c_f \) (the \( ch! \)). The \( c_d \) places from both sides are merged to give the result net \( c_b \rightarrow t_b \rightarrow c_d \rightarrow t_2 \rightarrow c_f \) for which \( c_b \) and \( c_f \) are respectively the entry and exit places.

\[ p \parallel q \text{ - Alternative} \] in which the entry places of \( p \) and \( q \) are merged, and so are their exit places. E.g., for \( T \parallel ch? \) of Fig. 1(i): the \( T \) is \( c_c \rightarrow t_1 \rightarrow c_g \); the \( ch? \) is \( c_c \rightarrow t_2 \rightarrow c_h \); and the result is the structure shown of these four elements, with \( c_c \) and \( c_e \) as respectively the entry and exit places. The initial primitive process of \( p \) (or \( q \)) acts as the "guard" for that process in this alternative. It is the ability to have an assignment to a shared variable as a guard in an alternative which allows construction of examples such as Fig. 4(i) which have only busy distribution.

\[ p \parallel q \text{ - Parallel} \] in which there is no control place merging. E.g. in Fig. 1(i): the two components of the \( \parallel \) are the left structure, with entry and exit \( c_a \) and \( c_g \); and the right structure with entry and exit \( c_h \) and \( c_e \); and the result as shown has two entry places \( c_a \) and \( c_h \), and two exit places \( c_g \) and \( c_e \). The presence of a \( \parallel \) constructor within other constructors means in general that the control place mergings for those must be defined in terms of multiple entry and exit places (as defined in [2]), but this does not concern us here.

\[ m :: p \text{ - Placement} \] declaration which includes machine \( m \) in the locality of every transition obtained from a primitive process within \( p \), unless that is governed by a nested placement declaration. E.g., in Fig. 9(i), \( t_2 \) includes \( A \) in its locality since it derives from a process, \( x := \neg x \), governed by \( A :: \); whereas \( t_3 \), is similarly given its locality by \( B :: \). The localities given in this way to transitions are their initial localities, which may be extended by compositions as described below.

**Other aspects of the Net compositions**

There are a number of circumstances in which data places and transitions are merged, and localities of transitions are extended (in order to satisfy the requirement that transitions with a common pre-place have a common machine.)
(i) **Transition Merging** (|| constructor). Two nets being composed may contain the two sides of a communication. E.g., in Fig.1(i), both sides of the || have a transition for \( t_2 \), which is \( ch? \) with locality \( \{A\} \) on the left side, and \( ch! \) with locality \( \{B\} \) on the right side. These transitions from component processes are merged in the result process with the locality of the merged transition being the union \( \{A,B\} \) of the localities of the mergees.

(ii) **Data Place Merging** (all constructors). The nets being composed may contain a data place \( d \) for the same variable value, e.g. in Fig.4(i), both sides of the || have a \( x_F \) as such a \( d \). The two instance of \( d \) are merged and for every transition \( t \) which decreases \( d \)’s marking, i.e. \( d \rightarrow t \rightarrow d \), the locality of \( t \) is added to that of every post transition of \( d \). In the || of Fig.4, the merging at \( x_F \) adds the prior locality \( \{A\} \) of \( t_2 \) to \( t_3 \); and the merging at \( y_F \) adds the prior locality \( \{B\} \) of \( t_3 \) to \( t_2 \).

(iii) **Entry Place Merging** ([ constructor). In \( p \)[\( q \) there is the merging of the entry place \( e \) of \( p \) and \( q \). Although not illustrated by any examples in the figures, each post-transition of \( e \) in either side acquires the locality of every post-transition of \( e \). E.g., in \( (A :: (x; z := 1)) || (B :: (y; w := 1)) \), the transitions for \( x \) and \( y \) would both have localities \( \{A, B\} \), whereas those for \( z := 1 \) would have locality \( \{A\} \), and those for \( w := 1 \) would have locality \( \{B\} \).

(iv) **Variable Declaration**. If on encountering the declaration of a variable, one of its data places has no decreasing transition, then to every post transition is added the locality of the declaring transition. E.g. in Fig.3, the transitions \( t_2 \) and \( t_3 \) acquire the \( C \) of the transition \( t_0 \) which declares \( x_F \). This is necessary to make the net being constructed a proper located net, and in general a sub-program with un-declared variables might not be such.

(The above is the compositional construction of the locality function for a net. It should also be possible to compositionally construct the actual distributed implementation of the net, with piece-meal use of the constructions illustrated in Fig.6 and Fig.7.)

**Acknowledgements**

It is a pleasure to acknowledge fruitful discussions with various participants in the DEMON project, and the efforts and useful comments of three anonymous referees.

**References**


