PB - 174

Huber et al.: Reachability Trees for HL-Nets

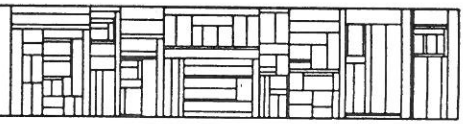
ISSN 0105-8517

Towards Reachability Trees for High-level Petri Nets

Peter Huber
Arne M. Jensen
Leif O. Jepsen
Kurt Jensen

DAIMI PB - 174
May 1985

DATALOGISK AFDELING
Bygning 540 - Ny Munkegade - 8000 Aarhus C
tlf. (06) 12 83 55, telex 64767 aausci dk
Matematisk Institut Aarhus Universitet



TRYK: DAIMI/RECAU

CONTENTS

TOWARDS REACHABILITY TREES FOR HIGH-LEVEL PETRI NETS

by
Peter Huber, Arne M. Jensen, Leif O. Jepsen, and Kurt Jensen
Computer Science Department, Aarhus University
Ny Munkegade, DK-8000 Aarhus C, Denmark

1. INTRODUCTION

High-level Petri nets [1, 4, 5, 6, 9] have been introduced as a powerful net type, by which it is possible to handle rather complex systems in a succinct and manageable way. The success of high-level Petri nets is undebatable when we speak about description, but there is still much work to be done to establish the necessary analysis methods. In [1,4,5] it is shown how to generalize the concept of place-invariants (s-invariants), from place/transition-nets (PT-nets) to high-level Petri nets (HL-nets). Analogously, [9] shows how to generalize transition-invariants (t-invariants). Our present paper constitutes the first steps towards a generalization of reachability trees, which is one of the other important analysis methods known for PT-nets [2, 7, 8].

The central idea in our paper is the observation, that HL-nets often possess classes of equivalent markings. As an example the HL-net describing the five dining philosophers in [4] has an equivalence-class consisting of those five markings in which exactly one philosopher is eating. These five markings are interchangeable, in the sense that their subtrees represent equivalent behaviours, where the only difference is the identity of the involved philosophers and forks. If we analyze one of these subtrees, we also understand the behaviour of the others.

This paper presents a proposal how to define reachability trees for HL-nets (HL-trees). For PT-nets the reachability trees in [2,7,8] are kept finite by means of covering markings (introducing ω-symbols) and by means of duplicate markings (cutting away their subtrees). For HL-trees we reduce by means of covering markings and by means of equivalent markings (for each equivalence-class we only develop the subtree of one node, while the other equivalent nodes become leaves of the tree). Reduction by equivalent markings is a generalization of reduction by duplicate markings. We describe an algorithm which constructs the HL-tree. The algorithm can easily be automated and we will soon start the work on an implementation. The constructed HL-trees turn out to be considerably smaller than the corresponding PT-trees (reachability trees for the equivalent PT-nets, obtained from the HL-nets by the method described in [4]).

The rest of the paper is organized as follows. Section 2 reviews the formal definition of HL-nets and ω-bags. In section 3 HL-trees are introduced by means of an example. Section 4 contains the formal definition of HL-trees and the algorithm to construct them. Section 5 discusses how to establish proof rules, by which properties of HL-nets can be derived from properties of the corresponding HL-trees. Section 6 contains two examples where HL-trees are constructed and compared with the corresponding PT-trees.

2. A BRIEF REVIEW OF HL-NETS AND DEFINITION OF ω-BAGS

In this section we review the basic concepts of HL-nets [6] and we generalize bags (allowing their elements to have multiplicity ω, representing an unlimited number of occurrences). Bags (multisets) are represented as formal sums as shown in [6]. By BAG(S) we denote the set of all finite bags over a non-empty set S. By $[A\to B]_L$ we denote the set of all linear functions with domain A and range B.

Definition An HL-net is a 6-tuple $H=(P,T,C,I_-,I_+,m_0)$ where

(1) P is a set of places
(2) T is a set of transitions
(3) $P\cap T = \emptyset$, $P\cup T \neq \emptyset$
(4) C is the colour-function defined from $P\cup T$ into non-empty sets
(5) I_- and I_+ are the negative and positive incidence-function defined on $P\times T$, such that $I_-(p,t)$, $I_+(p,t)\in[BAG(C(t))\to BAG(C(p))]_L$ for all $(p,t)\in P\times T$
(6) m_0, the initial marking, is a function defined on P, such that $m_0(p)\in BAG(C(p))$ for all $p\in P$. ■

Throughout this paper we assume P, T, C(p) and C(t) to be finite for all $p\in P$ and $t\in T$. A marking of H is a function m defined on P, such that $m(p)\in BAG(C(p))$ for all $p\in P$. A step of H is a function x defined on T, such that $x(t)\in BAG(C(t))$ for all $t\in T$. The step x has concession in the marking m iff $\forall p\in P: \sum_{t\in T} I_-(p,t)(x(t)) \leq m(p)$. A marking is dead iff only the empty step has concession in it.

When x has concession in m, it may fire and thus transform m into a directly reachable marking m', such that

$$\forall p\in P: m'(p) = m(p) - \sum_{t\in T} I_-(p,t)(x(t)) + \sum_{t\in T} I_+(p,t)(x(t)).$$

We indicate this by the notation $m[x>m'$. In this paper we will only consider steps which map a single transition $t \in T$ into a single firing-colour $c \in C(t)$, while all other transitions are mapped into the empty bag. Such a step is denoted by (t,c), where we sometimes omit the brackets. When for $n \geq 0$, $m[t_1,c_1>m_1[t_2,c_2>m_2 \ldots m_{n-1}[t_n,c_n>m'$ the sequence $\sigma = (t_1,c_1)(t_2,c_2)\ldots(t_n,c_n)$ is a <u>firing sequence</u> at m and m' is (forward) <u>reachable</u> from m, which we shall denote by $m[\sigma>m'$. By $R(m)$ we denote the set of all markings which are reachable from m. An HL-net is <u>bounded on place $p \in P$ and colour $c \in C(p)$</u> iff $\exists k \in \mathbb{N} \forall m \in R(m_0): m(p)(c) \leq k$, and it is <u>bounded</u> iff it is bounded on all places and all colours.

<u>Definition</u> An ω-bag over a non-empty set S is a function $b: S \to \mathbb{N} \cup \{\omega\}$ and it is represented as a formal sum $\sum_{s \in S} b(s)s$, where $b(s) \in \mathbb{N} \cup \{\omega\}$. ■

$b(s)$ represents the number of occurrences of the element s. If $b(s) = \omega$ the exact value is unknown and may be arbitrarily large. An ω-bag b over the set S is <u>finite</u> iff its support $\{s \in S \mid b(s) \neq 0\}$ is finite. The set of all finite ω-bags over the non-empty set S will be denoted by $\omega\text{-BAG}(S)$. Summation, scalar-multiplication, comparison, and multiplicity of ω-bags are defined in the following way, where $b_1, b_2, b \in \omega\text{-BAG}(S)$, $n \in \mathbb{N}$ and $m \in \mathbb{N} \cup \{\omega\}$:

$$\omega + m = \omega \qquad \omega > n \qquad\qquad m\omega = \begin{cases} \omega \text{ if } m \neq 0 \\ 0 \text{ if } m = 0 \end{cases}$$

$$\omega - m = \omega \qquad \omega \geq m$$

$$b_1 + b_2 = \sum_{s \in S} (b_1(s) + b_2(s))s \qquad m \times b = \sum_{s \in S} (mb(s))s$$

$$b_1 \geq b_2 \Leftrightarrow \forall s \in S: b_1(s) \geq b_2(s)$$

$$b_1 > b_2 \Leftrightarrow (b_1 \geq b_2) \wedge (\exists s \in S: b_1(s) > b_2(s))$$

When $b_1 \geq b_2$ we also define subtraction: $b_1 - b_2 = \sum_{s \in S} (b_1(s) - b_2(s))s$.

A function $F \in [S \to \text{BAG}(R)]$, where S and R are non-empty sets, can be extended uniquely to a linear function $\hat{F} \in [\text{BAG}(S) \to \text{BAG}(R)]$, called the <u>bag-extension</u> of F: $\forall b \in \text{BAG}(S): \hat{F}(b) = \sum_{s \in S} b(s) \times F(s)$.

Analogously we define the <u>ω-bag-extension</u> of $F \in [S \to \omega\text{-BAG}(R)]$ to be $\bar{F} \in [\omega\text{-BAG}(S) \to \omega\text{-BAG}(R)]$, where $\forall b \in \omega\text{-BAG}(S): \bar{F}(b) = \sum_{s \in S} b(s) \times F(s)$.

An <u>ω-marking</u> of H is a function m defined on P, such that $m(p) \in \omega\text{-BAG}(C(p))$ for all $p \in P$. The concepts of step, concession and reachability are generalized from markings to ω-markings by replacing the word "marking" by "ω-marking". An ω-marking m_1 <u>covers</u> another ω-

marking m_2, $m_1 \geq m_2$, iff $\forall p \in P: m_1(p) \geq m_2(p)$, and it strictly covers, $m_1 > m_2$, iff $m_1 \geq m_2 \wedge m_1 \neq m_2$.

3. INFORMAL INTRODUCTION TO REACHABILITY TREES FOR HL-NETS

In this section we give, by means of an example, an informal introduction to our notion of reachability trees for HL-nets. The basic idea of a reachability tree is to organize all reachable markings in a tree-structure where each node has attached a reachable marking, while each arc has attached a transition and a firing-colour (which transforms the marking of its source-node into the marking of its destination-node). Such a tree contains all reachable markings and all possible sequences of transition-firings. By inspection of the tree it is possible to answer a large number of questions about the system. However, in general the reachability tree described above will be infinite. For practical use it is necessary to reduce it to finite size. This is done by covering markings and by equivalent markings which is a generalization of duplicate markings. Reduction by covering markings and duplicate markings are well known from PT-trees. Reduction by equivalent markings is, however, a new concept suitable for HL-trees and this idea is the primary result of our paper.

Covering markings. When a node has a marking m_2, which strictly covers the marking m_1 of a predecessor, the firing sequence transforming m_1 into m_2 can be repeated several times starting from m_2[†]. Thus it is possible to get an arbitrarily large value for each coefficient which has increased from m_1 to m_2. In the tree we indicate this by substituting in m_2, the ω-symbol for each such coefficient. The situation is analogous to the idea behind the "pumping lemma" of automata theory and it means that some of the places can obtain an arbitrarily large number of tokens of certain colours.

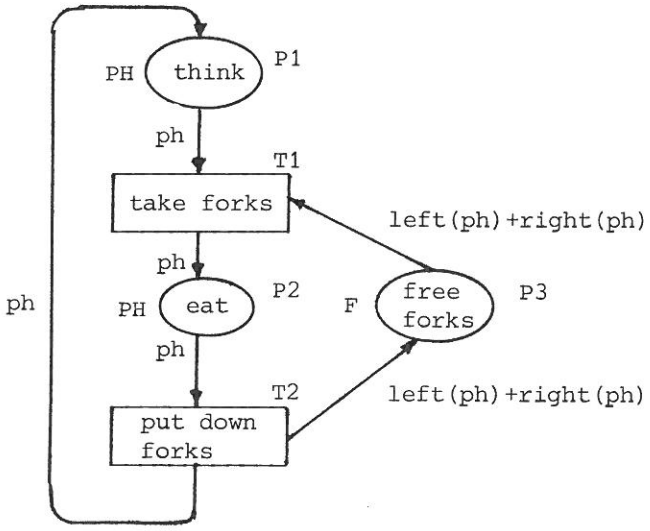
This kind of reduction results in a loss of information. In [8] it is shown, that if ω occurs in a PT-tree, it is not always possible to determine from the tree whether the net has a dead marking or not.

Duplicate markings. If there are several nodes with identical markings only one of them is developed further, while the others are marked as "duplicate". This reduction will not result in a loss of information

† If m_2 already contains ω the situation is more complicated, and it may be necessary to involve some extra firings, cf. the proof of lemma 3 in appendix 1.

because we can construct the missing subtrees from the one developed. Due to reduction by covering markings, two such subtrees may not be completely identical, but they will represent the same set of markings and firing sequences.

Equivalent markings. To introduce our notion of equivalent markings, we will now look at the HL-net for the five dining philosophers in [4]:



$PH=\{ph_1,ph_2,ph_3,ph_4,ph_5\}$
$F=\{f_1,f_2,f_3,f_4,f_5\}$
$left(ph_i) = f_i$
$right(ph_i) = f_{i\oplus 1}$

		T1	T2	m_0
		PH	PH	
P1	PH	-id	id	ΣPH
P2	PH	id	-id	
P3	F	-left -right	left +right	ΣF

Fig. 1: HL-net for the philosopher system.

We will now analyze the following markings:

$$m_1 = (ph_2+ph_3+ph_4+ph_5,\ ph_1,\ f_3+f_4+f_5)$$
$$m_2 = (ph_1+ph_3+ph_4+ph_5,\ ph_2,\ f_1+f_4+f_5)$$
$$m_3 = (ph_2+ph_4+ph_5,\ ph_1+ph_3,\ f_5)$$
$$m_4 = (ph_2+ph_3+ph_4+ph_5,\ ph_1,\ f_2+f_4+f_5)$$
$$m_5 = (ph_3+ph_4+ph_5,\ ph_1+ph_2,\ f_5).$$

By intuition we want m_1 and m_2 to be equivalent. The point is that we do not need to know the identity of eating philosophers, because all philosophers "behave in the same way". The marking m_3 contains a different number of eating philosophers and thus it is not equivalent to m_1 or m_2. However, two markings may be non-equivalent even though they have the same number of eating philosophers and the same number of free forks. In m_1 and m_2 the non-free forks are those belonging to the eating philosopher. This is not the case in m_4, and thus m_4 is not equivalent to m_1 or m_2. In m_5 the two eating philosophers are neighbours. This is not the case in m_3, and so these markings are not equivalent either.

To obtain equivalent markings we must demand that the identity of all philosophers and forks are changed by the same rotation. As an example, m_1 is obtained from m_2 by the rotation which adds 4 (in a cyclic way) to the index of each philosopher and fork.

To formalize the notion of equivalent markings we associate to the colour set PH the symmetry type "rotation" and we define a bijective correspondence between F and PH by a function $r \in [F \to PH]$, where $r(f_i) = ph_i$. Two markings m' and m" are equivalent iff there exists a rotation φ_{PH} of PH, such that

$$(*) \qquad \begin{aligned} m'(p) &= \overline{\varphi_{PH}}\,(m''(p)) && \text{for } p = P1, P2 \\ m'(P3) &= \overline{r^{-1} \circ \varphi_{PH} \circ r}\,(m''(P3)). \end{aligned}$$

In our example the markings m_1 and m_2 are equivalent because the rotation $\varphi_{PH} \in [PH \to PH]$, defined by $\varphi_{PH}(ph_i) = ph_{i \oplus 4}$, satisfies (*). On the other hand m_2 and m_4 are not equivalent. From the place P2 it is demanded that $ph_2 = \varphi_{PH}(ph_1)$, i.e. $\varphi_{PH}(ph_i) = ph_{i \oplus 1}$, but this does not work at P3:

$$m_2(P3) = f_1 + f_4 + f_5 \neq f_1 + f_3 + f_5 = \overline{r^{-1} \circ \varphi_{PH} \circ r}\,(m_4(P3)).$$

As a generalization of reduction by duplicate markings we will now reduce the reachability tree by equivalent markings: Only one element of each class of equivalent markings is developed further, and when a marking has several direct successors which are equivalent, only one of them are included in the tree.

Figure 2 shows an HL-tree obtained for the philosopher system. In the initial marking transition T1 can fire in all colours of PH producing five equivalent markings of which only one is included in the tree, while the existence of the others are indicated by the label attached to the corresponding arc. If we only reduced by covering markings and duplicate markings, the tree would have had 31 nodes (and exactly the same tree structure as the PT-tree corresponding to the equivalent PT-net).

The relation of equivalent markings is determined by the persons who analyze the system, and it must respect the inherent nature of the system. In the philosopher system, rotation is the suitable symmetry type. But in the telephone system of [6] arbitrary permutation would be the suitable symmetry type (since there is no special relation between a phone number and its nearest neighbours). In general, several symmetry types (rotation, permutation or identity-function) may be involved in the same system (for different colour sets).

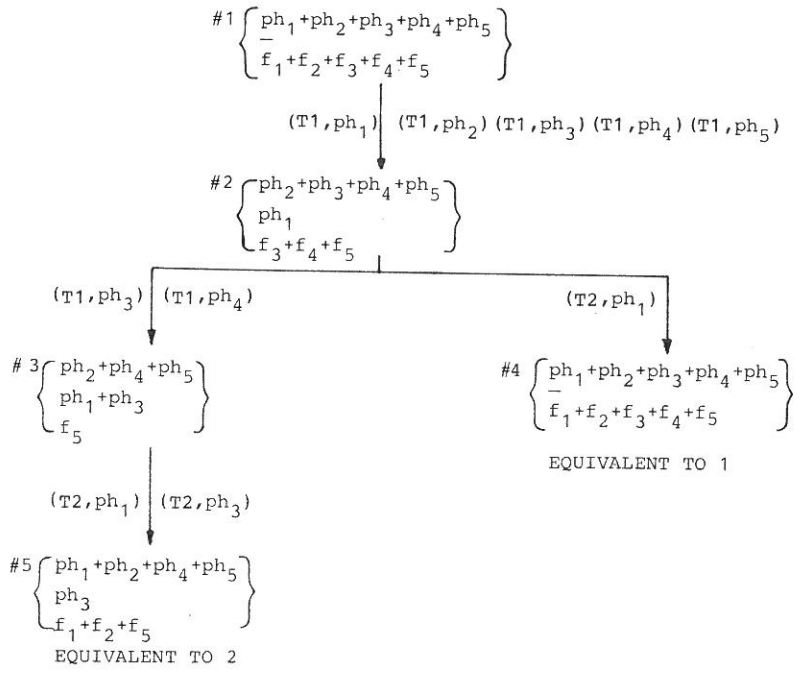


Fig. 2: HL-tree of the philosopher system. It is reduced by covering markings (none in this tree) and equivalent markings.

When the relation of equivalent markings is defined in a sound way (to be formalized in section 5), the reduction by means of covering markings and equivalent markings does not result in a loss of more information than reduction by covering markings and duplicate markings only. This means, that all net properties which can be proved by means of the PT-tree of the equivalent PT-net can also be proved by means of our (much smaller) HL-tree.

4. DEFINITION OF REACHABILITY TREES FOR HL-NETS

In this section we consider a fixed HL-net $H=(P,T,C,I_-,I_+,m_0)$.

Definition The set of colour sets $\{C(x) \mid x \in PUT\}$ is partitioned into three pairwise disjoint classes:

1) A is the set of atomic colour sets, where each Ca∈A has attached a symmetry type: sym(Ca)∈{permutation,rotation†,identity}.

2) R is the set of related colour sets, where each Cr∈R is related to an atomic colour set Ca by a bijective function r∈[Cr→Ca].

3) Π is the set of product colour sets, where each Cπ∈Π is the cartesian product of atomic and related colour sets. ■

Definition A symmetry (allowed by the given partition) is a set of bijective functions $\varphi = \{\varphi_C\}_{C\in A\cup R\cup\Pi}$ where $\varphi_C \in [C\to C]$ for all C and

(1) For all Ca∈A φ_{Ca} is a function of the kind specified by sym(Ca).

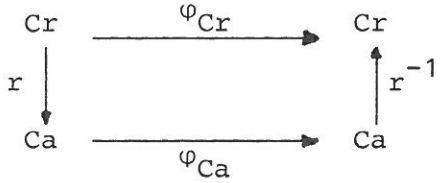
(2) For all Cr∈R, with r∈[Cr→Ca] $\varphi_{Cr} = r^{-1}\circ\varphi_{Ca}\circ r$.

(3) For all Cπ∈Π, with Cπ = C1×C2×...×Cn

$$\varphi_{C\pi} = \varphi_{C1}\times\varphi_{C2}\times\ldots\times\varphi_{Cn}.$$

The set of symmetries (allowed by the given partition) is denoted by Φ. It is finite since P, T, C(p) and C(t) are assumed to be finite for all p∈P and t∈T. ■

The definition of φ_{Cr} can be visualized by the following commutative diagram:



Since r is a bijection it follows that φ_{Cr} is a function of the kind specified by sym(Ca).

Technical remark: The definition of partition is here presented in its simplest form. In some cases (cf. the database example in section 6) it may be convenient/necessary to allow Π to contain subsets of cartesian products. If $C\pi = C^n\setminus\{(a,a,\ldots,a) \mid a\in C\}$ we define $\varphi_{C\pi} = (\varphi_C\times\varphi_C\times\ldots\times\varphi_C)|_{C^n}$, yielding a bijection on Cπ as requested. Secondly, in special cases, there can be sets in use to construct products in Π

† When an atomic colour set has rotation as symmetry type it must be a finite set, indexed by 1,2,...,n where n is the cardinality.

which are not themselves ordinary colour sets in the HL-net. These sets have to be included as atomic or related sets.

Given an ω-marking m, a firing sequence $\sigma = (t_1,c_1)(t_2,c_2)\ldots(t_n,c_n)$ and a symmetry $\varphi\in\Phi$, we define an equivalent ω-marking $\varphi(m)$ by

$$\varphi(m)(p) = \overline{\varphi_{C(p)}}(m(p)) \text{ for all } p\in P$$

and an equivalent firing sequence $\varphi(\sigma)$ by

$$\varphi(\sigma) = (t_1,\varphi_{C(t_1)}(c_1))(t_2,\varphi_{C(t_2)}(c_2))\ldots(t_n,\varphi_{C(t_n)}(c_n)).$$

Definition Two ω-markings, m_1 and m_2, of H are equivalent, $m_1\sim m_2$, iff there exists a symmetry $\varphi\in\Phi$ such that $m_1 = \varphi(m_2)$. It is easy to show that ~ is an equivalence relation. ■

We will draw attention to the fact, that given a net there are often several meaningful ways to define a partition. It is the user, who decides the partition and this choice determines the possible symmetries, and thus the relation of equivalent markings. In section 5 we define two soundness criteria for partitions and we establish four proof rules, which for sound partitions allow us to deduce properties of HL-nets from properties of the corresponding HL-trees.

Given the notions above we are now able to formalize the definition of reachability trees for HL-nets:

Definition The reachability tree, HL-tree, for an HL-net with an equivalence relation, ~, (specified by a partition) is the full reachability tree† reduced with respect to covering markings and equivalent markings:

(1) If a node y strictly covers a predecessor, z, then we assign $m_y(p)(c) := \omega$ for all $p\in P$ and $c\in C(p)$ satisfying $m_y(p)(c) > m_z(p)(c)$.

(2) Only one node in each (reachable) equivalence class of ~ is developed further. Only one node in a set of equivalent brothers is included in the tree. The other nodes are removed, but the arc to the included brother node contains information of their existence.

(3) Associated to each node is an ω-marking and a node-label. The node-label is a (possibly empty) sequence of status information, which may indicate that the marking is equivalent

† The full reachability tree contains all reachable markings and all firing sequences.

to the marking of an earlier processed node, covering the marking of a predecessor node, or dead.

(4) Associated to each arc from node n_1 to n_2 is an arc-label which is a list of firing information. Each element is a pair (t,c) where $t \in T$ and $c \in C(t)$. Each pair in the list has concession in the marking of n_1. Firing the first pair in the list results in the marking of n_2, whereas firing of the other pairs results in markings which are equivalent to the marking of n_2. ■

Now we will describe our algorithm to produce the HL-trees. To create a new node we use the operation "NEWNODE(m,ℓ)", where m and ℓ are the ω-marking and node-label of the node. A new arc is created by "NEWARC(n_1,n_2,ℓ)" where n_1,n_2 and ℓ are the source-node, destination-node and arc-label. It is possible to append new information to an existing label, ℓ, by the operation "APPEND(ℓ,new-inf)". The ω-marking and the node-label of a node x is denoted by m_x and ℓ_x, respectively. The arc-label of the arc from node x to node y is denoted by ℓ_{xy}. By "NEXT(m,t,c)" we denote the ω-marking obtained by firing transition t with colour $c \in C(t)$ in the ω-marking m.

ALGORITHM TO PRODUCE HL-TREES

```
UNPROCESSED := {NEWNODE(m0,empty)}; PROCESSED := ∅
REPEAT
   SELECT some node x∈UNPROCESSED
   IF mx~my for some node y∈PROCESSED
   THEN APPEND(ℓx,"equivalent to y")
   ELSE IF no pair (t,c) has concession in mx
        THEN APPEND(ℓx,"dead")
        ELSE BEGIN {x is non-equivalent and non-dead}
                   FOR ALL (t,c) having concession in mx DO
                   BEGIN
                       m := NEXT(mx,t,c); ℓ:=empty
                       FOR ALL ancestors z with m>mz DO
                       BEGIN
                           FOR ALL p∈P,c∈C(p) where m(p)(c)>mz(p)(c) DO
                              m(p)(c) := ω
                           APPEND(ℓ,"covering of z")
                       END
                       IF m~mu for some node u being a son of x
                       THEN APPEND(ℓxu,"(t,c)")
                       ELSE BEGIN
                                v := NEWNODE(m,ℓ)
                                UNPROCESSED := UNPROCESSED ∪ {v}
                                NEWARC(x,v,"(t,c)")
                            END
                   END
             END
   UNPROCESSED := UNPROCESSED\{x}; PROCESSED := PROCESSED ∪ {x}
UNTIL UNPROCESSED = ∅
```

■

The algorithm works in the following way: as long as there are more unprocessed nodes, one is selected and processed. The processing of a node starts with a check for equivalence with an already processed node, i.e. only the first processed node in each equivalence class of ~ is developed further.If no equivalent node is found, the node is checked for being dead. If it is not dead, for each pair (t,c) with concession a son is produced and included in the tree (unless it is an equivalent brother). Each HL-tree is a subtree of a PT-tree for the equivalent PT-net, obtained from the HL-net by the method described in [4]. In [2, 7, 8] it is shown that each PT-tree is finite. Thus each HL-tree is finite and our algorithm always halts.

Technical remark: The constructed HL-tree normally depends on the order in which the nodes are processed. This means that each HL-net may have several corresponding HL-trees. Normally an implementation enforces an ordering-rule for the processing of nodes, and this rule then determines the actual HL-tree, constructed for the HL-net by that implementation.

Technical remark: In an implementation of the algorithm it is crucial to minimize the time spent on testing for equivalence. In appendix 3 we describe a fairly effective algorithm to test two ω-markings for equivalence. Moreover our implementation will use hash coding to divide markings into subclasses in such a way, that equivalent markings always belong to the same subclass. This hash coding drastically decreases the number of pairs to be tested for equivalence.

5. WHAT CAN BE PROVED BY MEANS OF HL-TREES?

In this section we discuss how HL-trees can be used to prove properties of the corresponding HL-nets.

A proof rule is a theorem by which properties of HL-nets can be deduced from properties of HL-trees (or vice versa). For PT-trees [2,8] describe a number of such proof rules, from which it is possible to deduce information concerning: boundedness, coverability, reachability, liveness, etc. Some of the proof rules are total, in the sense that the question concerning presence or absence of the particular net property always can be answered by means of the proof rule. Other proof rules are partial, in the sense that the question only sometimes can be answered.

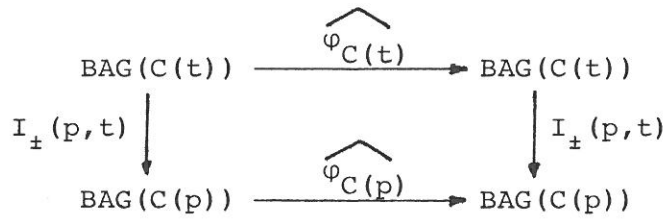
For HL-trees the situation is a bit more complicated, since the observed tree properties in a crucial way may depend on the chosen partition, which determines the relation of equivalent markings. Hence it is necessary to introduce the notion of a sound partition, which intuitively means that the partition respects the inherent symmetry properties of the HL-net. If for the philosopher system we allowed arbitrary permutation, instead of just rotation, this would be a typical example of a non-sound partition, since it neglects the fact that in this system there is another relationship between neighbours than between non-neighbours. Analogously, it would be non-sound to have both PH and F as atomic colour sets, since this would neglect the fact that there is another relationship between a philosopher and the two nearest forks than between the philosopher and the three remote forks.

Definition A partition is sound iff it satisfies the following criteria:

(SC1) $\forall p \in P \forall t \in T \forall \varphi \in \Phi: \widehat{\varphi_{C(p)}} \circ I_{\pm}(p,t) = I_{\pm}(p,t) \circ \widehat{\varphi_{C(t)}}$.

(SC2) $\forall \varphi \in \Phi: m_0 = \varphi(m_0)$. ■

SC1 can be visualized by the following commutative diagram:



SC1 demands that the chosen partition for the HL-net and hence the set of allowed symmetries must agree with the firing of transitions in the sense that equivalent colours have to be treated in the "same" way. SC2 demands, that the initial marking has to be symmetric. In practice it is often nearly trivial to verify the soundness criteria by means of the following rules:

(R1) Due to the linearity of the functions, SC1 can be verified by checking only steps of the form (t,c).

(R2) If $I_{\pm}(p,t)$ is an identity-function or a zero-function SC1 is always satisfied.

(R3) When $I_{\pm}(p,t)$ is a sum of several functions, SC1 can be verified for each of them, separately.

(R4) When a function appears in $I_{\pm}(p,t)$ for several places/transitions it needs only to be considered once to verify SC1.

(R5) When the symmetry type of C(t) is identity, SC1 is always satisfied.

(R6) When the symmetry type of C(t) is rotation it is enough to consider the "one step forward" rotation to verify SC1.

(R7) When the symmetry type of C(t) is permutation it is enough to consider transpositions (interchanging of two elements) to verify SC1.

(R8) SC2 is satisfied iff
$\forall p \in P[sym(C(p)) \neq identity \Rightarrow \exists k \in \mathbb{N}_0 : m_0(p) = k \times \Sigma C(p))]$
where $\Sigma C(p)$ denotes the bag which contains exactly one occurrence of each colour in C(p).

As an example, soundness of the partition, chosen for the philosopher system in section 3, can easily be verified. We only have to prove the following properties, where r is the function relating F to PH, while φ_{PH} is the "one step forward" rotation on PH:

$$r^{-1} \circ \varphi_{PH} \circ r \circ left = left \circ \varphi_{PH}$$
$$r^{-1} \circ \varphi_{PH} \circ r \circ right = right \circ \varphi_{PH} .$$

To formulate our proof rules we need some notation. $R(m_0)$ is the set of markings which are reachable from m_0. $R(m_0)(p) = \{m(p)(c) \mid m \in R(m_0) \wedge c \in C(p)\}$ is the coefficients appearing at place p, while $R(m_0)(p)(c) = \{m(p)(c) \mid m \in R(m_0)\}$ is the coefficients appearing at place p for colour c. $T(m_0)$ is the set of nodes in the HL-tree having m_0 as root. $T(m_0)(p)$ and $T(m_0)(p)(c)$ are defined analogous to $R(m_0)(p)$ and $R(m_0)(p)(c)$, respectively. Furthermore we define the function $map_{C(p)} \in [C(P) \rightarrow \mathbb{P}(C(p))]$ as follows

$$map_{C(p)}(c) = \{c' \in C(p) \mid \exists \varphi \in \Phi : \varphi_{C(p)}(c') = c\}.$$

Observation

(O1)
$$map_{C(p)}(c) = \begin{cases} \{c\} \text{ if } sym(C(p)) = identity \\ C(p) \text{ if } sym(C(p)) \in \{rotation, permutation\} \end{cases}$$

We now formulate our four proof rules for HL-trees. They are generalizations of the proof rules for PT-trees given in [8].

PROOF RULES FOR HL-NETS

(PR1) H is bounded $\Leftrightarrow \forall p \in P: \omega \notin T(m_0)(p)$
prerequisite: SC1

(PR2)
$$\sup R(m_0)(p)(c)^{\dagger} = \max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c')$$
prerequisite: SC1, SC2

(PR3) $\exists \alpha \in T(m_0)$: "dead" $\in \ell_\alpha \Rightarrow \exists m \in R(m_0)$: m is dead
prerequisite: none

(PR4) $\exists m \in R(m_0)$: m is dead $\Rightarrow$
$(\exists \alpha \in T(m_0)$: "dead" $\in \ell_\alpha) \vee (\exists p \in P: \omega \in T(m_0)(p))$
prerequisite: SC1

As an example on how the proof rules can be used, we again turn to the philosopher system with the HL-tree shown in figure 2. By applying PR1 we derive that the net is bounded, and from PR2 we see that 1 can be used as a uniform bound for all places and all colours. PR4 tells us that no reachable marking is dead.

To prove the correctness of our proof rules we need the following four lemmas:

Lemma 1 Assume SC1, then $\forall \varphi \in \Phi: m_1[\sigma > m_2 \Rightarrow \varphi(m_1)[\varphi(\sigma) > \varphi(m_2)$ for all ω-markings and all firing-sequences.

Proof See appendix 1.

Corollary Assume SC1 and SC2, then

a) $m_1 \sim m_2 \Rightarrow [m_1 \in R(m_0) \Leftrightarrow m_2 \in R(m_0)]$

b) $m_1 \sim m_2 \Rightarrow [m_1$ is dead $\Leftrightarrow m_2$ is dead$]$. ■

Given an ω-marking m_ω and a marking m we define that m_ω agrees with m, $m_\omega \geq m$, iff

$$\forall p \in P \forall c \in C(p): m_\omega(p)(c) \neq \omega \Rightarrow m_\omega(p)(c) = m(p)(c)$$

i.e. for each pair p and c the coefficients in m_ω and m are identical or that of m_ω is ω. It is easy to prove the following:

---

† By convention sup A = ω, for $A \subseteq \mathbb{N}$, when $\forall k \in \mathbb{N} \exists a \in A: a \geq k$.

Observations

(O2) $m_\omega \geq m \Rightarrow \varphi(m_\omega) > \varphi(m)$ for all $\varphi \in \Phi$

(O3) $m_\omega \geq m \wedge m[\sigma > m' \Rightarrow \exists m'_\omega : m_\omega [\sigma > m'_\omega \wedge m'_\omega \geq m'$ for all firing sequences σ.

■

Lemma 2 Assume SC1, then $\forall m \in R(m_0) \exists \varphi \in \Phi \ \exists \alpha \in T(m_0) : m_\alpha \geq \varphi(m)$.

Proof See appendix 1. ■

Given an ω-marking m and $k \in \mathbb{N}$, then we define m^k as follows:

$$m^k(p)(c) = \begin{cases} k & \text{if } m(p)(c) = \omega \\ m(p)(c) & \text{otherwise} \end{cases}$$

for all $p \in P$ and $c \in C(p)$.

Lemma 3 $\forall \alpha \in T(m_0) \forall k \in \mathbb{N} \ \exists m \in R(m_0) : m_\alpha \geq m \geq m_\alpha^k$.

Proof See appendix 1. The proof of this lemma is by far the most complicated and it involves several induction arguments.

Corollary a) $\omega \in T(m_0)(p)(c) \Rightarrow \sup R(m_0)(p)(c) = \omega$

b) $k \in T(m_0)(p)(c) \Rightarrow k \in R(m_0)(p)(c)$. ■

Lemma 4 Assume SC1 and SC2, then

$$\sup R(m_0)(p)(c) = \max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c')$$

If only SC1 is assumed we get "$\leq$" instead of "=".

Proof See appendix 1. ■

Theorem The four proof rules PR1-PR4 are valid, under the given prerequisites.

Proof

PR1: The proof is by contradiction. Assume that H is bounded, and $\exists p \in P$: $\omega \in T(m_0)(p)$. Then $\omega \in T(m_0)(p)(c)$ for some colour $c \in C(p)$ and by the corollary of lemma 3 $R(m_0)(p)(c)$ is unbounded - contradiction with H being bounded.

Next assume that $\omega \notin T(m_0)(p)$, and H unbounded, i.e.

(1) $\exists p \in P\ \exists c \in C(p)\ \forall k \in \mathbb{N}\ \exists m \in R(m_0)\colon m(p)(c) > k$.

For each of these m, by lemma 2,

(2) $\exists \alpha \in T(m_0)\ \exists \varphi^m \in \Phi\colon m_\alpha \succcurlyeq \varphi^m(m)$.

We then get

(3) $m_\alpha(p)(\varphi^m_{C(p)}(c)) \geq \varphi^m(m)(p)(\varphi^m_{C(p)}(c)) = m(p)(c) > k$

for each k in (1). "$\geq$" follows from (2), "=" is an immediate consequence of the way $\varphi^m(m)$ is defined, while ">" follows from (1). Since $T(m_0)$ and Φ are finite it follows from (3) that $\exists \alpha' \in T(m_0)\colon m_{\alpha'}(p)(\varphi^m(c)) = \omega$ - contradiction with $\omega \notin T(m_0)(p)$.

PR2: Identical to lemma 4.

PR3: Assume that $\exists \alpha \in T(m_0)$: "dead" $\in \ell_\alpha$. By lemma 3, $\exists m \in R(m_0)\colon m_\alpha \succcurlyeq m$. The marking m_α is dead, and since m is smaller it is dead too.

PR4: Assume that $\exists m \in R(m_0)$: m is dead, and $\forall \alpha \in T(m_0)$: "dead" $\notin \ell_\alpha$. By lemma 2, $\exists \varphi \in \Phi\ \exists \alpha \in T(m_0)\colon m_\alpha \succcurlyeq \varphi(m)$. The marking $\varphi(m)$ is dead, by the corollary of lemma 1. m_α is not dead and thus we conclude $m_\alpha > \varphi(m)$, which together with $m_\alpha \succcurlyeq \varphi(m)$ yields $m_\alpha(p)(c) = \omega$ for some $p \in P$ and $c \in C(p)$.

■

The following two lemmas are not necessary to establish the proof rules, but they provide useful insight in the structure of the reachability tree:

Lemma 5

$\forall \alpha 1, \alpha 2 \in T(m_0)$ with $(t,c) \in \ell_{\alpha 1\ \alpha 2}$

$\exists m_1, m_2 \in R(m_0)$ with $m_1[t,c>m_2$:

(i) $m_{\alpha 1} \succ m_1$ $\wedge$

(ii) $m_{\alpha 2} \succ \begin{cases} m_2 & \text{if } (t,c) = \text{head}(\ell_{\alpha 1\ \alpha 2}) \\ \varphi(m_2) \text{ for some } \varphi \in \Phi & \text{otherwise.} \end{cases}$

Proof: See appendix 1. ■

Lemma 6 Assume SC1, then:

$\forall m_1, m_2 \in R(m_0)$ with $m_1[t,c>m_2$

$\exists \varphi \in \Phi$

$\exists \alpha 1, \alpha 2 \in T(m_0)$ with $\varphi(t,c) \in \ell_{\alpha 1\ \alpha 2}$:

(i) $m_{\alpha 1} \geq \varphi(m_1)$ $\wedge$

(ii) $m_{\alpha 2} \geq \begin{cases} \varphi(m_2) \text{ if } \varphi(t,c) = \text{head}(\ell_{\alpha 1\ \alpha 2}) \\ \varphi' \circ \varphi(m_2) \text{ for some } \varphi' \in \Phi \text{ otherwise.} \end{cases}$

Proof: See appendix 1. ■

6. EXAMPLES OF THE USE OF HL-TREES

This section contains two examples which together with the system of the five dining philosophers, treated in section 3, illustrate a spectrum of the problems concerning the construction and analysis of HL-trees. The first example is a system, where the equivalence relation involves permutation, identity and products. The second example illustrates covering markings.

Data base system In [4] the system is described and analyzed by means of the invariant method. We define a partition by

atomic DBM:permutation; E: identity
product MB:subset of DBM×DBM.

An HL-tree for the data base system is shown in figure 3.
It is easy to verify that the chosen partition is sound (see appendix 2). By applying PR1 we derive that the net is bounded, and from PR2 we see that 1 can be used as a uniform bound for all places and all colours. PR4 tells us that no reachable marking is dead.

The leaves of the tree are identical with #1 and #6, respectively. This is, however, a coincidence and it changes if the nodes are processed in another order. As mentioned earlier, an alternative to the HL-net is to construct the PT-tree for the equivalent PT-net. In the following table we compare the size of the HL-tree with the size of the PT-tree (for different sizes of DBM). Normally, the HL-trees are not just smaller than the corresponding PT-trees, but they also grow slower when the sizes of the involved colour sets increase.

number of elements in DBM	number of nodes in the HL-tree	number of nodes in the PT-tree
2	5	9
3	9	43
4	14	225
5	23	>1400

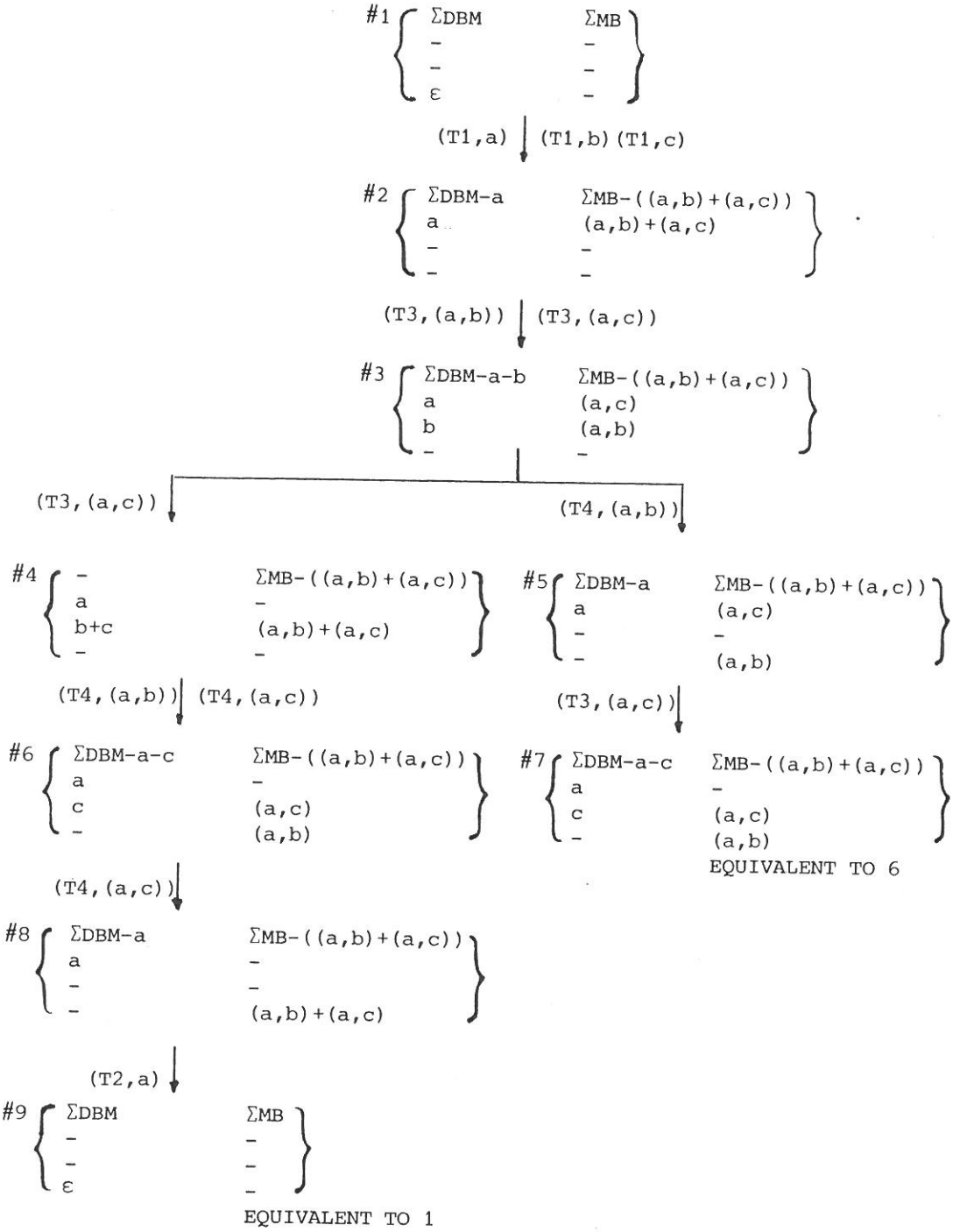


Fig. 3: HL-tree for the data base system.

Producer-consumer system We have also constructed an HL-tree for a system, where two producers send two different kinds of messages to a consumer via an unbounded buffer. The HL-net and HL-tree for this system can be found in appendix 2. Again, the chosen partition is sound, and the expected properties of the HL-net can be derived by means of the proof rules. The HL-tree has 30 nodes of which 17 are coverings (some of them even cover two other markings). As in the two other examples of this paper, the HL-tree for this system is remarkably smaller than the corresponding PT-tree, which has 93 nodes.

Acknowledgement Some of the ideas in this paper are founded on a student project at Aarhus University with the following participants: Arne M. Jensen, Peter A. Nielsen, Erik Schjøtt, Kasper Østerbye and Kurt Jensen (Supervisor).

References

[1] H.J. Genrich and K. Lautenbach: System modelling with high-level Petri nets, Theoretical Computer Science 13 (1981), 109-136.

[2] M. Hack: Decidability questions for Petri Nets. TR 161, MIT, 1976.

[3] P. Huber, A.M. Jensen, L.O. Jepsen and K. Jensen: Towards reachability trees for high-level Petri nets. Proceedings of 5th European Workshop on Applications and Theory of Petri Nets, Aarhus University 1984. Identical to the present paper, except that the 3 appendices are omitted.

[4] K. Jensen: Coloured Petri nets and the invariant-method. Theoretical Computer Science 14 (1981), 317-336.

[5] K. Jensen: How to find invariants for coloured Petri nets. In: J. Gruska, M. Chytill (eds.): Mathematical Foundations of Computer Science 1981, Lecture Notes in Computer Science, vol. 118, Springer-Verlag, 1981, 327-338.

[6] K. Jensen: High-level Petri nets. In: A. Pagnoni and G. Rozenberg (eds.): Applications and Theory of Petri Nets, Informatik-Fachberichte vol. 66, Springer-Verlag 1983, 166-180.

[7] R.M. Karp and R.E. Miller: Parallel program schemata. Journal of Computer and System Sciences, vol. 3 (1969), 147-195.

[8] J.L. Peterson: Petri net theory and the modellings of systems. Prentice-Hall 1981.

[9] W. Reisig: Petri nets with individual tokens. In: A.Pagnoni and G. Rozenberg (eds.): Applications and Theory of Petri Nets, Informatik-Fachberichte vol. 66, Springer-Verlag 1983, 229-249.

APPENDIX 1: Proof of Lemma 1-6

In the following λ denotes the empty firing-sequence.

Lemma 1:

Assume SC1, then $\forall\varphi\in\Phi$: $m_1[\sigma>m_2 \Rightarrow \varphi(m_1)[\varphi(\sigma)>\varphi(m_2)$ for all ω-markings and all firing-sequences.

Proof:

Let $\varphi\in\Phi$ and assume $m_1[\sigma>m_2$. The proof is by induction on the length of σ:

Basis: $|\sigma|=0$.

Then $\sigma=\lambda$ and $m_1=m_2$. Since $\varphi(\lambda)=\lambda$ we have to show $\varphi(m_1)[\lambda>\varphi(m_2)$, which is trivially satisfied.

Step: $|\sigma|>0$.

Then let $\sigma=\sigma'(t,c)$, $m_1[\sigma'>m'$ and $m'[t,c>m_2$ as shown in the left part of figure 4.

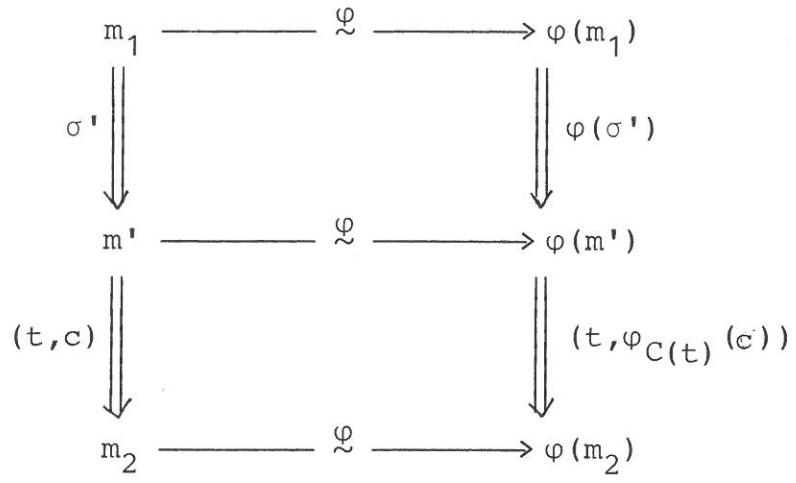


Figure 4. The connection between the markings in the proof of lemma 1.

By the induction hypothesis $\varphi(m_1)[\varphi(\sigma')>\varphi(m')$ and thus we are finished if we can prove $\varphi(m')[t,\varphi_{C(t)}(c)>\varphi(m_2)$ i.e.:

(1) $\forall p\in P$: i) $\varphi(m')(p) \geqq I_-(p,t)(\varphi_{C(t)}(c))$

ii) $\varphi(m_2)(p) = \varphi(m')(p) - I_-(p,t)(\varphi_{C(t)}(c)) + I_+(p,t)(\varphi_{C(t)}(c))$.

From $m'[t,c>m_2$ it follows that

(2) $\forall p\in P$: i) $m'(p) \geqq I_-(p,t)(c)$

ii) $m_2(p) = m'(p) - I_-(p,t)(c) + I_+(p,t)(c)$.

To show (1i) take $p\in P$. Then

$\Downarrow$ $m'(p) \geqq I_-(p,t)(c)$ by (2i)

$\Downarrow$ $\overline{\varphi_{C(p)}}\circ m'(p)) \geqq \overline{\varphi_{C(p)}}\circ I_-(p,t)(c) = \overline{\varphi_{C(p)}\circ I_-(p,t)(c)}$

$\Downarrow$ $\varphi(m')(p) \geqq I_-(p,t)\circ\overline{\varphi_{C(t)}}(c) = I_-(p,t)\circ\varphi_{C(t)}(c)$

where we have used that the linear ω-bag-extension of $\varphi_{C(p)}$ preserves "$\geqq$" for ω-bags (for the first implication) and used SC1 (for the second implication).
Analogously (1ii) follows from (2ii).

∎

Corollary:

Assume SC1 and SC2, then

a) $m_1 \sim m_2 \Rightarrow [m_1\in R(m_0) \Leftrightarrow m_2\in R(m_0)]$

b) $m_1 \sim m_2 \Rightarrow [m_1$ is dead $\Leftrightarrow m_2$ is dead$]$.

Proof:

To show a) assume $m_1 \sim m_2$ (with $m_2 = \varphi(m_1)$) and assume $m_1 \in R(m_0)$ (with $m_0[\sigma > m_1$). Using SC2 and lemma 1 we get:

$$m_0 = \varphi(m_0)[\varphi(\sigma) > \varphi(m_1) = m_2, \text{ i.e. } m_2 \in R(m_0).$$

The other direction of the biimplication follows from the symmetry of $\sim$.

To show b) assume $m_1 \sim m_2$ (with $m_2 = \varphi(m_1)$) and assume that m_1 is not dead (with $m_1[t,c > m'$). Using lemma 1 we get:

$$m_2 = \varphi(m_1)[\varphi(t,c) > \varphi(m'),$$

i.e. m_2 is not dead. As before the other direction of the biimplication follows from the symmetry of $\sim$.

∎

Lemma 2:

Assume SC1, then $\forall m \in R(m_0)\ \exists \varphi \in \Phi\ \exists \alpha \in T(m_0)$: $m_\alpha \geqslant \varphi(m)$.

Proof:

In order to deal with reduction by equivalent markings we have generalized the corresponding proof for PT-trees in [2], (lemma 3.7).

Let $m \in R(m_0)$ (with $m_0[\hat{\sigma}>m$). We will use induction on the length of σ:

Basis: $|\sigma|=0$.

Then $\sigma=\lambda$ and $m=m_0$. The root, α_0, can then be used as α and the set of identity-functions, ID, as φ, giving

$$m_{\alpha_0} = m_0 \geqslant m_0 = ID(m_0).$$

Step: $|\sigma|>0$.

Then let $\sigma=\sigma'(t,c)$, $m_0[\sigma'>m'$ and $m'[t,c>m$ as shown in the left part of figure 5.

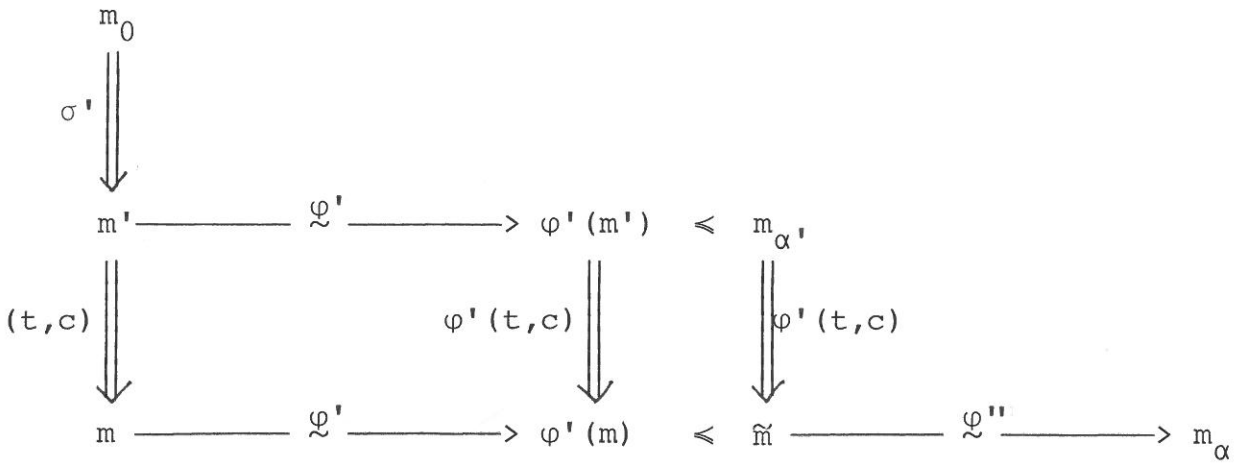


Figure 5. The connection between the markings in the proof of lemma 2.

By the induction hypothesis $\exists\varphi' \in \Phi \exists\alpha' \in T(m_0)$:

(1) $m_{\alpha'} \geqslant \varphi'(m')$.

If "equivalent" $\in \ell_{\alpha'}$, we will replace α' by the equivalent node, which is further processed (and change φ' accordingly). Hence we can assume that "equivalent" $\notin \ell_{\alpha'}$.

Using lemma 1 on $m'[t,c>m$ we get $\varphi'(m')[\varphi'(t,c)>\varphi'(m)$.
From observation O3 it then follows that $\exists\tilde{m}$:

$$m_{\alpha'}[\varphi'(t,c)>\tilde{m} \wedge \tilde{m} \geqslant \varphi'(m),$$

and since "equivalent" $\notin \ell_{\alpha'}$, there exists an arc from α' to another node α such that $\varphi'(t,c) \in \ell_{\alpha',\alpha}$ and $m_\alpha \sim \tilde{m}$ (with $m_\alpha = \varphi''(\tilde{m})$).
Observation O2 then yields $m_\alpha = \varphi''(\tilde{m}) \geqslant \varphi'' \circ \varphi'(m)$.
The desired φ is $\varphi'' \circ \varphi'$ and we are finished.

∎

Let $I = I_+ - I_-$. Then for each firing sequence

$$\sigma = (t_1,c_1)(t_2,c_2).....(t_n,c_n) \quad \text{with } n \geq 0$$

we define $\Delta(\sigma)$ to be the change in marking caused by σ:

$$\forall p \in P: \Delta(\sigma)(p) = \sum_{i=1}^{n} I(p,t_i)(c_i).$$

Analogously we define Δ_+ and Δ_- by means of I_+ and I_- respectively.†

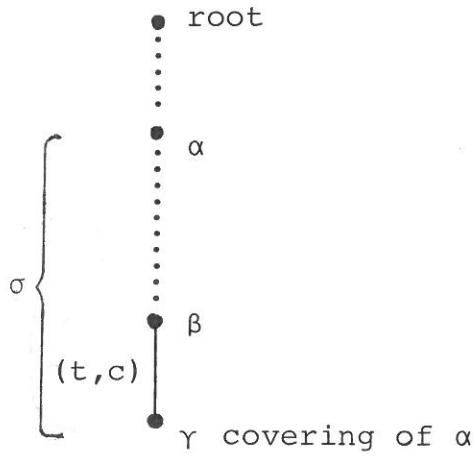
---

† In the terminology of [4] and [6]

$\Delta(\sigma) = I * \sigma$, where "$*$" is the generalized matrix product.

Propositions

Assume the following situation:



where:

i) "covering of α" $\in \ell_\gamma$

ii) $(t,c) = \text{head}(\ell_{\beta\gamma})$

iii) σ consists of the heads of all arc-labels between α and γ.

Then the following propositions are satisfied:

a) $m_\gamma \geqq m_\beta + \Delta(t,c) \geqq m_\alpha$.

b) $m_\beta(p)(c') \neq \omega \Rightarrow \Delta(\sigma)(p)(c') \geqq 0$ for all $p \in P$ and $c' \in C(p)$.

c) $m_\gamma(p)(c') \neq \omega \Rightarrow \Delta(\sigma)(p)(c') = 0$ for all $p \in P$ and $c' \in C(p)$.

d) σ is a firing sequence at an ω-marking, m, if

$$m(p)(c') \geqq \begin{cases} m_\alpha(p)(c') & \text{if } m_\beta(p)(c') \neq \omega \\ \Delta_-(\sigma)(p)(c') & \text{if } m_\beta(p)(c') = \omega \end{cases}$$

for all $p \in P$ and $c' \in C(p)$.

e) If $\alpha_1, \alpha_2, \ldots, \alpha_n$ are <u>all</u> nodes for which "covering of α_i" $\in \ell_\gamma$ (with firing sequences $\sigma_1, \sigma_2, \ldots, \sigma_n$ where σ_i consists of the heads of all arc labels between α_i and γ) we also get

$$e)\ (m_\beta(p)(c') \neq \omega \wedge m_\gamma(p)(c') = \omega)$$
$$\Rightarrow \exists i \in 1..n : \Delta(\sigma_i)(p)(c') > 0$$

for all $p \in P$ and $c' \in C(p)$.

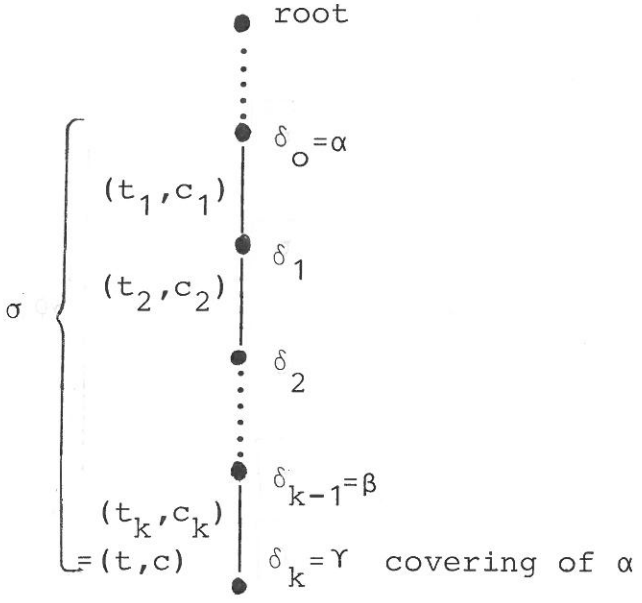
<u>Proof</u>:

Propositions a), b), c) and e) follow immediately from the HL-tree algorithm.

To prove d) let σ have the form

$$\sigma = (t_1,c_1)(t_2,c_2)\ldots(t_k,c_k) \text{ with } k \geq 1$$

and let $\delta_o, \delta_1, \ldots \delta_k$ be the nodes between α and γ:



The proof is by induction on $j \in 0..k$, using the following hypothesis:

There exists a marking m', such that

i) $m[(t_1,c_1)\ldots(t_j,c_j) > m'$

ii) $m'(p)(c') \geq \begin{cases} m_{\delta_j}(p)(c') \text{ if } m_\beta(p)(c') \neq \omega \\ \Delta_-((t_{j+1},c_{j+1})\ldots(t_k,c_k))(p)(c') \\ \qquad \text{if } m_\beta(p)(c') = \omega \end{cases}$

for all $p \in P$ and $c' \in C(p)$ and $j<k$.

When j=k part i) of the hypothesis immediately yields that σ is a firing sequence at m, and the proof of d) is finished.

Basis: j=0.

Then m can be used as m' in i). Part ii) follows immediately from the assumption of d).

Step: j>0.

By induction hypothesis there exists a marking m', such that

(1) $m[(t_1,c_1)\ldots(t_{j-1},c_{j-1}) > m'$ and

(2) $m'(p)(c') \geq \begin{cases} m_{\delta_{j-1}}(p)(c') \text{ if } m_\beta(p)(c') \neq \omega \\ \Delta_-((t_j,c_j)\ldots(t_k,c_k))(p)(c') \text{ if } m_\beta(p)(c) = \omega \end{cases}$

for all $p \in P$ and $c' \in C(p)$.

Since $(t_j,c_j) = \text{head}(\ell_{\delta_{j-1}\,\delta_j})$ it has concession at $m_{\delta_{j-1}}$ and by (2) it has concession at m' too, i.e. $\exists\, m''$:

(3) $m'[t_j,c_j > m''$.

Together with (1) this yields $m[(t_1,c_1)\ldots(t_j,c_j) > m''$, i.e. part i) of the induction hypothesis for j.

Now let $p \in P$ and $c' \in C(p)$ and $j<k$ to check part ii) of the induction hypothesis for j. There are two cases:

a) $\underline{m_\beta(p)(c') \neq \omega}$. Since $j<k$ then $\delta_j \neq \gamma$ and since $m_\beta(p)(c') \neq \omega$ there is not introduced an ω at place p and colour c' at any predecessor node of β. Especially $m_{\delta_j}(p)(c') \neq \omega$ and then

$$\begin{aligned} m_{\delta_j}(p)(c') &= m_{\delta_{j-1}}(p)(c') + \Delta(t_j,c_j)(p)(c') \\ &\leq m'(p)(c') + \Delta(t_j,c_j)(p)(c') && \text{by (2)} \\ &= m''(p)(c') && \text{by (3)} \end{aligned}$$

b) $\underline{m_\beta(p)(c') = \omega}$. Then

$$\begin{aligned} m''(p)(c') &= m'(p)(c') + \Delta(t_j,c_j)(p)(c') && \text{by (3)} \\ &\geq \Delta_-((t_j,c_j)(t_{j+1},c_{j+1})\cdots(t_k,c_k))(p)(c') \\ &\quad + \Delta(t_j,c_j)(p)(c') && \text{by (2)} \\ &= \Delta_-((t_j,c_j)(t_{j+1},c_{j+1})\cdots(t_k,c_k))(p)(c') \\ &\quad + \Delta_+(t_j,c_j)(p)(c') - \Delta_-(t_j,c_j)(p)(c') \\ &\geq \Delta_-((t_{j+1},c_{j+1})\cdots(t_k,c_k))(p)(c'). \end{aligned}$$

∎

Lemma 3:

$\forall \alpha \in T(m_0)\, \forall k \in \mathbb{N}\ \exists m \in R(m_0)$: $m_\alpha \geqslant m \geqq m_\alpha^k$.

Proof:

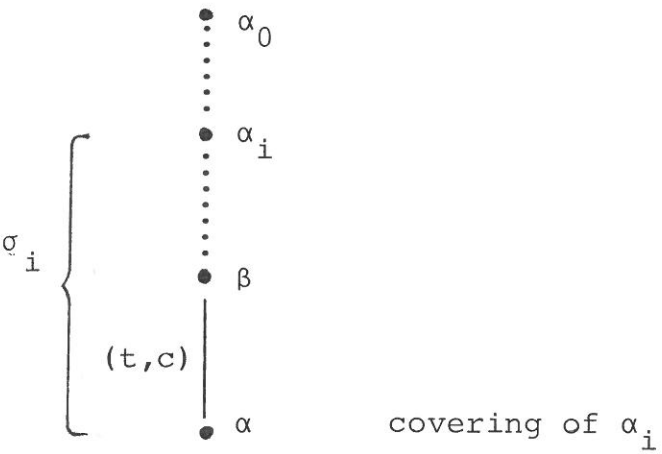
The proof is by induction on the number of arcs from the root to α.

Basis: Number of arcs = 0.

Then α is the root node and $m_\alpha = m_0$. We can use $m_0 \in R(m_0)$ as m for all $k \in \mathbb{N}$.

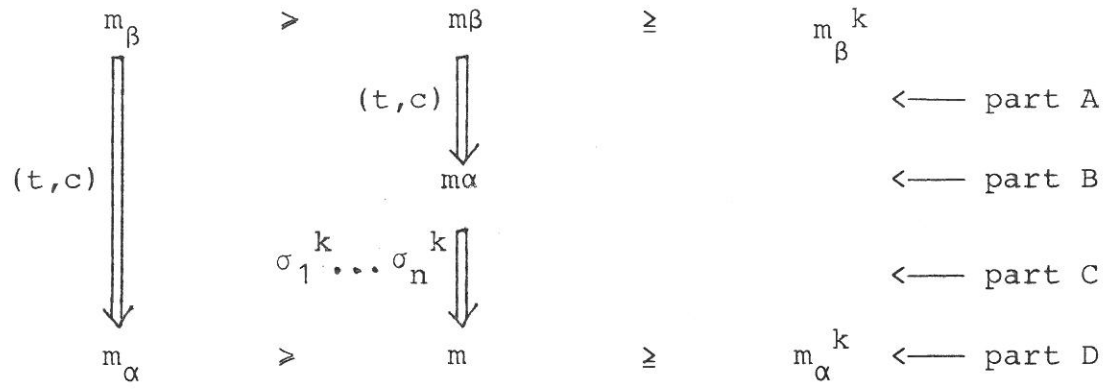
Step: Number of arcs > 0.

The general situation can be pictured as follows:



$\alpha_1, \cdots, \alpha_n$ are all nodes with "covering of α_i" $\in \ell_\alpha$, and σ_i consists of the heads of all arc-labels between α_i and α.

Now let $k \in \mathbb{N}$ be given. The induction step is rather complicated. The proof is divided into four parts, A-D, and the idea can be visualized as follows:



By the induction hypothesis we get the marking $m\beta \in R(m_0)$ corresponding to m_β.
From $m\beta$ (t,c) is fired to get the marking $m\alpha$. Then for each "covering of α_i" $\in \ell_\alpha$ σ_i is fired k times to get the marking m corresponding to m_α. When choosing $m\beta$ we have to ensure that (t,c) followed by $\sigma_1^k \cdots \sigma_n^k$ can be fired from $m\beta$.

Now let $K_1, K_2 \in \mathbb{N}$ satisfy

(1) $K_1 \geq \Delta_-(t,c)(p)(c')$ for all $p \in P$ and $c' \in C(p)$

(2) $K_2 \geq \Delta_-(\sigma_1^k \cdots \sigma_n^k)(p)(c')$ for all $p \in P$ and $c' \in C(p)$

By the induction hypothesis $\exists\, m\beta \in R(m_0)$:

(3) $\forall p \in P \forall c' \in C(p)$: $m_\beta(p)(c') \geq m\beta(p)(c') \geq m_\beta^{k+K_1+K_2}(p)(c')$

The choice of K_1 and K_2 assures that (t,c) followed by $\sigma_1^k \ldots \sigma_n^k$ can be fired from $m\beta$, which can be a problem if $m_\beta(p)(c')$ is infinite while $m\beta(p)(c')$ is finite.

Part A: To show (t,c) has concession at $m\beta$ take $p \in P$ and $c' \in C(p)$.

There are two cases:

a) $m_\beta(p)(c') \neq \omega$. Then by definition

$m_\beta^{k+K_1+K_2}(p)(c') \qquad m_\beta(p)(c')$ and

$$m\beta(p)(c') = m_\beta(p)(c') \quad \text{by (3)}$$

$$\geq \Delta_-(t,c)(p)(c')$$

since (t,c) has concession at m_β.

b) $m_\beta(p)(c') = \omega$. Then

$$m\beta(p)(c') \geqq k+K_1+K_2 \quad \text{by (3)}$$

$$\geqq \Delta_-(t,c)(p)(c') \quad \text{by (1)}$$

Since (t,c) has concession at $m\beta$ there exists a $m\alpha \in R(m_0)$ satisfying $m\beta[t,c > m\alpha$, i.e.

(4) $\forall p \in P \forall c' \in C(p): \quad m\alpha(p)(c') = m\beta(p)(c') + \Delta(t,c)(p)(c')$

Part B: The marking $m\alpha$ satisfies the following two conditions:

$\forall p \in P \forall c' \in C(p):$

(5)
$$(m_\alpha(p)(c') = \omega \Leftrightarrow m_\beta(p)(c') = \omega)$$
$$\Downarrow$$
$$m_\alpha(p)(c') \geqq m\alpha(p)(c') \geqq m_\alpha^{k+K_2}(p)(c')$$

(6)
$$m_\beta(p)(c') \neq \omega$$
$$\Downarrow$$
$$m\alpha(p)(c') = m_\beta(p)(c') + \Delta(t,c)(p)(c')$$

Condition (6) follows immediately from (3) and (4).

To show (5) take $p \in P$ and $c' \in C(p)$ and assume $m_\alpha(p)(c') = \omega \Leftrightarrow m_\beta(p)(c') = \omega$. Then

(7) $m_\alpha(p)(c') = m_\beta(p)(c') + \Delta(t,c)(p)(c')$.

There are now two cases:

a) $\underline{m_\alpha(p)(c') \neq \omega \wedge m_\beta(p)(c') \neq \omega}$. Then

$m_\beta(p)(c') = m\beta(p)(c')$ by (3)

From (4) and (7) it follows that $m_\alpha(p)(c') = m\alpha(p)(c')$ and since $m_\alpha(p)(c') \neq \omega$ we have $m_\alpha(p)(c') = m\alpha(p)(c') = m_\alpha^{k+K_2}(p)(c')$.

b) $\underline{m_\alpha(p)(c') = \omega \wedge m_\beta(p)(c') = \omega}$. Then

$m\alpha(p)(c') = m\beta(p)(c') + \Delta(t,c)(p)(c')$ by (4)

$\geqq m\beta(p)(c') - \Delta_-(t,c)(p)(c')$

$\geqq k+K_1+K_2-\Delta_-(t,c)(p)(c')$ by (3)

$\geqq k+K_2$ by (1)

and thus we have:

$m_\alpha(p)(c') = \omega \geqq m\alpha(p)(c') \geqq k+K_2 = m_\alpha^{k+K_2}(p)(c')$.

Part C: We will now show that $\sigma_1^k \ldots \sigma_n^k$ is a firing sequence at $m\alpha$.

Let $\sigma_1^k \ldots \sigma_n^k = \xi_1\ \xi_2$. We will use induction on the length of ξ_1 and the induction hypothesis is as follows

$$m\alpha[\xi_1 > m\xi_1$$

where $m\xi_1$ satisfies

i) $m_\beta(p)(c') \neq \omega$

$$\Downarrow$$

$$m\xi_1(p)(c') \geqq m_{\alpha_i}(p)(c') \text{ for all } i\in 1..n$$

ii) $m_\beta(p)(c') = \omega$

$$\Downarrow$$

$$m\xi_1(p)(c') \geqq k + K_2 - \Delta_-(\xi_1)(p)(c')$$

Basis of part C: $|\xi_1| = 0$.

Then $\xi_1 = \lambda$ and $m\xi_1 = m\alpha$.

i) Assume $m_\beta(p)(c') \neq \omega$. Then

$$m\alpha(p)(c') = m_\beta(p)(c') + \Delta(t,c)(p)(c') \qquad \text{by (6)}$$

$$\geqq m_{\alpha_i}(p)(c') \text{ for all } i\in 1..n$$

The inequality follows from proposition a) used on each of the involved coverings.

ii) Assume $m_\beta(p)(c') = \omega$. Then

$m_\alpha(p)(c') = \omega$ too and $m\alpha(p)(c') \geqq k + K_2$ by (5)

Step of part C: $|\xi_1| > 0$.

Then let $\xi_1 = \xi_0 \sigma_i$. By the induction hypothesis we have:

(8) $m\alpha\ [\xi_0 > m\xi_0$

where $m\xi_0$ satisfies

(9) $m_\beta(p)(c') \neq \omega$
$\Downarrow$
$m\xi_0(p)(c') \geqq m_{\alpha_i}(p)(c')$ for all $i \in 1..n$

(10) $m_\beta(p)(c') = \omega$
$\Downarrow$
$m\xi_0(p)(c') \geqq k + K_2 - \Delta_-(\xi_0)(p)(c')$

In the case of (9) it follows immediately that σ_i is a firing sequence at $m\xi_0$, while in the case of (10) it follows since

$$\begin{aligned}
m\xi_0(p)(c') &\geqq k + K_2 - \Delta_-(\xi_0)(p)(c') \\
&\geqq K_2 - \Delta_-(\xi_0)(p)(c') \\
&\geqq \Delta_-(\sigma_1^k \ldots \sigma_n^k)(p)(c') - \Delta_-(\xi_0)(p)(c') \quad \text{by (2)} \\
&= \Delta_-(\xi_1\ \xi_2)(p)(c') - \Delta_-(\xi_0)(p)(c') \\
&\geqq \Delta_-(\xi_1)(p)(c') - \Delta_-(\xi_0)(p)(c') \\
&= \Delta_-(\xi_0 \sigma_i)(p)(c') - \Delta_-(\xi_0)(p)(c') \\
&= \Delta_-(\sigma_i)(p)(c')
\end{aligned}$$

Thus let $m\xi_1 \in R(m_0)$ be defined by

(11) $m\xi_0\ [\sigma_i > m\xi_1$.

Then $m\alpha[\xi_1 > m\xi_1$ by (8).

Now $m\xi_1$ satisfies i) and ii) of the induction hypothesis:

i) Assume $m_\beta(p)(c') \neq \omega$. Then

$$m\xi_1(p)(c') = m\xi_0(p)(c') + \Delta(\sigma_i)(p)(c') \quad \text{by (11)}$$
$$\geqq m\xi_0(p)(c') \quad \text{by proposition b)}$$
$$\geqq m_{\alpha_i}(p)(c') \text{ for all } i\in 1..n \quad \text{by (9)}$$

ii) Assume $m_\beta(p)(c') = \omega$. Then

$$m\xi_1(p)(c') = m\xi_0(p)(c') + \Delta(\sigma_i)(p)(c') \quad \text{by (11)}$$
$$\geqq k + K_2 - \Delta_-(\xi_0)(p)(c') + \Delta(\sigma_i)(p)(c') \quad \text{by (10)}$$
$$\geqq k + K_2 - \Delta_-(\xi_1)(p)(c').$$

We have now proved that $\sigma_1{}^k \ldots \sigma_n{}^k$ is a firing sequence at $m\alpha$ and thus we let $m\in R(m_0)$ be defined by

(12) $\quad m\alpha[\sigma_1{}^k \ldots \sigma_n{}^k > m.$

Part D: To complete the induction step of lemma 3 we have to prove that $m_\alpha \geqq m \geqq m_\alpha{}^k$, i.e. the following inequalities:

$$\forall p\in P \forall c'\in C(p): \quad m_\alpha(p)(c') \geqq m(p)(c') \geqq m_\alpha{}^k(p)(c').$$

Take $p\in P$ and $c'\in C(p)$. There are three cases:

a) $m_\beta(p)(c') \neq \omega \wedge m_\alpha(p)(c') = \omega$. Then

$$m(p)(c') = m\alpha(p)(c') + \Delta(\sigma_1{}^k \cdots \sigma_n{}^k)(p)(c') \quad \text{by (12)}$$
$$\geqq \Delta(\sigma_1{}^k \ldots \sigma_n{}^k)(p)(c')$$
$$\geqq k \cdot [\sum_{i=1}^{n} \Delta(\sigma_i)(p)(c)]$$
$$\geqq k \cdot 1 \quad \text{by proposition b) and e)}$$

Thus $m_\alpha(p)(c') = \omega \geqq m(p)(c') \geqq k = m_\alpha{}^k(p)(c')$.

b) $\underline{m_\beta(p)(c') \neq \omega \wedge m_\alpha(p)(c') \neq \omega}$. Then

$$m(p)(c') = m\alpha(p)(c') + \Delta(\sigma_1^k \cdots \sigma_n^k)(p)(c') \qquad \text{by (12)}$$

$$= m\alpha(p)(c') \qquad \text{by proposition c)}$$

$$= m_\alpha(p)(c') \qquad \text{by (5) since } m_\alpha(p)(c') \neq \omega$$

Thus $m_\alpha(p)(c') = m(p)(c') = m_\alpha{}^k(p)(c')$.

c) $\underline{m_\beta(p)(c') = \omega \wedge m_\alpha(p)(c') = \omega}$. Then

$$m(p)(c') = m\alpha(p)(c') + \Delta(\sigma_1^k \cdots \sigma_n^k)(p)(c') \qquad \text{by (12)}$$

$$\geqq k + K_2 + \Delta(\sigma_1^k \cdots \sigma_n^k)(p)(c') \qquad \text{by (5)}$$

$$\geqq k + K_2 - \Delta_-(\sigma_1^k \cdots \sigma_n^k)(p)(c')$$

$$\geqq k \qquad \text{by (2)}.$$

Thus $m_\alpha(p)(c') = \omega \geqq m(p)(c') \geqq k = m_\alpha{}^k(p)(c')$.

Corollary:

a) $\omega \in T(m_0)(p)(c) \Rightarrow \sup R(m_0)(p)(c) = \omega$

b) $k \in T(m_0)(p)(c) \Rightarrow k \in R(m_0)(p)(c)$.

Lemma 4:

Assume SC1 and SC2, then

$$\sup R(m_0)(p)(c) = \max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c')$$

if only SC1 is assumed we get "$\leq$" instead of "=".

Proof:

Let $p \in P$ and $c \in C(p)$ be given. We will show the inequality in both directions:

Direction "$\geq$"

a) Assume $\max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c') = k \in \mathbb{N}$

Then we have, for some element $c' \in C(p)$ and some $\varphi \in \Phi$, that

(1) $k \in T(m_0)(p)(c')$ and

(2) $\varphi_{C(p)}(c') = c$.

By the corollary to lemma 3 we get from (1), that $k \in R(m_0)(p)(c')$, i.e. $\exists m \in R(m_0)$:

(3) $m(p)(c') = k$.

Then
$$\begin{aligned} \varphi(m)(p)(c) &= \varphi(m)(p)(\varphi_{C(p)}(c')) && \text{by (2)} \\ &= m(p)(c') && \text{see below} \\ &= k && \text{by (3)} \end{aligned}$$

The second equality can be seen by observing, that in the first expression, we permute the marking and then take a look at the coefficient of the image of c'. Instead we can simply look at the coefficient of c' in the non-permuted marking.

Since $m \in R(m_0)$ we get $\varphi(m) \in R(m_0)$ from the corollary to lemma 1, hence $k \in R(m_0)(p)(c)$.

b) Assume $\max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c') = \omega$.

Since $T(m_0)$ and $map_{C(p)}(c)$ are finite there exist an element $c' \in C(p)$ and a symmetry $\varphi \in \Phi$, such that $\omega \in T(m_0)(p)(c')$ and $\varphi_{C(p)}(c') = c$.

By the corollary to lemma 3, $R(m_0)(p)(c')$ is unbounded, hence

(4) $\forall k \in \mathbb{N}\ \exists\ m \in R(m_0)$: $m(p)(c') \geqq k$

Analogous to case a) we get

(5) $\varphi(m)(p)(c) \geqq k$ and $\varphi(m) \in R(m_0)$, for each m in (4).

Thus $R(m_0)(p)(c)$ is unbounded, i.e. $\sup R(m_0)(p)(c) = \omega$.

Direction $\leqq$

c) Assume $\sup R(m_0)(p)(c) = k \in \mathbb{N}$.

We choose m such that

(6) $m(p)(c) = k$.

By lemma 2 $\exists \varphi \in \Phi\ \exists\ \alpha \in T(m_0)$: $m_\alpha \geqslant \varphi(m)$. Then in particular

(7) $m_\alpha(p)(\varphi_{C(p)}(c)) \geqq \varphi(m)(p)(\varphi_{C(p)}(c))$

$= m(p)(c)$ see case a)

$= k$ by (6)

Let $c' = \varphi_{C(p)}(c)$. Then $c = \varphi^{-1}_{C(p)}(c')$ and (7) yields

$\sup R(m_0)(p)(c) = k \leqq m_\alpha(p)(c') \leqq \max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c')$.

d) Assume $\sup R(m_0)(p)(c) = \omega$.

Then p is unbounded on colour c, i.e.

(8) $\quad \forall k \in \mathbb{N} \ \exists m \in R(m_0) : m(p)(c) \geqq k.$

Analogous to case c) we have $\exists \varphi \in \Phi \exists \alpha \in T(m_0)$:

(9) $\quad m_\alpha(p)(\varphi_{C(p)}(c)) \geqq m(p)(c) \geqq k$, for each m in (8).

Thus $\sup \bigcup_{c' \in map_{C(p)}(c)} T(m_0(p)(c') = \omega$ and since $T(m_0)$ and $map_{C(p)}(c)$ are finite we get $\max \bigcup_{c' \in map_{C(p)}(c)} T(m_0)(p)(c') = \omega$

∎

Lemma 5:

$\forall \alpha 1, \alpha 2 \in T(m_0)$ with $(t,c) \in \ell_{\alpha 1\ \alpha 2}$

$\exists m_1, m_2 \in R(m_0)$ with $m_1[t,c>m_2$:

(i) $m_{\alpha 1} \geqslant m_1 \wedge$

(ii) $m_{\alpha 2} \geqslant \begin{cases} m_2 & \underline{\text{if}}\ (t,c) = \text{head}(\ell_{\alpha 1\ \alpha 2}) \\ \varphi(m_2) \text{ for some } \varphi \in \Phi & \underline{\text{otherwise}}. \end{cases}$

Proof:

Using lemma 3 with $\alpha 1$ for α and $\max\{I_{-}(p,t)(c)(c') \mid p \in P \wedge c' \in C(p)\}$ for k we get an $m_1 \in R(m_0)$ such that $m_{\alpha 1} \geqslant m_1$ and (t,c) has concession in m_1 (with $m_1[t,c>m_2$). If (t,c) = head $(\ell_{\alpha 1 \alpha 2})$ we have $m_{\alpha 1}[t,c>m_{\alpha 2}$ and (ii) follows directly from observation O3; otherwise we can get the deleted brother node by means of a symmetry, such that $\varphi^{-1}(m_{\alpha 2}) \geqslant m_2$, and we finish by observation O2.

■

Lemma 6:

Assume SC1, then:

$$\forall m_1, m_2 \in R(m_0) \text{ with } m_1[t,c>m_2$$

$$\exists \varphi \in \Phi$$

$$\exists \alpha 1, \alpha 2 \in T(m_0) \text{ with } \varphi(t,c) \in \ell_{\alpha 1\ \alpha 2}:$$

$$\text{(i)} \quad m_{\alpha 1} \geqslant \varphi(m_1) \quad \wedge$$

$$\text{(ii)} \quad m_{\alpha 2} \geqslant \begin{cases} \varphi(m_2) \text{ if } \varphi(t,c) = \text{head}(\ell_{\alpha 1\ \alpha 2}) \\ \varphi' \circ \varphi(m_2) \text{ for some } \varphi' \in \Phi \text{ otherwise.} \end{cases}$$

Proof:

The existence of φ and α_1 follows from lemma 2 and we have (i). (As in the proof of lemma 2 we choose α_1 to be the node in the actual equivalence class, which is further processed).
Lemma 1 yields:

$$(1) \quad \varphi(m_1)\ [\varphi(t,c) > \varphi(m_2).$$

We can then use observation O3 on (i) and (1) and get

$$(2) \quad m_{\alpha 1}[\varphi(t,c) > \tilde{m} \quad \wedge \quad \tilde{m} \geqslant \varphi(m_2).$$

Hence we can select $\alpha 2$ to be that son of $\alpha 1$, which has $\varphi(t,c) \in \ell_{\alpha 1\ \alpha 2}$. If $\varphi(t,c) = \text{head}(\ell_{\alpha 1\ \alpha 2})$ we have $m_{\alpha 2} = \tilde{m}$ and the second part of (2) gives (ii). Otherwise we have to apply a symmetry as in the proof of lemma 5.

∎

APPENDIX 2: Examples of the use of HL-trees

This appendix completes the two examples described in section 6.

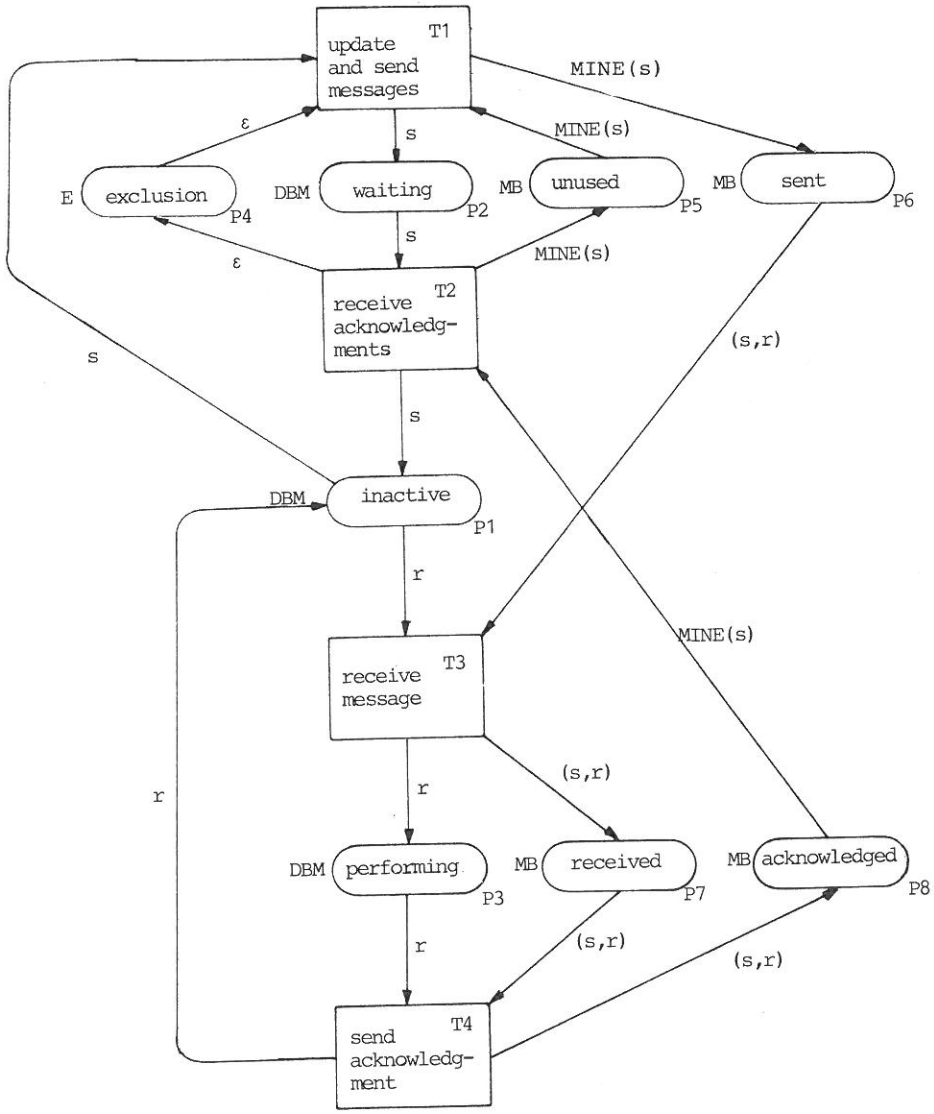
Example 1: Data base system

This example is taken from [4], but originally it was given by Genrich and Lautenbach.

Three database managers, DBM = {a,b,c}, communicate with each other. Each manager can make an update to his own database. At the same time he must send a message to each of the other managers thereby informing them about the update. Having sent this set of messages, the sending manager waits until all other managers have received his message, performed an update and sent an acknowledgment. When all acknowledgments are present, the sending manager returns to be inactive. At that time (but not before) another manager may perform an update and send messages.

Each manager can be in three states: "inactive", "waiting" (for acknowledgments) and "performing" (an update on request of another manager). The managers communicate via a fixed set of message buffers, MB = $\{(s,r) \mid s,r \in DBM \wedge s \neq r\}$, where s represents the sender and r represents the receiver. Each message buffer may be in four different states: "unused", "sent", "received" and "acknowledged".

The system can be described by the HL-net in Figure 6. The corresponding incidence matrix is shown in Figure 7.



$DBM = \{a,b,c\}$ $\quad MB = DBM \times DBM \setminus \{(u,u) \mid u \in DBM\}$ $\quad E = \{\varepsilon\}$

The function $MINE \in [BAG(DBM) \longrightarrow BAG(MB)]_L$ is defined by

$$MINE(s) = \sum_{r \neq s} (s,r) \quad \text{for all } s \in DBM$$

Figure 6. HL-net for the data base system

		T1	T2	T3	T4	m_0
		DBM	DBM	MB	MB	
P1	DBM	-ID	ID	-REC	REC	ΣDBM
P2	DBM	ID	-ID			
P3	DBM			REC	-REC	
P4	E	-ABS	ABS			ε
P5	MB	-MINE	MINE			ΣMB
P6	MB	MINE		-ID		
P7	MB			ID	-ID	
P8	MB		-MINE		ID	

The functions

$$\begin{aligned}
&\text{ID} \in [\text{BAG(DBM)} \longrightarrow \text{BAG(DBM)}]_L \\
&\text{ABS} \in [\text{BAG(DBM)} \longrightarrow \text{BAG(E)}]_L \\
&\text{MINE} \in [\text{BAG(DBM)} \longrightarrow \text{BAG(MB)}]_L \\
&\text{ID} \in [\text{BAG(MB)} \longrightarrow \text{BAG(MB)}]_L \\
&\text{REC} \in [\text{BAG(MB)} \longrightarrow \text{BAG(DBM)}]_L
\end{aligned}$$

are defined by:

$$\begin{aligned}
&\text{ID(s)} = s \quad \text{for all } s \in \text{DBM} \\
&\text{ABS(s)} = \varepsilon \quad \text{for all } s \in \text{DBM} \\
&\text{MINE(s)} = \sum_{r \neq s} (s,r) \quad \text{for all } s \in \text{DBM} \\
&\text{ID((s,r))} = (s,r) \quad \text{for all } (s,r) \in \text{MB} \\
&\text{REC((s,r))} = r \quad \text{for all } (s,r) \in \text{MB}
\end{aligned}$$

Fig. 7. Incidence matrix for the data base system.

We define a partition by:

atomic: DBM: permutation; E: identity

product: MB: subset of DBM×DBM

Soundness criterion SC1 is verified by means of rules R1-R7, given in section 5. By R1, R2 and R4 it is enough to check that the incidence functions ABS, MINE and REC satisfy:

$$\hat{\varphi}_{C(p)} \circ I_{\pm}(p,t)(c) = I_{\pm}(p,t) \circ \varphi_{C(t)}(c)$$

for each $\varphi \in \Phi$ and $c \in C(t)$.

ABS: Let $\varphi \in \Phi$ and $s \in DBM$. Then

$$\varphi_E(ABS(s)) = \varphi_E(\varepsilon) = \varepsilon = ABS(\varphi_{DBM}(s)).$$

MINE: Let $\varphi \in \Phi$ and $s \in DBM$. Then

$$\hat{\varphi}_{MB}(MINE(s)) = \hat{\varphi}_{MB}(\Sigma(s,x)) = \Sigma(\varphi_{DBM}(s),\varphi_{DBM}(x))$$
$$x \neq s \qquad x \neq s$$
$$= \Sigma(\varphi_{DBM}(s),y) = MINE(\varphi_{DBM}(s)).$$
$$y \neq \varphi_{DBM}(s)$$

In this particular case we do not use rule R7, since it is just as easy to prove the property for arbitrary permutations.

REC: Let $\varphi \in \Phi$ and $(s,r) \in MB$. Then

$$\varphi_{DBM}(REC(s,r)) = \varphi_{DBM}(r) = REC(\varphi_{DBM}(s),\varphi_{DBM}(r)) = REC(\varphi_{MB}(s,r)).$$

Soundness criterion SC2 follows immediately from rule R8 in section 5.

Having verified soundness for the partition we can now apply the proof rules on the HL-tree shown in section 6, figure 3.

PR1: The HL-net is bounded.

PR2: All places and all colours in the HL-net have 1 as a uniform bound. This follows from the following observations.

$map_{DBM}(s) = DBM$ for all $s \in DBM$ (by O1)

$map_{E}(\varepsilon) = \{\varepsilon\} = E$ (by O1)

$map_{MB}((s,r)) = MB$ for all $(s,r) \in MB$ (from the definition of map)

PR3: Cannot be applied.

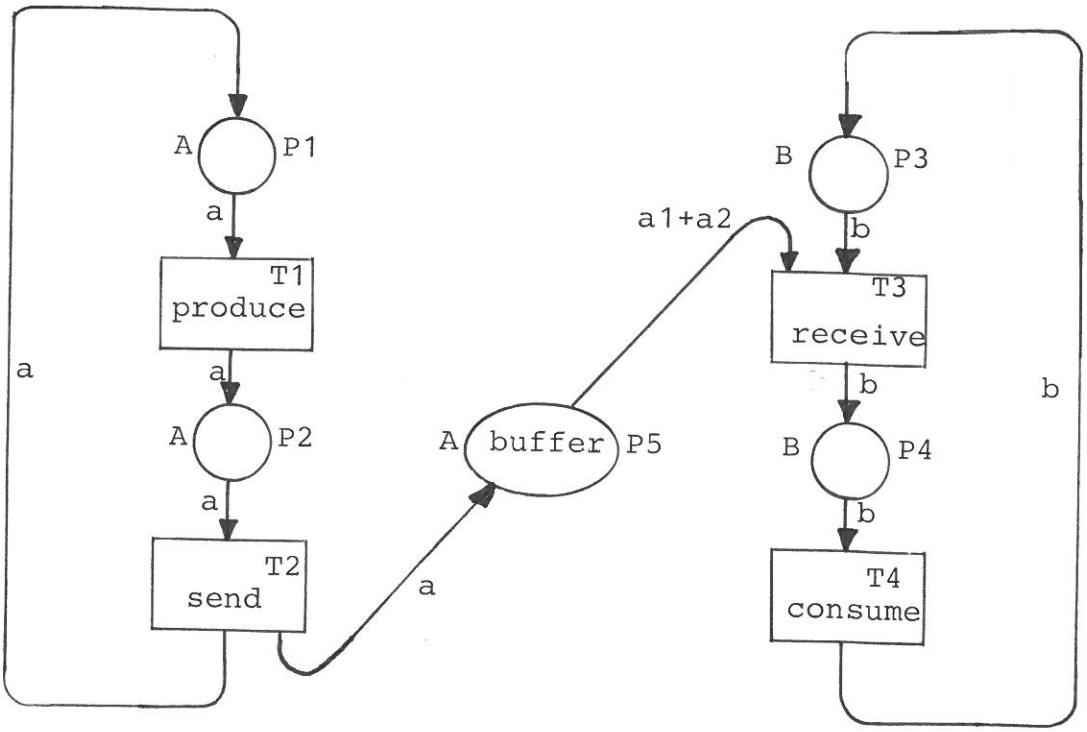
PR4: The HL-net has no reachable marking which is dead.

Example 2:

Producer-consumer system:

Two producers, A = {a1,a2}, each produces their own kind of message, which they repeatedly send to a consumer via an unbounded buffer. The consumer can only receive pairs of messages consisting of one message from each producer).

The system can be described by the HL-net in figure 8. The corresponding incidence-matrix is shown in figure 9.



A={a1,a2}

B={b}

Figure 8. HL-net for the producer-consumer system.

		T1	T2	T3	T4	m_0
		A	A	B	B	
P1	A	-ID	ID			ΣA
P2	A	ID	-ID			
P3	B			-ID	ID	ΣB
P4	B			ID	-ID	
P5	A		ID - CONNECT			

The function:

$$CONNECT \in [BAG(B) \longrightarrow BAG(A)]_L$$

is defined by:

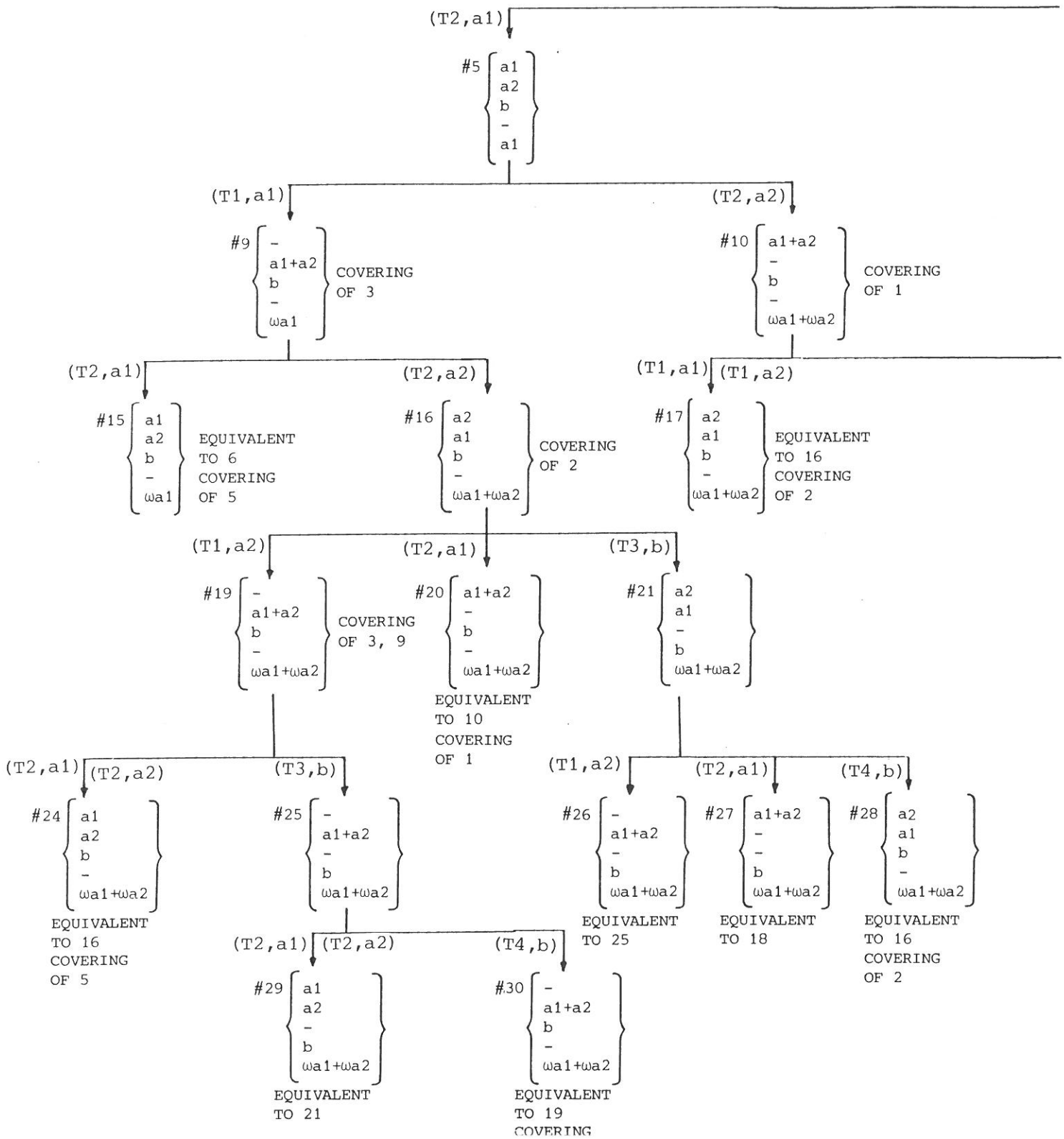
$$CONNECT(b) = a1+a2$$

Figure 9. Incidence matrix for the producer-consumer system.

We define a partition by:

atomic A: permutation; B: identity.

Soundness of the partition follows immediately from rule R1-R8 in section 5. One of the corresponding HL-trees is shown in figure 10.


(T2,a1)
#5 a1 a2 b - a1
(T1,a1)
(T2,a2)
#9 - a1+a2 b - ωa1
COVERING OF 3
#10 a1+a2 - b - ωa1+ωa2
COVERING OF 1
(T2,a1)
(T2,a2)
(T1,a1)
(T1,a2)
#15 a1 a2 b - ωa1
EQUIVALENT TO 6 COVERING OF 5
#16 a2 a1 b - ωa1+ωa2
COVERING OF 2
#17 a2 a1 b - ωa1+ωa2
EQUIVALENT TO 16 COVERING OF 2
(T1,a2)
(T2,a1)
(T3,b)
#19 - a1+a2 b - ωa1+ωa2
COVERING OF 3, 9
#20 a1+a2 - b - ωa1+ωa2
EQUIVALENT TO 10 COVERING OF 1
#21 a2 a1 - b ωa1+ωa2
(T2,a1)
(T2,a2)
(T3,b)
(T1,a2)
(T2,a1)
(T4,b)
#24 a1 a2 b - ωa1+ωa2
EQUIVALENT TO 16 COVERING OF 5
#25 - a1+a2 - b ωa1+ωa2
#26 - a1+a2 - b ωa1+ωa2
EQUIVALENT TO 25
#27 a1+a2 - - b ωa1+ωa2
EQUIVALENT TO 18
#28 a2 a1 b - ωa1+ωa2
EQUIVALENT TO 16 COVERING OF 2
(T2,a1)
(T2,a2)
(T4,b)
#29 a1 a2 - b ωa1+ωa2
EQUIVALENT TO 21
#30 - a1+a2 b - ωa1+ωa2
EQUIVALENT TO 19 COVERING

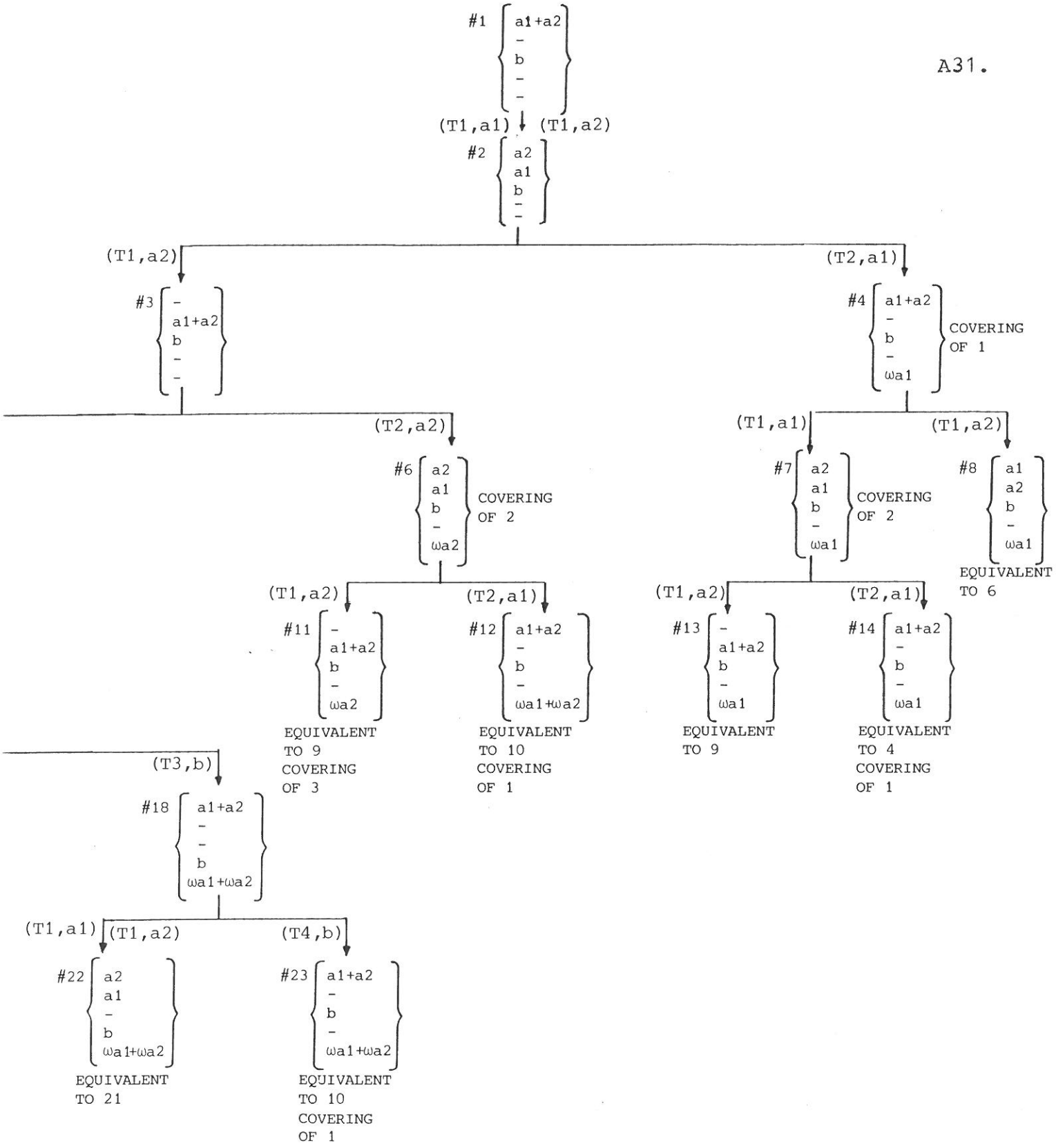


Fig. 10: HL-tree for the producer-consumer system.

We can now apply the proof rules.

PR1: The HL-net is unbounded.

PR2: The places P1-P4 are bounded for all colours, with 1 as a uniform bound.
The place P5, which represents the buffer, is unbounded for both colours in its colour set.

PR3: Cannot be applied.

PR4: The HL-net has no reachable marking which is dead.

APPENDIX 3: An algorithm to determine whether two markings are equivalent or not

In this appendix we sketch an algorithm to test two ω-markings for equivalence.
The algorithm is an integral part of the algorithm to produce HL-trees, presented in section 4. The equivalence-algorithm consists of six steps. However, before presenting the individual steps, we consider the internal representation used for ω-bags.

When an atomic or related colour set is defined, we demand the user to give it a finite, ordered set of element names (e.g. PH = $\{ph_1, ph_2, \cdots, ph_5\}$). In the HL-tree constructor we shall, however, always use small letters a,b,c,$\cdots$ to represent the different colours. Thus the bag $ph_2+ph_3+ph_4+ph_5$ (appearing in node #2 of fig. 2) is represented as b+c+d+e. Moreover we use a special representation of formal sums, which is convenient for a computerized version of the algorithm. Although the constructor apply this internal representation, it should be noted that the user always delivers input and receives output in the usual bag representation (formal sums and original colour names).

The internal representation saves space, by avoiding long element names. But it also makes it much easier to work with related colour sets. When a related colour set is defined, the order of the elements implicitly determine its relation to the corresponding atomic colour set, in the sense that element no. i of the related colour set is mapped into element no. i of the atomic colour set. This implies, that the bag $f_3+f_4+f_5$ over $F = \{f_1, f_2, \cdots, f_5\}$ (appearing in node #2 in fig. 2) is represented as c+d+e, which immediately can be understood as a bag over the atomic colour set PH. Thus bags over a related colour set can be handled in exactly the same way as bags over the corresponding atomic colour set, except that the small letters are translated into other element names when the final output is generated.

In fact the only purpose of related colour sets is to allow the system describer to attach mnemonic names to the elements. Without related colour sets the describer would have to name the token-colours representing the forks in the philosopher system by PH = $\{ph_1, ph_2, \cdots, ph_5\}$.

The next question is whether we also can handle product colour sets in the same way as atomic ones. Again the answer is confirmative. The bag (b,a)+(b,c)+(c,a)+(c,b) (appearing in node #2 of fig. 3) can be handled as two separate bags 2b+2c and 2a+b+c, obtained by the projections mapping (s,r) $\in$ MB into s and r, respectively. For each place with a product colour set, we will have as many bags as there are components in the product. Each of these bags will be handled in the same way as bags originating from places with atomic or related colour sets.

We summarize:

Given an atomic colour set C, we get bags over C (represented as bags over {a,b,c,$\cdots$}) in three different ways:

- from places, which have C as atomic colour set
- from places, which have a colour set related to C
- from places, which have a product colour set containing C (or a related colour set) as a component.

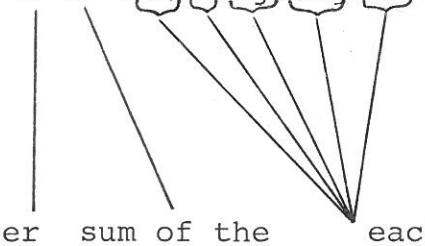
However, we are able to handle all these bags in exactly the same way, independent of their origin. †

---

† As we shall see later, this is not fully correct for bags originating from products. For this kind of bags we have to add an extra check.

We will represent bags in a rather special way, which turns out to be convenient for our algorithm. The bag 3a+2c+ωd+e+f+g+ωh+2j over {a,b,····,j} is represented by :

(*) 2-10-Wdh3a2cj1efg0bi



number of ω. sum of the remaining coefficients. each of these groups contains, in alphabetic order, those elements, which have the coefficient immediately preceeding the group; the groups are ordered by descending coefficients.

Alternatively,to save space we can choose the most frequent coefficient to be default, and remove the corresponding elements from the representation:

(**) 2-10-1-Wdh3a2cj0bi

most frequent coefficient (if two different coefficients occur equally often we choose the smaller one).

The two (three) opening numbers of our representation (2,10(and 1) in the above example) can be coded into a single integer, while the rest of the representation can be kept in a textstring.

We will now describe our algorithm to determine whether two markings are equivalent or not. To understand the algorithm it should be noted that all our symmetries consist of bijections. This implies that they cannot alter the number of ω-coefficients in a bag, nor the sum of the remaining coefficients. Moreover if we consider the coefficients as a bag (over $\mathbb{N} \cup \{\omega\}$) this bag cannot be altered by applying a symmetry.

The algorithm consists of a number of steps. In each step some properties are checked. If a property is found to be unsatisfied the markings in question are known to be non-equivalent, and there is no need to perform the remaining steps. However, if all steps are successfully performed the two markings are known to be equivalent. We have tried to arrange the steps in such an order, that the early steps take little time, but still have a large probability to disclose non-equivalency. For this reason the steps are not independent. Some of the early steps could be removed, since their properties would follow from the properties of a later step. This might, however, increase the computation time.

Let m_1 and m_2 be the two markings to be tested for equivalency. We assume the two markings to be in the form (*). †

Step 1: Check number of tokens

For each place $p \in P$, check whether $m_1(p)$ and $m_2(p)$ have the same number of ω and that their sums of remaining coefficients are identical.

By the way we represent bags the checks above only involve a single comparison of two integers (each representing number of ω and sum of remaining coefficients).

---

† In step 1-3 the markings can be represented either in the form (*) or (**). In steps 4-6 they must be in the form (*), which can be regenerated from (**).

Step 2: Check coefficients

For each place $p \in P$, check whether the coefficients in $m_1(p)$ and $m_2(p)$ form the same bag (over $\mathbb{N} \cup \{\omega\}$)†.

By the way we represent bags we only have to read through two textstrings, text1 and text2, checking that in each position i, either text1[i] and text2[i] are both small letters or they are identical.

However, to make step 3 easier we shall, during the check above, also record whether each pair of textstrings are identical or not. We carry on with step 3 even if the pairs of textstrings are not totally identical.

Step 3: Check identity colour sets

For each atomic colour set C with sym(C) = identity, check that the bags over C†† are identical in m_1 and m_2.

This was already recorded in step 2.

---

† If C(p) is a product set we check the coefficients for each component bag.

†† Obtained from places in the three different ways, described on page A34.

Step 4: Check rotation colour sets

For each atomic colour set C with sym(C) = rotation, check that there exists a rotation, which maps each bag over C in m_1 into the corresponding bag in m_2.

This check is performed in the following way. The number of possible rotations equals the finite number of elements in C, and thus we can use a boolean array to indicate which rotations are still possible candidates.

Initially all rotations are candidates. For the first bag in m_1 we consider each rotation, one at a time, and record whether it maps the bag into the corresponding bag in m_2 or not. This can be done by reading the two textstrings representing the bags and checking whether they are identical when the elements of the first textstring are rotated. The rotation may change the alphabetic order of a group of elements having the same coefficient. Thus we must reorder each group after its rotation, before comparison with the corresponding group in the other textstring. The most efficient way is to rotate and compare one group at a time. Then we can stop as soon as we find a discrepency.

When all rotations have been considered for the first bag, we continue with the next bag,but this time it is of course only necessary to consider those rotations which worked for the first bag. In this way we continue until we either have no more rotations as possible candidates (in which case the step fails) or we have considered all bags (in which case we continue with the next atomic colour set having rotation as symmetry type).

When we have finished a colour set the boolean array indicates the possible rotations, which can be used to map m_1 into m_2. If the bag originates from a product †, the possible rotations are recorded for later use in step 6.

---

† See page A34.

Step 5: Check permutation colour sets

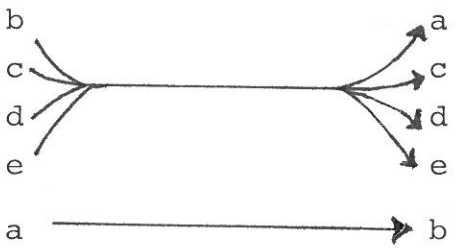
For each atomic colour set C with sym(C) = permutation, check that there exists a permutation, which maps each bag over C in m_1 into the corresponding bag in m_2.

We could consider each permutation one at a time and thus proceed in a way analogous to step 4. However, the number of possible permutations grows with the square of C's size. Thus we shall perform the check in another way.

Consider the first bag of m_1. For each group of elements having the same coefficient we know that the elements by a possible permutation must be mapped into the elements of the corresponding group in the first bag of m_2.

As an example consider the two markings m_1 and m_2 defined for the philosopher system in section 3, and let us for a moment assume that sym(PH) = permutation. The first bag of m_1 is represented by the textstring 1bcde0a while the first bag of m_2 is represented by 1acde0b. This implies the following connection between the elements, where each multiarc indicates that the source elements must be mapped by the permutation into the target elements:

(1)

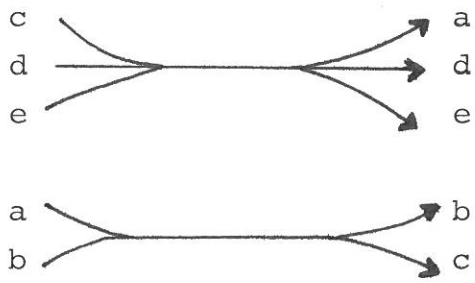


Thus we know that 'a' must be mapped into 'b', but for the other elements we only know that they are not mapped into 'b'.

When this has been done for the first bag, we continue with the next bag, but this time we will have to elaborate on the already constructed multiarcs by splitting some of the arcs into one or more parts.

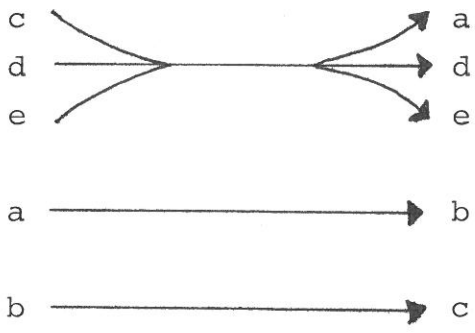
In our example the second bags of m_1 and m_2 are 1a0bcde and 1b0acde, respectively. This yields no further information and the multiarc diagram (1) is unaltered.
However, the third bags are 1cde0ab and 1ade0bc, respectively. This yields the following multiarc diagram:

(2)



and by combining (1) and (2), we obtain

(3)



If two multiarc diagrams cannot be combined into a single one, because they contain conflicting information, no permutation is possible and the step fails. If, however, all multiarc diagrams can be combined, the final diagram indicates the possible permutations. In our example the diagram (3) tells us that there are six possible permutations. † If the bag originates from a product, the possible permutations are recorded for later use in step 6.

† Exactly one of them is a rotation.

When we have successfully considered all bags for a given colour set, we continue with the next atomic colour set having permutation as symmetry type.

Before we present step 6, we have to consider products more closely. When a place has a product as colour set we obtain a number of bags from that place, by means of projections (in the way described on page A34). Until now we have handled these bags in exactly the same way, as bags originating from places with atomic or related colour sets. This introduces, however, a problem. Consider a product {a,b}×{a,b}, where the two components represent the same atomic or related colour set. According to our rules, each of the two bags

(a,b) + (b,a) and (a,a) + (b,b)

over {a,b}×{a,b}, is split into two bags over {a,b}

a+b a+b

and

b+a a+b

Independent of the symmetry type for {a,b} it is obvious that step 3-5 is successful for the upper set of bags and for the lower set. There is, however, no identity, rotation or permutation which maps (a,b) + (b,a) into (a,a) + (b,b). Each function in a symmetry is bijective, and thus it cannot map a pair, (a,b) or (b,a), consisting of two different elements into a pair, (a,a) or (b,b), consisting of the same element twice. † The problem arises because we have ignored that the product binds elements together into tuples. By using projections to create bags, over the components of the product, we split up the tuples without considering this binding.

---

† The argument above can be elaborated also to cover products consisting of unrelated components.

Step 6: Check products

Check whether any of the possible symmetries, calculated in step 3-5, maps m_1 into m_2.

For each place p with an atomic or related colour set, the construction method in step 3-5 guarantees that all possible symmetries, which we have calculated, map $m_1(p)$ into $m_2(p)$. Thus we only have to consider places, which have products as colour sets, and this check is done by means of a backtrack-algorithm, which stops as soon as we have found one symmetry mapping m_1 into m_2.

This step may, in the worst cases, take up a considerable amount of computing time. It should, however, be remembered that the check only involves the places with products. Moreover, we conjecture, that in most cases there will be only a few possible symmetries, or they will nearly all map m_1 into m_2.