数理解析研究所講究録
996巻 1997年 207-221

Towards Cycle Filling as Parallelization

Yukihide Takayama
Department of Computer Science, Ritsumeikan University
1916 Noji-cho, Kusatsu-shi, Shiga 525, Japan
email: takayama@theory.cs.ritsumei.ac.jp

September 24, 1996

Abstract

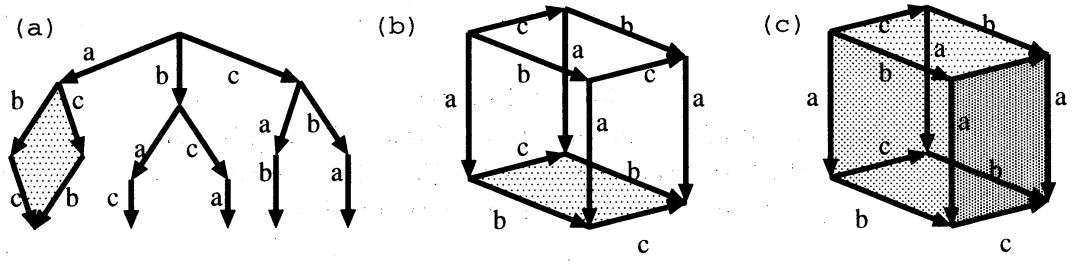
We give a new formulation of Higher-Dimensional Automata (HDA) called *cubical Bar-notation*. It gives an intrinsic definition of hypercubes which are building blocks of HDA. The gain is an explicit and simple definition of cycle filling map. Also we have an explicit representation of Hoare traces and Mazurkiewicz traces. The cycle filling map can be used to realize the idea of *cycle filling as parallelization* [2, 8]. Combined with the extraction algorithm of process expressions from HDA [9], the cycle filling map will realize a method of program transformation in concurrent programming languages as geometric transformation of higher dimensional spaces.

1 Introduction

In this paper, we will consider a parallelization problem in process algebra, which is stated as follows: Given a process expression P, how can we mechanically find a process expression Q such that P and Q are bisimilar and Q is more parallel than P? For example, $Q = (a \parallel b \parallel c)$ is a *parallelization* of $P = a\ ;\ (b \parallel c) + b\ ;\ (a\ ;\ c + c\ ;\ a) + c\ ;\ (a\ ;\ b + b\ ;\ a)$ where $(- \parallel -)$ and $(-\ ;\ -)$ are parallel composition and sequential composition. There may be many approaches to this problem. One candidate is the 'decomposition' method [4]. By decomposition we can check whether Q is a parallelization of P, but an explicit algorithm to find Q from P is not given. Another candidate is Mazurkiewicz style trace theory [5]. Let $[\![P]\!] =< H_P, I_P >$ be the interpretation of P. $H_P = \{abc, acb, bac, bca, cab, cba\}$ is the set of maximal Hoare traces for P and $I_P = \{< b, c >_a\}$ is the set of *prefixed* independence relation where $< b, c >_a$ means $abc = acb$. Then we find new set of relations $I = \{< a, b >, < b, c >, < c, a >, < c, a >_b, < a, b >_c\}$ with some suitable algorithm, and merge I and I_P to obtain the merged relation $J = \{< a, b, c >\}$, which means a, b and c are pairwise commutative. Then the obtained interpretation $< H_P, J >$ is equal to $[\![Q]\!]$. So we obtain a parallelization of P at the semantics level. However, for comparing the degree of parallelism between P and Q and for extracting the expression Q from the semantical parallelization $< H_P, J >$, we would need a richer structure in the semantics domain.

The approach we will take in this paper is based on Higher-Dimensional Automata (HDA) [6, 10] model of true concurrency. HDA model is a geometricly formalized noninterleaving transition systems. In the model, we interpret the parallel run of n atomic actions by the n-dimensional hypercube (n-cube) spanned by the arcs corresponding to the atomic actions. On the other hand, nondeterministic run of n atomic actions is interpreted simply by the tree spanned by

the arcs. Hence the degree of parallelism is simply the dimension of the semantics space. For example, the interpretation of Q above is the 3-cube spanned by arcs a, b and c while the interpretation of P above is the space (a) in the following figure. We can see that the degree of



parallelism of P is 2 since there is a 2-cube in (a). Also, by identifying some vertices and edges in this space, we obtain the space (b). We find five 2-dimensional cycles in the space. Namely, the two closed paths a, b, a, b, two closed paths a, c, a, c and one closed paths c, b, c, b. If we fill these cycles with five 2-cubes, we obtain the space (c). This space has a higher-dimensional cycle, namely the 3-dimensional empty space surrounded by the 2-faces of the cube. Finally, we fill the cycle with a 3-cube and we obtain the interpretation of Q. This is the idea of *cycle filling as parallelization* which has been introduced by E. Goubault [2] in a more general setting and by the author [8] in a specific process language. Also, the author gave an algorithm to extract the process expression Q from the obtained space [9].

In this paper, we will refine the idea of cycle filling as parallelization. We give a new formulation of HDA by introducing new system of notation called *cubical Bar-notation*. As compared to the formalization of HDA as seen in [10, 3, 1], our formalization gives a intrinsic definition of n-cubes and a HDA is defined as a well-labeled cubical complex ($\mathcal{WCC}$), which is roughly a collection of cubes of various dimensions. Also, for application to process algebra we will introduce reachable well-labeled cubical complexes ($\mathcal{RWCC}$). The advantages of the $\mathcal{RWCC}$ are, first of all, we can formally define the cycle filling map, and secondly, $\mathcal{RWCC}$ explicitly contains Mazurkiewicz style traces. Hence our model is an extension of Mazurkiewicz traces.

We will now sketch the configuration of this paper. We will develop the formalization of $\mathcal{RWCC}$ in the next section. We do not employ a particular process language and its interpretation by $\mathcal{RWCC}$. Rather, we work in a setting where processes are viewed as higher-dimensional transition systems. So in section 3 we define similar process constructions to those in CCS-like process languages in the domain of $\mathcal{RWCC}$. Section 4 formalizes the parallelization problem in the language of $\mathcal{RWCC}$ and defines the parallelization algorithm with cycle filling map. The concluding remark will be given in the final section.

2 Reachable Well-labeled Cubical Complexes: $\mathcal{RWCC}$

Intuitively, a HDA is a higher-dimensional space which is a collection of cubes of various dimensions. The cubes are glued to each other at their boundaries, so that the 'shape' of a HDA is formally described by the boundary operators. Traditionally, the cubes are defined externally. Namely, a cube is an element of a set satisfying some properties. We will give here an internal definition of cubes with *cubical Bar-notation* and a HDA is formalized as a well-labeled cubical complex ($\mathcal{WCC}$). For application to process algebra, we will also define reachable cubical complexes ($\mathcal{RWCC}$).

2.1 n-cubes in cubical Bar-notation

Let $\mathcal{A}$ be a denumerable set of symbols called labels, and let $\mathcal{M}$ be a monoid with unit 1 generated over $\mathcal{A}$. We will view the elements of $\mathcal{M}$ as words over $\mathcal{A}$ and view 1 as the empty word. We first define *pre n-cubes* in cubical Bar-notation over $\mathcal{M}$.

Definition 1: A *pre n-cube* is a cubical Bar-expression $w[a_1 \mid a_2 \mid \cdots \mid a_n]$ with $w \in \mathcal{M}$ and distinct elements $a_1, \ldots, a_n \in \mathcal{A}$. w will be called the *base point*.

Intuitively, a pre n-cube $w[a_1 \mid a_2 \mid \cdots \mid a_n]$ is a cube spanned by n arcs $a_1, \ldots, a_n$ whose common starting point is w. Next we will consider the boundary operators of pre n-cubes. For example, the boundary of the pre 2-cube $w[a \mid b]$ is the collection of four line segments which are also pre 1-cubes. In cubical Bar-notation this collection is described as $\{w[a], w[b], wa[b], wb[a]\}$. Also, the base point wa of the pre 1-cube $wa[b]$, for example, is one of the end points of the pre 1-cube. Another end point is represented by the word wab. This situation is illustrated in Figure 1, where the term 'commutativity constraint' will be explained later.

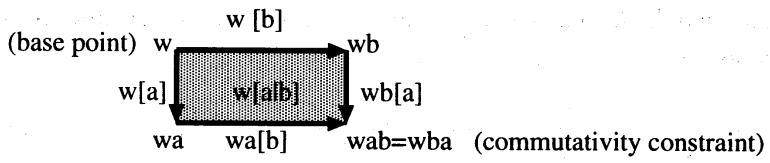


Figure 1: (pre) 2-cube $w[a \mid b]$

Now we define the boundary operators for pre n-cubes.

Definition 2: Let C_n be the set of pre n-cubes. Notice that the elements of C_0 is in the form of $w[]$ ($w \in \mathcal{M}$). Hence we will identify C_0 with $\mathcal{M}$ by the bijection $w[] \leftrightarrow w$. Then, the *set theoretic boundary operator* d_n is the map $d_n = d_n^0 \cup d_n^1 : C_n \longrightarrow C_{n-1}$ where

$$d_n^0(w[a_1 \mid \cdots \mid a_n]) = \bigcup_{i=1}^{n} \{w[a_1 \mid \cdots \overset{i}{\vee} \cdots \mid a_n]\}, \quad d_n^1(w[a_1 \mid \cdots \mid a_n]) = \bigcup_{i=1}^{n} \{wa_i[a_1 \mid \cdots \overset{i}{\vee} \cdots \mid a_n]\}$$

when $n \geq 2$. "$\cdots \overset{i}{\vee} \cdots$" means omitting a_i. $d_1^0(w[a]) = \{w\}$ and $d_1^1(w[a]) = \{wa\}$.

For algebraic treatment of cubes, it is often convenient to represent the boundary of a pre-cube as the formal sum of pre-cubes with boolean coefficients. Hence we also give another definition.

Definition 3: Let S_n be the set of formal sums of pre n-cubes with boolean coefficients. In other words, S_n is the vector space over $\mathbf{Z}/(2)$ generated by pre n-cubes. As in the previous definition, we will identify S_0 with the vector space $V(M)$ over $\mathbf{Z}/(2)$ generated by $\mathcal{M}$. Then, the *boundary operator* ∂_n is the linear map $\partial_n = \partial_n^0 + \partial_n^1 : S_n \longrightarrow S_{n-1}$ where

$$\partial_n^0(w[a_1 \mid \cdots \mid a_n]) = \Sigma_{i=1}^{n} w[a_1 \mid \cdots \overset{i}{\vee} \cdots \mid a_n], \quad \partial_n^1(w[a_1 \mid \cdots \mid a_n]) = \Sigma_{i=1}^{n} wa_i[a_1 \mid \cdots \overset{i}{\vee} \cdots \mid a_n]$$

when $n \geq 2$. $\partial_1^0(w[a]) = w$ and $\partial_1^1(w[a]) = wa$. We will often call ∂_n a *total boundary* and will call ∂_n^j ($j = 0, 1$) a *j-boundary*.

In the context where the dimension 'n' is understood, we will sometimes denote as d, ∂, ∂^0 and ∂^1 by omitting the suffix 'n'. Notice that the operators have the property $\partial\partial = \partial^j\partial^j = 0$ ($j = 0$ or 1), $\partial^0\partial^1 = \partial^1\partial^0$ and $d^0d^1 = d^1d^0$.

An n-cube is a pre n-cube together with *commutativity constraint*. For example, consider again the pre 2-cube $w[a \mid b]$ in Figure 1. The boundary is $\partial(w[a \mid b]) = w[b] + w[a] + wa[b] + wb[a]$. If we wish $w[a \mid b]$ to be a real cube, $w[b]$, $w[a]$, $wa[b]$ and $wb[a]$ must meet at the four vertices of the cube and the vertices must be the boundaries of these pre 1-cubes. Thus $w[a \mid b]$ is a real cube if and only if $wab = wba$. This equational relation will be called the commutativity constraint.

Definition 4: An *n-cube* is a pre n-cube $w[a_1 \mid a_2 \mid \cdots \mid a_n]$ together with the commutativity constraint $< a_1, \ldots, a_n >_w$ which means that, for arbitrary $\{i_1, i_2, \ldots, i_k\} \subset \{1, 2, \ldots, n\}$, $wa_{i_1}a_{i_2}\cdots a_{i_n} = wa_{\sigma(i_1)}a_{\sigma(i_2)}\cdots a_{\sigma(i_k)}$ in $\mathcal{M}$ for all permutation $(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$ of the sequence $(i_1, i_2, \ldots, i_k)$. We will always denote $< a_1, \ldots, a_n >_1$ simply by $< a_1, \ldots, a_n >$. The dimension function of cubes is defined by $dim(w[a_1 \mid a_2 \mid \cdots \mid a_n]) = n$. The boundary operators for n-cubes are the same as those for pre n-cubes.

2.2 Product and Decomposition of Cubes

Now we will define two basic construction and destruction operators for n-cubes. Namely, *product* and *decomposition*.

Definition 5: Let $w[a_1 \mid \cdots \mid a_m]$ and $u[b_1 \mid \cdots \mid b_n]$ $(m, n \geq 0)$ be cubes such that $\{a_1, \ldots, a_m\} \cap \{b_1, \ldots, b_n\} = \phi$. Then their product is

$$w[a_1 \mid \cdots \mid a_m] \star u[b_1 \mid \cdots \mid b_n] = wu[a_1 \mid \cdots \mid a_m \mid b_1 \mid \cdots \mid b_n]$$

together with the commutativity constraint: $< a_1, \ldots, a_m, b_1, \ldots, b_n >_{wu}$.

Notice that the product is a non-commutative operation because wu and uw are not always equal. Next we will define the decomposition operation. We can view that an n-cube C consists of the interior of C and its boundary $B = d(C)$. Also, each element of B has the similar structure. This representation is called *cellular decomposition*.

Definition 6: For an n-cube $w[a_1 \mid \cdots \mid a_n]$, we define the sets $B_0, B_1, \cdots, B_n$ inductively as follows: $B_n = \{w[a_1 \mid \cdots \mid a_n]\}$ and, for $0 \leq k < n$, $B_k = \{k\text{-cube } C \mid C \in d(C') \text{ for some } C' \in B_{k+1}\}$. Then we define the *cellular decomposition* by $decomp(w[a_1 \mid \cdots \mid a_n]) = \bigcup_{k=0}^{n} B_k$. Also, for a set S of cubes we naturally extend the map: $decomp(S) = \{decomp(c) \mid c \in S\}$.

The cellular decomposition is characterized as follows.

Lemma 1 *For a cube C, (i) $decomp(decomp(C)) = decomp(C)$, (ii) for arbitrary cube D, $D \in decomp(C)$ if and only if $D \in d^n(C)$ for some $n \geq 0$.*

Example 1: $decomp(w[a \mid b \mid c]) = B_0 \cup B_1 \cup B_2 \cup B_3$ where

$$B_0 = \left\{ \begin{array}{l} w, wa, wb, wc, wba(= wab), wac(= wca), wcb(= wbc), \\ wcba(= wbca = wcab = wacb = wbac = wabc) \end{array} \right\}$$

$$B_1 = \left\{ \begin{array}{l} w[a], w[b], w[c], wb[a], wa[b], wa[c], wc[a], wc[b], wb[c], \\ wcb[a](= wbc[a]), wca[b](= wac[b]), wba[c](= wab[c]) \end{array} \right\}$$

$$B_2 = \{w[a \mid b], w[a \mid c], w[b \mid c], wc[a \mid b], wb[a \mid c], wa[b \mid c]\}, \quad B_3 = \{w[a \mid b \mid c]\}$$

and the equalities between elements in B_0 and B_1 are due to the commutativity constraint $< a, b, c >_w$.

2.3 Well-labeled Cubical Complexes

Now we define well-labeled cubical complexes.

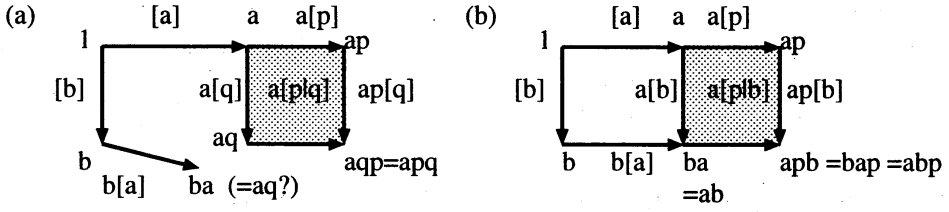
Definition 7: Let C be a set of cubes with $1 \in C$ and $C_0(\subseteq C)$ be the set of 0-cubes in C. Then, a *well-labeled cubical complex* ($\mathcal{WCC}$) is the tuple $S = (S_{<C>}, \partial^0, \partial^1, F, Const)$ where

- $S_{<C>}$ is the vector space over $\mathbf{Z}/(2)$ generated by $decomp(C)$. $S_{<C>}$ will often be represented by the stratified form. Namely, $S_{<C>} = \bigoplus_{n\geq 0} S^n_{<C>}$ where $S^n_{<C>}$ is the vector space over $\mathbf{Z}/(2)$ generated by the set $C_n = \{decomp(c) \in C \mid dim(c) = n\}$. C_n will be also denoted by $Base(S, n)$.
- $\partial^0, \partial^1 : S_{<C>} \to S_{<C>}$ are the j-boundary operators.
- F is a subset of C_0.
- $Const$ is the collection of commutativity constraints of the cubes in C and other commutativity constraints. The equational reasoning for the elements in $S_{<C>}$ is carried out with $Const$ and the following **Right Extension Rule**

$$w_1 = w_2 \ (w_1, w_2 \in C_0) \Rightarrow w_1 M = w_2 M \ (M \in \mathcal{A} \text{ or } M = [a_1 \mid \cdots \mid a_n], a_i \in \mathcal{A})$$

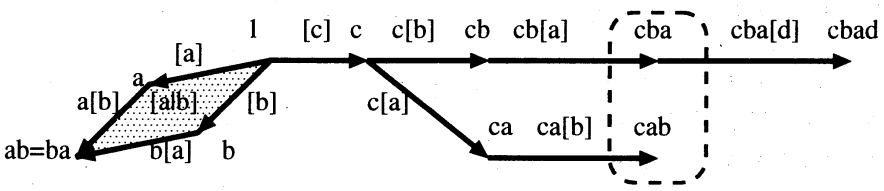
and the reflexity, symmetry and transitivity rules.

We will briefly explain 'well-labeledness' of $\mathcal{WCC}$. Consider the $\mathcal{WCC}$ $(S_{<C>}, \partial^0, \partial^1, \{apq\}, \{< p, q >_a\})$ with $C = \{[a], [b], b[a], a[p \mid q]\}$ (figure (a) below). We cannot define the space in which



the labels ba and aq denotes the same point because we cannot represent the equation $ba = aq$ by a commutativity constraint. However, we can define a space which has the desired shape by changing the labels. Namely, the complex $(S_{<D>}, \partial^0, \partial^1, \{apb\}, \{< p, b >_a, < a, b >\})$ with $D = \{[a], [b], b[a], a[p \mid b]\}$ (figure (b)). Notice that $< p, b >_a$ is the commutativity constraint accompanied with the cube $a[p \mid b]$, but the constraint $< a, b >$ is the additional one only to link the points ba and ab. The equation $ba = ab$ and $apb = abp$ follow directly from the commutativity constraints, but the equation $bap = abp$ follows from the constraint $< a, b >$ and the right extension rule.

The reason why we do not allow 'left extension rule', $w_1 = w_2 \Longrightarrow \alpha w_1 = \alpha w_2$ $(\alpha \in \mathcal{A})$, is that the commutativity constraint is of local nature. Consider the $\mathcal{WCC}$ in the following figure.



We have $< a, b >$ because of the existence of the 2-cube $[a \mid b]$. If we allow left extension, we obtain $< a, b >_c$, namely $cab = cba$, but this does not hold in this $\mathcal{WCC}$.

2.4 Reachable Well-labeled Cubical Complexes

Now we will define reachable well-labeled cubical complexes ($\mathcal{RWCC}$). By reachable, we mean that we can arrive at any point in the space by a path from the distinguished point 1. Hence we will first define the notion of paths in cubical complexes.

Definition 8: Let $S = (S_{<C>}, \partial^0, \partial^1, F, Const)$ be a $\mathcal{WCC}$. Then a sequence $(c_1, c_2, \ldots, c_t)$ of elements from C_1 with $\partial^0(c_1) = 1$ and $\partial^1(c_i) = \partial^0(c_{i+1})$ $(i = 1, \ldots, t-1)$ is called a *path* of S.

Definition 9: A $\mathcal{RWCC}$ $S = (\bigoplus_{n\geq 0} S^n_{<C>}, \partial^0, \partial^1, F, Const)$ is a $\mathcal{WCC}$ such that, for arbitrary $w \in C_0$, there is a path $(c_1, \ldots, c_t)$ in S with $\partial^1(c_t) = w$. A $\mathcal{RWCC}$ will be often denoted simply by $S_{<C>}$ or $\bigoplus_{n\geq 0} S^n_{<C>}$ where other parts are not important.

3 $\mathcal{RWCC}$ as Noninterleaving Transition System

HDA formalized as $\mathcal{RWCC}$ has nice properties. Namely, we can clearly understand relation between the general notion of HDA model and traditional noninterleaving semantics theories in $\mathcal{RWCC}$. Because the cubes, which are building blocks of HDA, are defined internally with cubical Bar-notation, we can describe Hoare traces explicitly in our HDA model. Moreover, the commutativity constraints can be understood as the prefixed independence relations for Hoare traces explained in the introduction. Therefore, $\mathcal{RWCC}$ explicitly contains a Mazurkiewicz style trace theory. Also, the commutativity constraints describe the homotopy relation which is a central notion of HDA model. In this section, we show the relation with Hoare traces and Mazurkiewicz style trace theory. Also, we define typical process constructions similar to those in CCS-like process languages in the domain of $\mathcal{RWCC}$. For the simplicity, we will assume that a $\mathcal{RWCC}$ $S_{<C>}$ is such that $decomp(C) = C$ in the following.

3.1 Hoare Traces, Homotopy and Mazurkiewicz Traces

We will briefly review the idea of HDA as a noninterleaving transition system. Now we will reconsider the 2-cube $w[a \mid b]$ in Figure 1. This can be understood as a noninterleaving transition system when $w = 1$. We find two paths $p_1 = ([a], a[b])$ and $p_2 = ([b], b[a])$. p_1 represents the trace of action a followed by action b. Also, p_2 represents the trace of action b followed by action a. Thus the paths in $\mathcal{RWCC}$ represent Hoare traces. Then, what does the 2-cube $w[a \mid b]$ mean? Geometrically, the paths p_1 and p_2 can be continuously translated to each other through the 2-cube. We say this situation that the paths p_1 and p_2 are *homotopic*. According to the theory of HDA model, the 2-cube means the higher-dimensional state, or higher-dimensional transition, in which the actions a and b may run in parallel. Hence, true concurrency is represented by the homotopy relation between paths caused by the existence of higher-dimensional spaces between them. We can also consider higher-dimensional homotopy of higher dimensional paths, but the following lower-dimensional homotopy suffices to our discussion.

Definition 10: Two paths $p = (p_1, p_2, \ldots, p_m)$ and $q = (q_1, q_2, \ldots q_n)$ in a $\mathcal{RWCC}$ $S_{<C>}$ are said to be *adjacent*, $p \approx q$, if $n = m$, $\partial^1(p_m) = \partial^1(q_n)$ and there exists i, $1 \leq i < n$ such that

1. $p_k = q_k$ for all $1 \leq k < i$ and $i + 1 < k \leq n$
2. $p_i + p_{i+1} + q_i + q_{i+1} = \partial(c)$ for some $c \in C_2$.

Homotopy is the smallest equivalence relation on the paths containing adjacency.

Now we will consider Hoare traces in $\mathcal{RWCC}$. Consider a $\mathcal{RWCC}$ $S_{<C>}$. We will call the elements of C_0 the *state labels*. By definition we know that there is a one to one correspondence between state labels and paths in $S_{<C>}$. Hence Hoare traces are faithfully represented by the state labels in a $\mathcal{RWCC}$. Moreover, a commutativity constraint of $\mathcal{RWCC}$ is an explicit description of homotopy relation. Namely, we have the following.

Lemma 2 *Let p_1 and p_2 be paths in a $\mathcal{RWCC}$ corresponding to some state labels. Then $p_1 \approx p_2$ iff the state labels are equal with the commutativity constraints of cubes.*

Now we recall the Mazurkiewicz traces.

Definition 11: [11] A *Mazurkiewicz trace language* consists of (M, L, I) where L is a set, $I \subseteq L \times L$ is a symmetric, irreflexive relation called the *independence* relation, and M is a nonempty subset of strings L^* such that

- *prefix closed*: $sa \in M \Rightarrow s \in M$ for all $s \in L^*, a \in L$,
- *I-closed*: $sabt \in M$ & $aIb \Rightarrow sbat \in M$ for all $s, t \in L^*, a, b \in L$,
- *coherent*: $sa \in M$ & $sb \in M$ & $aIb \Rightarrow sab \in M$ for all $s \in L^*, a, b \in L$.

For $s, t \in M$ define $\asymp$ to be the smallest equivalence relation such that $sabt \asymp sbat$ if aIb for $sabt, sbat \in M$. Call an equivalence class $\{s\}_{\asymp}$, for $s \in M$, a *Mazurkiewicz trace*.

In the Mazurkiewicz trace language, true concurrency is represented by the independence relation which means the commutativity of the action labels in Hoare traces (I-closedness). Hence our commutativity constraint is an independence relation and prefixed closedness of the Hoare traces is assured in $\mathcal{RWCC}$. A Mazurkiewicz trace is an equivalence class of a path via homotopy relation. If a $\mathcal{RWCC}$ does not contain a redundant commutativity constraint, namely a constraint $< a, b >_w$, for example, such that $wab \notin C_0$ or $wba \notin C_0$, then we know that coherency is assured. Notice that the independence relation aIb should be of local nature. Namely, if the actions a and b occur more than once in a whole computation, a and b may run independently in some part but they may depend on each other in other part. The locality is not explicitly represented in the above trace language. On the other hand, our commutativity constraints do not marely specify commutativity relation but have prefix as w in $< a, b >_w$. This prefix can represent the causality of the actions a and b. This enriched structure of commutativity constraints would solve the locality problem in the original Mazurkiewicz trace language.

3.2 CCS-like Process Constructors of $\mathcal{RWCC}$

We will define CCS-like processes in the domain of $\mathcal{RWCC}$. The process constructors will be defined with three basic operations, $\star$ (product), Θ (synchronization) and *Rest* (restriction). In this subsection, we will assume that all the commutativity constraints are those associated with cubes. In order to describe synchronous communication, we will set $\mathcal{A}$ as follows. Let $\mathcal{A} = \mathcal{L} \cup \overline{\mathcal{L}} \cup \{\tau_a \mid a \in \mathcal{L}\}$ where $\mathcal{L}$ is a denumerable set of symbols called labels, $\overline{\mathcal{L}} = \{\bar{a} \mid a \in \mathcal{L}\}$ is the set of co-labels such that $\bar{\bar{a}} = a$ for all $a \in \mathcal{L} \cup \overline{\mathcal{L}}$. We will understand that the labeled τ-action τ_a is the macro of the word $a\bar{a}$. Namely $\tau_a = a\bar{a}$.

Definition 12: Let $S_1 = (S_{<C_1>}, \partial^0, \partial^1, F_1, Const_1)$ and $S_2 = (S_{<C_2>}, \partial^0, \partial^1, F_2, Const_2)$ be $\mathcal{RWCC}$s. Then their product, $S_1 \star S_2$, is $(S_{<C_1 \star C_2>}, \partial^0, \partial^1, F, Const)$ where $C_1 \star C_2 = \{c_1 \star c_2 \mid c_i \in C_i \ (i = 1, 2)\}$, $F = F_1 \star F_2 = \{uw \mid u \in F_1, w \in F_2\}$ and $Const$ is the collection of the commutativity constraints for the cubes in $C_1 \star C_2$.

Before we define the synchronization map Θ, we define the function $\varphi(\{b_1,\ldots,b_n\}) = \{c_1,\ldots,c_m\}$ on finite sets of elements from $\mathcal{A}$. $\{c_1,\ldots,c_m\}$ is obtained from $\{b_1,\ldots,b_n\}$ as follows: If there is a pair b_i and b_j $(i<j)$ such that $\bar{b_i} = b_j$ then replace b_i by τ_{b_i} (if $b_i \in \mathcal{L}$) or $\tau_{\overline{b_j}}$ (if $b_i \in \overline{\mathcal{L}}$) and remove b_j. Carry out the same procedure for any such pairs in $\{b_1,\cdots,b_n\}$. $\Theta(S)$ generates new cubes with labels from $\{\tau_a \mid a \in \mathcal{L}\}$ by searching the label, co-label pairs in every cube from S with the function φ.

Definition 13: Let $S = (\bigoplus_{n\geq 0} S^n_{<C>}, \partial^0, \partial^1, F, Const)$ be a $\mathcal{RWCC}$. Then the *synchronization* of S, $\Theta(S) = (\bigoplus_{n\geq 0} S^n_{<D>}, \partial^0, \partial^1, F', Const)$ is the $\mathcal{RWCC}$ such that $F' = F \cap decomp(D)$ and

$$D = \bigcup_{\substack{c\in C \\ dim(c)\geq 2}} decomp(\Phi(c))$$

where

$$\Phi(w[a_1 \mid \cdots \mid a_n]) = \begin{cases} w[b_1 \mid \cdots \mid b_m] & \text{if } \varphi(\{a_1,\ldots,a_n\}) = \{b_1,\ldots,b_m\} \neq \{a_1,\ldots,a_n\} \\ 1 & \text{if } \varphi(\{a_1,\ldots,a_n\}) = \{a_1,\ldots,a_n\} \end{cases}$$

Example 2: Let $S = (S_{<C>}, \partial^0, \partial^1, F, Const)$ be a $\mathcal{RWCC}$ with $C = \{[a \mid \bar{a}], a\bar{a}[a \mid \bar{a} \mid b]\}$, $F = \{a\bar{a}a\bar{a}b\}$ and $Const = \{< a,\bar{a} >, < a,\bar{a},b >_{a\bar{a}}\}$. Then, $\Theta(S) = (S_{<D>}, \partial^0, \partial^1, F', Const)$ is a $\mathcal{RWCC}$ with $D = decomp(\{[\tau_a], a\bar{a}[\tau_a \mid b]\})$ and $F' = F$.

Next we define the restriction operation. $Rest(S, L)$ removes all the cubes involving the labels from $L \cup \overline{L}$.

Definition 14: Let $S = (\bigoplus_{n\geq 0} S^n_{<C>}, \partial^0, \partial^1, F, Const)$ be a $\mathcal{RWCC}$ and let L be a finite subset of $\mathcal{L}$. Then the *restriction* of S, $Rest(S, L) = (\bigoplus_{n\geq 0} S^n_{<D>}, \partial^0, \partial^1, F', Const)$ is a $\mathcal{RWCC}$ with $F' = F \cap decomp(D)$ and

$$D = \left\{ w[a_1 \mid \cdots \mid a_n] \in C \;\middle|\; \begin{array}{l} \{a_1,\ldots,a_n\} \cap (L \cup \overline{L}) = \phi, \text{ and if } \alpha \in L \cup \overline{L} \text{ occurs} \\ \text{in } w, \text{ then } \overline{\alpha} \text{ also occurs in } w \end{array} \right\}$$

Using there basic operations, we define CCS-like process constructors. We first introduce a few notations. Let $S, S_1, S_2 \subseteq \mathcal{M}$. Let $V(B)$ be the vector space over $\mathbf{Z}/(2)$ generated by a set of cubes B, and let $Const$ be a set of commutativity constraints. Then $S_1 \cdot S_2 = \{uv \mid u \in S_1, v \in S_2\}$, $S \cdot V(B) = V(S \cdot B)$ with $S \cdot B = \{wu[a_1 \mid \cdots \mid a_n] \mid w \in S, u[a_1 \mid \cdots \mid a_n] \in B\}$ and $S \cdot Const = \{< a_1,\ldots,a_n >_{wu} \mid w \in S, < a_1,\ldots,a_n >_u \in Const\}$.

Definition 15: A $\mathcal{RWCC}$ is called a *process* if it is constructed inductively as follows:

Atm: $\alpha \equiv ((1,\alpha,[\alpha]), \partial^0, \partial^1, \{\alpha\}, \phi)$ for $\alpha \in \mathcal{A}$,
Sum: $P + Q \equiv (S_p + S_q, \partial^0, \partial_1, F_p \cup F_q, Const_p \cup Const_q)$,
Par: $P \,||\, Q \equiv (S_1 + S_2, \partial^0, \partial^1, F_1, Const)$ where $(S_1, \partial^0, \partial^1, F_1, Const) = P \star Q$ and $(S_2, \partial^0, \partial^1, F_2, Const) = \Theta(P \star Q)$,
Seq: $P\,;\,Q \equiv (S_p + F_p \cdot S_q, \partial^0, \partial^1, F_p \cdot F_q, Const_p \cup F_p \cdot Const_q)$,
Res: $P\backslash L \equiv Rest(P, L)$, **Var:** $X \equiv ((1, X, [X]), \partial^0, \partial^1, \{X\}, \phi)$ and
Rec: $fix(X = P) \equiv ((1, X, [X]) + X \cdot S_p, \partial^0, \partial^1, F_p - \{X\}, \{X\} \cdot Const_p)$ with extra equations $\{X = XwX \mid Xw[X] \in X \cdot S_p\}$ where X is guarded and sequential in P

where $P = (S_p, \partial^0, \partial^1, F_p, Const_p)$ and $Q = (S_q, \partial^0, \partial^1, F_q, Const_q)$ are processes.

The idea of defining processes in $\mathcal{RWCC}$ is basically the same as the HDA interpretation of a CCS-like process algebra given in [3]. But we need to give a few words to our definition of recursive processes. We gave a finite representation of guarded and sequential recursive processes. The special cubes $w[\cdots \mid X \mid \cdots]$ (X is a process variable) with the extra equation $X = XwX$ only describe the loop structure and do not represent any actions. When we extend the language to that with guarded but not always sequential recursive processes, we need to introduce more complex relations with regard to X. We will give a few examples of our recursive processes in Figure 2.

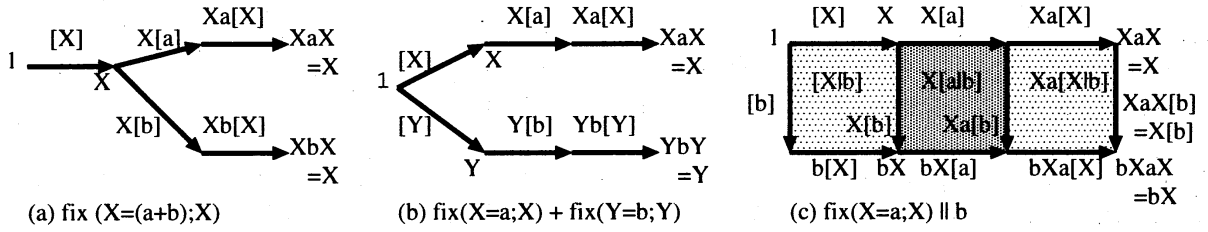


Figure 2: Recursive Processes

We know from the definition that all the commutativity constraints of a process are those associated with cubes in the $\mathcal{RWCC}$. Although each element of S in a $\mathcal{RWCC}$ $(S, \partial^0, \partial^1, F, Const)$ is a higher dimensional transition, we can define the transition in the traditional sense.

Definition 16: Let $P = S_{<C>}$ and $Q = S_{<D>}$ be $\mathcal{RWCC}$s and $\alpha \in \mathcal{A}$. We will say that $P \xrightarrow{\alpha} Q$ is a *lower dimensional transition* if one of the following holds:

Case 1: $[\alpha] \in C_1$ and $D(-\alpha)$

Case 2: There exists a path $([X], X[\alpha], X\alpha[X])$ in P and $D = C[X] \cup \{c \in C \mid c$ is in the form of $w[\cdots \mid X \mid \cdots]\}$

Case 3: There is a path $([X], X[\alpha])$ in P, $X\alpha[X] \notin C_1$ and $D = C(-X\alpha) \cup F(C(-X\alpha)) \cdot C(-X)$ where $C[w] = \{wu[a_1 \mid \cdots \mid a_l](\in C)\}$, $C(-w) = \{u[a_1 \mid \cdots \mid a_l] \mid wu[a_1 \mid \cdots \mid a_l] \in C\}$ and $F(S) = \{\partial^1(p_l) \mid (p_1, \ldots, p_l)$ is a maximal path in $S\}$.

Lemma 3 *The following transition rules hold for processes where $\alpha, \alpha_j \in \mathcal{A}$ and $a \in \mathcal{L}$:*

$$\alpha \xrightarrow{\alpha} \phi \qquad \frac{p \xrightarrow{\alpha} p'}{(p \parallel q) \xrightarrow{\alpha} (p' \parallel q)} \qquad \frac{q \xrightarrow{\alpha} q'}{(p \parallel q) \xrightarrow{\alpha} (p \parallel q')} \qquad \frac{p \xrightarrow{a} p' \quad q \xrightarrow{\bar{a}} q'}{(p \parallel q) \xrightarrow{\tau_a} (p' \parallel q')}$$

$$\frac{p_j \xrightarrow{\alpha_j} q_j}{\Sigma_i p_i \xrightarrow{\alpha_j} q_j} \qquad \frac{p \xrightarrow{\alpha} p'}{p \,;\, q \xrightarrow{\alpha} p' \,;\, q} \qquad \frac{p \xrightarrow{\alpha} p' \quad \alpha \notin L \cup \overline{L}}{p \backslash L \xrightarrow{\alpha} p' \backslash L} \qquad \frac{p[fix(X = p)/X] \xrightarrow{\alpha} p'}{fix(X = p) \xrightarrow{\alpha} p'}$$

4 Cycle Filling as Parallelization

In this section, we will consider how we can parallelize a given $\mathcal{RWCC}$ process with the idea of cycle filling as parallelization. Recall the example in Introduction. The first step is to find which part of the $\mathcal{RWCC}$ should be identified. This step will be called the *merging procedure*. The

merging procedure is essentially equivalent to finding additional prefixed independence relations as explained in Introduction. Next we find cycles to be filled in the $\mathcal{RWCC}$. The cycles in HDA are found by calculating homology groups. The final step is to fill the cycles with the cycle filling map. Because the cycle filling procedure may produce new cycles, the second and the third steps must be repeated until there is no cycle any more. We first explain homology groups. Then we define the cycle filling map. Finally we will give a parallelization algorithm.

4.1 Cycles and Homology Groups

A cycle is a subspace of $\mathcal{RWCC}$ with no boundary. The formal definition is as follows.

Definition 17: Let $S_{<C>}$ be a $\mathcal{RWCC}$. Let $Ker^{\times}\ \partial_n = \{D \in S^n_{<C>} \mid \partial_n(D) = 0, D$ does not contain cubes $w[a_1 \mid \cdots \mid a_n]$ such that $\{a_1, \ldots, a_n\}$ contains process variables.$\}$. Then $Ker^{\times}\ \partial_n$ is a sublinear space of $S^n_{<C>}$ and its elements will be called *n-cycles*.

The reason why we postulated the condition $w[\cdots \mid X \mid \cdots] \notin D$ is that we do not want to regard the loop representing recursion as a cycle to be filled.

Notice that an n-cycle does not always mean the *hole* in the space. For example, consider the $\mathcal{RWCC}$ $(S_{<\{[a|b]\}>}, \partial^0, \partial^1, \{ab\}, \{< a, b >\})$. $[a] + [b] + a[b] + b[a] (\in S^1_{<C>})$ is a 1-cycle, but it is at the same time the total boundary of the 2-cube $[a \mid b]$. Hence, the 1-cycle is already *filled*. If such a cube does not exist, the cycle means a real 2-dimensional hole. This idea leads us to the notion of homology groups [7].

Definition 18: Let $S = (\bigoplus_{n\geq 0} S^n_{<C>}, \partial^0, \partial^1, F, Const)$ be a $\mathcal{RWCC}$. Let $Im^{\times}\ \partial_{n+1} = \{\partial_{n+1}(D) \mid D \in S^{n+1}_{<C>}, D$ does notcontain cubes $w[a_1 \mid \cdots \mid a_n]$ such that $\{a_1, \ldots, a_n\}$ contains process variables.$\}$. Then, the residue class linear space $H_n(S) = Ker^{\times}\ \partial_n / Im^{\times}\ \partial_{n+1}$ over $\mathbf{Z}/(2)$ $(n \geq 1)$ will be called the *n-homology group* of S.

The n-cycles representing real $(n+1)$-dimensional holes are described by the elements of the n-homology group.

Example 3: Consider the $\mathcal{RWCC}$ $S = (S_{<C>}, \partial^0, \partial^1, F, Const)$ with $C = \{[a], [b], [c], b[a], c[a], bc[a], c[b], b[c], a[b \mid c]\}$ and $Const = \{< a, b >, < b, c >, < c, a >, < b, c >_a, < c, a >_b\}$. Figure 3 illustrates the geometric realization of $C' = decomp(C)$. $S_{<C'>} = \bigoplus_{n=0}^{3} S^n_{<C'>}$ is as follows:

$$
\begin{aligned}
S^0_{<C'>} &= (1, a, b, c, ab(=ba), ac(=ca), bc(=cb), abc(=acb=bac=bca=cab=cba)), \\
S^1_{<C'>} &= ([a], [b], [c], a[b], a[c], b[a], b[c], c[a], c[b], ab[c], ac[b], bc[a]) \text{ and } S^2_{<C'>} = (a[b \mid c])
\end{aligned}
$$

where $(g_1, \ldots, g_n)$ denotes the vector space over $\mathbf{Z}/(2)$ generated by the set $\{g_1, \ldots, g_n\}$. Then the homology groups are

$$
\begin{aligned}
H_1(S) &= \begin{pmatrix} [b]+[c]+b[c]+c[b], & b[a]+b[c]+ab[c]+bc[a], & [a]+[b]+a[b]+b[a], \\ [a]+[c]+a[c]+c[a], & c[a]+c[b]+ac[b]+bc[a] & \end{pmatrix} \\
H_2(S) &= 0
\end{aligned}
$$

Notice that the generators of $H_1(S)$ represent the obvious 2-dimensional holes in the space. Other elements of $H_1(S)$ such as $([b]+[c]+b[c]+c[b]) + (b[a]+b[c]+ab[c]+bc[a]) = [b]+[c]+c[b]+b[a]+ab[c]+bc[a]$ also represent 2-dimensional holes but they are obtained by combination

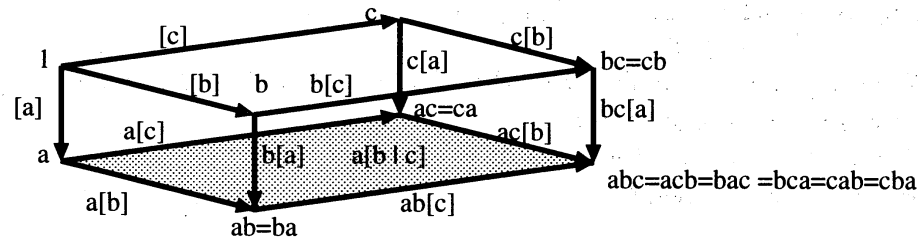


Figure 3: Geometric Realization of $decomp(C)$

of the holes represented by the generators. Also there is no 3-dimensional hole in the space. This is indicated by the vanishing of $H_2(S)$.

Of course, the generators of a linear space are not uniquely determined and we will call the generators such as in the above example the *minimal generators*. A minimal generator is the boundary of a single cube. The precise definition is as follows.

Definition 19: The set of generators $\{g_1, \ldots, g_k\}$ of the homology group $H_n(S)$ is called *minimal* if each element g_i is the formal sum of length $2(n+1)$.

Lemma 4 *For a $\mathcal{RWCC}$ S, the set of minimal generators of the n-homology group $H_n(S)$ always exists uniquely.*

In the following, when we refer generators of a homology group, we will always assume the minimal ones. The generators of a homology group can be computed by a rather easy algorithm, such as Gauss method, in linear algebra as demonstrated in [2].

4.2 Cycle Filling Map

Now we will define the cycle filling map s_n of $\mathcal{RWCC}$. Consider a $\mathcal{RWCC}$ $S_{<C>}$. Let $c_n \in Ker^* \partial_n$ be the total boundary of an $(n+1)$-cube $c_{n+1} \in C_{n+1}$: $c_n = \partial_{n+1}(c_{n+1})$, then s_n will reconstruct c_{n+1} from c_n: $s_n(c_n) = c_{n+1}$.

Definition 20: Let $S_{<C>}$ be a $\mathcal{RWCC}$. Then, the *cycle filling map* $s_n : S^n_{<C>} \longrightarrow S^{n+1}_{<C>}$ $(n \geq 1)$ is a function such that, for arbitrary $M = \Sigma_i w_i [a_1^{(i)} \mid a_2^{(i)} \mid \cdots \mid a_n^{(i)}] (\in S^n_{<C>})$, $s_n(M)$ is computed as follows:

Step 1: Let $M = wN$ where the word w is the longest word such that for all i, $w_i = wu_i$ for some word u_i.

Step 2: Calculate $\varphi_n(N)$ where φ_n is the linear map $S^n_{<C>} \longrightarrow V(\mathcal{A} \times Pow(\mathcal{A}))$ such that $\varphi_n(u[a_1 \mid \cdots \mid a_n]) = < u, \{a_1, \ldots, a_n\} >$.

Step 3: If there is a set $\{b_1, \ldots, b_{n+1}\} (\in Pow(\mathcal{A}))$ such that $\varphi_n(N)$ is in the form of $\varphi_n(N) = \Sigma_{i=1}^{n+1} < 1, P_i > + \Sigma_{i=1}^{n+1} < b_i, P_i >$, where $P_i = \{b_1, \ldots, \overset{i}{\vee}, \ldots, b_{n+1}\}$, then $s_n(M) = w[b_1 \mid \cdots \mid b_{n+1}]$ otherwise $s_n(M) = 0$.

Notice that the cycle filling map s_n checks whether M is a minimal generator of the n-homology group $H_n(C)$ and construct the suitable $(n+1)$-cube to fill the cycle. Hence we can define the cycle filling algorithm as follows.

Cycle Filling Algorithm: Given a $\mathcal{RWCC}$ $S = S_{<C>}$, then

1. If $H_n(S) = 0$ for all $n \geq 1$, then goto 4.
2. Let N be the smallest number such that $H_N(S) \neq 0$ and let $NewCubes = \{s_N(C) \mid C \in H_N(S)$ is a (minimal) generator$\}$.
3. Let $C := C \cup NewCubes$, then goto 1.
4. Return the obtained $\mathcal{RWCC}$.

Notice that the cycle filling map does not increase the 1-cubes nor the commutativity constraints in the given $\mathcal{RWCC}$. From this fact, we obtain the following.

Lemma 5 *For arbitrary $\mathcal{RWCC}$ $S_{<C>}$ such that C is a finite set, the cycle filling algorithm terminates in finite steps and the obtained data is again a $\mathcal{RWCC}$.*

Example 4: We will reconsider the $\mathcal{RWCC}$ S in Example 3. By applying the cycle filling map s_1 to the generators of $H_1(S)$ we obtain the new $\mathcal{RWCC}$ $S' = S_{<C'>}$ where $C' = C \cup \{[a \mid b], [b \mid c], [a \mid c], b[a \mid c], c[a \mid b]\}$. Then $H_1(S') = 0$ and $H_2(S') = ([a \mid b] + [b \mid c] + [a \mid c] + b[a \mid c] + c[a \mid b] + a[b \mid c])$. Thus S' has a 2-cycle. Now we apply s_2 to the generator of $H_2(S')$ to obtain the 3-cube $[a \mid b \mid c]$. The constructed $\mathcal{RWCC}$ $S'' = S_{<C''>}$ is such that $C'' = decomp(\{[a \mid b \mid c]\})$. Now $H_n(S_{<C''>}) = 0$ for all $n \geq 1$. Then the algorithm terminates by returning the $\mathcal{RWCC}$ S''.

4.3 Parallelization Algorithm with Cycle Filling Map

Now we will formulate the parallelization problem in the language of $\mathcal{RWCC}$ according to the general idea given by E. Goubault [2]. First of all, we define the degree of parallelism.

Definition 21: Let $P = \bigoplus_{n=0}^{N} S^n_{<C>}$ be a $\mathcal{RWCC}$. Then $par(P) \equiv N$ is called the *degree of parallelism* of P.

Lemma 6 *For arbitrary processes P and Q, we have*

1. $par(\alpha) = 1$ *if α is an* **Atm**,
2. $par(P + Q) = par(P \,;\, Q) = max\{par(P), par(Q)\}$.
3. $par(P \,||\, Q) = par(P) \times par(Q)$, *and*
4. $par(fix(X = P)) = par(P)$.

This lemma claims that the definition of par is an acceptable definition degree of parallelism of processes. Although there seems to be no syntactical definition of degree of parallelism for label restriction $P\backslash L$, we can define $par(P\backslash L)$ at the semantics level.

Definition 22: Let S_1 and S_2 be $\mathcal{RWCC}$. If there are maps $\varphi_i : Base(S_1, i) \to Base(S_2, i)$ $(0 \leq i \leq par(S_1))$ such that (i) $\varphi_i(c) = c$ for all $c \in Base(S_1, i)$, (ii) φ_0 induces a surjection between maximal Hoare traces of S_1 and S_2, and (iii) φ_0 and φ_1 are surjections, then we will say that there is an *embedding* $\varphi = \bigoplus_i \varphi_i : S_1 \to S_2$.

Example 5: Let P and Q be as in Figure 4. We have surjections φ_0 and φ_1 such that $\varphi_0(1) = 1$, $\varphi_0(a) = a$, $\varphi_0(b) = b$, $\varphi_0(ab) = ab = ba = \varphi_0(ba)$, $\varphi_0(abc) = abc = bac = \varphi_0(bac)$, $\varphi_1([a]) = [a]$, $\varphi_1([b]) = [b]$, $\varphi_1(a[b]) = a[b]$, $\varphi_1(b[a]) = b[a]$ and $\varphi_1(ab[c]) = ab[c] = ba[c] = \varphi_1(ba[c])$. The sets of maximal Hoare traces in P and Q are $\{abc, bac\}$ and $\{abc(= bac)\}$, so that φ_0 induces a surjection between them.

We will say that a $\mathcal{RWCC}$ S_2 is a *parallelization* of a $\mathcal{RWCC}$ S_1 if there is an embedding $\varphi : S_1 \to S_2$ and $par(S_1) \leq par(S_2)$. Now we will give an parallelization algorithm with the

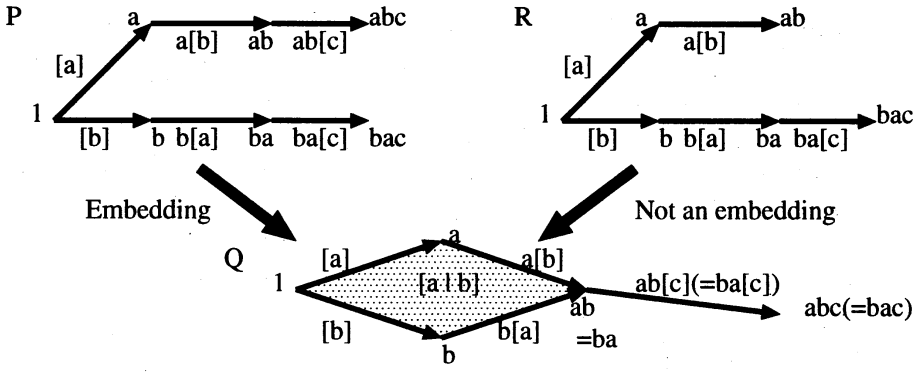


Figure 4: Embedding of Processes

cycle filling map. Recall the example of parallelization in Introduction. The first step is the merging procedure: to find which part of the given $\mathcal{RWCC}$ should be identified. Such part of the space is determined with the merging candidate defined below.

Definition 23: Let S be a $\mathcal{RWCC}$. Then a pair of state labels in the form of $\ll w\alpha\beta, w\beta\alpha \gg$ $(w \in \mathcal{M}, \alpha, \beta \in \mathcal{L} \cup \overline{\mathcal{L}})$ is called the *merging candidate* of S if $C_i(-w\alpha\beta) = C_i(-w\beta\alpha)$ $(i = 0, 1)$ where $C_i(-w)$ is as in Definition 16.

Notice that the cycles are generated only by equating the merging candidates because we have the following.

Lemma 7 *Any process P is cycle free. Namely, $H_n(P) = 0$ for all $n \geq 1$.*

A merging candidate is a pair of state labels to be equated in the merging procedure. If the state labels in the merging candidate are equated, some of other cubes in the $\mathcal{RWCC}$ are also equated with the right extension rule. The reason why we postulate the condition $C_i(-w\alpha\beta) = C_i(-w\beta\alpha)$ is as follows. We cannot equate all the pair of elements in the form of $\ll w\alpha\beta, w\beta\alpha \gg$. For example, consider a process $R = a\ ;\ b + b\ ;\ a\ ;\ c$ (a, b, c are **Atms**). Then $Base(R, 0) = \{1, a, b, ab, ba, bac\}$ so that we have a pair $\ll ab, ba \gg$ in $Base(R, 0)$. If we postulate $ab = ba$ and fill the cycle, we obtain the $\mathcal{RWCC}$ $Q = (S_{<C>}, \partial^0, \partial^1, \{abc\}, \{< a, b >\})$ with $C = \{[a \mid b], ab[c]\}$ but there is no embedding of R into Q (see Figure 4).

Now we can define our parallelization algorithm.

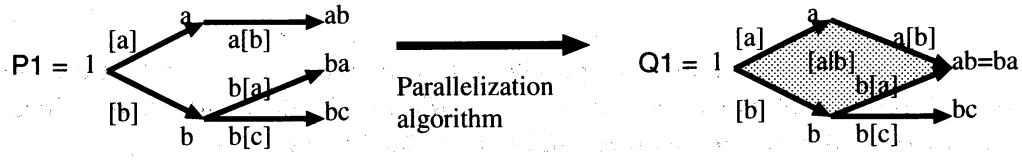
Parallelization Algorithm: Given a $\mathcal{RWCC}$ $S = (S_{<C>}, \partial^0, \partial^1, F, Const)$, then

1. Find the set of merging candidates MCP from C_0
2. Make the set $Const_0 = \{< \alpha, \beta >_w \mid \ll w\alpha\beta, w\beta\alpha \gg \in MCP\}$
3. Replace $Const$ by $Const \cup Const_0$
4. Carry out the cycle filling algorithm on S with the new $Const$.

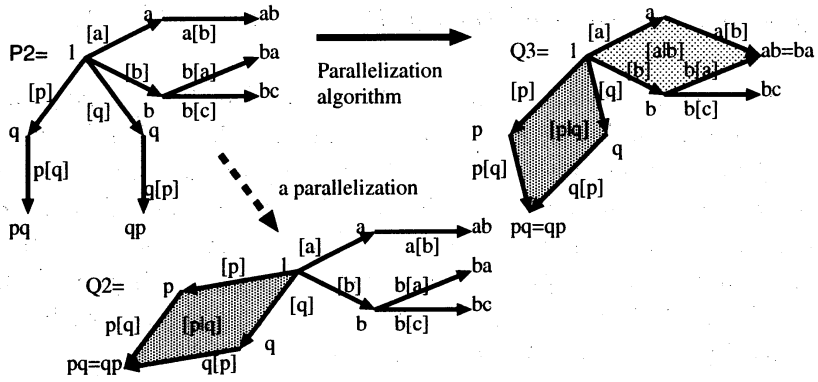
Example 6: Consider the process $P = a\ ;\ (b \parallel c) + b\ ;\ (a\ ;\ c + c\ ;\ a) + c\ ;\ (a\ ;\ b + b\ ;\ a)$ given in Introduction. It is realized as the $\mathcal{RWCC}$ $P = (S_{<C>}, \partial^0, \partial^1, \{abc(= acb), bac, bca, cab, cba\}, \{< b, c >_a\})$ where $C_0 = \{1, a, b, c, ab, ba, bc, cb, ca, ac, abc, acb, bac, bca, cab, cba\}$, $C_1 = \{[a], [b], [c], a[b], a[c], b[a], b[c], c[a], c[b], ab[c], ac[b], ba[c], bc[a], ca[b], cb[a]\}$, $C_2 = \{a[a \mid b]\}$. We find the set of merging candidate $\{\ll ab, ba \gg, \ll bc, cb \gg, \ll ac, ca \gg, \ll cab, cba \gg, \ll abc, acb \gg, \ll$

$bac, bca \gg\}$ from which we obtain the additional commutativity constraints $\{< a, b >, < b, c >, < a, c >, < a, b >_c, < b, c >_a, < a, c >_b\}$. Notice that with the right extension rule and other rules we know other equations $abc = acb = bac = bca = cab = cba$, $ab[c] = ba[c], ac[b] = ca[b], bc[a] = cb[a]$. Then the obtained $\mathcal{RWCC}$ is in fact equal to the $\mathcal{RWCC}$ in Example 3, and the cycle filling algorithm is carried out as in Example 4. Then the obtained $\mathcal{RWCC}$ is in fact the process $Q = (a \parallel b \parallel c)$.

Notice that our parallelization algorithm does not always generate a $\mathcal{RWCC}$ in the form of process. For example, by applying the algorithm to $P_1 = a\ ;\ b + b\ ;\ (a + c)$, we obtain the $\mathcal{RWCC}$ $Q_1 = (S_{<C>}, \partial^0, \partial^1, \{ab, ba, bc\}, \{< a, b >\})$ with $C = \{[a \mid b], b[c]\}$. This does not denote any process.



Also, if a process Q is a parallelization of a process P, our parallelization algorithm cannot always construct Q. For example, consider a process $P_2 = p\ ;\ q + q\ ;\ p + a\ ;\ b + b\ ;\ (a + c)$. Then $Q_2 = (p\ ;\ q) + a\ ;\ b + b\ ;\ (a + c)$ is a parallelization of P_2, but our parallelization algorithm construct $Q_3 = (S_{<D>}, \partial^0, \partial^1, \{ab, ba, bc, pq, qp\}, \{< a, b >, < p, q >\})$ with $D = \{[p \mid q], [a \mid b], b[c]\}$ and this is not a process. This indicates that the merging procedure should be carried our with more subtle information on the given process.



Despite these problems, we have the following.

Theorem 1 *(1) For any process P, if the parallelization algorithm outputs a process Q then Q is a parallelization of P, (2) If the parallelization algorithm does not change the input process P, then there is no non-trivial parallelization of P.*

Whether the parallelized $\mathcal{RWCC}$ is a process or not can be checked by the combinatorial geometric analysis called reverse interpretation [9].

5 Discussion and Conclusion

We presented a new formulation of Pratt-Glabbeek-Goubault-Jensen style Higher-Dimensional Automata (HDA) model. The idea is to give an intrinsic definition of hypercubes with cubical Bar-notation. The obtained formulation is called reachable well-labeled cubical complex ($\mathcal{RWCC}$) and it makes clear relation between the general idea of HDA and traditional semantics theories such as Hoare traces and Mazurkiewicz traces. Using the language of $\mathcal{RWCC}$, we tried to solve the parallelization problem of CCS-like process algebra, which is a refinement of the works by E. Goubault [2] and the author [8]. The cycle filling map can be formally defined in $\mathcal{RWCC}$ and we gave a parallelization algorithm using the map. As explained in section 4, our algorithm does not always construct parallelized processes from given processes. E. Goubault [2] seems to indicate the basic idea to solve this problem. Namely, a parallelized CCS-like process is generally a part of a more parallel process in an extended process calculi. This idea has not been thoroughly investigated in our theory. Also, we did not consider bisimulation issues extensively in this paper. More technical detail must be presented to discuss this issue and it will be reported in other occasion.

References

[1] E. Goubault. Domains of Higher-Dimensional Automata. In *4th International Conference on Concurrency Theory, LNCS 715.* Springer Verlag, 1993.

[2] E. Goubault. Schedulers as Abstract Interpretations of Higher-Dimensional Automata. In *Proc. of PEPM'95.* ACM Press, 1995.

[3] E. Goubault and T. P. Jensen. Homology of Higher Dimensional Automata. In *3rd International Conference on Concurrency Theory, LNCS 630.* Springer Verlag, 1992.

[4] J. F. Groote and R. Moller. Verification of Parallel Systems via Decomposition. In *3rd International Conference on Concurrency Theory, LNCS 630.* Springer Verlag, 1992.

[5] A. Mazurkiewicz. Trace Theory. In *Petri Nets: Applications and Relationships to Other Models of Concurrency, LNCS 255.* Springer Verlag, 1986.

[6] V. Pratt. Modeling Concurrency with Geometry. In *18th Annual ACM Symposium on Principles of Programming Languages.* ACM, 1991.

[7] E. H. Spanier. *Algebraic Topology.* McGraw-Hill, 1966.

[8] Y. Takayama. Parallelization of Concurrent Processes in Higher Dimensional Automata. In *RIMS Workshop on Term Rewriting Systems and its Application.* RIMS Kyoto University, 1995.

[9] Y. Takayama. Extraction of Concurrent Processes from Higher Dimensional Automata. In *CAAP'96, LNCS 1059.* Springer Verlag, 1996.

[10] R. van Glabeek. Bisimulation semantics for higher dimensional automata. Technical Report, Stanford University, 1991.

[11] G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science, Vol.4.* Oxford Science Publications, 1995.