Algebra and dynamics of manufacturing-like processes

Enrico Canuto, Sergio Tonani
Dipartimento di Automatica e Informatica, Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
enrico.canuto@polito.it

Abstract

A mathematical framework which has been specifically developed for manufacturing-like systems is presented, focusing on time-free dynamics and control, as treated by Finite State Automata. The approach is based on the key concepts of object types, locations and manufacturing operations enriched with an algebra (Manufacturing Algebra) for building complex operations as algebraic series-parallel compositions. The same framework has been already extended to treat discrete-event dynamics forced by timed-events described as [fact, time], a formulation not employed hereafter.

1 Introduction

Several methods have been devised and are currently used for modeling dynamics and designing control strategies of manufacturing-like systems, where discrete parts (objects) are progressively transformed and assembled by a set of (manufacturing) operations up to yield a sequence of products fulfilling criteria and demands. Think of well known Petri Nets, Finite State Automata, Queueing Networks, ... Hereafter a mathematical framework which has been specifically developed for manufacturing-like systems is presented, focusing on time-free dynamics and control, as treated by Finite State Automata. The approach is based on the key concepts [2] of object types, locations and manufacturing operations (MOs) enriched with an algebra (Manufacturing Algebra, MA) for building complex MOs as algebraic series-parallel compositions. The same framework has been already extended to treat discrete-event dynamics forced by timed-events described as [fact, time] [3], a formulation not employed hereafter.

After Section 2 devoted to key concepts: object types and locations and their quantities, MOs and their representations are defined in Section 3. Two algebraic operations, series and parallel, for composing them into complex sequences of MOs are then introduced, laying down Manufacturing Algebra. Using MA we first derive state equations (Section 4) describing the time-free dynamics of arbitrary manufacturing processes, forced by a sequence of MO actuations as external events. State variables are object quantities stored in locations (stocks) and object quantities going out of the whole process (throughput). Then using MA, we shall formulate any sequence of actuated MOs as a single complex MO, thus favoring aggregation of state equations. Aggregate dynamics will thus be forced by a sequence of aggregate events, becoming the new command sequence to be designed in order to fulfill production demand. Some control problems pertaining to aggregate dynamics will be presented and solved.

2 Object types, locations and quantities

We are considering a production or computational process, called for historical reasons manufacturing process receiving as input an ordered or partially ordered set of objects, which are transformed for producing as output other objects. Part of the input objects are employed during the process and then are cycled back at the end of the process. Part of the output objects, the so-called finished products goes out of the process being the process output. Along the process, before and after some transformation, objects can be located in some buffers called locations. The set of all objects is assumed to be countable, but they can be grouped into a finite number of types and located in a finite number of locations.

Object types. Given a countable set O, the universe of the (manufacturing) objects α, we assume that different equivalence relations may exist between objects, grouping them into equivalence classes. The most important one is the type-equivalence Rtype partitioning the universe into a finite set of disjoint object classes O_k, k = 1, ..., K called (object) types. Each class O_k can be seen as an element of a finite set (or alphabet) Κ = O_k/Rtype = {κ_1, ..., κ_k, ..., κ_K}, which is given a lexicographic order. Object types will be often denoted just by their index k.

Object locations. A second relation is the location-equivalence Rloc partitioning the universe into a finite set of disjoint object classes O_l, l = 1, ..., L called locations. Each location can be seen as an element of an alphabet L = O_l/Rloc = {λ_1, ..., λ_l, ..., λ_L}, which is given a lexicographic order. Object locations will be often denoted just by their index l.
3 Manufacturing operations

3.1 Definition and representations

(Manufacturing) objects can be transformed into other objects through a sequence of stages called manufacturing operations.

**Definition 1 - Manufacturing operation (shortly MO).**

An MO $A : Q^{KL} \rightarrow Q^{KL}$ is a mapping $y = A(u)$ transforming an input vector $u$ into an output vector $y$. ■

An MO can be represented by a graph having two types of nodes: (i) MOs are represented by a square or rectangle (ii) pairs (object type $k$, location $l$) are represented by a circle. The nodes are connected by directed links (arrows) representing for each pair $(k, l)$ one of the input quantity $u(k, l)$ employed by MO $A$ and drawn from location $l$ or the output quantity $y(k, l)$ yielded by MO $A$ and delivered to location $l$. Zero quantities are not represented. Input and output quantities are written besides each arrow. In case of a single location, $L = 1$, only the object type index $k$ is shown. Figure 1 shows a single MO $A$ drawing input objects from locations $l = 1, 2$ and delivering output objects to locations $l = 1, 3$. The MO can be interpreted as assembling two types $k = 1, 2$ into a third type $k = 3$ with the help of a fourth type $k = 4$ which can be again employed.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{Figure1.png}
\caption{Symbol of a manufacturing operation}
\end{figure}

Manufacturing operations can be classified into representations (i) the input-output one (IO), (ii) the balance vector one.

**Definition 2 - Input-output representation (IOR).**

A (manufacturing) operation $A : y = A(u)$ is represented by the ordered pair $a = (u, y)$ of the input and output vectors. We shall use capital letters like $A$ for MOs and a small letter like $a$ for their IOR. The Cartesian product $A = Q^{KL} \times Q^{KL}$ is called the IOR space. ■

**Definition 3 - Balance vector.** The balance vector of a (manufacturing) operation $A : y = A(u)$ is defined by $b = y - u$. ■

**Remark.** The balance vector specifies the order between input and output objects through the sign of
their quantities, positive for output and negative for input ones, but some information is lost with respect to IO representation. For instance if an object type \((k, l)\) is released at the same location \(l\) with the same input quantity \(y(k, l) = u(k, l)\), it disappears from the balance vector, i.e. \(b(k, l) = 0\).

Consider an alphabet \(\mathcal{F} = \{A, B, C, \ldots\}\) of MOs defined over a quantity space \(Q\) of size \(KL\). Its elements will be called elementary MOs. Let \(A\) and \(B\) be generic elements of \(\mathcal{F}\) and denote with \(a = (u_a, y_a)\) and \(b = (u_b, y_b)\) their IORs.

**Lemma 2** - Type partition. Any alphabet \(\mathcal{F}\) of MOs induces a unique partition of the type set \(K\) into four disjoint subsets, 
\[K = K_0 \cup K_r \cup K_a \cup K_f\] such as \(K_0 \cap K_r \cap K_a \cap K_f = \emptyset\). The same partition applies to any quantity vector \(q\) defined over \(K\).

**Proof.** Consider the following subsets: (i) the idle type subset \(K_0 = \{k\}\) such that \(u_a(k) = y_a(k) = 0\) for all \(k \in \mathcal{F}\); (ii) the raw material subset \(K_r = \{k\}\) such that \(y_a(k) = 0\ \forall A \in \mathcal{F}\), but there exists at least a \(B\) such that \(u_a(k) \neq 0\); (iii) the finished product subset \(K_f = \{k\}\) such that \(u_a(k) = 0\ \forall A \in \mathcal{F}\), but there exists at least a \(B\) such that \(y_a(k) \neq 0\). All three subsets are unique and disjoint by construction. The fourth disjoint subset \(K_s = \{k\}\) is \(K_0 \cup K_r \cup K_a \cup K_f\), the semi-finished subset, is such that there exists at least a pair \(A\) and \(B\) possibly equal, such that \(y_a(k) \neq 0\) and \(u_a(k) \neq 0\). ■

### 3.2 Manufacturing algebra

Consider an alphabet \(\mathcal{F} = \{A, B, C, \ldots\}\) of elementary MOs. We want to describe complex (manufacturing) processes as algebraic compositions of expressing sequential order (series) or parallel execution of them. Serial composition will be usually needed when MOs employ object types produced by other MOs. One of the main result of the Manufacturing Algebra is that algebraic compositions can be still treated as complex MOs, thus favouring aggregation processes. Complex MOs built over an alphabet \(\mathcal{F}\) will be denoted by \(\mathcal{C}(\mathcal{F})\) and their IO representation \(c(\mathcal{F})\) will be easily obtained by applying the following rules.

**Axioms.** (i) A pair of MOs \((A, B)\) can be composed either in series, written \(AB\), or in parallel, written \(A + B\). (ii) Series is associative, i.e. \(A(BC) = (AB)C\), but not commutative, i.e. \(AB \neq BA\), therefore strictly ordering \(A\) and \(B\). Specifically \(A\) precedes \(B\) in \(AB\), because language ordering from left to right will be adopted. (iii) Parallel is commutative and associative, i.e. \(A + B = B + A\) and \(A + (B + C) = (A + B) + C\), meaning that no order is imposed. (iv) Series is not distributive with respect to parallel, i.e. \(A(B + C) \neq (AB) + (AC)\), because any repeat of the same MO in a process like in the right hand side is effective. ■

We will now show that the above axioms are satisfied by a natural definition of series and parallel in the space \(\mathcal{A}\) of IO representations. They also create an algebra within \(\mathcal{A}\). In the following we shall confuse MOs with their IOR.

**Definition 4 - Series.** Given a pair of MOs \(a = (u_a, y_a) \in \mathcal{A}\) and \(b = (u_b, y_b) \in \mathcal{A}\), their series \(c = ab = a \cdot b\) is defined by the equalities:

\[u_c = u_a + u_b - u_a \cap y_a, \quad y_c = y_a + y_b - u_b \cap y_b.\] (2)

The former MO, \(a\), is called predecessor and the latter MO, \(b\), successor. The \(\beta\)-times series of the same MO \(a\) will be denoted with the power symbol \(a^\beta, \beta \in \mathbb{Z}\). ■

**Remark.** In a serial composition the successor must employ as input objects the output objects of the same type and quantity yielded by the predecessor. The quantity vector \(c_{ab}\) of such objects, called compensation, equals the quantity \(c_{ab} = u_a \cap y_a\) to be dropped from the the IO vectors of the series.

**Theorem 1 - The series composition is not commutative but associative.**

**Proof.** (i) **Series is not commutative**: the series \(ab\) and \(ba\) would be equal if and only if \(c_{ab} = c_{ba}\). But the latter equality would imply the equality \(u_a \cap y_a = u_b \cap y_b\), which can not hold in general. (ii) **Series is associative**: \((ab)c = a(bc)\) holds if and only if their compensations are equal, i.e. \(c_{a(bc)} = c_{(ab)c}\). They can be easily computed as:

\[c_{a(bc)} = y_b \cap u_a + y_c \cap (u_a + u_b - y_b \cap u_a),\]
\[c_{(ab)c} = u_b + u_a \cap (y_c + y_b - y_c \cap u_b).\] (3)

By applying the distributive property of vector addition with respect to vector intersection (Lemma 1) one obtains:

\[c_{a(bc)} = c_{(ab)c} = (u_a + u_b) \cap (y_b + y_c) \cap (u_a + y_c),\] (4)

proving that series is associative ■

The series plays the role of multiplication and defines for each \(a \in \mathcal{A}\) a left and right identity (unit element). The unit element being not unique, \(\mathcal{A}\) is a semi-group with respect to multiplication.

Parallel composition plays the role of addition.

**Definition 5 - Parallel composition.** Given a pair of MOs \(a = (u_a, y_a) \in \mathcal{A}\) and \(b = (u_b, y_b) \in \mathcal{A}\), their parallel \(c = a + b\) is defined by the equalities:

\[u_c = u_a + u_b, y_c = y_a + y_b.\] (5)

Parallel composition is commutative and associative. Moreover it defines the null MO, \(o = (0, 0)\), and the
negative MO of any \( a, a + (-a) = 0 \). Therefore \( \mathcal{A} \) is an Abelian group with respect to +. The \( \beta \)-times parallel of the same MO \( a \) will be denoted using the scalar multiplication symbol \( \beta a, \beta \in \mathbb{Z} \).

The last axiom to be proved is that the series is not distributive with respect to parallel.

**Theorem 2** - Series is not distributive with respect to parallel.

*Proof.* It is sufficient to show a case where distributivity fails. Consider three MOs of \( \mathcal{A} \), namely \( a = (u_1, y_a) \), \( b = (u_2, y_b) \) and \( c = (u_3, y_c) \) and assume for simplicity that \( u_b \cap y_a = u_b \cap y_a = 0 \), i.e. no compensation exists, and that \( y_a \neq 0 \). Then the output vector of \( a(b + c) \) holds \( y_a(b + c) = y_a + y_b + y_c \), while the output vector of \( ab + ac \) holds \( y_{ab + ac} = 2y_a + y_b + y_c \neq y_a(b + c) \), proving the theorem. ■

As already mentioned, the order between object types described by IORs is partly lost by the balance vector representation. The following lemma, given without proof, confirms the assertion.

**Lemma 3** - The balance vectors of the series \( ab \) and \( ba \) and of the parallel \( a + b \) are equal. ■

**Example.** To conclude consider the alphabet \( \mathcal{F} = \{ A_k \}, \) sized \( H = 8 \), shown graphically in Figure 2. The alphabet is built over an object universe with a size \( K \), (ii) the location alphabet \( \mathcal{L} \) of size \( L \), (iii) the alphabet \( \mathcal{F} = \{ A_1, \ldots, A_H \} \) of \( H \) elementary MOs, together with their IORs \( \mathcal{A}_k = \langle u_k, y_k \rangle \). The triple \( \{ \mathcal{K}, \mathcal{L}, \mathcal{F} \} \) will be called a process. The input/output vectors of the process can be collected in a pair of matrices, \( B_a \) and \( B_b \), called input and output matrices:

\[
B_a = |u_1 \ldots u_H|, B_b = |y_1 \ldots y_H| \quad (7)
\]

**Dynamics.** We shall describe a time-free dynamics \(^1\) as a sequence of stages denoted by \( i \geq 0 \) during which one or more MOs, working in parallel, draw input objects from their locations and deliver output objects the next stage. At a stage \( i \), MOs are actuated by an integer command vector \( s(i) \geq 0 \) listing the number of time \( s_k(i) \) an elementary MO \( A_k \) can be actuated in parallel. The command sequence \( \{s(i)\} \) plays the role of a sequence of external events forcing dynamics. It can be shown that any algebraic composition \( C(F) \) can be converted into a command subsequence. For instance the composition of equation (6) translates into (zero entries are written as dots):

\[
|s(0) \ldots s(3)| = \begin{bmatrix}
 2 & 3 & \ldots & \\
 3 & 2 & \ldots & \\
 \vdots & \vdots & \ddots & \\
 1 & 1 & \ldots & 1
\end{bmatrix} \quad (8)
\]

**State and input sequences.** Dynamics is uniquely defined by state and input sequences. (i) The state is a two-block vector including the vector \( x(i) \) listing the object quantities at each location (stock) and the vector \( z(i) \) listing the deliveries to each pair \( (k, l) \) (throughput). (ii) The input is a three-block vector including the MO command vector \( s(i) \) and the so-called input and output flows, i.e. the quantity vectors \( r(i) \) and \( p(i) \) listing raw materials delivered to and finished products drawn from locations. Input and output flows are usually treated as external disturbances. State and input vectors are constrained to be integer.

**State equations.** To write state equations we use the input and output matrices \( B_a \) and \( B_b \) of the MO alphabet:

\[
\begin{align*}
\begin{bmatrix} x(i + 1) \\ z(i + 1) \end{bmatrix} &= \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(i) \\ z(i) \end{bmatrix} + \begin{bmatrix} -B_a & I - I \\ B_b & 0 \\ 0 \end{bmatrix} \begin{bmatrix} s(i) \\ r(i) \\ p(i) \end{bmatrix} \\
\begin{bmatrix} x(0) \\ z(0) \end{bmatrix} &= \begin{bmatrix} x_0 \\ z_0 \end{bmatrix}, & \begin{bmatrix} \frac{x(i)}{z(i)} \end{bmatrix} &\geq 0, Lx(i) \leq \bar{I}(i), & (9)
\end{align*}
\]

\(^1\) It means that each stage \( i \) is neither explicitly associated to a time instant \( t \in \mathbb{R} \), nor absolute and relative time (delays) will enter the state equations as parameters. The term logical is often used with a similar meaning. The term dynamics means that a state equation exists.

---

2532
The quantities \( x(i, k, l) \) of type \( k \) stored at location \( l \) may be bounded independently or share a single bound. Both cases can be expressed by matrix \( L \). Also the command vector may be bounded.

**State equation eigenvalues.** The eigenvalues of the state equation (9) are either unit or zero. Therefore the state equation is unstable.

**Control problems.** Control problems are related to controllability properties and amount to find command sequences \( \{s(i)\} \) satisfying criteria like limitation of location stock and fulfillment of a finished product demand, \( x(i) + z(i) \geq y(i) \). Such problems will be not treated hereafter, except for the following example.

**Example.** Consider the process of Figure 2. By assuming that the input flow is controllable, it can be shown that (i) the input flow \( r(i) = Fs(i) \), \( F^T = [B_{u,r}] \), \( B_{u,r} \) is the input matrix restricted to raw materials, and (ii) the command sub-sequence defined in equation (8) keep untouched the initial stock \( x(0) \) of semifinished types, except for \( k = 4 \) and can fulfill a finished product demand \( p(i) = 0, \ i = 0, 1, 2, 3 \) and

\[
p^T(4) = \begin{bmatrix} 1 & 2 \end{bmatrix}.
\]

Note that \( k = 4 \) is a peculiar type, called reusable, which differently than other semifinished, is released by the process and can be employed again (think of tools, processing units, ...)

**Remark.** No mention has been done of processing units describing facilities where MOs can be actuated. In time-free setting they are just described by reusable types which are employed and immediately released by MOs. Of course the same unit can be employed by different MOs thus imposing them serial processing. On the other hand the same unit type might be available as a multiple object thus facilitating parallel processing. The share of the same reusable by different MOs can give rise to deadlock and conflict conditions which can stop or delay the process. Related problems are not treated here. Note however that object \( k = 4 \) in Figure 2 is a typical reusable object.

**4.1 Aggregate dynamics**

Different kinds of aggregation can be applied to a state equation like (9): (i) aggregation of object types and locations reducing state dimensionality (ii) aggregation of input events reducing sequence dimensionality. Although both kinds are related, we shall focus on the latter one. Therefore aggregation will mean replacing a finite sub-sequence \( \{s(i), i_0 \leq i \leq i_1 \} \) of consecutive MO actuations with the actuation of a single complex MO, defined by the corresponding algebraic composition.

**Definition 6** - Complex MO. A complex MO \( C(F) \) is any algebraic composition of the elementary MOs \( A_h \) of an alphabet \( F \). The algebraic composition defines the IOR \( c = (u, y) \), the balance vector \( b = y - u \) and the vector \( m \) listing the cardinality \( m_h \) of each elementary MO \( A_h \) in \( C(F) \). The vector \( m \) will be called Bill of Manufacturing Operations (BOMO) and, for generality, it will be assumed to be an element of the rational space \( \mathcal{M} = Q^H \). Integer and non negative BOMOs will be said to be feasible.

**Remark.** Note that any BOMO \( m \) can correspond to a multiple of complex MOs having different algebraic compositions.

A fundamental equation relates the IO representations of the alphabet \( F \) to the IOR of any complex MO \( C(F) \).

**Theorem 3** - Consider an algebraic composition \( C(F) \) over an alphabet \( F = \{ A_h, h = 1, ..., H \} \). The following linear equation relates input/output and balance vectors of \( C(F) \) to its BOMO \( m \):

\[
\begin{align*}
\mathbf{u} &= B_u \mathbf{m}, \\
\mathbf{y} &= B_y \mathbf{m}, \\
\mathbf{b} &= (B_y - B_u) \mathbf{m} = \mathbf{Bm},
\end{align*}
\]

where matrix \( \mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_H] \) is the balance matrix of \( F \)

**Proof.** Since the balance vectors of a series and a parallel composing the same MOs are equal, let us replace series with parallel in the algebraic expression of \( C(F) \), thus obtaining the new complex MO \( C_+(F) \). Because parallel is commutative and associative, one can write

\[
C_+(F) = \sum_{h=1}^H m_h A_h
\]

and consequently

\[
\mathbf{b} = \sum_{h=1}^H m_h \mathbf{b}_h,
\]

thus proving the theorem. The same holds for input and output vectors.

**Remark.** The type partition induced on quantity vectors \( \mathbf{q} \in \mathcal{Q} \) by the alphabet \( F \) subdivides the balance matrix \( B \) into four row blocks \( B^T = [B_0^T B_1^T B_2^T B_3^T] \), where some block could be void.

**Example.** The balance matrix of the alphabet \( F \) in Figure 2, partitioned into three row blocks due to \( K_0 = \emptyset \), is shown next (zero entries are replaced by dots):

\[
B = \begin{bmatrix}
B_u & -1 & -2 & \\
& -1 & -2 & \\
& 1 & -1 & -2 \\
& 1 & -2 & -1 \\
& 1 & -1 & \\
& . & 1 & -1 & 2 -1 \\
& . & 1 & -1 & 2 -1 \\
& . & . & . & \\
B_f & & & \\
& . & . & . & \\
& . & . & . & 1
\end{bmatrix}
\]
Consider now a sequence \( \{C(0), C(1), \ldots, C(j), \ldots \} \) of complex MOs over an alphabet \( \mathcal{F} \) having an input, output and balance matrices \( B_u, B_y, B \). At any stage \( i \) we assume that a complex MO \( C(i) \) is actuated, implying that the command sequence (external events) forcing state variation is the sequence \( \{m(j)\} \) of their BOMOs. State variables and input/output flows remain the same, although now referred to the actuation stage of \( C(j) \). A state equation like (9) results, upon replacing the command vector \( s(j) \) with \( m(j) \).

**Remark.** The main difference between equation (9) and its aggregate version concerns the meaning of the command sequence. Here the command sequence is an arbitrary sequence of BOMOs to be designed in order to fulfill some criteria. Since any BOMO corresponds to a set of complex MOs having different algebraic compositions, the problem arises of selecting the most appropriate one or in other words, given \( m(j) \), of finding the most appropriate sub-sequence \( s(i) \), \( i \leq i \leq i(j + 1) \) such that \( m(j) = \sum_{i(i+j)} s(i) \). Such control problems, as already said, will not be treated here.

**Aggregate control problem.** A usual problem is the following: given a demand \( \{p(i)\} \) of finished products, find a sequence \( \{m(i)\} \) (i) producing on demand, i.e. satisfying \( x(i) + x(i) \geq p(i) \forall i \) and (ii) minimizing the location stock \( x(i) \).

**Steady state solution.** Let us start by investigating a steady state solution, such that (i) the stock is kept untouched, i.e. \( x(i) = x \), (ii) input and output (demand) flows are steady, i.e. \( p(i) = p \cdot r(i) = r \).

**Theorem 4** - The BOMO \( m \) solving the steady state problem is the solution of the linear equation:

\[
p - r = b = \beta m
\]

where \( b \) is a balance vector.

**Proof.** It follows directly from state equation.

The steady state balance vector \( b \) is free of semifinished types. It means that the input quantity of semifinished types is balanced by an equal output quantity. Complex MOs satisfying this property are very appropriate because they leave intact the stock of semifinished types. They will be called **balanced**.

**Definition 7** - **Balanced MOs.** A complex MO \( C(\mathcal{F}), c = (u, y) \) over an alphabet \( \mathcal{F} \) of elementary MOs is said to be **balanced** when its balance vector \( b = y - u \) is void of semifinished, i.e. the semifinished block is zero, \( b_s = 0 \).

Equation (13) can be employed to find the BOMO of a balanced MO \( C(\mathcal{F}) \) yielding a desired quantity \( p \) of finished products. Problems of this sort, existence and solution, have been already studied in [4]. The solution, if it exists, is two-fold: (i) the BOMO \( m \) of the balanced MO \( C(\mathcal{F}) \) and (ii) the raw material vector \( r \) to be supplied to \( C(\mathcal{F}) \). Then, given the BOMO \( m \), a further problem is to find an algebraic composition completely defining the MO. The balance vector \( b = -r + p \) plays the role of the **Bill of Materials** of the balanced MO.

**Example.** With reference to the alphabet \( \mathcal{F} \) of Figure 2, the product vector \( p = p(4) \) listed in equation (10) is yielded by a balanced MO \( C(\mathcal{F}) \) having as BOMO \( m \) the sum of the command sub-sequence of equation (8). The BOMO and the corresponding balance vector \( b \) are reported below:

\[
b^T = \begin{bmatrix} -2 & -5 & -8 & \ldots & \ldots & 1 & 2 \end{bmatrix} \quad \quad m^T = \begin{bmatrix} 2 & 3 & 2 & 1 & 2 & 1 & 1 & 1 \end{bmatrix}
\]

(14)

The solution of the aggregate control problem is strictly related to controllability problems of state equation (9) forced by the BOMO \( m \). The analysis is not afforded here, but it is important to remark that equation (9) is in general not controllable. The simple fact is that as in Figure 2, elementary MOs defining the command vector \( m \) may have more input/output object types. Complete controllability would occur in case of a single input/output object for each elementary MO. Due to instability of its eigenvalues, state equation (9) becomes not stabilizable if controllability fails.

5 **Conclusion**

A quite general formulation in term of state equations of the time-free dynamics of manufacturing-like processes has been provided. It is believed to be a useful starting point for solving real-time control problems.

**References**


