Manufacturing systems: parallel system models and some theoretical results

Sanjay Parthasarathy and Steven H. Kim
Laboratory for Manufacturing and Productivity, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

Abstract: A suitable model of the manufacturing environment must incorporate constructs to represent and manipulate information about concurrency, physical change, and time. The actor$^+$ model of the manufacturing system is one such model, based on the actor model of computation and the transformation model of production processes. This paper formalises the basic elements of the actor$^+$ model, which is then used to study the behaviour and capabilities of autonomous production systems. The new approach also offers some interesting insights into the design of completely autonomous production environments. Theoretical results, such as the partial decidability of a general process planning problem, can be derived using the actor$^+$ model.

Key words: actor-based manufacturing systems (ABMS), actor$^+$ model, manufacturing systems, parallel system models.

Reference to this article should be made as follows: Parthasarathy, S. and Kim, S.H. (1990) 'Manufacturing systems: parallel system models and some theoretical results', *International Journal of Computer Applications in Technology*, Vol. 3, No. 4, pp. 225–238.

1 INTRODUCTION

1.1 Background

The manufacturing environment can be viewed as a heavily parallel processing system, consisting of numerous processors (machines, robots, etc.) that transform the raw materials flowing through the system. The manufacturing system can therefore be modelled as a concurrent system whose computations correspond to physical operations.

An analogy can be drawn between the manufacturing system and the concurrent computer system. The significance of parallel systems lies in what they can accomplish in timely fashion. In scope, parallel computers can do no more than sequential machines or even Turing Machines; but they can dramatically reduce requirements on programming effort and task completion times. Advances in parallel computation and communication hold important lessons for the design of manufacturing systems, since the design considerations for parallel computing systems are analogous to those of the production environment.

An important step in the formalisation of the concepts of manufacturing systems is a model of the system itself. Any model of the manufacturing system must represent its inherent parallelism in addition to its transformational nature. The model must also represent and manipulate temporal information about concurrency and parallelism. The ability to reason about time is key to the planning and scheduling functions of the manufacturing system. Thus, any suitable model of the manufacturing system must incorporate constructs to represent and manipulate information relating to:

- concurrency and parallelism,
- transformation or change,
- time

Again, computer science offers interesting insights into formal models of parallel systems. The actor model, for example, is a model of concurrent communication in which several active and self-contained entities called actors process communications in parallel [1, 2]. The actor model together with the transformation model of production processes [3], a process-based perspective of the manufacturing system, can be used to develop an actor$^+$ model of the manufacturing system. The superscript *plus* (+) in actor$^+$ refers to the model's ability to simulate physical flows and transformations in addition to information flows and computations. To incorporate the ability to reason about time in the actor$^+$ model, we will extend the work on point and interval representations of time [4, 5, 6, 8], temporal reasoning [5, 9] and collections of intervals [7].

1.2 Outline

This paper proposes a model of the manufacturing system called the actor$^+$ model. The new approach is based on the actor model of computation and the transformation model of production processes.

The paper investigates the implications of the actor$^+$ model and the practical implications of parallelism and concurrency on the performance of production systems. In addition, the attachments review the foundations of the actor$^+$ model: parallel processing systems in Appendix A, the actor model of computation in Appendix B and the transformation model of production processes in Appendix C.

Section 2.1 demonstrates the rationale behind our choice of the actor model as the basis for the model of the production environment. It illustrates the correspondence between a formal representation of the manufacturing system and that of the actor model of computation. To overcome the actor model's inability to represent physical flows and transformations, it is combined with the transformation model to obtain the actor$^+$ model. Section 2.3 formalises the concepts of the actor$^+$ model and demonstrates its expressive power.

The actor$^+$ model offers some interesting insights into the design of autonomous production systems: those in which each component (whether a machine, robot, storage element, transport network or other mechanism) can be considered as an individual agent, while planning, scheduling and other system functions are automated. The actor-based manufacturing system (ABMS), an architecture for an autonomous production system, is discussed in Section 2.2. The actor$^+$ model is then used to obtain some interesting results about the capabilities of ABMS, such as batch (off-line) planning and real-time (on-line) scheduling.

The autonomous decision-making capabilities of the ABMS depend on translator modules associated with each component of the system. A translator module consists of an interpretation function that translates communications received from other system components. Section 2.4 formalises the function of the translator module.

Section 2.5 demonstrates the utility of the actor$^+$ model by proving the partial decidability of generalised process planning within ABMS's and notes the implications of the proof of partial decidability. Finally, the paper discusses future work in temporal reasoning.

2 PARALLEL MODEL OF THE MANUFACTURING SYSTEM

2.1 Actor$^+$ model of the manufacturing system

The actor model of computation is one in which several independent agents called actors send, receive and process communications in parallel. Since this model does not impose any control on the flow of communications, each actor can communicate with any other.

The actor model fulfils one of the requirements of a model of the manufacturing system. However, it models only information flows and computations on data streams, and as such is inadequate for modelling the transformational activities in the manufacturing systems. But this limitation can be addressed by combining the actor model and the transformation model of production processes. We call the result the actor$^+$ model of the manufacturing system. The general structure of the actor$^+$ model is similar to that of the actor model of computation (see Appendix B). But, communications in the actor$^+$ model are defined by the ordered pair ⟨part, information⟩.

Some concepts of the actor model may introduce unnecessary complexity in modelling the behaviour of the manufacturing environment:

The creation of actors, a concept crucial to the actor model, may seem irrelevant to manufacturing systems. Machines or robots within the system cannot create copies of themselves or of other machines. Moreover, actor creation implies a potentially infinite set of actors. These conceptual difficulties may be ovecome through the definition of an actor in the actor$^+$ model. Each machine, robot, etc. within the manufacturing system is assigned a unique 'address' or identifier. An actor is then defined as any distinct operation or transformation that can be performed at any of the addresses within the system. The concept of actor creation is retained and operations can now 'create' other operations.

Coordination of information and physical flows in the actor$^+$ model can be achieved by synchronisation of events within the system or by providing each actor with a buffer in which to store information until the corresponding physical resources arrive. Furthermore, the assumption that an actor is an independent agent necessitates access to information concerning other actors as well as process plans and schedules.

The mathematical structure of the actor$^+$ model is equivalent to that of the actor model of computation (see Appendix B) [10] and is given by:

$$\langle \mathbf{E, A, T, \text{-act}\rightarrow, Arr} \rangle$$

We propose that the actor$^+$ model be used to model the behaviour of the manufacturing system.

Proposition:
A manufacturing system can be simulated by an actor$^+$ model.

The rationale is as follows:
The simulation will work if a formal structure of the manufacturing system can be represented using the actor$^+$ model.

Let each component of the manufacturing system (machine, robot, store etc.) be assigned a unique address. Let each operation, performed at any of the addresses within the system, be represented by an actor. Then, the

set of actors within the manufacturing system, represented by $\mathbf{A}=\{a_1, a_2, a_3, \ldots\}$, is potentially infinite. This infinitude is consistent with the actor model of computation.

Let the three basic transformations be represented respectively by g, i, and u. Then the set of all possible events within a manufacturing system is given by:

$$\Sigma^* = \{\Lambda, g, i, u, gg, gi, gu, ig, ii, iu, ug, ui, uu, giu, gui, ggi, ggu, ggg, \ldots\}$$

where Λ is the null transform and $\Sigma=\{g, i, u\}$ is the alphabet. Each element of Σ^* is called a word w. A word such as giu implies that g precedes i which in turn precedes u. A sentence s is then a collection of words, i.e. $s = w_1 \cdot w_2 \cdot w_3 \cdot \cdots \cdot w_n$ where $w_1 \ldots w_n \in \Sigma^*$. Now, the set of all operations that can be performed within a given manufacturing system is defined as

$$\mathbf{E}=\{w \mid w \in \Sigma^*\}.$$

Consider the schedule $\mathscr{S}$ for a manufacturing system. In essence, the schedule $\mathscr{S}$ is a function that maps a set of operations to a set of manufacturing system components. Therefore,

$$\mathscr{S}: \mathbf{E} \rightarrow \underline{\mathbf{A}}$$

where $\underline{\mathbf{A}}$ is the set of manufacturing system components. Moreover, the schedule $\mathscr{S}$ specifies the arrival times of parts at appropriate machines, the duration of operations, etc., through a rigorous timetable of events.

Consider the set of process plans $\mathscr{P}$ for a set of parts that can be processed within the given system. The set of plans $\mathscr{P}$ specifies operations pertaining to the parts and is a partial ordering of the operations required to manufacture them.

The manufacturing system can now be represented by the following 4-tuple:

$$\langle \mathbf{A}, \mathbf{E}, \mathscr{S}, \mathscr{P} \rangle$$

The manufacturing system structure corresponds to the actor$^+$ model structure. The schedule $\mathscr{S}$ contains the information embodied in the target function (**T**) and the arrival ordering (**Arr**) of the actor model. The set of process plans contains the information embodied in the activation ordering (**-act→**) of the actor model. Thus, the manufacturing system has a structure similar to that of the actor$^+$ model. Hence, the actor$^+$ model can simulate the manufacturing system.

2.2 Actor-based manufacturing systems (ABMS)

In this section, an autonomous manufacturing system called the actor-based manufacturing system (ABMS) is proposed. The actor-based manufacturing system is one in which each machine, robot, and other system component is an independent agent that can communicate with other agents and make decisions concerning system operation and control.

Each such system component is assigned a unique mail address. To each address is assigned a translator module. The translator module consists of an interpretation function. Each actor can behave as an independent agent since it can access the translator module. Communications are a ⟨part, information⟩ couple, and information is represented using the notation of the transformation model.

Consider the simple ABMS shown in Figure 1, consisting of three machines M_1, M_2 and M_3 and a store S. Let A and B be two products that can be produced within the system. Let the direction of the arrows connecting the system components indicate admissible information and material flow channels. Let the flow paths for A and B be:

Path (A): $S \rightarrow M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow S$

Path (B): $S \rightarrow M_2 \rightarrow M_3 \rightarrow S$

Let the operations performed in the system for one unit of A and one unit of B be as shown in Figure 2. Figure 2 shows operations at individual machines and two possible overall sequences of events. Sequence 1 makes better use of time and material resources than Sequence 2. In order that the ABMS, operating autonomously, selects Sequence 1 over Sequence 2, information concerning the simultaneous operation of machines M_1 and M_2 must be conveyed and processed. Such information is specified in the arrival ordering (**Arr**) or the schedule $\mathscr{S}$ for the system. Moreover, information concerning the partial order of events is specified through the activation ordering (**-act→**) or through the process plans for A and B.

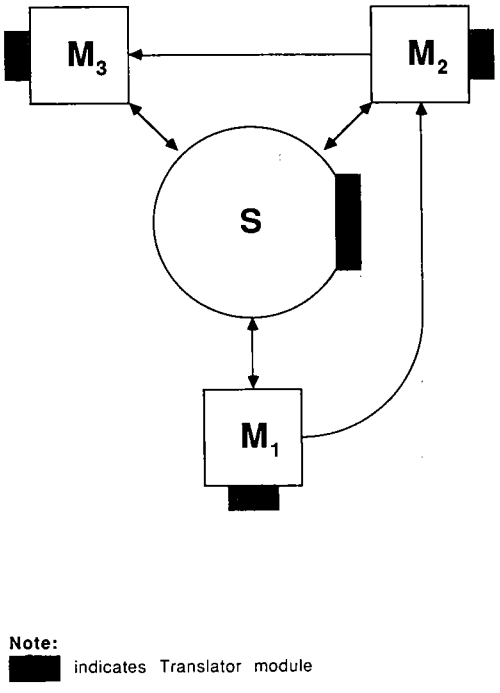
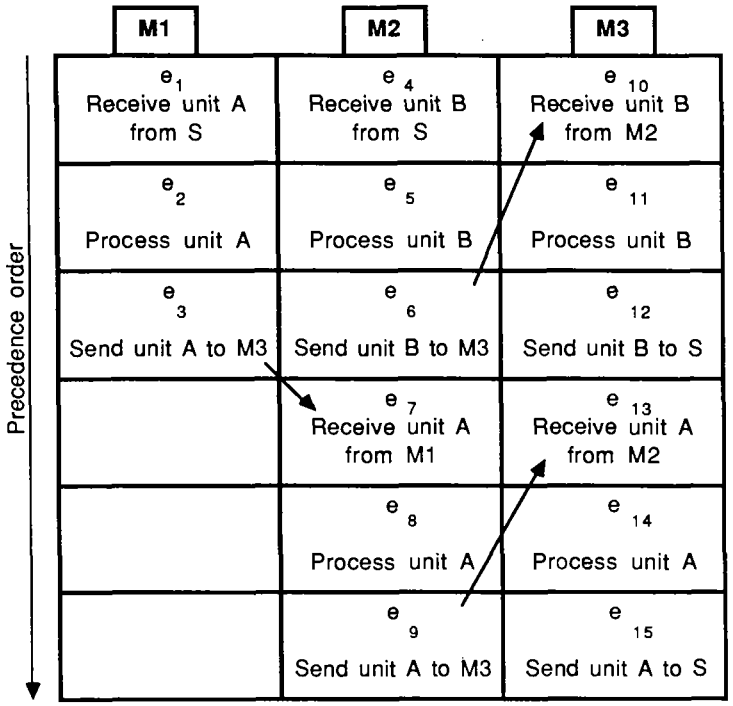


Figure 1 *An actor-based manufacturing system (ABMS).*



$\underline{A} = \{ M_1, M_2, M_3, S \}$

$E = \{ e_1, e_2, e_3, \ldots, e_{15} \}$

$T = \{ e_1, e_2, e_3 \} \rightarrow M_1; \{ e_4, e_5, e_6, e_7, e_8, e_9 \} \rightarrow M_2;$
$\{e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15} \} \rightarrow M_3$

Activation ordering = $e_3 \rightarrow e_7; e_6 \rightarrow e_{10}; e_9 \rightarrow e_{13}$

Arrival ordering = simultaneous (e_1, e_4); simultaneous (e_7, e_{10})

Sequence 1:
$< e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15} >$

Sequence 2:
$< e_1, e_2, e_3, e_7, e_8, e_9, e_{13}, e_{14}, e_{15}, e_4, e_5, e_6, e_{10}, e_{11}, e_{12} >$

Figure 2 *Events within ABMS.*

2.2.1 The translator module

2.2.1.1 Purpose of the translator module

Associated with each address within the ABMS is a translator module. The function of the translator is two-fold:

- to determine whether an operation or a sequence of operations can be executed at that address (feasibility);
- to determine whether the operation or sequence of operations, given the initial raw material condition, achieves the specified final conditions (acceptability).

The feasibility check refers to system capabilities. It involves checking that the dimensions, tolerance, surface finish, hardness and other gross specifications of the final product are achievable within the system. Such a check may be effected by matching the product specifications with the system capabilities. The acceptability test, on the other hand, involves a detailed analysis of the operation parameters (speed, feed, depth of cut, temperatures, pressures, etc.), fixturing, and precedence relationships among operations.

The mechanism for testing, both for feasibility and for acceptability, depends on the representation of knowledge. The rest of this section formalises the general principles of acceptability and feasibility testing.

The translator module consists of an interpretation function $\mathcal{I}_a$. This interpretation function translates communications received from the rest of the system. The *domain* of the function (D_a), corresponds to the set of operations that can be performed at the address. Then, the *translator* associated with an address within the system consists of a function $\mathcal{I}_a$ that maps transformation words to the set of operations that can be performed at that address.

The interpretation functions enable each address to interpret communications received from other addresses.

The function translates instructions that are represented using the notation of the transformation model (Appendix C). Moreover, the interpretation function also translates instructions that are well formed formulas defined over the standard logical operators such as $\exists$, $\forall$, $\wedge$, $\vee$, etc. The interpretation functions of addresses within the ABMS assist in online (real-time) scheduling within the system.

2.2.1.2 Interpretation function: definition and illustration

Let $\mathscr{I}_c$ be a mapping of constants (denoted by $w \in \Sigma^*$) to D_a, and $\mathscr{I}_v$ a valuation function which maps variables (denoted by x, y, z, etc.) to D_a. Then the interpretation function can be defined as a pair consisting of a model M and the valuation function $\mathscr{I}_v$. The model M is a pair consisting of the domain of the mapping D_a and the function $\mathscr{I}_c$. Thus, the interpretation function is defined as:

$$\mathscr{I}_a : \Sigma^* \rightarrow D_a^*$$

here the following relationships hold:

$$D_a^* = \Pi_{j=0 \text{ to } n} D_a^j$$
$$\mathscr{I}_a = \langle M, \mathscr{I}_v \rangle$$
$$\mathscr{I}_a = \langle \langle D_a, \mathscr{I}_c \rangle, \mathscr{I}_v \rangle$$

Consider the following example of an interpretation function for a lathe:

$$\mathscr{I}_{\text{lathe}} = \langle \langle \{\text{turning, drilling}\}, \{W \rightarrow \text{turning}, w \rightarrow \text{drilling}\} \rangle, \{x \rightarrow \text{turning}, x \rightarrow \text{drilling}\} \rangle$$

This function specifies that the lathe can have two distinct behaviours, namely, that of a turning agent and a drilling agent. Now, $\mathscr{I}_{\text{lathe}}$ will map all geometric transformations into one of these two behaviours.

Now, consider the interpretation of a logical formula which states that there exists a word w that is the combination of two other words x and y such that x precedes y. This formula can be satisfied by a material removal operation which is composed of two successive removal operations that achieve the same end result.

$$\mathscr{I}_{\text{lathe}}(\exists w, x, y \in \Sigma^*(xy = w))$$

Now, consider the possible interpretations of the above sentence by the function $\mathscr{I}_{\text{lathe}}$. $\mathscr{I}_{\text{lathe}}$ assigns all possible combinations of elements in its domain to w, x and y. The set of all possible assignments is:

1 $w \rightarrow$turning; x, $y \rightarrow$turning
2 $w \rightarrow$turning; $x \rightarrow$turning, $y \rightarrow$drilling
3 $w \rightarrow$turning; $x \rightarrow$drilling, $y \rightarrow$turning
4 $w \rightarrow$turning; x, $y \rightarrow$drilling
5 $w \rightarrow$drilling; x, $y \rightarrow$turning
6 $w \rightarrow$drilling; $x \rightarrow$turning, $y \rightarrow$drilling
7 $w \rightarrow$drilling; $x \rightarrow$drilling, $y \rightarrow$turning
8 $w \rightarrow$drilling; x, $y \rightarrow$drilling.

Since a turning operation can be composed of two separate turning operations or a drilling operation of two separate drilling operations, only assignments 1 and 8 are admissible.

2.2.1.3 Group interpretation functions: definition and illustration

Interpretation functions associated with groups of machines, robots, and others, may now be defined. For example:

$$\mathscr{I}_{\text{lathe}} = (\{\{\text{turning, drilling}\}, \{w \rightarrow \text{turning}, w \rightarrow \text{drilling}\}), \{x \rightarrow \text{turning}, x \rightarrow \text{drilling}\})$$
$$\mathscr{I}_{\text{grind}} = (\{\{\text{grinding}\}, \{w \rightarrow \text{grinding}\}), \{x \rightarrow \text{grinding}\})$$
$$\mathscr{I}^+_{\text{lat-gri}} = (\{\{\text{turning, drilling, grinding}\}, \{w \rightarrow \text{turning}, w \rightarrow \text{drilling}, w \rightarrow \text{grinding}\}), \{x \rightarrow \text{turning}, x \rightarrow \text{drilling}, x \rightarrow \text{grinding}\}).$$

where $\mathscr{I}^+_{\text{lat-gri}}$ is the interpretation function corresponding to the group comprised of the interpretation functions $\mathscr{I}_{\text{lathe}}$ and $\mathscr{I}_{\text{grind}}$.

A group of addresses is associated with an intepretation function $\mathscr{I}^+_m$, where m specifies the addresses that define the function $\mathscr{I}^+_m$. For example, the interpretation function associated with a cell consisting of machines M_1, M_2 and M_3 can be defined as:

$$\mathscr{I}^+_{123} = \langle \langle D^+_{123}, \mathscr{I}^+_{c123} \rangle, \mathscr{I}^+_{v123} \rangle$$

where the following relationships hold:

$$D^+_{123} = U_{\{a\}} D_a$$
$$\mathscr{I}^+_{c123} = U_{\{a\}} \mathscr{I}_c$$
$$\mathscr{I}^+_{v123} = U_{\{a\}} \mathscr{I}_v.$$

Here, $U_{\{a\}}$ is defined as the union over the set $\{M_1, M_2, M_3\}$.

The interpretation function associated with an entire system ($\mathscr{I}_s$), can play the role of a knowledge-based advisory system since it contains the information required for planning. A database containing interpretation functions associated with the addresses can be used for batch (off-line) planning and as a knowledge based advisory system.

2.2.1.4 Satisfiability

We can now define the concept of 'satisfiability'. A sentence s is satisfied ($|$) by a system ($ABMS_i$) if and only if there exists, within the system, a set of addresses whose interpretation function can assign elements from its domain to the words $w_1, \ldots, w_n$ in s such that the specified initial and final conditions are achieved. Let the set of addresses be denoted by $\underline{A}$ and let each individual address is denoted by $\underline{a}_i$, where i ranges from 1 to n and n is the total number of addresses. Then:

$$(ABMS_i \mid s) \text{ iff } \exists \{\underline{a}_i\} (\underline{A} \supseteq \{\underline{a}_i\}) \wedge (I^+_m \mid s))$$

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.