Data & Knowledge Engineering 6 (1991) 421–443
North-Holland

Time sequence ordering extensions to the Entity-Relationship model and their application to the automated manufacturing process*

James R. Moyne, Toby J. Teorey and Leo C. McAfee, Jr.
Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, MI 48109-2122, USA

Abstract

Moyne, J.R., T.J. Teorey and L.C. McAfee, Jr., Time sequence ordering extensions to the entity relationship model and their application to the automated manufacturing process, Data & Knowledge Engineering 6 (1991) 421–443.

New extensions to the entity-relationship (E-R) model have been developed to represent *time sequencing* and *ordering* aspects of information flow, and to represent the integration of control (programming) information into a database. The model constructs specify an implementation *ordering* of records in a relational database table. *Time sequencing* refers to an implementation of process information flow as a result of this ordering. The modeling constructs are needed to more completely model process recipe information flow in a typical automated manufacturing facility. The development of these E-R extensions is pursued using a data model for the factory of the future as a motivation and development vehicle. The extensions are defined formally and possible variations of the constructs are given. Further, the incorporation of the constructs into the existing E-R model semantics and the transformation of these extensions to ordering properties and integrity constraints on a corresponding database implementation is discussed.

Keywords. Automated manufacturing; database design; data models; Entity-Relationship model; integrity constraints; ordering; process flow language; time sequencing.

1. Introduction

Process and facility automation is increasingly becoming a topic of research and discussion. A primary goal of the designers of the 'factory of the future' automated facility is increased yield and production through complete automation of the manufacturing process [4, 6, 7, 8, 9, 11]. One aspect of automation that must be addressed is data modeling of the facility. A significant 'first' effort has been documented in the literature [9]. As a result of this effort, the hierarchical automated facility structure and many of its properties have been identified. Further, a generic database model of process recipe information flow in the facility has been developed [9, 10]. The model was developed using entity relationship (E-R) constructs [1, 13, 15]. (An overview of pertinent E-R constructs is presented in Appendix A.) The model quite adequately represents many aspects of process recipe information flow in the facility, but the need was realized for further extensions to existing E-R modeling constructs to represent ordering and time sequencing of process recipe information.

* This work is supported by the Semiconductor Research Corporation (SRC) under contract #89-MC-085.

Research documented in this paper builds directly on recent work on database modeling of the automated facility [9]. In this paper the issue of E-R modeling extensions needed to conceptualize time sequencing and ordering of process recipe information in automated manufacturing is addressed. The motivation for the development of these extensions is discussed and the development of the extensions is pursued through case studies of various automated manufacturing facility types. Problems with existing techniques used to model the process information flow are identified and those problems indicate the need for further extensions to the E-R model to incorporate ordering (scheduling) information. The extensions model an *ordering* of records in a relational database table. In an actual implementation (such as the automated facility) using such a database, this ordering would be interpreted to a *time sequence* for process information flow. For each extension proposed to the E-R model, the need for the extension is developed and illustrated in Section 2. In Section 3 a formal syntactic and semantic definition for each extension is provided and the interpretation of these extensions to a database implementation is given. It is important to note that the E-R extensions developed in this paper impact database structure, data integrity, processing overhead and applications processing. However, none of the extensions impact on the existing formalism of the E-R model. Also, in addition to modeling the database aspects, these extensions give insight into information flow and information processing. Indeed the extensions model a database that incorporates control information into its data and data structure; such control information might otherwise be incorporated into application programming accessing the database.

2. Case study for development of time sequence ordering extensions to the E-R model: automation in the semiconductor manufacturing facility

A primary goal of any manufacturing facility is, of course, to manufacture a product at minimum cost. The collective ordered information that results in the manufacture of a particular product in a manufacturing facility is referred to as the *process recipe* for that product. In implementing such a process recipe in an automated hierarchical facility, the recipe may be subdivided at each level of the facility hierarchy according to the location at which a task is performed or coordinated. Further, the detail of recipe information at a node location in the facility is such that the node can interpret recipe commands from its parent node (if a parent node exists) and instruct its children node(s) to perform tasks in implementing a command (the parent/child node relationship is described in the next paragraph). The resulting qualities of division of labor and transparency that are apparent in the implementation of a process recipe are attractive features in a heterogeneous vendor equipment and controller facility environment (an environment typical of many manufacturing facilities) [9].

The detail of the division of the process recipe for implementation in a hierarchical facility has been documented in recent literature [9] and is illustrated in *Fig. 1*. At the top of the facility hierarchy the *Factory Computer* contains the *Global Process Recipe* (*GPR*) of the product. The GPR in general is a high level overview of the product recipe. The GPR may be subdivided into *Global Process Recipe Steps* (*GPRSs*). A GPRS is usually sent to a *Local Controller* (*LC*) (a child of the parent factory computer) over a local area network, where it is 'interpreted to' a *Local Process Recipe* (*LPR*). In a similar manner, the LPR may be further subdivided into *Local Process Recipe Steps* (*LPRSs*), each of which is usually sent over a subnetwork to a *Sub-Local Controller* and interpreted to a *Sub-Local Process Recipe* (*SLPR*). This recipe division process may continue down the levels of the facility hierarchy terminating at the facility equipment (see *Fig. 1*) [9].

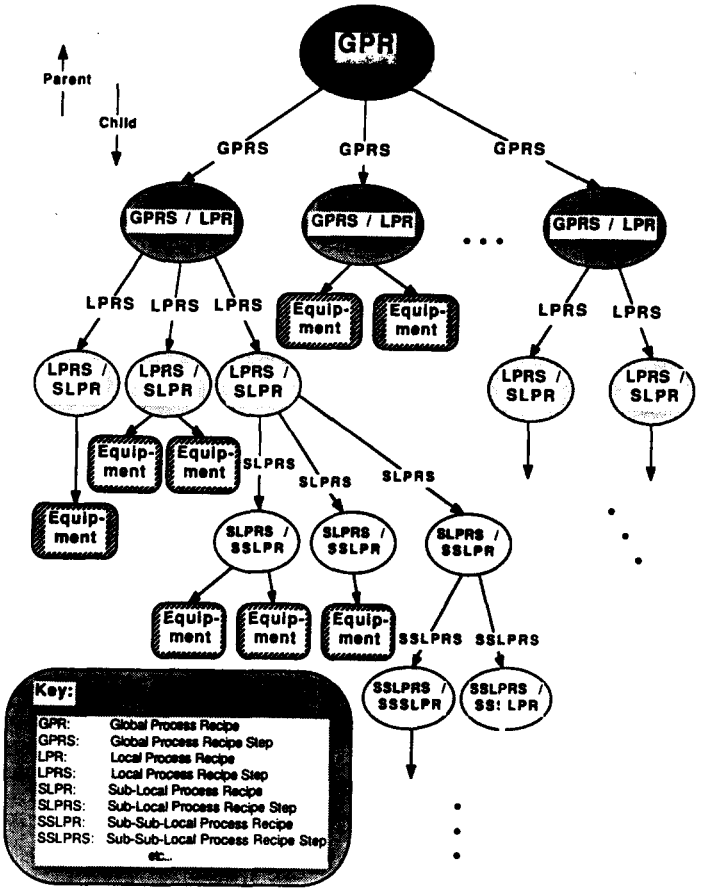


Fig. 1. The division of the process recipe and process recipe flow.

A desirable property of the factory of the future is the automated implementation of a process recipe. Such an implementation is data intensive. Further, the structure of the database and the data it contains could also represent the recipe implementation procedure, i.e. process control information may be incorporated into the database. Such a practice of incorporating procedural information into the database portion of an application would generally reduce the size of application code. It would also result in a decrease of update costs as it is easier to update data in a database than update procedural code.

In light of these issues it is clear that a data model of process recipe information flow in the facility is an important contribution. A data model has been developed to generically represent process recipe information flow in a hierarchical automated facility. This model uses existing E-R constructs and its development is detailed in Moyne et al. [9]. An E-R model of a typical hierarchical automated facility is shown in *Fig. 2.* The model shows that each recipe step at a layer in the facility is related exclusively to one recipe at one location in the next lower layer of the facility. For example each GPRS corresponds to exactly one LPR (or internal command) at exactly one location. An in-depth analysis of the model indicates that the represented facility exhibits qualities of division of labor, transparency between layers, generic attributes, structure, and modularity [9].

Entity-relationship constructs thus can be successfully used to model many aspects of process recipe information flow in the automated facility. However, there are time sequencing and ordering properties of the recipe flow not conveyed by existing model constructs,

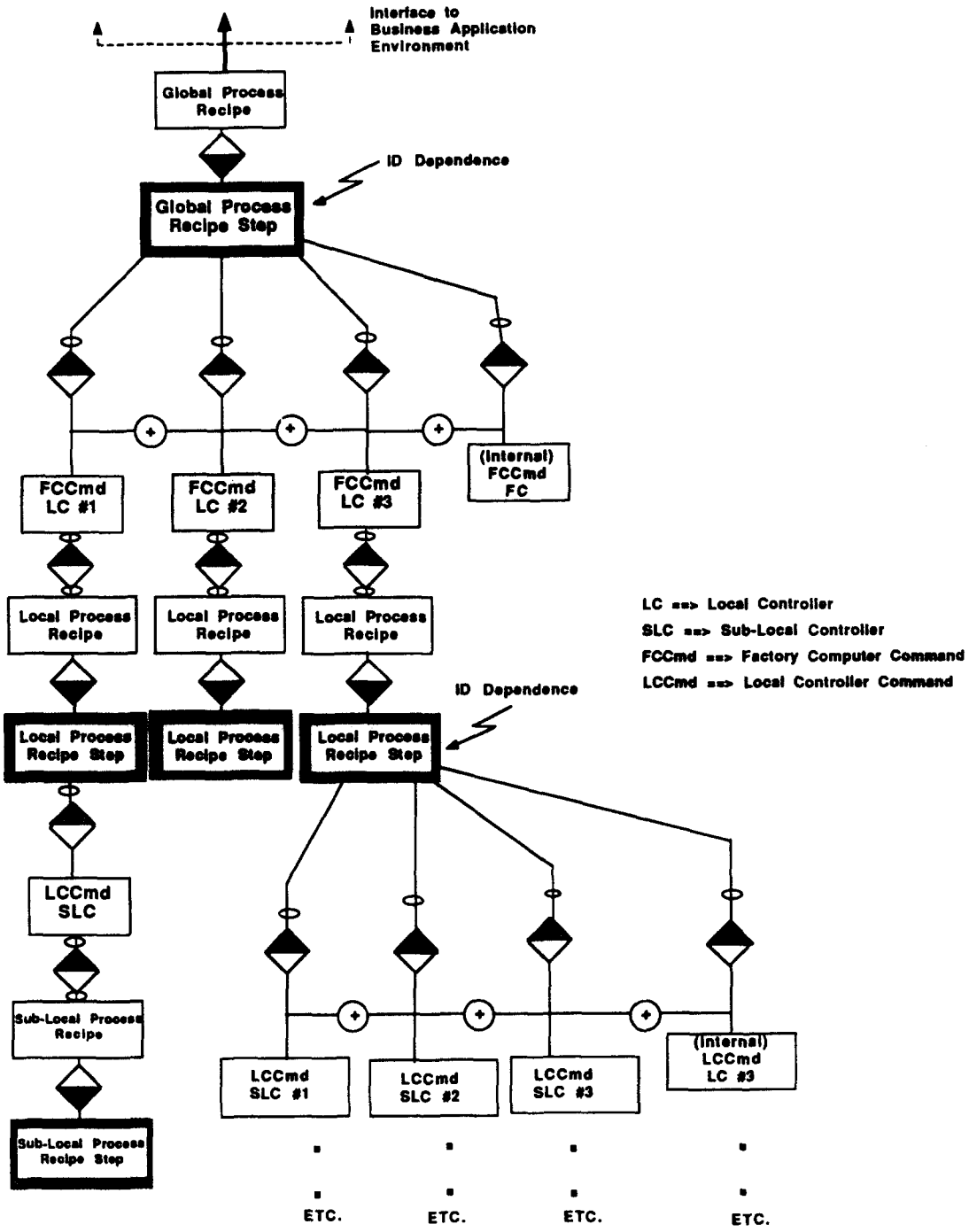


Fig. 2. Entity-Relationship model of a typical hierarchical automated facility.

thus the model is incomplete. The implementation of a recipe is a time sequential process [5, 11]. This sequentiality is derived from the ordered information in the facility database. This information ordering may reflect a strict well-defined physical as well as logical order of processing (e.g. an assembly line), a random physical and logical order of processing (e.g. an environment where multiple products are manufactured and information ordering varies according to the manufacture of each product), or any of the many degrees of physical and/or logical ordering between these two extremes (e.g. an automated semiconductor manufacturing facility). A process recipe information flow data schema should contain this ordering and sequencing information as the database application must efficiently determine recipe step ordering and destinations of process messages sent over the backbone network or any subnetwork.

In all instances described, an ordering of information is indicated as a necessary quality of the database implementation for proper facility operation. Thus any corresponding database model should convey this ordering quality. An investigation of E-R techniques and existing extensions reveals that there are no extended E-R constructs to represent any ordering properties of a database [9]. Thus, the modeling of the recipe process in the automated facility requires the development of new E-R model extensions to represent these sequencing and ordering properties. The following general statements can be made concerning the description of these extensions:

> The extensions indicate an *ordering* of records in a relational database table. In the design of an actual database these extensions would translate to an index on an attribute set or an ordering specified by a data manipulation language statement applied to a view.
>
> In an actual implementation using such a database, this ordering would be interpreted to a *time sequence* for process information flow (such as for the automated facility).

In the following subsections, these extensions are developed to represent the various degrees of sequentiality in a process and the resulting ordering of data that are possible. In the first subsection, extensions are developed to represent a strict physical time sequential ordering. An example application is a simple assembly line process where the product visits all stations on the line, once each, in a strict sequence. In the second subsection, E-R extensions are developed to represent a future facility with a robotic type interprocess transport mechanism where process information (and material) flow is not physically fixed by an assembly line. This scenario is referred to as a logical sequential ordering situation. Concepts of pseudo ordering and repetition of recipe sequences are explained and modeled. In the final subsection, hybrids and combinations of constructs developed in the first two subsections are considered and a hierarchical example facility model of the automated semiconductor manufacturing facility is provided.

2.1. E-R extensions to represent time sequencing and ordering properties in a physically ordered situation: the assembly line

A *physical ordering* of the processing step sequence for a product occurs in an *assembly line* processing environment. Cases to consider depend on the attributes upon which the process step ordering is based and also on the number of steps to be executed consecutively at a particular controller in the processing sequence. Examples to consider that illustrate the possibilities of the number of steps executed consecutively at a location include: (1) a product visits each processing station for exactly one process recipe step implementation, (2)

a product visits each processing station for between one and n_s process recipe step implementations (where $n_s \geq 1$, s = station#), and (3) the product visits each processing station for between zero and n_s process recipe step implementations (i.e. $n_s \geq 0$).

Examples to consider that illustrate the various possible (time sequence) *ordering schemes* include:

1. order by recipe step within a recipe, i.e. complete the ith step of recipe j before implementing the $i+1$ step of recipe j – order of recipe implementation is not specified.
2. order by recipe step within a recipe – process recipes in order, i.e. begin recipe j before beginning recipe $j+1$.
3. order by recipe step over all recipes, i.e. complete the ith step of *all* recipes before implementing the $i+1$ step of *any* recipe.

In a first example an implementation of the process flow 'order by recipe step within a recipe – order of recipe implementation is not important' is analyzed. The simple assembly line of a generic automated facility is considered and illustrated in *Fig. 3a* (the bold arrow at the bottom of the figure indicates the direction of process flow). Let P_x identify a type of product in which $1 \leq x \leq k$ and k is the total number of types of products that can be produced. The product P_x is assembled using the Global Process Recipe GPR_{P_x}. Local controllers, LC_1 through LC_n each receive exactly one GPRS and implement exactly one corresponding LPR. The GPRSs are sequentially ordered from 1 through n. Furthermore, for any i ($1 \leq i \leq n-1$), $GPRS_{i+1}$ cannot begin until $GPRS_i$ has been completed. (Note then that LC_i implements $GPRS_i$.) Thus the process flow proceeds as shown in *Fig. 3b* according to the ordering constraint of 'order by recipe step within a recipe – order of recipe implementation is not important'.

A data model for the process using existing E-R constructs would resemble that of *Fig. 2* [9]; however, necessary information concerning the time sequence (ordering) of the process is not indicated by the model. The time sequence ordering of the process can be conceptually illustrated with an arrow added to the exclusive OR extension as shown in *Fig. 4*. The arrow points in the direction of the sequence and, in keeping with E-R modeling tradition, attempts should be made for arrows (and thus the process sequence) to point from left to right or top to bottom. Also, an additional arrow with no source (open ended) points to the 'starting point' or the first LC in the processing sequence, and another arrow with no destination emanates from the 'ending point' or the last LC in the processing sequence. The arrow notation of *Fig. 4* can be read as 'occurs before'.

This notation, however, is still incomplete as information of attribute(s) over which the information ordering is based is not indicated. In this example, the ordering information that must be conveyed is that ordering is decided by recipe step within a recipe. This information may be conveyed by listing pertinent ordering attributes above and below the exclusive OR symbol. The significance of the placing of the pertinent attributes is the following:

> The attribute set placed *below* the exclusive OR symbol determines the subsets of the elements of the parent entity (a partitioning) over which ordering is to occur. A null attribute set is to be interpreted that ordering occurs over the parent entity as a whole.
>
> The attribute set placed *above* the exclusive OR symbol indicates the attributes to be used to determine ordering within the subsets determined by the attribute set placed below the exclusive OR symbol.

Variations in the ordering scheme for the assembly line in *Fig. 3a* (within the strict

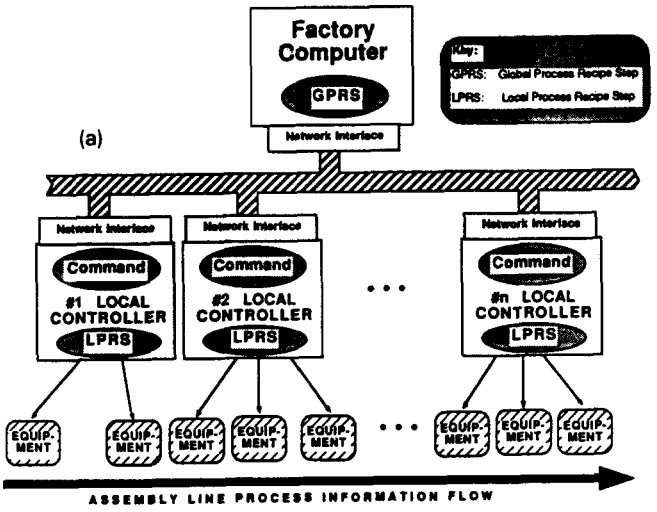


Fig. 3a. The assembly line: physical ordering.

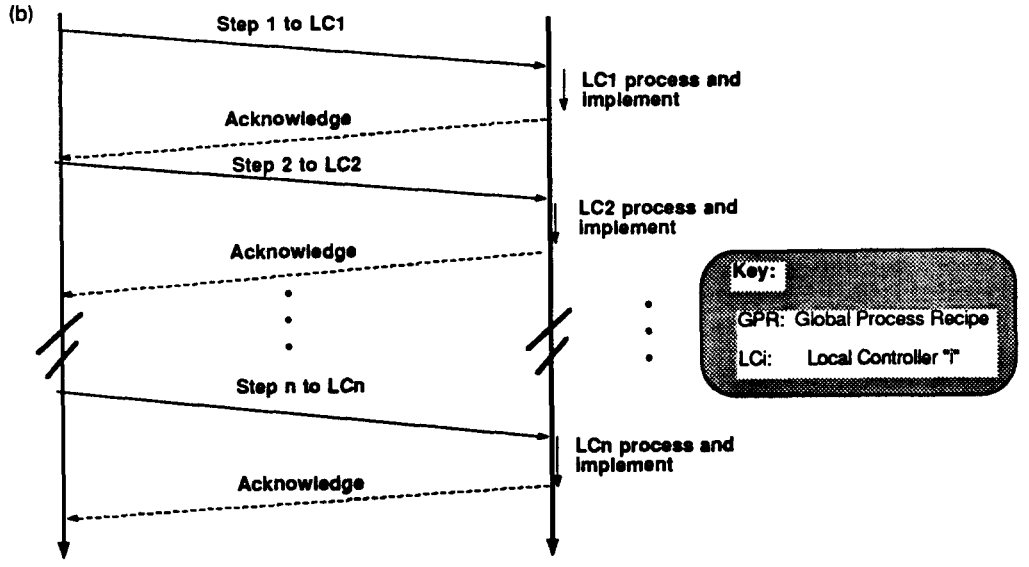


Fig. 3b. Process recipe information flow for a GPR.

ordering scheme implied by the physical layout of the assembly line) can be modeled by altering the attribute sets placed above and below the exclusive OR symbols. In the first example ordering scheme introduced above. '(time sequence) order by recipe step within a recipe', the assembly line scenario outlined above is a manufacturing example. Ordering is determined by the attribute Global Process Recipe Step Number within subsets (or partitions) of the parent relation 'GPRS'. These subsets are determined by unique occurrences of the 'Global Process Recipe Number' parameter in the parent relation 'GPRS'. The ordering information is represented with extended entity relationship (EE-R) constructs as shown in *Fig. 4.* Note that each subset to be ordered is determined by a unique value of recipe number; so the parameter GPR# is placed *below* the exclusive OR symbol. Also note that

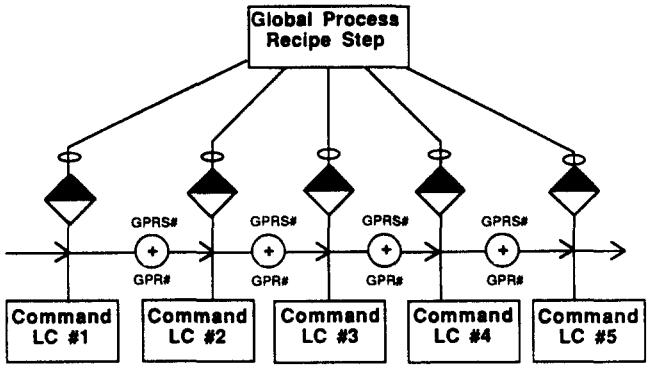


Fig. 4. EE-R model: illustration of the physical sequential ordered exclusion function (ordered by Global Process Recipe Step Number within each Global Process Recipe).

ordering within a GPR is determined by GPRS#, therefore the parameter GPRS# is placed *above* the exclusive OR symbol.

The second example ordering scheme introduced was '(time sequence) order by recipe step within a recipe – process recipes in order'. The subset to be ordered is the entire parent relation 'GPRS'; therefore a null symbol is placed below each exclusive OR symbol. Ordering is determined first by GPR# and then by GPRS# within GPR, therefore the attribute set '(GPR#, GPRS#)' is placed above each exclusive OR symbol. The resulting extended entity-relationship (EE-R) model is shown in *Fig. 5.*

As to the third ordering scheme, '(time sequence) order by recipe step over all recipes' (i.e complete the *i*th step of *all* recipes before implementing the $i + 1$ step of *any* recipe), the subset to be ordered is again the entire parent relation 'GPRS' and a null symbol is placed below each exclusive OR symbol. Ordering within the subset, however, is determined first by GPRS# and then by GPR#, therefore the attribute set '(GPRS#, GPR#)' is placed above each exclusive OR symbol. (The resulting EE-R model then would be similar to *Fig. 5*, with the exception that the attribute set '(GPRS#, GPR#)' would be placed above each exclusive OR symbol.)

A second extension to E-R model constructs is used to indicate the number of process steps to be implemented consecutively at a particular location. As a first example consider the assembly line of *Fig. 3a* again, but with the relaxed constraint that local controllers, LC_1 through LC_n may now receive one or more consecutive GPRSs in processing a product. The

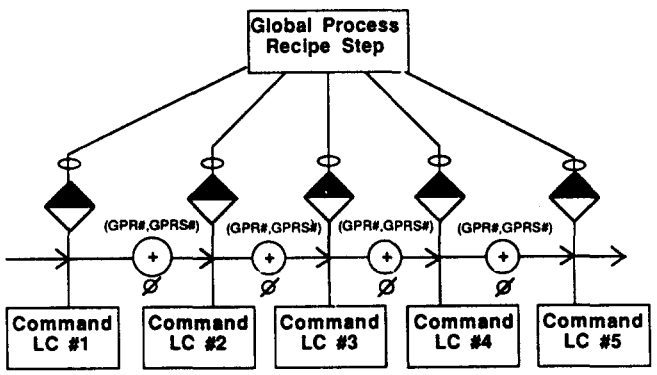


Fig. 5. EE-R model: illustration of the physical sequential ordered exclusion function (ordered by Global Process Recipe Step Number within each Global Process Recipe, Process Global Process Recipes in order).

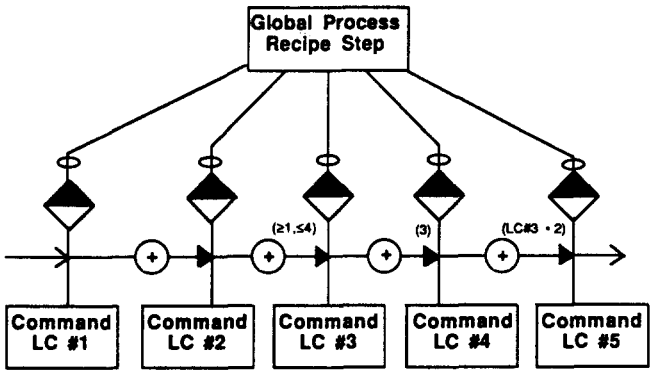


Fig. 6. EE-R model: illustration of the physical sequential ordered exclusion function (illustrating specification of number of process steps to be executed consecutively at each Local Controller).

number of process steps to be executed consecutively at a local controller for a given sequence is now greater than or equal to one, i.e. the number is 'many'. With current E-R modeling techniques, 'many' is indicated with a shaded area (as in a 'one to many' binary relationship). The property of 'many' process steps occurring at a location in a sequence will be specified by shading in the sequence arrow head pointing to that location. In addition, terminology may be placed near the arrow head to further specify the number of process steps occurring at that location. As an example consider the assembly line of *Fig. 6*. Here the number of process steps at each local controller for a given sequence are as follows: LC_1 = 'one' (note unshaded arrow head), LC_2 = 'many' (unspecified number), LC_3 = '1 to 4', $LC_4 = 3$, and $LC_5 = LC_3 * 2$, (where '=' refers to equivalence). Note that many other numeric ranges are possible. It should also be noted that the arrow head 'many' construct has a different conceptual meaning than the many-to-one 'Is Interpreted To' relationship. The many-to-one relationship reads: 'Many GPRSs may be interpreted to the same Local Controller command'. The arrow head 'many' construct reads: 'The number of consecutive steps of a sequence determined by the parameters placed above and below the exclusive OR construct may be *many*, i.e. many commands may be received and interpreted consecutively at the local controller in the same process sequence'.

A second example dealing with the number of consecutive process steps in a sequence at a location addresses the issue of optionality. Again consider the example of *Fig. 3a*, but with the relaxed constraint that local controllers, LC_1 through LC_n may now receive zero or more GPRSs in processing a product. The number of process steps executed at a local controller for a given sequence may now be zero, thus the issue of optionality must be conveyed. The commonly used optionality 'ring' construct is used here also and is placed over the line between the exclusive OR symbol and the arrow head. The symbol may be interpreted as: 'The number of steps of a sequence determined by the parameter placed above and below the exclusive OR construct may be *zero* i.e. there may be no commands received and interpreted at a local controller for some process sequence'.

As a final depiction of the assembly line process, *Fig. 7* is included to illustrate the various constructs developed in a hybrid manufacturing environment. A hybrid environment is typical of an actual manufacturing process in most industries.

2.2. E-R extensions to represent time sequencing and ordering properties in a logically ordered situation

A *logical ordering* of the processing sequence for a product occurs in an environment where there may be no physical assembly line or other strong dictation of an exact time

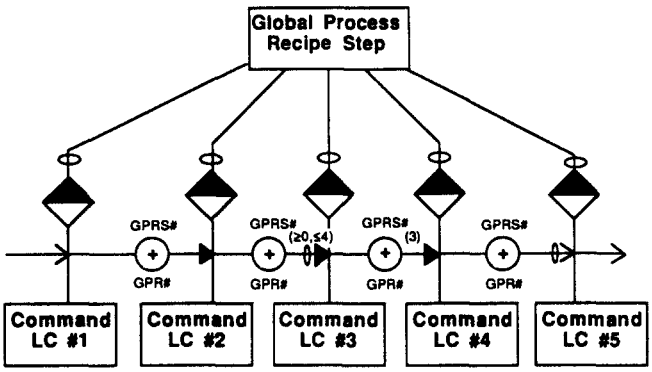


Fig. 7. EE-R model: illustration of the physical sequential ordered exclusion function in a hybrid environment (ordered by Global Process Recipe Step Number within each Global Process Recipe) illustrating specification of number of process steps to be executed consecutively, and illustrating optionality at each Local Controller.

sequence in ordering of processes. This section contains a discussion of the possibilities of the ordering of LC-to-LC travel for a product in a logically ordered manufacturing environment along with the constructs developed as a result. Note that extensions to the E-R model developed in this section may co-exist with extensions developed in Section 2.1.

(1) *Random*: No predictable sequence of the LC-to-LC travel for an IC/wafer is indicated. In this case the extended E-R diagram must indicate that the factory processing has time sequential properties, but that the sequencing is nonconstant or not specified. This concept is easily related with two opposite arrow heads on the line containing the exclusive OR. Note that the arrow heads do not conform to the E-R modeling tradition that arrows point from left to right (or top to bottom). Arrow heads indicate a sequential property, but the opposing directions indicate that the sequence is nonconstant or not specified. *Fig. 8* illustrates a typical random sequence situation. Note that representations of the number (of consecutive steps per location in the sequence – shaded/unshaded arrow head) as well as *optionality* must be illustrated in *all* arrows that point to an LC. In addition, all arrows that point to a particular LC must indicate the *same* number constraint and this constraint must indicate the widest range of the *composite* of all possible LC to LC scenarios, (i.e. optional takes precedence over nonoptional and 'many' takes precedence over 'one'). This is because the number of consecutive steps per location indicated must pertain to and allow all possible

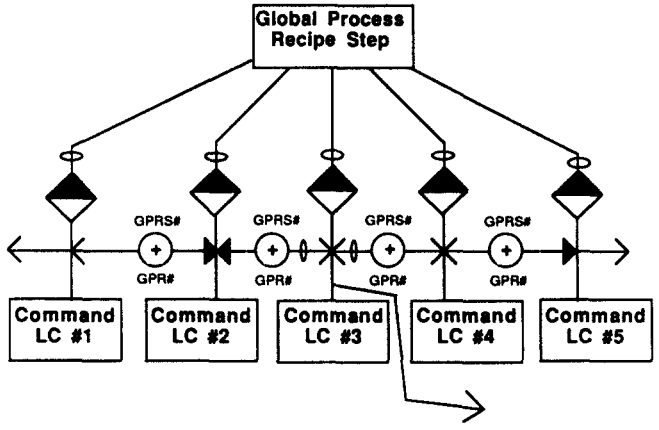


Fig. 8. EE-R model: illustration of the logical random sequential ordered exclusion function.

LC to LC scenarios; therefore the constraints implied by the model must be at least as lax as the most lax constraint of all scenarios. Note also that no starting point or first LC in the processing sequence is specified (indicating that it is random), but multiple ending points are specified. In general, zero, single or multiple starting or ending points may be specified.

The assembly line schema (Figs 3 through 7) is an example of a subset (a more restricted case) of the random sequence schema. Note that the random sequence schema would rarely be used to completely describe a manufacturing environment since a necessary condition on processing in such an environment is a knowledge of the specific sequence of processing. However, the random sequence schema is useful in conceptualizing process flow in a manufacturing environment where the order of processing is not (yet) known.

(2) *Multiple but specificable sequences*; Depending on the product, there is an identifiable subset of LC-to-LC process sequences. Here multiple arrows are used to illustrate the various possible sequences. *Fig. 9* provides an example. Note that representations of the number of consecutive steps at a point in the sequence (shaded/unshaded arrow head) as well as optionality must be illustrated in *all* arrows pointing to an LC. Arrows pointing to the same LC do *not* have to indicate the same number of consecutive steps to be executed (the number may be a function of the previous LC in the process). Note also in this example that a single starting LC and ending LC is specified. There may be multiple starting and/or ending points.

(3) *Processing loops*: A subset of LC-to-LC sequence may occur repetitively (in a loop) a specified or unspecified number of times. A loop is represented as an arrow line with an embedded exclusive OR. Terminology at the beginning of the arrow line (between the originating LC and the exclusive OR symbol) indicates knowledge (where applicable) of the number of times the loop may be traversed. As an example, in *Fig. 9* a loop exists in the processing sequence from LC #5 to LC #4 that is to be traversed twice in a particular process.

(4) *Conditional sequencing*: An LC-to-LC sequence transition may be the result of an event (e.g. an error). This event can be indicated with terminology inserted at the beginning of the arrow line (between the source and the exclusive OR symbol) of the branch. (This construct would prove valuable to model situations where feed forward and feedback are employed.) *Fig. 10* contains an example.

(5) *Hybrid*: A combination of the above situations may occur in the factory.

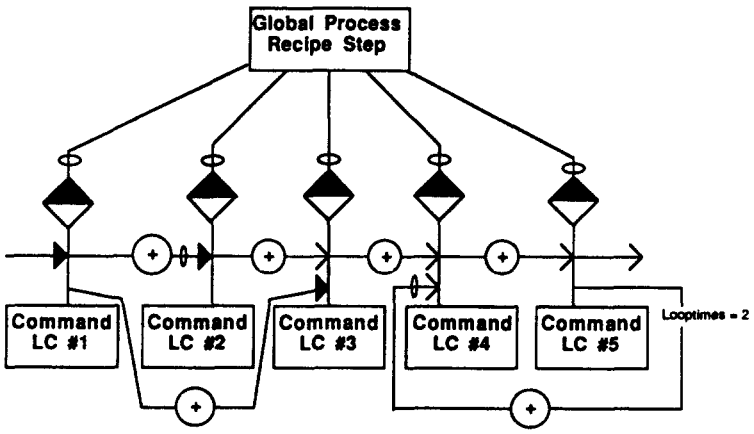


Fig. 9. EE-R model: illustration of multiple but specific ordering in a logical sequential ordered exclusion function.

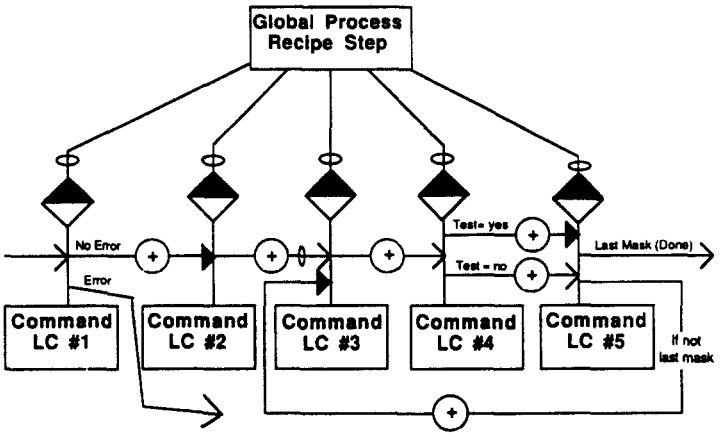


Fig. 10. EE-R model: illustration of conditional sequencing in a logical sequential ordered exclusion function.

2.3. E-R extensions to represent time sequencing and ordering properties in a hybrid, hierarchical production environment

The above E-R extensions may be incorporated to construct a better model of the automated facility. The extensions may be used in modeling relations between the factory computer and local controllers (LCs) as previous examples in this paper have illustrated. The extensions, however, could also be applied to satisfactorily model the various lower levels of the process hierarchy. Therefore it can be concluded that sufficient extensions to the E-R model have been developed to better characterize a database system in the factory of the future.

As an example consider the automated semiconductor manufacturing facility. This facility type in general exhibits 'logical ordering' of processes. The product of the semiconductor manufacturing facility is the integrated circuit (IC) 'chip'. Chips are produced in batch form on semiconductor wafers. In the facility these wafers are subjected to many processes during production [3]. Depending on the desired characteristics of the product being manufactured, the sequence of processing with respect to the local controllers may vary. A simplified hierarchical EE-R model of a possible database system for recipe processing in an automated semiconductor manufacturing facility is given in *Fig. 11*. Note that at the highest level of the hierarchy, the process is physically (and of course logically) sequential and thus resembles that of the physical assembly line. At lower levels of the hierarchy, physical as well as logical sequential, multiple specifiable sequential, loops, and conditional sequencing concepts are all evident. The primary contribution that has been made and documented in this paper is that techniques now exist to model and differentiate these various time sequencing and ordering concepts and thus modeling constructs now exist to develop a more complete model of the factory of the future.

3. Formal definition of new extensions to the Entity-Relationship model

The proposed extended entity-relationship (EE-R) constructs are formally defined in this section in a nonapplication specific manner, i.e. the definitions are not restricted to the automated facility. With each syntactic definition, a schematic representation assigned to the construct is given and applications of and possible variations to the extension are discussed. The interpretation of these extensions to database implementation is then presented.

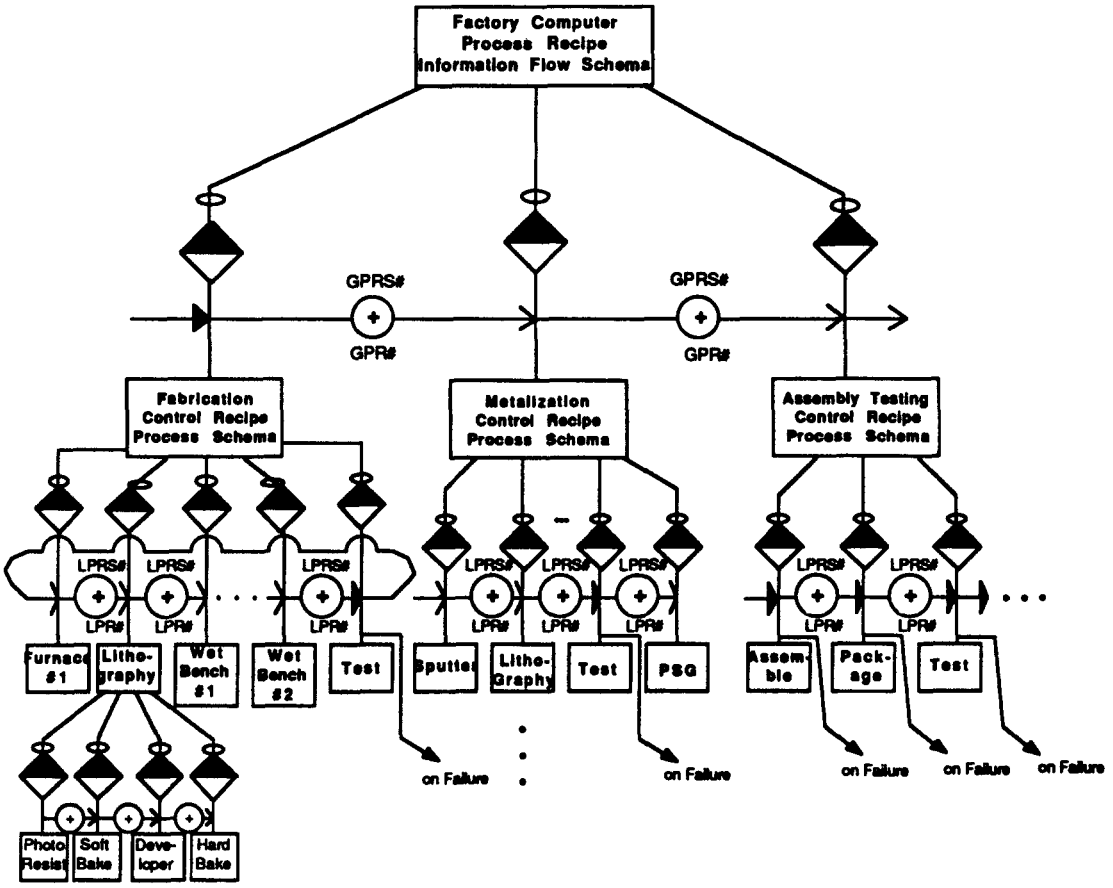


Fig. 11. An example EE-R model for process recipe information flow in an automated semiconductor manufacturing facility.

Note: The above schematic is an abbreviated and simplified representation of an example automated semiconductor manufacturing facility. ER model clustering techniques are applied at each controller modeled in this facility, thus this model serves to illustrate the controller to controller interaction over the various networks in the facility in managing process recipe information flow [13].

3.1. The Sequential Ordered Exclusion (SOE) function

The SOE function is used to model subschemas to which the exclusion E-R extension apply [13] and an ordering of tuples of the parent entity is indicated (i.e. a sequence of information flow is indicated between children of the subschema). The ordering in the SOE function is based on an ordering attribute set, OAS, which is a subset of attributes of the parent entity. Ordering occurs within partition subsets of records of that parent entity. These partition subsets are determined by each unique occurrence of a partitioning attribute set, PAS, also a subset of attributes of the parent entity. The attribute subset of the parent entity that determines ordering, OAS, is indicated *above* the exclusion function. The attribute subset of the parent entity that partitions the parent entity into subsets over which ordering is to occur, PAS, is placed *below* the exclusion function. *Fig. 12* provides an illustration of the SOE function.

A more precise definition of the attribute sets PAS and OAS may be realized. Both PAS and OAS are subsets of the attributes of the parent entity. The parent entity may be defined in terms of its candidate key attribute sets and nonkey attributes in the following manner.

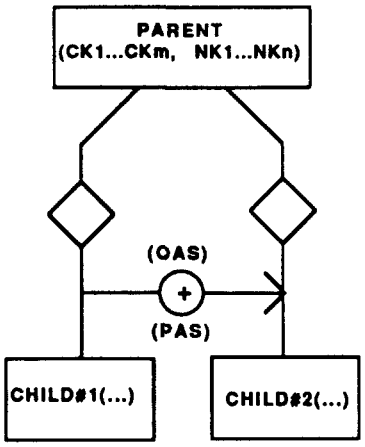


Fig. 12. Example of the sequential ordered exclusion function: the strong sequential ordered exclusion function.

$$\text{Parent}\ (\text{CK}_1, \text{CK}_2, \ldots, \text{CK}_m, \ \text{NK}_1, \ldots, \text{NK}_n) \tag{1}$$

where: $\text{CK}_i = \{\text{CK}(i, 1), \ \text{CK}(i, 2), \ldots, \text{CK}(i, ki)\} \Rightarrow$ A Candidate Key attribute set
$\text{CK}(x, y)$ = The yth attribute of the xth candidate key
ki = The number of attributes comprising candidate key i
m = The total number of candidate key attribute sets
$\text{NK}_j \Rightarrow$ Non-Key attribute
n = The total number of non-key attributes.

The partitioning attribute set PAS determines a partitioning of the records of the parent entity into subsets that each have a unique value of PAS attributes. For nontrivial ordering, PAS must determine subsets with an element count of greater than one for each subset. Therefore PAS may contain any of the attributes of the parent entity so long as it does not contain a complete candidate key of the parent entity (i.e. PAS may contain any of the attributes of the parent entity as long as there does not exist a subset of these attributes that comprises a candidate key of the parent entity). Note that PAS may be null which is to be interpreted that the parent entity is not partitioned, i.e. the parent entity is taken as a whole.

The ordering attribute set OAS determines the ordering of records within each subset (partition) determined by PAS. For nontrivial ordering, these subsets each have an element count greater than one. OAS must specify a unique ordering within each subset. Therefore there must exist a subset of attributes comprising the set union of OAS and PAS that is a candidate key of the parent entity. Note that OAS may contain many candidate keys and non-key attributes of the parent relation. Further note that it can also be shown that, to achieve any particular ordering of partitions of the parent entity, there exists a minimal set of attributes for PAS and OAS such that PAS and OAS are disjoint. *Fig. 13* provides a graphic illustration showing how a parent entity may be partitioned by PAS and ordered by OAS.

As an example, consider the assembly-line like process described in Section 2.1 where the processing sequence is 'order by recipe step within a recipe'. The entity to be partitioned and ordered in GPRS where the attributes of GPRS are:

GPRS(*GPR#*, *GPRS#*, . . . , Local Controller#, . . . , recipe parameters)

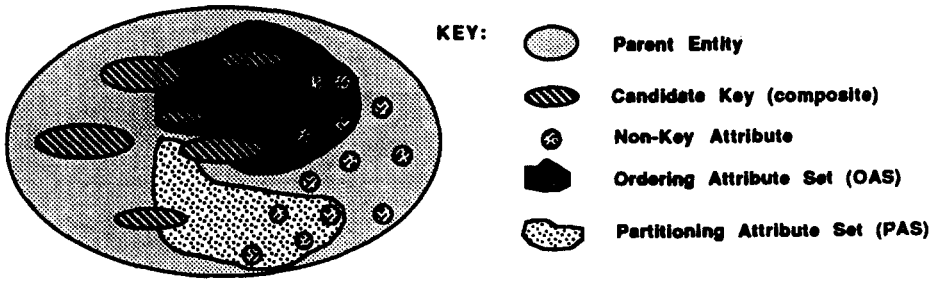


Fig. 13. Partitioning of parent entity attribute set by PAS and ordering by OAS.

(See Moyne et al. for a further description of GPRS [9]. Note that (GPR#, GPRS#) is a composite candidate key of GPRS.) Each partition of GPRS is determined by a unique value of the attribute GPR# among the tuples of GPRS, while ordering the tuples within these partitions is determined by values of the attribute GPRS#. Thus for this example:

PAS = GPR#
OAS = GPRS#

Note that the set union of PAS and OAS contain a candidate key (GPR#, GPRS#) of GPRS. Therefore OAS specifies a unique ordering within each subset partition determined by PAS. *Fig. 14* (*A* through *C*) provides an illustration of how PAS and OAS are applied to the entity GPRS to achieve the desired ordering of the records of that entity.

There are two basic types of SOE functions reflecting the degree of ordering of information. The first subtype is referred to as the *Strong Sequential Ordered Exclusion* (*SSOE*) *Function*. The SSOE function is used to model subschemas to which the exclusion E-R extension apply and a specific sequence of information flow is indicated between the children of the subschema. For instance, in the example of *Fig. 4*, information flows in a strict assembly line sequence. (*Figs 5* through *7* also contain illustrations of the SSOE function.) The SSOE function is schematically represented as shown in *Fig. 12*. The direction of the time sequencing (ordering) is indicated with an arrow head placed on the line through the exclusion function. This line with arrow head and exclusive OR symbol is also referred to as a *path*. The arrows points in the direction of increasing time and thus indicates an ordering of records of the parent relation that can be interpreted to an execution sequence. Section 2.1 contains an example application that uses the SSOE function.

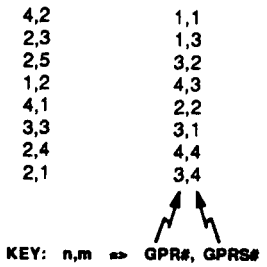


Fig. 14a. GPRS before partitioning.

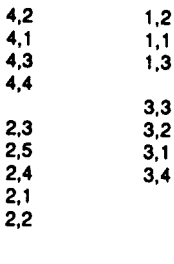


Fig. 14b. GPRS after partitioning according to PAS (=GPR#) but before ordering.

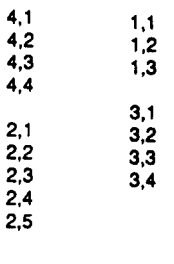


Fig. 14c. GPRS after partitioning according to PAS and ordering according to OAS(=GPRS#).

Fig. 14. Partitioning of GPRS for ordering constraint: 'order by recipe step within each process recipe'.

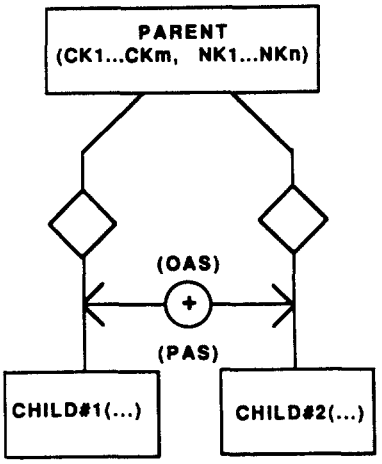


Fig. 15. The weak sequential ordered exclusion function.

The second subtype of SOE functions is referred to as the *Weak Sequential Ordered Exclusion (WSOE) Function*. The WSOE function is also used to model a subset of those subschemas to which the exclusion E-R extension applies. The basis of determination of the ordering scheme is the same as for the SSOE. The WSOE, however, is used to model (usually abstract) random process sequencing situations or situations where details of process flow are not (yet) known. The WSOE function is schematically represented as shown in *Fig. 15*. An arrow head is placed at each end of the line through the exclusion function, and the arrow heads point away from each other. Therefore, if a subschema is as described in Section 3.1 with the exception that there is weak time sequential ordering (by OAS) on subset partitions of the parent record set determined by PAS, then the EE-R model for this subschema is as shown in *Fig. 15*. Note that the WSOE is used to describe an abstract or somewhat vague situation and may be replaced in the modeling process by an SSOE (or other construct) as more details of the process become known. This modeling exercise illustrates the use of EE-R modeling as a design tool.

3.2. Representation of Number of Consecutive Ordered Records in the Sequential Ordered Exclusion (SOE) Function

Extensions to the SOE function are needed to indicate the number of consecutive ordered records of a parent entity involved with a child entity for ordered subsets of records of the parent entity. Each of these ordered subsets of records of the parent entity is determined by unique occurrences of PAS within records of the parent entity. As an example, if a subschema is as described in Section 3.1 with (strong or weak) sequential ordering on subsets of the parent record set determined by PAS, then the number of consecutive ordered records of a parent entity involved with a particular child c is in the range a to b where:

$a = \min \mid \mathrm{OAS}_c \mid_{\mathrm{PAS}}$, i.e. the minimum number, over all record subsets determined by PAS, of consecutive ordered records within that subset pertaining (related) to child c.

$b = \max \mid \mathrm{OAS}_c \mid_{\mathrm{PAS}}$, i.e. the maximum number over all record subsets determined by PAS, of consecutive ordered records within that subset pertaining (related) to child c,

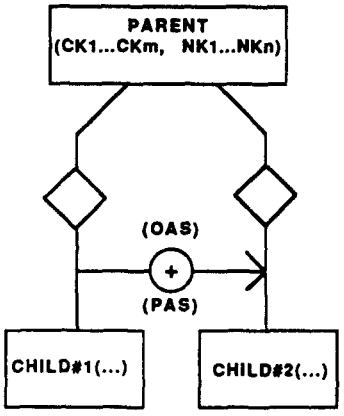


Fig. 16. Representation of number* = 1 in SOE functions.

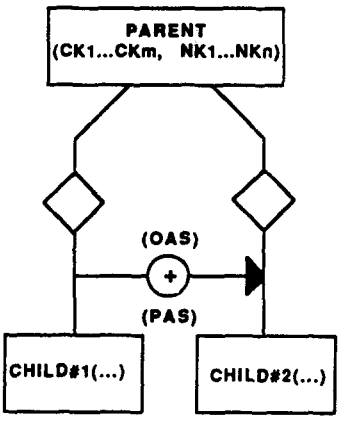


Fig. 17. Representation of number* = many in SOE functions.

where
$|OAS_c|_{PAS}$ is the number '$|\ldots|$' of consecutive ordered records (tuples) of a partition of the parent entity (a partition determined by PAS and ordered by OAS) that pertain to (are implemented at) child c.

A maximum number (of consecutively ordered records of the parent entity) of exactly one for a child c is indicated with the sequential ordering function arrow head symbol (pointing to the branch for child c) left open as in *Fig. 16*. A maximum number of 'many' for the child c is indicated with the same arrow head shaded as in *Fig. 17*. A minimum number of one is implied in both cases. A minimum number of zero (optionality) may be indicated with an 'optional' ring symbol placed over the exclusion function line between the arrow head corresponding to child c and the exclusive OR symbol (as shown earlier in *Fig. 7*). The number of consecutively ordered records of the parent entity pertaining to child c may be further specified with terminology placed between the arrow head corresponding to child c and the exclusive OR symbol (as shown earlier in *Fig. 6*).

3.3. Multiple sequence path extensions to the sequential ordered exclusion function

The SOE function may be extended conceptually to accommodate schemas having multiple path, time sequences (where a *path* is the EE-R arrow containing the embedded exclusive OR symbol that conceptually represents child-to-child information ordering/flow). The above constructs for the SOE functionality and representation may be applied recursively to the various path time sequences. However, care should be taken to make sure that the constructs representing the various paths do not conflict. The appearance of conflicts would indicate a poor data model.

With multiple path situations, conditions relating to the decision of which path to take in a sequence (i.e. conditional sequencing) may be indicated with terminology placed over the exclusion function line between a line end *not* containing an arrow and the exclusive OR symbol, i.e. the terminology should indicate conditions for *leaving* on that path. *Fig. 10* (Section 2.2) provides three examples.

* Where 'number' refers to the number of consecutively ordered records of the parent entity involved with the child entity through the relation pointed to by the arrow.

3.4. *Starting and Ending Point(s) in the Sequential Ordered Exclusion Function*

In subschemas containing ordered time sequences, starting points for these sequences are indicated with an arrow pointing to each starting point (the arrow has no originating point). Representations may be applied to this arrow that indicate conditions to take a path as well as the number of consecutively ordered records of the parent entity pertaining to the indicated child. Ending points for these sequences are indicated with an arrow pointing away from each ending point (the arrow has no destination). Representations of conditions to take the path may be applied to this arrow. Representations of 'number' (i.e. the number of consecutively ordered records of the parent entity pertaining to a particular child) may not be applied to this arrow. Examples of this construct may be found in *Figs 6* through *11*.

3.5. *Hybrid applications of Sequential Ordered Extensions*

The extended E-R constructs may be applied together in a hybrid system to obtain a better model of the system. Care should be taken to avoid any conflicts in functionality implied by the constructs. The following is a list of possible conflicts. It should be noted that this list is not exhaustive, but gives an indication of areas of possible conflict.

1. If a subschema is indicated as containing an exclusion construct with weak ordering among the children, care should be taken to make sure that the number of consecutively ordered records of the parent entity indicated as pertaining to a child entity (indicated by the form of arrow head and ring around the exclusion line) is consistent among the two arrows pointing at that child. This is necessary because, with weak ordering, no specific path is implied; therefore the number specified at a node must encompass all possible information flow paths.
2. If conditions are listed on branches leaving a node, these conditions between branches must be mutually exclusive, i.e. there should be no ambiguity as to which branch to take following an event at a node.
3. Any node that has an incoming path should have at least one outgoing path. Similarly, any node that has an outgoing path should have at least one incoming path.

3.6. *Interpretation of Time Sequential Ordered Extensions: Database implementation*

A data model using existing E-R constructs can convey a great deal of information about a database. For example the E-R data model of the hierarchical automated facility (*Fig. 1*) indicates the database entities and their interrelationships. Further, the Exclusive OR, optionality and ID dependence constructs are interpreted into integrity constraints in the actual database [9, 13].

The existing E-R model, however, does not contain the construct base to convey the concept of record ordering in a database implementation – a property crucial to databases implemented in arenas such as facility automation. The time sequence ordering extensions to the E-R model developed in this paper convey information on the ordering of entity occurrences (i.e. records within a relation) and rules for ordering. In an actual implementation, relation ordering is usually accomplished through an indexing on pertinent attributes in a relation or through a view created using a data manipulation language statement (such as Structured Query Language (SQL)) that specifies a record ordering. (Note that procedural information has been encoded into the database as the database record ordering implies a process sequence. This lessens the burden on application code.)

Thus the use of a time sequence ordering E-R extension in a database model would imply that the appropriate index(es) be used in the actual database or the appropriate ordering

syntax be included in the data manipulation language statement applied to the view. As an example consider the assembly line database model illustrated in *Fig. 4*. Here GPRS#'s are to be ordered consecutively within each GPR#. This may be accomplished with an index placed on GPRS# for each GPP# in the relation GPRS. Alternatively, the following view may be created that indicates record ordering:

Select GPRS#, . . . , LC#, Command, . . .
From GPRS
Where GPR# = 'XX'
Order by GPRS#

Note that this view indicates the order of processing for GPR# 'XX'.

Representations of 'number' as well as other syntax in the time sequence ordering E-R extensions also translate to the database implementation, but in the form of integrity constraints. As an example consider the assembly line depicted in *Fig. 7*. The following are a few of the integrity constraints implied by the model:

1. Within each set of records with a specific GPR#, the record corresponding to the first and only the first GPRS# contains the foreign key value corresponding to the key of relation Command at LC#1.
2. Within each set of records with a specific GPR#, one or more records with consecutive GPRS#'s contain the foreign key value corresponding to the key of relation Command at LC#2.
3. Within each set of records with a specific GPR#, if a record contains the foreign key value corresponding to the key of relation Command at LC#5, it is the record that contains the highest GPRS# for that GPR# (i.e. it is the last record in that sequence).

In addition to indicating properties of the database being modeled, time sequence ordering extensions to the E-R model also play another important role in that they illustrate data information flow and thus provide a pictorial insight into the process being modeled.

4. Summary and conclusions

Time sequencing and ordering extensions to the entity-relationship model have been proposed and formal definitions provided. The motivation for these extensions has been revealed in recent literature though an analysis of an E-R data model of the automated manufacturing facility. Research presented in this paper has built on that analysis and one result is the development of such extensions. The model extensions have been developed through case studies of process recipe information flow in various types of automated facilities.

The interpretation of these new modeling constructs to database implementation has been defined. Also, it has been indicated that the newly developed constructs have no impact on the existing formalism of the E-R model. The expanded set of modeling constructs have been applied to automated manufacturing, resulting in a more complete data model of process recipe information flow in the automated manufacturing facility. A comparison of *Figs 2* and *11* provides an indication of the advancement available using the expanded set of E-R constructs to model the automated semiconductor manufacturing factory of the future.

As a result of the effort described above, the E-R modeling tool set has been rendered more complete and versatile. The use of the E-R modeling technique has been expanded to

incorporate control information, softening the boundary between databases and procedural code. Further E-R modeling has been used as a process design tool. The combination of research in factory automation and database modeling and design has led to benefits to both areas and will impact on many potential areas of current and future research. These areas include the development and standardization of communication methodologies over the various networks in the automated facilities, the development and standardization of recipe process flow languages [2], the further development of design methodologies for databases [14], the specification of distributed and object oriented database systems in automated manufacturing [12], and, in general, modeling of the many aspects of the automated facility.

Acknowledgements

The authors gratefully acknowledge the technical and editing contributions of Ms. Elizabeth Armstrong. The authors also extend their gratitude to the reviewers for their helpful and in-depth comments on the manuscript.

Appendix A: Overview of Entity-Relationship (E-R) theory and constructs

The E-R approach to database modeling was initially proposed by Chen [1] and provides semantics for conceptual design of databases. Extensions have since been proposed to the model [13, 14, 15]; however the underlying approach remains the same.

With the E-R approach, information is represented in terms of entities, their attributes, and relationships between entities, where following definitions apply. The modeling semantics corresponding to each definition are illustrated in *Fig. 18.*

Entity: A principal object about which information is collected. For example, in a database containing information about personnel of a company, an entity might be 'Employee'. In E-R modeling an entity is represented with a box.

Attribute: A label that gives a descriptive property to an entity, e.g. name, color. Two

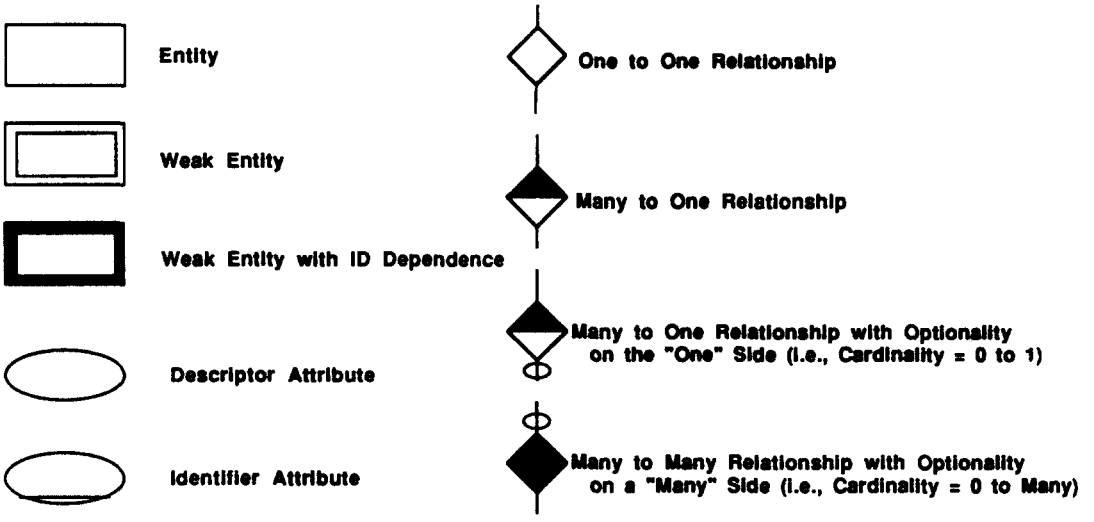


Fig. 18. Entity-Relationship modeling semantics.

types of attributes exist. Identifier attributes distinguish among occurrences of an entity, e.g. social security number. These attributes may also be referred to as candidate keys. Descriptor attributes merely define an entity occurrence, e.g. gender, weight. In E-R modeling an attribute is represented with an oval tied to the entity (box) to which it pertains. In many cases, attributes are not included in the E-R model.

Relationship: A relationship is a connectivity exhibited between entity occurrences. Relationships may be one to one, one to many, and many to many, and participation in a relationship by an entity may be optional or mandatory. For example, in the database containing information about personnel of a company, a relation 'married to' among Employee entity occurrences is one to one (if it is stated that an employee has at most one spouse). Further, participation in the relation is optional as there may exist unmarried employees. As a second example, if company policy dictates that every employee have exactly one manager, then the relationship 'managed by' among Employee entity occurrences is many to one (many employees may have the same manager), and mandatory (every employee must have a manager). In E-R modeling a relationship is represented with a diamond if it relates one or two entities, and is represented with an n-sided polygon is it relates n entities (where n is greater than two). Connectivity in a relationship is denoted with shading of the diamond; a connectivity of 'one' is denoted with the appropriate portion of the diamond unshaded while a connectivity of 'many' is denoted with the appropriate portion of the diamond unshaded. Optionality of entity participation in a relationship is indicated with a ring around the line segment between the entity and the relationship.

Entities may be one of two types. Strong entities have internal identifiers that uniquely determine an entity occurrence. Weak entities derive their existence from the identifying attribute of another entity referred to as the parent entity, i.e. a weak entity exhibits existence dependence with a parent entity. As an example, the employee entity referred to above is a strong entity. An entity in the company database, an entity called 'Active Project' would be a weak entity if a project is deemed active only if it has at least one employee working on it. The parent entity of Active Project is Employee. A weak entity may further be referred to as ID dependent if the entity also derives its uniqueness from a parent entity. The Active Project entity above is not ID dependent as a project is not uniquely identified by the employee working on it. Note that all entities that exhibit ID dependence also exhibit existence dependence, while the converse is not necessarily true.

A detailed description of E-R model semantics, extensions and E-R modeling techniques may be found in Teorey [15].

Listing of acronyms

CK	Candidate Key
E-R	Entity-Relationship
EE-R	Extended Entity-Relationship
GPR	Global Process Recipe
GPRS	Global Process Recipe Step
LC	Local Controller
LPR	Local Process Recipe
LPRS	Local Process Recipe Step
NK	Non-Key
OAS	Ordering Attribute Set

PAS	Partitioning Attribute Set
SLPR	Sub-Local Process Recipe
SLPRS	Sub-Local Process Recipe Step
SOE	Sequential Ordered Exclusion
SQL	Structured Query Language
SSOE	Strong Sequential Ordered Exclusion
WSOE	Weak Sequential Ordered Exclusion

References

[1] P.P. Chen, The Entity-Relationship model – toward a unified view of data, *ACM Trans. Database Systems* 1(1) (March 1976) 9–36.

[2] DARPA/SRC CIM Workshop, Session on Process Specification, Stanford University, California, August 1988.

[3] P.R. Gray and R.G. Meyer, *Analysis and Design of Analog Integrated Circuits* (John Wiley, New York, 1977).

[4] J. Grenier, The Mitsubishi Saijo factory: a new and fully automated IC facility, *Solid State Technol.* 29 (1) (January 1986) 47–48.

[5] B. Johnston, Process control in wafer fabrication, *Solid State Technol.* 29 (5) (May 1986) 195–199.

[6] L. McAfee, R. Atallah and J. Moyne, A MAP network for automated semiconductor manufacturing, *Technical Proc. SEMICON/East 87* (1988) 203–213.

[7] L. McAfee, J. Moyne and R. Atallah, Network consideration for automated semiconductor manufacturing, *Gateway: The MAP/TOP Reporter* (July/August 1988) 13–16.

[8] J. Moyne, J. Birchak, L. McAfee and F. Brehm, A MAP-SECS network for automated semiconductor manufacturing, *Gateway: The MAP/TOP Reporter* (May/June 1988) 20–23.

[9] J. Moyne, L. McAfee and T. Teorey, An application of Entity-Relationship modeling techniques to the automated manufacturing process. *Proc. Second Internat. Conf. on Data and Knowledge Systems for Manufacturing and Engineering (IEEE/ACM)* (October 1989) 206–215.

[10] H.V. Parunak and J.F. White, A synthesis of factory reference models, *IEEE 1987 Workshop on Languages for Automation* (August 1987) 109–112.

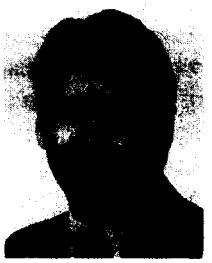
[11] S. Shinoda, Total automation in wafer fabrication, *Semiconductor Internat.* (September 1986).

[12] J. Stein and D. Maier, Concepts in object-oriented data management, *Database Programming Design* (April 1988) 58–67.

[13] T.J. Teorey, G. Wei, D.L. Bolton and J.A. Koenig, ER model clustering as a communication and documentation tool, *CACM* 32 (8) (August 1989) 915–987.

[14] T.J. Teorey, D. Yang and J.P. Fry, A logical design methodology for relational databases using the extended Entity-Relationship model, *ACM Comput. Surveys* 18 (2) (June 1986) 197–222.

[15] T.J. Teorey, *Database Modeling and Design: The Entity-Relationship Approach* (Morgan Kaufmann, San Mateo, CA, 1990).



James Moyne received his B.S.E.E. and B.S.E.-Math degrees from The University of Michigan, Dearborn in 1983, and his M.S.E.E. and Ph.D. degrees from The University of Michigan, Ann Arbor in 1984 and 1990 respectively. His dissertation title is *System Design for Automation in Semiconductor Manufacturing*. From 1987 to 1990 he served on the Communications Committee of Semiconductor Equipment and Materials International where he was the chief technical designer of the SECS Message Service, which has been internationally adopted as a standard for semiconductor equipment communications. He is currently a post-doctoral research fellow in the Department of Electrical Engineering and Computer Science at The University of Michigan, Ann Arbor. His research topics include database design and implementation, manufacturing automation, networking, and cell controller design. James has a background in local area networks, discrete-event simulation, solid-state circuits, mathematics, and databases. His internet address is moyne@um.cc.umich.edu at The University of Michigan.

Toby J. Teorey received the B.S. (1964) and M.S. (1965) degrees in Electrical Engineering from the University of Arizona, Tucson, and a PhD in Computer Science (1972) from the University of Wisconsin, Madison. He served as an EDP Officer in the US Air Force from 1965 to 1969 and was a White House Social Aide for President Lyndon Johnson from 1966 to 1968. At the University of Wisconsin (1969 to 1972) he became the coordinator of the Operations Research Group in the Madison Academic Computing Center. He is currently Associate Professor of Electrical Engineering and Computer Science at the University of Michigan and a senior staff member of both the Software Systems Research Laboratory (SSRL) and the Center for Information Technology Integration (CITI). At CITI he directed the development of NetMod, a HyperCard-based tool for local area network design and

analysis. Professor Teorey's most recent book *Database Modeling and Design: The Entity-Relationship Approach* (Morgan Kaufmann) was published in 1990. His current interests include distributed database and distributed directory design, and the modeling and analysis of large interconnected local area networks. His internet address of teorey@ub.cc.umich.edu at the University of Michigan.

Leo C. McAfee, Jr., was born in Marshall, TX, on December 15, 1945. He received the B.S. degree from Prairie View A & M University, Praire View, TX, in 1966 and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1967 and 1970, respectively, all in electrical engineering.

He is presently an Associate Professor in the Department of Electrical Engineering and Computer Science at The University of Michigan, Ann Arbor. His present research interests focus on techniques for automated semiconductor manufacturing. His research activities include local area networks and evaluation of network performance, automation in computer integrated manufacturing, and simulation of local area networks within semiconductor manufacturing facilities. He has extensive prior research experience in modeling, simulation, analysis, optimization, and numerical analysis for solid-state circuits and devices.

Dr. McAfee is a member of Eta Kappa Nu, Tau Beta Pi, Sigma Xi, Phi Kappa Phi, and Alpha Kappa Mu.