CHAPTER

DATA MINING FOR PROCESS AND QUALITY CONTROL IN SEMICONDUCTOR INDUSTRY

Mark Last
mlast@bgumail.bgu.ac.il
Department of Information Systems Engineering, Ben-Gurion University, Beer-Sheva 84105, Israel

Abraham Kandel
kandel@csee.usf.edu
Department of Computer Science and Engineering, University of South Florida , 4202 E. Fowler Avenue, ENB 118, Tampa, FL 33620 USA

ABSTRACT

Like in any other industry, manufacturing departments of semiconductor plants are evaluated by their ability to meet the delivery schedules. However, the final quantities and the flow times of individual semiconductor batches are affected by multiple uncertain factors, like material quality, process variability, equipment condition, and others. Thus, the tasks of predicting the batch quality (measured by yield) and its total flow time are an important part of the production planning activities. Beyond prediction, the plant management is interested in identifying the main causes of yield excursion and process delays.

In this paper, we are applying several methods of data mining and knowledge discovery to WIP (Work-in-Process) data, collected in a semiconductor plant. The information on each manufacturing batch includes its design parameters, process tracking data, line yield, etc. The data is prepared for data mining by converting a sequential dataset into a relational format. Classification models for predicting line yield and flow times are built from the pre-processed data by using the Info-Fuzzy Network (IFN) methodology. Fuzzy-based techniques of automated perception are used for post-processing the data mining results. We conclude the paper with a critical evaluation of the discovered knowledge and the methods used.

Keywords: data mining, knowledge discovery, info-fuzzy network, automated perceptions, yield management.

INTRODUCTION

The outgoing *yield* of manufactured batches is the basic measure of profitability in semiconductor industry. Overall, or line yield of the manufacturing process is defined as the ratio between the number of good parts (microelectronic chips) in a completed batch and the maximum number of parts, which can be obtained from the same batch. Since capitalization costs constitute the major part of manufacturing costs in semiconductor industry, the *direct cost* of producing a single batch is almost fixed. However, the *income* from a given batch is proportional to the number of good chips. Thus, there is a direct relationship between the yield and the profits of semiconductor companies, which usually treat the yield performance as one of their top commercial secrets.

The expected yield of every batch is also an important parameter for the production planning and control. An "optimistic" estimate of the outgoing yield may cause delays in the order delivery due to insufficient quantities produced. On the other hand, "overbooking" in the number of batches designated for a given order may lead to a waste of precious resources, like technicians, machines, and electricity. Unnecessary batches may also cause delays in the production of other, more critical batches. Consequently, this is the primary interest of the planning personnel to have an accurate prediction of the actual yield.

The problem of yield prediction is closely related to another problem of planning the supply of orders, namely the problem of predicting the *flow times* of individual batches. Though the net time of each production step in semiconductor industry is nearly fixed, the waiting times between the operations are very hard to predict, due to such variable factors as equipment downtimes, process failures, and line congestions.

Controlling and preserving the yield is a complex engineering problem. Both new and mature semiconductor products suffer from variability of yield within and between individual batches and even on specific wafers of the same batch. Improved understanding of this variability can save significant manufacturing costs by focusing on problematic processes and taking appropriate actions, whenever excursion of yield is expected for a given batch, wafer, etc.

Although the amount of manufacturing data collected by semiconductor companies is constantly increasing, it is still hard to identify the most important parameters required for yield modeling and prediction. As indicated by Tobin et al. (2000), the amount of data generated is exceeding the yield engineer's ability to effectively monitor and correct unexpected trends and excursions. Consequently, there is a strong need for automated yield management systems, which will be able to explain and predict yield excursions by using sophisticated data management and data mining tools.

The use of standard data mining methods for the yield and the flow time prediction in semiconductor industry faces several difficulties. First, such common methods as association rules (see Agrawal et al., 1996) and decision trees (see Quinlan, 1993) are aimed at analyzing structured data, organized in two-dimensional tables, where all the variables relevant to the target are located in a single row (record) of one table. In a real-world situation, the factors affecting the final yield of a manufacturing batch are spread across many data tables, which may be even stored in separate database systems. Moreover, multiple records of parametric data, collected in the course of the manufacturing process, may be related to the outcome of a single batch. Important information may be lost if the timing of those records is ignored by the data mining tools.

Second, the basic assumption of most data mining methods is that all the data stored in a database is complete and correct. The actual semiconductor data may not comply with the assumption of data correctness for several reasons including erroneous data entry, inaccurate measurements, etc. Since it is not practical to ensure the 100% data quality, the methods used should be robust to the presence of errors in the mined data. Another common problem of data quality is the absence of values for certain attributes due to either incomplete data entry process or non-existence of the values themselves (e.g., missing measurements of a scraped wafer). In both cases, a data mining method should deal with a missing value without ignoring the values of the other attributes in the same record.

Finally, the abundance and the diversity of automatically collected data pose another potential problem to data mining. Most of the available attributes may be completely irrelevant to the target (e.g., the final yield). Since most data mining methods (especially those based on the Bayesian approach) are trying to incorporate as many attributes as possible in the model, the learning process may be misled by the irrelevant attributes, resulting in over complex and inaccurate representations of discovered knowledge. The continuous nature of the target (dependent) variables, like the yield and the flow time, precludes them from being used directly by *classification* techniques (e.g., decision trees and Bayesian methods), which assume the target to take a limited number of distinct values.

The process of knowledge discovery in semiconductor databases, presented in this paper, is based on a novel method for mining real-world data, termed IFN for Info-Fuzzy Network (see Maimon and Last, 2000). The method builds upon the principles of Shannon's information theory (see Cover, 1991) and fuzzy logic (see Klir and Yuan, 1995). It is applicable to databases of mixed nature containing quantitative (continuous), qualitative (nominal), and binary-valued attributes. The relevant features are selected automatically in the process of constructing the prediction model (IFN). Ranked association rules between the selected factors and the target attribute

(e.g., final yield) can be extracted from the network structure. The network output can also be used for evaluating reliability of target data. In (Maimon and Last, 2000), the IFN method is shown to have several benefits, including: built-in feature selection, compact and interpretable representation of extracted knowledge, reasonable predictive accuracy (compared to other methods), stability of obtained models, and robustness to noisy and incomplete data.

Section 2 of this Chapter describes the information-theoretic fuzzy approach to knowledge discovery in databases. The process of rule fuzzification and reduction is presented in Section 3. In Section 4, we proceed with a detailed case study of knowledge discovery in manufacturing data. The Chapter is concluded by Section 5, which evaluates the potential of the information-fuzzy approach for data mining in semiconductor industry.

THE IFN METHODOLOGY FOR KNOWLEDGE DISCOVERY AND DATA MINING

The *Info-Fuzzy Network* (IFN) methodology, presented by us in (Maimon and Last, 2000), is a novel and unified approach to automating the process of *Knowledge Discovery in Databases* (KDD). The main stages of the KDD process handled by IFN include discretization of continuous attributes, feature selection, prediction and classification, extraction of association rules, and data cleaning. The method is aimed at maximizing the *mutual information* (see Cover 1991) between input (predicting) and target (dependent) attributes. The following sub-sections describe the data model used by IFN, the general structure of an info-fuzzy network, the network construction algorithm, and the procedure for extracting association rules from the network structure.

The Data Model

The IFN method distinguishes between the following types of attributes in a database:

1) O - a subset of *target* ("output") attributes ($O \subset R$, $|O| \geq 1$). The information-theoretic network is constructed to predict the values of target attributes, based on the values of *input* attributes (see below).
2) C - a subset of *candidate input* attributes ($C \subset R$, $|C| \geq 1$). This is a subset of attributes, which *can be* used to predict the values of *target* attributes.
3) I_i - a subset of *input* attributes selected by the network construction procedure for predicting the value of the target attribute i ($\forall i$: $I_i \subset C$).

A database is assumed to satisfy the following conditions:

1) 40 $\forall i : I_i \cap O = \varnothing$ (An attribute cannot be both input and a target). This means that no cyclic relationships between attributes may be revealed by the method.
2) $\forall i : I_i \cup O \subseteq R$ (Some attributes may be neither input, nor target). Usually, the key attributes are not used in the knowledge discovery process, since they have no practical meaning, except for the purpose of identifying individual records.

The Info-Fuzzy Network Structure

An Info-Fuzzy Network (see Figure 1 below) has the following components:

1) $|I_i|$ - total number of hidden (internal) layers in a network of a target attribute *i*. The network in Figure 1 has two internal layers (No. 1 and No. 2). Each internal layer is uniquely associated with an input attribute by representing the interaction of that attribute and the input attributes of the previous layers. Layer No. 0 includes only the root node and is not associated with any input attribute. The layers of the network differ from the decision-tree structure used by CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993) in the following aspects: only one input attribute is used to split the nodes of the same layer, multiple splits of continuous attributes are allowed, and the partitioning of continuous attributes is identical at all the split nodes.
2) L_l - a subset of nodes *z* in a hidden layer No. *l* . Each node represents a conjunction of values of the first *l* input attributes, which is similar to the definition of an internal node in a standard decision tree. In Figure 1, the first input attribute has three values, represented by nodes no. 1,2, and 3 in the first layer, but only nodes no. 1 and 3 are split due to the statistical significance testing (see next sub-section). The second layer has four nodes standing for the combinations of two values of the second input attribute with two split nodes of the first layer.
3) K_i - a subset of distinct target nodes V_{ij} in a network of the target attribute *i* (the target layer). $|K_i| = M_i$. Each target node is associated with a value in the domain of the target attribute *i*. This layer is missing in the standard decision-tree structure. In our example, the target attribute has three values, represented by three nodes in the target layer.

$w_z{}^{ij}$- an information-theoretic weight of the connection between a terminal (unsplit) node *z* and a target node V_{ij}. Each connection represents an association between a conjunction of input attribute-values and a value of the target attribute. In Figure 1, there are 15 connections between the five terminal (unsplit) nodes and the three target nodes. The calculation of the weights is explained in the next sub-sections.

The connectionist nature of our system (each terminal node may be connected to every target node) resembles the structure of multi-layer Neural Networks (see Mitchell, 1997). Consequently, we define our system as a *network* and not as a *tree*.

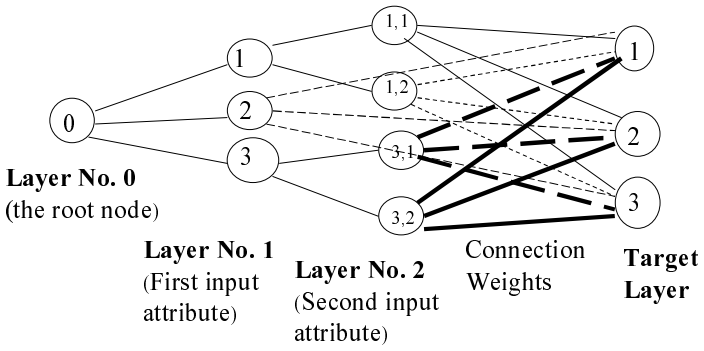

Figure 1 IFN: An Example

Network Construction Procedure

Without loss of generality, we present here an algorithm for constructing an information-theoretic network of a *single* target attribute A_i. In a general case, the network should be re-built for every target attribute defined in a database. The pseudocode of the network construction procedure is shown in Table 1 below.

Table 1 The IFN Construction Algorithm

Inputs

A_i is the target attribute
C is a subset of candidate input attributes
S is a subset of training examples
α is the significance level used by the algorithm

Procedure **IFN** (A_i, C, S, α)
$I_i = \varnothing$ // *initialize the set of input attributes to an empty set.*
$|L_0| = 1$ // *Initialize layer 0 to the root node*
Define *the layer of target nodes*
For $l = 1$ *to* $|C|$ // *repeat for the maximum number of layers (number of candidate input attributes)*
  For $i' = 1$ *to* $|C|$ // *repeat for every candidate input attribute*
    If $A_{i'} \notin I_i$ // *if an attribute is not an input attribute*
      $MI\ (A_i; A_{i'}) = 0$ // *Initialize the mutual information between the attribute* $A_{i'}$ *and the attribute* A_i *to zero.*
      For $z = 1$ *to* $|L_{l-1}|$ // *repeat for every node of the final hidden layer*
        Calculate $MI\ (A_{i'}; A_i / z)$ // *calculate conditional mutual information of a candidate input attribute i' and a target attribute i, given a node z (see below)*
        If $MI\ (A_{i'}; A_i / z)$ is greater than zero at the significance level α (see below), $MI\ (A_i; A_{i'}) = MI\ (A_i; A_{i'}) + MI\ (A_{i'}; A_i / z)$
  If $\max_{i'} MI\ (A_{i'}; A_i) = 0$, **Stop and** return the set of input attributes (I_i)
  $i'^* = \arg \max_{i'} MI\ (A_{i'}; A_i)$ // *find the best input attribute*
  $I_i = I_i \cap A_{i'^*}$ // *update the set of input attributes*
  Update the network structure with a new layer of hidden nodes
Calculate the information-theoretic weights of the input-target connections (see below)
Return the set of input attributes (I_i)

The network construction procedure starts with a single-node network representing an empty set of input attributes. A node in the network is split if it provides a statistically significant decrease in the *conditional entropy* of the target attribute. As indicated in (Cover, 1991), conditional entropy measures the uncertainty of a random variable Y, given the values of other variables X_1, ..., X_n. A decrease in the conditional entropy of a random variable is termed "*conditional mutual information*." The conditional mutual information of a candidate input attribute *i'* and a target attribute *i*, given a node *z* ($MI\ (A_{i'}; A_i / z)$), is estimated by the following formula (based on Cover, 1991):

$$MI\ (A_{i'}; A_i / z) = \sum_{j=0}^{M_i - 1} \sum_{j'=0}^{M_{i'} - 1} P(V_{ij}; V_{i'j'}; z) \bullet \log \frac{P(V_{i'j'}^{ij} / z)}{P(V_{i'j'} / z) \bullet P(V_{ij} / z)}$$

where

M_i / $M_{i'}$ - number of values of the target attribute i /candidate input attribute i'. This formula assumes that all continuous attributes are discretized to a finite number of intervals.

$P(V_{i'j'} / z)$ - an estimated conditional (*a posteriori*) probability of a value j' of the candidate input attribute i', given the node z.

$P(V_{ij} / z)$ - an estimated conditional (*a posteriori*) probability of a value j of the target attribute i, given the node z.

$P(V_{i'j'}^{ij} / z)$ - an estimated conditional (*a posteriori*) probability of a value j' of the candidate input attribute i' and a value j of the target attribute i, given the node z.

$P(V_{ij}; V_{i'j'}; z)$ - an estimated joint probability of a value j of the target attribute i, a value j' of the candidate input attribute i' and the node z.

The statistical significance of the estimated conditional mutual information, is evaluated by using the likelihood-ratio statistic (based on Attneave, 1959):

$$G^2(A_{i'}; A_i / z) = 2 \bullet (\ln 2) \bullet E^*(z) \bullet MI(A_{i'}; A_i / z)$$

Where $E^*(z)$ is the number of tuples associated with the node z.

The null hypothesis (H_0) of the likelihood-ratio test is that the conditional mutual information is zero (which means that the attributes are conditionally independent, given the node). If H_0 holds, then the likelihood-ratio statistic $G^2(A_{i'}; A_i / z)$ is distributed as chi-square with $(NI_{i'}(z) - 1) \bullet (NT_i(z) - 1)$ degrees of freedom, where $NI_{i'}(z)$ is the number of values of a candidate input attribute i' at node z and $NT_i(z)$ is the number of values of a target attribute i at node z (based on Rao and Toutenburg, 1995). The default significance level (*p-value*), used by the information-theoretic algorithm, is 0.1%. We have found empirically that the higher values of the *p-value* tend to decrease the generalization performance of the network.

A new input attribute is selected to maximize the total significant decrease in the conditional entropy as a result of splitting the nodes of the last layer. The nodes of a new hidden layer are defined for a Cartesian product of split nodes of the previous hidden layer and the values of the new input attribute. According to the chain rule (see Cover, 1991), the *mutual information* between a set of input attributes and the target (defined as the overall decrease in the conditional entropy) is equal to the sum of drops in conditional entropy at all the hidden layers. If a candidate input attribute significantly decreasing the conditional entropy of the target attribute cannot be found, the network construction stops.

The conditional entropy of the target attribute can only be calculated with respect to attributes taking a finite number of values. The algorithm performs discretization of continuous attributes "on-the-fly" by using an approach, which is similar to the information-theoretic heuristic of Fayyad and Irani (1993): recursively finding a binary partition of an input attribute that minimizes the conditional entropy of the target attribute. However, the stopping criterion we are using is different. Rather than searching for a *minimum description length* (minimum number of bits for encoding the training data), we make use of a standard statistical *likelihood-ratio test* (see above). The search for the best partition of a continuous attribute is *dynamic*: it is performed each time a candidate input attribute is considered for selection. Detailed descriptions of the algorithm steps, including the dynamic discretization procedure, are provided in (Maimon and Last, 2000).

The IFN construction procedure is a highly scalable algorithm. As shown in (Maimon and Last, 2000), its run time is quadratic in the number of candidate input attributes. It is also linear in the number of records and the number of values taken by each candidate input / target attribute. The dynamic discretization procedure increases the run time per each continuous attribute by the factor of *m log m*, where *m* is the total number of distinct attribute values (bounded by the number of data records).

Rule Extraction and Prediction

Each connection between a terminal node and a node of the target layer represents an association rule of the form *if conjunction of input values, then the target value is likely / unlikely to be...* These are not *prediction rules,* like the rules extracted by the C4.5 algorithm (Quinlan, 1993), since multiple rules may be associated with the same terminal node. An information-theoretic weight of an association rule between a terminal node z and a target value V_j is given by:

$$w_z^j = P(V_j;z) \bullet \log \frac{P(V_j / z)}{P(V_j)}$$

Where $P(V_j;z)$ is an estimated joint probability of the value V_j and the node z, $P(V_j/z)$ is an estimated conditional (*a posteriori*) probability of the value V_j, given the node z, and $P(V_j)$ is an estimated unconditional (*a priori*) probability of the value V_j.

According to the information theory (see Cover, 1991), the above weight represents a contribution of a node-pair to the total mutual information between the input attributes and the target attribute. The weight is positive if the conditional probability of a target attribute value, given the node, is higher than its unconditional probability and negative otherwise. A zero weight means that the target attribute value is independent of the node value. Thus,

each positive connection weight can be interpreted as the *information content* of an appropriate rule of the form *if node, then target value is…*. Accordingly, a negative weight refers to a rule of the form *if node, then target value is* ***not…*** .

Since IFN represents a disjunction of conjunctions of the input attribute values, each record in a relational data table having the same schema like the training data set can be associated with a single terminal node in the network. The target attribute in that record can be assigned a predicted value j^* by the following *maximum a posteriori* (MAP) rule:

$$j^* = \arg\max_j P(V_j / z)$$

For discrete target attributes, V_j stands for an actual attribute value. If a target attribute is continuous, V_j represents a discretized interval, which is converted by IFN into a continuous predicted value by using the mean of the corresponding interval.

The IFN method of prediction and classification is based on the Bayesian approach to learning from data (see Mitchell, 1997). According to the Bayesian reasoning, a *consistent* learning algorithm (i.e., an algorithm that outputs an error-free model over noiseless training data) has to use a MAP hypothesis for prediction.

Fuzzification and Reduction of Association Rules

The number of rules extracted from IFN may be quite large. It is bounded by the product of the number of terminal nodes and the number of target nodes and the previous applications of the algorithm show that this bound is relatively sharp. Although every rule is important for the predictive accuracy of the network, the user may find it difficult to comprehend the entire set of rules and to interpret it in natural and actionable language. As we have shown in (Last and Kandel, 2001 and Last, Klein, and Kandel, 2001), the *fuzzification* of the information-theoretic rules provides an efficient way for reducing the dimensionality of the rule set, without losing its actionable meaning. The process of rule reduction includes the following stages:

Stage 1 - Fuzzifying the information-theoretic rules
Stage 2 – Reducing the set of fuzzified rules by conflict resolution
Stage 3 – Merging rules from the reduced set

Fuzzifying Association Rules

Although the boundaries of the discretized intervals, determined in the process of network construction (see sub-section 0 above), are aimed at minimizing the uncertainty of the target attribute, the user may be more interested in the linguistic descriptions of these intervals, rather in their

precise numeric boundaries. For example, the user is more interested in the rules of the form "If current is high, then the yield is low", which is closer to the human way of reasoning, rather than "If current is between A and B then the yield is between X and Y". People tend to "compute with words" rather than with precise numbers. In addition, the total number of rules, extracted from a typical dataset may be much larger than the number of rules generally used by people in their decisions.

As indicated by (Klir and Yuan, 1995), the "linguistic ranges" of continuous attributes may be expressed as lists of terms that the attributes can take ("high", "low", etc.). The user perception of each term may be represented by fuzzy membership functions. According to (Zadeh 1999), this is the first stage in an automated reasoning process, based on the Computational Theory of Perception (CTP), which can directly operate on perception-based, rather than measurement-based, information. Subsequent CTP stages include constructing the initial constraint set (ICS), goal-directed propagation of constraints, and creating a terminal constraint set, which is the end result of the reasoning process.

Sometimes the "crisp" rules cannot be presented to outsiders, because they contain some sensitive information. This may be an obstacle to open exchange of technological information in forums like professional conferences, multi-company consortia, etc. According to (Shenoi 1993), fuzzification of numeric attributes in a real-world database may be used for an additional purpose: *information clouding*. The user may be unwilling to disclose the actual values of some critical performance indicators associated with manufacturing, marketing, sales, and other areas of business activity. In many cases, data security considerations prevent results of successful data mining projects from being ever published. The application part of this chapter deals with highly sensible data obtained from a semiconductor company. Direct presentation of rules extracted from this data could provide valuable information to the company competitors. However, we are going to "hide" the confidential context of the rules by presenting them in their fuzzified form only.

The terms assigned to each simple condition and to the target (consequence) of the association rule are chosen to maximize the membership function at the middle point of the condition / consequence interval. Thus, we convert a crisp rule into a *fuzzy relation* (Klir and Yuan, 1995). Since a complex condition is a conjunction of simple conditions, an algebraic product is used to find the fuzzy intersection of the simple conditions. Fuzzy implication of Mamdani type (see below) is applied to each rule. Mamdani implication is more appropriate for the fuzzification of the information-theoretic rules due to the local nature of these rules. The informativeness of each fuzzified rule is represented by weighting the implication grade by the information-theoretic weight of the original crisp rule (see sub-section 0

above). If the weight is positive, the rule is stated as "*If <conjunction of terms assigned to rule conditions>, then <term assigned to the rule target >*". If the weight is negative, the rule will be of the form "*If <conjunction of terms assigned to rule conditions>, then* ***not*** *<term assigned to the rule target >*". The expression for calculating the weighted membership grade of an association rule is given below.

$$\mu_R = w \bullet [\prod_{i=1}^{N} \max_j \{\mu_{A_{ij}}(V_i)\}] \bullet \max_k \{\mu_{T_k}(O)\}$$

Where

w – information-theoretic weight of the crisp rule

N – number of simple conditions in the crisp rule

V_i – crisp value of the simple condition i in the crisp rule (middle point of the condition interval)

O – crisp value of the rule target (middle point of the target interval)

$\mu_{A_{ij}}(V_i)$ - membership function of the simple condition i w.r.t. term j

$\mu_{T_k}(O)$ - membership function of the target value O w.r.t. term k

Conflict Resolution

Since an information-theoretic ruleset includes association rules between conjunctions of input values and all possible target values, several rules may have the same IF parts, but different THEN parts. Moreover, the rule consequents may differ in their numeric values, but be identical in their linguistic values. This means that the set of fuzzy rules, produced above, may be *inconsistent*. To resolve this conflict, we calculate the grade of each distinct fuzzy rule by summing up the grades of all identical fuzzified rules and choose from each conflict group the target value that has a maximum grade. A similar approach is used by (Wang and Mendel, 1992) for resolving conflicts in fuzzy rules generated directly from data.

In our procedure, there is no explicit distinction between positive and negative rule grades. For example, the fuzzified rules of the form "If A then B" and "If A then not B" are associated with the same consequent in the same distinct rule. However, their combined grade will be equal to the *difference* of their absolute grades, giving a preference to one of possible conclusions (**B** or **not B**). Eventually, the target value with the maximum *positive* grade will be chosen by the above procedure. This closely agrees with the interests of the database users, who need to estimate *positively* the value of the target attribute.

Merging Reduced Rules

In the previous sub-section, we have shown a method for handling rules having *identical antecedents* and *distinct consequents*. However, the resulting set of conflict-free rules may be further reduced by merging the rules having *distinct antecedents* and *identical consequents*. Thus, any two rules (I) and (II) having the form:

1. *If a is A and b is B and c is C, then t is T*
2. *If d is D and e is E and f is F, then t is T*

can be merged into a single rule of the following form:

3. *If a is A and b is B and c is C* ***or*** *d is D and e is E and f is F, then t is T*

Using the above approach, we can create a rule base of a minimal size, limited only by the number of distinct target values. However, this approach may produce a small number of long and hardly useable rules (like the rule 3 above). Therefore, we perform the merging of disjunctive values *for the last rule condition only*. The procedure of merging fuzzy conjunctive rules (see Last and Kandel, 2001) is based on the assumption that each fuzzy rule has the same partial ordering of input attributes, which is true for any rule extracted from a given IFN (see sub-sections 0 and 0 above). The grade of the merged rule is calculated by using a fuzzy union ("max" operation). The resulting rule base can be considered a terminal constraint set in a CTP process (Zadeh 1999).

KNOWLEDGE DISCOVERY IN SEMICONDUCTOR DATA

In this section, we are applying the information-theoretic fuzzy approach to a real-world data set provided by a semiconductor company. The semiconductor industry is a highly competitive sector, and the original data used in our analysis, as well as any conclusions made from that data are considered highly sensitive proprietary information. Consequently, we were forced to omit or change many details in the description of the target data and the obtained results. As indicated in sub-section 0 above, fuzzification of continuous attributes has helped us to "hide" the proprietary information from the unauthorized (though, probably, curious) reader. As part of the "information clouding" effort, we also refrain here from mentioning the name of the company that has provided the data.

Data Description

The company has provided us with the data on the manufacturing batches that completed their production in the beginning of 1998. The data was derived

from the company database in the form of a single table ("view" in the database terminology), which is shown schematically in Figure 2 below.

Record_ID	Batch_ID	Spec_ID	Priority	Oper_ID	Date_Fin	Qty_Trans	Qty_Scrap

Figure 2 The Original Data Table

The obtained table includes about 110,000 records. A short explanation about each attribute in the table and its relationships with other attributes is given below.

- *Record_ID*. This is the primary key attribute of the table, which uniquely identifies a specific manufacturing operation applied to a given batch.
- *Batch_ID*. This is the identification number of each manufacturing batch.
- *Spec_ID*. This is a specification (part) number of a batch No. *Batch_ID*. It specifies the manufacturing parameters of the batch, like current, voltage, frequency, chip size, etc.
- *Priority*. This is the priority rank of a batch, assigned by the marketing department.
- *Oper_ID*. The ID number of a specific operation applied to the batch No. *Batch_ID*. To preserve the confidentiality of the data, we have converted the actual operation codes into meaningless serial numbers.
- *Date_Fin*. The date when the operation *Oper_ID* was completed. After completion of an operation, the batch is transferred to the next fabrication step on its routing list.
- *Qty_Trans*. The quantity of good chips transferred to the next step. If a batch consists of wafers (before they are cut into individual chips), the number of good chips is calculated automatically from the number of wafers.
- *Qty_Scrap*. This is the number of chips scraped at the operation *Oper_ID*. It is equal to the difference between the number of chips transferred from the previous operation and the number of chips transferred to the next step. If an entire wafer is scraped, the number of scraped chips is calculated automatically by the maximum number of good chips, which can be obtained from a wafer.

By directly applying a data mining algorithm to the above records, one can easily obtain some basic statistical results like the distribution of the number of scraped chips at each operation as a function of the *Spec_ID*. This type of analysis is performed routinely by process and quality engineers. However, much more important and less obvious information may be hidden *between* the records. To discover those "nuggets" of knowledge, some pre-

processing of data is required. The process of data preparation is described in the next sub-section.

Data Preparation

The original dataset included batches from a variety of microelectronic products, each having a different set of functional parameters and requiring a different sequence of operations ("routing"). Rather than trying to build a single data mining model from all the records, we have decided to focus our analysis on a group of 1,635 batches related to a single product family. The batches of this family have three main parameters (chip size, capacitance, and tolerance) and their manufacturing process requires about 50 operations. The selected batches were represented by 58,076 records in the original dataset.

As indicated by (Pyle, 1999), the process of data preparation strongly depends on the specific objectives of knowledge discovery. Here, we are interested to predict the following two parameters: the *yield* and the *flow time* of each manufacturing batch. The process of preparing the selected dataset for the data mining included the following steps:

- *Data normalization.* The original data table does not comply with the *third normal form* of relational tables (see Korth and Silberschatz, 1991), since the attributes *Spec_ID* and *Priority* are fully functionally dependent on the attribute *Batch_ID*, which is a non-key attribute. Consequently, we have moved *Spec_ID* and *Priority* from the original table (which was named *Batch_Flow*) to a new table *Batches*, where *Batch_ID* was defined as the primary key attribute.
- *Calculating batch parameters.* The product parameters (chip size, capacitance, and tolerance) were extracted from the attribute *Spec_ID* by using the available metadata on the attribute's encoding schema and stored in the *Batches* table. Chip size (*Size*) and tolerance (*T_Code*) are nominal attributes, which take a limited number of values, while capacitance is a continuous attribute. In our presentation or results (see below), we have replaced the actual size and tolerance codes with meaningless letters and numbers.
- *Calculating the yield.* The line yield of each batch can be found from dividing the value of the attribute *Qty_Trans* in the last operation of the batch manufacturing process by its value in the first operation. However, the line yield is difficult to analyze, since during the first part of the manufacturing process, the batches are transferred and scraped as *wafers* rather than individual chips. Consequently, the overall line yield is a combination of two yields: the so-called *wafer yield* and the *pieces yield*. A loss in the pieces yield is more expensive than a loss in the wafer yield, since it means that the defects are discovered in one of the final tests, after the manufacturing costs have already been incurred. Our objective here is

to predict the pieces yield (*Yield_P*), which is calculated as the ratio between *Qty_Trans* in the last operation and its value immediately after the wafers are cut ("died") into chips. The attribute *Yield_P* is stored in the *Batches* table.

- *Calculating the flow times*. Another target attribute, the flow time of a batch, was derived as the difference between the values of the attribute *Date_Fin* in the last and the first operation.
- *Discretization of target attributes*. Since the information-theoretic algorithm of sub-section 0 above cannot be applied directly to continuous targets, the attributes *Yield_P* and *Flow_Time* were discretized into ten intervals of approximately equal frequency.
- *Storing completion dates*. The completion dates of all the operations applied to each batch were retrieved from the *Batch_Flow* table and stored in the *Batches* table. This required defining a new attribute for each operation occurring at least once in the routings of the selected batches. This step has created certain amount of redundancy in the database, but it was necessary to eliminate the need of accessing the relatively large *Batch_Flow* table by the data mining algorithm. The *Batch_Flow* table has about 58,000 records vs. 1,635 records only in the *Batches* table.

Constructing the Info-Fuzzy Networks

The information-fuzzy networks related to the target attributes *Yield_P* and *Flow_Time* were constructed by using the algorithm of sub-section 0 above. To predict *Yield_P*, the network construction algorithm was trained only on the records of 1,378 batches, where the number of pieces immediately after the wafers were cut into chips was reported into the system. In Table 2 below, we show the four input attributes included in the network of *Yield_P*. The selected attributes are chip size, product capacitance, and the completion dates of two operations (No. 32 and 38). The column "Mutual Information" shows the cumulative association between a subset of input attributes, selected up to a given iteration inclusively, and the target attribute. Since the mutual information is defined as the difference between unconditional and conditional entropy (Cover 1991), it is bounded by the unconditional entropy of *Yield_P*, which is 3.32 ($log_2 10$). The estimated net increase in the mutual information, due to adding each input attribute, is presented in the column "Conditional MI". The last column "Conditional Entropy" shows the difference between the unconditional entropy (3.32) and the estimated mutual information.

Table 2 Selected Attributes (Target: Yield_P)

Iteration	Attribute Name	Mutual Information	Conditional MI	Conditional Entropy
0	Size	0.111	0.111	3.212
1	Date_32	0.237	0.126	3.086
2	Capacitance	0.286	0.05	3.036
3	Date_38	0.306	0.02	3.017

From the engineering point of view, it is not surprising that the product parameters *Size* and *Capacitance* are related to the yield value. A more interesting and potentially useful result is the selection of completion dates for two operations (32 and 38) as additional factors that affect the yield. This finding may be interpreted as an indication of process instability: batches that went through these operations during certain periods of time had lower quality than batches manufactured in other periods. The boundaries of each period were determined automatically by the dynamic discretization procedure of the algorithm. To find the real causes of yield excursion, the process engineers should compare between the tool conditions in these operations during the "good and the "bad" periods. Other time-dependent factors (e.g., staffing) may be examined as well.

The attributes included in the network of *Flow_Time* are shown in Table 3 below. The selected attributes represent the three main parameters of the product in question: capacitance, chip size, and tolerance. This means that different products experience different kinds of delays in the production line. Some of these delays are probably caused by product-specific manufacturing problems, which may be revealed from the routing documentation of the relevant batches.

One attribute is conspicuous by its absence in Table 3. This attribute is *Priority*, which represents the priority rank assigned to every batch in the beginning of the manufacturing process. Surprisingly enough, the data mining algorithm showed that the batch flow time is *not* affected by its priority. This strange result can be partially understood from a close look at the data: about 95% of all batches in the dataset have the same priority rank, which, of course, undermines the basic idea of prioritizing. Anyway, it indicates a failure to impose a pre-defined schedule on the order of batch production.

Table 3 Selected Attributes (Target: Flow_Time)

Iteration	Attribute Name	Mutual Information	Conditional MI	Conditional Entropy
0	Capacitance	0.107	0.107	3.204
1	Size	0.266	0.159	3.045
2	T_Code	0.354	0.088	2.958

Rule Extraction and Prediction

The Yield Network

The information-fuzzy network built for the *Yield_P* target attribute has four layers (corresponding to four input attributes), total of 23 hidden nodes, 16 terminal (unsplit) nodes, and ten target nodes representing the ten intervals of the discretized target attribute. Thus, the network can have up to 16*10=160 connections between its 16 terminal nodes and the ten nodes of the target layer. The actual number of connections having non-zero information-theoretic weights is 125. Each connection represents an association rule of the form

If Size = V [and Date_32 is between A and B],[and Capacitance is between C and D], [and Date_38 is between E and F], then Yield_P is [not] between G and H

where *V* represents a valid value from the domain of the corresponding nominal attributes (*Size*) and *A,..., H* stand for the boundaries of intervals in discretized continuous attributes (*Capacitance, Date_*, and Yield_P*). The rules having the highest positive and the smallest negative connection weights are given below (confidential information was replaced by meaningless letters).

- **Rule No. 77**: If Size is Z and Date_32 is between 02-Dec-97 and 10-Feb-98, then Yield_P is more than C (weight = 0.0671).
- **Rule No. 27**: If Size is W and Date_32 is between 02-Dec-97 and 10-Feb-98, then Yield_P is not more than C (weight = -0.0163).

The predictive accuracy of the above information-fuzzy network was estimated by holding out one third of the data as a validation set. Thus, we chose randomly 433 validation records and used the other 945 records for constructing the network. The IFN classification accuracy (the probability of identifying the correct interval out of ten) is 23.4% on the training set and 17.3% on the validation set. The 95% confidence interval for the validation accuracy is between 13.8% and 20.9%. The low accuracy of the information-fuzzy network results from the large number of target classes (10) and the inherent noisiness of the manufacturing data. Still, it appears to

be higher than the validation accuracy of the C4.5 algorithm (15.9% only), which was applied to the same data, though the difference between the accuracy of the two algorithms is not statistically significant. At the same time the classification model produced by C4.5 is much more complex than the IFN model: after pruning, the C4.5 tree includes 399 nodes vs. only 23 nodes in IFN. C4.5 uses 37 attributes for prediction, while IFN is satisfied with four input attributes only. Default settings were used for both C4.5 (see Quinlan, 1993) and IFN (see Maimon and Last, 2000).

The Flow Time Network

The information-fuzzy network of the *Flow_Time* target attribute consists of three layers (corresponding to three input attributes), total of 37 hidden nodes (including 28 terminal nodes), and ten target nodes representing the ten intervals of the discretized target attribute. The network can have up to 28*10=280 connections between its 28 terminal nodes and the ten nodes of the target layer, but the actual number of connections having non-zero information-theoretic weights is only 191. Each connection represents an association rule of the form

If Capacitance is between A and B [and Size is V_1], [and T_Code is V_2] then Flow_Time is [not] between C and D

where V_i represents a valid value from the domain of the corresponding nominal attributes (*Size* or *T_Code*) and *A,..., D* stand for the boundaries of intervals in discretized continuous attributes (*Capacitance,* and *Flow_Time*). The rules having the highest positive and the smallest negative connection weights are given below (confidential information was replaced by meaningless letters).

- **Rule No. 135**: If Capacitance is between A and B and Size is W and T_Code is 2 then Flow_Time is between C and D (weight = 0.0282).
- **Rule No. 25**: If Capacitance is between A and B and Size is V then Flow_Time is not between E and F (weight = -0.0057).

The predictive accuracy of the above information-fuzzy network was estimated by holding out one third of the data as a validation set. Thus, we chose randomly 533 validation records and used the other 1,102 records for constructing the network. The IFN classification accuracy (the probability of identifying the correct interval out of ten) is 23.9% on the training set and 18.6% on the validation set. The 95% confidence interval for the validation accuracy is between 15.3% and 21.9%. The low validation accuracy of the information-fuzzy network results again from the inherent noisiness of the manufacturing data. It is lower than the validation accuracy of the C4.5 algorithm (22%), which was applied to the same data, though the difference between the accuracy of the two algorithms is statistically significant at the 5% level only (not at the 1% level). Like in the previous case, the

classification model produced by C4.5 is much more complex than the IFN model: after pruning, the C4.5 tree includes 168 nodes vs. 37 nodes only in IFN. C4.5 uses four attributes for prediction, while IFN is satisfied with three input attributes only.

Rule Fuzzification and Reduction

The Yield Rules

The "crisp" rules extracted from the information-fuzzy network that was described in sub-section 0 above cannot be presented here in their explicit form due to confidentiality of the information they contain. However, their explicit presentation to the users (process engineers) could be hardly useful either, since it is very difficult to generalize and interpret manually a set of more than 100 numeric rules. To make the rule set more compact, interpretable, and less explicit, we have fuzzified two numeric attributes included in the extracted rules: *Capacitance* and *Yield_P*. The following terms (words in natural language) were chosen by us for each fuzzified attribute:

- *Capacitance*: low, high.
- *Yield_P*: low, normal, high.

To convert the above attributes into linguistic variables, we have defined triangular membership functions associated with each term by using the frequency histograms of attribute values. Triangular functions are frequently used in the design of fuzzy systems (Wang 1997). The membership functions are shown in Figures 3 and 4 below without the values of the X-axis, to protect the confidentiality of the original data.

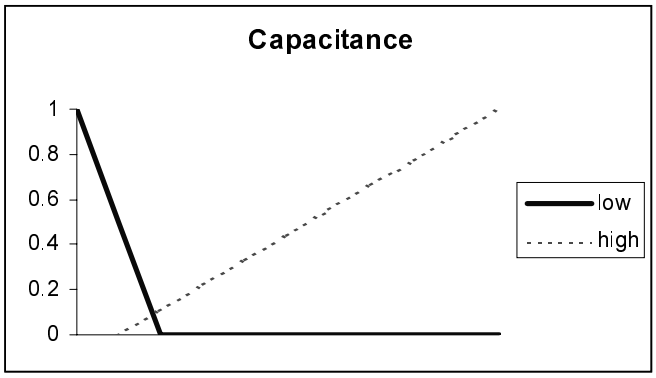

Figure 3 Membership Functions of Capacitance

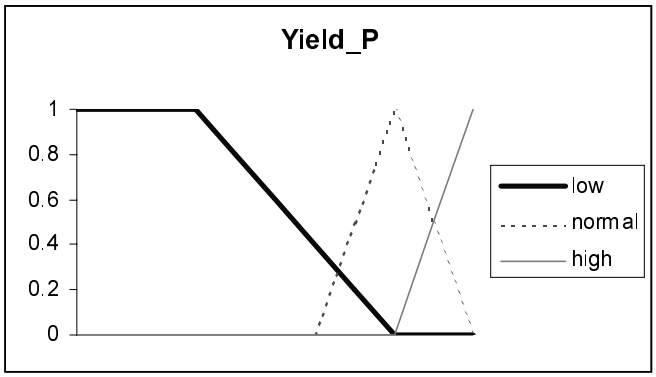

Figure 4 Membership Functions of Yield_P

Fuzzification of the "crisp" rules having the highest and the lowest connection weights (see sub-section 0 above), results in the following linguistic rules:

- **Rule No. 77**: If Size is Z and Date_32 is between 02-Dec-97 and 10-Feb-98, then Yield_P is normal (grade = 0.0387).
- **Rule No. 27**: If Size is W and Date_32 is between 02-Dec-97 and 10-Feb-98, then Yield_P is not normal (grade = - 0.0094).

In Table 4 below, we present the consistent set of fuzzy rules, extracted from the set of fuzzified rules by using the conflict resolution procedure of sub-section 0 above. The last column represents the number of original rules (crisp / fuzzified), associated with a given fuzzy rule. As one can see, the size of the fuzzy rule base has been significantly reduced from 125 original rules to 16 rules only (a decrease of almost 90%).

All the rules in shown in Table 4 are *conjunctions* of fuzzy and "crisp" conditions. However, some rules, like rules No. 2, 3, and 4, can be merged into a *disjunction*, since they have the same consequent (*Yield is normal*). The formal algorithm for merging fuzzy rules was described in sub-section 0 above. The resulting set of nine merged fuzzy rules is shown in Table 5 below.

The users (process engineers) would be particularly interested in the rules describing problematic situations, i.e., rules predicting the yield to be low. Thus, Rule 1 indicates that batches of size W, which passed the operation 32 before December 2, 1997, tend to have a low yield. This means that the engineers should look carefully both at the routing records of all batches that meet these criteria and at the condition of tools and machines used by operation 32 before the above date. Rules 6 and 7 intensify the suspicion that something went wrong with the equipment of operation 32 before Dec. 2, 1997, since two other groups of batches processed at the same time had yield problems. These groups include batches of size X, which have either high or low capacitance, given that the low capacitance batches were processed at operation 38 after Dec. 22, 1997. The inspection of the tool conditions and routing records related to the problematic period at operation 32 may lead to changes in maintenance guidelines, process control limits, and other working procedures.

Table 4 Yield: The Set of Consistent Fuzzy Rules

Rule No	Rule Text	Grade	Number of Crisp Rules
0	If Size is Y then Yield_P is normal	0.0084	7
1	If Size is W and Date_32 is between 21-Dec-95 and 02-Dec-97 then Yield_P is low	0.0237	10
2	If Size is W and Date_32 is between 02-Dec-97 and 10-Feb-98 then Yield_P is normal	0.0097	10
3	If Size is W and Date_32 is between 10-Feb-98 and 12-Mar-98 then Yield_P is normal	0.021	10
4	If Size is W and Date_32 is after 12-Mar-98 then Yield_P is normal	0.0087	10
5	If Size is X and Date_32 is between 10-Feb-98 and 12-Mar-98 then Yield_P is normal	0.0324	9
6	If Size is X and Date_32 is after 12-Mar-98 then Yield_P is normal	0.0096	10
7	If Size is Z and Date_32 is between 21-Dec-95 and 02-Dec-97 then Yield_P is normal	0.0051	4
8	If Size is Z and Date_32 is between 02-Dec-97 and 10-Feb-98 then Yield_P is normal	0.0338	7
9	If Size is Z and Date_32 is between 10-Feb-98 and 12-Mar-98 then Yield_P is normal	0.0073	5
10	If Size is Z and Date_32 is after 12-Mar-98 then Yield_P is normal	0.001	2
11	If Size is X and Date_32 is between 21-Dec-95 and 02-Dec-97 and Capacitance is high then Yield_P is low	0.0065	7
12	If Size is X and Date_32 is between 02-Dec-97 and 10-Feb-98 and Capacitance is low then Yield_P is normal	0.0185	10
13	If Size is X and Date_32 is between 02-Dec-97 and 10-Feb-98 and Capacitance is high then Yield_P is normal	0.0036	9
14	If Size is X and Date_32 is between 21-Dec-95 and 02-Dec-97 and Capacitance is low and Date_38 is between 18-Aug-97 and 22-Dec-97 then Yield_P is low	0.0094	10
15	If Size is X and Date_32 is between 21-Dec-95 and 02-Dec-97 and Capacitance is low and Date_38 is after 22-Dec-97 then Yield_P is normal	0.0055	5

Table 5 Yield: The Set of Merged Fuzzy Rules

Rule No	Rule Text	Target	Grade
0	If Size is Y then Yield_P is	normal	0.0084
1	If Size is W and Date_32 is between 21-Dec-95 and 02-Dec-97 then Yield_P is	low	0.0237
2	If Size is W and Date_32 is after 02-Dec-97 then Yield_P is	normal	0.0097
3	If Size is X and Date_32 is after 10-Feb-98 then Yield_P is	normal	0.021
4	If Size is Z and Date_32 is after 21-Dec-95 then Yield_P is	normal	0.0087
5	If Size is X and Date_32 is between 02-Dec-97 and 10-Feb-98 then Yield_P is	normal	0.0096
6	If Size is X and Date_32 is between 21-Dec-95 and 02-Dec-97 and Capacitance is high then Yield_P is	low	0.0324
7	If Size is X and Date_32 is between 21-Dec-95 and 02-Dec-97 and Capacitance is low and Date_38 is between 18-Aug-97 and 22-Dec-97 then Yield_P is	low	0.0051
8	If Size is X and Date_32 is between 21-Dec-95 and 02-Dec-97 and Capacitance is low and Date_38 is after 22-Dec-97 then Yield_P is	normal	0.0338

The Flow Time Rules

In addition to fuzzifying the attribute *Capacitance* (see sub-section 0 above), we have defined the following terms for the attribute *Flow_Time* (to avoid disclosing its real values): *short*, *medium*, and *long*. The triangular functions for these terms are shown in Figure 5 below.

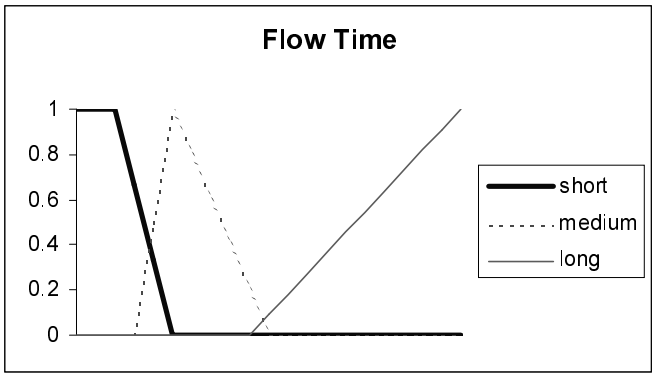

Figure 5 Membership Functions of Flow_Time

Fuzzification of the "crisp" rules having the highest and the lowest connection weights (see sub-section 0 above), results in the following linguistic rules:

- **Rule No. 135**: If Capacitance is low and Size is W and T_Code is 2 then Flow_Time is medium (grade = 0.0129).
- **Rule No. 25**: If Capacitance is low and Size is V then Flow_Time is not long (weight = -0.0057).

In Table 6 below, we present the consistent set of fuzzy rules, extracted from the set of fuzzified rules by using the conflict resolution procedure of sub-section 0 above. The last column represents the number of original rules (crisp / fuzzified), associated with a given fuzzy rule. As one can see, the size of the fuzzy rule base has been significantly reduced from 191 original rules to 12 rules only (a decrease of more than 90%).

The seven rules merged by the procedure of sub-section 0 above are shown in Table 7 below. It appears from the table that the company has more delays in the low capacitance batches than in the high capacitance ones. Special problems with flow times were experienced for sizes V and Y (see Rule 1) and for tolerance 1 of size W (see Rule 5). The engineers should study the routing records of the relevant products to find out, what were the leading delay factors of the batches in question.

Table 6 Flow Time: the Set of Consistent Fuzzy Rules

Rule No	Rule Text	Grade	Number of Crisp Rules
0	If Capacitance is high then Flow_Time is short	0.0082	16
1	If Capacitance is low and Size is V then Flow_Time is long	0.0176	25
2	If Capacitance is low and Size is X then Flow_Time is short	0.0056	18
3	If Capacitance is low and Size is Y then Flow_Time is long	0.0019	10
4	If Capacitance is low and Size is Z then Flow_Time is medium	0.0056	20
5	If Capacitance is low and Size is W then Flow_Time is medium	0.0149	20
6	If Capacitance is low and Size is W and T_Code is 0 then Flow_Time is medium	0.0075	10
7	If Capacitance is low and Size is W and T_Code is 1 then Flow_Time is long	0.009	14
8	If Capacitance is low and Size is W and T_Code is 2 then Flow_Time is medium	0.019	15
9	If Capacitance is low and Size is X and T_Code is 0 then Flow_Time is medium	0.0011	13
10	If Capacitance is low and Size is X and T_Code is 1 then Flow_Time is medium	0.0053	11
11	If Capacitance is low and Size is X and T_Code is 2 then Flow_Time is medium	0.004	19

Table 7 Flow Time: The Set of Merged Fuzzy Rules

Rule No	Rule Text	Target	Grade
0	If Capacitance is high then Flow_Time is	short	0.0082
1	If Capacitance is low and Size is V or Y then Flow_Time is	long	0.0176
2	If Capacitance is low and Size is X then Flow_Time is	short	0.0056
3	If Capacitance is low and Size is Z or W then Flow_Time is	medium	0.0019
4	If Capacitance is low and Size is W and T_Code is 0 or 2 then Flow_Time is	medium	0.0056
5	If Capacitance is low and Size is W and T_Code is 1 then Flow_Time is	long	0.0149
6	If Capacitance is low and Size is X and T_Code is 0 or 1 or 2 then Flow_Time is	medium	0.0075

CONCLUSIONS

In this chapter, we have presented a systematic approach to mining the process and quality data in semiconductor industry. The construction of the data mining model is based on the *Information-Fuzzy Network* (IFN) methodology introduced in (Maimon and Last, 2000). Post-processing of the IFN output follows the Computational Theory of Perception (CTP) approach and it includes information-theoretic fuzzification of numeric association rules, removal of conflicting rules and merging of consistent rules. As demonstrated by the case study of an actual semiconductor database, the method results in a compact and reasonably accurate prediction model, which can be transferred into a small set of interpretable rules. The fuzzification of the rules can also be used for hiding confidential information from external users.

The sound theoretical basis of the information-fuzzy approach and the promising results obtained so far encourage us to further develop this methodology in several directions. These include mining very large and non-stationary datasets, using IFN as a feature selector in the knowledge discovery process, and combining multiple IFN models (see chapter by Maimon and Rokach in this volume).

ACKNOWLEDGMENTS

This work was partially supported by the USF Center for Software Testing under grant no. 2108-004-00.

REFERENCES

Agrawal, R., Mehta, M., Shafer, J., and Srikant, R., "The Quest Data Mining System," in Proceedings of KDD-96, pp. 244-249, 1996.

Attneave, F., Applications of Information Theory to Psychology, Holt, Rinehart, and Winston, 1959.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, P.J., Classification and Regression Trees, Wadsworth, 1984.

Cover, T. M., Elements of Information Theory, Wiley, 1991.

Fayyad, U. and Irani, K., "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," in Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022-1027, 1993.

Korth, H.F. and Silberschatz, A., Database System Concepts, McGraw-Hill, Inc., 1991.

Last, M. and Kandel, A., "Fuzzification and Reduction of Information-Theoretic Rule Sets," in Data Mining and Computational Intelligence, pp. 63-93, Physica-Verlag, 2001.

Last, M., Klein, Y., and Kandel, A., "Knowledge Discovery in Time Series Databases," IEEE Transactions on Systems, Man, and Cybernetics, 31 (1), 160-169, 2001.

Maimon, O. and Last, M., Knowledge Discovery and Data Mining, The Info-Fuzzy Network (IFN) Methodology, Boston: Kluwer Academic Publishers, 2000.

Mitchell, T.M., Machine Learning, McGraw-Hill, 1997.

Pyle, D., Data Preparation for Data Mining, Morgan Kaufmann, 1999.

Quinlan, J. R., C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

Rao, C.R. and Toutenburg, H., Linear Models: Least Squares and Alternatives, Springer-Verlag, 1995.

Shenoi, S., "Multilevel Database Security Using Information Clouding," in Proceedings of IEEE International Conference on Fuzzy Systems, pp. 483-488, 1993.

Tobin, K. W., Karnowski, T.P., and Lakhani, F., "A Survey of Semiconductor Data Management Systems Technology," in Proceedings of SPIE's 25th Annual International Symposium on Microlithography, Santa Clara, CA, February 2000.

Wang, L.-X. and Mendel, J.M., "Generating Fuzzy Rules by Learning from Examples," IEEE Transactions on Systems, Man, and Cybernetics, 22 (6), 1414-1427, 1992.

Wang, L.-X., A Course in Fuzzy Systems and Control, Prentice-Hall, 1997.

Zadeh, L. A., “A New Direction in System Analysis: From Computation with Measurements to Computation with Perceptions,” in New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, pp. 10-11, Springer-Verlag, 1999.