3

Abstraction Hierarchies for Conceptual Engineering Design

Esra Aleisa, Ph.D.
Kuwait University
Kuwait

Abstract
Research in Artificial Intelligence has been a forerunner in developing the most detailed and formalized theories that characterize and create consistent abstraction hierarchies for planning and problem solving. However, the representational methods to exploit these theories are complicated, which limit their application into many disciplines, specifically engineering. The objective of this chapter is threefold: to simplify the representation of current AI-based planning, to identify the properties that ensure effective development of abstraction hierarchies, and accordingly, to develop a methodology for effective and consistent generation of abstraction hierarchies. The proposed methodology achieves these objectives by integrating the well-established AI hierarchical abstraction theories with Steward's practical Design Structure Matrices (DSM). The developed methodology consists of three phases. In the first phase, the literal space and its interactions are formally defined and their interactions are represented as a DSM. The second phase involves clustering literals to abstract classes according to the manner by which they communicate. Finally, in the third phase, the abstract literals are stratified into a loop-free abstraction hierarchy. The approach expands the capabilities of AI-based planning and problem solving abstraction hierarchies and improves their functionality by enabling integration and compatibility with practical DSM planning tools. The effectiveness of the developed methodology is demonstrated by applying it to the conceptual design on a high voltage cable manufactring facility.
Keywords: Abstraction; Design Structured Matrix; Hierarchy; Planning; State-Space Representation

1. Introduction

Abstraction Hierarchies (AHs) are used commonly to represent various large-scale and complex problems (Lam 1996; Holte & Choueiry 2003; Sebastia, Onaindia et al. 2006). Their values have been realized across a wide spectrum of applications mainly to reduce the complexity of problems and to improve solution efficiency (Holte & Choueiry 2003; Aleisa 2005). AHs are also used to speed up the development time, save resources, and provide aggregate intelligent output (Goldin & Klahr 1981; Aleisa 2008). In addition, AH produces designs that are easier to interpret validate and update compared to not using hierarchies. Moreover, AHs help explore design alternatives and produce intelligent decisions at an

early stage of the design or plan (Sacerdoti 1974; Taylor & Henderson 1994; Reddy 1996) . Furthermore, AHs assist in focusing on important aspects of the design problem(Hoover & Rinderle 1994; Sarjoughian, Zeigler et al. 1998; Zeigler, Praehofer et al. 2000). For computational efficiency, AHs have also allows parallel execution of models (Kiran, Cetinkaya et al. 2001), facilitates the utilization of the off-shelf models legacy (McGraw & MacDonald), and enhances model reusability and rapid prototyping (Zeigler 1987; Lin, Yeh et al. 1996; Pidd 1996; Praehofer 1996; Chen & Ghosh 1997; Pidd & Castro 1998; Aleisa & Lin 2008). However, despite AHs' significant benefits, there is a lack of formal methodologies for hierarchical abstraction generation suitable for engineering design. In fact, hierarchical abstraction in general has been described as a "black art" (Knoblock 1994). In this research, we aim to provide a formal hierarchical abstraction methodology to represent and plan engineering design problems at multiple levels of abstraction. Such that partial design solutions obtained at some abstraction level is preserved while the design is augmented at more detailed levels. The objectives of the methodology are three fold:

(1) to develop a representation for engineering design that supports hierarchical abstraction,
(2) to specify the clustering criteria according to which the abstraction process is defined, and
(3) to develop a layering method, by which clusters of abstracted design parameters should be stratified in a hierarchy, without inducing any backtracking in the design process.

In other words, this research proposes a representation, extracts properties that characterize efficient abstraction methods, and proposes a methodology that utilizes an AI-based analysis of efficient systems but overcomes their complications. The methodology consists of three phases. In Phase I, a literal space representation is proposed to represent planning problems in a DSM-based format. In Phase II, the interactions within the literal space framework are utilized to cluster literals into abstract classes. Finally, in Phase III the abstract classes are stratified to construct loop-free abstraction hierarchies.

The reminder of this chapter is structured as follows: first we provide a brief literature review of some of the most persistent abstraction systems and the reason why they are cumbersome when applied to engineering designs. This necessitates the need for this research. Next we dedicate a separate section to explain each of the three developmental phases of our hierarchical abstraction methodology. Then we provide some analysis on the methodology and theoretically proof that it is loop-free. Finally, we demonstrate the effectiveness of the methodology on the design process of a local high voltage cable manufacturing facility.

2. Background

As indicated earlier, AHs have been used to investigate and explore different alternatives earlier in the plan. Moreover, AHs have assisted analysts in focusing on vital aspects of a problem (Hoover & Rinderle 1994; Sarjoughian, Zeigler et al. 1998), leaving inferior details to be determined later. Despite AHs' benefits, the process of developing hierarchical models is more of an art form (Knoblock 1994). The most detailed analysis of abstraction was conducted by research in Artificial Intelligence, specifically, in the fields of planning and problem solving (Giunchiglia & Walsh 1992; Armano, Cherchi et al. 2003). ABSTRIPS

(Sacerdoti 1974; Giunchiglia 1999), one of the earliest abstractions, uses a state-space representation based on a STRIPS (Stanford University Research Institute Planning System) framework. The successors of ABSTRIPS are many, including PRODIGY/EBL (Minton 1988), ABTWEAK (Yang 1990), PABLO (Christensen 1991), ALPINE(Knoblock 1994), HIGHPOINT (Bacchus & Yang 1992) and more (see (Friske & Ribeiro 2006; Marie, Priyang et al. 2008)). A comparison of the most persistent abstraction research is provided in Table 1.

3. The Effectiveness of Abstraction Methods and Applications

Hierarchical models are a result of an iterative application of some abstraction methods. That is, an ordered sequence of abstraction spaces constitutes the skeleton of an abstraction hierarchy (Knoblock 1994; Giunchiglia 1999). Therefore, since abstraction processes are the building blocks of an AH, the efficiency of the abstraction process directly influences that of the AH. For this reason, the properties of effective abstraction need to be thoroughly investigated, which is the topic of the next section.

Author(s)	Measure of detail	Abstraction Approach	Autom-atic?	Assumptions/ Notes/ Contributions
Hobbs '85 Subramanian '89	Piece of data	Reasoning arguments	-	Suggested abstraction by proposing arguments without developing an algorithm
Knoblock '90 Ellman '93	Piece of data	Relevance reasoning	-	Showed computational savings gained by using abstractions both empirically and theoretically when applying relevance reasoning approaches No backtracking is assumed
Knoblock '94	Domain dependent	tractable algorithm that drops irrelevant literals from original problem	Yes	Domain independent, only input is problem formulation, satisfies monotonicity property
Giunchiglia & Walsh '92	Elements of the system or the language	Mapping between systems, set theory and reasoning	-	Established the foundation for abstraction theory and classified various types of abstractions
Bacchus & Yang '92	Piece of data	Constantly removing details to simplify the search space	Yes	Discussed the Downward refinement property (DRP) and showed how hierarchical problem solving techniques that lack this property has no advantage over nonhierarchical methods
Holte *et al.*'96	Piece of data	Used caching techniques to avoid expanding the same searches in successive searches	Yes	An admissible A* search technique
Lu & Tcheng '91	Number of decision variable in the model	Combined inductive learning approaches with optimization techniques to evaluate decisions made at different levels of abstraction	Yes	Proposed AIMS (Adaptive and integrative modeling system) methodology that automatically abstracts detailed systems using machine learning approaches
Pooley '91	Atomic processes	Abstract using graphical technique configuration diagrams	No	Atomic processes from the activity diagram are coupled to form configuration diagrams

Yager '94	Piece of data	Neural Networks	Yes	Can Handle nonnumeric data, developed a function that transforms a group of data into a single data point
Hoover & Rinderle '94	System parameters/ variables	Relevance reasoning, concept of focusing abstractions and Gröbner bases	Yes	Based on Gröbner bases, assumed focusing abstractions change the scope not accuracy, limited to polynomial equation formulations
Kramer & Unger'92	Number of operators in each level	Subsuming Abstraction	Yes	The process is type oriented operator abstracting process that aims on diminishing the number of operators in the detailed level
Taylor & Henderson '94	Features and forms of a mechanical design	Generalization/ specialization and aggregation/ decomposition	No	Showed the relationships between forms and features in a mechanical design and showed how abstraction could aid the design process
Bisantz & Vicente '94	Components and detailed functions of a system	Aggregation/ decomposition and a physical/ functional abstraction approaches	No	Presented how to abstract a system using two orthogonal dimensions simultaneously, the part/whole and the physical functional dimension
Reddy '96	Details of system design specifications	Form empirical models from training examples using multiple learning algorithms	Yes	Multiple learning approaches includes: statistical regression, neural networks, inductive learning algorithms
Fox & Long '95	Details of a plan	Subsumption abstractions	-	Discussed how DRP would indicate if a hierarchical decomposition is worthwhile
Sisti & Farr '98	Depends on the model to be abstracted	Abstracted Models using boundary, behavior and form abstractions	No	Objective was to improve accuracy at aggregate level, compared the terms accuracy, complexity and level of detail and showed how to create model hierarchies that can be interconnected and reused

Table 1. A comparison of the most persistent research in abstraction

4. The Seven Desirable Properties of Abstraction Methods

This section extracts properties that would render an abstraction method to be effective. These include the following characteristics:

(1) Formal. Abstraction methodologies are by large case-dependent, with little to be generalized. Thus, there is a need to develop abstraction methods using well-structured languages and consistent terminology, and to support them with a sound theoretical basis.

(2) Complete. A complete abstraction hierarchy is one that achieves all the steps and preconditions required (Russell & Norvig 1995). On the other hand, an incomplete abstraction hierarchy is described as a theory-decreasing (TD) abstraction (Giunchiglia & Walsh 1992). TD abstractions exhibit deficiency by losing information while abstracting, therefore lacking integrity and affecting the quality of obtained abstract solutions.

(3) Computable. Despite the indispensable need for expertise to articulate effective abstractions, abstraction methods must consist of quantifiable and computable techniques to enable automation(Friske & Ribeiro 2006) and generalization(Pels 2006).
(4) Produce simpler models. When applied to a problem, an abstraction method should produce simpler models that are easier to understand, handle, and solve compared to the original problem representation (Zeigler 1976; Lu & Tcheng 1991; Manfaat, Duffy et al. 1998; Kemke & Walker 2006).
(5) Tractable. Abstraction methods should not involve computational complexities(Gimenez & Jonsson 2008). If so, then the purpose of abstraction is defeated and abstraction will be futile.
(6) Reduce cost. For abstraction to be effective, the cost of creating an abstract model, solving the problem with the abstract model and mapping the solution back to the original representation should be inexpensive, compared to solving the problem directly using its original (or detailed) representation (Bacchus & Yang 1992; Levy 1994; Debbie 2003; Zucker 2003)
(7) Produce consistent and cumulative refinement. This is achieved when backtracking is avoided during the exploitation of an abstraction hierarchy. Eliminating backtracking means that there is no need to resolve any established elements from higher abstract levels in the abstraction hierarchy. As this property is particularly important for achieving efficient designs, it is further elaborated in the next section.

4.1 Consistent and cumulative refinement (ccr) properties

This chapter uses the term consistent and cumulative refinement (CCR) properties to refer to properties that preserve intermediate solutions or results obtained at abstract levels. The essence of the CCR properties is that already established aspects at higher abstraction levels need not be altered as more details are introduced at lower abstraction levels(Zucker 2003). Among the most formalized CCR properties is the Ordered Monotonicity Property (OMP) of Knoblock (Knoblock 1990; Knoblock 1994). According to Knoblock (Knoblock 1994), OMP guarantees that the structure of an abstract solution is not changed by the process of refining it. For this property to hold, the abstraction hierarchy needs to partition a problem, such that the parts of the problem already solved in an abstract space are maintained while the remaining parts of the problem are solved. OMP has the advantage of being computationally tractable, while it is also able to capture a large class of abstraction problems. However, OMP is a heuristic, and thus does not guarantee a reduction of the search space.

Another CCR property is the Downward Refinement Property (DRP) (Bacchus & Yang 1992; Helmert 2006). A planning domain is said to possess DRP if all abstract plans can be consistently refined without backtracking across abstraction levels (Fox & Long 1995). Bacchus and Yang (Bacchus & Yang 1992) emphasized that when DRP holds, backtracking needs never occur across various levels of the abstraction hierarchy, indicating a hierarchical decomposition is worthwhile(Zucker 2003). However, being a heuristic, DRP encounters difficulties similar to those of OMP.

5. Integrating the Design Structured Matrix to AI Hierarchical Abstraction

Formulated by Steward (Steward 1981), the Design Structure Matrix (DSM) a.k.a. dependency structure matrix, is a project modeling tool to plan, represent and analyze the flow of information among different tasks of complex design projects (McCord 1993; Browning 2001). DSM is a square binary matrix with rows and columns, where is the number of design tasks under consideration (Warfield 1973). If task i is dependent on task j, then the entry of the respective column j and row i is unity or marked with an X (Browning 1999; Yassine, Falkenburg et al. 1999). Off-diagonal marks represent coupling between tasks, marks in the upper triangle in DSM represent feedforward coupling, and marks in the lower triangle represent feedback coupling (Rogers 1996). The DSM tasks are rearranged in order to eliminate feedback marks. Then, the DSM is partitioned into blocks of tasks that simultaneously depend on one another. Three different relationships can be identified from a partitioned DSM: sequential, parallel and coupled tasks. A task can be performed sequentially if its row contains a mark just below the diagonal; a task is parallel if there are no marks linking it with other tasks; coupled tasks are ones that hinder a partitioned DSM to be lower triangular (Yassine, Falkenburg et al. 1999). Finally, feedback marks are removed from the DSM in a processes called tearing (Steward 1981) to initiate sequencing within blocks (Eppinger, Whitney et al. 1994).
This research intends to utilize the DSM representational advantage to simplify AI-based abstraction hierarchies.

6. Hierarchical Abstraction Methodology for Structuring Literal Spaces

The presented hierarchical abstraction methodology consists of three phases: representation, abstraction and layering. The representation phase were literal spaces are formulated into a transposed DSM. In the second phase, the abstraction phase, the problem literals are clustered into mutually-exclusive abstract equivalence classes (AECs). Finally, in the third phase, the layering phase, the different AECs are stratified into multiple levels of a hierarchy using a level assignment algorithm (LAA). The three phases of the methodology are illustrated in Fig. 1and are discussed in greater detail in the following sections.

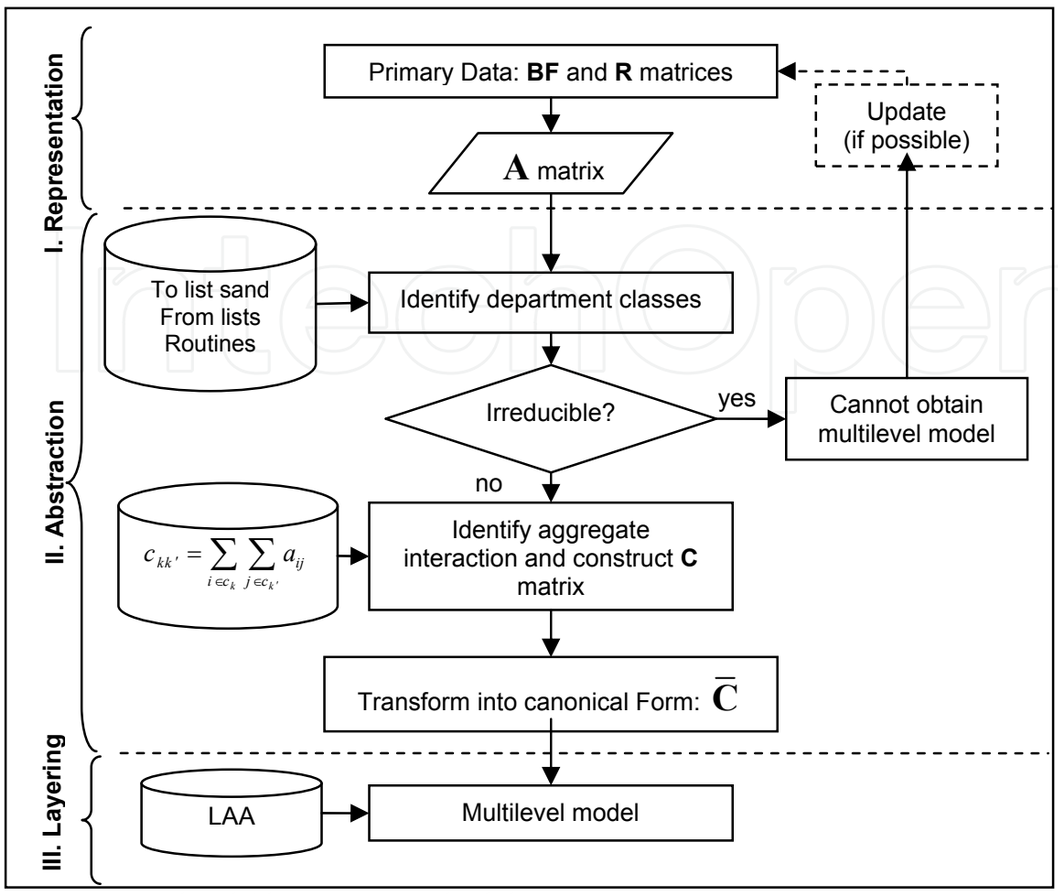

Fig. 1. Hierarchical Abstraction Methodology

6.1 Phase I: The literal space representation

Let l_i denote a positive or a negative literal i, where a literal is defined as an atomic expression or the negation of an atomic expression (Luger 2002). The literal space Ω is a set that consists of all literals under consideration. Similarly, O denotes the set of operators, $(k = 1,2,...,p)$, such that $O = \{o_1, o_2, ..., o_p\}$. Analogous to the STRIPS framework, for each operator , let p_k be the set that contains all the preconditions. A precondition set p_k is the set of some literals $l_i \in \Omega$ that need to be achieved prior to the application of an operator. Similarly, we define e_k to be the set of effects of operator o_k, where an effect is the set of some achieved literals $l_i \in \Omega$ that resulted from applying an operator. For that we write each operator a as an ordered tuple of $o(p,e)$. It is possible for p_k or to be empty, indicating that a specific operator k does not require preconditions nor result in any effects respectively. As an example, describes an operator o_k with no preconditions and literal l_i of effects, which is typical for initialization operators.

6.1.1 Formalizing Interactions among literals and operators

The representation of operators in terms of their preconditions and effects indicates a causal relationship between them. This is the result of having some operators $o_k, o_{k'} \in O$ $(k \neq k')$ where; hence o_k contains among its effects some literal $l \in \Omega$ that is part of the preconditions of $o_{k'}$ that are required for it to be applied. If the above holds, then we say that o_k *establishes* some literals for $o_{k'}$. Establishment is formally defined below.

- ***Establishment definition:*** Let operators $o_k, o_{k'} \in O$ where $k \neq k'$, and literal $l \in \Omega$. Let e_k be the set of effects of , and $p_{k'}$ be the set of preconditions of $o_{k'}$. We say that o_k *establishes* literal l for $o_{k'}$ $(establishes(o_k, o_{k'}, \{l\}))$ if and only if

$$\exists l \in \Omega \text{ , such that} \quad (1)$$
$$l \in e_k \text{ , and} \quad (2)$$

Establishment has been requisitely used in the literature of planning and problem solving within the field of Artificial Intelligence (AI). However, establishment definitions usually impose an additional restriction on the precedence between two operators with respect to a plan. Nevertheless, this restriction is not necessary in this context since it is not intended to produce the shortest possible sequence of operators that transform the initial state to the goal state.

When operators' precedence constraints are not imposed within the establishment definition, establishments can be interpreted as causal links common in engineering applications. In engineering practices, it is customary to represent causality in a matrix representation (Warfield 1973). In this research, we define two types of causal links that result among operators and literals respectively. These causal links are discussed in the following sections.

Operator causality definition: Let $o_k, o_{k'} \in O$ and $l \in \Omega$; the operator causality link $a_{kk'}$ is defined as follows:

$$a_{kk'} = \begin{cases} 1 & \text{if } establishes(o_k, o_{k'}, \{l\}) \text{ for some } l \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The above definition indicates that if $establishes(o_k, o_{k'}, \{l\})$ holds for some operators $o_k, o_{k'} \in O$ and some $l \in \Omega$, then according to the establishment definition l must be a precondition and an effect in o_k and $o_{k'}$ respectively (i.e. , $l \in e_{k'}$). If this is the case, then the operator causal link $a_{kk'}$ is greater than zero. Fig. 2 shows a digraph of the operator causality definition.

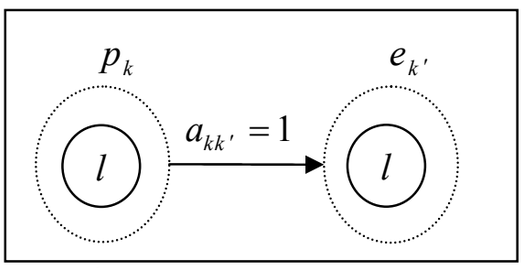

Fig. 2. Digraph of operator causality

- **Causality within Operators**

Operator causality defined earlier identifies the relationships among the different operators. Literal causality however, describes the relationship among literals within operators.

Literal causality definition: Let literals $l_i, l_j \in \Omega$ and let p_k, e_k be respectively the preconditions and effects of operator $o_k \in O$; the literal causality link is defined as follows:

$$r_{ij} = \begin{cases} 1 & \text{if } \exists o_k \in O;\ l_i \in p_k,\ \text{and } l_j \in e_k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Therefore, the causality link r_{ij} is nonnegative when literals l_i and l_j belong to the set of preconditions and effects respectively of any arbitrary operator in O. Fig 3 illustrates literal causality among three operators, while Fig. **4** shows the corresponding operator causality and establishment of the former Figure.

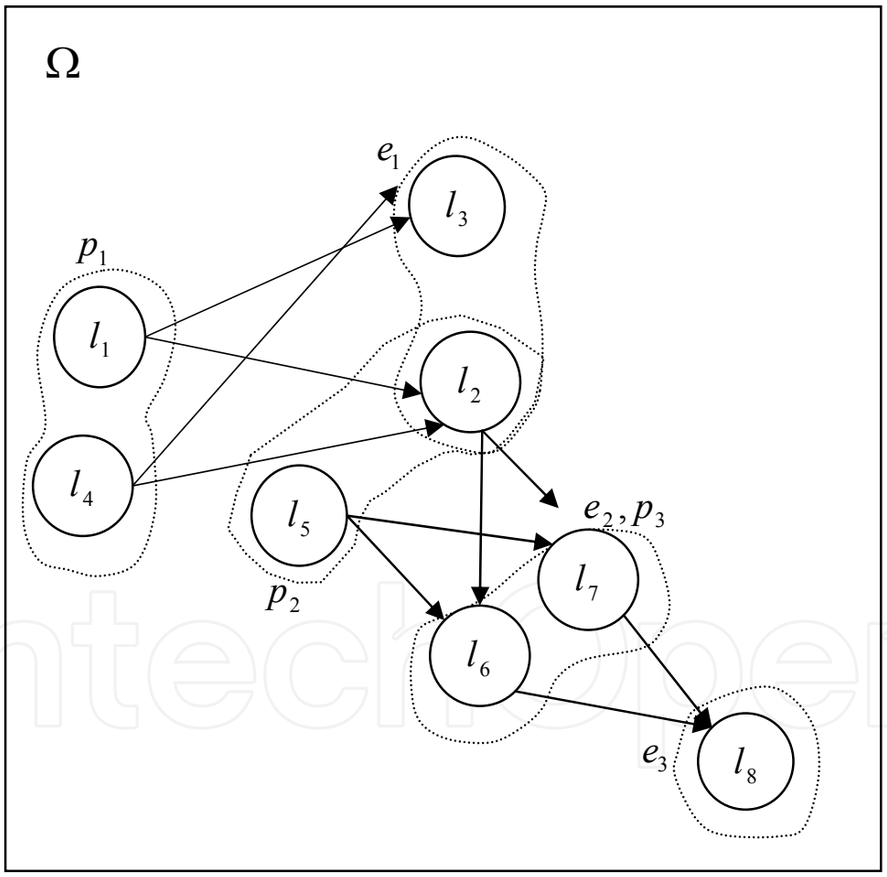

Fig. 3. Literal causality among the literals of three related hypothetical operators

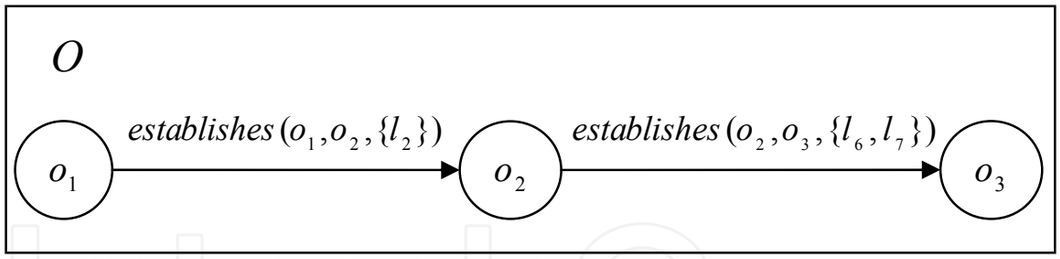

Fig. 4. Establishment resulting from the example given in Fig. 3

Let $\mathbf{R} = [r_{ij}]$ define a literal causality matrix of size $n \times n$ whose entries follow Eq (2). In graph theory (Deo 1974), $\mathbf{R}$ corresponds to a node-to-node incidence matrix. Moreover, the transposed form of the $\mathbf{R}$ matrix (i.e. $\mathbf{R}^{\mathbf{T}}$) has an equivalent structure to that of a DSM(Aleisa & Lin 2009). This allows us to exploit the well-established methods of DSM to structure literal spaces, while still remaining consistent with the previous literature on state-space literature, the theory of ordered relations (Dartmouth College Writing Group. & Cogan 1958) and Markov Chains by considering the transposed form.

6.2 Accessibility and communication among literals

In this research, we use $\mathbf{R}^{(\mathbf{S})}$ to indicate that the matrix $\mathbf{R}$ is multiplied s times by itself. Based on matrix theory, we can interpret $r_{ij}^{(S)} \geq 0$ as the ability to reach literal j from i, passing through s literals or alternatively through the application of s operators. Note that we shall refer to $r_{ij}^{(1)}$ by r_{ij} for simplicity. Based on the interpretation of $r_{ij}^{(S)} \geq 0$, we define literal accessibility and communication.

Accessibility: We say that l_j is *accessible* from l_i ($accessible(l_i, l_j)$), if and only if $r_{ij}^{(s)} > 0$ through a number of operators, $s = 1,2,\ldots$.

If there is no operator o_k applied on l_i, then the value of l_i is assumed to remain unaffected. Hence, it is legitimate to assume that every literal is accessible at least by itself, therefore:

$$r_{ij} \geq 0 \ , \forall i = j \tag{3}$$

Therefore, accessibility has two relational properties:

(1) Reflexive, based on Eq.(3).

(2) Transitive, since:

$$\begin{aligned} & accessible(p_i, p_j) \cap accessible(p_j, p_k) \\ & \Rightarrow accessible(p_i, p_k), \qquad , \\ & \forall p_i, p_j, p_k \in \Omega \end{aligned} \tag{4}$$

Communication: Let $l_i, l_j \in \Omega$, l_i and l_j communicate ($communicate(l_i, l_j)$) if and only if the following holds:

$$accessible(l_i, l_j) \cap accessible(l_j, l_i) \tag{5}$$

Alternatively, communication between two literals $l_i, l_j \in \Omega$ implies that the following hold:

$$r_{ij}^{(s)} > 0\,,\ r_{ji}^{(s)} > 0 \ \text{ for some } s = 1,2,... \tag{6}$$

Let $\mathbf{R} = [r_{ij}]$ define a literal causality matrix of size $n \times n$. The transposed form of the $\mathbf{R}$ matrix (i.e.) has an equivalent structure to that of a DSM. This allows us to exploit the well-established methods of DSM to structure literal spaces.

7. Phase II: Abstraction of the Literal Space

This phase creates an abstract literal space of Ω, denoted by ω by clustering the literals under consideration into mutually-exclusive partitions.
Eq. (6) shows that communication is a reflexive, symmetric and transitive relation. A relation that exhibits these properties is an equivalence relation (Kemeny & Snell 1960). Equivalence relations have the ability to partition the universe Ω upon which it is defined to disjointed partitions (Dartmouth College Writing Group. & Cogan 1958). Each of these partitions defines a unique cluster of communicating literals, which is referred to as abstract equivalence classes.

7.1 Abstract equivalence classes

An abstract equivalence classes (AEC), denoted by c_k ($k = 1,2,...,m$), is a set of literals by which all members of this set communicate with one another. k corresponds to the number of $c_k \subseteq \omega$. If the abstract literal space ω consists of a single AEC (i.e. $k = 1$), is called *irreducible* to be consistent with the terminology used in Markov Chains (Kao 1997). Therefore, irreducibility implies that the literals of the original literal space Ω all communicate with one another. Because AECs are developed based on an equivalence relation (i.e. communication), then the following must hold:

$$\bigcap_{\forall k} c_k = \varnothing\,,\ \bigcup_{\forall k} c_k = \Omega\,,\ \forall k \tag{7}$$

7.2 The formation of AECS

The equivalence class formation algorithm (ECFA) is used to abstract the literal space Ω into ω. In ECFA, T_i B denotes the to-list of l_j, such that each T_i contains all the literals that l_j can access through one or more operators. Similarly a from-list F_i is defined to contain all the literals from which l_j is accessible through one or more operators. c_i is a set of communicating literals that contains l_i. The codes for constructing a To lists and a From lists are provided in Fig. 5 and Fig. 6 respectively. Detailed steps of these routines are provided in (Gaver & Thompson 1973).

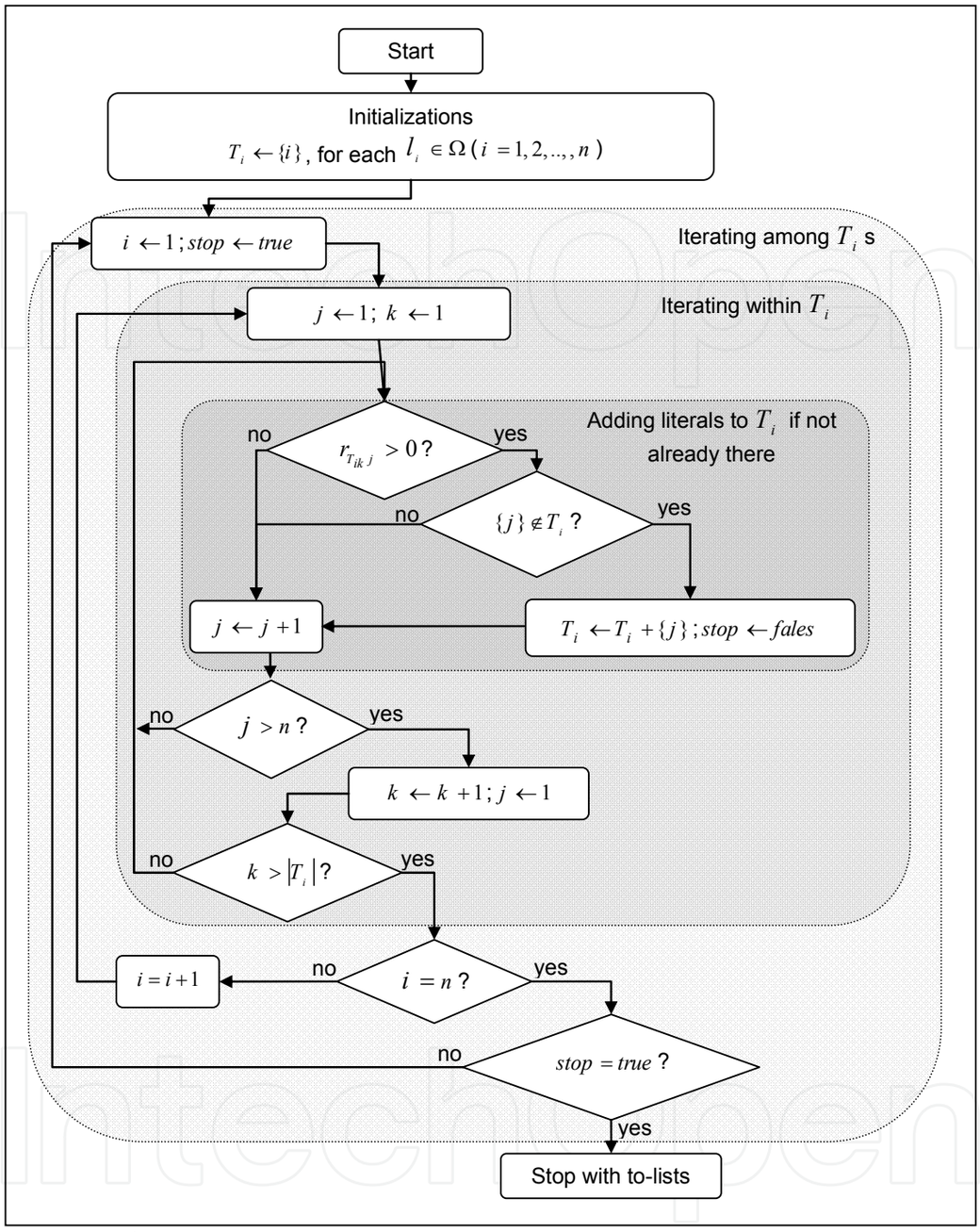

Fig. 5. Routine for constructing to-lists

Constructing equivalence classes of literals: Having obtained the to-list and from-list for each l_i, AECs can be obtained by intersecting the two sets T_i and F_i:

$$c_i = T_i \cap F_i, \ \forall i \tag{8}$$

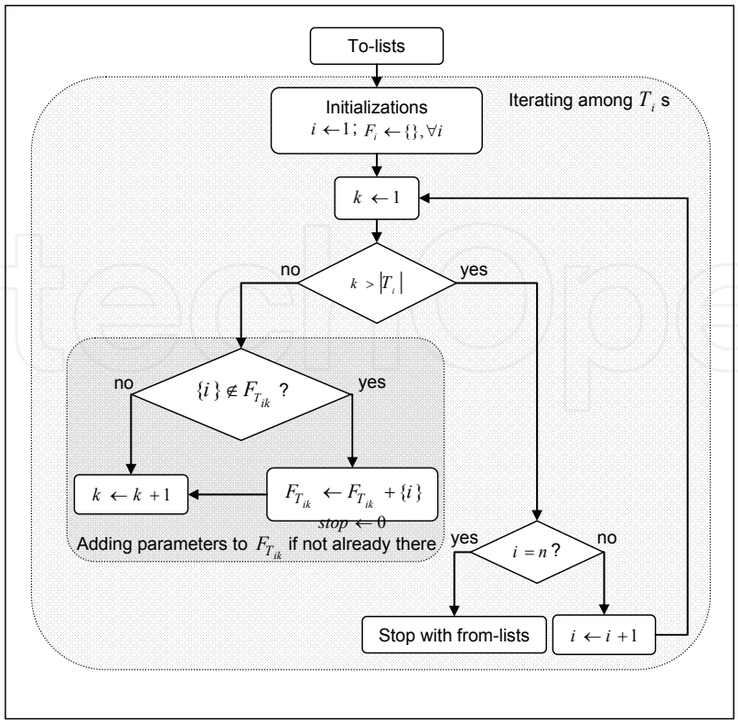

Fig. 6. Routine for constructing From-lists

7.3 The aggregate interaction matrix

The classification of literals into AECs leads to the discussion on aggregate interaction or flow that results among them. Let $\mathbf{C} = [c_{kk'}]$ denote the AECs interaction matrix for aggregate flow. Each entry $c_{kk'}$ of $\mathbf{C}$ is defined by the *Boolean* sum of the following equation:

$$c_{kk'} = \sum_{i \in c_k} \sum_{j \in c_{k'}} r_{ij} \tag{9}$$

$\mathbf{C}$ is a square matrix of size $m \times m$, where m is the number of AECs in ω. Each $c_{kk'}$ represents the summation of corresponding rows and columns of the $\mathbf{R}$ matrix. Here, $\mathbf{C}^{(s)}$ denotes the $\mathbf{C}$ matrix multiplied s times by itself. As in the entries of the $\mathbf{R}$ matrix, in $\mathbf{C}$, if $c_{kk'}^{(s)} > 0$ for some $s = 1,2,\ldots$, then there is an interaction between the two AECs k and k' passing through s aggregate interactions. Hence AEC k' is accessible from AEC k. This leads to the definition of AEC accessibility.

7.4 Classification of AECS:

Another important characterization of AECs is whether an AEC is absorbing or transient, or maximal transient:

- *Absorbing AEC (AAEC):* an AEC that does not access any other AEC but itself. Therefore, an AAEC $c_k \subseteq \Omega$ is one where:

$$c_{kk'} = 0 \ , \forall k \neq k' \tag{10}$$

- *Transient AEC (TAEC):* an AEC capable of accessing other AECs besides itself. A TAEC satisfies the following:

$$\exists\ c_{kk'} > 0\ ,\ k \neq k' \tag{11}$$

- *Maximal transient AEC (MTAEC):* Is TAEC not accessed by any other TAEC beside itself, such that is must satisfy Eq.(9) together with:

$$\neg\exists\ c_{k''k'} > 0\ ,\ k'' \neq k' \tag{12}$$

7.5 Canonical form of the $\mathbf{C}$ matrix

To prepare the $\mathbf{C}$ matrix for the layering phase, Its rows and columns are rearranged, such that the first $m-t$ ones contain the AAECs, while the remaining t ones contain the TAECs. When this segregation is applied to the $\mathbf{C}$ matrix, then it is said to be in canonical form, denoted by $\bar{\mathbf{C}}$. A general structure of a $\bar{\mathbf{C}}$ matrix is given below:

$$\bar{\mathbf{C}} = \begin{array}{c} \\ m-t \\ \\ t \\ \\ \end{array} \begin{array}{c} \begin{array}{cc} m-t & \quad t \end{array} \\ \left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{T} & \mathbf{Q} \end{array}\right) \end{array}$$

The resultant submatrices of $\bar{\mathbf{C}}$ are as follows:

(1) $\mathbf{I}_{(m-t)\times(m-t)}$ is the identity matrix, because an AAEC has only access to itself.

(2) $\mathbf{0}_{(m-t)\times(t)}$ consists entirely of zeros, since AAECs cannot access TAECs.

(3) $\mathbf{T}_{(t)\times(m-t)}$ represent accessibility from TAECs to each AAEC.

(4) $\mathbf{Q}_{(t)\times(t)}$ depicts accessibility among TAECs.

8. Phase III: Constructing the Hierarchy

The construction of an AH is conducted in a recursive and bottom-up manner, where it starts from the lowest level of detail (level zero) and subsequently building higher levels based on the abstract class accessibility relationships that exist among different AECs. The layering process is designed to eliminate backtracking in the plan.

Level zero is designated to include the details that can be postponed until the end when solving the problem hierarchically. However, level n, the highest level of abstraction, includes the details that need to be considered in the beginning. Therefore, the algorithm builds the hierarchy in a bottom-up fashion, but expects it to be executed in a top-down fashion.

8.1 Constraints for Loop-Free Level Assignments

The assignment of literals to levels is based on the following constraints to guarantee loop free AHs.

- *Constraint 1(Literal Level Assignment Constraint):* Let $level(l_i)$ denote the level of the design literal l_i in an AH. For all $l_i, l_j \in \Omega$, if $r_{ij}^{(s)} > 0$ for some $s > 0$, then $level(l_i) \geq level(l_j)$ to avoid backtracking.

The above constraint indicates that if l_i accesses l_j, then l_i should at least be at the same or a higher level than. This confirms findings from previous literature on abstraction hierarchies for planning and problem solving, particularly, Knoblock's (Knoblock 1994) restriction to automatically generate loop-free AHs for planning and problem solving.

- *Constraint 2 (Communicating Literals Level Assignment Constraint):* Let $level(l_i)$ denote the level of design literal l_i in the AH. For all $l_i, l_j \in \Omega$, if $communicate(l_i, l_j)$, then $level(l_i) = level(l_j)$.

If $communicate(l_i, l_j)$, then by definition there exists $r_{ij}^{(s_1)} > 0$ and $r_{ji}^{(s_2)} > 0$ for some $s_1, s_2 > 0$. Hence, by Constraint 1, $level(l_i) \geq level(l_j)$ and $level(l_i) \leq level(l_j)$, which implies $level(l_i) = level(l_j)$.

- *Constraint 3 (AECs Level Assignment Constraint):* Let $level(c_k)$ denote the level of AEC k in an AH. For all $c_k, c_{k'} \subset \Omega$ where $k \neq k'$, if $c_{kk'} > 0$, then $level(c_k) > level(c_{k'})$ to avoid backtracking.

Constraint 3 is a direct result of applying Constraints 1 and 2. Based on the definition of accessibility, if $c_{kk'}^{(s)} > 0$ then, $\exists l_i \in c_k$ and $\exists l_j \in c_{k''}$ such that $r_{ij}^{(s)} > 0$ for some $s > 0$. Based on Constraint 1, $level(l_i) \geq level(l_j)$. Since classes consist of communicating literals, then $level(c_k) > level(c_{k'})$. But classes cannot communicate; therefore, it is not possible to have $level(c_k) = level(c_{k'})$ when $c_{kk'}^{(s)} > 0$. Therefore, $level(c_k) > level(c_{k'})$ for $c_{kk'}^{(s)} > 0$, and hence c_k need to be considered before $c_{k'}$ to avoid backtracking. The following theorem shows that applying Constraint 3 will result in loop-free AHs.

Theorem1. : Any AH developed using Constraint 3 is loop-free.

Proof.

Looping (backtracking) occurs if $\exists c_k, c_{k'} \subset \omega$, where $classaccessible(c_k, c_{k'})$ and $level(c_k) > level(c_{k'})$. Here it shows that this never occurs, considering the three cases of AAECs, TAECs and MTAECs

- *Case I (AAECs)*: if c_k is absorbing, then $level(c_k) = 0$. Also $\neg\exists c_{k'} \subset \omega$, where $classaccessible(c_k, c_{k'})$; thus $level(c_{k'}) > level(c_k)$ cannot occur.
- *Case II (MTAECs)*: if c_k is a MTAEC, then $level(c_k) = n$. Thus $\neg\exists c_{k'} \subset \omega$, where $level(c_{k'}) > level(c_k)$.
- *Case III (TAECs):* if c_k is a TAEC, then it must be true that $\exists c_{k',k''} \subset \omega$, where $classaccessible(c_{k'}, c_k)$ and $classaccessible(c_k, c_{k''})$. Thus, according to Constraint 3, $level(c_{k'}) < level(c_k) < level(c_{k''})$, and a reverse order can never occur.

From these three cases, it can be concluded that $level(c_{k'}) > level(c_k)$ will never occur for all $classaccessible(c_k, c_{k'})$. Hence the AH is loop-free.

This proof demonstrates that an AH developed by the methodology in hand will always produce loop-free AHs.

9. The Level Assignment Algorithm

The Level Assignment Algorithm (LAA) generates AHs by assigning AECs to their appropriate level of abstraction. In LAA, the assignments are accomplished on the premises of the preceding developed constraints.

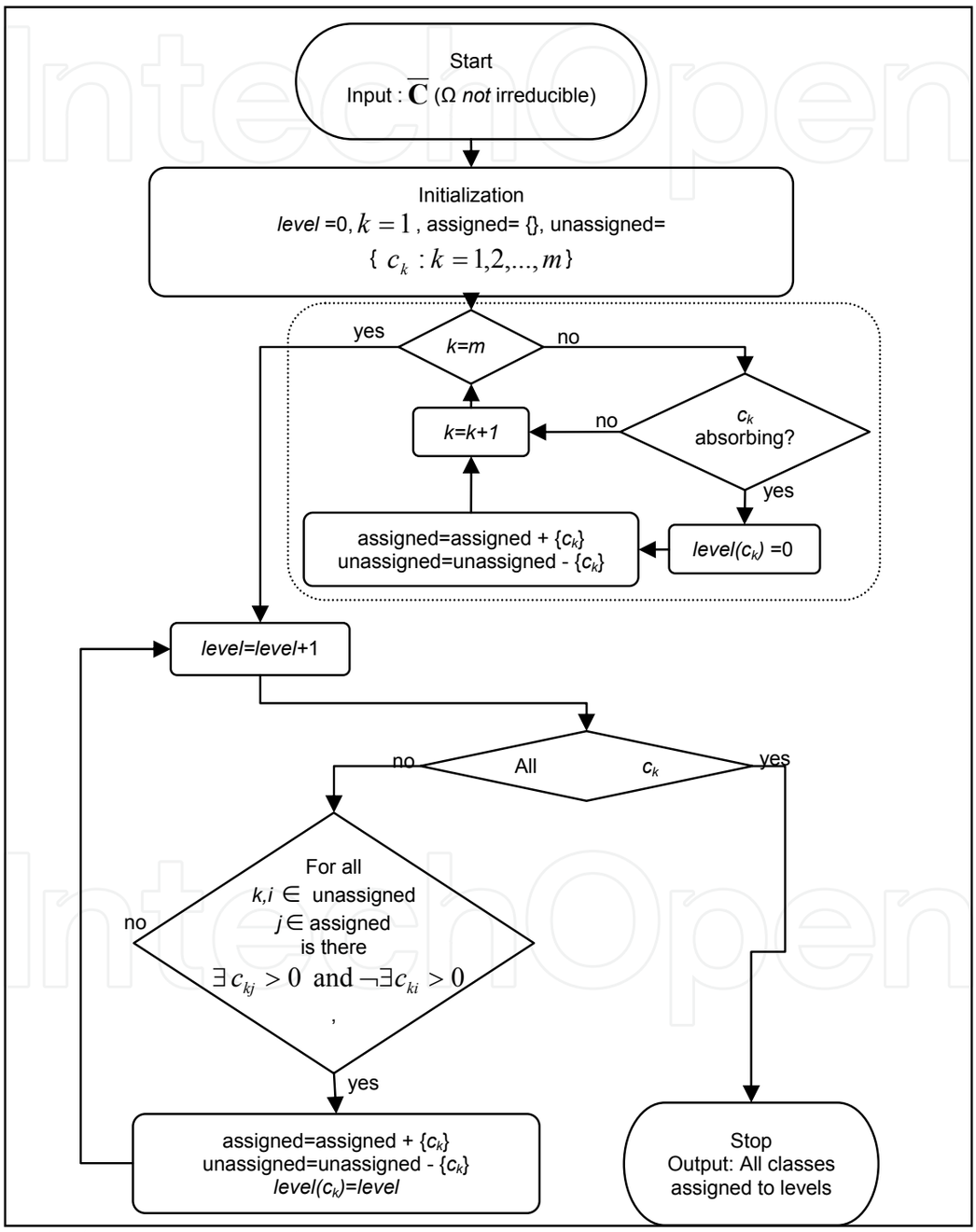

Fig. 7. The level assignment algorithm

10. Illustrative Example

In this section, effectiveness of the developed methodology is demonstrated through the design of a layout for manufacturing plant that produces high voltage power cables. The plant produces a few variations of the high voltage cable shown in Fig. 8, based on customer specifications regarding conductor properties, insulation thickness, cable color coding, armoring metals, and so forth.

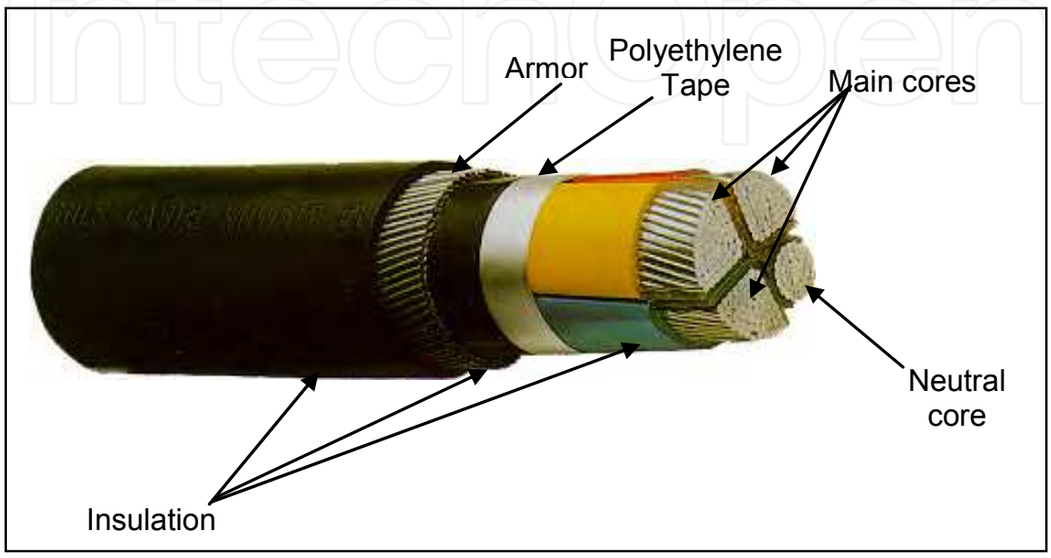

Fig. 8. Components of the high voltage cable

As shown in Fig. 8, the high voltage power cable consists of three main aluminum cores, each of which has a diameter of 300 mm, and a neutral core of 185 mm diameter. The three main cores and the neutral core consist of 61 and 37 insulated stranded aluminum rods, respectively. The four cores are warped with polyethylene tape that is supported by a layer of insulation. Finally, the cable is armored with steel and wires for protection and is sheathed by an additional layer of insulated. The flow chart shown in Fig. 9, describes the flow of the cable across the different stations.

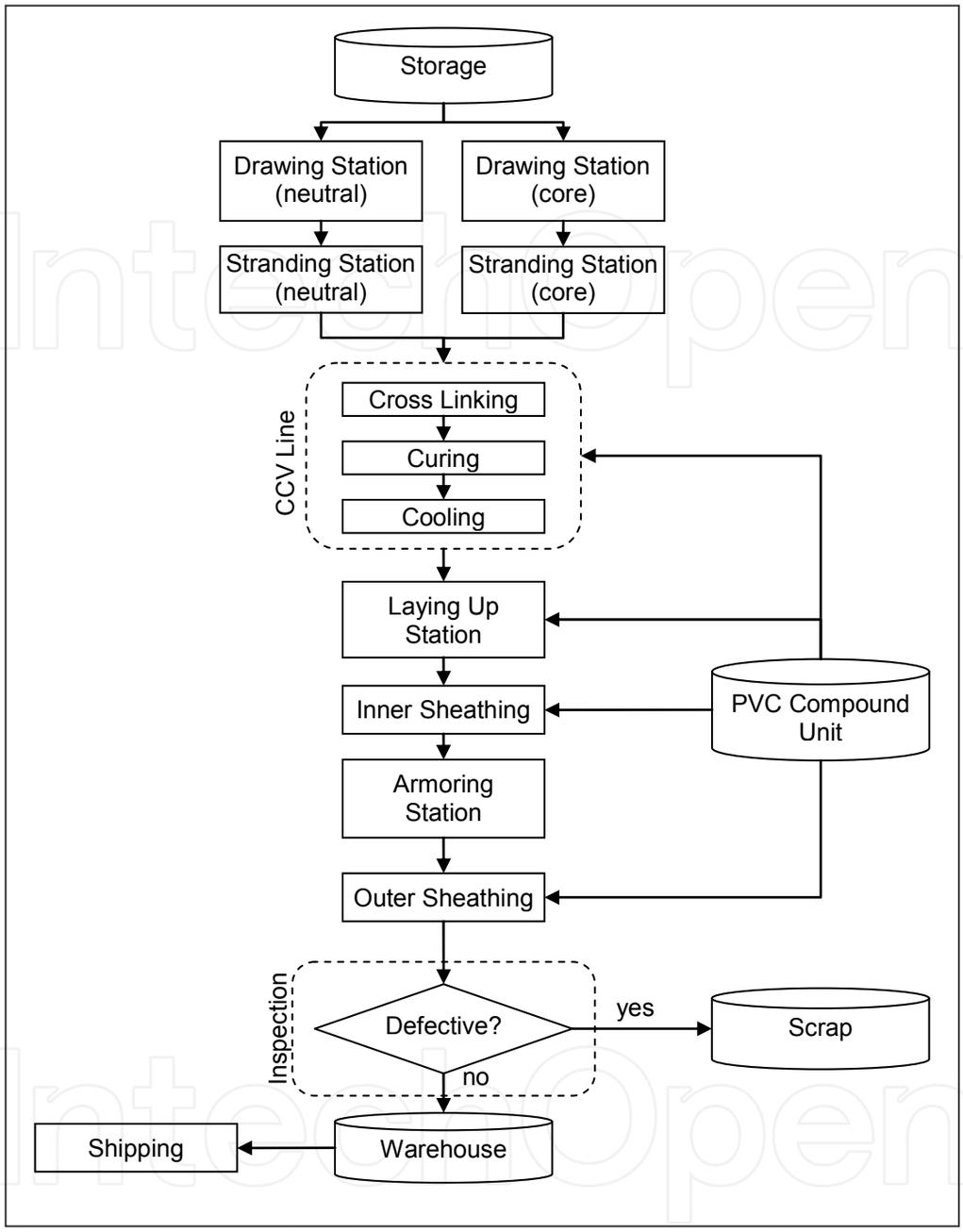

Fig. 9. Flowchart of the manufacturing processes for the high voltage cable

10.1 Phase I: the literal space representation for the cable manufacturing company

The high voltage cable facility consists of 74 machines and areas that are distributed within the cable manufacturing stations shown in Fig. 9 together with WIP areas, forklift parking, storages, warehouses, shop floor offices, lounges, etc. These areas are shown in Table 2.

l_i	Code	Name	l_i	Code	Name
0	PD1	Preliminary Drawing Station	38	SHS3	Sheathing Station
1	PD2	‘’	39	SHS4	“
2	PD3	‘’	40	XL1	Cross-Linking Station
3	PD4	‘’	41	XL2	“
4	ID1	Intermediary Drawing Station	42	XL3	“
5	ID2	“	43	XL4	“
6	ID3	“	44	CU1	Curing Machinery
7	ID4	“	45	CU2	“
8	ID5	“	46	CU3	“
9	ID6	“	47	CU4	“
10	ID7	“	48	CO1	Cooling Station
11	ID8	“	49	CO2	“
12	DD1	Main Detailed Drawing Station	50	CO3	“
13	DD2	“	51	CO4	“
14	DD3	“	52	L1	Lay-up Station
15	DD4	“	53	L2	“
16	DD5	“	54	A1	Armoring Station
17	DD6	“	55	A2	“
18	DD7	“	56	PVC	PVC Compound Unit
19	DD8	“	57	INS	Inspection
20	DD9	“	58	QC	Quality Control Unit
21	DD10	“	59	ST	Storage
22	DD11	“	60	WH	Warehouse
23	DD12	“	61	WIP1	Work-In-Process
24	DD13	Neutral Detailed Drawing	62	WIP2	“
25	DD14	“	63	WIP3	“
26	DD15	“	64	WIP4	“
27	DD16	“	65	FP1	Forklift Parking
28	MCS1	Main Core Stranding	66	FP2	“
29	MCS2	“	67	FP3	“
30	MCS3	“	68	SC	Scrap Center
31	MCS4	“	69	OFF	Main Office
32	MCS5	“	70	MC	CCV Maintenance
33	MCS6	“	71	LOU1	Employee Lounge
34	NCS1	Neutral Core Stranding	72	LOU2	“
35	NCS2	“	73	DOK1	Docking Station
36	SHS1	Sheathing Station	74	DOK2	“
37	SHS2	“			

Table 2. The machines and support areas for the high voltage cable facility

There are 74 literals ($n = 74$) in the literal space Ω for this problem.

- *Interactions among literals:* The constraints of the problem define the interactions among the twelve literals listed above. One indicates causality based on accessibility definition between two literals, and zero otherwise. The causality links are depicted in the $\mathbf{R}$ matrix provided in Table 3.

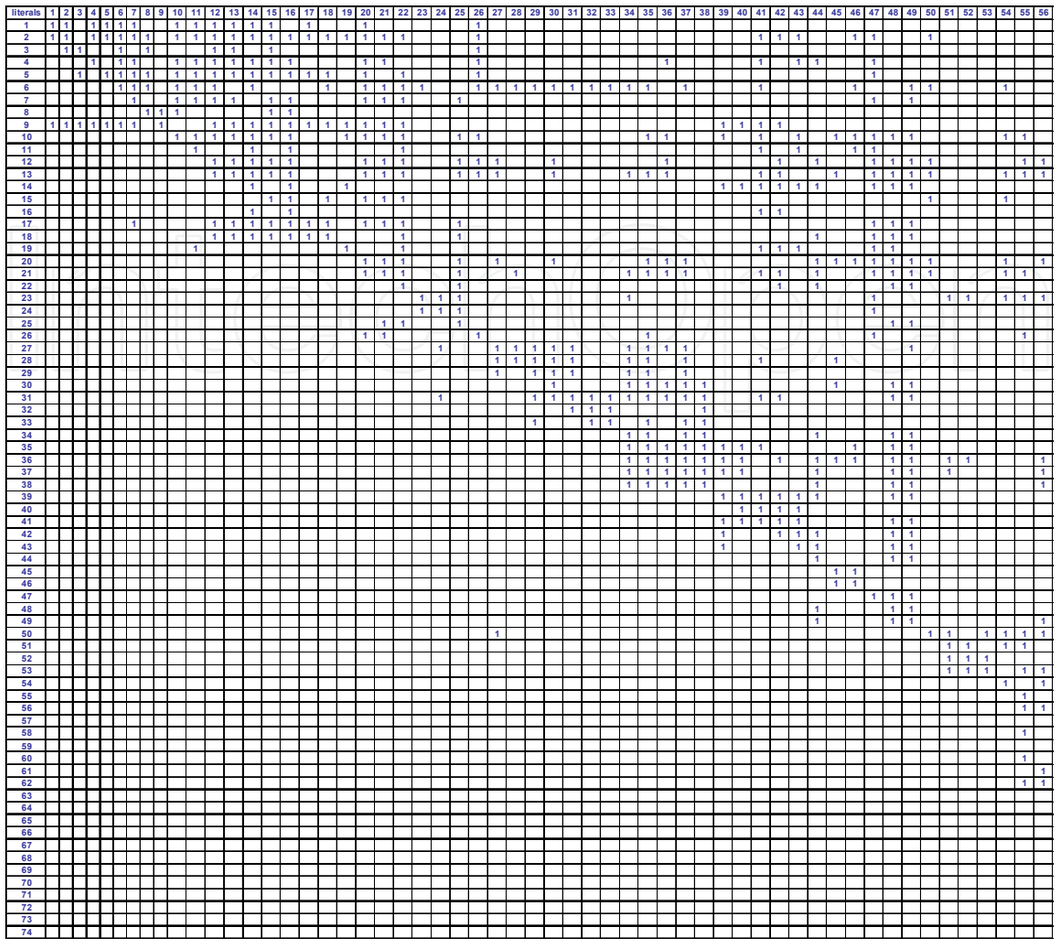

Table 3. The R matrix for the cable manufacturing facility

10.2 Phase II: abstraction of the literal space of the cable manufacturing company

Given the $\mathbf{R}$ matrix for the cable manufacturing facility, the literal space is portioned into mutually exclusive AECs using ECFA. Each cluster of AECs constitutes literals that need to be considered simultaneously. AECs obtained from using ECFA are shown in Table 4.

Classes	Literals (lBB$_{iBB}$)	Number of Literals (NBB$_{kBB}$)	Class Classification
0	0, 1, 2, 3,	4	Transient
1	4, 5, 6, 7, 8, 9, 10, 11	8	Transient
2	12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 61, 65, 34, 35, 62	27	Transient
3	36, 37, 38, 39, 54, 55, 56, 40, 41, 42, 43, 52, 53, 44, 45, 46, 47, 48, 49, 50, 51, 70, 66, 63, 64,,	25	Transient
4	57, 58, 60, 67, 68, 72, 74	7	Absorbing
5	59, 71, 73	3	Transient
6	69	1	Maximal transient

Table 4. The AECs for the cable manufacturing facility

From Table 4, the abstracted literal space ω consists of seven AECs. This reduced the problem tremendously to a manageable size.

- *Aggregate interactions among AECs:* The aggregate interactions among AECs can be obtained using Eq.(9). Accordingly, the $\mathbf{C}$ matrix is constructed and is transformed it to the canonical form $\bar{\mathbf{C}}$ which are provided below.

The entries of $\mathbf{C}$ and $\bar{\mathbf{C}}$ matrix are:

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \bar{\mathbf{C}} = \left(\begin{array}{c|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

10.3 Phase III: constructing the AH for the cable manufacturing case study

In this phase, the interactions among the different AECs are utilized to recursively develop an AH to structure the cable manufacturing facility. As indicated in the methodology, AHs are designed to be loop-free. In terms of the problem in hand, obtaining partial solutions at a given abstraction level need *not* be altered as the process progresses hierarchically to more detailed levels.

Each AEC is assigned to its appropriate abstraction level using LAA as shown in Figure 7. Table 5 illustrates the resultant abstraction hierarchy for the cable manufacturing facility. The levels of the hierarchy indicate the order in which each literal should be introduced to the problem gradually to facilitate loop-free problem execution.

Level	Classes	Literals (lB_{iBB})	Number of Literals (NBB_{kBB})
6	6	69	1
5	5	59, 71, 73	3
4	0	0, 1, 2, 3,	4
3	1	4, 5, 6, 7, 8, 9, 10, 11	8
2	2	12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 61, 65, 34, 35, 62	27
1	3	36, 37, 38, 39, 54, 55, 56, 40, 41, 42, 43, 52, 53, 44, 45, 46, 47, 48, 49, 50, 51, 70, 66, 63, 64	25
0	4	57, 58, 60, 67, 68, 72, 74	7

Table 5. The levels of the Abstraction hierarchy for the high voltage cable facility

Executing the abstraction hierarchy top-down and feeding results to a facility layout routine result in the layout provided in Fig 10.

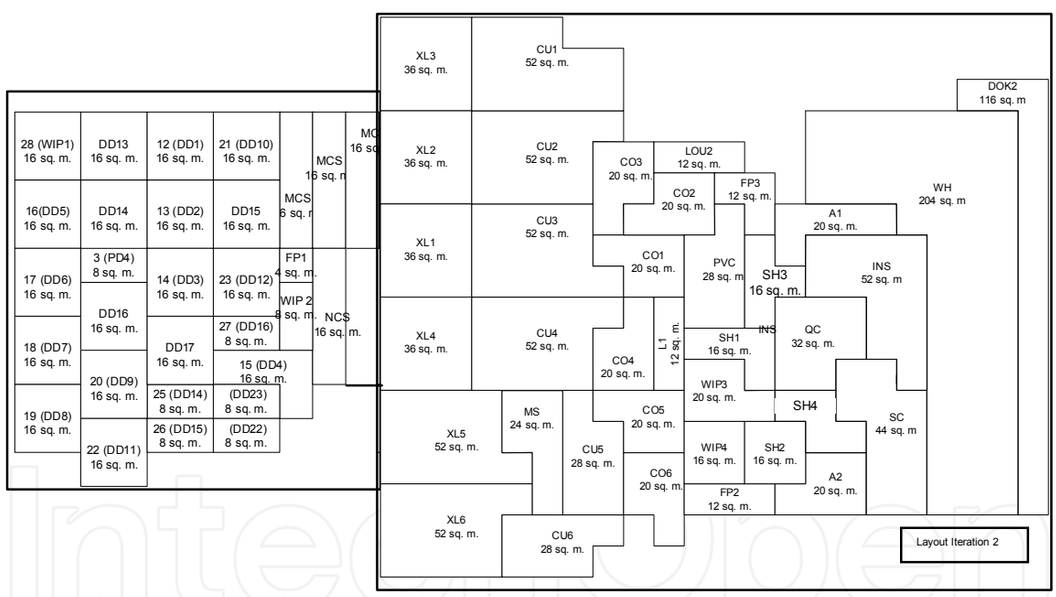

Fig 10. Facility layout for the High voltage cable manufacturing facility

11. Conclusions

This research establishes a rigid foundation and a general platform that produces consistent abstraction spaces and hierarchies applicable to various contexts, especially those involving planning and problem solving. The methodology presented adheres to the efficiency measures and specifications prescribed by the latest advances of AI-based abstraction theory. Yet, our hierarchical abstraction methodology exhibits additional practicality as it integrates the theory of abstraction with the convenient representation scheme of Design Structured Matrices. This expands the application of abstraction theories and enhances their

feasibility to be used in practice. Within the presented methodology, we have also developed several effective methods to efficiently structure and analyze systems to be hierarchically decomposed. These methods were integrated from graph, relation and matrix theories. In addition we have utilized Markov Chains classes' classification methods to identify special behavior in system components and to detect in advance whether or not a system representation is better using hierarchies. The strength of the methodology relies on its ability to structure problems in abstraction hierarchies that result in no backtracking. However, the efficiency of the methodology depends on the context to which it is applied. That is, little gain is expected to be realized when applying the methodology to domains that undergo significant interaction due to the irreducibility problem. The steps of the developed methodology are illustrated in stratifying the design aspects of high voltage cable company into multiple levels of abstraction. This advantageously contributed in introducing the design details of the problem gradually as needed earlier in conceptual stage of planning of the facility. Future research is directed towards quantifying binary relations of literals, developing measures of efficiency and means of eliminating irreducability and inclusion of initial and goal states to the literal space.

12. References

Aleisa, E. (2005). Multilevel Integration of Simulation and Facilities Planning for Large-Scale Systems. *Department of Industrial Engineering*. Buffalo, NY, The State University of New York at Buffalo. Vol., No.

Aleisa, E. (2008). *An Overview Of Multilevel And Hierarchical Methods For Discrete-Event Simulation Of Complex Systems. Industrial Simulation Conference (ISC08)*, Lyon, France.

Aleisa, E. & L. Lin (2008). Abstraction Hierarchies for Engineering Design. *International Journal of Electrical, Computer, and Systems Engineering* 2(1): 20-32.

Aleisa, E. & L. Lin (2009). A Design Structure Matrix Approach For Generating Planning Abstraction Hierarchies. *Kuwait Journal of Science and Engineering (KJSE)*: To appear.

Armano, G.;G. Cherchi & E. Vargiu (2003). Planning by abstraction using HW[]. *Ai(Asterisk)Ia 2003: Advances in Artificial Intelligence, Proceedings*. 2829**:** 349-361.

Bacchus, F. & Q. Yang (1992). *Expected value of hierarchical problem-solving. AAAI-92.*

Browning, T. R. (1999). The Design Structure Matrix. *Technology Management Handbook.* R. C. Dorf. Boca Raton, FL, Chapman & Hall/CRCnetBASE,**:** 103-111.

Browning, T. R. (2001). Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE Transactions on Engineering Management* 48(3 August): 292-306.

Chen, L.-R. & S. Ghosh (1997). Modeling and simulation of a hierarchical, distributed, dynamic inventory management scheme. *Simulation* 68(6 Jun): 340-362.

Christensen, J. (1991). Automatic Abstraction in Planning. *Department of Computer Science*. Stanford, Ca, Stanford University. Vol., No.**:** 153.

Dartmouth College Writing Group. & E. J. Cogan (1958). *Modern mathematical methods and models; a book of experimental text materials.* Ann Arbor, MI.

Debbie, R. (2003). Knowledge-Based System Explanation: The Ripple-Down Rules Alternative. *Knowledge and Information Systems* 5(1): 2.

Deo, N. (1974). *Graph theory with applications to engineering and computer science.* Englewood Cliffs, N.J., Prentice-Hall.

Eppinger, S. D.;D. E. Whitney;R. P. Smith & D. A. Gebala (1994). A Model-Based Method for Organizing Tasks in Product Development. *Research in Engineering Design-Theory Applications and Concurrent Engineering* 6(1): 1-13.

Fox, M. & D. Long (1995). *Hierarchical planning using abstraction. IEE Control Theory and Applications*.

Friske, L. M. & C. H. C. Ribeiro (2006). Planning under uncertainty with abstraction hierarchies. *Intelligent Data Engineering and Automated Learning - Ideal 2006, Proceedings*. 4224**:** 1057-1066.

Gaver, D. P. & G. L. Thompson (1973). *Programming and probability models in operations research.* Monterey, Ca, Brooks/Cole Pub. Co.

Gimenez, O. & A. Jonsson (2008). The complexity of planning problems with simple causal graphs. *Journal of Artificial Intelligence Research* 31: 319-351.

Giunchiglia, F. (1999). Using Abstrips abstractions - Where do we stand? *Artificial Intelligence Review* 13(3): 201-213.

Giunchiglia, F. & T. Walsh (1992). A Theory of Abstraction. *Artificial Intelligence* 57(2-3): 323-389.

Goldin, S. E. & P. Klahr (1981). *Learning and Abstraction in Simulation. International Joint Conference on Artificial Intelligence*, American Assoc for Artificial Intelligence.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26: 191-246.

Holte, R. C. & B. Y. Choueiry (2003). Abstraction and reformulation in artificial intelligence. *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences* 358(1435): 1197-1204.

Hoover, S. P. & J. R. Rinderle (1994). *Abstractions, design views and focusing. 6th International Conference on Design Theory and Methodology American Society of Mechanical Engineers, Design Engineering Division (Publication) DE*, ASME, New York, NY.

Kao, E. P. C. (1997). *An introduction to stochastic processes.* Belmont, Calif., USA, Duxbury Press.

Kemeny, J. G. & J. L. Snell (1960). *Finite markov chains.* Princeton, N.J., Van Nostrand.

Kemke, C. & E. Walker (2006). Planning with action abstraction and Plan Decomposition Hierarchies. *2006 Ieee/Wic/Acm International Conference on Intelligent Agent Technology, Proceedings*: 447-451.

Kiran, A. S.;T. Cetinkaya & J. Cabrera (2001). Hierarchical modeling of a shipyard integrated with an external scheduling application. *Winter Simulation Conference Proceedings* 2: 877-881 (IEEE cat n 01CH37304).

Knoblock, C. (1990). *Learning Abstraction Hierarchies for Problem Solving. AAAI-90*, Boston, MA.

Knoblock, C. (1994). Automatically generating abstractions for planning. *Artificial Intelligence* 68(2 Aug): 243-302.

Knoblock, C. A. (1994). Automatically generating abstractions for planning. *Artificial Intelligence* 68(2 Aug): 243-302.

Lam, K. P. (1996). *Hierarchical Method for Large Scale Two-Dimensional Layout. DET-96*, American Society of Mechanical Engineers.

Levy, A. Y. (1994). *Creating abstractions using relevance reasoning. AAAI-94*, Menlo Park, CA.

Lin, J. T.;K. C. Yeh & L. C. Sheu (1996). A context-based object-oriented application framework for discrete event simulation. *Computers & Industrial Engineering* 30(4): 579-597.

Lu, S. C. Y. & D. K. Tcheng (1991). Building layered models to support engineering decision making. A machine learning approach. *Journal of Engineering for Industry* 113(1): 1-9.

Luger, G. F. (2002). *Artificial intelligence : structures and strategies for complex problem solving*. Harlow, England ; New York, Pearson Education.

Manfaat, D.;A. H. Duffy & B. S. Lee (1998). SPIDA: Abstracting and generalizing layout design cases. *Artificial Intelligence for Engineering Design, Analysis & Manufacturing: Aiedam* 12(2 Apr): 141-159.

Marie, d.;R. Priyang & G. Lise (2008). Learning structured Bayesian networks: combining abstraction hierarchies and tree-structured conditional probability tables. *Computational Intelligence* 24(1): 1.

McCord, K. R. a. E., Steven D. (1993). Managing the Integration Problem in Concurrent Engineering, M.I.T. Sloan School of Management, Cambridge. Vol., No.

McGraw, R. M. & R. A. MacDonald (2000). Abstract modeling for engineering and engagement level simulations. *Winter Simulation Conference Proceedings* 1: 326-334.

Minton, S. (1988). Learning Effective Search Control Knowledge: An Explanation-Based Approach, Carnegie-Mellon University. Vol., No.**:** 231.

Pels, H. J. (2006). Classification hierarchies for product data modelling. *Production Planning & Control* 17(4): 367.

Pidd, M. (1996). *Five simple principles of modelling. Winter Simulation Conference Proceedings*, Publ by IEEE, IEEE Service Center, Piscataway, NJ, USA.

Pidd, M. & R. B. Castro (1998). *Hierarchical modular modelling in discrete simulation. Winter Simulation Conference*, IEEE, Piscataway, NJ.

Praehofer, H. (1996). *An Environment for DEVS-based multi-formalism modeling and simulation in C++. Proceedings of AI, Simulation and Planning in High-Autonomy Systems*, Tucson, AZ.

Reddy, S. Y. (1996). Learning abstract models for system design. *Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing* 10(2): 167-169.

Rogers, J. L. (1996). *DeMAID/GA - An Enhanced Design Manager's Aid for Intelligent Decomposition. 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Seattle, WA.

Russell, S. J. & P. Norvig (1995). *Artificial intelligence : a modern approach.* Englewood Cliffs, N.J., Prentice Hall.

Sacerdoti, E. (1974). Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5(2): 115-135.

Sarjoughian, H. S.;B. P. Zeigler & F. E. Cellier (1998). *Evaluating model abstractions: A quantitative approach. Proceedings of SPIE Enabling Technology for Simulation Science II*, Orlando, FL, United States.

Sebastia, L.;E. Onaindia & E. Marzal (2006). Decomposition of planning problems. *Ai Communications* 19(1): 49-81.

Steward, D. V. (1981). The Design Structure-System - a Method for Managing the Design of Complex-Systems. *Ieee Transactions on Engineering Management* 28(3): 71-74.

Taylor, L. E. & M. R. Henderson (1994). *Roles of features and abstraction in mechanical design. 6th International Conference on Design Theory and Methodology American Society of Mechanical Engineers, Design Engineering Division (Publication) DE*, New York, NY, ASME.

Warfield, J. N. (1973). Binary Matrices in System Modeling. *Ieee Transactions on Systems Man and Cybernetics* SMC3(5): 441-449.

Yang, Q., Tenenberg, J (1990). *Abtweak: Abstracting a Nonlinear, Least Commitment Planner. AAAI-90*, Boston, MA.

Yassine, A.;D. Falkenburg & K. Chelst (1999). Engineering design management: an information structure approach. *International Journal of Production Research* 37(13): 2957-2975.

Zeigler, B. P. (1976). *Theory of modelling and simulation.* New York, Wiley.

Zeigler, B. P. (1987). Hierarchical, Modular Discrete-Event Modelling in an Object-Oriented Envirionment. *Simulation* 49(5): 219-230.

Zeigler, B. P.;H. Praehofer & T. G. Kim (2000). *Theory of modeling and simulation : integrating discrete event and continuous complex dynamic systems.* San Diego, Academic Press.

Zucker, J. D. (2003). A grounded theory of abstraction in artificial intelligence. *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences* 358(1435): 1293-1309.

New Advanced Technologies

Edited by Aleksandar Lazinica

ISBN 978-953-307-067-4
Hard cover, 350 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

This book collects original and innovative research studies concerning advanced technologies in a very wide range of applications. The book is compiled of 22 chapters written by researchers from different areas and different parts of the world. The book will therefore have an international readership of a wide spectrum.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Esra Aleisa Ph.D. (2010). Abstraction Hierarchies for Conceptual Engineering Design, New Advanced Technologies, Aleksandar Lazinica (Ed.), ISBN: 978-953-307-067-4, InTech, Available from: http://www.intechopen.com/books/new-advanced-technologies/abstraction-hierarchies-for-conceptual-engineering-design

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen