Journal of the Korean Institute
of Industrial Engineers
Vol.22, No.3, September, 1996



# 동시성공학 모형의 계층적 제약식 네트워크 표현 방법론*

# Hierarchical Constraint Network Representation of Concurrent Engineering Models*

김 영 호**

Yeongho Kim**

**Abstract**

Constraint networks are a major approach to knowledge representation in Concurrent Engineering (CE) systems. The networks model various factors in CE as constraints linked by shared variables. Many systems have been developed to assist constraint network processing. While these systems can be useful, their underlying assumption that a solution must simultaneously satisfy all the constraints is often unrealistic and hard to achieve. Proposed in this paper is a hierarchical representation of constraint networks using priorities, namely Prioritized Constraint Network (PCN). A mechanism to propagate priorities is developed, and a new satisfiability definition taking into account the priorities is described. Strength of constraint supporters can be derived from the propagated priorities. Several properties useful for investigating PCN's and finding effective solving strategies are developed.

## 1. Introduction

Recently, it has been widely conceived that Concurrent Engineering (CE) can provide a competitive edge for companies that have confronted with rapidly changing market requirements, such as shortening of product life-cycles, high quality and low cost products,

* 이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

** 서울대학교 산업공학과

diversity of customer demand, and so on. CE is generally recognized as a practice to bridge the gap between designing a product and other various life-cycle activities, such as manufacture, assembly, test, and maintenance, at an early stage of design. The emphasis in the development of CE systems is on the coordination of the various aspects involved in to eliminate problems due to lack of communication and compatibility among different areas of concern. The primary capability of the CE systems is to handle different, often conflicting goals, that need to be integrated into a working whole by taking many constraints restricting the design into account.

Since the influence of the early design stage on the product's overall cost, performance, and quality is usually much more significant than that of the later stages, many researchers have proposed a number of different approaches to help designers with CE. Many of them have mainly addressed techniques to represent CE knowledge and to process the knowledge to support CE activities. Constraint networks are a major approach to knowledge representation in CE systems [20]. The background and operation of the networks is reviewed in [9, 20].

While the constraint networks appear to be a promising approach, it is a main drawback that all of the constraints constituting a CE model are dealt with equally importantly. However, in reality, some constraints are important, but others are less. Furthermore, it is often very hard to find a solution that simultaneously satisfies all the constraints provided. Sometimes constraints are conflicting with each other so that some of them may have to be ignored to satisfy the others. It is especially true when a problem gets larger or over-constrained. Indeed a programmer relies on relaxing less important constraints when all of them cannot be easily satisfied. If the constraints can be ranked as their importance, the relaxation can be performed more systematically. This notion of importance of constraints leads us to the consideration of prioritizing constraints. Prioritization of constraints has been studied by a few authors including [4, 10, 19]. They have suggested several hierarchical schemes to compare different solutions based on priorities.

## 2. Constraint Network Models and Concurrent Engineering

### 2.1. Constraint Satisfaction Problem

A constraint is a relation among a set of variables. A relation specifies allowed relationships among the objects represented by the variables. A variable can be instantiated with a value taken from its domain. A constraint satisfaction problem is a problem of finding an assignment of values to the variables, such that all the constraints are simultaneously satisfied. Formal definitions of the problem are found in much literature including [17]. The following definition is adapted from [4];

***Constraint Satisfaction Problem:*** A constraint satisfaction problem involves a set of $n$ variables, $X = \{X_1, X_2, \cdots, X_n\}$, and a set of $m$ constraints, $C = \{C_1, C_2, \cdots, C_m\}$. A variable $X_i$ has its domain, $\Omega_i$, which is the set of values that the variable can have. A constraint $C_j$ is a $k$-ary relation on $X_j' = \{X_{j_1}, X_{j_2}, \cdots, X_{j_k}\} \subseteq X$, i.e., $C_j(X_{j_1}, X_{j_2}, \cdots, X_{j_k})$, and is a subset of the Cartesian product $\Omega_{j_1} \times \Omega_{j_2} \times \cdots \times \Omega_{j_k}$. An assignment of values to all the variables, such that all the constraints are simultaneously satisfied is called a solution. A constraint satisfaction problem is a problem of finding such a solution or all the solutions.

Constraints are useful in many applications, such as geometric layouts, physical simulations, user interface design, document formatting, algorithm animation, and design and analysis supporting mechanical devices and electrical circuits [4, 19].

Table 1. Various constraint network representations

| Networks | Author(s) | Const. Type[1] | Domain[2] | N[3] | Objective |
|---|---|---|---|---|---|
| Primal/Dual | Dechter & Pearl ('89) [3] | R | DC/I | n | To identify a backtrack free tree structure |
| Expanded Net | Haralick ('78) [6] | R | D/F | 2 | To show complete relations b/w two sets of domain values |
| Montanari's Net | Montanari ('74) [13] | R | D/F | 2[4] | To derive minimal network with algebraic operations |
| Hypergraph | Montanari & Rossi ('87) [14] | R | DC/I | n | To represent constraints involving multi-variables |
| Consistency Net | Freuder ('78) [5] | R | D/F | n | To preprocess networks with generalized consistency theory |
| Operands Net | Sussman & Steele ('80) [18] | E | C/I | n | To represent equality constraints for mainly electrical circuit design |
| Pr/T Net | Murata & Zhang ('88) [15] | L | D/F | n | To separate logic from control Horn clause logic program (concurrence & parallelism) |
| Bipartite | Young et al. ('92) [20] | G | CD/I | n | To visualize general constraints for CE knowledge representation |
| Meta-Net | Kim & O'Grady ('95) [9] | G | CD/I | n | To study structural and operational properties of CE constraint networks |

1) R: Relation, E: Equation, L: Logic, G: General
2) C: Continuous, D: Discrete, DC: Discrete and/or continuous, F: Finite, I: Infinite
3) Number of variables in a constraint
4) Expandable to $n$ variables ($n > 2$)

### 2.2. Constraint Network Representations

If a set of constraints is represented in a form of network, it is called a constraint network. A constraint network can be considered as a graphical version of a list of constraints. The constraint network is a convenient form of expressing declarative knowledge, allowing a designer to focus on local relationships among entities in the domain that he/she considers. A set of constraints collectively defines a feasible solution space where the modeled system can be set up. The constraints that restricts a particular object are clustered, so that their interrelationships can be easily identified.

There has been a body of literature in constraint networks, including [12] and [16]. Depending on the objects and their interrelationships that one may want to represent, networks of constraints have different forms. Usually nodes are utilized to represent objects, such as variables, constraints, and domain values. Existence of a relationship between two objects is represented as an arc connecting the relevant nodes. Literature dealing with various ways of constraint network representations is presented in Table 1. A more detailed discussion can be found in [9] as well as the references cited.

### 2.3. Constraint Networks and CE

A CE design problem can be modeled by a constraint network. Any restriction on a design or design knowledge can be declaratively specified by a form of constraint. It is an advantage of the constraint-based approach that the declarativity allows for designers to focus only on local relationships among entities in the design domain [20]. Since a constraint involves a set of relation symbols, variables, constants, and functional symbols, a CE COnstraint Network (CECON) model is written by a conjunction of such constraints as follows;

$$(1)\ a\_feasible\_design(T) :- \Lambda_{i=1}^{m} \phi_i(\Gamma_i, T_i, A_i, F_i)$$

where $m$ is the total number of constraints, $T$ is the set of all the variables, and $\phi_i$ is a constraint defined by $\Gamma_i$, $T_i$, $A_i$, and $F_i$, which are a set of relation symbols, variables, constants, and functional symbols, respectively. The purpose of the CECON model is to find a solution that satisfies every constraint.

After Sutherland's Sketchpad, a constraint-based system for drawing and editing pictures on a computer, there have been a number of constraint programming languages, such as ThingLab [1], MOLGEN [17], CLP( $\mathcal{R}$ ) [7], etc. Many existing constraint languages are reviewed in [9, 12]. Most of them are heavily dependent on application domains, such as graphical user interface design, molecular genetic experimental design, puzzle problems, mechanical design and analysis, etc. Therefore, their capability to handle constraints are limited to the type relevant to their domain. SPARK is somewhat more general systems for CE [20]. With its capabilities, such as user-aided con-

straint satisfaction, bi-directional inference, violation detection and correction suggestion, SPARK can effectively support designers.

The constraint definitions employed in the above approaches commonly use a triple (relations, objects, and values) whereby no room is provided for expressing how much important a constraint is. If a model is over-constrained, as means that it has no feasible solution, one may have to relax some of the constraints. On the other hand, for an under-constrained model wherein a large number of solutions exist, one may try to choose a better solution. However, the definitions provide no sound criteria either to choose constraints to be relaxed or to sort out a better solution. Such a notion is prevailing in CE design problems. Thus, the existing definitions and the constraint languages based on the definitions have inherent limitations when applied to the CE problems. Here in this paper, the fourth entity, namely *priority*, is attached to the definition shown in (1), so as to represent the importance of constraints.

## 3. Schemes of Hierarchical Representations

Decomposition is a promising approach to representing exceedingly large and complex systems whose whole profile can't be readily seen with a single enormous representation. The decomposition is often represented by a hierarchically organized structure. A hierarchical formalism places emphasis on the arrangement of entities in hierarchies or on the hierarchical relationships between entities. Very often, the structure can be represented in a hierarchical tree form, where a node is associated with an entity (activity) and the node is divided into subnodes along with arcs. A node at a lower level represents a more detailed entity or a more basic concept of the represented system, while a node at a higher level is a more generalized entity.

A *conjunctive hierarchy* is perhaps the most common structure where a node is disaggregated into several subnodes. Each subnode represents a proper subset of its parent node in a more detailed manner. In order to find a solution for a node, all of its subnodes must have solutions. The other common hierarchical structure is a *disjunctive hierarchy*. In contrast with the conjunctive hierarchy, the distinct subnodes sharing a common parent represent alternative methods of satisfying the parent. A solution always exists for a node if at least one of its subnodes is solvable. A *hybrid hierarchy* is of somewhat general since it uses themes from both conjunctive and disjunctive hierarchies. AND/OR trees [20] are useful for representing such structures. A simple AND/OR tree is illustrated in Fig. 1. Conjunctive and disjunctive hierarchies are respectively associated with OR and AND arcs. An AND arc is indicated by an arch connecting all the subnodes. The properties of AND/OR trees are discussed in [9].

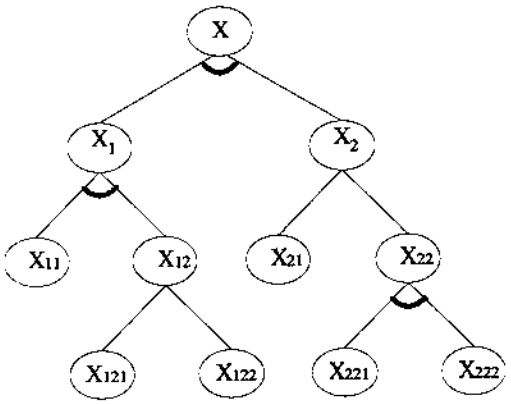



Fig. 1. AND/OR Tree

The above AND/OR hierarchy can be further generalized using priorities, namely *priority hierarchy*. This is useful for the case where a priority can be assigned to each node. The priority of a node can be determined by its relative degree of importance or preference. The nodes with the highest priority are called *required nodes*. The other nodes having lower priorities are called *preferential nodes*. The priority hierarchy provides a more generalized OR scheme since the subnodes can be ranked with respect to their priorities. A required nodemust be solved, and thus determines the feasibility of the problem. This node directly follows a conjunctive hierarchy. On the other hand, a preferential node has a similarity to a disjunctive hierarchy since it can remain unsolved. However, it is considered to be desirable if the node is solved. The preferential nodes provide us a way to deal with the quality of solutions. The more highly prioritized constraints a solution satisfies, the better the solution is.

## 4. Prioritized Constraint Networks (PCN)

A new method that can combine the concept of priority hierarchy and a constraint network representation is proposed. Let $p_i$ denote the priority assigned to constraint $\phi_i$, and $\Phi_i = p_i \phi_i$. A priority can be a numerical value in the interval [0, 1]. A constraint whose priority is equal to 1 is called a required constraint. Otherwise, a preferential constraint. If all the priorities used in a constraint network model are required, then it is equivalent to a classical non-prioritized model. Instead of the numeric priorities, symbols, such as [required, very strong, strong, weak, very weak], can also be used. Now, a CECON model is restated by a conjunction of prioritized constraints as follows;

(2) $a\_feasible\_design(T) :- \Lambda_{i=1}^{m} p_i \phi_i(\Gamma_i, T_i, A_i, F_i)$

It is a requirement of a solution that all the required constraints be satisfied, whereas it is optional to satisfy preferential constraints. The purpose of this model can be to find a solution satisfying as many highly prioritized constraints as possible. In order to compare the solutions, various schemes, such as sum of $\bar{p}_i$'s (where $\bar{p}_i$ is the priority of a satisfied constraint), weighted sum of $\bar{p}_i$'s, sum of square of $\bar{p}_i$'s, etc., can be used.

### 4.1. Constraint satisfaction in non-prioritized models

A constraint is *satisfied* if the variable instantiation meets the constraint specification. Otherwise, *unsatisfied*. If the instantiation is not enough to assess the constraint, it is *indeterminate*. The fact that constraint $\phi$ is satisfied by interpretation $I$ and variable instantiation $U$ is written as $\models_I \phi[U]$. Many constraint satisfaction algorithms, including search methodologies, consistency checking, local propagation, mathematical programming, etc., have been developed to solve CSP [9, 12]. However, unfortunately, the problem is known as an NP-hard problem unless special cases.

The system considered in this paper does not employ automatic constraint satisfaction. Rather, it supposes user-aided constraint satisfaction [20], whereby the system and a user cooperate with each other to solve the problem. The system provides a deductive mechanism based on local propagation of known states [1, 12, 18], which is one of the simplest and easiest constraint satisfaction mechanisms to implement. A constraint itself enforces satisfying the relationship specified on its variables. If the information propagated into a constraint node along its arcs is enough to deduce a new value, the value is computed. This value may be able to be used for other deduction steps. Whenever this deductive process can't repeat further, the user triggers a new round of the process.

Local propagation alone can't solve every CSP problem. However, it is very powerful when a constraint network possesses many algebraic equations and it is sparse, as [18] has pointed out. Since it requires no predetermined sequence of processing unknown variables and constraints, declarativity can be effortlessly implemented. It also facilitates keeping dependency relationships among constraints and providing explanation about the results of system's inference, which is particularly important in the user-aided constraint satisfaction [9].

### 4.2. Priority propagation

Local propagation in a traditional constraint network disseminates known values over the network, whereas that in PCN propagates priorities as well. Consider a prioritized constraint $\Phi(\tau_1, \cdots, \tau_n)$. Suppose that a value for $\tau_k$ is able to be deduced from $\Phi$ when variables in $\tau$, which is a proper subset of $(\tau_1, \cdots, \tau_n)$, are known. Then, this value is propagated into other constraints having $\tau_k$, and it carries the minimum among the propagated priorities of $\tau$ and the priority of $\Phi$. The priority accompanied by $\tau_k$ is defined as follows.

$$(3)\quad g_k = \mathrm{Min}\{p,\ \mathop{\mathrm{Min}}_{j}\{g_j \mid \tau_j \in \tau\}\}$$

This propagated priority $g_k$ is one of the constraint priorities which have contributed to the deduced value. Since, in this paper, a priority represents the importance of a constraint, $g_k$ is the priority of the least important one among the constraints that have affected

the overall deducing process.

### 4.3. Satisfiability definition of prioritized constraints

To deal with a prioritized constraint, its semantics need to be explicitly specified. The concept of *level of satisfiability* is introduced on the top of traditional satisfiability definitions (*satisfied, unsatisfied, indeterminate*). The level depends on the priorities of constraints which have contributed to the variable instantiation in the constraints. The history of such priority information is maintained by priority propagation. For simple discussion, the notations and terminology in Genesereth and Nilsson [8] are generally used, but modified to represent the level of satisfiability.

If prioritized constraint $\Phi$ is satisfied by an interpretation $I$ and a variable instantiation $U$ and if its level of satisfiability is $q$, then $\Phi$ is *q-level satisfied* and denoted by $|=_I^q \Phi[U]$. If it is unsatisfied, $\Phi$ is *q-level unsatisfied* and $|\neq_I^q \Phi[U]$. If the interpretation is not enough to assess the constraint, it is *indeterminate* as in the traditional definition.

First, consider an atomic constraint which involves only one relation symbol, such as =, $>$, $<$, $\geq$ , $\leq$, or a predicate. The satisfiability of a prioritized atomic constraint is formally defined as follows.

(4) $|=_I^q p\,\phi(\tau_1, \cdots, \tau_n)[U]$ iff $\langle T_{IU}(\tau_1), \cdots, T_{IU}(\tau_n)\rangle \in I(\phi)$, *where* $q=\mathrm{Min}(g_1, \cdots, g_n)$

Since the level of satisfiability, $q$, is the minimum of the propagated priorities $g_i$'s, it is simple to calculate and intuitively acceptable.

The followings are satisfiability definitions for various forms of prioritized non-atomic constraints which involve at least one relation symbol, as $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\Rightarrow$ (implication), $\Leftrightarrow$ (equivalence), universal quantification, and existential quantification. The level of satisfiability of a non-atomic constraint is driven from those of its component atomic constraints.

(5) $|=_I^{q_1}(\neg p\,\phi)[U]$ iff $|\neq_I^{q_2} p\,\phi[U]$, where $q_1=1-q_2$

(6) $|=_I^q(p(\phi_i \wedge \phi_j))[U]$ iff $|=_I^{q_i}\phi_i[U]$ and $|=_I^{q_j}\phi_j[U]$, where $q=\mathrm{Min}\{q_i,q_j\}$

(7) $|=_I^q(p(\phi_i \vee \phi_j))[U]$ iff $|=_I^{q_i}\phi_i[U]$ or $|=_I^{q_j}\phi_j[U]$, where $q=\underset{K\in\theta}{\mathrm{Max}}\{q_K\}$ and $\theta=\{K \mid |=_I^{q_K}\phi_K[U]\}$

(8) $|=_I^q(p(\phi_i \Rightarrow \phi_j))[U]$
iff $|\neq_I^{q_i}\phi_i[U]$, where $q=q_i$, or
iff $|=_I^{q_i}\phi_i[U]$ and $|=_I^{q_j}\phi_j[U]$,
where $q=\mathrm{Min}\{q_i,q_j\}$

(9) $|=_I^q(p(\phi_i \Leftrightarrow \phi_j))[U]$ iff $|=_I^{q_1}(\phi_i \Rightarrow \phi_j)[U]$, $|=_I^{q_2}(\phi_i \Leftarrow \phi_j)[U]$, where $q=\mathrm{Min}\{q_1,q_2\}$

(10) $|=_I^q(p(\forall v\,\phi))[U]$ iff for any object $d_j$ in the universe of discourse $|=_I^{q_j}\phi[V]$ where $V(v)=d_j$ and $V(v)=U(\mu)$ for $\mu \neq v$, where $q=\mathrm{Min}\{q_1, \cdots, q_{|I|}\}$

(11) $\models_I^q(p(\exists v\,\phi))[U]$ iff for some object $d_j$ in the universe of discourse $\models_I^{q_j}\phi[V]$ where $V(v)=d_j$ and $V(v)=U(\mu)$ for $\mu \neq v$, where $q=\underset{K\in\theta}{\mathrm{Min}}\{q_K\}$, and $\theta=\{K \mid \models_I^{q_K}\phi[V]\}$

### 4.4. Overriding mechanism

While processing a plain constraint network, the local propagation propagates values through the network. This traditional propagation mechanism is concerned with satisfiability of the constraints involved. Since a solution, by definition, should satisfy all the constraints, various user inputs may have to be tried until every violation is resolved away. As the constraint network is very large and complex, the job becomes more and more difficult and cumbersome. When the mechanism encounters such a constraint violation too often, it may be considered to explicitly relax some of the constraints. This, however, is not an easy task for the user to perform without knowing the importance of constraints.

In comparison with plain constraint network processing, the local propagation in PCN has a radically differed feature. This new mechanism involves both value and priority propagation. The priority propagation is used to evaluate the levels of satisfiability explained in a previous section. The value propagation is the same with that in the traditional one when no violation takes place. However, when it produces violations, the value propagation is controlled by the priority propagation. A violation is caused by propagating multiple values into the same variable. One value is randomly chosen in a traditional propagation. The priority propagation makes the value with the highest priority override the others. This value guarantees that the most highly prioritized constraint is always satisfied, while allowing less important ones to be unsatisfied. Recall that it is unnecessary to satisfy all the constraints with PCN. The major concern is discovering a better solution by satisfying as many highly prioritized constraints as possible. Actually, the priority propagation provides an implicit constraint relaxation mechanism.

### 4.5. An illustrative example

An informal illustrative example is presented in Fig. 2. Consider designing a triangular plate shown in Fig. 2 (a). Two sides, $A$ and $B$, must be perpendicular, and the length of side $A$ be 16 Cm. It is suggested that the length of side $B$ is $14Cm \leq B \leq 20Cm$. Angle needs to be at least 45° , but less than 55° . The plate can be cut out from a steel strip currently available at stock whose width is 18 Cm. For ease of cutting operations, the designer wants $A=B$. These design specifications are represented in constraint forms as in Fig. 2 (b). Priorities assigned to each constraint appear in parentheses. The strings in 〈 〉 are constraint ID.

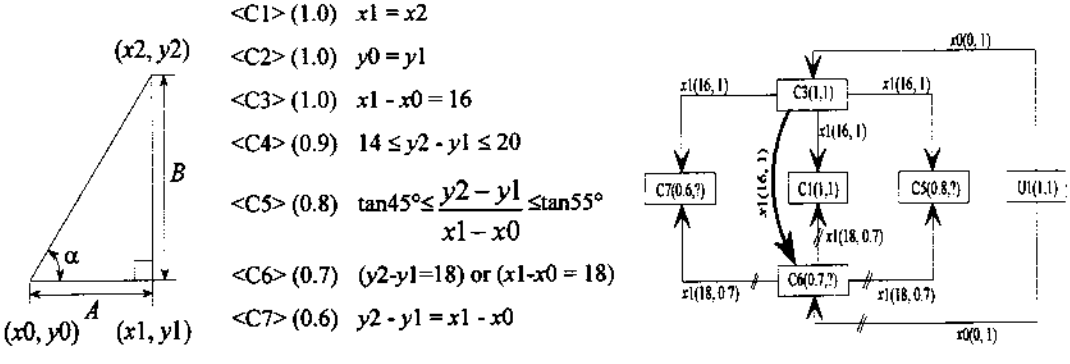
A graph is an effective tool to show local propagation as depicted in Fig. 2 (c). Shown in a box is either a constraint or a user input.

Each of these is identified by ID with its prioritan and level of satisfiability. 'U1' indicates a user input. It is assumed that the user input has its priority of 1.0. Directional arcs represent the paths of value and priority propagation. Each arc is associated with a propagated value and its propagated priority. As soon as a user asserts $x0=0$, constraints ⟨C3⟩ and ⟨C6⟩ deduce $x1=16$ and $x1=18$, respectively. These two values for the same variable propagate into constraints ⟨C7⟩, ⟨C1⟩, and ⟨C5⟩. Since the priority of $x1=16$ is greater than that of $x1=18$, the former overrides the latter. Therefore, three arcs transmitting $x1=18$ and one arc transmitting $x0=0$ are disconnected. Into ⟨C6⟩ does $x1=16$ also propagate, as is represented by a thick arc. This propagation forces the previously satisfied constraint to be indeterminate. Constraint ⟨C1⟩ can deduce $x2=16$, but has no place to be propagated. Resultantly, the value and priority propagations end up with the satisfaction of ⟨C3⟩ and ⟨C1⟩, whose levels of satisfiability are 1. Other constraints are indeterminate yet. This example shows that the overriding mechanism can satisfy more important constraints if possible. If $x1=18$ were used instead, constraint ⟨C3⟩ which is more important than ⟨C6⟩ would be violated. This implicit constraint relaxation is further discussed in the next section.

## 5. Properties IN PCN

### 5.1. Strength of supporters

In order to assess constraint $\Phi(\tau_1, \cdots, \tau_n)$, all or some of the variables may have to have values. The values are provided by either local propagation or user input. If $\Phi'$ is the constraint from which $\tau_i$ receives a value, $\Phi'$ is called a 1-*step* or *direct supporter* of $\Phi$. The list of all the direct supporters of $\Phi$ is denoted by $S^1(\Phi)$. Let $\Phi''$ be a direct supporter of $\Phi'$, that is $\Phi'' \in S^1(\Phi')$. $\Phi''$




(a) Triangular plate (b) Constraint model (c) Propagation and Overriding

Fig. 2. An Illustrative Example

is called a 2-*step supporter* of $\Phi$. The list of all 2-step supporters of $\Phi$ is denoted by $S^2(\Phi)$. This can be similarly generalized into $S^n(\Phi)$, which is the list of *n-step supporters.* An element in $S^n(\Phi)$ is, obviously, a direct supporter of an element in $S^{n-1}(\Phi)$. There exist $(n-1)$ intervening constraints between a constraint in $S^n(\Phi)$ and $\Phi$. If $\Phi$ is a user input, it is self-supportive and, in other words, has no supporters. $S^{\cup}(\Phi)$ denotes the set of all $K$-step supporters, $k \geq 1$, *i.e.*, $S^{\cup}(\Phi) = \cup_{i=1}^{\infty} S^i(\Phi)$

Consider a constraint $\Phi$ which has several supporters. Some of them are more important than $\Phi$, others are less. Suppose that the priority of $\Phi$ is $p$ and its level of satisfiability is $q$. Recall $q$ is the minimum of the propagated priorities. Comparing $p$ and $q$ provides a useful insight into the relationships between the constraint and its supporters. If $q \geq p$, which means all the supporters of $\Phi$ are more important than $\Phi$, the constraint is *strongly supported.* Symmetrically, if $q < p$, $\Phi$ is *weakly supported.* For the latter case, at least one supporter is less important than $\Phi$.

Let's define several notations used in the next section. $Q^i(\Phi)$ is the list of levels of satisfiability, such that its $k$-th member is the level of satisfiability of the $k$-th member of $S^i(\Phi)$. Similarly, $P^i(\Phi)$ is the list of priorities of the supporters. Symbol $\propto$ is a relation, such that $[a_1, \cdots, a_I] \propto [B_1, \cdots, B_I]$ if $a_i \leq b_{i_j}$ for $\forall i,j$ where $B_I = [b_{i1}, b_{i2}, \cdots, b_{iJ_i}]$. For example, [1, 2, 3] $\propto$ [[1, 2], [2], [5, 3, 4]].

### 5.2. Properties of supporters

In this section several properties of supporters are discussed. The readers who may need formal proofs can refer to [11]. The next property is concerned with the fact that levels of satisfiability decrease monotonically as propagation proceeds.

**Property 1**: Let $\psi$ be a list of constraints that are assessed. Then, $Q^{i-1}(\psi) \propto Q^i(\psi)$ for $\forall j$.

This is directly resulting from the definitions of level of satisfiability, priority propagation, and overriding mechanism.

A strongly supported constraint implies that all of its supporters are satisfied at least the level of its own priority. This observation can be proved by Property 1.

**Property 2**: For a constraint $\Phi(=p\,\phi)$ in a PCN, any element in $Q^{\cup}$ is greater than or equal to $p$ if and only if $\Phi$ is strongly supported.

Since $\Phi$ is strongly supported, $p \leq q = Q^0$ by definition. Property 1 says that $Q^0 \propto Q^1 \propto Q \cdots \propto Q^{\infty}$ Obviously, any element in $Q^{\cup}$ is greater than or equal to $p$. Proving the other side is straightforward.

The propagated priority of a variable is determined by the corresponding supporter. Thus, a strongly supported constraint is supported by the constraints whose levels of satisfiability are as large as its priority. A designer may want constraints to be not only satisfied but strongly supported. Suppose that constraint $\Phi$ be unsatisfied and strongly supported. This status is determined by the constraints which

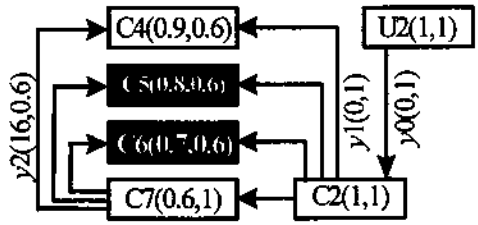
are more important than $\Phi$. If one wants to satisfy this constraint, some of more highly prioritized constraints should be sacrificed. However, this clearly deteriorates the quality of solution. Therefore, strongly supported and unsatisfied constraints are automatically relaxed by the priority propagation mechanism. In other words, the constraint is internally removed from a PCN. This is beneficiary particularly when the PCN is an over-constrained model.

From Property 2, we can also show that the priority of a strongly supported constraint is less than or equal to all of its supporters' priorities.

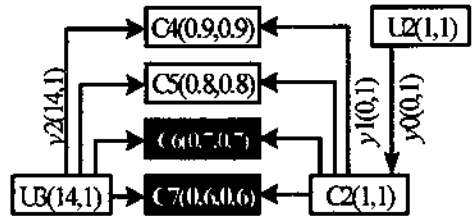
**Property 3**: For a constraint $\Phi$ in a PCN, if the priority associated with any constraint in $x^{\cup}$ is greater than or equal to $p$, $\Phi$ can be strongly supported.

Noting that $q^{(j-1)} \leq g_K^{(j-1)} = \text{Min}\,(p^{(j)}, \text{Min}(g_1^{(j)}, \cdots, g_L^{(j)})) \leq p^{(j)}$, the proof is straightforward. This provides a strategy of triggering local propagation. In order to find a solution that makes every constraint strongly supported, the most important should be considered first. Its results carry highly prioritized values into other less important ones. Otherwise, it is not possible $\Phi$ to strongly support all the constraints.

Finally, for a weakly supported constraint $\Phi$, at least an element in $Q^1(\Phi)$ is less than $p$. In other words, $\Phi$ is currently affected by the constraints whose levels of satisfiability are less than $p$. When $\Phi$ is satisfied, its justification is supported by a less important constraint $\Psi$. The status, however, can be changed when more important constraints come into play and provide highly prioritized variable instantiation. This overrides the effect of $\Psi$. If, on the other hand, $\Phi$ is unsatisfied, the violation is undesirable since less important constraints are satisfied. Now, the overriding mechanism may be convened by a user input. The input which forces $\Phi$ to be satisfied but may cause to be violated can improve the solution quality.




(a) Weakly supported constraint




(b) User input improving solution quality

Fig. 3. Overriding a weakly supported constraint

Consider the example shown in Fig. 2 again. After the propagation in Fig. 2 (c), the user may initiate another local propagation by instantiating $y0$ with 0. This results in violating ⟨C5⟩ and ⟨C6⟩ as shown in Fig. 3 (a). Notice that both of these are weakly supported. Since ⟨C5⟩ is more important than ⟨C7⟩, the user may want to satisfy the former rather than the latter. Inputting $y2$ = 14 with a higher priority

than 0.6, the user can force the input value to override the propagated value. Now, constraint ⟨C5⟩ is satisfied, but ⟨C7⟩ violated. The resulting propagation is depicted in Fig. 3 (b).

## 6. Conclusion

Representation and reasoning are the twin pillars of AI-based systems. The criteria determining the usefulness of a system, such as range of applicability, limitations, and ease of use, are very much dependent on the representation methods and reasoning schemes used. PCN is proposed as an approach to representing importance of constraints. PCN can provide several advantages over traditional constraint networks through facilitating systematic handling of the constraints. First, PCN can greatly improve the expressive power of a constraint network representation. Second, it gives more flexibility in dealing with constraint satisfaction. Satisfying all the constraints is not necessary anymore while providing a basis of both choosing constraints to be relaxed and comparing different solutions. Third, an overriding mechanism forces less important constraints to be violated to satisfy important ones. This is especially helpful under an over-constrained problem. Finally, but not less importantly, several properties associated with PCN can be useful for investigating PCN and developing effective solving strategies.

## REFERENCES

[1] Borning, A.,"The Programming Language Aspects of ThingLab - a Constraint-Oriented Simulation Laboratory,"*ACM Trans. on Prog. Lang. and Sys.*, Vol. 3, No. 4, pp. 353-387, 1981.

[2] Dechter, R. and Pearl, J.,"Network-Based Heuristics for Constraint-Satisfaction Problems,"*AI*, Vol. 34, pp. 1-38, 1988.

[3] Dechter, R. and Pearl, J.,"Tree Clustering for Constraint Network,"*AI*, Vol. 38, pp. 353-366, 1989.

[4] Freeman-Benson B. et al.,"An Incremental Constraint Solver,"*Comm. of the ACM*, Vol. 33, No. 1, pp. 54-63, 1990.

[5] Freuder, E. C.,"Synthesizing Constraint Expressions,"*Comm. of the ACM*, Vol. 21, No. 11, pp. 958-966, Nov. 1978.

[6] Haralick, R. M., Davis, L. S., and Rosenfeld, A.,"Reduction Operations for Constraint Satisfaction,"*Info. Sci.*, Vol. 14, pp. 199-219, 1978.

[7] Heintze, N. et al.,"CLP() and Some Electrical Engineering Problems,"*P. of 4-th Intl. Conf.*, Vol. 2, pp. 675-703, 1987.

[8] Genesereth, M. R. and Nilsson, N. J., *Logical Foundations of AI*, Morgan-Kauffmann, CA, 1988.

[9] Kim, Y. and O'Grady, P.,"A Methodology for Analyzing Large-Scale Concurrent Engineering Systems,"*IJPR*, 1995, (In press).

[10] Kim, Y.,"Prioritized Constraint Network Representation and Processing for Concur-

rent Engineering Models,"*P. of 18th Intl. Conf. on Com. & IE*, Shanghai, China, (to appear soon).

[11] Kim, Y.,"Priority Propagation, Overriding Mechanism, and Its Properties,"Internal Report at Department of Industrial Engineering, Seoul National University, Korea, 1995.

[12] Leler, W., *Constraint programming languages*, Addison Wesley, Reading, Mass., 1988.

[13] Montanari, U.,"Networks of Constraints: Fundamental Properties and Applications to Picture Processing,"*Info. Sci.*, Vol. 7, pp. 95-132, 1974.

[14] Montanari, U. and Rossi, F.,"An Efficient Algorithm for the Solution of Hierarchical Networks of Constraints,"In Graph-Grammars and Their Application to Computer Science, 3rd Intl. Workshop, Warrenton, VA, H. Ehrig et al. (eds), Springer-Verlag, Berlin, pp. 440-457, 1986.

[15] Murata, T. and Zhang, D.,"A Predicate-Transition Net Model for Parallel Interpretation of Logic Programs,"*IEEE Trans. on Software Eng.*, Vol. 14, No. 4, pp. 481-497, Apr. 1988.

[16] Nadel, B. A.,"Constraint Satisfaction Algorithm,"*Comp. Intell.*,Vol. 5, No. 4, pp. 188-224, 1989.

[17] Stefik, M.,"Planning with Constraints (MOLGEN: Part 1 & 2),"AI, Vol. 16, pp. 111-170, 1981.

[18] Sussman, G. et al.,"Constraints-a Language for Expressing Almost-Hierarchical Descriptions,"*AI*, Vol. 14, pp. 1-39, 1980.

[19] Wilson, M. and Borning, A.,"Extending Hierarchical Constraint Logic Programming: Nonmono-tonicity & Inter-Hierarchy Comparison,"*Logic Prog.:P. of North American. Conf.*,pp. 3-19, 1989.

[20] Young, R. et al.,"An Artificial Intelligence-Based Constraint Network System for Concurrent Engineering,"*IJPR*, Vol. 30, No. 7, pp. 1715-1735, 1992.

95년 10월 최초 접수, 96년 6월 최종 수정