Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1997), *11*, 409–417. Printed in the USA.

Generalized set propagation operations for concurrent engineering

WALID E. HABIB AND ALLEN C. WARD
Department of Mechanical Engineering, The University of Michigan, Ann Arbor, MI 48109, U.S.A.

(RECEIVED December 22, 1995; ACCEPTED July 2, 1997)

Abstract

This paper defines a number of general operations that accept arbitrary sets of values for two variables and general relations among three variables and generate a variety of third sets that are useful in design. Although the operations are defined without respect to mathematical or engineering domain, computing these operations depends on the specific mathematical domain, and algorithms are available for only a few domains. Appropriate software could make this complexity transparent to the designer, allowing the same conceptual operations to be used in many contexts. The paper proves a number of useful characteristics of the operations and offers examples of their potential use in design.

Keywords: Constraint Propagation; Set Propagation Operations

1. INTRODUCTION

This paper shows that a number of general operations can be defined that accept arbitrary sets of values for two variables and general relations among three variables and that generate a variety of third sets that are useful in design reasoning.

For example, suppose that we are designing a two-dimensional robotic manipulator (Figure 1).

$$\theta = \begin{bmatrix} \theta_a \\ \theta_b \end{bmatrix}, \quad l = \begin{bmatrix} l_a \\ l_b \end{bmatrix}, \quad z = \begin{bmatrix} z_x \\ z_y \end{bmatrix}$$

might be, respectively, a vector of joint angles, a vector describing the relative locations of each pair of joints (fixed by the arm segment design), and the endpoint location vector. *G might be the kinematic relationship among these variables, described below.*

$$G \equiv \begin{bmatrix} z_x \\ z_y \end{bmatrix} - \left[\begin{bmatrix} \cos \\ \sin \end{bmatrix} \times [\theta_a \quad \theta_b] \right] \times \begin{bmatrix} l_a \\ l_b \end{bmatrix} = [0],$$

where $\begin{bmatrix} \cos \\ \sin \end{bmatrix}$ operates on the vector

$$\begin{bmatrix} \theta_a \\ \theta_b \end{bmatrix} \text{ in } \left[\begin{bmatrix} \cos \\ \sin \end{bmatrix} \times [\theta_a \quad \theta_b] \right]$$

to produce

$$\begin{bmatrix} \cos(\theta_a) & \cos(\theta_b) \\ \sin(\theta_a) & \sin(\theta_b) \end{bmatrix}.$$

Given any value for θ and l (i.e., given a particular manipulator in a particular position), G allows us to find the corresponding position z. Given values for z and l, we can find a (often more than one) corresponding vector of angles θ. A designer could use this capability by trial and error, picking particular values and testing them.

A more powerful approach would simultaneously reason about the space of required locations and the possible and required joint angles and segment shapes, together with similar relationships and value sets based on dynamics, strength of materials, and the design of actuators to identify all of the acceptable designs. Imagine an operation, call it **Image**, that accepts the sets of possible arm segment lengths and joint angles and returns the set of points in space outside which no manipulator configuration can reach. Imagine another operation, call it **SufElem**, that accepts the set of points in space that the manipulator must reach and the set of possible joint angles that may be used and that returns the set of arm segment length pairs, each of which would guarantee covering the required space.

Reprint requests to: Dr. Allen C. Ward, Ward Synthesis, 3446 Gettysburg Road, Ann Arbor, MI 48105, U.S.A. E-mail: award@dip.eecs.umich.edu.

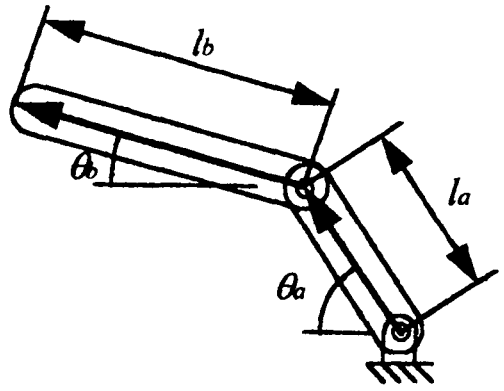

Fig. 1. Two-dimensional robotic manipulator.

Using such operations, the de-signer could reason about sets of possibilities simultaneously.

This paper defines five such generalized set propagation operations: **Image**, **Domain**, **SufElem**, **SufSets**, and **IndepSet**. The **Image**, **Domain**, and **SufElem** operations are generalized versions of the **Image**, **Domain**, and **SufPt** operations of the labeled interval calculus (Ward et al., 1989), and **SufSets** and **IndepSet** are new. The operations are not domain specific by definition, but computing them depends on the specific mathematical domain. The operations may in some cases be as difficult and as computationally intensive as optimization methods, but appropriate software could make this complexity transparent to the designer.

The generalized set propagation operations are defined to be used to answer engineering or design questions. Even if computing the operations is domain specific (some may not be computable at all in some domains), the design questions are the same, irrespective of the domain of the problem. Whether you want to know the set of manipulator arm segment lengths that would guarantee covering a given space of operation and the set of possible joint angles for the manipulator or the set of wing spar sections that would result in a safe airplane given a set of possible loads and performance characteristics, you are basically asking the same questions and can therefore use the same operation to get the results.

This paper is not primarily concerned with computing the set propagation operations. Furthermore, one cannot do so for all possible mathematical domains, but this does not undermine the usefulness of the operations. The set propagation operations are meaningful and useful, just as the addition operation is useful even if it is computed differently for different domains (e.g., adding vectors, complex numbers, and integers) and sometimes cannot be computed at all or can be only approximated (adding e to π).

Several kinds of computations are already possible. Ward et al. (1989) showed how to compute with some of the operations for the case of monotonic relations among variables with intervals of real numbers. The research underlying this paper does the same with the rest of the operations. Many engineering problems fall within this class. Bains and Ward (1993) computed some operations for the case of nonmonotonic algebraic relations, and Chen and Ward (1995a,b,c) computed some operations for the case of equations of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ and $\mathbf{b}$ are interval vectors of values and $\mathbf{A}$ is an interval matrix. In general, it should be possible to compute approximations to all the operations by simulation. Appropriate software should make the complexity of computation transparent to the designer, allowing the same conceptual operations to be used in many domains.

Section 2 presents some background to the paper. Sections 3–8 define the generalized operations for the propagation of generalized sets, state and prove various properties of the operations, and illustrate with examples their potential utility in a variety of design problems. Section 9 is the conclusion.

2. BACKGROUND

Constraint propagation has been advocated as a vehicle for concurrent engineering (Sutherland, 1963; Sussman & Steel, 1980; Ward, 1990; O'Grady et al., 1992). In the design process, constraints are imposed from the different sources contributing to the design. These constraints are continuously modified until all of the agents agree that a satisfactory solution to the problem has been reached. Constraint propagation ensures that the information provided by an agent is communicated to those that are affected by it.

There is, however, a progression of notions about constraint propagation. One common method of constraint propagation defines equality constraints (equations relating the system's variables) and propagates single values of variables (equality constraints such as $x = 3$) through these constraints. The resulting constraints are then compared with the allowable values for the output variables. Although this method communicates values among the different parts of the design (or different agents), it only provides information about a single point in the design space at a time. If a conflict occurs, another point is chosen and the process is repeated. Infeasible solutions are eliminated one by one, and all solutions can be investigated only if the entire design space is searched. Many design optimization methods can be viewed as belonging to this category; they follow a cycle of choosing individual values of the free variables, propagating them, and verifying the constraints until the optimal solution is achieved.

Other methods of constraint propagation involve the propagation of interval of values (or constraints of the form $x \in [3 \quad 4]$) for variables through the constraint equations. These "set-based" methods present an advantage over point-based methods because the entire design space is dealt with, thus reducing the need for search. Davis (1987) and Rinderle and Krishnan (1990) propagated interval of values using interval arithmetic as defined by Moore (1979). Interval propagation methods use intervals to describe the possible values

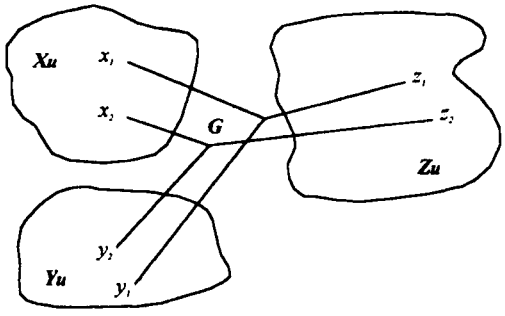

Fig. 2. Relation G among x, y, and z.

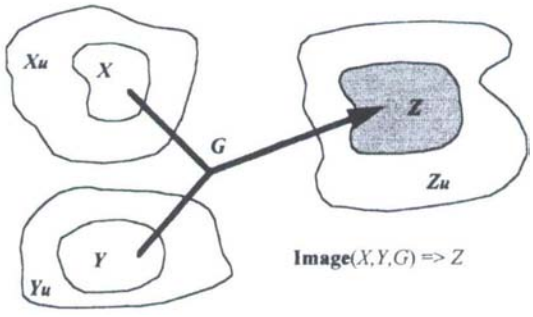

Fig. 3. The image operation.

of variables and the techniques of interval mathematics to propagate them. Unfortunately, an interval represents only one kind of constraint on the values of a variable: an upper and a lower bound. It offers no information about the distribution of values within these bounds.

At a third level, in the labeled interval calculus, Ward et al. (1989) added labels to intervals to describe the variation of the values that variables can assume and define propagation operations to propagate the labeled intervals. The methods of the labeled interval calculus can address more kinds of design problems than interval propagation methods.

This paper generalizes the three labeled interval calculus operations to arbitrary sets of mathematical quantities and adds a couple of new operations.

3. GENERALIZED SET PROPAGATION OPERATIONS

The following sections define and describe the generalized set propagation operations. The letters x, y, and z denote three variables (of any kind). X_u, Y_u, and Z_u are the "universal sets" of all of the assignments to x, y, and z in which we are interested. X, Y, and Z are three sets of values for x, y, and z, respectively. $G(x,y,z)$ represents an expression establishing a relationship among the three variables, that is, a set of triples of assignments (Figure 2). The operations are invoked in forms such as **Operation**$(G,X,Y) = Z$ or **Operation**$(G,Z,X) = Y$. In both cases, X in the operation corresponds to x in the relation, even though the position of X in the operation form (and usually its role in the operation) is different.

4. THE IMAGE OPERATIONS

Image,[1] the simplest and most traditional of the operations, accepts a relationship among three variables and sets of assignments for two of the variables and returns the set of compatible assignments in the third variable. **Image** is important because it defines an upper bound for the set of possibilities for the third variable.

DEFINITION. **Image**$(G,X,Y) \equiv \{z{:}\exists x \in X, \exists y \in Y \cdot (x,y,z) \in G\}$. ■

Image(G,X,Y) returns Z, the set of assignments for z such that there exists a value for x in X and a value for y in Y, satisfying G (Figure 3).

For example, if Θ were the set of all possible joint angle vectors, L the set of possible arm segment lengths from which the designer may choose, and G the manipulator kinematic equations, **Image**(G,L,Θ) would return A, the set of possible endpoint locations. Given this information, the manufacturing engineer responsible for the workspace layout would know that, no matter the final configuration of the robot, no point outside A can be reached by the arm. Figure 4 shows how the variations in θ and l affect each arm of the robot separately, and Figure 5 shows the resultant set of endpoint locations A. (Computation in this case was done graphically.)

Conversely, given a set of possible endpoint locations and segment lengths, **Image**(G,A,L) would return Θ_2, the set of possible corresponding joint angle vectors. Figure 6 il-

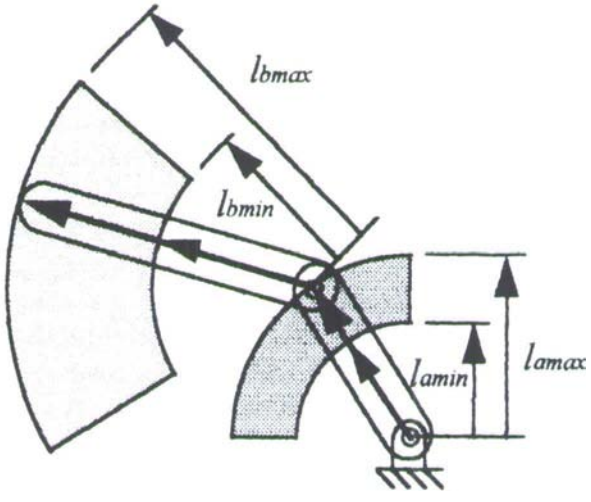

Fig. 4. Effect of the variation in θ and l on the robot arm.

[1] The **Image** operation was called "**Range**" in the original LIC papers. The name change brings the operation more in line with standard mathematical practice.

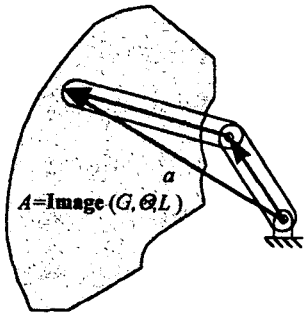

Fig. 5. Resulting endpoint locations A.

lustrates the fact that, given interval Z = **Image**(G,X,Y), **Image**(G,Z,Y) will not return the original interval X.

Why is $X_2 \neq X$? Because in general there is more than one pair x, y corresponding to each z. There also may be more than one x corresponding to each pair y, z.

4.1. Properties of Image

The **Image** operation has other simple properties that can sometimes be useful.

COROLLARY 1.

IF: Z = **Image**(G,X,Y),
THEN: $\forall x \in X$, **Image**$(G,\{x\},Y) \subseteq Z$ (and $\forall y \in$ Y, **Image**$(G,X,\{y\}) \subseteq Z$). ■

The proof is straightforward and the implication is trivial but useful: A robot with arm lengths chosen from the set L and Θ as the set of possible joint angles will never reach a point outside the space A.

Note also that the **Image** operation is commutative: **Image**(G,Y,X) = **Image**(G,X,Y).

COROLLARY 2.

IF: Z = **Image**(G,X,Y),
THEN: $\forall z \in Z$, **Image**$(G,\{z\},Y) \cap X \neq \phi$
(and $\forall z \in Z$, **Image**$(G,\{z\},X) \cap Y \neq \phi$). ■

Proof: *Pick a* z^* in Z. By Z = **Image**(G,X,Y), there is an x^* in X and y^* in Y such that $(x^*,y^*,z^*) \in G$; x^* is also in **Image**$(G,\{z^*\},Y)$. ■

In the robot example, it follows from this corollary that, for any point in the workspace A, there exists an arm configuration (set of arms) that would be able to reach that point.

4.2. Other examples of useful Image operations

Suppose that many alternatives are still being considered for the hood of a new car (a set of features geometrically related to each other). Given the geometrical relation between the features, **Image** could be used to find the space that contains all of the alternatives. Passing this information to the die designers would allow them to start working on the die design, choosing a die blank and a "hit number" for all of the hood alternatives. See Lee (1996) for examples of this use.

Given a set of vectors of possible forces applied on a finite element structure and a set of stiffness matrices describing the possible materials, **Image** would return the set of possible displacements of the nodes. See Chen and Ward (1995*a*).

5. THE DOMAIN OPERATIONS

The **Domain** operation is defined as follows.

DEFINITION. **Domain**$(G,X,Y) \equiv \{z: \forall y \in Y \ \& \ \forall x \cdot (x,y,z) \in G, x \in X\}$. ■

The definition means that **Domain**(G,X,Y) is the set of all z such that, for any y in Y and any x such that the triplet (x,y,z) is in G, x is in X.

The **Domain** operation is important because it can be thought of as a "partial" inverse to **Image** (Corollary 3 defines **Domain** in terms of the **Image** operation). If Z = **Domain**(G,X,Y), then **Image**$(G,Z,Y) \subseteq X$ (Figure 7). Therefore, given sets X and Y for variables x and y, respectively, **Domain**(G,X,Y) returns the set Z for variable z such that any value of z in Z combined with any value of y in Y yields a value of x in X.

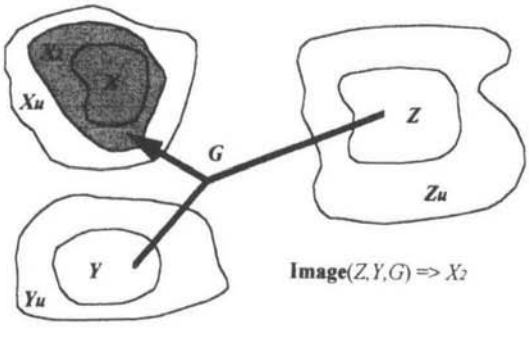

Fig. 6. Reversing image does not invert the image.

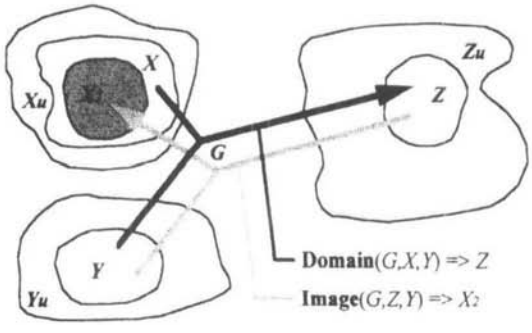

Fig. 7. The domain operations.

For example, suppose that, due to obstructions, the workspace in which our robot can operate safely is restricted to a certain volume A. If we know that the angular displacement of the joints considered will take every value in the set Θ, **Domain**(G,A,Θ) will return a set of arm segment lengths L, each of which never results in an endpoint location outside A, no matter which value in Θ is used. None of these robot designs will be able to crash into an obstacle (Figure 8). (Computation in this case exploited the fact that we had previously graphically computed the **Image** of the required set of arm lengths. In the general case, computation would require numerical approximation.)

Domain is defined in terms of the **Image** operation.

COROLLARY 3. **Domain**$(G,X,Y) \equiv \{z : \textbf{Image}(G, \{z\},Y) \subseteq X\}$. ■

Proof: Pick any z^* in **Domain**(G,X,Y). From the definition of **Image**, for any x^* in **Image**$(G,\{z^*\},Y)$, there is a y in Y such that (x^*,y,z^*) is in G. From the definition of **Domain**(G,X,Y), for any such z^*, y in Y and (x,y,z^*) in G, x is in X and **Image**$(G,\{z^*\},Y)$ is a subset of X.

Now pick a z' such that **Image**$(G,\{z'\},Y)$ is a subset of X. From the definition of **Image**, for any x in **Image** $(G,\{z'\},Y)$, there is a y in Y such that (x,y,z') is in G; x is in X, so from the definition of **Domain**, z' is in **Domain**(G,X,Y). ■

5.1. Other examples of a Domain operation: A particular case of vector quantities

Let $\vec{b}$ and $\vec{c}$ be two vectors representing points in the areas bounded by the polygons B and C, respectively (Figure 9), let $\vec{C}$ and $\vec{B}$ be the set of such vectors, and let $\vec{a} + \vec{b} = \vec{c}$ be a vector equation. **Domain**$(G,\vec{C},\vec{B})$, where G represents the vector relation, can be computed as follows.

For each combination of vertices from B and C, vector $\vec{a}$ can be computed as $(\vec{a} = \vec{c} - \vec{b})$, yielding a point in the plane. Each of these vectors is then combined with each of the vertices of B, yielding a point. If any of the resulting points is outside C, the vector $(\vec{a})$ is rejected. The tips of the remaining vectors are then connected to form polygon A, which represents the set of vectors $\vec{A}$, the result of the **Domain** operation.

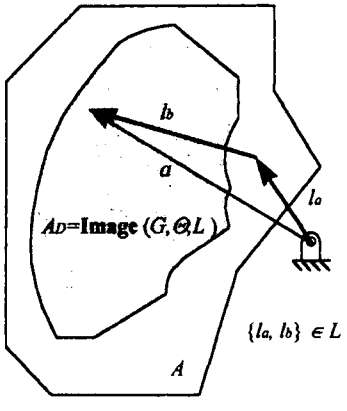

Fig. 8. L = **Domain**$(G,A,\Theta) \Rightarrow$ **Image**$(G,\Theta,L) \subseteq A$.

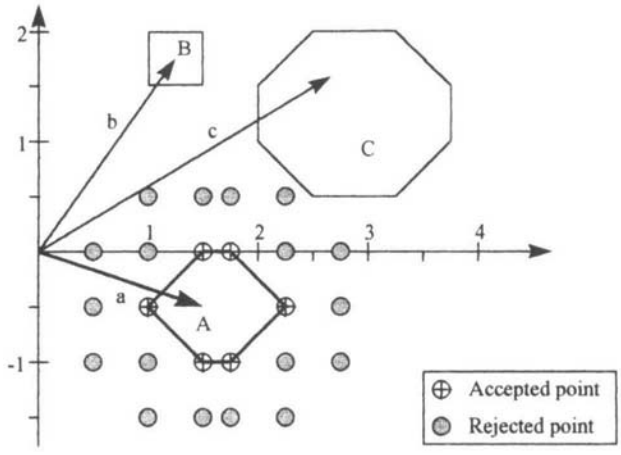

Fig. 9. Computing **Domain** for vector quantities.

This method is not a general technique for computing the **Domain** operation for the sum of two vectors: For some shapes of B and C, the **Image** of B with each of the vertex vectors $\vec{a}$ will fall partially outside C, even though a solution A exists. This method works well when the polygons are rectangles and could be used in the construction management domain (Tommelein & Zouein, 1993). For more complex cases, more sophisticated algorithms, similar to the "offset" algorithms of computer graphics, seem likely to be feasible.

Given a set of vectors of allowable displacements of the nodes in a finite element structure and a set of stiffness matrices describing the possible materials, **Domain** would return the set of forces that, when applied to the structure, would result in displacements within the allowable range. See Chen and Ward (1995*b*) for examples of this use.

Given the set of possible values for the impedance in an electrical circuit (which might vary because of manufacturing variation, temperature, adjustment, design uncertainty, or all of the above), **Domain** would return the set of values for the voltage that would limit the current to a certain interval.

Given the toleranced reserved space for an assembly of two parts (two geometrical shapes) and the toleranced dimensions for one of them, **Domain** would return selected toleranced dimensions for the second part.

Given the set of possible values for the stiffness of the springs used in a motorcycle suspension and the limits on the suspension displacements, **Domain** would return the set of allowable forces that can be applied to the suspension.

Darr (1997) and Darr and Birmingham (1996) used the **Domain** operation for additive equations to accelerate the solution of the constraint satisfaction problems.

6. THE SUFELEM OPERATIONS

The **SufElem** operation is defined as follows.

DEFINITION. $\mathbf{SufElem}(G,X,Y) \equiv \{z: \forall x \in X, \exists y \in Y \cdot (x,y,z) \in G\}$.

SufElem also can be thought of as another "partial" inverse to **Image** (Corollary 4 defines **SufElem** in terms of the **Image** operation). The **Image** of any single element of Z and set Y will include set X. In other words, any element of Z combined with set Y "covers" the entire set X.

SufElem is a contraction of sufficient elements; each element of Z, together with the Y values, is sufficient to generate all of X. Thus, in Figure 10, if $Z = \mathbf{SufElem}(G,X,Y)$ and z_1 and z_2 are elements of Z, then $X \subseteq \mathbf{Image}(G,\{z_1\},Y) = X_1$ and $Z \subseteq \mathbf{Image}(G,\{z_2\},Y) = X_2$.

As an example, imagine that, in the manipulator design problem, every point in an area represented by set A must be reached by the manipulator. Knowing this and the set Θ of possible joints angles, the designer could use **SufElem** (G,A,Θ) to find the set of possible relationships L between the joints, to be established by the arm segment lengths (Figure 11). $As = \mathbf{Image}(G,L,\Theta)$ represents the set of points that can be reached with any of these robot arms; it covers area A. (Again, we have exploited our previous knowledge of the generation of A to perform the computation. The general case would require numerical techniques.)

This is a different problem than the previous one. In this case, the manipulator is required to cover the entire set of points in A, whereas in the previous case, the manipulator was to be restricted to A but not required to reach every point of A. The current case requires the operation **SufElem**, which returns the set L (of arm segment lengths), any element of which will combine with Θ to result in an $\mathbf{Image}(G,\Theta,\{l\})$ that includes (or covers) the entire workspace A.

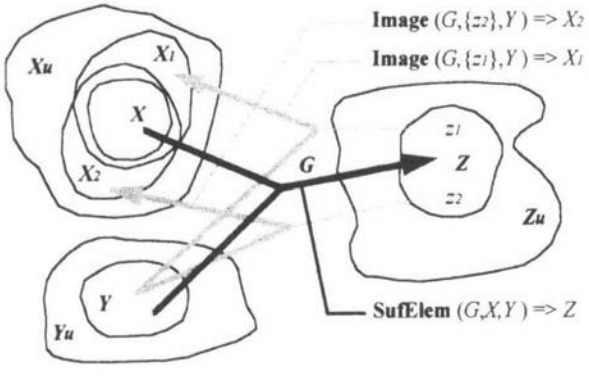

Fig. 10. The **SufElem** operation.

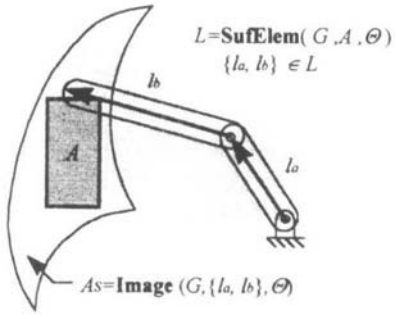

Fig. 11. $L = \mathbf{SufElem}(G,Z,\Theta) \Rightarrow Z \subseteq \mathbf{Image}(G,\{la,lb\},\Theta)$.

6.1. Properties of SufElem

COROLLARY 4 (**SufElem** defined in terms of the **Image** operation). $\mathbf{SufElem}(G,X,Y) \equiv \{z: X \subseteq \mathbf{Image}(G,\{z\},Y)\}$. ■

Proof: Pick any z^* in $\mathbf{SufElem}(G,X,Y)$. From the definition of **SufElem**, for any such z^* and for any x^* in X, there is a y in Y such that (x^*,y,z^*) is in G. Hence, x^* is in $\mathbf{Image}(G,\{z^*\},Y)$.

Now pick a z' such that X is a subset of $\mathbf{Image}(G,\{z'\},Y)$. From the definition of **Image**, for any x' in X, there is a y in Y such that (x',y,z') is in G. For any such z', from the definition of **SufElem**, z' is in $\mathbf{SufElem}(G,X,Y)$. ■

LEMMA 1. $\mathbf{SufElem}(G,X,Y) = \bigcap_{x \in X} \mathbf{Image}(G,\{x\},Y)$.

Proof: From the definition of **Image**, the intersection of $\mathbf{Image}(G,\{x\},Y)$ over set X is the set Z such that, for any x in X, there is a y in Y such that (x^*,y,z) is in G. This is the definition of $\mathbf{SufElem}(G,X,Y)$.

COROLLARY 5.

IF: $Z = \mathbf{SufElem}(G,X,Y)$,
THEN: $\forall x \in X, Z \subseteq \mathbf{Image}(G,\{x\},Y)$
AND: $\forall Z^* \subset Z, \exists x^* \in X \cdot Z^* \not\subseteq \mathbf{Image}(G,\{x^*\},Y)$.

The proof is straightforward from Lemma 1.

The following example illustrates the meaning of Corollary 5.

Suppose that the robot designer conveys to the manufacturing engineer the set of possible joint relationship values L, of which one will eventually be chosen, and that the angular displacement of all of the joints considered will take every value in the set Θ. From Lemma 1 and Corollary 5, set A returned by $\mathbf{SufElem}(G,L,\Theta)$ represents the area that will be reached no matter which manipulator from the possible set is used. Set A is the intersection of the workspaces of all of the possible robot configurations. Furthermore, any point outside the set A will not be reachable by at least one manipulator configuration. If the manufacturing engineer de-

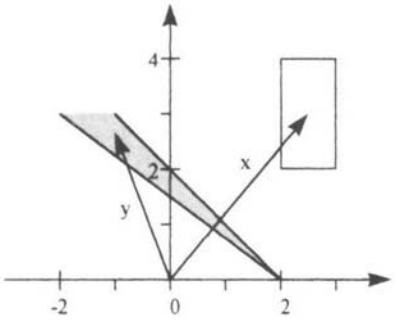

Fig. 12. Computing **SufElem** for the vector dot product.

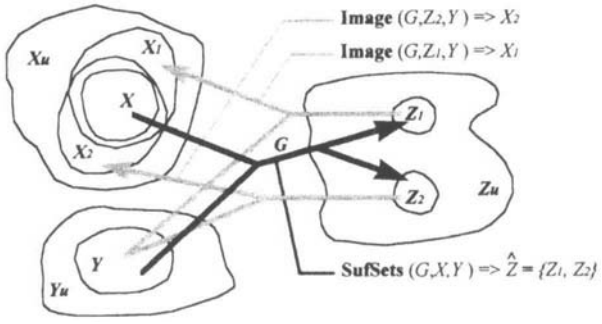

Fig. 13. The **SufSets** operation.

signs the process to be restricted within the areas A, it will be valid irrespective of the robot ultimately used. The obvious way to compute A is to superimpose the workspaces of the manipulators, graphically or computationally.

6.2. Other examples of a SufElem operation: The vector dot product

Suppose that the dot product d of vectors $\vec{x} = (a,b)$ and $\hat{y} = (c,d)$ should be within $D = [4 \quad 6]$. Knowing that vector $\vec{x}$ is within $\vec{X} = ([2 \quad 3],[2 \quad 4])$, $\mathbf{SufElem}(d - \vec{x}\cdot\hat{y} = 0, D, \vec{X})$ is computed by solving the inequalities:

$$2c + 2d \leq 4$$

$$3c + 4d \geq 6$$

and is shown in Figure 12. $\mathbf{SufElem}(d - \vec{x}\cdot\hat{y} = 0, D, \vec{X})$ is the set of vectors $\hat{y}$, with tips within the shaded area.

Suppose that a structure is designed to define a set of displacements under input forces. Given a set of vectors of possible forces applied to the structure, the **SufElem** operation returns the stiffness matrices that would yield the desired displacements. See Chen and Ward (1995*c*) for examples of this use.

Suppose that a circuit is being designed to provide an output current that covers every value within a range so that the input voltage will vary within a set of values. The **SufElem** operation can be used to return the set of impedances of the circuit, any element of which produces the required current range.

If a load requires a variable force within a certain range, given the range of available inlet pressure, then **SufElem** could return the set of hydraulic cylinder areas, each of which can produce the required range of forces.

7. THE SUFSETS[2] OPERATIONS

DEFINITION. $\mathbf{SufSets}(G,X,Y) = \hat{Z} = \{Z1,\ldots,Zi,\ldots,Zn\}$ such that

$$\forall Zi \in \hat{Z}, \forall x \in X, \exists z \in Zi, \exists y \in Y\cdot(x,y,z) \in G$$

and

$$\forall Z^* \subset Zi, \exists x \in X\cdot\forall z \in Z^*, \forall y \in Y, (x,y,z) \notin G.$$

SufSets is yet another partial inverse to the **Image** operation and can be defined in terms of it (Corollary 6). It returns the set of sets $\hat{Z}$, where each member Zi is a *minimal* set satisfying $X \subseteq \mathbf{Image}(G,Zi,Y)$; that is, every set Zi in $\hat{Z}$, together with Y, generates all of X. In other words, to "cover" set X, given set Y, z has to take every value in Z.

Figure 13 shows that $\mathbf{SufSets}(G,X,Y) = \hat{Z} = \{Z1,Z2\}$, which implies that $X \subseteq \mathbf{Image}(G,Z1,Y), X \subseteq \mathbf{Image}(G,Z2,Y)$.

As an example, suppose that the stamping shown in Figure 14 is to be painted with a spray gun. Knowing the size of the piece and the diameter of the spray, one might find all of the locally minimal paths of the gun that would ensure a complete covering of the piece with paint. If X represents the set of points in the plane occupied by the stamping, Y the set of points representing the spray area, as measured relative to the spray nozzle, and G a relationship among spray area, gun motion, and covered area, then any element Zi in $\hat{Z} = \mathbf{SufSets}(G,X,Y)$ represents the set of paths that the gun may follow to cover the entire piece with paint. There are

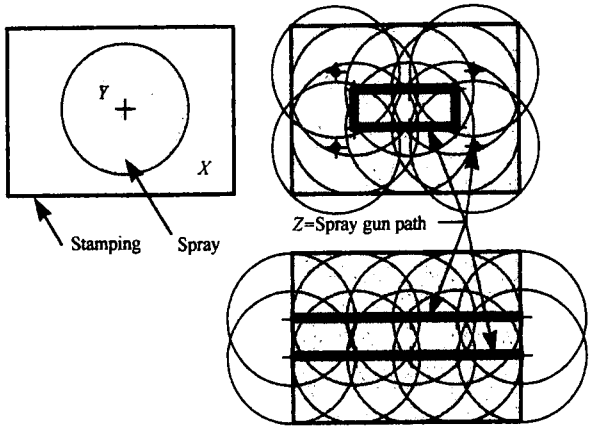

Fig. 14. Applying the **SufSets** operation.

[2] **SufSets** is short for sufficient sets.

many spray gun paths that can yield a complete covering of the piece with paint: Figure 14 shows two such sets, which were computed graphically. An obvious use of the set $\hat{Z}$ would be to search for the set's shortest member. Note that *each element* of **SufElem** is a member of **SufSets**, as shown in Corollary 7.

COROLLARY 6. $\mathbf{SufSets}(G,X,Y) = \hat{Z} = \{Zl,\ldots,Zi,\ldots,Zn\}$, where $\{Zi\colon X \subseteq \mathbf{Image}(G,Zi,Y)\ \&\ \forall Z^* \subset Zi,\ X \subseteq \mathbf{Image}(G,Z^*,Y)\}$.

The proof is straightforward from the definition of **Image**.

COROLLARY 7.

IF: $Z = \mathbf{SufElem}(G,X,Y)$,
THEN: $\forall z \in Z, \{z\} \in \mathbf{SufSets}(G,X,Y)$.

Proof: By definition of $\mathbf{SufElem}(G,X,Y)$, any set $\{z\} \in \mathbf{SufSets}(G,X,Y)$ is such that $X \subseteq \mathbf{Image}(G,\{z\},Y)$, which is minimal because $\{z\}$ is a single element set.

The following paragraphs show examples of **SufSets** operations.

SufSets would return one or more ranges of values for the variable impedance in a circuit that produces the required range of the output current for a given variation in the input voltage.

Suppose that a flexible structure must assume a defined set of displacements under input forces. Given a set of possible stiffness matrices for the structure, the **SufElem** operation returns the set of force vectors that should be applied to any such structure to yield the desired displacements. (Chen & Ward, 1995a, b, c, did not know of this generalized operation, but it seems likely that their methods could be extended to compute this result for appropriate sets.)

Given the sets of torques and speeds of the loads and the torque/speed/throttle curve for an engine, **SufSets** returns the set (or sets) of throttle openings that can drive the loads.

Given the required set of flow rates needed, the possible interval for the fluid levels in a tank, and the valve characteristic curves, **SufSets** returns the set of valve openings required to deliver the flow rates demanded.

In general, **SufSets** can be more difficult to compute than the previous operations because of the large number of sets that must sometimes be represented.

8. THE INDEPSET OPERATIONS

Finally, we define the **IndepSet**[3] operation:

DEFINITION. $\mathbf{IndepSet}(G,X,Y) \equiv \{z\colon \forall x \in X, \forall y \in Y, (x,y,z) \in G\}$.

In the manipulator example, let Z be the set of points representing a given workspace, let L be the set of arm segment lengths, and let G' be the relation on workspace, arm segment lengths, and joints. $\mathbf{IndepSet}(G',X,L)$ will return the set of joint actuators J, any one of which will cover the given workspace for any arm segment lengths l in L. With this method, the designer can choose any joint from the set J without actually knowing the arm geometry that will be finally chosen for the actuator: Any joint in J and will work with any arm geometry in L.

Another example of the **IndepSet** operation is to suppose that the load l on a bolted connection is in L = [400 700] lb. and that the maximum stress allowed for the connection is not yet decided but is known not to be outside the interval S = [24 60] kpsi. Let G be a catalog of bolts, with a maximum load given for each bolt. Then, $\mathbf{IndepSet}(G,L,S)$ returns a set of bolt ordering, any of which can take the loads.

9. CONCLUSION

This paper defines a number of set propagation operations: the generalized **Image** operation and its partial inverses **Domain**, **SufElem**, **SufSets**, and **IndepSet**. It defines and proves various properties of the operations and illustrates their utility on a variety of design problems.

These operations should be useful in design and are not domain specific; rather, they are defined in terms of general sets and general relationships among variables. The computation of the operations is domain specific. Indeed, computation may not be possible, for example, because the relationship G cannot be inverted or because the sets cannot be represented compactly. However, the meaning of the operations is general, and over time we can expect that the number of computable cases will increase.

The operations define mathematical set propagations only; which operation to use is dictated by the specific case. These operations can be used directly by people. However, for these operations to be useful in design automation, a system must be designed to recognize which operation is needed when. Formulation of such a system is the subject of other work.

ACKNOWLEDGMENTS

We gratefully acknowledge the contribution of Bill Finch, who discovered the **IndepSet** propagation operation described in this paper, and the National Science Foundation and the Office of Naval Research for their support of the research under contracts DDM-9010393 and N00014-91-5-1918.

REFERENCES

Bains, N., & Ward, A. (1993). Labeled interval operations involving nonmonotonic equations. 1993 ASME Conf. on Design Theory and Methodology, Albuquerque, NM.

Chen, R., & Ward, A. (1995*a*). RANGE family of propagation operations for intervals on simultaneous linear equations. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 9(3)*, 183–196.

[3] **IndepSet** is short for independence set.

Chen, R., & Ward, A. (1995*b*). DOMAIN family of propagation operations for intervals on simultaneous linear equations. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 9(3)*, 197–210.

Chen, R., & Ward, A. (1995*c*). SUFFICIENT-POINTS family of propagation operations for intervals on simultaneous linear equations. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 9(3)*, 211–217.

Darr, T.P. (1997). *A constraint-satisfaction problem computational model for distributed part selection.* Ph.D. Dissertation. Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor.

Darr, T., & Birmingham, W. (1996). An attribute-space representation and algorithm for concurrent engineering. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10*, 21–36.

Davis, E. (1987). Constraint propagation with interval labels. *Artificial Intelligence 32*, 281–331.

Lee, J. (1996). *Set-based design systems for stampings and flexible fixture workspaces*, Ph.D. thesis. The University of Michigan, Ann Arbor, Michigan.

Moore, R.E. (1979). Methods and applications of interval analysis. *SIAM.*

O'Grady, P., Kim, J.Y., & Young, R.E. (1992). Concurrent engineering system for rotational parts. *International Journal of System Automation: Research and Applications 2*, 245–258.

Rinderle, J., & Krishnan, V. (1990). Constraint reasoning in concurrent design. 1990 ASME Conf. on Design Theory and Methodology, Chicago, IL.

Sussman, G. & Steel, G. (1980). Constraints—A language for expressing almost-hierarchical descriptions. *Artificial Intelligence 14*, 1–39.

Sutherland, I. (1963). *Sketchpad—A man-machine graphical interface.* Ph.D. thesis. Massachusetts Institute of Technology, Cambridge, MA.

Tommelein, I.D., & Zouein, P.P. (1993). Interactive dynamic layout planning. *Journal of Construction Engineering and Management 119(2)*, 266–287.

Ward, A. (1990). A formal system for quantitative inferences of sets of artifacts. *Proc. First International Workshop on Formal Methods in Engineering Design, Manufacturing, and Assembly.* Colorado Springs, CA.

Ward, A., Lozano-Perez, T., & Seering, W. (1989). Extending the constraint propagation of intervals. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing 4(1)*, 47–54.

Allen C. Ward has been an Assistant Professor of Mechanical Engineering at the University of Michigan since 1989. His Ph.D. and M.Sc. in mechanical engineering are from MIT, and he holds Bachelor's degrees in mechanical engineering and history from the universities of Hawaii and Oregon, respectively. His research interests include the automation and management of distributed concurrent design processes and the design of innovative computer-controlled machines.

Walid E. Habib holds a B.E. degree in Mechancial Engineering from the American University of Beirut and a M.S.M.E. from the University of Michigan. He is currently pursuing a Ph.D. in Mechanical Engineering at the University of Michigan in Ann Arbor, where his current research interests include design automation and management of the design process. Previously, Habib was on the Engineering Faculty at the American University of Beirut for 5 years and is currently working as a design engineer.