A Z based Theory of Design Change using STEP Designs and KIF Design Intents exemplified by a 3 Dimensional Transverse Bulkhead

Frank M. Brown and Guy E. Jacobs
Artificial Intelligence Laboratory
University of Kansas
Lawrence, Kansas, 66045
brown@eecs.ukans.edu jacobs@eecs.ukans.edu

ABSTRACT

The mental process of predicting the reasonable consequences of a change in the design of a three dimensional artifact is an important step in the design process. This paper addresses issues in developing a theory of design change based on the idea that the incremental changing of the geometric design of an artifact may be formally explained by logical and mathematical theories, similar to those used by the automated planning and simulation systems of Artificial Intelligence. A computational model of the deductive process of designing a three dimensional transverse bulkhead for a ship is discussed. This model represents the frame laws and nonmonotonic default constraints in the modal quantificational logic Z which is a monotonic logic. The artifact designs are expressed in the STEP language standard and the design intents are expressed in the KIF language standard. Translators from STEP and KIF to logic and the Z based deduction laws are expressed in the very high level Logistica Programming Language.

1. INTRODUCTION

One important step in creating the design of an artifact to be manufactured is the mental process of changing a design and predicting the reasonable consequences of that design change. This process may be viewed as applying either to the initial development of the design starting from a simple shape or to the modification of an already sophisticated design satisfying most of the requirements for the desired artifact. In either case, the representation of the geometric object is being changed by some drafting procedure representing that mental action. Some recent papers [23,16] introduce the idea that the incremental changing of the geometric design of an artifact illustrated by the Sketchpad system [26] and in [4] may be formally explained by logical and mathematical theories similar to those used by the automated planning and simulation systems of Artificial Intelligence to predict the consequences of executing physical robotic actions. Since there is now a substantial body of research on formal theories of action addressing and solving the famous Frame Problem [24,9] with effective formal (i.e. monotonic) axiomatic theories of nonmonotonic reasoning, [12], this suggestion of applying Artificial Intelligence based research on theories of action merits serious attention.

The purpose of this paper is to further develop this idea by moving from its application to abstract two dimensional designs as discussed in [16] to its application to three dimensional artifact designs produced by real CAD systems. Specifically the initial artifact design is produced by a CAD system which then outputs the design expressed in the STEP language. The Standard of Transfer and Exchange of Product Model Data (i.e. STEP) is ISO Technical Committee TC 184's proposed international standard for expressing artifact designs [3,25]. Likewise the designer's design intent is initially expressed in KIF. The Knowledge Interchange Format (i.e. KIF) is a proposed standard for expressing general laws expressible in first order logic [18]. In this case design intent expresses the properties about basic features of the artifact which either must remain the same or if possible should remain the same by default as the artifact's design is modified. The design expressed in STEP and the design intent expressed in KIF are bound together by an interpreter which accepts actions to produce design changes. This is done by translating the STEP and KIF code into a logical system of rules which can be executed in a very high level deductive programming language called Logistica. The Logistica code includes rules based on our modal quantificational logic Z [10,12] which is an axiomatic theory of nonmonotonic reasoning, on certain laws for solving the Frame Problem implemented in Z as suggested in [11], and on an implementation in the Logistica language and deduction system [2,13,14] of the Z based frame laws. The modal logic Z is used to formally define the effects of the desired design change. Translators from STEP and KIF to logic and from logic back to STEP and the Z based deduction laws are expressed in Logistica.

The principles involved in Logistica were described in [8] and other applications of Logistica related to design are described in [15]. Logistica is a multivalued symbolic programming system which extends Scheme with multiple values and pattern matching and which provides an alternative to the works of [17,19,21,22].

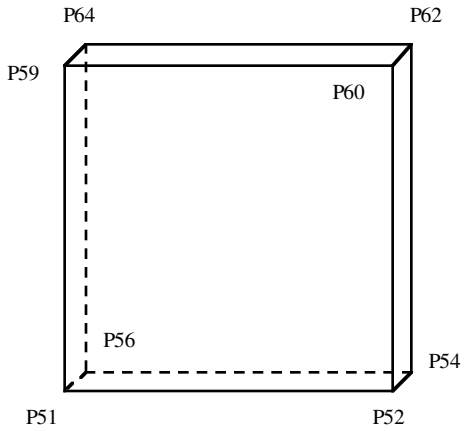
These ideas are illustrated by applying them to a three dimensional transverse bulkhead. Although the

examples discussed herein are very simple, this theory of design change is applicable to arbitrary designs expressed in the first order predicate logic in the sense that it specifies what are the appropriate solutions to any given design change. The ability of such a theory to specify the correct answer in even complicated cases expressed in the full range of first order logic should not however be confused with the tough mathematical and software engineering problems associated with actually producing those solutions. Because this theory is applicable to all of first order logic, it does however provide a framework for understanding and organizing more tractable methods applicable to limited cases of the first order logic.

Section 2 describes the initial design which we wish to change by merely moving one line in that design to some other location. The succeeding sections 3, 4, 5, 6, and 7 describe the result of making this change under various conditions. Section 3 describes the change in the absence of any frame laws and discusses the need for solving the frame problem with a general theory. Section 4 describes design change using a solution to the frame problem based on the Z modal quantificational logic. Section 5 illustrates the need for constraints to capture the design intent and how this need effects design change. Section 6 illustrates the problem of over-constrained designs and how default constraints can alleviate the over-constraining. Section 7 introduces the problem of under-constrained design changes which produce multiple solutions and how the notion of priority among default constraints can reduce the number of solutions in a rational manner. Finally, some conclusions are drawn in section 8.

2. THE INITIAL DESIGN

We start initially with a design representing in three dimensions a plate of steel given in figure 1 below.



The Initial Design
figure 1

This initial design constitutes a parallelepiped consisting of eight vertex points labeled P51, P52, P54, P56, P59, P60, P62, and P64 at the corner positions <0 0 0>, <240 0 0>, <240 0 -1/4>, <0 0 -1/4>, <0 240 0>, <240 240 0>, <240 240 -1/4>, and <0 240 -1/4> respectively. In addition there are twelve edge curves between the pairs of points [P52 P51], [P54 P52], [P56 P54], [P51 P56], [P60 P59], [P59 P64], [P64 P62], [P62 P60], [P59 P51], [P60 P52], [P62 P54], and [P64 P56]. This logical representation for the plate has been taken from the following STEP description:

```
#49=CARTESIAN_POINT(',(0.E0,0.E0,0.E0));
#50=CARTESIAN_POINT(',(240.E0,0.E0,0.E0));
#51=VERTEX_POINT(',#49);
#52=VERTEX_POINT(',#50);
#53=CARTESIAN_POINT(',(240.E0,0.E0,-2.5E-1));
#54=VERTEX_POINT(',#53);
#55=CARTESIAN_POINT(',(0.E0,0.E0,-2.5E-1));
#56=VERTEX_POINT(',#55);
#57=CARTESIAN_POINT(',(0.E0,240.E0,0.E0));
#58=CARTESIAN_POINT(',(240.E0,240.E0,0.E0));
#59=VERTEX_POINT(',#57);
#60=VERTEX_POINT(',#58);
#61=CARTESIAN_POINT
        (',(240.E0,240.E0,-2.5E-1));
#62=VERTEX_POINT(',#61);
#63=CARTESIAN_POINT(',(0.E0,240.E0,-2.5E-1));
#64=VERTEX_POINT(',#63);
#70=EDGE_CURVE(',#52,#51,#4,.T.);
#72=EDGE_CURVE(',#54,#52,#8,.T.);
#74=EDGE_CURVE(',#56,#54,#12,.T.);
#76=EDGE_CURVE(',#51,#56,#16,.T.);
#86=EDGE_CURVE(',#60,#59,#20,.T.);
#88=EDGE_CURVE(',#59,#64,#24,.T.);
#90=EDGE_CURVE(',#64,#62,#28,.T.);
#92=EDGE_CURVE(',#62,#60,#32,.T.);
#103=EDGE_CURVE(',#59,#51,#36,.T.);
#106=EDGE_CURVE(',#60,#52,#40,.T.);
#119=EDGE_CURVE(',#62,#54,#44,.T.);
#132=EDGE_CURVE(',#64,#56,#48,.T.);
```

The above is not the complete STEP file, just the portions of STEP that apply to this example. In the logical representation of the artifact the cartesian points (#49, #50, #53, #55, #57, #58, #61, #63) have been eliminated using equality substitution supported by the following law of classical logic:

$$\exists x\ (x=a \wedge \Phi x) \leftrightarrow \Phi a$$

The edge geometries (#4, #8, #12, #16, #20, #24, #28, #32, #36, #40, #44, #48) and the same-sense truth values (.T. in the edge curve definitions) have not been included in our logical representation of the plate as we are assuming that the edges are all straight line segments between their vertex points. We make the following definitions:

[P Q] =df (EDGE-CURVE P Q)

<X Y Z> =df (VERTEX-POINT
(CARTESIAN-POINT X Y Z))
where [P Q] is an edge curve from P to Q drawn as a straight line and <X Y Z> is a vertex point with rectangular cartesian coordinates X, Y, and Z.

The initial design is called the database K0 where K0 is formally asserted to be logically equivalent to the conjunction of the following stated facts:
K0 ≡ P51=<0 0 0> ∧ P52=<240 0 0>
∧ P54=<240 0 -1/4> ∧ P56=<0 0 -1/4>
∧ P59=<0 240 0> ∧ P60=<240 240 0>
∧ P62=<240 240 -1/4> ∧ P64=<0 240 -1/4>
∧ L70=[P52 P51] ∧ L72=[P54 P52]
∧ L74=[P56 P54] ∧ L76=[P51 P56]
∧ L86=[P60 P59] ∧ L88=[P59 P64]
∧ L90=[P64 P62] ∧ L92=[P62 P60]
∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]

3. DESIGN CHANGE AND THE FRAME PROBLEM

We allow designs to be changed by a single mental action of moving an edge curve from its current location to a new location. This action is called: (move Zn A B) where A and B are vertex points and Zn is a the name of the edge curve being moved. The result of executing the action of moving an edge curve to a new location is assumed to be precisely that the edge curve is at the new location:

(result(move Zn A B))) ≡ (Zp = A) ∧ (Zq = B)

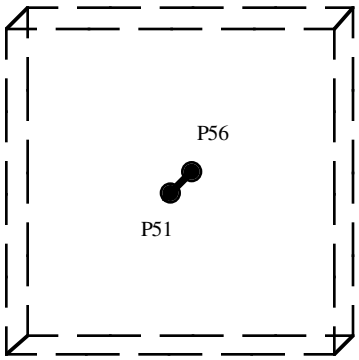
where Zn = [Zp Zq] is an edge in the original database K0. Finally, we let K represent the new database obtained by applying the move action.

K ≡ (result(move Zn A B))

We want to change the initial design given in figure 1 so that it looks more like half of a transverse bulkhead of a ship where the bottom face (P51, P52, P54, P56) is welded to the bottom plate, the right face (P52, P54, P62, P60) is welded to the centerline bulkhead, the top face (P59, P60, P62, P64) is welded to the deck plate, and the left face (P51, P56, P64, P59) is welded to the side plate. We do this by moving the edge curve L76 (i.e. [P56 P51]) from the initial end point locations <0 0 -1/4> <0 0 0> to a new pair of locations <120 120 -1/4> <120 120 0> where we expect the side and bottom plates to meet. Thus we write the obvious statement that the new database is the result of applying the move action: (move L76 <120 120 -1/4> <120 120 0>) giving:
K ≡ (result(move L76 <120 120 -1/4> <120 120 0>))

which simplifies to:
K ≡ P56 =<120 120 -1/4> ∧ P51 = <120 120 0>

This resulting situation K is illustrated in figure 2 below as an edge and two points superimposed on the original design K0 represented by the broken lines:



The Frame Problem
figure 2

The problem with this resulting database, as one can see, is that it says nothing at all about the other vertex points, nor about the edge curves. This is the Frame Problem [21]: namely that even though we defined the results of the action in an apparently obvious fashion (that when you move an edge curve to a new location, it is at the new location) this is not sufficient to describe the resulting situation, because we somehow automatically assume that many other facts will also hold such as for example that the point P60 remains at location <240 240 0> and that L92 is still an edge from P62 to P60. These implicit assumptions are not trivial; for if P51 were moved to location <240 0 0> we would no longer expect [P52 P51] to be an edge curve (according to STEP Informal Proposition IP2 for edge curves [25]) since P51 and P52 would coincide and the extent of the edge would be 0, and if P60 were rigidly attached to P51, it would no longer be at <240 240 0>. The frame problem can thus be restated as how to determine what additional facts hold in the resulting database when an action is applied.

First let us note a method that does not work, namely the idea of enumerating all the additional non changing facts to the description of the action:
K ≡ (result (move L76 <120 120 -1/4> <120 120 0>))
∧ (extra (move L76 <120 120 -1/4> <120 120 0>))
where:
(extra(move Zn A B)) =df
P52=<240 0 0> ∧ P54=<240 0 -1/4>
∧ P59=<0 240 0> ∧ P60=<240 240 0>
∧ P62=<240 240 -1/4> ∧ P64=<0 240 -1/4>
∧ L70=[P52 P51] ∧ L72=[P54 P52]
∧ L74=[P56 P54] ∧ L76=[P51 P56]
∧ L86=[P60 P59] ∧ L88=[P59 P64]
∧ L90=[P64 P62] ∧ L92=[P62 P60]
∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]

This method does not work in general because (a) we don't normally think of the facts that remain unchanged when describing these actions, (b) there are too many such facts in any non trivial problem, (c) such descriptions are almost always inconsistent since there are usually a few contradictory cases such as where P60 is rigidly attached to P51, or where P51 is moved to <240 0 0> contradicting that [P52 P51] is an edge curve, and (d) the execution of two actions in parallel will have contradictory results as each description of moving a point will say that the other point remains where it was.

4. THE Z MODAL LOGIC FRAME LAW

For these reasons we look for a more sophisticated approach which states that the new database K is obtained from the conjunction of the results of moving edge Zn to a new pair of positions A and B, and the frame law relating the old database to the new database:

K ≡ (result (move Zn A B)) ∧ (frame K0 K)

This expression is not an explicit definition of K but is merely an equation in which K may have 0 or more solutions. The frame law states that a proposition of the form: A=B holds if it necessarily follows from the laws and facts in the initial database K0 and if the conjunction of that proposition with the laws of the new database K is possible.

(frame K0 K) =df
  ∀(A B)
    ((K0 → A=B) ∧ ◊(K ∧ A=B)) → A=B[1]

The idiom that a proposition is possible is defined to mean that its negation is not necessary and the idiom that two propositions are necessarily equivalent is defined to mean that necessarily each implies the other.

(◊s) =df ~ ~s

(K ≡ s) =df (K ↔ s)

Finally, a proposition is necessary is defined to mean that it is logically necessary that it follows from the axioms of a real linear vector space:

s =df (Logically-Necessary(RLVS → s))

RLVS is the conjunction of the axioms of a real linear vector space [1] and Logically-Necessary is the necessity symbol of the Z Modal Quantificational Logic which axiomatizes the notion of theoremhood of first order logic:

(Logically-necessary(meaning S))
iff S is a theorem of First Order Logic

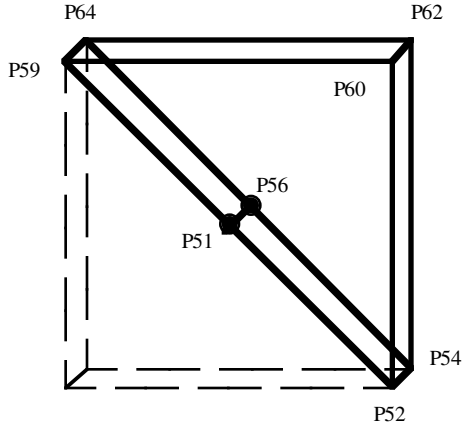
The modal Logic Z which is stronger than the well known S5 modal logic, its axiomatization, and the above metatheorem are described in [12]. The meaning function was described in [6] and a quantifier free version was used in [5].

Using Z and the above frame law we again move the edge curve L76 to the pair of points <120 120 -1/4> <120 120 0>:

K ≡ (result(move L76 <120 120 -1/4> <120 120 0>))
  ∧ (frame K0 K)

This time, all the extra facts that held in K0 remain facts in the new database K since the above expression simplifies by the laws of the Z modal logic to:

K ≡ P51=<120 120 0> ∧ P56=<120 120 -1/4>
  ∧ P52=<240 0 0> ∧ P54=<240 0 -1/4>
  ∧ P59=<0 240 0> ∧ P60=<240 240 0>
  ∧ P62=<240 240 -1/4> ∧ P64=<0 240 -1/4>
  ∧ L70=[P52 P51] ∧ L72=[P54 P52]
  ∧ L74=[P56 P54] ∧ L76=[P51 P56]
  ∧ L86=[P60 P59] ∧ L88=[P59 P64]
  ∧ L90=[P64 P62] ∧ L92=[P62 P60]
  ∧ L103=[P59 P51] ∧ L106=[P60 P52]
  ∧ L119=[P62 P54] ∧ L132=[P64 P56]



Design Change with a Frame Law
figure 3

This result may be deduced as follows: First, P51=<120 120 0> and P56=<120 120 -1/4> in K are the results of the action: (move L76 <120 120 -1/4> (120 120 0>). The remaining facts in K are obtained from the frame law as each such fact holds in K0 and is possible with K. However, P51=<0 0 0> is not propagated into K since K must contain the results of

---

[1] As explained in [12] it is intensions rather than extensions that are propagated by the frame law. The actual Frame Law restricts A and B to be specific intensions as follows:

∀(A B) ((Res A B)
  ∧ ((K0 → A=B) ∧ ◊(K ∧ A=B)) → A=B)

where:

(Res A B) =df (name-int A)
  ∧ ((edge-curve-int B) ∨ (vertex-point-int B))

where name-int, edge-curve-int and vertex-point-int are intensional predicates.

the action as previously mentioned and <120 120 0> does not equal <0 0 0> according to the laws of a real linear vector space and P56=<0 0 -1/4> is not in K for the same reason. The resulting solution for K may be checked by plugging it for K back into the original equation and simplifying the result to true. The resulting database representing the design of a triangular piece of metal is now represented by the above picture:

5. DESIGN INTENT CONSTRAINTS

Although the design produced in figure 3 is certainly better than the design produced in figure 2, one might say that in making the design change one really intended the edge curves in the resulting design to remain parallel with respect to the x, y, and z axes corresponding to the default that steel plates in a ship are most often horizontally or vertically oriented. This leads to a new axiom for K where K consists of the results of the moving action, the plate orientation constraints, and the frame laws as before:

K ≡ (result(move Zn A B)) ∧ constraints
  ∧ (frame K0 K)

In this case the constraints state the edge curves are to remain orthogonal with respect to the axes: that the x-coordinates of P51, P56, P64, and P59 are the same; the x-coordinates of P52, P54, P62, and P60 are the same; the y coordinates of P51, P52, P54, and P56 are the same; the y coordinates of P59, P60, P62, and P64 are the same; the z-coordinates of P51, P52, P60, and P59 are the same; and the z-coordinates of P56, P54, P62, and P64 are the same. These constraints are represented in the Knowledge Interchange Format (KIF) as follows:

```
 (parallel-plate/plane-constraint
     P51 P52 P60 P59 P56 P54 P62 P64)
(deffunction parallel-plate/plane-constraint
     (?p1 ?p2 ?p3 ?p4 ?p5 ?p6 ?p7 ?p8) :=
     (and  (samex ?p1 ?p5 ?p8 ?p4)
           (samex ?p2 ?p6 ?p7 ?p3)
           (samey ?p1 ?p2 ?p6 ?p5)
           (samey ?p4 ?p3 ?p7 ?p8)
           (samez ?p1 ?p2 ?p3 ?p4)
           (samez ?p5 ?p6 ?p7 ?p8)))
(deffunction samex (=(samex ?p1 ?p2 ?p3 ?p4)
  (and(= (xof ?p1) (xof ?p2)) (= (xof ?p2) (xof ?p3))
      (= (xof ?p3) (xof ?p4))(= (xof ?p4) (xof ?p1)))))
(deffunction samey (=(samey ?p1 ?p2 ?p3 ?p4)
  (and(= (yof ?p1) (yof ?p2)) (= (yof ?p2) (yof ?p3))
      (= (yof ?p3) (yof ?p4))(= (yof ?p4) (yof ?p1)))))
(deffunction samez (=(samez ?p1 ?p2 ?p3 ?p4)
  (and(= (zof ?p1) (zof ?p2)) (= (zof ?p2) (zof ?p3))
      (= (zof ?p3) (zof ?p4))(= (zof ?p4) (zof ?p1)))))
(deffunction xof
 (= (xof (vertex-point (cartesian-point ?x ?y ?z))) ?x))
(deffunction yof
 (= (yof (vertex-point (cartesian-point ?x ?y ?z))) ?y))
(deffunction zof
 (= (zof (vertex-point (cartesian-point ?x ?y ?z))) ?z))
(deffunction =
   (?name (vertex-point (cartesian-point ?x ?y ?z))) :=
   (and(= (xof ?name) ?x) (= (yof ?name) ?y)
       (= (zof ?name) ?z)))
```

The last definition relates the representation of a vertex point in cartesian notation to its corresponding cartesian selection functions used in the previous definitions in the KIF intent. We have chosen to define the vertex points in terms of the selection functions rather than the other way around because the automatic theorem proving characteristics provide simpler proofs and we can avoid the use of extra existential quantifiers[2].

Translating the KIF design intent into our internal logical representation and using the previously described frame law, moving edge curve L76 to [<120 120 0> <120 120 -1/4>] now gives:

K ≡ (result(move L76 <120 120 0> <120 120 -1/4>))
  ∧ (parallel-plate/plane-constraint
       P51 P52 P60 P59 P56 P54 P62 P64)
  ∧ (frame K0 K)

where:

```
(parallel-plate/plane-constraint
   (p1 p2 p3 p4 p5 p6 p7 p8) =df
       (samex p1 p5 p8 p4) ∧ (samex p2 p6 p7 p3)
     ∧ (samey p1 p2 p6 p5) ∧ (samey p4 p3 p7 p8)
     ∧ (samez p1 p2 p3 p4) ∧ (samez p5 p6 p7 p8)
(samex p1 p2 p3 p4)[3] =df
    (xof p1)=(xof p2) ∧ (xof p2)=(xof p3)
 ∧ (xof p3)=(xof p4) ∧ (xof p4)=(xof p1)
(samey p1 p2 p3 p4) =df
    (yof p1)=(yof p2) ∧ (yof p2)=(yof p3)
 ∧ (yof p3)=(yof p4) ∧ (yof p4)=(yof p1)
(samez p1 p2 p3 p4) =df
    (zof p1)=(zof p2) ∧ (zof p2)=(zof p3)
 ∧ (zof p3)=(zof p4) ∧ (zof p4)=(zof p1)
(xof <x y z>) =df x
(yof <x y z>) =df y
(zof <x y z>) =df z
(= name <x y z>) =df
   (xof name)=x ∧ (yof name)=y ∧ (zof name)=z
```

---

[2] When these definitions are made the frame law restriction is changed to be:

```
(Res A B) =df
       ((name-int A) ∧ (edge-curve-int B))
    ∨ (((xof-int A) ∨ (yof-int A) ∨ (zof-int A))
         ∧ (numeral-int B))
```

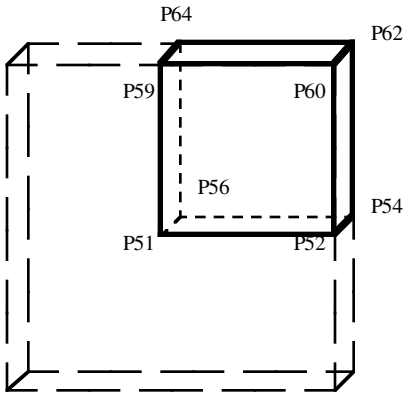
where xof-int, yof-int, zof-int, name-int, numeral-int, and edge-curve-int are intensional predicates.

[3] We used four (circular) constraints for the "same" definitions as empirically we observed faster running times than with the necessary three definitions, yet using all six did not yield any faster execution.

The above equation then simplifies to the following[4]:

K ≡ L70=[P52 P51] ∧ L72=[P54 P52]
∧ L74=[P56 P54] ∧ L76=[P51 P56]
∧ L86=[P60 P59] ∧ L88=[P59 P64]
∧ L90=[P64 P62] ∧ L92=[P62 P60]
∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]
∧ P52=<240 120 0> ∧ P54=<240 120 -1/4>
∧ P60=<240 240 0> ∧ P62=<240 240 -1/4>
∧ P59=<120 240 0> ∧ P64=<120 240 -1/4>
∧ P56=<120 120 -1/4> ∧ P51=<120 120 0>

which is represented by following picture:



Rectilinear Constraints
figure 4

6. OVER-CONSTRAINED DESIGN AND DEFAULT CONSTRAINTS

One problem with allowing design intent constraints, is that they may be inconsistent with the results of the design action. For example, if the additional constraint that the position of edge curve L72 remains at [<240 0 -1/4> <240 0 0>] (corresponding to the point where the centerline bulkhead is welded to bottom plate) were added to the previous plate orientation constraints, then the constraints would be inconsistent with moving L76 to [<120 120 0> <120 120 -1/4>]. Thus letting (in KIF):

```
(deffunction weld-to-keel-constraint () :=
    (and  (= L72 [P54 P52])
          (= L72 [<240 0 -1/4> <240 0 0>])))
```

which is represented in logic as:

(weld-to-keel-constraint) =df
L72=[P54 P52] ∧ L72=[<240 0 -1/4> <240 0 0>]

and moving L76 now gives:

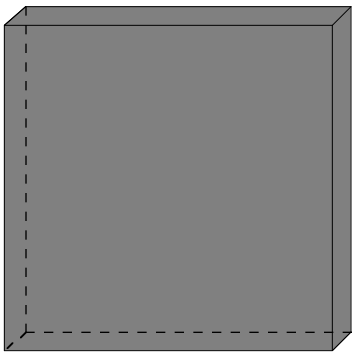
K ≡ (result(move L76 <120 120 0> <120 120 -1/4>))
∧ (parallel-plate/plane-constraint
P51 P52 P60 P59 P56 P54 P62 P64)
∧ (weld-to-keel-constraint)
∧ (frame K0 K)

which simplifies to just:

K ≡ #f

since the two constraints (= (yof P51) (yof P52)) and (= (yof P52) 0) together contradict the result of moving L76 to [<120 120 0> <120 120 -1/4>][5]. (Note that moving L76 to [<120 0 0> <120 0 -1/4>] would not have caused any problem as the L72 at [<240 0 -1/4> <240 0 0>] constraint and the plate orientation constraints could still be satisfied.)

The resulting over-constrained database is illustrated by the following picture since anything logically follows from falsity:



Over-Constrained Design Results in Anything
figure 5

The contradiction illustrated in figure 5, indicates that reasonable constraints cannot always be satisfied by the desired design change. This situation leads naturally to the idea of default constraints. Default constraints are constraints which are satisfied only if it is possible for them to be satisfied. In this case the constraints are themselves a function of the resulting database and the new database will be obtained from the expression:

K ≡ (result(move L76 <120 120 0> <120 120 -1/4>))
∧ (parallel-plate/plane-constraint
P51 P52 P60 P59 P56 P54 P62 P64 K)
∧ (weld-to-keel-constraint)
∧ (frame K0 K)

For example, if one of the offending constraints, namely (= (yof P51) (yof P52)) were replaced by a default constraint:

(◊(K ∧ (yof P51)=(yof P52)))

[4] After the theorem prover is finished, the KIF definitions are used in reverse whenever they can put the database into the original STEP form. e.g. (xof P51)=120 ∧ (yof P51)=120 ∧ (zof P51)=0 is output as P51=<120 120 0>.

[5] The constraints (= (yof P54) (yof P56)) and (= (yof P54) 0) would also have contradicted the result of the action.

→ (yof P51)=(yof P52))

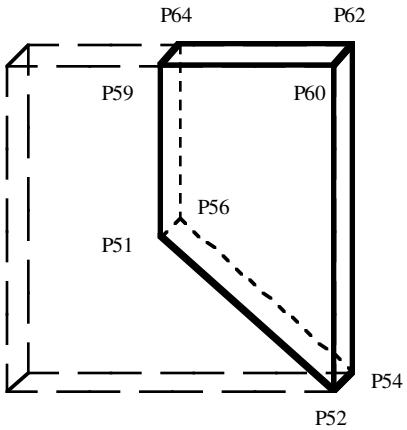
then the over-constrained nature of the resulting database would disappear. Redefining constraints and defining def[6] as the default symbol:

(parallel-plate/plane-constraint-version2
(p1 p2 p3 p4 p5 p6 p7 p8) =df
(samex p1 p5 p8 p4)
∧ (samex p2 p6 p7 p3)
∧ (def (samey p1 p2 p6 p5) K)
∧ (def (samey p4 p3 p7 p8) K)
∧ (samez p1 p2 p3 p4)
∧ (samez p5 p6 p7 p8)
(def ($p_1 \wedge \ldots \wedge p_n$) K) =df
((def p_1 K) ∧ … ∧ (def p_n K)
(def q K) =df ◊(K ∧ q) → q
where q is not a conjunction

The first definition for def states that def distributes across conjunctions so that in reality the default applies to propositions within those conjunctions. The second definition states that a proposition (which is not a conjunction) is defaulted in K if and only if it holds whenever it is logically possible with K. Now using version2 of the parallel-plate/plane-constraint when moving L76 and simplifying gives:

K ≡ L70=[P52 P51] ∧ L72=[P54 P52]
∧ L74=[P56 P54] ∧ L76=[P51 P56]
∧ L86=[P60 P59] ∧ L88=[P59 P64]
∧ L90=[P64 P62] ∧ L92=[P62 P60]
∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]
∧ P52=<240 0 0> ∧ P54=<240 0 -1/4>
∧ P60=<240 240 0> ∧ P62=<240 240 -1/4>
∧ P56=<120 120 -1/4> ∧ P51=<120 120 0>
∧ P64=<120 240 -1/4> ∧ P59=<120 240 0>

This database is represented by the following picture which begins to resemble a transverse bulkhead:



Fixing an Over-Constrained Design
figure 6

[6] def is an intensional defined symbol.

7. UNDER CONSTRAINED DESIGN CHANGE AND PRIORITIES

The previous section described an approach to solving the problem of over-constrained designs by replacing some constraint which contradicts the results of the design action by a default constraint. Since we cannot know in advance of making a specific action which constraint might be contradicted, it is only reasonable to allow many (if not all) constraints to be default constraints. For example, letting all the parallel plate constraints be default constraints and then moving L76 now gives:

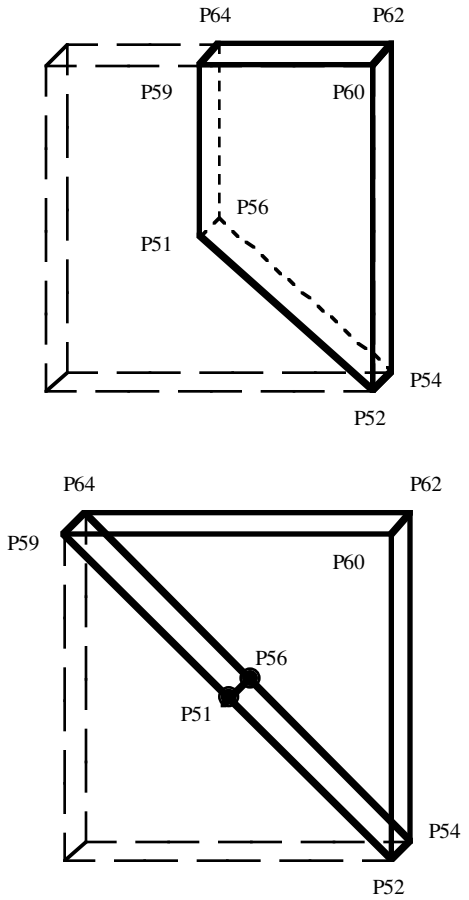
K ≡ (result(move L76 <120 120 0> <120 120 -1/4>))
∧ (def (parallel-plate/plane-constraint
P51 P52 P60 P59 P56 P54 P62 P64 K))
∧ (weld-to-keel-constraint)
∧ (frame K0 K)

which simplifies to the disjunction of two equivalencies:

K ≡ L70=[P52 P51] ∧ L72=[P54 P52]
∧ L74=[P56 P54] ∧ L76=[P51 P56]
∧ L86=[P60 P59] ∧ L88=[P59 P64]
∧ L90=[P64 P62] ∧ L92=[P62 P60]
∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]
∧ P52=<240 0 0> ∧ P54=<240 0 -1/4>
∧ P60=<240 240 0> ∧ P62=<240 240 -1/4>
∧ P56=<120 120 -1/4> ∧ P51=<120 120 0>
∧ **P64=<120 240 -1/4>** ∧ **P59=<120 240 0>**
∨
K ≡ L70=[P52 P51] ∧ L72=[P54 P52]
∧ L74=[P56 P54] ∧ L76=[P51 P56]
∧ L86=[P60 P59] ∧ L88=[P59 P64]
∧ L90=[P64 P62] ∧ L92=[P62 P60]
∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]
∧ P52=<240 0 0> ∧ P54=<240 0 -1/4>
∧ P60=<240 240 0> ∧ P62=<240 240 -1/4>
∧ P56=<120 120 -1/4> ∧ P51=<120 120 0>
∧ **P64=<0 240 -1/4>** ∧ **P59=<0 240 0>**

Two solutions are produced because of the interaction between the default constraints and the frame laws. In both solutions the edge curves are the same since we've not altered the function between the endpoints and we have added no new points; P51 and P56 are at <120 120 0> <120 120 -1/4> respectively because that is the result of the action; P52 and P54 are at <240 0 0> and <240 0 -1/4> because they are constrained to be there; and P60 and P62 are propagated by the frame law to be at locations <240 240 0> <240 240 -1/4>. Here is the splitting point of K: the x-coordinates of P51, P56, P59, and P64 are constrained to be equal; however, if the x-coordinate of P59 and P64 are constrained by default to be the x-coordinates of P51 and P56

namely 120 then P59 and P64 cannot by default be propagated by the frame law to be at their original locations and the first solution for K is obtained. Likewise if the locations of P59 and P64 are propagated by the frame law to be at <0 240 0> and <0 240 -1/4> then the x-coordinates of those two cannot by default be constrained to be equal to the x-coordinates of P51 and P56 and the second solution for K is obtained. These two solutions for the new database K are illustrated below:



Under-Constrained Design with 2 Solutions
figure 7

We see from this example that the multiple solutions derivable from constraints are representable in the Z modal logic formalism. We now address the question of how to eliminate some of these multiple solutions representing under-constrained designs.

The problem of under-constrained designs can in principle be solved whenever the frame laws and default constraints can be prioritized so that each law and each instance of each law is applied in a specific order[6]. In a specific case it suffices to prioritize just those defaults causing the multiple solutions. Thus any partial prioritizing of defaults will go a long way to eliminating multiple solutions. Prioritizing the defaults and frame laws has two conceptual problems: The first problem is to determine in a reasonable fashion the order of priority. A second more subtle problem is to decide whether the frame laws defaults should be prioritized or whether the production of multiple answers is desirable. As an initial answer to these vexing questions we propose that the default constraints always be given a higher priority than the frame laws. The reason for this is that the default constraints are something specifically added whereas the frame laws are more the unconscious fabric of our commonsense reasoning.

In general prioritizing is implemented by forming the expression:

$K1 \equiv (\text{result}(\text{move Zn P Q})) \wedge (\text{defaults}_0\ K0\ K1)$
$\wedge\ K2 \equiv K1 \wedge (\text{defaults}_1\ K0\ K2)$
$\wedge \ldots$
$\wedge\ Kn \equiv Kn\text{-}1 \wedge (\text{defaults}_{n-1}\ K0\ Kn)$
$\wedge\ K \equiv Kn \wedge (\text{defaults}_n\ K0\ K)$

where defaults_i represents the frame laws or default constraints of priority i with the lower number being the higher priority. Therefore to prioritize all the default constraints before all the frame laws we write:

$K1 \equiv (\text{result}(\text{move Zn P Q}))$
$\wedge$ (parallel-plate/plane-constraint
P51 P52 P60 P59 P56 P54 P62 P64 K1)
$\wedge$ (weld-to-keel-constraint)
$\wedge\ K \equiv K1 \wedge (\text{frame K0 K})$

Simplifying K1 results in[7]:

$K1 \equiv$ L72=[P54 P52]
$\wedge$ P56=<120 120 -1/4> $\wedge$ P51=<120 120 0>
$\wedge$ P52=<240 0 0> $\wedge$ P54=<240 0 -1/4>
$\wedge$ P64=<120 (yof P59) -1/4>
$\wedge$ P59=<120 (yof P60) 0>
$\wedge$ P60=<240 (yof P62) 0>
$\wedge$ P62=<240 (yof P64) -1/4>

The y-coordinates of P59, P60, P62, and P64 have not been determined in K1. K1 only states that they are constrained to be equal. It is not until one of them has been propagated in by the frame law that the y-coordinates of these points become known[8]. This result K1 is then used to simplify K giving:

$K \equiv$ L70=[P52 P51] $\wedge$ L72=[P54 P52]
$\wedge$ L74=[P56 P54] $\wedge$ L76=[P51 P56]
$\wedge$ L86=[P60 P59] $\wedge$ L88=[P59 P64]
$\wedge$ L90=[P64 P62] $\wedge$ L92=[P62 P60]

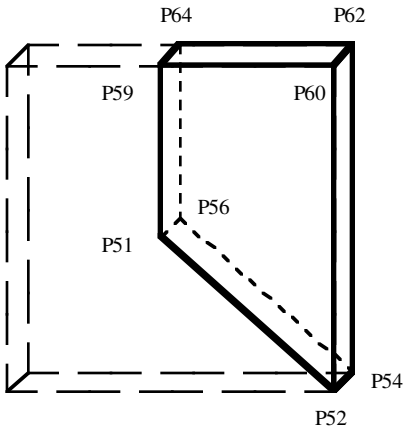
---

[7] The system which applies the KIF definitions in reverse causes symbolic yof values to appear in the point structures.

[8] It does not matter which is propagated first as they are all equal. If at least one was not the same as the others the database would split again and would have to further prioritize the defaults or be willing to accept multiple solutions.

∧ L103=[P59 P51] ∧ L106=[P60 P52]
∧ L119=[P62 P54] ∧ L132=[P64 P56]
∧ P52=<240 0 0> ∧ P54=<240 0 -1/4>
∧ P60=<240 240 0> ∧ P62=<240 240 -1/4>
∧ P56=<120 120 -1/4> ∧ P51=<120 120 0>
∧ P64=<120 240 -1/4> ∧ P59=<120 240 0>

The database again results in the transverse bulkhead:



Fixing an Under-Constrained Design
figure 8

8. CONCLUSION

A formal theory of design change of three dimensional artifacts has been presented where the designs are represented in STEP [3,25] and the design intent constraints are expressed in KIF [18]. This theory is based on modeling the action of changing a design with the use of frame laws and defaults expressed in the modal quantificational logic Z [8,10,12]. Z provides a framework for explaining a number of phenomena including the propagation of properties from one design state to another, the use of constraints and default constraints to specify design intent, the appearance of over-constrained and under-constrained designs, and the use of priority.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. MIP9526532. We thank Dr. Kamal Abdali and Dr. Robert B. Grafton for their helpful suggestions.

REFERENCES

[1]Apostol, Thomas M., 1967. *CALCULUS volume I One-Variable Calculus, with an Introduction to Linear Algebra*, second edition Blaisdell Publishing Company, Waltham, Mass.

[2]Araya, C. and F.M.Brown, 1990. Schemata: A Language for Deduction, *9th. European Conference on Artificial Intelligence*, Stockholm, Sweden 1990.

[3]Bedworth, D., Henderson, M., and Wolfe, P. 1991. Computer-Integrated Design and Manufacturing, Chapter 6. McGraw Hill Inc., New York, NY.

[4]Borning,A.H. 1981. The Programming Language Aspects of ThingLab, a Constraint Oriented Simulation Laboratory, *ACM Trans. on Programming Languages and Systems,* 3(4):353-387.

[5]Boyer,R.S. and J Strother Moore, 1981. Metafunctions: proving them correct and using them efficiently as new proof procedures, *The Correctness Problem in Computer Science,*, Academic Press, New York.

[6]Brown,F.M., 1977. The Theory of Meaning, Dept. of Artificial Intelligence Research Report 35, University of Edinburgh, 1977. Also in *Edinburgh Artificial Intelligence Research Papers 1973-1986*, Scientific Datalink microfische, 1989.

[7]Brown,F.M., 1986 A Commonsense Theory of NonMonotonic Reasoning, *Proceedings of the 8h International Conference on Automated Deduction, CADE8,* Lecture Notes in Computer Science, vol 230, Oxford, England, Springer Verlag, New York.

[8]Brown,F.M., 1986. "An Experimental Logic based on the Fundamental Deduction Principal", *Artificial Intelligence* , vol. 30 no. 2, 146 pages from page 117 to page 263, North Holland.

[9]Brown,F.M. editor, 1987. *The Frame Problem in AI, Proceedings of the 1987 AAAI Workshop*, Morgan Kaufmann Publishers, Los Altos, CA.

[10]Brown,F.M. 1987. A Modal Logic for the Representation of Knowledge, *The Frame problem in AI, Proceedings of the 1987 AAAI Workshop*, Morgan Kaufmann, Los Altos, CA.

[11]Brown,F.M. and S. Park 1987 Action, Reflective Possibility and the Frame Problem, *The Frame Problem in AI, Proceedings of the 1987 AAAI Workshop*, Morgan Kaufmann, Los Altos, CA, 1987. Revised version in 3rd International Conference on CAD/CAM Robotics & Factories of the Future, August 1988.

[12]Brown,F.M., 1991. The Modal Quantificational Logic Z Applied to the Frame Problem, *International Journal of Expert Systems Research and Applications, Special Issue: The Frame Problem. Part A.* eds. Kenneth Ford and Patrick Hayes vol. 3 number 3, pp169-206 JAI Press 1991. Reprinted in *Reasoning Agents in a Dynamic World: The FRAME problem*, editors: K. M. Ford, Patrick J. Hayes, JAI Press 1991.

[13]Brown, Frank .M. et. al. 1993. *Logistica 1.0 Reference Manual*, Artificial Intelligence Research, Inc.

[14]Brown, Frank .M., et. al 1993. *Logistica 1.0 Programmers Manual,* Artificial Intelligence Research, Inc.

[15]Brown,F.M., D.E.Leasure,and N. Peterson, 1993, *Logistica: A High-Level Programming language for Implementing Design Verification Systems*, June AAAI workshop on Systems Engineering.

[16]Brown, Frank M., 1994. *Prologue to a Theory of Design Change and its Automation.* Mid-America Conference on Intelligent Systems, October 1994.
[17]Colmerauer, A., Kanoui, H., Pasero, R. and Roussel, P., 1973. Un Systeme de Communication Homme-machine en Francais, Research Report, Groupe Intelligence Artificielle, Universite Aix-Marseille II, France.
[18]Genesereth, M. and Fikes, R 1992. Knowledge Interchange Format Version 3.0 Reference Manual, Logic Group Report, Stanford University, Stanford.
[19]Hayes, P. J., 1973. Computation as Deduction, Proceedings 2nd MFCS Symposium, Czechoslovakia Academy of Sciences, Prague, Czechoslovakia.
[20]Hilbert,D., 1899. Grundlagen der Geometrie 1899, translated as: The Foundations of Geometry, Open Court Publishing Company 1947.
[21]KowalskiR., 1974. Predicate Logic as Programming Language, *Proceedings of IFIP74* ,North-Holland, Amsterdam.
[22]Kowalski,R., 1979. Algorithm = Logic + Control, *CACM*.
[23]Leasure,D.,&Araya,C. 1994. Defaults and the Frame Law Applied to Overconstrained Problems in Constraint-based Design Systems, Technical Report, Texas A&M University - Corpus Christi.
[24]McCarthy,J. and P.J.Hayes, 1969. Some philosophical problems from the standpoint of artificial intelligence, *Machine Intelligence 4*, Edinburgh, B. Meltzer and D. Mitchie (eds.), University Press, pp. 463-502.
[25]Industrial Automation Systems and Integration - Product Data Representation and Exchange - part 42: Integrated Generic Resources: Geometric and Topological Representation, ISO TC 184/SC4/WG3.
[26]Sutherland,I.B. 1963. SKETCHPAD, A Man-Machine Graphical Communication System. *Proc. Spring Joint Computer Conference*, pp. 329-346, IFIPS.
[27]Wu,Wen-tsun, 1984. Some Recent Advances in Mechanical Theorem-Proving of Geometries, *Automated Theorem Proving: After 25 Years*, AMS, Contemporary Mathematica 29 p239-242,