Applications of Artificial Intelligence in Engineering V

Vol 2: Manufacture and Planning

Proceedings of the Fifth International Conference, Boston, USA, July 1990

Editor: G. Rzevski

Computational Mechanics Publications
Southampton Boston

Co-published with

Springer-Verlag

A Calculus for Assembly

H.H. Hexmoor
Artificials Intelligence Programs, University of Georgia, Athens, Georgia 30605, and *Artificial Intelligence Atlanta, Decatur, Georgia 30030, USA*

Abstract

Manipulating parts in an assembly embodies constraints on parts due to their geometries, motion, and forces involved. Automatic discovery and maintenance of part manipulation constraints will significantly increase the flexibility of adapting the assembly process to the manipulation environment. We will present an axiomatic formalization of Solidobjects, introduced in Hayes [7], and define Assemblies consisting of an organized collection of Solidobjects. Axioms used in defining Assemblies form a calculus for reasoning about a large class of Assemblies in common daily use. A system for maintaining and propagating manipulation constraints will be described. It will be shown that task-level plans can be modified in response to changes in the manipulation constraints.

1 Introduction

Using a CAD/CAM solid model, part masses, and coefficients of friction, a set of constraints is derived for an Assembly. We consider three types of constraints on parts due to their geometries, motion, and forces involved. An Assembly is qualitatively described in terms of the constraints and is a solution for the set of constraints describing it. When a part is added or removed from an Assembly the constraints change. We propose a calculus that derives qualitative descriptions for an Assembly using quantitative CAD/CAM models. The derived qualitative models are concise and describe an Assembly adequately for manipulation tasks and flexible automation.

In the latter part of this paper we describe networks made up of contact, motion, and force constraints used to specify an assembly, called Assembly Domain Constraint Networks (ADCN). Our axioms over constraints are used to propagate consequences of changes through ADCNs in response to observed effects in the environment as well as hypothetical changes used for discovering ramifications of manipulation plans.

Some everyday objects are collections of simpler objects brought into contact relationships. Examples of these objects are pens, cups, layered cakes, houses

from any other object via objects in contact, an Assembly is formed. In this paper we present a definition for Solidobjects, introduced in Hayes [7], and consider an organized collection of them to be an Assembly. A calculus is presented for reasoning over Solidobject parts and constraints among them.

Hayes gave the following two definitions which state that a Solidobject is either made of rigid stuff or otherwise is an Assembly and an Assembly is a collection of solid parts. This allowed for an Assembly to sometimes be a Solidobject itself.

$$(x)\{\text{Solidobject}(x) \rightarrow \text{Assembly}(x) \veebar (\exists m)[\text{Stuff}(m) \wedge \text{Rigid}(m) \wedge \text{madeof}(x) = m]\}$$

$$(x)\{\text{Assembly}(x) \rightarrow (\exists o1,..,\exists on)[\text{Solid}(o1) \wedge ... \wedge \text{Solid}(on) \wedge \text{Part}(o1,x) \wedge ... \wedge \text{Part}(on,x)]\}$$

$\veebar$ in the above definition is the exclusive or connective. We present an axiomatization of the commonsense concepts necessary for defining Assemblies of Solidobjects and reasoning about the process of Assembly and manipulation. Axioms described in this paper are considered as a set of permanent constraints governing reasoning in the Assembly domain. This effort belongs to the discipline of commonsense physics Forbus [4]. It relies on foundational work in the geometry of solids Tarski [15], and the calculus of individuals and mereology deLaguna [3], Whitehead [17], Clarke [1] to establish concepts for reasoning about spatio-temporal entities. We follow related work on naive physics of Hayes [7], naive kinematics of Shoham [13], and qualitative physics of Forbus [4] and Nielsen [11]. Our purpose is to represent Assemblies of Solidobjects in adequate detail so that inherent constraints on manipulation are discovered and maintained. In our formalism we will use first order quantificational theory with identity.

We will begin with a calculus of individuals for spatio-temporal entities followed by a definition and calculus of Solidobjects and an initial definition of an Assembly. We will present the formalisms necessary to reason about local interactions among Solidobjects without regard to the forces involved. We will then introduce forces, and present interactions due to force propagation among objects due to gravity and external forces. This is followed by a discussion of properties and types of an Assembly of Solidobjects and reasoning about the process of Assembly. Finally we will discuss constraints in an assembly and a system for maintaining and propagating manipulation constraints. Since our discussion will be limited to Assemblies of Solidobjects, parts of an Assembly refer to Solidobjects.

2 A Calculus of Spatio-Temporal Entities and Solidobjects

Tarski in his geometry of solids defined solid objects as an arbitrary sum of spheres which he took as primitive. He offered solid objects as an alternative primitive. Unlike Tarski and our earlier attempt Hexmoor and Underwood [8], following mereology (science of parts), spatio-temporal entities are taken as primitive. Individual variables x, y, z in the following range over spatio-temporal entities. We make the following observation: If we limit the spatio-

temporal entities to all belong to a Solidobject, a theory about spatio-temporal entities (mereology) can be used as a theory about chunks of substance making up a Solidobject.

"Being a part of" is the only predicate needed for axioms in mereology. The predicate **Part(x,y)** is used to denote that entity x resides within the space occupied by entity y. The predicate **Part** is reflexive, anti-symmetric and transitive. Two spatio-temporal entities are identical if they are mutually part of one another.

In addition to part relationships, we need to speak about interconnections among them. Whitehead introduced the binary predicate **Connected** by which he meant 'extensionally connected to'. Whitehead did not axiomatize his system. An axiomatization is given by B.L. Clarke [1]. The predicate **Connected** is reflexive and symmetric. We will take **Connected** to be primitive and define **Part** is defined in terms of **Connected**.

Definition: Part

$$\text{Part}(x,y) =_{df} (z)[\text{Connected}(z,x) \rightarrow \text{Connected}(z,y)]$$

The following two axioms are adopted from B. Clarke [1].

Axiom : Mereology

$$(x,y)\{\text{Connected}(x,x) \wedge [\text{Connected}(x,y) \rightarrow \text{Connected}(y,x)]\}$$

Axiom : Uniqueness

$$(x,y,z)[(\text{Connected}(z,x) \leftrightarrow \text{Connected}(z,y)) \rightarrow x=y]$$

The binary predicate **DisConnected** is used to denote disconnected and it is the opposite of the **Connected** predicate.

Definition: DisConnected

$$(x,y)[\text{DisConnected}(x,y) =_{df} \neg \text{Connected}(x,y)]$$

The predicate **Connected** is imprecise with regard to to the types of connections among objects. We introduce the predicate **Contact** to represent objects in contact sharing one or more boundary points but not sharing internal points. This predicate is similar to connected, but it is restricted to connections external to objects. **Contact**(x,y) would mean that two solid objects x and y are "touching". This predicate is anti-reflexive, but symmetric. **Contact** can be defined as follows:

Definition: Contact

$$\text{Contact}(x,y) =_{df} \text{Connected}(x,y) \wedge \neg (\exists z)(\text{Part}(z,x) \wedge \text{Part}(z,y))$$

We can be more exact about the types of contact and define contacts at points, lines, and surfaces. This allows definition of kinematic pairs. Lower kinematic pairs are ones with contact along a surface. Contact between higher pairs is along a line or a point Hunt [9]. The contact types are represented by predicates **Pcontact** for point contact, **Lcontact** for line contact, and **Scontact** for surface contacts. Any theorem that can be proved for **Contact** is applicable to

more specific types of contact, although the reverse does not apply. Contact captures the first form of Assembly domain constraint.

Continuity is an important property of spatio-temporal entities. We capture the property of continuity with the notion of decomposability and argue that spatio-temporal entities are nondecomposable.

Definition: Decomposable
w is decomposable if can be divided into disconnected x and y.

$$\text{Decomposable}(w) =_{df} (x,y)[\text{Part}(x,w) \wedge \text{Part}(y,w) \wedge \text{DisConnected}(x,y)] \wedge \neg (\exists z)[\text{Part}(z,w) \wedge \text{DisConnected}(x,z) \wedge \text{DisConnected}(y,z)]$$

The predicate STE denotes spatio-temporal entities.

Axiom: Nondecomposability

$$(x)[\text{STE}(x) \rightarrow \neg \text{Decomposable}(x)]$$

Next, we will define an object as the largest nondecomposable spatio-temporal entity all made of the same material. The function **madeof** (similar to Hayes') returns the type of material for a spatio-temporal entity if there is one. The predicate **Object** represents an object and is defined below.

Definition: An Object

$$(o)\{\text{Object}(o) =_{df} (\exists x)\{\neg \text{Decomposable}(x) \wedge \text{Part}(x,o) \wedge \neg (\exists y)[\text{Part}(y,o) \wedge \text{Contact}(x,y)]\} \wedge (x)(y)[\text{Part}(x,o) \wedge \text{madeof}(x) = m \wedge y <> x \rightarrow \text{madeof}(y) = m]\}$$

Hager [6] identified invariances in topology, volume, and shape as three important properties of objects . Chunks of substance in an object usually tend to stay together due to bonding among molecules in chunks. Bonding in objects in turn gives rise to topological stability such that chunks of substance maintaining relative position. Hager axiomatized topological invariance in terms of chunks of objects which maintain their relative position over time. A unary predicate **Stable_topology** is used to denote invariance in topology. 'Stable' is used interchangably with 'invariant'. Some objects do not have this property demonstrated by the following frame axioms:

$$(o,x)[\text{Object}(o) \wedge \text{Part}(x,o) \wedge \text{madeof}(x) = \text{mercury} \rightarrow \neg \text{Stable\_topology}(o)]$$

$$(o,x)[\text{Object}(o) \wedge \text{Part}(x,o) \wedge \text{madeof}(x) = \text{butter} \rightarrow \neg \text{Stable\_topology}(o)]$$

Volume invariance is a property of objects that maintain a fixed volume. A unary predicate **Stable_volume** is used to denote invariance in volume. Some objects do not have this property demonstrated by the following frame axioms:

$$(o)(x)[\text{Object}(o) \wedge \text{Part}(x,o) \wedge \text{madeof}(x) = \text{plastic} \rightarrow \neg \text{Stable\_volume}(o)]$$

$$(o)(x)[\text{Object}(o) \wedge \text{Part}(x,o) \wedge \text{madeof}(x) = \text{water} \wedge \text{Frozen}(o) \rightarrow \neg \text{Stable\_volume}(o)]$$

Every object at a given time has a unique shape. Some objects do not maintain their shape. A unary predicate **Stable_shape** is used to denote invariance in shape. Some objects do not have this property demonstrated by the ollowing frame axioms:

$$(o)(x)[\text{Object}(o) \wedge \text{Part}(x,o) \wedge \text{madeof}(x) = \text{paper} \rightarrow \neg\, \text{Stable\_volume}(o)]$$
$$(o)(x)[\text{Object}(o) \wedge \text{Part}(x,o) \wedge \text{madeof}(x) = \text{softwood} \rightarrow \neg\, \text{Stable\_volume}(o)]$$

We are now ready to define those objects that are Solidobjects.

Definition: Solidobject
A Solidobject is an object with the properties of shape invariance, volume invariance, and topology invariance.

$$(o)\{\text{Solidobject}(o) =_{df} \text{Object}(o) \wedge \text{Stable\_topology}(o) \wedge \text{Stable\_volume}(o) \wedge \text{Stable\_shape}(o)\}$$

We note that most everyday objects we may call solid do not fall into our idealization of Solidobjects.

3 A Calculus of Solidobjects and an Assembly

We now consider a collection of Solidobjects and discuss a calculus over them. The predicates **Part**, **Contact**,and **Disconnected** are used as before with the variables ranging over Solidobjects. A calculus is readily available for a collection of Solidobjects with similar axioms as before, namely:

$$(o1,o2)\{\text{Connected}(o1,o1) \wedge [\text{Connected}(o1,o2) \rightarrow \text{Connected}(o2,o1)]\}$$
$$(o1,o2)[\text{DisConnected}(o1,o2) =_{df} \neg\, \text{Connected}(o1,o2)]$$
$$(o1,o2)[\text{Contact}(o1,o2) =_{df} \text{Connected}(o1,o2) \wedge \neg\, (\exists o3)(\text{Part}(o3,o1) \wedge \text{Part}(o3,o2))]$$

A fundamental property of a collection of solid objects is that there is contiguous contact among its Solidobjects. A collection of Solidobjects are in contiguous contact iff any Solidobject can be reached from any other Solidobject by traversing the contact surfaces. The predicate **Reachable** is defined below.

Definition: Reachable

$$(o1,o2)[\text{Reachable}(o1,o2) =_{df} \text{Contact}(o1,o2) \vee (\exists o3)\,(\text{Reachable}(o1,o3) \wedge \text{Reachable}(o3,o2))]$$

In the following definition the second argument of the predicate **Part** is a collection of Solidobjects. This is similar to Hayes' **made-up-of** predicate.

Definition: Contiguous Contact

$$(c)\{\text{Contiguous\_contact}(c) =_{df} (o1)(o2)[\text{Part}(o1,c) \wedge \text{Part}(o2,c) \wedge \text{Reachable}(o1,o2)]\}$$

We are now ready to define an Assembly.

finition: Assembly

Assembly is either a Solidobject or a collection of Solidobjects contiguously nected, but not both.

:)(Assembly(c) =df Solidobject(c) $\vee$ [(o)(Part(o,c) $\rightarrow$ Solidobject(o)) $\wedge$ Contiguous_contact(c)]

e note that a part of an Assembly is still an Assembly.

Spatial Reasoning

o reason about Assemblies, one should be able to reason about local part ating constraints. For this reason we will first focus on reasoning about pairs Solidobjects in contact. We distinguish a local three-dimensional coordinate stem for each Solidobject originating from its origin (geometric center). hoice of axes is dependent on the shape of the Solidobject. Roughly this noice corresponds to the intuitive directions of motion. For example if a olidobject is cylindrical, one of the axes should be chosen to be along the line f symmetry. Clearly we have not presented an automated means of choosing xes and more work remains in this area. Translation is a straight line motion in Solidobject coordinate system decomposable into motions along positive (+) r negative (-) directions of each axis referred to by Tx, Ty, and Tz. Rotation is bout the origin and one of the three axes in either clockwise (+) or counter-lockwise (-) directions. We will refer to rotations as Rx, Ry, and Rz.

rom the property that any two Solidobjects may not share interior points, it ollows that each of two Solidobjects in contact arrest the degrees of freedom of notion of the other. Shoham [13] presented an analysis of the concept of *reedom* which he defined to be motions allowed for one Solidobject when a econd one is stationary. Nielsen [11] provides an alternative representation of reedom regions for translation and rotation which requires definition of *place ocabularies* in configuration space. Instead of defining the locus of free egions for a Solidobject, which is tedious and computationally expensive, we choose to define degrees of freedom along each axis in two stages. First, pairwise constraints between two objects are identified. This is followed by representing the cumulative effects of pairwise constraints for each Solidobject. We later show that we can determine motion in an arbitrary direction.

If a Solidobject is in contact with a second Solidobject and it is blocking motion (translationally) along an axis, that direction on the axis is not "free". The function **freet**, is defined for any two Solidobjects and a translational direction to have a value '0' (zero) if that direction is completely free. The function **freet** has a two-dimensional 'obstruction region' as a value if the second Solidobject completely blocks the first Solidobject along an axis on a given direction. We define an 'obstruction region' for a Solidobject in a dimension against an obstacle by projecting the 'images' of the Solidobject and the obstacle onto the plane formed by the two axes and computing the intersecting region. Freedom regions are useful for precision in specifying motion constraints in the spatial dimensions. An implementation of this function will involve two dimensional sweeps.

Axiom: Translation Freedom
For any contact between a Solidobject and another Solidobject, there are six obstruction regions.

$$(o1,o2)\{\text{contactf}(o1,o2) = c \rightarrow (\exists f1,f1,f3,f4,f5,f6)\{\text{freet}(o1,o2,x+) = f1 \wedge \text{freet}(o1,o2,x-) = f2 \wedge \text{freet}(o1,o2,y+) = f3 \wedge \text{freet}(o1,o2,y-) = f4 \wedge \text{freet}(o1,o2,z+) = f5 \wedge \text{freet}(o1,o2,z-) = f6]\}$$

Similarly, if a Solidobject is in contact with a second Solidobject and it is blocking motion (rotationally) along an axis. The function **freer**, is defined to have as its value the angle through which it is free to rotate and to have a value '1' (one) if it is completely blocked for that rotation.

Axiom: Rotation Freedom
For any contact between a Solidobject and another Solidobject, there are six angles of freedom.

$$(o1,o2)\{\text{contactf}(o1,o2) = c \rightarrow (\exists f1,f1,f3,f4,f5,f6)[\text{freer}(o1,o2,x+) = f1 \wedge \text{freer}(o1,o2,x-) = f2 \wedge \text{freer}(o1,o2,y+) = f3 \wedge \text{freer}(o1,o2,y-) = f4 \wedge \text{freer}(o1,o2,z+) = f5 \wedge \text{freer}(o1,o2,z-) = f6]\}$$

Figure 1 below illustrates that obstacle1 is in contact with a Solidobject and obstacle1 blocks motion in positive z direction for the rectangularly shaped Solidobject, but it is free to move in negative z, positive y direction, and both directions in x. Obstacle2 blocks motion in positive x direction for the Solidobject, but it is free to move in negative x, and both directions in y and z. Obstacle 2 also blocks rotation in positive z direction, and allows about 270 angular degrees of freedom in negative z, '0' rotation in positive y and about 270 angular degrees of freedom in negative y, and totally free for x rotation.

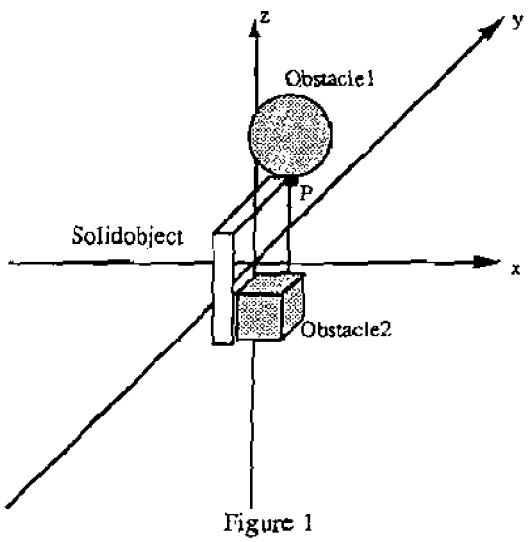


Figure 1

The function $\mathbf{dof_{obs}}$ (an abbreviation of degrees of freedom for a Solidobject and an obstacle which is another Solidobject) identifies for a Solidobject the

obstructions opposing its rotation and translation. This is a function representing pairwise constraints against the motion one Solidobject may produce for another.

Definition: dof_{obs}

$$(o1)(o2)\ dof_{obs}(o1,o2) = [f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f10,f11,f12] =df$$
$$freet(o1,o2,x+) = f1 \wedge freet(o1,o2,x-) = f2 \wedge freet(o1,o2,y+) = f3 \wedge$$
$$freet(o1,o2,y-) = f4 \wedge freet(o1,o2,z+) = f5 \wedge freet(o1,o2,z-) = f6 \wedge$$
$$freer(o1,o2,x+) = f7 \wedge freer(o1,o2,x-) = f8 \wedge freer(o1,o2,y+) = f9 \wedge$$
$$freer(o1,o2,y-) = f10 \wedge freer(o1,o2,z+) = f11 \wedge freer(o1,o2,z-) = f12]$$

For example, dof_{obs}(Solidobject,obstacle1) = [0,0,0,0,R1,0,0,0,0,0,0,0] and dof_{obs}(Solidobject,obstacle2) = [R2,0,0,0,0,0,0,270,270,0].

Axiom : Pairwise Freedom

No Solidobject restricts freedom for itself. If a Solidobject restricts freedom for a second Solidobject, the same restrictions in the opposite direction holds for the second Solidobject as for the first one, provided the Solidobject coordinate systems are conformable.

$$(oi,oj)\{Conformable(oj,oi) \wedge (dof_{obs}(oi,oj) = [a,b,c,d,e,f,g,h,i,j,k,l] \rightarrow$$
$$dof_{obs}(oj,oi) = [b,a,d,c,f,e,h,g,j,i,l,k]\}$$

We define two coordinate systems to be conformable only if like axes are parallel and point in the same direction. The predicate **Conformable** denotes conformable coordinate systems for two Solidobjects.

Information in $\mathbf{dof_{obs}}$ functions are additive and a function **dof** summarizes the results of all $\mathbf{dof_{obs}}$ functions on a Solidobject. The predicate **Subsumes** is used to denote motion constraint subsumption. In Figure 1, dof(Solidobject)= [R2,0,0,0,R1,0,0,0,0,270,270,0].

Definition: dof

$$dof(o) = [f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f10,f11,f12] =df$$
$$(oi)(dof_{obs}(o,oi) = [fi1,fi2,fi3,fi4,fi5,fi6,fi7,fi8,fi9,fi10,fi10,fi11,fi12] \wedge$$
$$Subsumes(f1,fi1) \wedge Subsumes(f2,fi2) \wedge Subsumes(f3,fi3) \wedge$$
$$Subsumes(f4,fi4) \wedge Subsumes(f5,fi5) \wedge Subsumes(f6,fi6) \wedge$$
$$Subsumes(f7,fi7) \wedge Subsumes(f8,fi8) \wedge Subsumes(f9,fi9) \wedge$$
$$Subsumes(f10,fi10) \wedge Subsumes(f11,fi11) \wedge Subsumes(f12,fi12)$$

A direction is represented by a triplet (sx,sy,sz) of signs each of which corresponds to a direction in a dimension. Signs are one of {+, -, | } corresponding to towards, against, and neutral. Freedom of a Solidobject translation in an arbitrary direction is determined using the predicate **Blocked.**

Axiom: Arbitrary Translation

An arbitrary direction (sx,sy,sz) is blocked against translation, if one of the corresponding directions in the Solidobject's dof set is nonzero (free).

$$(o)\ \{dof(o) = [f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12] \wedge$$
$$[(d\_pat = (+,+,+) \wedge (f1 <> 0 \vee f3 <> 0 \vee f5 <> 0)) \vee$$
$$(d\_pat = (+,+,-) \wedge (f1 <> 0 \vee f3 <> 0 \vee f6 <> 0)) \vee (d\_pat = (+,+,|)) \wedge$$
$$(f1 <> 0 \vee f3 <> 0)) \vee \ldots\ldots]$$
$$\rightarrow Blocked(d\_pat)\}$$

5 Forces

We are interested in forces which are either received by Solidobjects or imparted to other Solidobjects via contact. The notion of a *tendency* is adopted to represent the relationship between a force and a Solidobject. A tendency is a behavioral response property of a Solidobject when there is a force acting at a surface point of a Solidobject. Common tendencies for Solidobjects are tendencies to fall, to translate, and to turn. The predicate **Force** is used to represent a force. The predicate **Point** is used to represent a point. The predicate **Tendency** will be used to represent a tendency with a Solidobject, a force, and a point of the Solidobject. We adopt the following axiom about tendencies similar to Hager's.

Axiom: Tendency
A Tendency will have a Solidobject, a point of the Solidobject and a Force applied at the point on the surface of the Solidobject.

$$(x, o,f,p)[Tendency(x,o,f,p) \rightarrow Solidobject(o) \wedge Point(p) \wedge Force(f)]$$

A force has a magnitude, a point of application, and a direction of application. The function **magnitude** is used to represent the magnitude of a force. In order to specify directions, we consider a three-dimensional coordinate system corresponding to the three-dimensional space of objects. The function **direction** will return a triplet (Sx,Sy,Sz) of signs each of which corresponds to a direction in a dimension. Signs are one of {+, -, | } corresponding to towards, against, and neutral. The function **where** identifies the point at which a force meets a Solidobject. Our scheme of direction patterns is inadequate for representing circular directions. Therefore forces applied in circular directions are not considered. Examples of forces with circular directions are bending and twisting, Hager [6].

Axiom: Force
A force will have a point of application, a magnitude, and a direction.

$$(f)\{Force(f) \rightarrow (\exists d\_pat,p,m)[direction(f) = d\_pat \wedge where(f) = p$$
$$\wedge\ magnitude(f) = m]\}$$

Every Solidobject on earth experiences a common tendency due to the earth's gravitational pull that we will call the tendency to fall. The magnitude of a force associated with the tendency to fall is known as the Solidobject's weight and we will use the function **weight** to represent it. Also, the direction of this tendency is uniquely known to be pointing toward the ground. This unique direction in our coordinate system will be represented by the constant name **toward_ground**. Later we will show that any Solidobject at rest exerts a force on Solidobjects with which it is in contact equal to its weight. The function

Axiom: Tendency to fall
For every Solidobject there is a tendency to fall with a force magnitude equal to the Solidobject's weight from the origin of the Solidobject pointing toward the ground.

$$(o)(\exists f,p)[\text{Tendency}(\text{fall},o,f,p) \rightarrow \text{magnitude}(f) = \text{weight}(o) \wedge \text{direction}(f) = \text{toward\_ground}]$$

Since we are interested in analyzing the propagation of forces among Solidobjects in contact we identify three types of forces for a Solidobject: (1) weight, (2) input forces, and (3) output forces. Figure 2 shows these forces associated with a Solidobject. The binary predicate **Input** denotes an input force for a Solidobject. Input forces give rise to tendencies to translate and turn. We will discuss these tendencies later in this section. The binary predicate **Output** denotes an output force from a Solidobject.

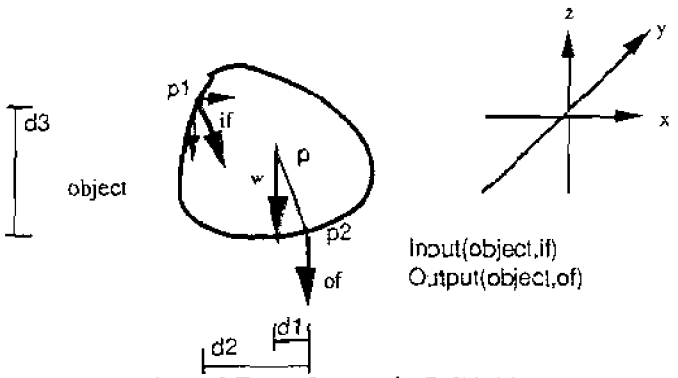


Figure 2 Three forces of a Solidobject

Two forces acting at a common point can be summed to a unique equivalent force.

Axiom: force addition at a point
Two forces acting at a point are additive and produce a unique force acting at the same point with a magnitude equal to addition of magnitudes of the two forces with a • direction.

$$(f1,f2)(\exists f3)[\text{Force}(f3) \rightarrow \text{Force}(f1) \wedge \text{Force}(f2) \wedge \text{where}(f1) = \text{where}(f2) = \text{where}(f3) \wedge \text{direction}(f1) \bullet \text{direction}(f2) = \text{direction}(f3) \wedge \text{magnitude}(f3) = \text{magnitude}(f1) + \text{magnitude}(f2)]$$

A qualitative algebra is used for addition of any two forces. We used • as the symbol for qualitative algebra. If two forces with (d1x,d1y,d1z) direction pattern and (d2x,d2y,d3z) direction pattern are applied at the same point, they produce a resultant force with (dx,dy,dz) where if d1i and d2i are like signed or one is neutral, di takes the non-neutral sign. If d1i and d2i are signed opposite, quantities of forces are compared and di adopts sign of the one with dominant force.

Axiom: Parallel force addition at a distance
Two forces parallel and away from a point of interest by distances d1 and d2 are additive and produce a unique force acting at a distance d3 from the point of interest with a magnitude proportional to them and the distance.

$$(f1)(f2)(d)\,(\exists f3)[\text{Force}(f3) \rightarrow \text{Parallel}(\text{direction}(f1),\text{direction}(f2)) \wedge \text{where}(f1) <> \text{where}(f2) \wedge \text{distance}(f1) = d1 \wedge \text{distance}(f2) = d2 \wedge \text{direction}(f3) = \text{direction}(f1) \bullet \text{direction}(f2) \wedge \text{magnitude}(f3) = (\text{magnitude}(f1)*d1 + \text{magnitude}(f2)*d2\,)/d]$$

We introduce a binary function **padd** to return the resultant force of two parallel forces as given in the above axiom.

Moments are forces applied a rotation point. The predicate **Moment** denotes a moment at a point. The function **distance** returns the shortest distance value between a point a force direction.

Definition: Moment
A moment about a center of rotation point will have a force, a point of application, a direction, a perpendicular distance d between the center of rotation the force line and point of application, and a magnitude equal to the product of d and the force magnitude.

$$(m,p)\{\text{Moment}(m,p) \rightarrow (\exists f,d\_pat,p,d)[\text{Force}(f) \wedge \text{direction}(m) = d\_pat \wedge \text{magnitude}(f) = fm \wedge \text{magnitude}(m) = fm * d]\}$$

Since forces are additive, moments generated due to concurrent forces about the same point in space are moments due to resultant force and therefore additive. Varignon's Theorem Tuma [16] is a version of this theorem for coplanar concurrent forces.

Theorem: Moment additivity about a point
Two moments generated about a common rotation point are additive and produce a unique moment acting at the same center of rotation point with a magnitude equal to addition of magnitudes of the two forces.

$$(m1,m2,p)(\exists m3)[\text{Moment}(m3,p) \rightarrow \text{Moment}(m1,p) \wedge \text{Moment}(m2,p) \wedge \text{magnitude}(m3) = \text{magnitude}(m1) + \text{magnitude}(m2)]$$

As shown in Figure 2 an input force to a Solidobject can be decomposed into forces one of which in the direction of **toward_ground**. The other components of this force will contribute to moments about given points on the surface of a Solidobject. The predicate **Parallel** identifies parallel directions.

Theorem: Force Reaction
Parallel component of an input force of a tendency to translate acting at a point on a given Solidobject is parallel added with the force of Solidobject's falling tendency which produces an output force for the Solidobject.

$$(o,p)\{\text{Tendency}(\text{translate},o,f1,p1) \wedge \text{Tendency}(\text{falling},o,w,op) \rightarrow (\exists f2)[\text{Padd}(f1,w) = f2 \wedge \text{Output}(o,f2)]\}$$

We define the shadow of an output force for each Solidobject to be the area of primary concern for potential force transfer from one Solidobject to another.

For a Solidobject and an output force the function **shadow** will return the space that will be covered by the Solidobject if the Solidobject starts moving due to its output force. An output force of a given Solidobject has a unique shadow. Figure 3 depicts a two-dimensional example of a shadow which is constructed by extending the extremum points of the Solidobject with respect to the Solidobject's output force. This scheme is easily extendable fto three-dimensional Solidobjects.

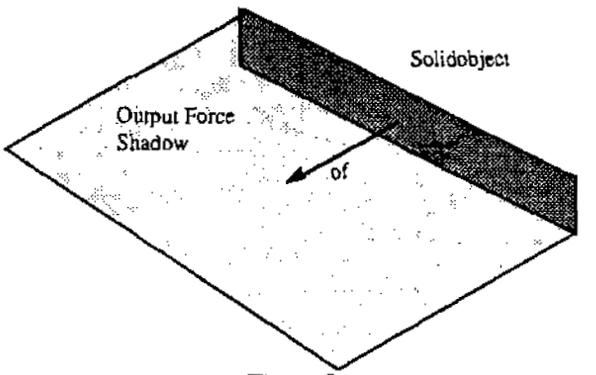


Figure 3

We are now ready to characterize force distribution. The types of forces we are concerned with are transferable from one Solidobject to another only through contact between the two objects. We will discuss what happens to forces at the surfaces of objects after we analyze how to distribute forces among contact points. Figure 4 illustrates the gist of our axiom. Proportionality comparison of forces can be represented by qualitative order relations. In Figure 4, f1 < ef, f2 < ef, and f1 >f2.

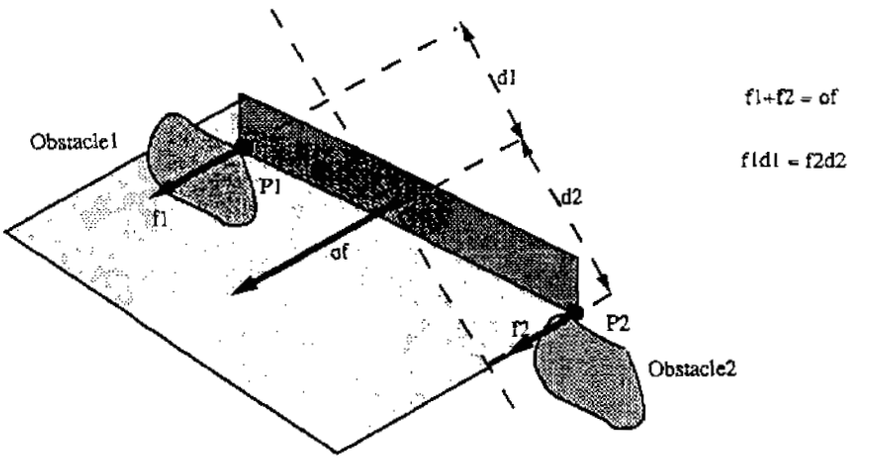


Figure 4

Axiom: Force distribution
When a Solidobject contacts obstacles in its shadow of output force, forces distributed among contact points among obstacles are proportional to distances between contact points and the normal to the output force from the Solidobject.

$$(o,oi,f)\{shadow(o,f) = s \wedge Output(o,f) \wedge Solidobject(oi)$$
$$\wedge\ contactf(oi,o) = c \wedge In(c,s) \rightarrow (\exists fi)[Tendency(push,o,fi,p)\}$$

contactf is a function which returns the contact area between two Solidobjects. The predicate **In** characterize the intersection of two areas (regions) like contact area and shadow. In the static analysis of objects, forces are decomposed into forces normal and forces tangent to the contact surfaces. Opposite forces cancel at the contact areas. This analysis assumes that one of the two objects is secured, which is tantamount to assuming infinite equilibrium limits for the Solidobject. Only the ground and workbenches may qualify for this assumption.Therefore by relaxing this assumption, we are ready for the first axiom of force transfer which says that the output force of a Solidobject becomes the input force for the single Solidobject with which it is in contact. Figure 5 depicts the results stated in our axiom. We are ignoring friction in our first model of force transfer. After talking about friction, we will revisit force transfer.

Axiom : Force Transfer W/O Friction
If a Solidobject gains contact in the shadow of output force with a second Solidobject, it will gain an input force equal to the output force.

$$(o1,o2)[contactf(o1,o2) = c \wedge Output(o1,f) \wedge shadow(o2,f) = s \wedge In(c,s)$$
$$\rightarrow Input(o2,f)]$$

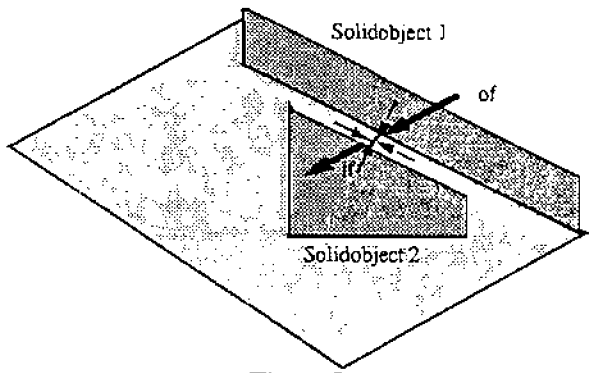


Figure 5

Once an input force is established for a Solidobject, we can use the reactions to determine the new output forces. Unlike previous attempts like Jockowicz' *Contact Rule* [5], our axiom does not distinguish the type of contact between two objects.

Dry friction is a force developed between two Solidobjects in contact that resist a tendency to slide. Unlike other types of forces, dry friction gives rise to a potential tendency to slide only when there are opposing tendencies. The magnitude of the dry friction force is proportional to the types of molecular

bonding between substances of Solidobjects in contact. In this paper we only consider dry friction. We will refer to dry friction as friction and the predicate **Friction** denotes a friction force for a Solidobject when in contact with another Solidobject. The function **normal** returns the component of a force normal to the surface of a Solidobject. The function fc returns a friction coefficient between two substances when two Solidobjects made of them are in contact.

Axiom: Friction
There is a normal component of an input force for a Solidobject in contact with another Solidobject with a magnitude equal to the product of a friction coefficient and the magnitude of the normal component of the input force.

$$(o)\{Friction(f1,o) \rightarrow (\exists f,o1)[Contact(o,o1) \wedge Input(o,f) \wedge normal(f,o) = n \wedge fc(madeof(o1),madeof(o2)) = ß \wedge magnitude(f1) = n * ß]\}$$

We will now revisit the Axiom for force transfer including friction.

Axiom : Force Transfer With Friction
If a Solidobject gains contact in the shadow of output force with a second Solidobject, it will gain an input force equal to the normal component of the output force.

$$(o1,o2)[contactf(o1,o2) = c \wedge Output(o1,f) \wedge shadow(o1,f) = s \wedge In(c,s) \rightarrow (\exists f1)\, Snormal(f1) \wedge Input(o2,f1)]$$

Any Solidobject at rest has an envelope of tolerance or equilibrium for forces acting on it before it loses its rest status. We define the predicate **Snormal** to denotes a surface normal direction. Equilibrium is a composite of all dry friction forces due to all contacts between a Solidobject and other Solidobjects. We capture the equilibrium limits for a Solidobject by the function **equilibrium** which returns a set of triplets contact area, surface normal, and dry friction force, (C,N,F). Our concept of equilibrium is broader than the static equilibrium where the resultant forces and couples add up to zero. See Tuma [16] for a definition of static equilibrium.

Definition: Equilibrium
Every element of the equilibrium set of a Solidobject indicates a contact with a Solidobject, a surface normal, and a friction force.

$$(o)\{(c,n,f) \in equilibrium(o) \rightarrow (\exists o1)[Contactf(o,o1) = c \wedge Snormal(n) \wedge Friction(f,o)]\}$$

The simplest case of equilibrium is when a Solidobject is at rest on the Ground or on a Workbench, (G/W), since the G/W can be assumed to have infinite equilibrium limits. A strategy for computing equilibrium for a given structure of Solidobjects is first to determine equilibrium for objects in contact with the G/W, then objects in contact with them, and so on.

A Solidobject tends to move if it overcomes its bounds of static equilibrium. The tendency to move may manifest itself as translation or rotation. The following axioms hold for tendencies to translate and turn.

Axiom: Tendency to translate
A tendency to translate is associated with the output force of a Solidobject when it exceeds the bounds of its equilibrium.

$$(o)[\text{Tendency}(\text{translate},o,f,p) \rightarrow \text{Force}(f) \wedge \text{Exceeds}(\text{magnitude}(f),\text{equilibrium}(o))]$$

Axiom: Tendency to turn
A tendency to turn about a point on the surface of a Solidobject is associated with an input force when the moment generated exceeds the bounds of its equilibrium.

$$(o)[\text{Tendency}(\text{turn},o,f,p) \rightarrow (\exists p1)\text{Input}(o,f) \wedge \text{Point}(p1,o) \wedge \text{Moment}(m,p1) \wedge \text{Exceeds}(\text{magnitude}(m),\text{equilibrium}(o))]$$

6 Assembly Revisited

In this section we present properties common to all Solidobject Assemblies, define types of Assemblies of Solidobjects with special properties, and consider Semi-solidobjects which are more prevalent in our daily lives.

A property of all Solidobject Assemblies is that if a part free to move in a direction gains a tendency to move in that direction, no other part gains a tendency to move. We call this property being reversible. In an Assembly of Solidobjects every part has the property of being reversible. The predicate **Reversible** is used to represent an object in a collection is reversible.

Definition: Reversible
If an object in a collection is reversible iff it is free to move (translate, turn, or fall) in a direction and it gains a tendency to translate in that direction, implies that no other part will have a tendency to move.

$$(o,c)\{\text{Reversible}(o,c) =_{df} \text{Part}(o,c) \wedge \text{Direction}(f) = d \wedge \neg \text{Blocked}(d) \wedge \text{Tendency}(x,o,f,p) \wedge (x = \text{translate} \vee x = \text{turn} \vee x = \text{fall}) \rightarrow \neg (\exists o1)[(o1 <> o) \wedge \text{part}(o1,c) \wedge \text{Tendency}(\text{translate},o1,f,p)]\}$$

Axiom: Solidobject Assemblies are Reversible
All Solidobjects of an Assembly are reversible.

$$(o,c)[\text{Assembly}(c) \wedge \text{Part}(o,c) \rightarrow \text{Reversible}(o,c)]$$

Our earlier definition of Assemblies considered a contiguously connected collection of Solidobjects to be an Assembly of Solidobjects. The predominant feature of these Assemblies is contiguous contact among a collection of Solidobjects. This definition includes hinged Assemblies of Solidobjects - ones that allow for relative motion among Solidobjects without loss of contiguous contact. Examples are automobiles, and swings. We consider a class of Assemblies which further restrict Assemblies to those with no moving parts in their design. We call this class of Assemblies static assemblies and the Predicate **Static_Assembly** identifies a collection of Solidobjects which is an Assembly with no relative motion between Solidobjects. Examples in this class of Assemblies are houses of cards and stacks of blocks, and some chairs. In a static Assembly if a part gains a tendency to move, all other parts gain the same tendency to move.

Definition: Static Assemblies
A static Assembly is an Assembly where if a part gains a tendency to translate, to turn, or to fall, then every other part will have that tendency.

$$(c)\{\text{Static\_Assembly}(c) =_{df} \text{Assembly}(c) \wedge$$
$$(o)[\text{Part}(o,c) \wedge \text{Tendency}(x,o,f,p) \wedge (x = \text{turn} \vee x = \text{translate} \vee x = \text{fall})$$
$$\rightarrow (o1)[(o1 <> o) \wedge \text{part}(o1,c) \wedge \text{Tendency}(x,o,f,p)]]\}$$

A large portion of everyday objects are static Assemblies most of the time, but allow for relative motion among parts characterized by a propagation of a tendency to move in predefined ways such that when one part gains a tendency to move other parts experience a different tendency to move. A four-bar-linkage is example of a type of Assembly where every part gains a tendency to move when one part gains a tendency to move.

We call the class of Assemblies with independent subassemblies modular Assemblies. If a Solidobject gains a tendency to move in a direction for which it is not blocked, a subset of remaining Solidobjects which are in contact with it gain the same tendency. The subset and the Solidobject in consideration are called an independent sub-Assembly. The predicate **Modular_Assembly** identifies a modular Assembly.

Definition: Modular Assemblies
A modular Assembly is an Assembly in which a subset of its parts will gain a tendency to move, when any part of the Assembly gains a tendency to move.

$$(c)\{\text{Modular\_Assembly}(c) =_{df} \text{Assembly}(c) \wedge$$
$$[(\exists o)[\text{Part}(o,c) \wedge \text{Tendency}(x,o,f,p) \wedge (X = \text{turn} \vee x = \text{translate} \vee x = \text{fall}) \rightarrow (o1)(o1 <> o) \wedge \text{Contact}(o,o1) \wedge \text{part}(o1,c) \wedge \text{Tendency}(x,o,f,p)]]\}$$

More definitions and refinements in this area are needed. For instance, a large class of Assemblies employ fasteners like screws which are often themselves Solidobjects. Hinges are another popular means of guiding predefined motion among parts of an Assembly.

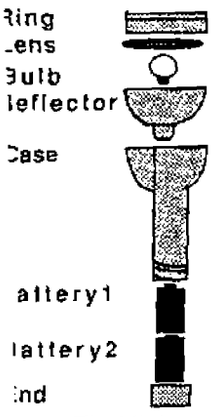
Our calculus for Solidobjects has left out the largest class of everyday objects used in Assemblies. Examples of these objects are carpentry wooden boards, hard plastic parts molded or extruded, and most metal alloy manufactured parts. These objects allow very small changes in their shapes in response to input forces while retaining topology invariance and volume invariance. Input forces to these objects may produce tendencies to stretch, compress, twist, and bend to a limited degree. Another property of some of these objects is springiness, a tendency to return to their original shape. We leave an analysis of this class of objects to future efforts.

7 Manipulation Constraints

Contact, motion, and force constraints inherent in an assembly are used to specify an assembly represented in a network, an Assembly Domain Constraint Networks (ADCN). Flexibility is achieved in manipulation planning since

naintain consistent models of assemblies and the burden of specifying ions of planning operators is shifted to ADCN management. ADCNs networks of assembly domain constraints consistent with the axioms our calculus.

der contacts between two parts sharing boundary points as a binary . Configuration of an assembly can be specified with a finite set of onstraints. Motion constraints represent relative motions allowed wo parts. This is used in determining strategies for dislodging a part sembly or bringing parts into contact. Motion constraints come in two rwise constraints are between two parts and represented by a binary Each part having contact constraints with other parts is said to have a motion constraint which can be represented by a unary constraint for Vhen in contact with other parts force constraints represent relative each part and forces a part may receive and transmit to other parts. orward inference to propagate values through constraints in the This is known as *assimilation*, Davis [2]. ADCNs are hybrid since ent three types of constraints and their associated reasoning. ADCNs entally constructed. Initially, an ADCN is a graph with a node for nd contact and motion constraints as edges of the graph. Contact and otion constraints are *nonconstructive* in the sense that they do not gnment of values to nodes. An additional function of contact is indexing of parts as to shared boundary points since motion apply only if there is contact constraint between two parts. In general motion constraints without contact constrains but we will consider straints only when there is a contact constraint between two parts. /e will only consider force constraints for two parts if there is a nstraint between them.Figure 8 shows an ADCN for a simple epicted in Figure 6 made up of 8 parts. The flashlight example was ed by Sanderson and deMello [12].



mple Flashlight

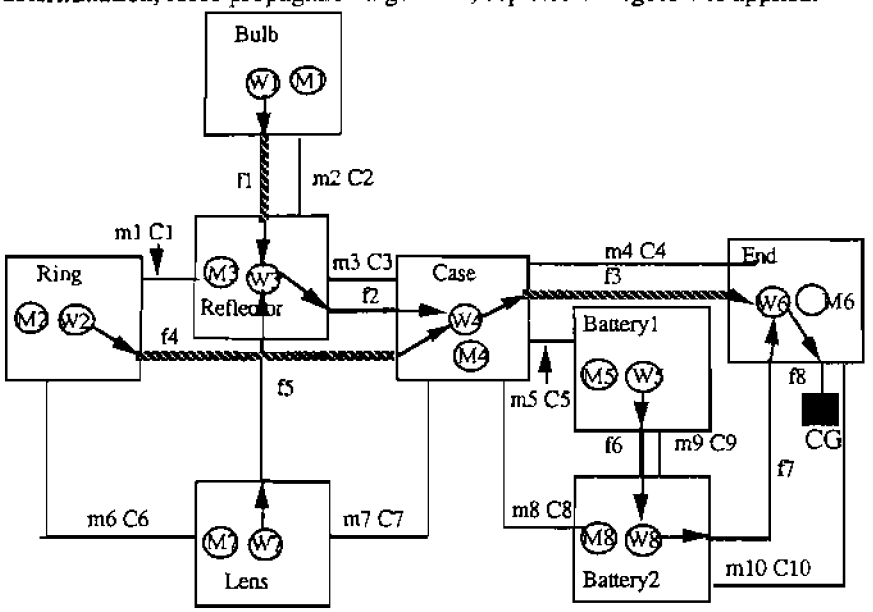
Set the new part output force as the object's new output
While (CG has not received an input force)
a) compute shadows of all new-output-forces,
b) compute input allocation for objects in contact,
c) for every object receiving an input force propagate new-output-forces for the object

Figure 7 Constraint propagation algorithm

nilated to an ADCN when a new contact constraint is reported and id contact constraint edge is established.

over nodes and contact constraint edges are bidirectional. Pairwise motion constraints are specified by the predicate $\mathbf{dof_{obs}}$ for all parts having contact constraints. Cumulative motion constraints are computed for parts that have pairwise motion constraints with other parts. Since these constraints are in the object's local coordinate system, a transformation is needed to compute motion constraints in terms of a global coordinate system. When a new contact constraint is added to an ADCN, contact constraints are reported to the model generator which in turn are added to the ADCN.

Force constraints are constructive since they facilitate assignment of of force values to parts. We consider nodes to have internal structures. In Sussman and Steele [14] terms, each node is said to be a cell and of type Solidobject. Let us assume parts as having point masses in which case each Solidobject has three components, a force that it accepts as input, its force output, and an internal force due to its weight. For example, an object resting on a table experiences an internal force from the ground due to gravitational pull and exerts an output force to the table. If the object receives an input force like being pushed down on the table, its output force to the table becomes greater by the amount of force it receives. This transmittal of force to the table is captured as propagation of force values through force constraints. The table is assumed to absorb infinite force without moving. We call such an object as the *Commonsense Ground* (CG). As a part is assimilated to an ADCN, following contact and motion constraint determination, force propagation algorithm, depicted in Figure 7 is applied.



Mi- Cumulative motion constraint for part i
Wi- Weight of part i

Figure 8 An ADCN for a simple flashlight standing upright

8 Summary and Conclusion

We have defined Solidobjects and Assemblies of Solidobjects. We characterized concepts involved in spatial reasoning and reasoning about forces on

Solidobjects. If we limit spatio-temporal entities to be Solidobjects, a theory about spatio-temporal entities (mereology) can be used as a theory about chunks of substance making up a Solidobject. Reasoning about the mating of Solidobject parts requires reasoning about their contact geometries and ways in which they arrest motion for one another in local coordinate systems of Solidobjects.

Our exploratory work in representing Assemblies supports automated reasoning for robotic assembly. We described a constraint based system for Solidobject Assemblies. Our calculus allow us to represent and reason about simple relations among Solidobjects involving contact, motion, and forces. A network of manipulation constraints was introduced and termed ADCN. An ADCN maybe used by the planner as a set of conjunctive goals, as a consistent world model, or a means of lookahead search. Furthermore, flexibility is achieved in planning since ADCNs maintain consistent models of assemblies and the burden of specifying ramifications of planning operators is shifted to ADCN management.

We have presented a very simplified static model of forces acting on objects to illustrate their use in ADCN management. More realistic laws for forces acting on objects can be developed and integrated for force propagation in ADCNs. Also ADCNs can be extended for dynamic analysis by considering axiomatized kinematic and dynamic laws of motion. Our definition of Solidobjects is not realistic since most objects considered solid allow small perturbation in their shape and some have the property of springiness. We consider extensions of our calculus in the following two areas: 1) an analysis of fasteners which are themselves Solidobjects, and 2) properties of materials which allow small perturbations in their shape and some that have the property of springiness. We ignored fasteners in this paper. See Miller and Hoffman [10] for more on fasteners.

Acknowledgement

I wish to thank Dr. William Underwood for his enthusiasm and constructive suggestions and Professor Donald Nute for comments on an earlier draft of this paper. This research was supported by the internal research and development funds of Artificial Intelligence Atlanta, Inc. 119 East Court Square, Decatur, Georgia 30030.

References

1. Clarke, B.L. A Calculus of Individuals Based on 'Connection', Notre Dame Journal of Formal Logic, Vol. 22, no. 3, 1981.

2. Davis, E. Constraint Propagation with Interval Labels, Artificial Intelligence 32, 281-331, 1987.

3. de Laguna, T. Points, Lines and Surfaces, as Sets of Solids, The Journal of Philosophy, Vol. 19, pp. 449-461, 1922.

4. Forbus, K.D. Commonsense Physics: A Review, In Annual Review of Computer Science 3, pp. 197-132, 1988.

5. Jockowicz, L. Shape and Function in Mechanical Devices, Proceedings of

6. Hager, G. Naive Physics of Materials: A Recon Mission, In Commonsense Summer: Final Report. CSLI-85-35, 1985.

7. Hayes, P.J. Naive Physics I: Ontology for Liquids, In Formal Theories of the Commonsense World, (EDs. J. Hobbs and R. Moore), Ablex Publishing, NJ, 1985.

8. Hexmoor, H.H., Underwood, W.E. An Ontology and Representation for Flexible Assembly, In AAAI Spring Symposium, Stanford University, CA, 1989.

9. Hunt, K.H. Kinematic Geometry of Mechanisms, Clarendon Press, New York, 1978.

10. Miller, J.M. and Hoffman, R. L. Automatic Assembly Planning with Fasteners, In Proc. IEEE Int Conf on Robotics and Automation, 1989.

11. Nielsen, P.E. A Qualitative Approach to Rigid Body Mechanics, Proceedings of AAAI-7, pp. 270-274, 1988.

12. Sanderson, A.C. and Homem-de-Mello, L.S. Task Planning and Control Synthesis for Flexible Assembly Systems, NATO ASI Series, Vol F33, pp. 331-353, Springer-Verlag, 1987.

13. Shoham, Y. Naive Kinematics: One Aspect of Shape. In Proccedings of IJCAI-8. pp 436-442, 1985.

14. Sussman, J.G. and Steele,G.L. CONSTRAINTS - A Language for Expressing Almost-Hierarchical Descriptions. Artificial Intelligence 14, pp. 1-39, 1980.

15. Tarski, A. Foundations of the Geometry of Solids. In Logic, Semantics, Mathematics. transl, J. H. Woodger, Hackett Publishing, IA, 1983.

16. Tuma, J. Handbook of Physical Calculations. McGraw-Hill Book Company, New York, 1976.

17. Whitehead, A.N. Process and Reality. The MacMillan Company, New York, 1929.