Intelligent Computer Aided Design
D.C. Brown, M. Waldron and H. Yoshikawa (Editors)
Elsevier Science Publishers B.V. (North-Holland)
© 1992 IFIP. All rights reserved.

Geometric Constraint Management with Quanta*

Remco C. Veltkamp

CWI, Department of Interactive Systems
Kruislaan 413, 1098 SJ Amsterdam, the Netherlands
e-mail: remco@cwi.nl

This paper presents an incremental approach to geometric constraint satisfaction that categorizes solutions into so called quanta. A quantum is a geometric primitive describing a range of solutions with uniform geometric characteristics. Both alternative discrete solutions and continuous ranges of solutions are represented by quanta. This approach is suitable for interactive design because it can handle (perhaps temporarily) incomplete specifications and alternative solutions, and performs satisfaction locally and incrementally. The system keeps the intermediate solutions in the geometric domain, so that new geometric constraints can be interpreted on the same high level of abstraction, allowing powerful reasoning. Temporarily unsolvable constraints are handled by delayed satisfaction in a uniform way.

1 Introduction

Constraints are used in a wide range of applications including human-computer interfaces [Szekely and Meyers, 88], simulation environments [Steele Jr. and Sussman, 79], animation [Badler and Kamran, 87], and geometric modeling. Constraints specify relations between objects that must be satisfied and maintained. A *geometric* constraint can fix one or more degrees of freedom for positioning, orientation and dimensioning. For example, when a circle of fixed radius is constrained to be tangent to a fixed line segment, the position of the circle center is restricted to two line segments parallel to the given one, at a distance that equals the radius.

A constraint satisfaction system relieves some of the workload on the user: problems can be solved by specifying constraints, and the user need not specify how to solve the constraints. Even if the system cannot satisfy all constraints that can occur in a given domain, it frees the user of solving the many little problems, something which is still a

*This research is part of the IIICAD project, supported by NWO (Dutch Organization for Scientific Research) under Grant NF-51/62-514

time-consuming and error-prone activity. Constraint management is therefore an important facility in design systems.

This paper is about *geometric* constraint management, supporting the geometric modeling part of a design task. Geometric constraint satisfaction can be part of a basic layer of a design system, but this layer in itself is not an application system. It provides a good basis for CSG modeling and Euler operators [Mäntylä, 88], feature-based modeling [Pratt, 87], and intelligent human–computer interaction interfaces [Helander, 88].

In order to provide meaningful feedback to the user, incremental constraint satisfaction is needed in interactive and incremental design applications. Because under-constrained specifications are typical in incremental design, we focus on the representation and manipulation of solutions in under-determined situations. Our approach categorize alternatives or ranges of solutions into classes featuring the same characteristics: quanta. We exploit geometric manipulation [Arbab and Wing, 85], as opposed to algebraic manipulation and numerical computation, to find solutions of geometric constraints. General geometric knowledge is combined with properties of (classes of) geometric objects to infer relations between geometric objects, in order to satisfy constraints. This high level geometric reasoning provides geometrically meaningful solutions and tries to avoid computationally expensive algebraic and numerical methods.

The rest of this paper is organized as follows. The next section gives an introductory overview of some constraint satisfaction systems that have been developed, and explains some shortcomings with respect to geometric modeling that motivates the present research. Section 3 introduces the quantum approach, and section 4 describes the components of our constraint management system. Section 5 presents two elaborate case studies that illustrate our method in 2D and 3D. Finally, section 6 gives some conclusions and suggests future research directions.

2 Overview

Some constraint systems dealing with geometry, as well as general purpose constraint languages are mentioned below, together with the satisfaction techniques that they use. In the subsequent subsection we mention some problems that motivate our current research.

2.1 Constraint systems

Sketchpad [Sutherland, 63] is the first constraint-based drawing system. It satisfies constraints using propagation of degrees of freedom. When this fails, it resorts to relaxation.

[Lin et al., 81] describe a system that translates dimensional constraints into a system of equations, which is solved numerically by the Newton-Raphson method. The dimensional constraints are defined by equations on coordinates of characteristic points. This method, called variational geometry, is unstructured: every time a dimensional value is changed, the whole system of equations must be solved.

ThingLab [Borning, 81] enlarges the possibilities of Sketchpad with extensibility and object oriented techniques, so that new classes of objects and constraints can be defined. It uses both propagation of degrees of freedom and propagation of known states.

Juno [Nelson, 85] is a simple system based on one geometric primitive: the point. It exploits a Newton-Raphson iteration technique to solve constraints. The user must supply an initial value to start the iteration.

In [Brüderlin, 86], sketched polyhedra are dimensioned by constraints. The constraints are restricted to relations between the vertices. The system evaluates the constraints symbolically, and can detect inconsistency. However, it makes no distinction between redundant and inconsistent information. An object is defined, if for each vertex there exists a predicate determining its position.

[Rossignac, 86] presents an operational interpretation of constraints in CSG modeling. Constraints are specified by the user in terms of relations between boundary features, and are transformed by the system into rigid motions of parts of the CSG tree. An under-constrained situation can simply not occur. The user must specify the order of evaluation, and is responsible for solving conflicts.

OTP (Operational Transformation Planning) [Arbab and Wang, 89] also provides an operational interpretation of constraints. It exploits solution reduction, one solution is presented in an under-constrained specification. The satisfaction process is planned by means of symbolic reasoning on the geometric level (geometric reasoning).

In [Emmerik, 90] constraints relate coordinate systems. Constraints between degrees of freedom (for example between the x- and y-coordinate because of a distance constraint) are evaluated after lower-order constraints (for example a constraint that uniquely determines the x-coordinate). This is a form of delayed satisfaction. Selection among alternative solutions to constraints is based on the goal to minimize the resulting disturbance.

Real general purpose constraint programming languages can be used in a wide range of applications, but are usually limited in their satisfaction power in each specific domain. Most constraint languages are therefore biased to a more specific type of constraints, frequently numerical constraints.

Bertrand [Leler, 88] is a rule-based system which uses augmented term rewriting. It has a form of abstract data types, allowing new types and constraints to be defined. It is a general purpose constraint language, but deals primarily with numeric constraints.

A general purpose constraint language based on logic programming is CLP($\mathbb{ID}$), which stands for Constraint Logic Programming language on domain $\mathbb{ID}$. For numeric constraints this yields CLP($\mathbb{IR}$) [Heintze et al., 87], over the domain of real numbers. This system does not use unification, but numeric deductions instead.

2.2 Motivation

Many systems detect over-constrained situations, but cannot handle under-constrained cases. Indeed, numerical methods cannot solve a set of under-constrained equations. A constraint logic programming system for solving numeric constraints, such as CLP($\mathbb{IR}$), returns a set of deduced equations that form the solution. When used in a geometric context however, this has no direct geometrical meaning.

The operational approach, and selection of alternatives based on minimal disturbance yield only one of all the solutions. Two potential problems are:

- The proposed solution is as was intended, but the user is not aware of the ambiguity of the specification. This may cause problems for post-processing.

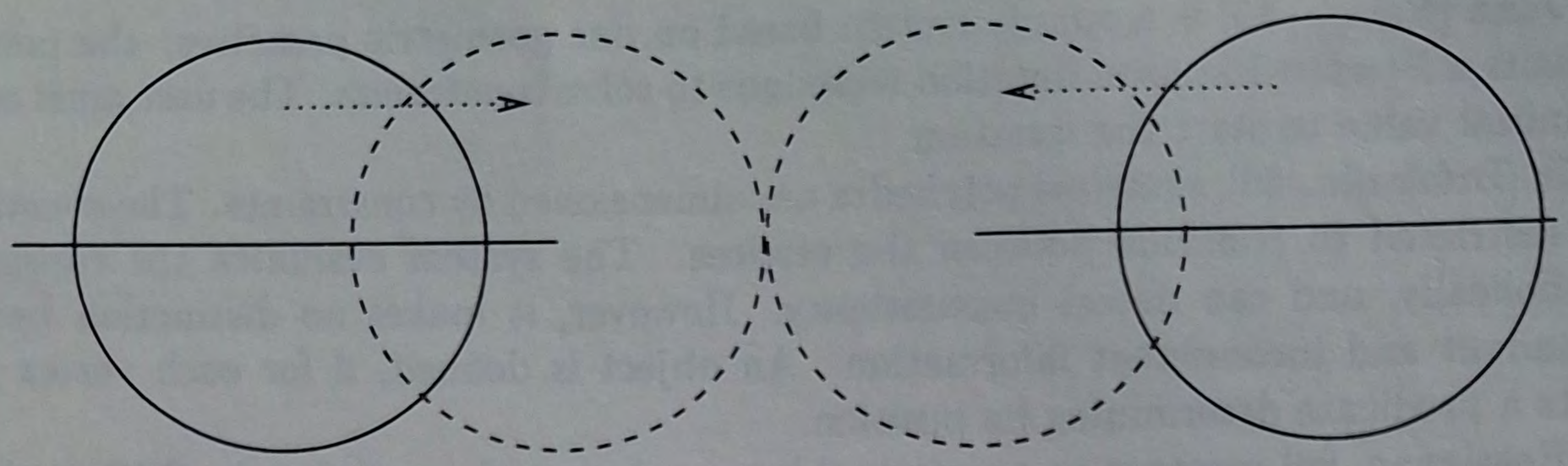


Figure 1: Simultaneous evaluation of mutually constrained circles.

- The solution is not as was intended, and the user must interfere because the system cannot propose alternative solutions.

We distinguish several types of alternative solutions:

1. A constraint can have several discrete solutions. For example a circle through two points and a fixed diameter, can have two positions (if the diameter is greater than the distance between the points).

2. A constraint can have a continuous range of solutions. For example the locus of a point having a fixed distance to a fixed point, is a circle.

Some systems mentioned above deal only with numeric constraints. Most graphics oriented constraint systems directly translate geometric constraints into a set of numeric constraints or equations. Often only linear equations can be solved, but recall that even a simple distance constraint results in a quadratic equation. In contrast, we want to solve constraints at a high level of geometric abstraction. This allows powerful reasoning which can avoid some of the problems that occur with numeric constraints.

Simultaneous evaluation of mutually constrained objects is a difficult problem. Look for example at figure 1. Each of the two circle centers must lie on a line segment; next, the circles are constrained to be tangent to each other on the outside. It is too complex for an operational technique to translate both circles along the line segment simultaneously to let them touch. Delayed evaluation is then preferred to relaxation, in order to deal with a geometrically meaningful representation as long as possible.

In this paper we focus on providing ranges of solutions and discrete alternative solutions in under-constrained situations, mutually restricting constraint operands, and on keeping the (intermediate) solutions in the geometric domain, thereby allowing geometric reasoning. In combination with geometric reasoning and delayed satisfaction this gives a more powerful problem solving capability than usually provided by constraint systems.

3 Quantum approach

The quantum approach is first introduced in [Veltkamp, 91] in the 2D domain. We first give an informal presentation; the ideas are formalized in the next section.

Our approach to geometric constraint satisfaction is based on reduction of both problem complexity and solution domain complexity by geometric simplification. Problem

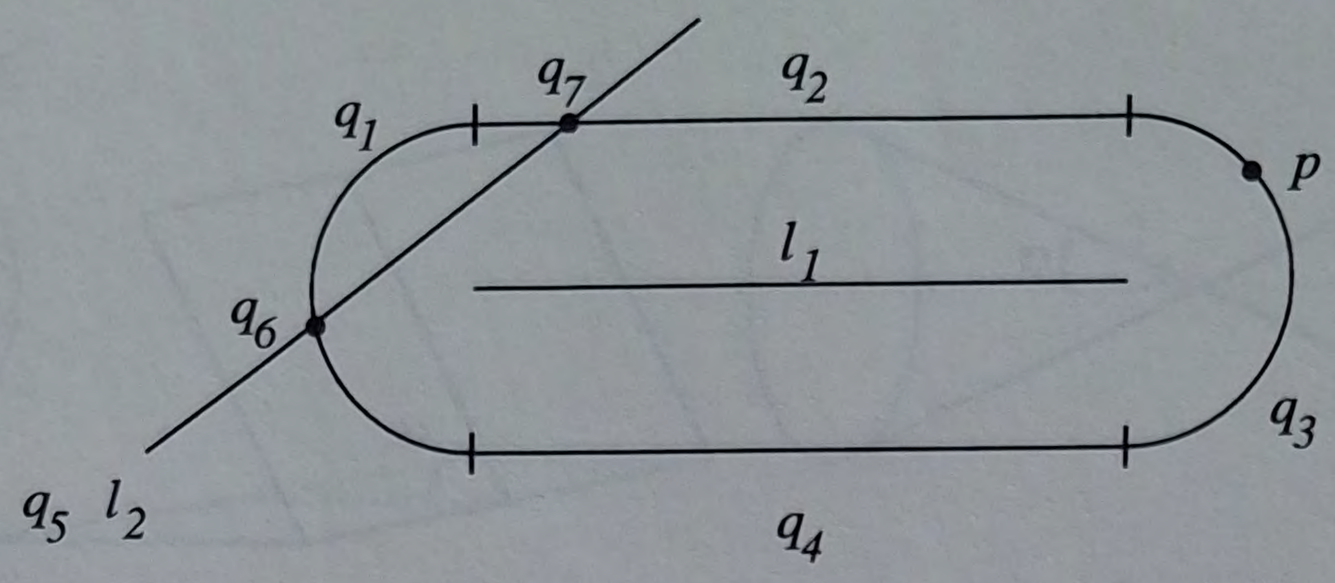


Figure 2: Quanta involved with the constraints distance(p, l_1, d) and on(p, l_2).

complexity reduction is achieved by expressing geometric primitives in terms of characteristic parts. For example, the position of a circle is reduced to the position of a point: the circle center. Constraint solving amounts to recording the solution of the constrained geometric primitives in terms of their characteristic parts. The unit of the solution domain is a 'quantum', which is itself a geometric primitive:

> A quantum is a geometric primitive describing a range of solutions with a uniform geometric characteristic.

The nature of a quantum is that it describes a sharply bounded quantity of some phenomenon, in our case the geometric properties of a solution set. As such, it is a more special notion than just a solution set, a region, or a quantity. Note that our geometric quanta are not indivisible, nor need they be disconnected. By means of these quanta, solutions are categorized into ranges or classes featuring the same geometric characteristics. In this way, subsequent processing of the solutions can exploit the geometric properties of the quanta.

For example the locus of a point p having a fixed distance to a line segment l_1, consists of the union of two line segments, and two half circles, see figure 2. The solution is split into the quanta $q_1, \ldots, q_4$, representing the two line segments and the two half circles. These quanta represent ranges of alternative solutions. Incremental specification of a geometric object with constraints will successively restrict its solution domain. If p is further constrained to lie on a line l_2 (another quantum), the solution is restricted to the intersection of l_2 with all the alternative solutions so far. Since calculating the intersection of a line and a circle differs from intersecting two lines, we take advantage of the subdivision into quanta $q_1, \ldots, q_4$.

Some simple constraints (with a fixed second operand) and their associated quanta are listed below:

	primitive	constraint	primitive	quanta
1	point	distance	point	circle
2	point	distance	line segment	2 line segments + 2 half circles
3	point	left/right of	line	half-plane
4	circle	tangent	line	2 lines
5	circle	tangent	circle	2 circles

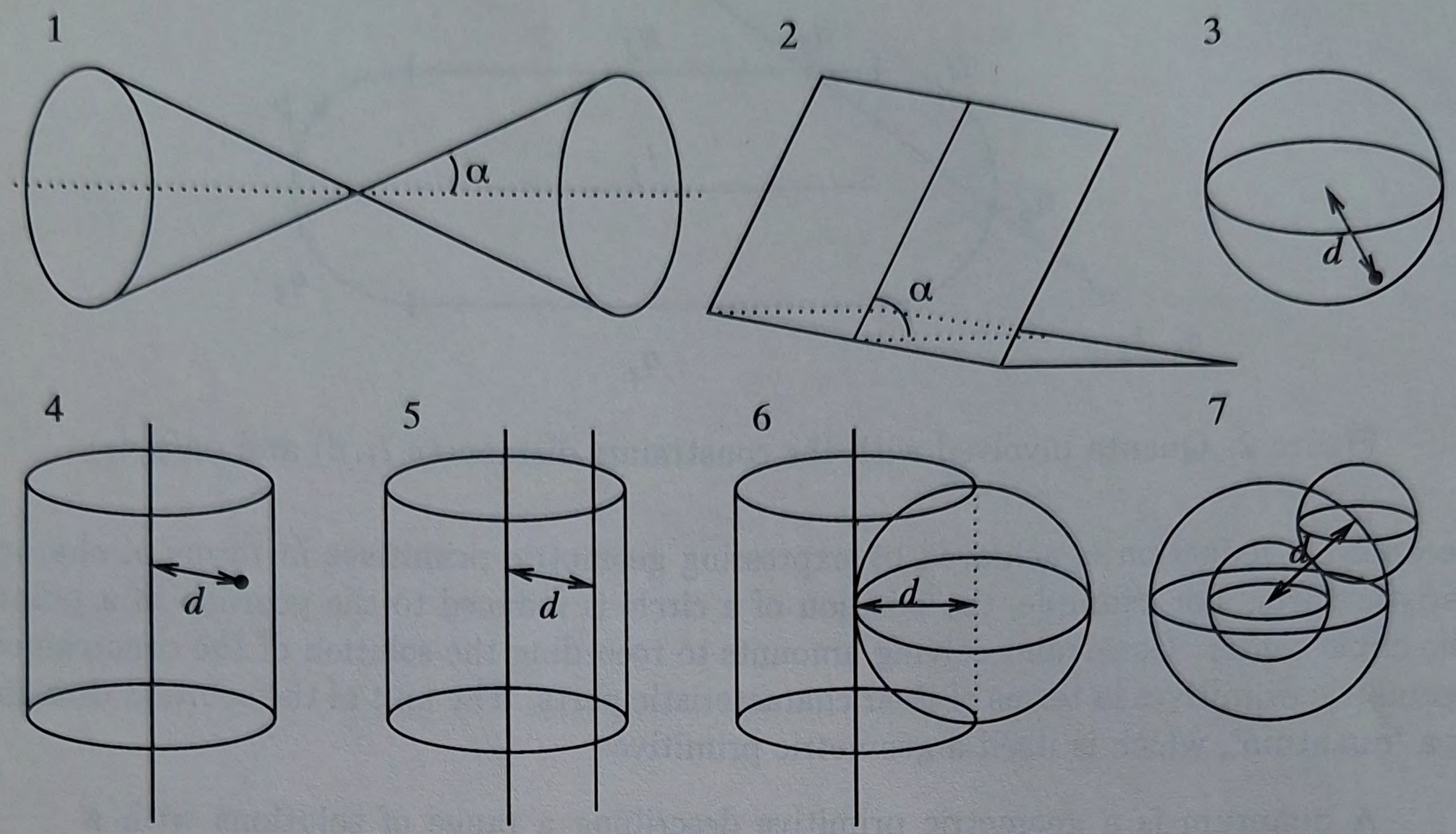


Figure 3: Quanta involved with 3D constraints.

The same principle can be used in 3-space. In general, however, a solution set may not be representable with a fixed set of geometric primitives, especially in 3-space. In that sense the set of constraints and geometric primitives should agree with each other. A set of useful constraints in 3D is listed below and illustrated in figure 3 (again the second operand is fixed):

	primitive	constraint	primitive	quanta
1	line	angle	line	cone
2	line	angle	plane	plane
3	point	distance	point	sphere
4	point	distance	line	cylinder
5	line	distance	line	cylinder
6	sphere	tangent	line	cylinder
7	sphere	tangent	sphere	2 spheres

In section 5, we give an example using 3D constraints.

4 Constraint management

A set of constraints and geometric primitives can be represented as a network, such that the constraint node is connected to its variables. The network can be disconnected, in which case the variables in one connected component and those in the other connected components, are not related by constraints. Constraints are represented in rectangular boxes, geometric primitives in circular nodes. A quantum for a variable v, consistent with constraint $c(\ldots, v, \ldots)$, can be considered as a label on an edge between (v) and [c].

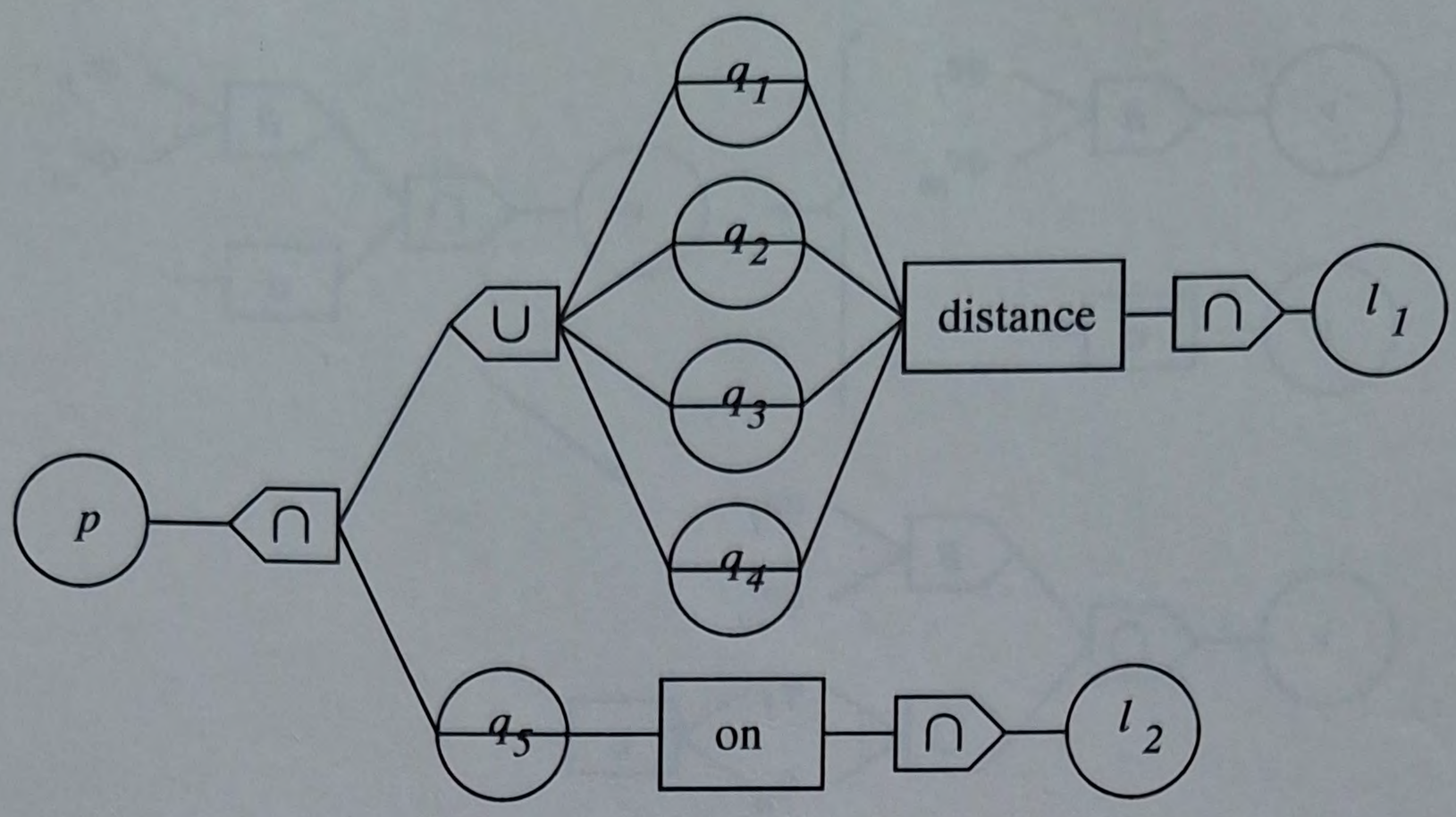


Figure 4: Intermediate network corresponding to figure 2.

Being a geometric primitive, a quantum is represented as a circle, but with the edge that it labels drawn through.

In order to represent unions and intersections of quanta, the special Boolean set nodes ∪ and ∩ are used. These nodes are operators whose operands are the quantum labels of edges coming in at the flat sides of the nodes, and whose results are the quantum labels of the edges going out at the sharp end of the node. If there are no quantum labels on the edges going out, these quanta are not (yet) computed. Similarly, if there are no quantum labels at the incoming edges, they are not yet generated or computed. In both cases, the logical relation between these forthcoming quanta is already incorporated in the network.

Between a constraint and an associated variable is a sequence of (zero or more) quanta interleaved with Boolean set nodes, on a path of edges. In the direction from the constraint to the variable, each quantum includes the next one. Figure 4 shows a network that corresponds to the example of figure 2. The network shown is an intermediate result in the satisfaction process.

Our model consists of a set of constraints C, a set of constraint variables V, a set of quanta Q, a quantum generating function G, and a function Tolerate(c) that satisfies a constraint c and performs propagation. The constraint variables and quanta are geometric primitives.

A constraint is a relation between a number of variables from V. This can be any number, depending on the type of constraint. Constraints are multi-directional in the sense that each of the involved variables constrains the others. In contrast to many other systems, many variables may be undetermined, that is, they restrict each other simultaneously.

The quantum generating function $G : (c, v_i) \mapsto (q_1, \ldots, q_n)$ generates the set of quanta for v_i that are consistent with the constraint $c(v_1, \ldots, v_k)$ and all the variables v_j, $j = 1, \ldots, k$, $i \neq j$, with their current quanta, regardless of the current quanta of v_i. Note that not all combinations of types of constraints and variables are geometrically meaningful; G

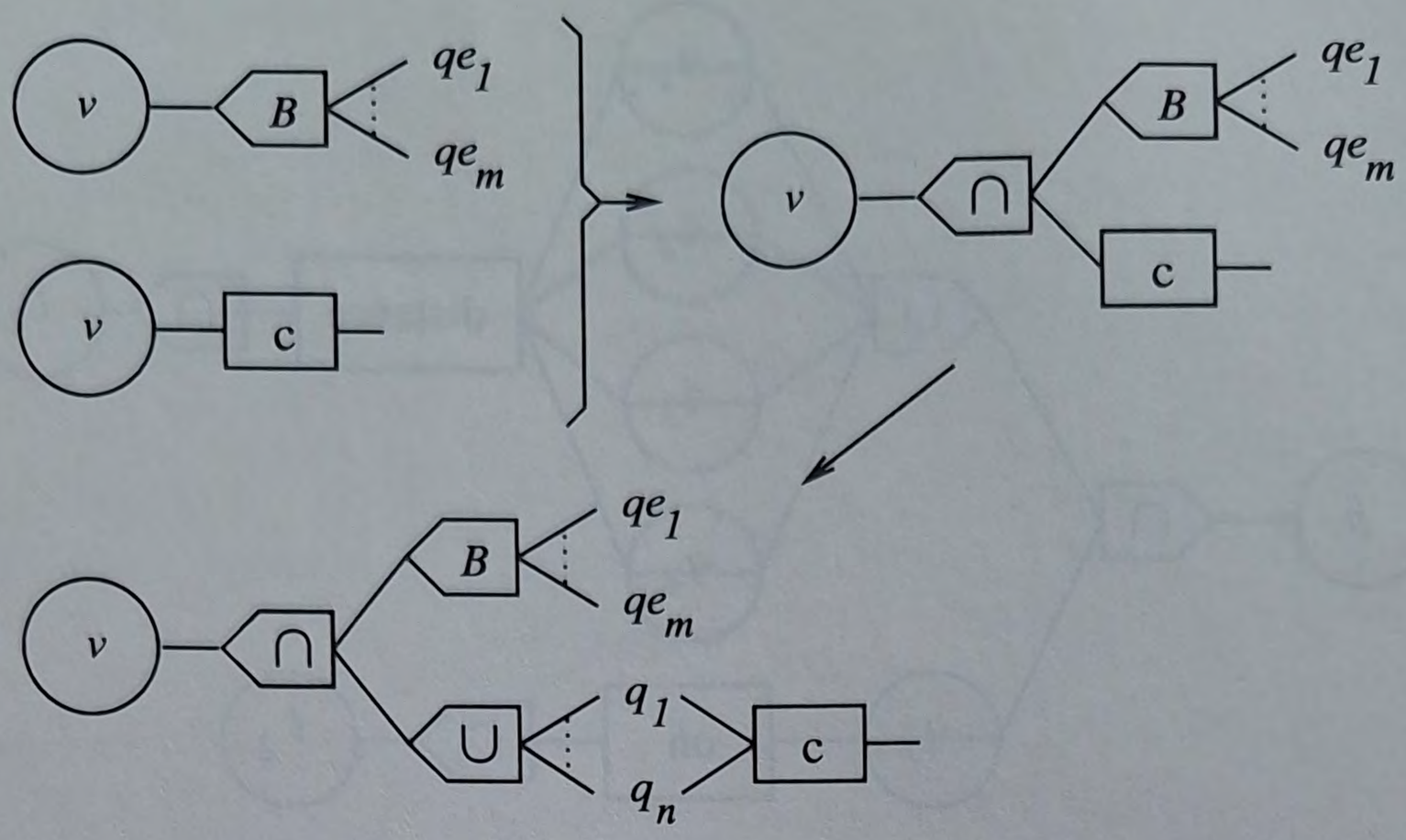


Figure 5: Addition of $c(v, \ldots)$ to a network, and generation of quanta.

is defined only for a finite number of combinations. The generation of the quanta involves geometric reasoning, taking into account the nature of the constraint and the current quanta of the variables.

When a new constraint is added to the network, the quantum generating function is applied to each of its variable. For a single variable, the union of the resulting quanta, $\cup(q_1, \ldots, q_n)$, is the solution to that one constraint, which must be intersected with the variable's current quanta. In this way we get an expression of quanta, which is the solution set under the constraints so far. In general, this quantum expression is of the form $B(qe_1, \ldots, qe_m)$, where B is one of the Boolean set operators union or intersection, and the qe_i are quantum expressions. So, after addition of a new constraint, generating new quanta $q_1, \ldots, q_n$, the solution set becomes $\cap(B(qe_1, \ldots, qe_m), \cup(q_1, \ldots, q_n))$. It may sometimes be difficult to compute the intersection; in that case we leave the intersection unevaluated. That is why a qe_j can be an *intersection* of quanta or other qe-s: $qe_j = \cap(qe'_1, \ldots, qe'_k)$.

Figure 5 illustrates the addition of a constraint to a network and the subsequent generation of quanta. Only the part of the network in the neighborhood of variable v is depicted.

When a new constraint is specified it must be added to the network, and the resulting changes must be propagated to related constraints. This may cause new changes which in turn must be propagated, and so on. This propagation is repeated until the network is stable. In principle the propagation need not terminate, since an infinite loop can occur. A loop can be easily detected, but it is generally not known whether the loop is finite, or a prefix of an infinite loop. The constraints in a loop can be satisfied with some other method, for example relaxation; the resulting quanta must be propagated again. The full propagation algorithm and an analysis of its characteristics is presented in [Veltkamp and Arbab, 91].

An over-constrained situation is detected when a variable has only empty quanta. Otherwise the variable tolerates the new constraint. In general, the current quanta form the solution of the present set of constraints, and these need not specify a single position.

This happens in a (possibly temporary) under-constrained specification.

A quantum need not be fixed. It is for example possible to specify an assembly of geometric primitives by constraints, relative to an object that is fully specified only afterwards. It is therefore necessary to be able to process incomplete information. To keep intermediate solutions simple we want to evaluate them as far as possible, and also use latent geometric information. Dealing with incomplete, and deriving latent information are typical in geometric reasoning.

Operations on a variable, such as translate, scale, mirror, and project, can be handled by performing proper corresponding operations on its quanta. These new quanta must then be propagated through the network.

For interactive CAD purposes, a constraint must also be removable. In our system, deletion of a constraint from a network involves removal of the sequence of quanta running towards the constraint's variables. Sequences from other constraints to the same variables must be 'repaired' by intersecting the proper quanta. The resulting new quanta must then be propagated through the new network.

5 Case studies

In this section we consider two cases that are simple in their specification, but non-trivial in their solution. The first case is a two-dimensional example, the second one a three-dimensional application. To simplify the network drawings, only partial networks in the neighborhood of a single variable are shown.

5.1 Planar linkage

Consider the four-bar linkage of figure 6 (top left), involving points A, B, C, D, M. The points A and B are fixed and a distance d_1 apart. The constraints on C, D, and M are the following:

constraint 1: the distance between A and C is d_1;
constraint 2: the distance between B and D is d_2;
constraint 3: the distance between C and D is also d_2;
constraint 4: M is the midpoint of C and D;
constraint 5: the line through A and C has a fixed slope.

These constraints are specified in the given order, and we are interested in the position of C, D, and M.

We can solve this problem algebraically by computing the locus of C, D, and M each time a constraint is specified. As an illustration, set A to $(0,0)$ and B to $(0,d_1)$, without loss of generality. Then, the locus of $M=(x,y)$ just after specifying constraint 4 is given by:

$$16(x^4+y^4)+32x^2y^2-16d_1(x^3+xy^2)-(12d_1^2+4d_2^2)(x^2+y^2)+(8d_1^3-4d_1d_2^2)x=d_1^2d_2^2-4d_1^4$$

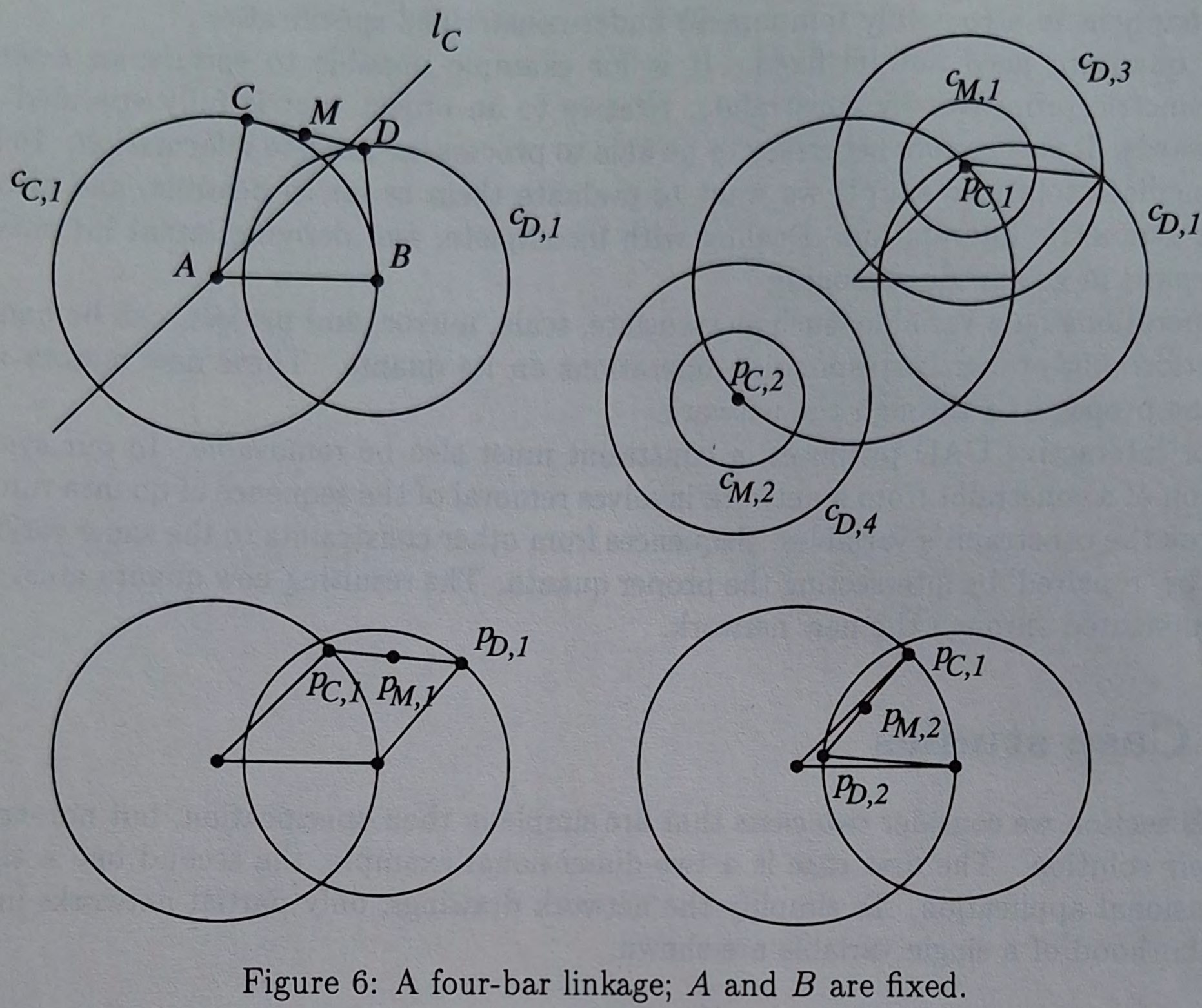


Figure 6: A four-bar linkage; A and B are fixed.

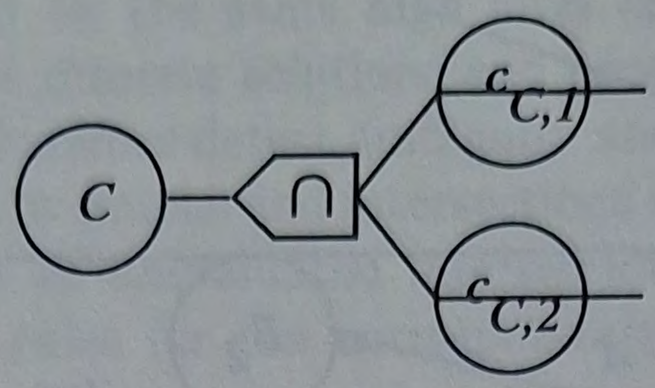
The solution to these equations can be derived by an algebraic manipulation system (see [Chou and Gao, 89]), but it is of little use here, because it does not interact in a simple way with the subsequent constraint.

Alternatively, our constraint management system comes up with the following quanta:

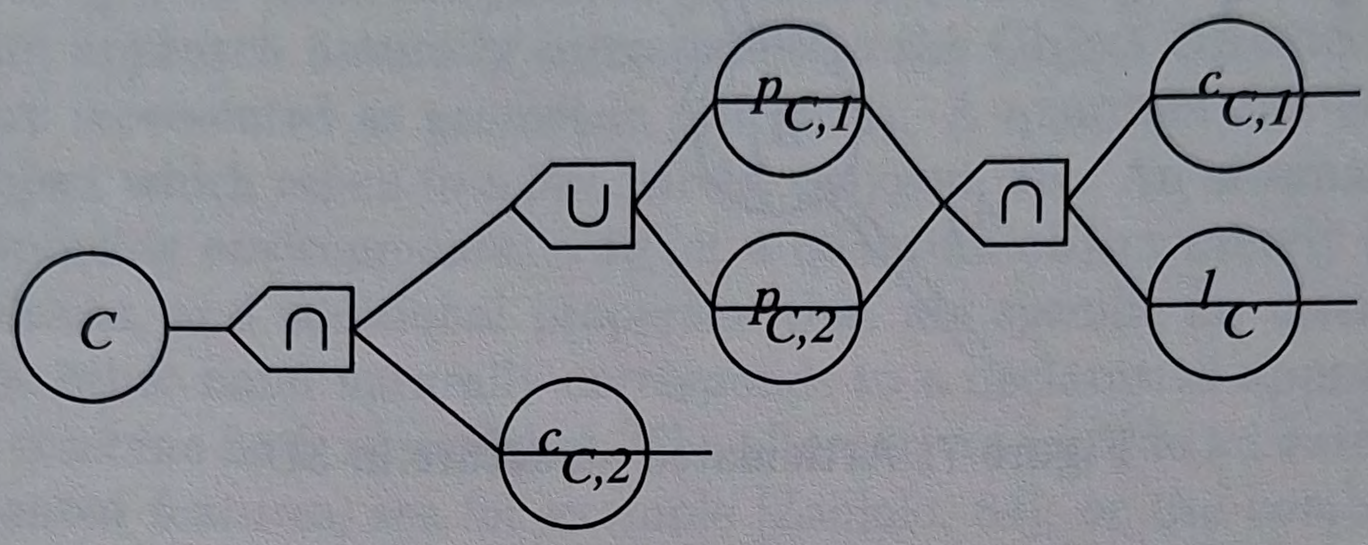
by constraint 1: C is restricted to a quantum that is the circle $c_{C,1}$ centered at A and having radius d_1;
by constraint 2: D is restricted to the circle $c_{D,1}$ centered at B with radius d_2;
by constraint 3: for C: a circle $c_{C,2}$ with radius d_2 which has no fixed position, but is centered at D; similarly for D: a circle $c_{D,2}$ concentric to C, having radius d_2;
by constraint 4: because M is the midpoint of two points whose positions are unknown, no quantum is generated yet for M;
by constraint 5: since the line through A and C must have a fixed slope, another quantum is associated to C: line l_C.

The quanta $c_{C,1}$, $c_{D,1}$, and l_C are shown in figure 6, top left.

The quanta resulting from constraint 3, $c_{C,2}$ and $c_{D,2}$, are not fixed but can move around. Consequently, it is difficult to intersect these quanta with the other quanta $c_{C,1}$ and $c_{D,1}$. The intersection remains unevaluated, but is represented in the network; for example for C:

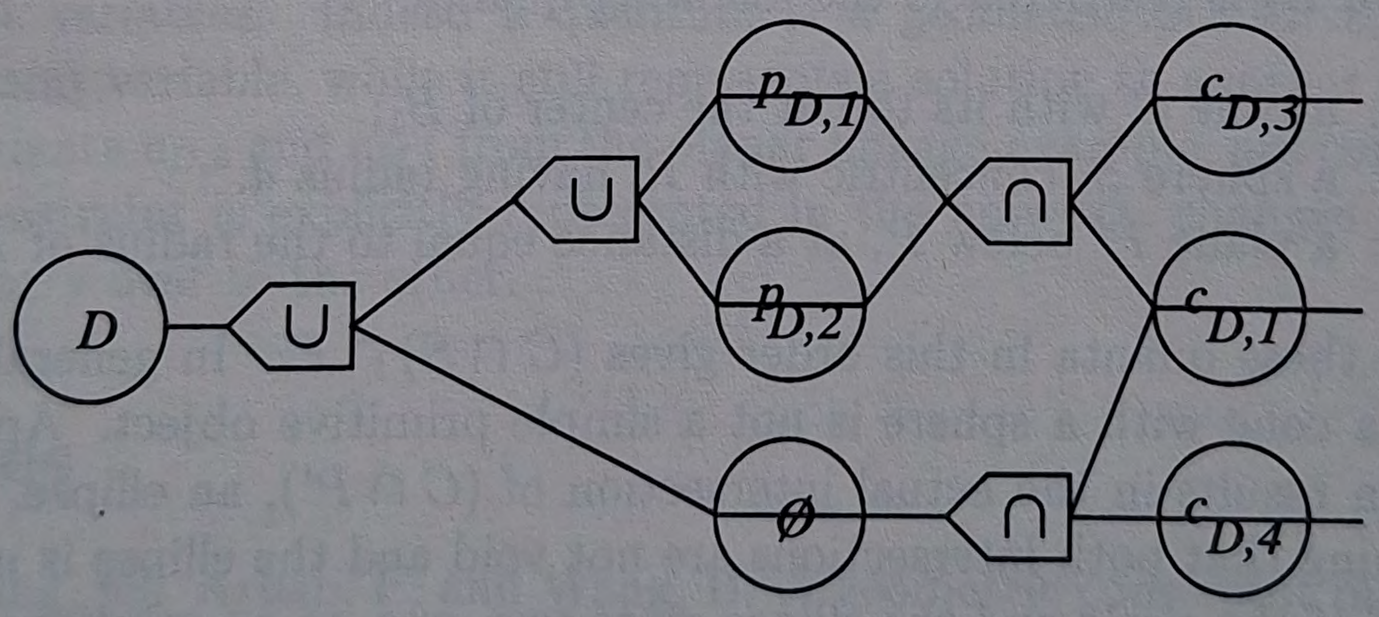


Quantum l_C must be intersected with the current quanta of C. Again, the intersection with $c_{C,2}$ remains unevaluated. The intersection of $c_{C,1}$ and l_C gives two points $p_{C,1}$ and $p_{C,2}$:



This intermediate result must be propagated to objects related to C. Because C is now restricted to $p_{C,1}$ and $p_{C,2}$, the movable quantum $c_{D,2}$ of D is now replaced by two circles $c_{D,3}$ and $c_{D,4}$ centered at $p_{C,1}$ and $p_{C,2}$. These circles can be seen as two specific instances of $c_{D,2}$. Also, since M is the midpoint of C and D, it is now restricted to two circles $c_{M,1}$ and $c_{M,2}$, concentric with $c_{D,3}$ and $c_{D,4}$ but with half their radius. The new quanta are depicted in figure 6, top right.

Quanta $c_{D,3}$ and $c_{D,4}$ are intersected with $c_{D,1}$: $\cap(c_{D,3}, c_{D,1}) = \cup(p_{D,1}, p_{D,2})$ and $\cap(c_{D,4}, c_{D,1}) = \emptyset$. The corresponding part of the network associated with D is as follows:



Again this intermediate result must be propagated. Since D is now restricted to $p_{D,1}$ and $p_{D,2}$, the quanta $p_{C,2}$ and $c_{M,2}$ are deleted, and $c_{M,1}$ is replaced by $p_{M,1}$ and $p_{M,2}$. The two possible solution configurations are shown in the lower row of figure 6.

5.2 Articulated structure in space

The application in this subsection is illustrated in figure 7. The 3D scene in this example contains a plate P, a spherical source of radiation R, and an articulated structure supporting P, consisting of two arms A_1 and A_2, having balls B_1 and B_2 respectively at

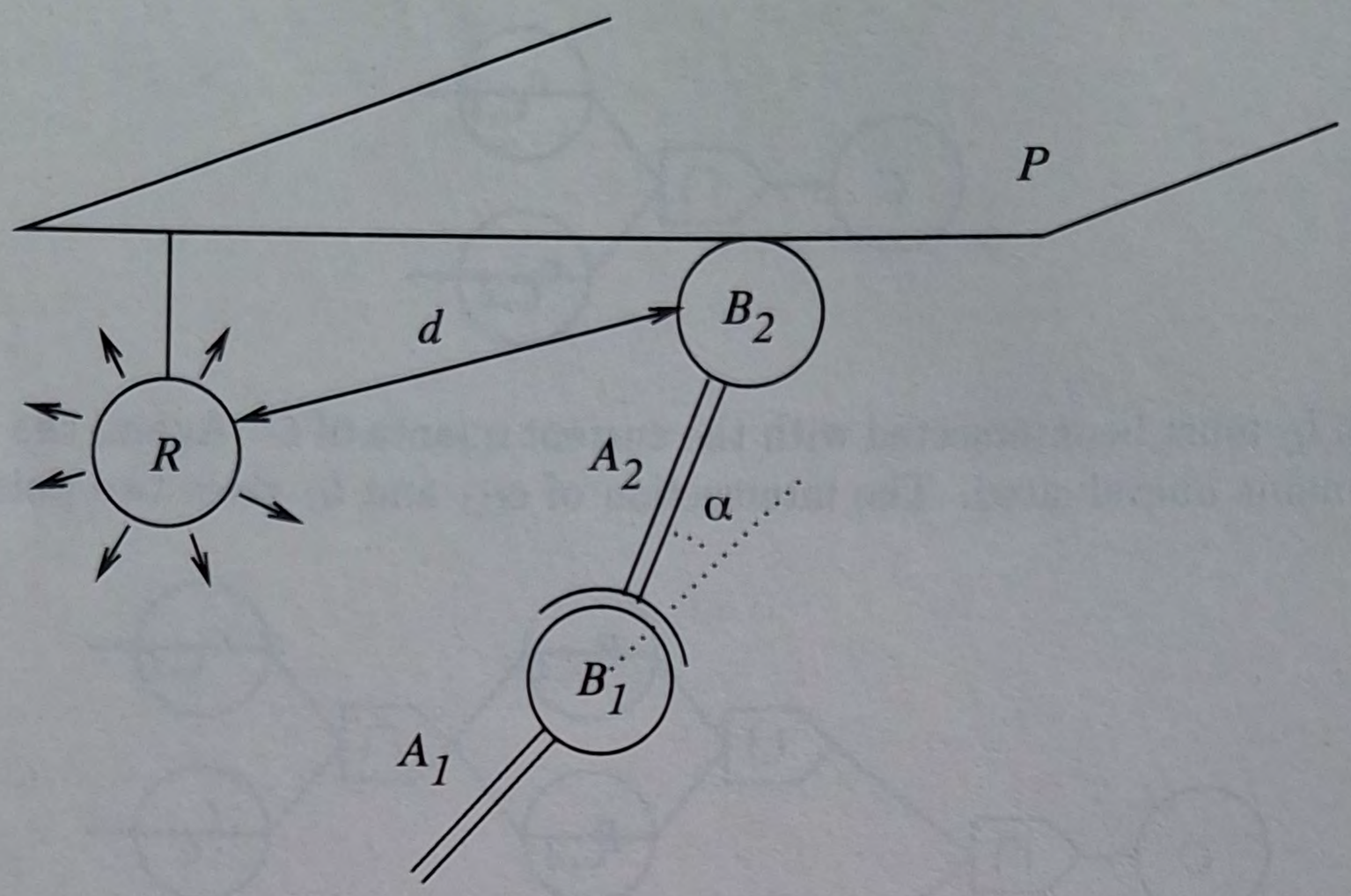


Figure 7: Articulated structure in 3D.

their endings. P, R, A_1 and B_1 are totally fixed. The constraints on A_2 and B_2 are the following:

constraint 1: A_2 must have a fixed angle α with respect to A_1;
constraint 2: B_2 must be at a fixed distance d from R;
constraint 3: B_2 must be tangent to P at the bottom side.

The constraints are incrementally specified in this order. We want to know the allowable positions of ball B_2.

The center of B_2 is restricted to the following quanta:

by constraint 1: a cone C with its top at the center of B_1;
by constraint 2: a sphere S, concentric with R, having radius d.
by constraint 3: a plane P' below P, at a distance equal to the radius of B_2;

Intersecting these quanta in this order gives $(C \cap S) \cap P'$. In general, however, the intersection of a cone with a sphere is not a simple primitive object. Applying delayed evaluation again results in the actual intersection of $(C \cap P')$, an ellipse, with $(S \cap P')$, a circle. Assuming that both intersections are not void and the ellipse is non-degenerate, the intersection of the circle and the ellipse gives two, one, or no solutions, depending on the actual positions and orientations of R, P, and A_1.

6 Concluding remarks

We presented a constraint satisfaction system based on the quantum approach. This approach is suitable for interactive design because it can handle incomplete specifications and alternative solutions, and performs satisfaction locally and incrementally. The system keeps the intermediate solutions in the geometric domain, so that new geometric

constraints can be interpreted on the same high level of abstraction, allowing powerful reasoning. Both alternative discrete solutions and continuous ranges of solutions are determined. The system can therefore detect ambiguity and present alternative solutions.

The solution is represented by unions and intersections of quanta. Delayed satisfaction amounts to the representation of unevaluated intersections, which may be resolved by subsequent constraints. The rules for the corresponding network transformations are given in [Veltkamp and Arbab, 91].

The quantum approach can also be applied to pure numeric constraint problems. The solution is then derived using ranges for parameter values, instead of geometric quanta. Algebraic reasoning then takes the place of geometric reasoning.

The quantum approach naturally corresponds to the Object Oriented Paradigm, because quanta are represented as geometric primitives. A quantum can be designed as a polymorphic object which refers to a line, circle, polygon, etc. An alternative in multiple inheritance supporting environments, is to let a quantum object inherit properties from geometric primitives, and additional properties that are specific for quanta. Geometric reasoning on the other hand naturally corresponds to a declarative approach. There are several ways to combine both approaches. The alternatives include an extension of Prolog with object oriented features, see for example [Zaniolo, 84], or the combination objects and relational logic as in Oar (Objects and reasoning) [Arbab and Wang, 89]. Object oriented concepts can also be combined with a functional language, see for example CLOS (Common Lisp Object System) [Moon, 89]. All these languages specify objects, hierarchies, and methods in a declarative way.

More research is needed to further exploit the capabilities of geometric reasoning. In particular implied constraints, which can be derived from other constraints and variables, can be used by the satisfaction system to solve complex constraints more directly. Since an implied constraint is derived, it does not introduce new facts, but it can simplify constraint satisfaction. Implied constraints can operate on quanta as well as already used constraint variables. Indeed a quantum is a geometric primitive, and can thus act as a constraint variable, while it still represents a solution to another constraint. For example, the quanta $c_{D,3}$ and $c_{M,1}$ from the planar linkage example, are concentric. When this implied constraint is explicitly represented in the network, changes to one of them mechanically carry over to the other.

References

[Arbab and Wang, 89] Arbab, F. and Wang, B. A geometric constraint management system in Oar. In ten Hagen, P. J. W. and Veerkamp, P. (editors), *Intelligent CAD Systems III – Practical Experience and Evaluation (Proceedings 3rd Eurographics Workshop on Intelligent CAD Systems, 1989)*. Springer-Verlag, 1989, 231 – 252.

[Arbab and Wing, 85] Arbab, F. and Wing, J. M. Geometric reasoning: A new paradigm for processing geometric information. In Yoshikawa, H. and Warman, E. A. (editors), *Design Theory for CAD (Proceedings of the IFIP W. G. 5.2 Working Conference, 1985)*. Elsevier Science Publishers, 1985, 145 – 165.

[Badler and Kamran, 87] Badler, N. I. and Kamran, H. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics & Applications*, 7(6), 1987, (special issue on articulated figure animation,) 28 – 38.

[Borning, 81] Borning, A. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4), 1981, 353 – 387.

[Brüderlin, 86] Brüderlin, B. Constructing three-dimensional geometric objects defined by constraints. In [Crow and Pizer, 86], 111 – 129.

[Chou and Gao, 89] Chou, S.-C. and Gao, X.-S. A collection of 120 computer solved geometry problems in mechanical formula derivation. Technical Report TR-89-22, University of Texas, Department of Computer Science, 1989.

[Crow and Pizer, 86] Crow, F. and Pizer, S. M. (editors). *Proceedings of the 1986 ACM Workshop on Interactive 3D Graphics*. ACM Press, 1986.

[Emmerik, 90] Emmerik, M. J. G. M. v. A system for interactive graphical modeling with 3D constraints. In *Preliminary proceedings CGI '90, Singapore*, 1990.

[Heintze et al., 87] Heintze, N. C., Michaylov, S., and Stuckey, P. J. CLP(ℝ) and some problems in electrical engineering. In Lassez, J.-L. (editor), *Proceedings of the 4th International Conference on Logic Programming*. MIT Press, 1987.

[Helander, 88] Helander, M. (editor). *Handbook of Human–Computer Interaction*. North-Holland, 1988.

[Leler, 88] Leler, W. *Constraint Programming Languages, Their Specification and Generation*. Addison-Wesley, 1988.

[Lin et al., 81] Lin, V. C., Gossard, D. C., and Light, R. A. Variational geometry in computer-aided design. *Proceedings SIGGRAPH '81, Computer Graphics*, 15(3), 1981, 171 – 177.

[Mäntylä, 88] Mäntylä, M. *An Introduction to Solid Modeling*. Computer Science Press, 1988.

[Moon, 89] Moon, D. A. The common lisp object-oriented programming language. In Kim, W. and Lochovsky, F. (editors), *Object-Oriented Concepts, Databases, and Applications*. ACM Press/Addison Wesley, 1989.

[Nelson, 85] Nelson, G. Juno, a constraint-based graphics system. *Proceedings SIGGRAPH '85, Computer Graphics*, 19(3), 1985, 235 – 243.

[Pratt, 87] Pratt, M. J. *Form Features and their Applications in Solid Modeling, Tutorial SIGGRAPH '87*. ACM, 1987.

[Rossignac, 86] Rossignac, J. R. Constraints in constructive solid geometry. In [Crow and Pizer, 86], 93 – 110.

[Steele Jr. and Sussman, 79] Steele Jr., G. L. and Sussman, G. J. CONSTRAINTS. *APL Quote Quad*, 9(4–Part 1), 1979, 208 – 225.

[Sutherland, 63] Sutherland, I. Sketchpad: A man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference (IFIPS).* 1963, 329 – 345.

[Szekely and Meyers, 88] Szekely, P. and Meyers, B. A user-interface toolkit based on graphical objects and constraints. In *Proceedings of the 1988 ACM Conference on Object-Oriented Programming Systems, Languages, and Applications.* ACM, 1988, 36 – 45.

[Veltkamp, 91] Veltkamp, R. C. A quantum approach to geometric constraint satisfaction. In Laffra, C. and Blake, E. (editors), *Advances in Object Oriented Graphics II (proceedings of the 2nd Eurographics Workshop on Object Oriented Graphics, 1991).* To be published by Springer-Verlag.

[Veltkamp and Arbab, 91] Veltkamp, R. C. and Arbab, F. Geometric constraint propagation with quantum labels. In *Preliminary proceedings of the Eurographics Workshop on Computer Graphics and Mathematics*, Genova, Italy, October 1991.

[Zaniolo, 84] Zaniolo, C. Object-oriented programming in prolog. In *Proceedings of the International Symposium on Logic Programming.* IEEE, 1984, 265 – 270.