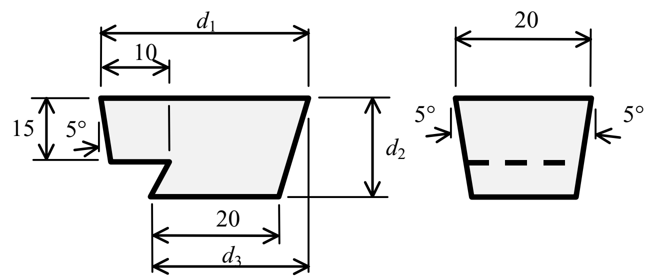
Algebraic & Geometric Relationships

Algebraic & Geometric Relationships

- In the last lecture we started talking about "Knowledge Representation." We specifically looked at the representation of <u>product</u> information (and more specifically, geometric model information.)
- This lecture will introduce **relationships** that can be specified (or queried) between geometric entities in information or knowledge representations.

Algebraic & Geometric Relationships



$d_3 = 0.7\, d_1$

$30 \le d_1 \le 40$
$20 \le d_2 \le 25$

Basis of Relationships

- Algebraic and Geometric relationships can be based on:
  - I. Equality expressions
    - Equations, geometric constraints
  - II. In-equality expressions
    - Inequalities, solution selectors
  - III. Objective-function expressions

I. Equality-based Relationships

- In the first CAD systems geometric entities, (e.g., points, curves, primitive shapes), were created in a given position and joined with the rest of the model.
- To make changes to the geometry required manually moving these entities or deleting and recreating new ones.
- Parametric modeling systems introduced the idea of "dimension-driven geometry."

Parametric Modeling

- Parameters come from:
  - dimensions in constrained 2D sketches
  - dimensions directly on constrained 3D geometry (new)
  - modeling operation parameters
  - variables in user-equations
- Entire part can be modified by changing one parameter!
- Two types of systems:
  - parametric
  - variational

“Variational” vs. “Parametric”

- CAD systems can be classified as variational or parametric according to how they handle equation solving.
- Variational systems such as I-DEAS allow you to type equations in any form. E.g.:

  $x + 3 y = 5; x - y = 8$
- Parametric systems such as Unigraphics require you to re-arrange the equations to solve for one variable in terms of other previously solved variables. E.g.:

  $x = 29/4; y = x - 8$

“Variational” vs. “Parametric”

Variational Shaft Design

Power_HP	= 8
Power_kW	= Power_HP x 0.75
RPM	= 6000
rot_per_sec	= RPM / 60
Torque_Nm	= Power_kW / rot_per_sec x 1000
t_allowable_Pa	= 200 x10^6 / 2
r_m	= J x t_allowable_Pa / Torque_Nm
J	= pi x r_m^4 / 2
D_mm	= 2 x r x 1000

“Variational” vs. “Parametric”

Parametric Shaft Design

Power_HP = 8
Power_kW = Power_HP x 0.75
RPM = 6000
rot_per_sec = RPM / 60
Torque_Nm = Power_kW / rot_per_sec x 1000
T_allowable_Pa = 200 x10^6 / 2
r_m = (2 x Torque_Nm / pi / t_allowablePa) $^{1/3}$
D_mm = 2 x r_m x 1000

Geometric Constraints

- In both parametric and variational systems there are three types of geometric (equality-based) relations:
  - Dimensional Constraints
    - Distance, Angle, Radius
  - Geometric (Logical) Constraints
    - Parallel, Perpendicular, Tangent
  - Transformation/Mapping-based
    - Rigid set, 2D/3D mappings

Deriving a Standard Set of Geometric Constraints (Bettig & Shah, 2001)

A standard geometric constraint representation would be useful for:

- **CAD data exchange**
- **geometric solver API's**
- **CAD user interfaces**
- **communication between designers**

Deriving a Standard Set of Geometric Constraints

Objectives – a representation of declarative geometric constraints that is:

- **complete**
- **minimal**
- **at user level of abstraction**

Derivation Procedure

1. Determine morphology based on literature review
2. Create taxonomy
3. Examine combinations of entity/ constraint situations

Taxonomy Outline

Geometric Constraint

Distance and Angle-based vs. Transformation-based

No. of entities related by constraint: 2/1/>2

Physical or resolved geometry

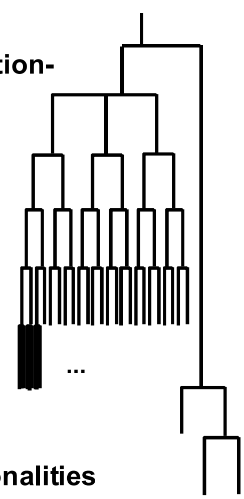
Distance or Angle

One point or all points of curve/surface

Dimension is regular, limiting case or unknown

Set or Mapping

Same dimensionality or between dimensionalities



Resulting Binary Constraints

<u>Binary Distance/Angle-based constraints:</u>

- DISTANCE
- PARALLEL_OFFSET
- COINCIDENT
- INCIDENT
- TANGENT*

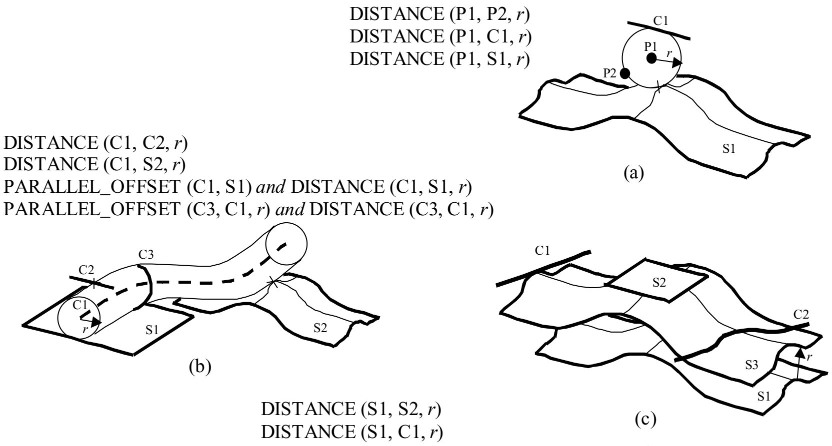
- ANGLE
- SAME/OPPOSITE DIRECTION
- PERPENDICULAR*

*** redundant constraint type**

Visualizing Distance Constraints

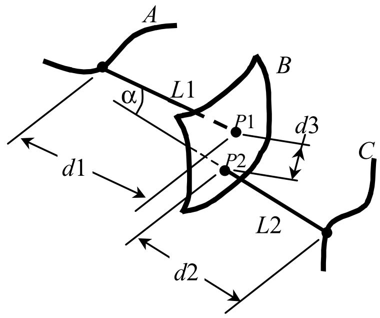
DISTANCE (P1, P2, *r*)
DISTANCE (P1, C1, *r*)
DISTANCE (P1, S1, *r*)

DISTANCE (C1, C2, *r*)
DISTANCE (C1, S2, *r*)
PARALLEL_OFFSET (C1, S1) *and* DISTANCE (C1, S1, *r*)
PARALLEL_OFFSET (C3, C1, *r*) *and* DISTANCE (C3, C1, *r*)

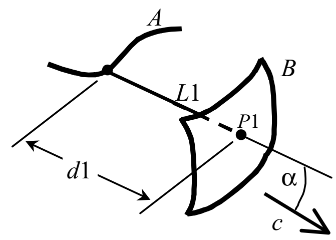


DISTANCE (S1, S2, *r*)
DISTANCE (S1, C1, *r*)
PARALLEL_OFFSET_DISTANCE (C2, S1) *and* DISTANCE (C2, S1, *r*)
PARALLEL_OFFSET_DISTANCE (S3, S1) *and* DISTANCE (S3, S1, *r*)

Procedures for examining n-ary constraint combinations



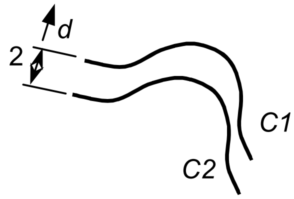
(a) a Procedure 1 combination



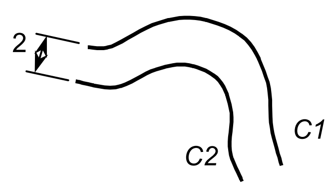
(b) a Procedure 2 combination

N-ary Geometric Constraints

- TRANSLATED



TRANSLATED(*C1*, *C2*, *d*, 2)



OFFSET_CURVE(*C1*, *C2*)
DISTANCE(C1, C2, 2)

Translated vs. parallel offset

N-ary Geometric Constraints

- ROTATED
- SYMMETRIC
- TANGENT_AT_POINT
- TANGENT_AT_CURVE
- CURVE_POINT_DIRECTION
- SURFACE_POINT_NORMAL

Constraints on Resolved Geometry

- COINCIDENT_LOCATING_POINT
- LOCATING_POINT_DISTANCE/ANGLE*
- COINCIDENT_AXIS
- AXIS_DISTANCE/ANGLE *
- COINCIDENT_SEMI_AXIS
- SEMI_AXIS_DISTANCE/ANGLE*

*** redundant constraint type**

Unary Constraints

- RADIUS
- SEMI_AXIS1/2_DIMENSION
- FOCAL_DISTANCE
- SEMI_ANGLE
- MAJOR/MINOR_RADIUS

- FIXED
- FIXED_DIRECTION

Rigid Set Constraints

Sets (entities)

- RIGID_SET (Euclidean)
- SCALABLE_SET
- 3_DIRECTION_SCALABLE_SET
- SKEWABLE_SET

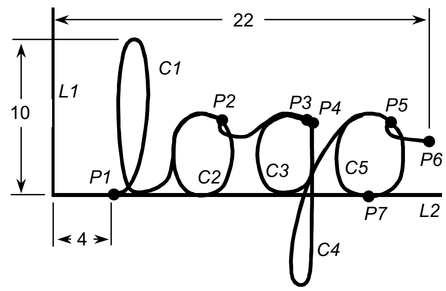
Constraints

- IN_SET constraint

Rigid Set Constraints

Two Direction Scalable (2D) Set: *S1*
IN_SET(*S1*, *P1*) ... IN_SET(*S1*, *P6*)
IN_SET(*S1*, *C1*) ... IN_SET(*S1*, *C5*)

INCIDENT(*P1*, *L2*)
TANGENT_AT_POINT(*C5*, *L2*, *P7*)
DISTANCE(*P1*, *L1*, 4)
DISTANCE(*P6*, *L1*, 22)
DISTANCE(*C1*, *L2*, 10)



SCALABLE_SET (2D)

Mapping Constraints

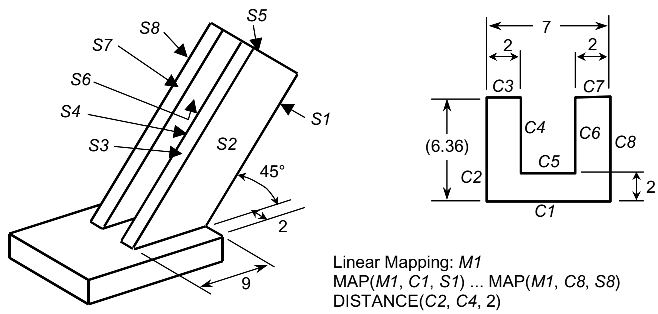
<u>Mapping (entities)</u>

- **COPY_MAPPING** **(3D to 3D)**
- **WIREFRAME_MAPPING** **(2D to 3D)**
- **LINEAR_MAPPING** **(2D to 3D)**
- **ROTATIONAL_MAPPING** **(2D to 3D)**
- **HELICAL_MAPPING** **(2D to 3D)**

<u>Constraints</u>

- **MAP constraint**

Mapping Constraints



Linear Mapping: *M1*
MAP(*M1*, *C1*, *S1*) ... MAP(*M1*, *C8*, *S8*)
DISTANCE(*C2*, *C4*, 2)
DISTANCE(*C6*, *C8*, 2)
DISTANCE(*C2*, *C8*, 7)
DISTANCE(*C1*, *C5*, 2)
COINCIDENT(*C3*, *C7*)
RIGHT_ANGLE(*C1*, *C2*)

LINEAR_MAPPING **(2D to 3D)**

Benefits of this Scheme

Present taxomony is more comprehensive, small number of constraint types:

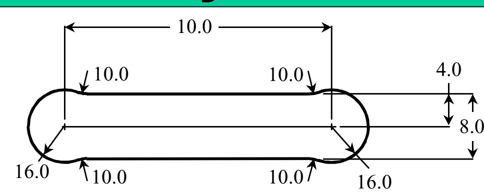
	Dist. & Angle		All	
	#sit'n	#con	#sit'n	#con
Present taxonomy	40	12	77	35
Parametrics Framework	24	15	27	18
ENGEN	22	15	26	20
D-Cubed 3D DCM	24	9	26	11

II. Inequality-Expressions

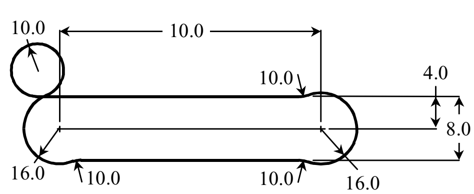
- Algebraic Inequalities
  - E.g., *d1* < 10
- Inequality-based Geometric Constraints
  - E.g., Left_Of(*Curve1*, *Point1*)

Why Is Solution Selection Necessary?

If I input this:

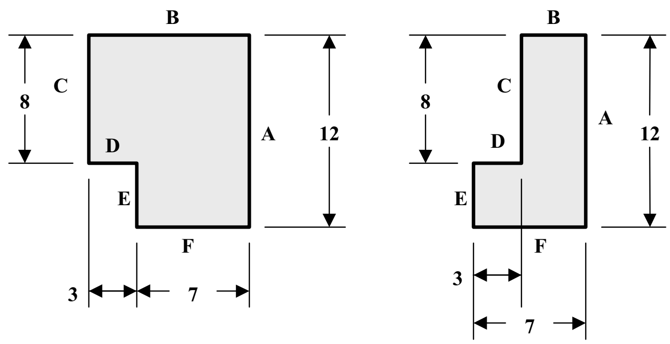


Why does the computer give me this?



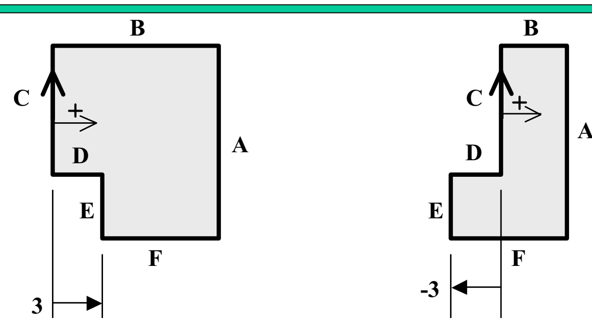
Is it because I am not using the software properly?

Multiple Solution Problem



These are 2 of 16 configurations that satisfy the geometric constraints

Existing Approaches



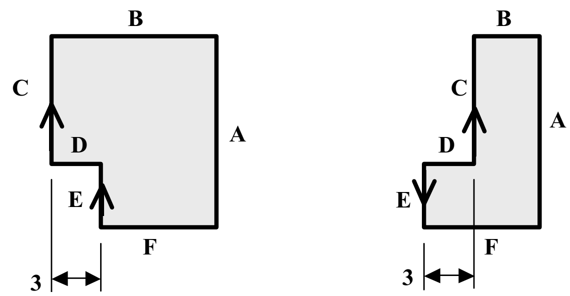
Distance (C, E, 3) **(a)**

Distance (C, E, -3) **(b)**

1. Using signed dimensions

(E.g., van Emmerick, 1991, Solano & Brunet, 1993, Kramer, 1992)

Existing Approaches



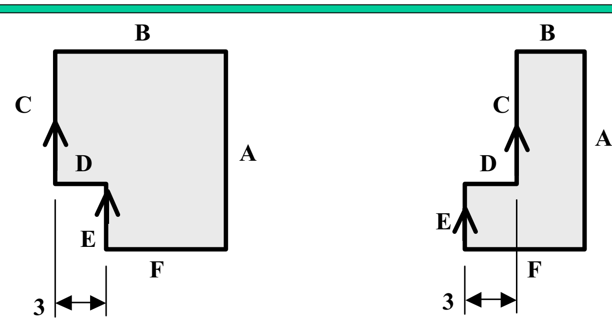
Distance (C, E, 3, right, same)
(a)

Distance (C, E, 3, left, opposite)
(b)

2. Attached constraint attributes

(Shimizu et al., 1991, Aldefeld, 1988, Owen, 1991)

Existing Approaches



Side_of (C, E, right)
(a)

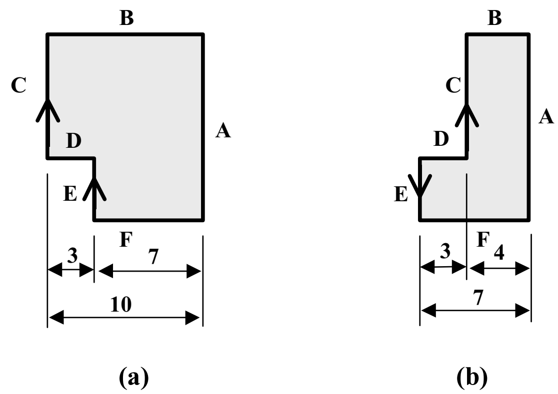
Side_of (C, E, left)
(b)

3. Additional relative position constraints

(Suzuki et al. (1990), Aldefeld (1988))

MEEM5990 Design Automation: Theories and Implementation

Existing Approaches



4. Additional Dimensions

(Buchanan and de Pennington (1993))

Existing Approaches

5. Heuristics

e.g. Line E stays on same side of C as in original sketch

(Bouma, et al. (1995))

6. Uncontrolled outcome

Iterates to the solution "closest" to the original sketch

(all purely numerical solvers)

Evaluation of Approaches

Solution Selection Method must be:

1. Sufficient to (always) choose designer’s intended solution
2. Flexible to choose using designer’s criteria (without reformulation)
3. User-oriented

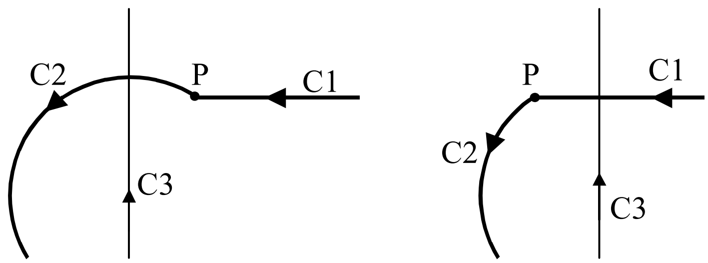
   (at geometric level of abstraction;

   only use observable orderings)

Evaluation of Approaches

1. Signed dimensions **can not be used because there are more solution decisions than there are dimensions.**
2. Constraint attributes **do not work because both solutions have the same tangency conditions.**
3. Heuristic rules **would give wrong solution.**
4. Additional dimensions **can not be used because the distances have to be EXACT.**
5. Only relative position constraints work!

Evaluation of Approaches



Real problem is to place point P on correct side of constructed line C3.

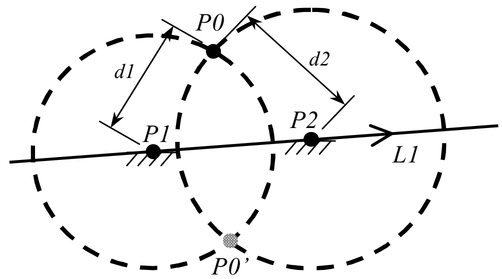
New Solution Selector Types

Procedure:

1. Identify the number of solutions resulting from each constraint/entity combinations.
2. For each solution choice, identify Solution Selectors that can be used.

Example of considering one combination in Procedure

E.g. Solving for *P0*, given *P1*, *P2*, *d1* and *d2*



2 solutions; possible Solution Selector is
On_Side(*P1*, *P2*, *P0*, left)

Issues in Deriving S.S. Types

- Arbitrary ordering
- Exponential growth in combinations

  Solution: reference auxiliary elements (e.g., create *L1* to run from *P1* to *P2;* On_Side(*L1*, *P0*, left);)
- Reference geometry doesn't always separate two solutions

  Solution: introduce objective function constraints (e.g. Closest_To(*P1*, *P2*))

Results for Planar Geometry

Geometric Constraint Combination	max # of sol's (i=1..t)	Solution Selectors that can be used
Solving for Point *P0*		
DISTANCE (*P0*, *P1*, *d1*) DISTANCE (*P0*, *P2*, *d2*)	**2**	These require auxiliary entities: C, E, H, I
DISTANCE (*P0*, *P1*, *d1*) DISTANCE (*P0*, *C2*, *d2*)	$\frac{2^{(n_2-1)} m_2}{2^r}$	A, C These require using auxiliary entities: E, H, I
DISTANCE (*P0*, *C1*, *d1*) DISTANCE (*P0*, *C2*, *d2*)	$\frac{\prod_i 2^{(n_i-1)} m_i}{2^r}$	C, E Using auxiliary entities: H, I
Solving for Curve *C0*		
PARALLEL_OFFSET (*C1*, *C0*) DISTANCE (*C1*, *C0*, *d1*)	**4**	A, C, F, J Using auxiliary entities: E, G, H, K
DISTANCE (*C0*, E_i, d_i) i=1..t	$2\prod_i \frac{2^{(n_i-1)} m_0 m_i}{2^{r_i}}$	Using auxiliary entities: C, E, F, G, H, J, K

Solution Selector Types

Relative Position Solution Selectors

A. ORDER_ON

B. IN_FRONT_OF/BEHIND

C. LEFT_OF/RIGHT_OF

D. ON_INDICATED_SIDE_OF_SURFACE
OPPOSITE_INDICATED_SIDE_ OF_ SURFACE

E. ON_INDICATED_SIDE_OF_CURVE
OPPOSITE_INDICATED_SIDE_OF _CURVE

Relative Orientation Solution Selectors

F. SAME_ORIENTATION
OPPOSITE_ORIENTATION

G. ORIENTED_LEFT_OF
ORIENTED_RIGHT_OF

Solution Selector Types

Position S.S. based on Objective Function

H. FARTHEST_FROM
CLOSEST_TO

I. FARTHEST_IN_INDICATED_DIRECTION OPPOSITE _INDICATED_DIRECTION

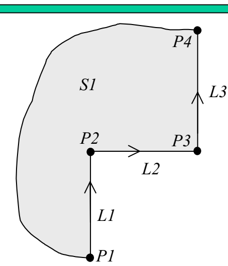
Orientation S.S. based on Objective Function

J. CLOSEST_ORIENTATION
MOST_OPPOSITE_ORIENTATION

K. ORIENTED_MOST_LEFT_OF
ORIENTED_MOST_RIGHT_OF

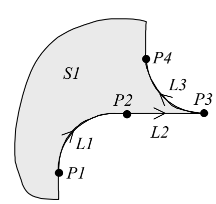
Derived Solution Selector Types (using topological information)

- CONVEX
  CONCAVE



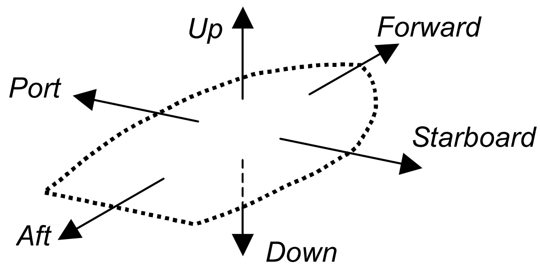
Order_On(*L1*, *P1*, *P2*)
Order_On(*L2*, *P2*, *P3*)
Order_On(*L3*, *P3*, *P4*)
Concave(*L1*, *L2*, *S1*, *P2*)
Concex(*L2*, *L3*, *S1*, *P3*)

- SMOOTH
  SHARP



Order_On(*L1*, *P1*, *P2*)
Order_On(*L2*, *P2*, *P3*)
Order_On(*L3*, *P3*, *P4*)
Smooth(*L1*, *L2*, *S1*, *P2*)
Sharp(*L2*, *L3*, *S1*, *P3*)

Coordinate-Independent Directions



Parallel (*Forward*, *Aft*)
Parallel (*Port*, *Starboard*)
Parallel (*Up*, *Down*)

Perpendicular (*Forward*, *Up*)
Perpendicular (*Starboard*, *Up*)
Perpendicular (*Forward*, *Starboard*)

Opposite_Orientation (*Forward*, *Aft*)
Opposite_Orientation (*Port*, *Starboard*)
Opposite_Orientation (*Up*, *Down*)
Oriented_Right_Of (*Forward*, *Starboard*, *Up*)

Solution Selectors – Discussion/Conclusions

- Solution Selectors could lead to user-oriented specification of which solution is desired
- But, constraint problems with Solution Selectors are likely NP-Complete to solve.