Intelligent Reuse of CAD models and Manufacturing Plans

Chandan Pitta
Department of Electrical and Computer Engineering
The University of Arizona
chandanp@ece.arizona.edu

Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Objective

- Develop and implement an Automated System to Intelligently produce process plans for a feature based prismatic machinable parts by integrating Variant and Generative process planning

Motivation

- Produce low volumes of a large variety of products, hence shorter lead times for manufacturing
- The process of manufacturing a part involves
  - Develop a 3D solid model
  - Using the solid model, retrieve similar parts from database
  - Modify old plan or develop it from scratch
- Traditionally process plans were developed by manual process planners who gain their skills through experience
- Manual process planning technique takes considerable time depending on the skill and experience of the planner
- Automating all the tasks reduces lead time. Integration of CAD and CAM

Background

- CAD describes geometry and topology
- CAM focuses on process plan
- Process plan for a part is a plan that outlines the processing route, operations, fixtures and tools required to produce the part at least cost
- Computer aided process planning can be classified as
  - Variant process planning
  - Generative process planning

Variant Process Planning

- Given a new part P, this process retrieves process plan PP' of an old part P' which is similar to P and modifies it to fit to the new part

- Advantages
  - Easy to implement
  - Reuses knowledge generated from previous manufacturing processes and hence is faster

- Disadvantages
  - Retrieval of similar parts depends on classification scheme
  - If there is no similar part in the database, then it is not possible to use this method
  - The optimality of the generated process plan depends on the optimality of the previous plans stored in the database

Generative Process Planning

- Creates the process plan from scratch using information of available machines, tools, their capabilities and rules for planning

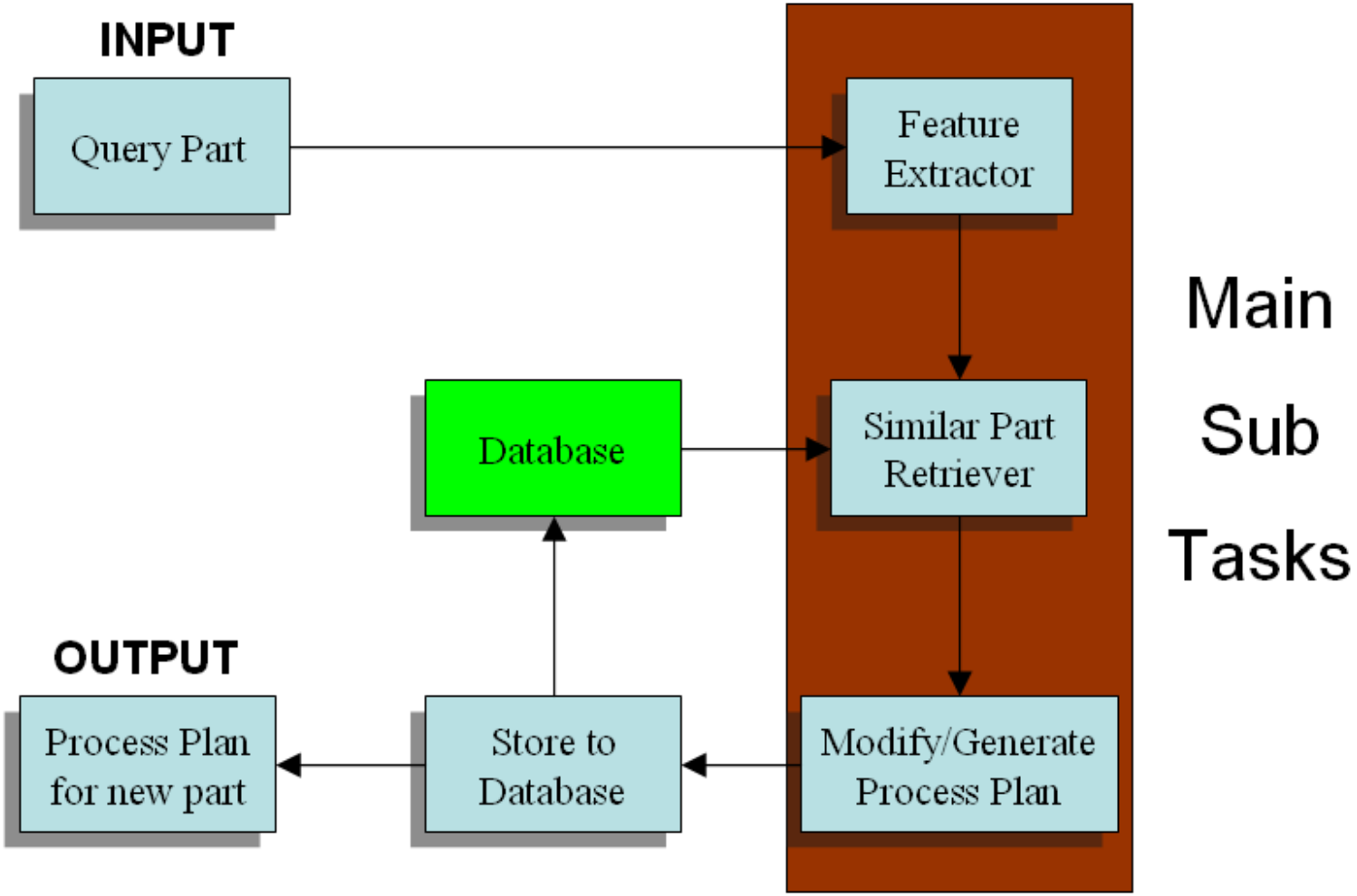
- Advantages
  - Process plan for any part can be generated, given that enough resources are available and the rules for determing the processes, tools and setups are accurate
  - Optimality is assured based on the optimizing algorithm

- Disadvantages
  - Plans from scratch for every part even if a similar plan has already been manufactured. Does not reuse knowledge
  - A complete solution to generate process plan has not been developed. It is difficult to implement

Current Research - Hybrid Approach

- Combines the advantages of both methods
- Given a new part:
  - Ability to retrieve similar part from database if one exists
  - Modify the old plan to fit the new part
  - Or generate a plan if similar part does not exist

General Framework



Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Related Work

- Feature Recognition
  - Graph Based Heuristics for Recognition of Machined Features from a 3-D Solid Model (Joshi and T. C. Chang)
    - Graph based approach
    - Problems with feature Intersections
  - Feature Extraction by Volume Decomposition (T. Woo)
    - Convex hull decomposition
    - May generate undesirable features

Related Work (cont.)

- Feature Recognition
  - Geometric Computation for the Recognition of Spatially Interacting Machinable Features (J. H. Vandenbrande and A. A. G. Requicha)
    * Hint based Approach
    * Expensive computations may not lead to a feature
  - Constraint-Based Feature Recognition: Handling Non uniqueness in Feature Interactions (Ming-Hsuan Yang and Michael M. Marefat)
    * Geometric Constraint Satisfaction
    * Interacting features are recognized correctly

Related Work (cont.)

- Retrieval of similar parts
  - Defining Specialized Design Similarity Measures (Jeffrey W. Herrmann, Sundar Balasubramanian and Gurdip Singh)
    - Neural networks for learning, based on maximum inter vertex distance, maximum vertex-edge distance and total enclosed area
    - Used only for fixture planning
  - Using Shape Distributions to Compare Solid Models (Cheuk Yiu Ip, Daniel Lapadat, Leonard Seiger, William C. Regli)
    - Uses shape distribution histogram
    - Works only for simple parts

Related Work (cont.)

- Retrieval of similar parts
  - Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects (Hans Peter Kriegel, Stefan Brecheisen, Peer Kroger, Martin Pfeifle, Matthias Schubert)
    * Voxel based similarity
    * Slow
  - An Algorithm for Deciding Similarities of 3D Objects (Shinji Mukai, Susumu Furukawa, Mitsuru Kuroda)
    * Based on comparing convex components
    * Fast, able to determine scale factor and similarity factor
    * Not rotation invariant

Related Work (cont.)

- Retrieval of similar parts
  - Feature-based Similarity Assessment of Solid Models (Alexei Elinson, Dana S. Nau, William C. Regli)
    - Signature comparison using graphs
    - Not complete (some parts cannot be identified uniquely)
  - Machining Feature-based Comparisons of Mechanical Parts (Vincent Cicirello, William C. Regli)
    - Graph isomorphism problem
    - Heuristic measure hence may not produce best match
  - Automatic Retrieval of Similarly Shaped Machinable Components (Mark Ascher and Michael M. Marefat)
    - Graph based matching, Type Abstraction Hierarchy
    - Similarity calculated based on single part

Related Work (cont.)

- Reuse/Generate plans
  - Knowledge Representation for Automated Process Planning (Dusan N. Sormaz, Behrokh Khoshnevis)
    * A knowledge scheme for integration of CAPP is introduced
  - An intelligent feature-based process planning system for prismatic parts (Lalit Patil and S. S. Pande)
    * Uses Feature Based Modelling and Intelligent process planner
    * Does not take advantage of previously developed parts

Related Work (cont.)

- Reuse/Generate plans
  - Plan Reuse and Plan Merging in Manufacturing Process Planning (M. Marefat, J Britanik)
    * Hierarchical Plan Merging, using multiple plan
    * Does not take interactions into account
  - Adaptation of Plans via Annotation Verification (Subbarao Kambhampati)
    * Mapping processes
    * Does not take interactions into account

Related Work (cont.)

- Reuse/Generate plans
  - Toward Hybrid Variant/Generative process planning (Alexei Elinson, Jeffrey W. Herrmann, Ioannis E. Minis, Dana S. Nau, Gurdip Singh)
    * Graph Isomorphism method
    * Slow in matching
  - The Research and Application on Hybrid Intelligence CAPP System Based on Object Oriented Model Driving (Jia Xiao-liang, Zhang Zhen-ming, Xu Jian-xin, Huang Nai-kang)
    * Hybrid intelligent process planning
    * Requires user intervention

Related Work (cont.)

- Reuse/Generate plans
  - Integrated machining tool path planning using feature free spaces (Yong Se Kim, Eric Wang, Il-kyu Hwang and Hyung Min Rho)
    * Automatic machining tool path generation method that integrates local and global tool path planning
  - Integrated Process Planning using Tool/Process Capabilities and Heuristic Search (Ramanujam Raman, Michael M. Marefat)
    * Sequencing algorithm for setups and tools to optimize process plan

Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Importance

- Generating process plan from scratch, wastes a lot of time
- Using an old part that is similar, and modifying it as required is faster. Maintains optimality
- If a large database of parts is involved, it consumes a lot of time even for an experienced person to retrieve similar part
- Hence manufacturing industries use GT codes to classify the parts for easy indexing
- Given a new part, the GT code for it is generated which directly gives the index to the set of parts that are similar in the database

Importance (cont.)

- Problems with GT classification
  - The classification should take care of all types of parts
    - This can lead to a large code
    - Difficult to avoid overlaps in classification
  - The codes should be chosen in such a manner that too many or too few parts do not fall in each family
    - 100 parts and 2 families with 50 parts each - too many parts in one family to compare to the new part
    - 100 parts classified into 50 families of 2 parts each - index may not point to the right family

Importance (cont.)

- Automating GT code generation and indexing is not helpful
- We develop a scheme which uses shape and manufacturing similarities for classification and finding similar parts
- Integrate variant and generative process planning

Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Contributions

- A database indexing scheme is proposed
  - Eliminate dissimilar parts
  - Speeds up retrieval process
- A heuristic graph matching algorithm is developed
  - Based on spatial relationship
  - Manufacturing similarities
- Integrate variant and generative process planning
  - Reuses planning knowledge
  - Reduces lead time
  - Produce process plan even if no similar part is exists
- Implement a plan optimization algorithm
  - For generative process planning
  - For variant process planning

Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Problem Formulation

Definition 1 *A* **feature** *is the most basic volume. A combination of such features produces the delta volume*

Definition 2 Delta volume *of a part is the volume when removed from the stock produces the part.*

Definition 3 *A* **block** B *is a set of features* $\{F_1, F_2, \ldots\}$ *such that* $F_i \in B$ *interacts with some* $F_j \in B, \forall i$.

Definition 4 *Two features are said to be* **interacting** *if a face of one feature is touching the face of another feature or if both features share a common volume.*

Problem Formulation (cont.)

Given:

P = set of parts previously manufactured = $\{p_1, p_2, \ldots\}$
B_i = set of blocks constituting the delta volume of p_i = $\{b_{i1}, b_{i2}, \ldots\}$

$B = \{B_1, B_2, \ldots\}$
BP_i = set of block process plans for $b_{ij} \in B_i, \forall j$ = $\{bp_{i1}, bp_{i2}, \ldots\}$
$BP = \{BP_1, BP_2, \ldots\}$
R = available resources (machines and tools)
p = new part to be manufactured

We want to develop an optimized process plan opp for the new part p as

opp = OptimizeProcessPlan (pp)
pp = GenerateProcessPlan (p, PPB, R)
PPB = BestMatch (p, b, P, B, BP) where $b = \{b_1, \ldots\}$, blocks of p
b = FeatureRecognition (p)

Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

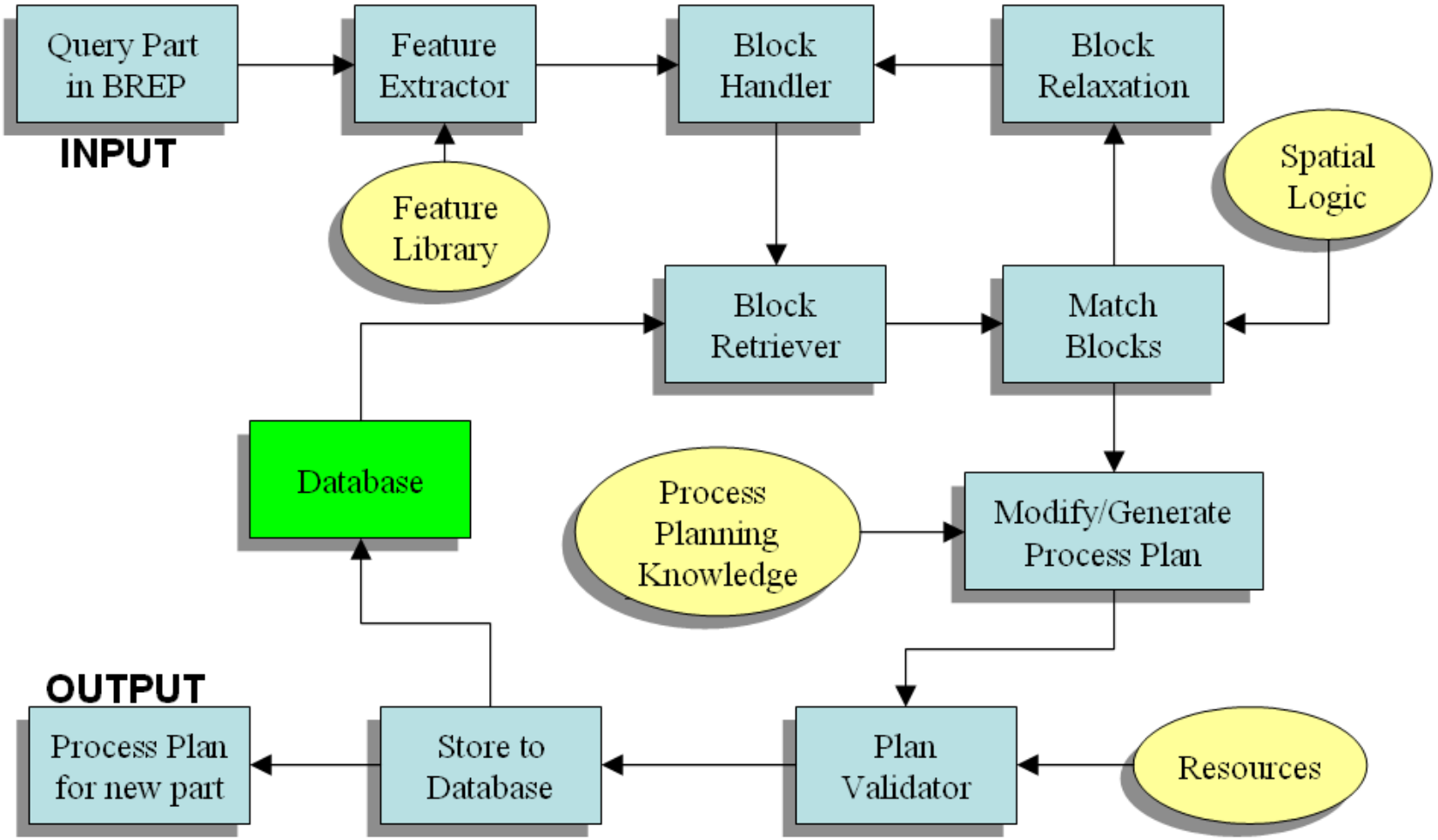
Solution Approach

- Feature Recognition
  - Identify features using CAD file
- Retrieving Similar parts
  - Indexing method used to retrive candidate similar parts
  - Retrieved parts are matched and ranked
- Modify/Generate Plan
  - Modify old plans to fit the new part with the available resources
  - Generate a new plan if old plan does not exist
- Optimize process plan
- Validate plan

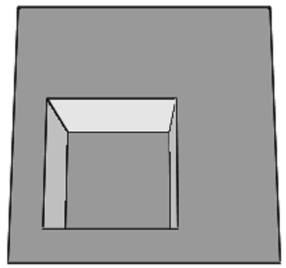
Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

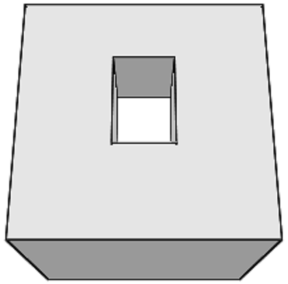
System Framework



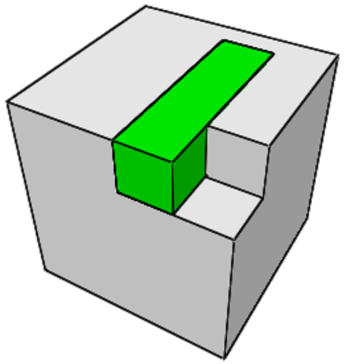
Feature Primitives



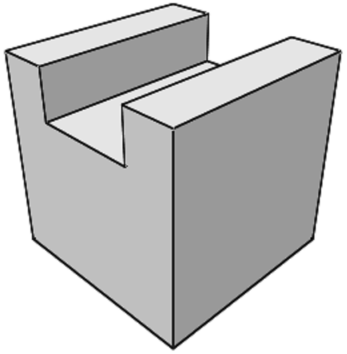
Pocket



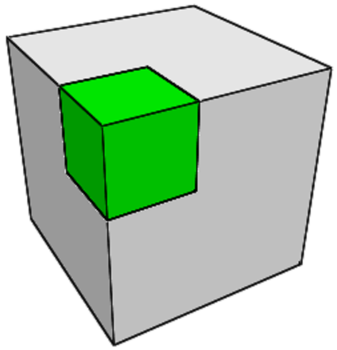
Hole



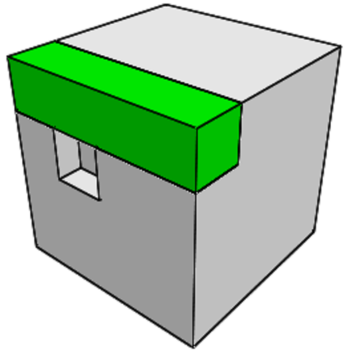
Blind Slot



Slot

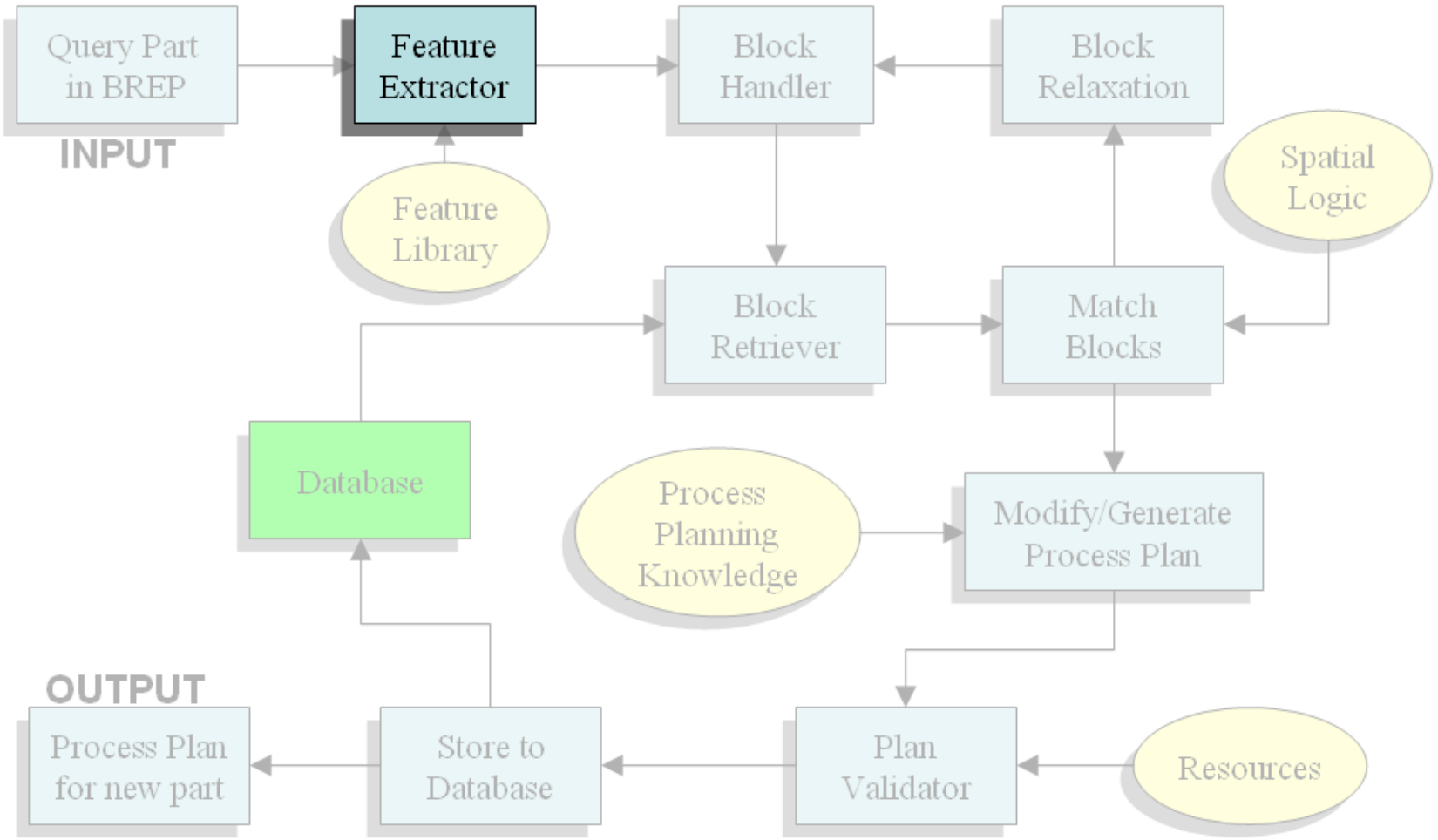


Blind Step



Step

Feature Recognizer



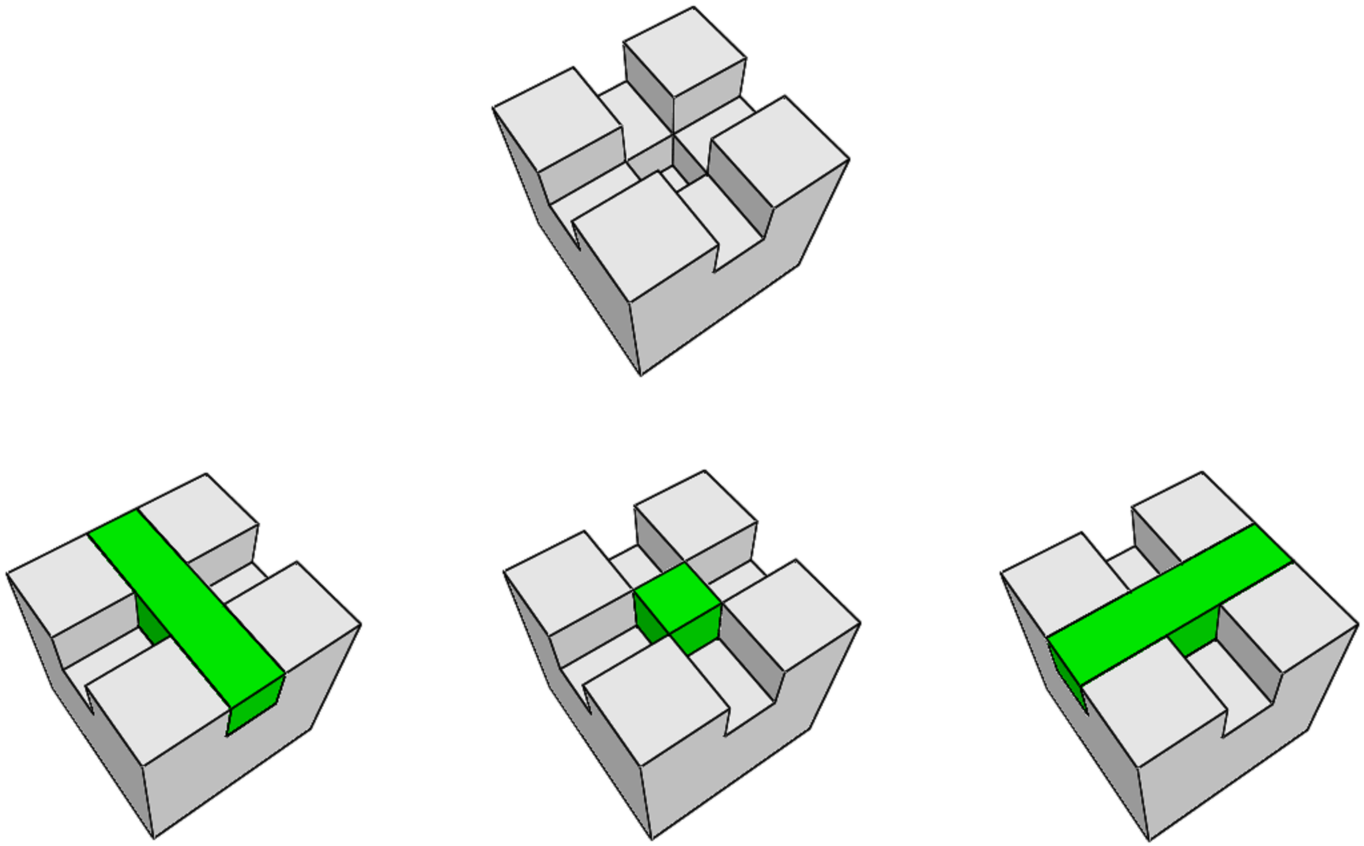
Feature Recognizer (cont.)

- Constraint-Based Feature Recognition: Handling Non uniqueness in Feature Interactions (Ming-Hsuan Yang and Michael M. Marefat)
  - Works for prismatic parts
  - Interacting features are recognized correctly
- A constraint graph is constructed for each primitive feature using the faces as nodes and the predicates as the edges between faces
- Using the constraint graph we try to map a face in the new part to the face in the feature graph

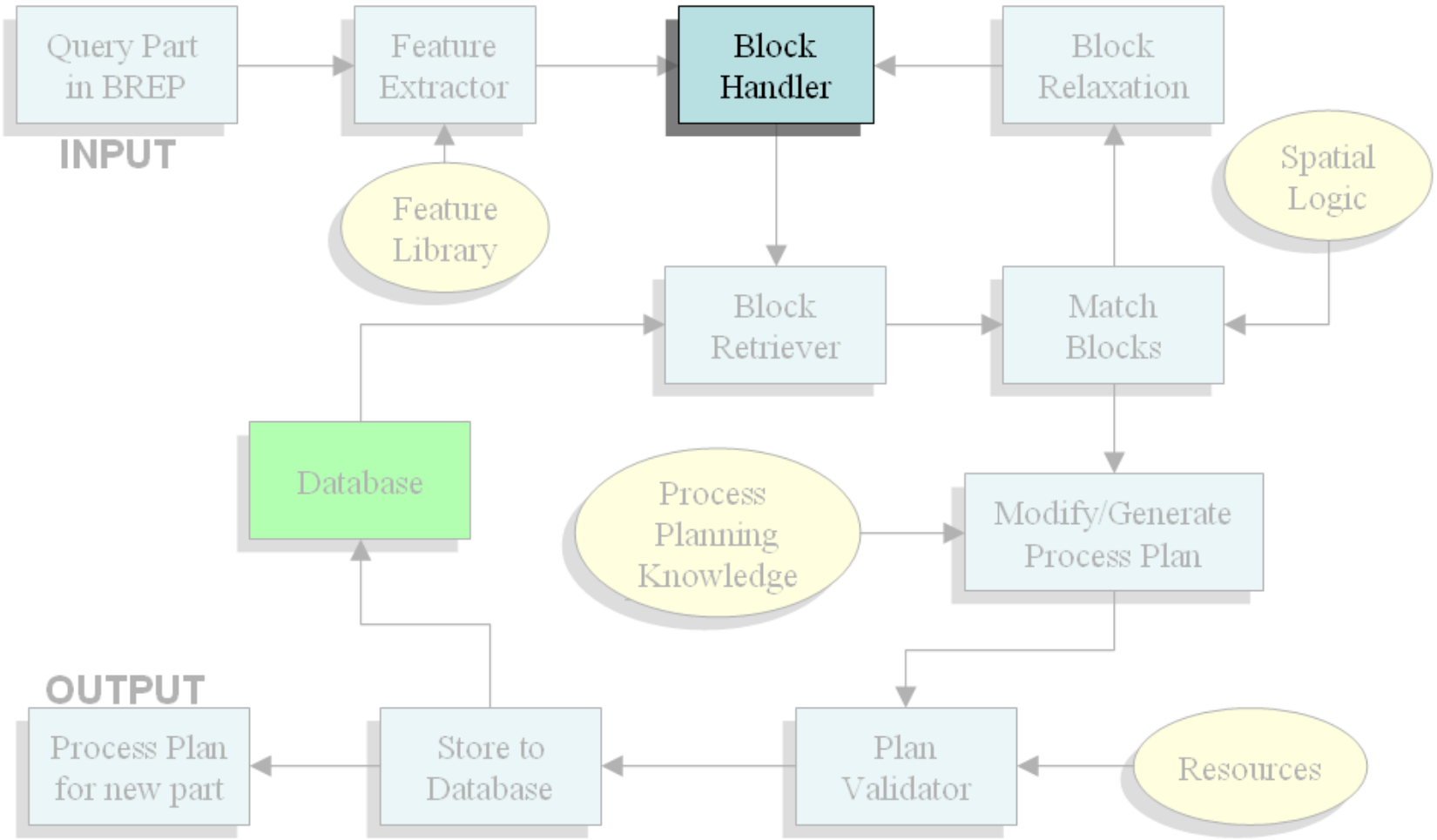
Feature Recognizer (cont.)

- After the assignment of nodes the edges are checked for predicate constraints
- If the constraints between the part faces exactly match the feature graph constraints then we found the feature
- But since the face information in the part might change because of interactions, some relaxation is allowed
- Finally we find the vertices using intersection of the three faces.

Feature Recognizer (cont.)

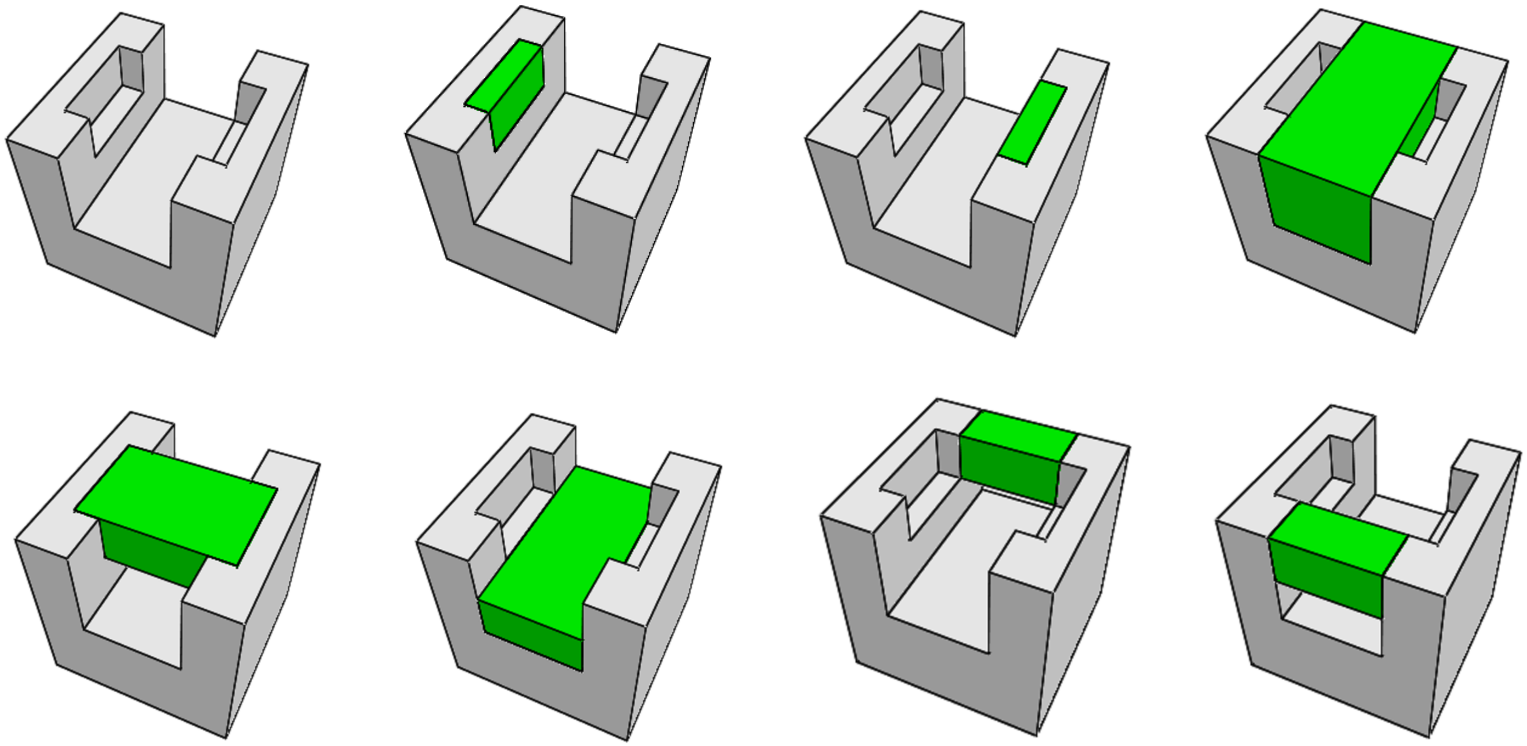


Block Handler



Block Handler (cont.)

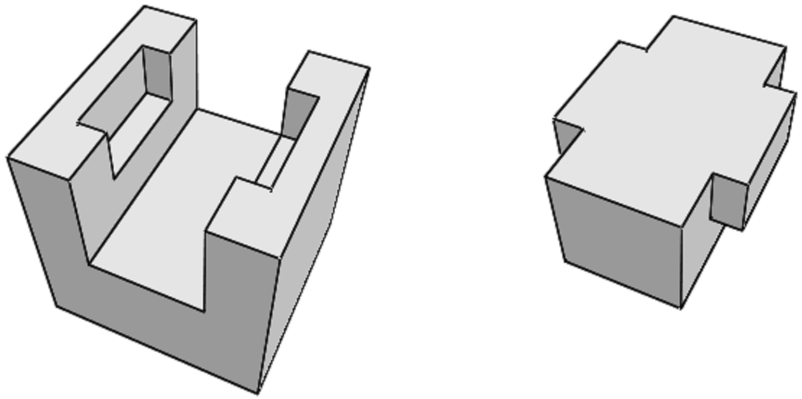
- A block handler combines features together to form several blocks which together make up the delta volume.
- Problem of multiple interpretations - use maximal features



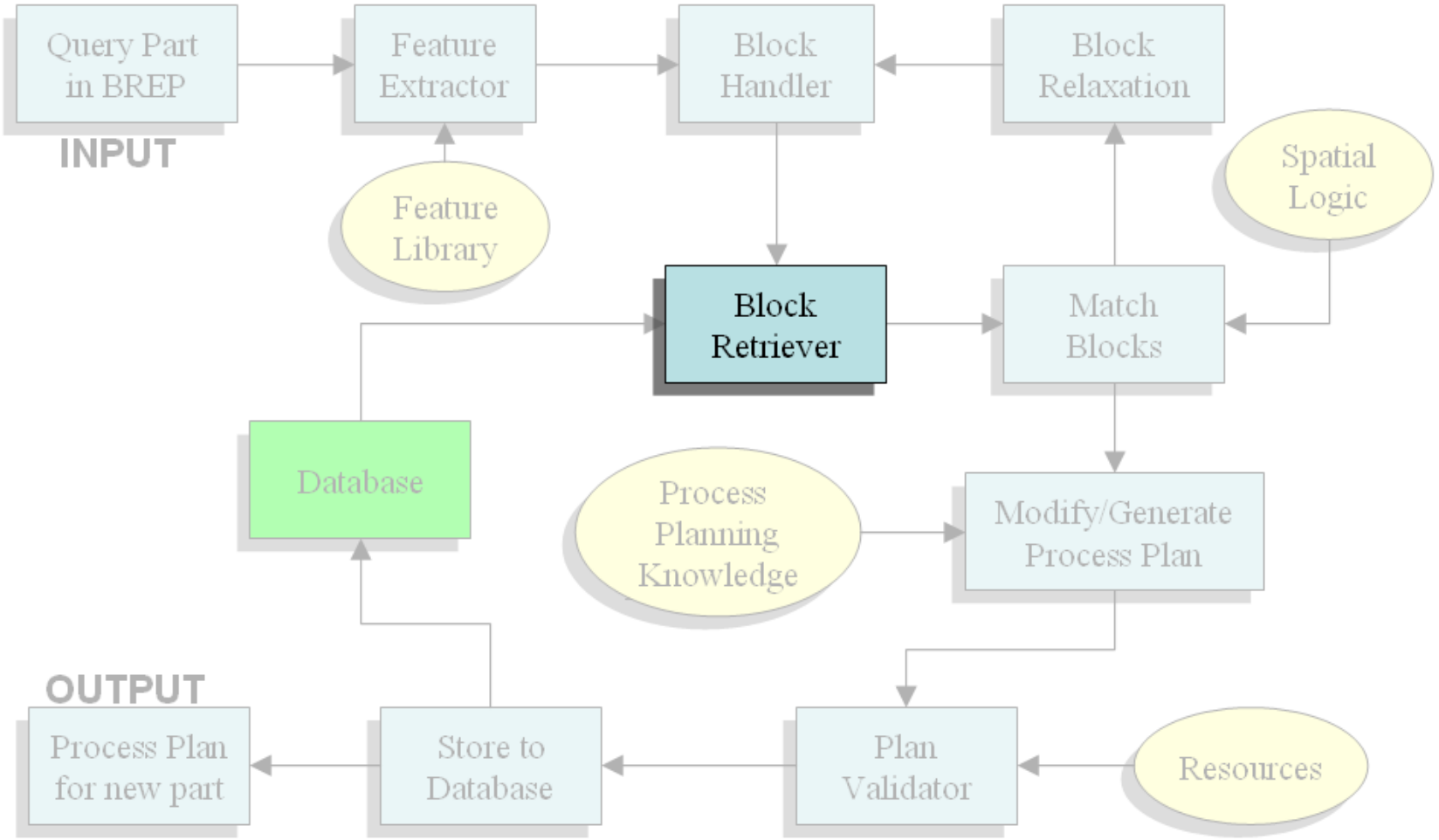
Block Handler (cont.)

- Use maximal features only

  Definition 5 *A* **maximal feature** *is a feature that is not subsumed by any other feature. If* F *is a set of* n *features such that* $f_m \in F$*, and* $f_{i=0\ldots m-1,m+1\ldots n} \cap f_m \neq f_m$*,* $\forall i \in F$*, then* f_m *is a maximal feature*



Block Retriever



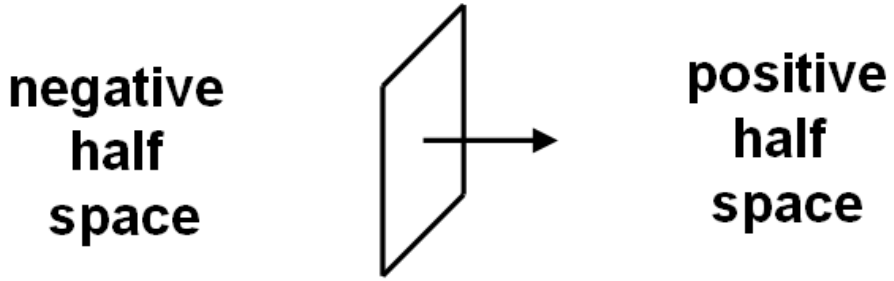
Block Retriever (cont.)

- The block retriever, prepares an index into the database and obtains relevant parts

- Spatial Relationship
  - A spatial relationship between two features is represented by a 3 letter string
  - The three letters represents the interactions between the three face pairs in a feature
  - The relationship is defined in terms of half spaces of the faces of the interacting features

Spatial Relationship

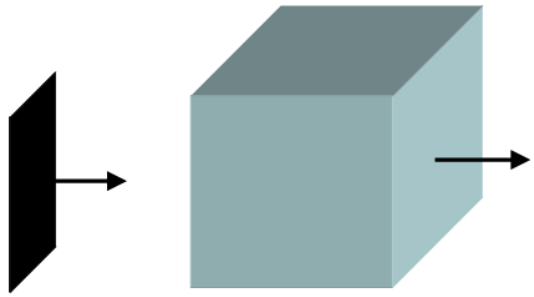
Definition 6 *Given a face, f, the plane in which f lies, divides the space into two half spaces. The half space in which the normal of the face points is called the* **positive half space** *denoted as f_+ and the other half space is called the* **negative half space** *denoted by f_-.*



Definition 7 *A* **face pair** *is the pair of opposite faces in a feature. The three face pairs in a feature are {left, right}, {top, down} and {front back}.*

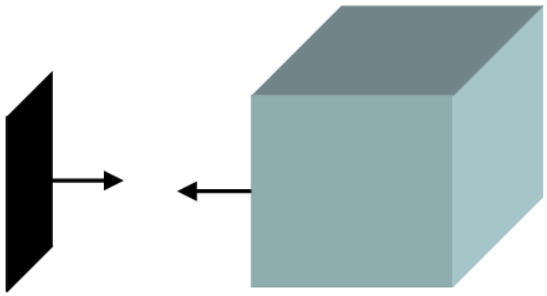
Spatial Relationship (cont.)

Definition 8 *For any two features $F_i \in F$ and $F_j \in F$, where F is the set of all features of a part, there exists a mapping function G, such that $G : F_i \rightarrow F_j$. The result of this mapping is that for each face $f_{ia} \in F_i$ there is a face $f_{jb} \in F_j$, such that the normal of f_{ia} is pointing in the same direction as the normal of f_{jb}. This relationship is termed as* **same as** *relationship. Henceforth the same as face of face f_{ia} in F_j is represented as $f_{ia\triangleright}$. The face $f_{ia\triangleright}$ is f_{jb} from the definition.*



Spatial Relationship (cont.)

Definition 9 *For any two features $F_i \in F$ and $F_j \in F$, where F is the set of all features of a part, there exists a mapping function H, such that $H : F_i \rightarrow F_j$. The result of this mapping is that for each face $f_{ia} \in F_i$ there is a face $f_{jc} \in F_j$, such that the normal of f_{ia} is pointing in the opposite direction as the normal of f_{jc}. This relationship is termed as* **opposite of** *relationship. Henceforth the opposite of face f_{ia} in F_j is represented as $f_{ia\triangleleft}$. The face $f_{ia\triangleleft}$ is f_{jc} from the definition.*



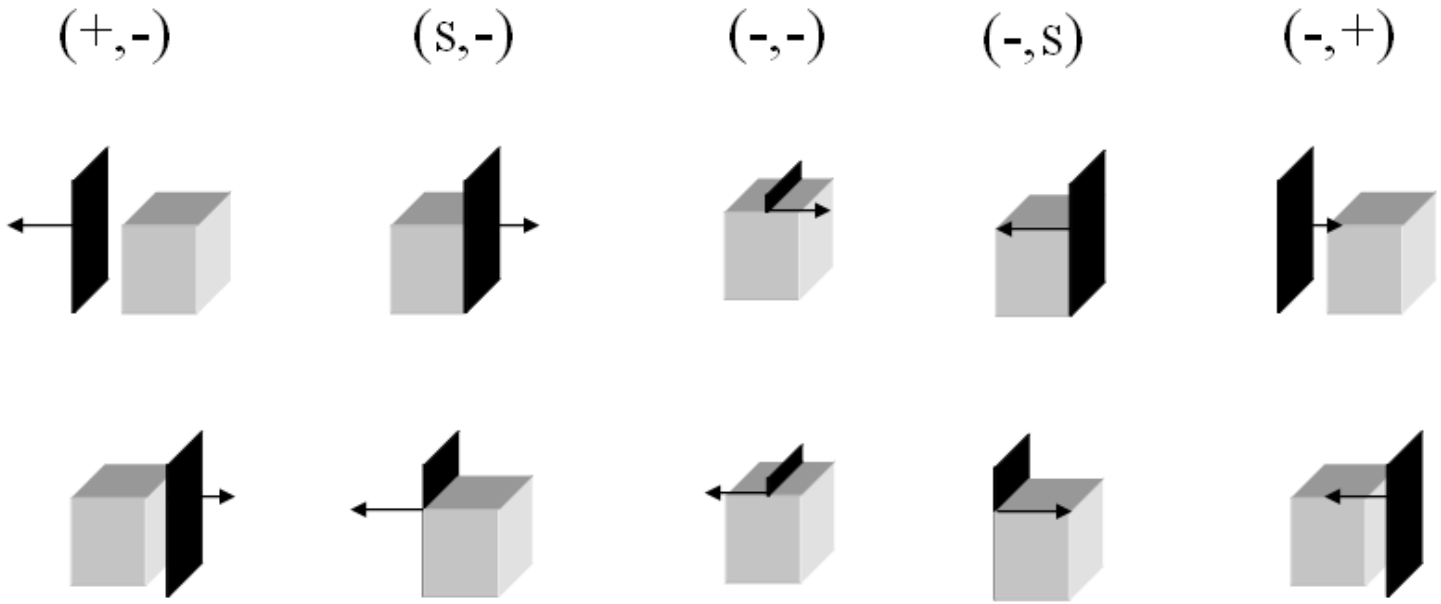
Spatial Relationship (cont.)

Definition 10 *The* **spatial face relation** *between a face* f_{ia} *of feature* F_i *and a pair of faces* $f_{ia\triangleright}$ *and* $f_{ia\triangleleft}$ *of the other feature* F_j *that interacts with* F_i *is a two tuple* (x, y) *such that*

$$x = \begin{cases} + & \text{if } f_{ia} \text{ lies in } f_{ia\triangleright +} \\ - & \text{if } f_{ia} \text{ lies in } f_{ia\triangleright -} \\ s & \text{if } f_{ia} \text{ lies in the same plane as } f_{ia\triangleright} \end{cases}$$

$$y = \begin{cases} + & \text{if } f_{ia} \text{ lies in } f_{ia\triangleleft +} \\ - & \text{if } f_{ia} \text{ lies in } f_{ia\triangleleft -} \\ s & \text{if } f_{ia} \text{ lies in the same plane as } f_{ia\triangleleft} \end{cases}$$

Spatial Relationship (cont.)



Definition 11 *A* **spatial relation** *between two features* F_i *and* F_j *is represented by a three letter sequence* XYZ *where* $X, Y, Z \in \{A, B, C, D, E, F, G\}$

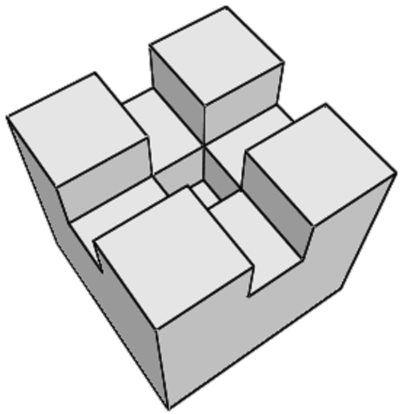
Spatial Relationship (cont.)

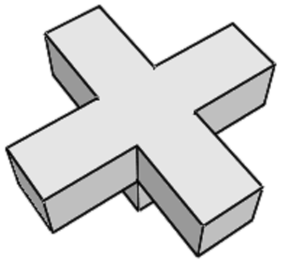
Spatial face relationship table

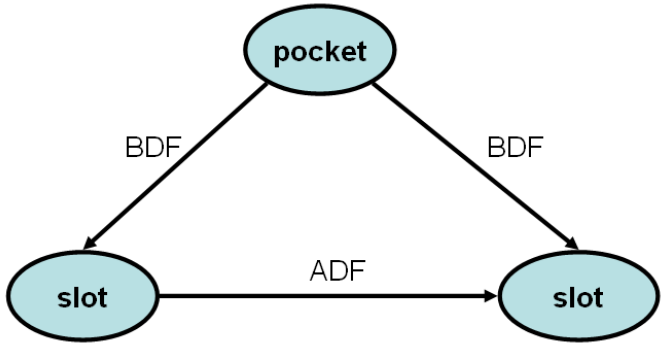
Type	Face f_1	Opp. of f_1	Relation with the other feature
A	(-, -)	(-, -)	Both faces inside
B	(s, -)	(-, -)	One face is shared, other inside
C	(-, -)	(+, -)	One face inside, other outside
D	(s, -)	(s, -)	Both faces shared
E	(s, -)	(+, -)	One face shared, other outside
F	(+, -)	(+, -)	Both faces Outside
G	(+, -)	(-, s)	One face shared, other outside
	(s, -)	(-, s)	Not possible
	(-, -)	(-, s)	Not possible
	(-, s)	(-, s)	Not possible

Feature Interaction Graph

- A feature interaction graph is a graph representing the features and the spatial relationships between the features in a block
- The nodes of the FIG are represented by features and the edges are represented by the spatial relationship between the features

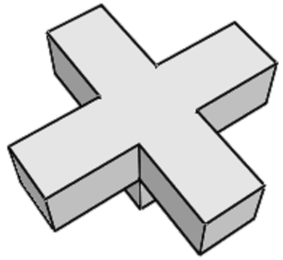






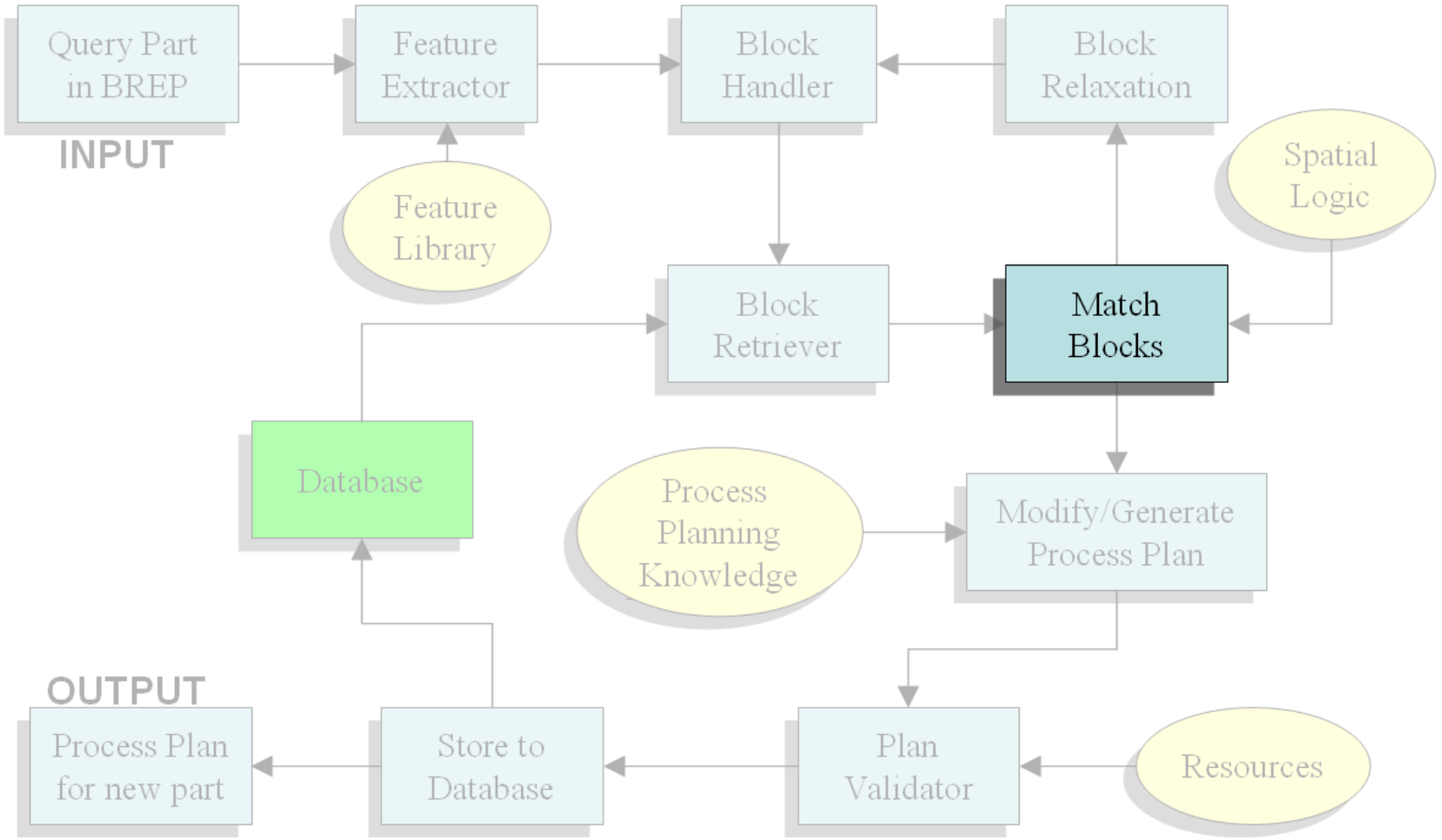
Database Indexing

- The number of features and the number of interactions of each part are stored in the database for indexing purpose.



- Number of features = 3, number of interactions = 3
- Given a new part with n features and m interactions, the block retriever returns all the parts from the database with n features and m interactions.

Match Blocks

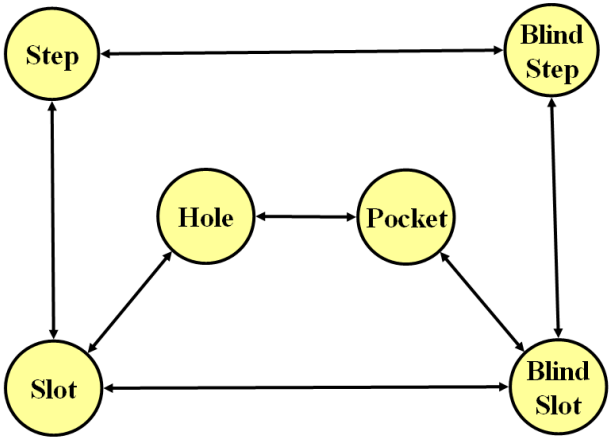


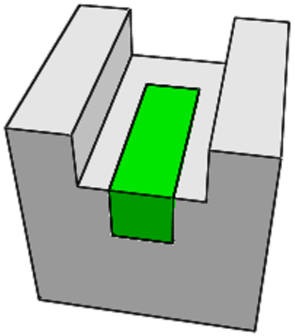
Match Blocks (cont.)

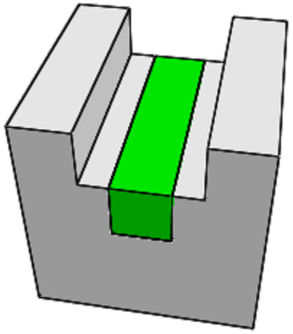
- Feature interaction graph matching is isomorphism problem
- Graph isomorphism is an NP-complete problem
- Heuristic approach to solve the problem of graph matching
- Construct cost matrices
  - Node Cost Matrix
  - Arc Cost Matrix
  - Alignment Cost

Node Cost Matrix

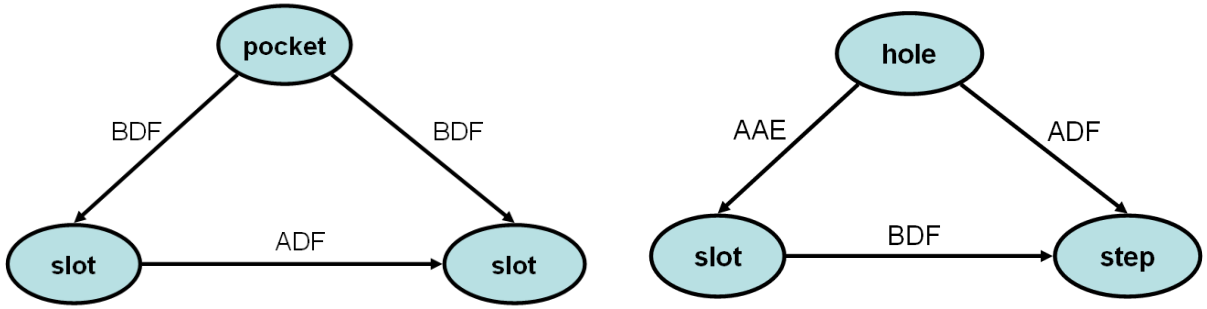
- The cost matrix is a $n \times n$ matrix
- Cost to match each node $v_k \in f_i$ with each node $v_l \in f_j$
- Since the nodes might be different relaxation is allowed
- Amount of relaxation is considered as the cost of converting node $v_{i,k}$ to $v_{j,l}$







Node Cost Matrix (cont.)



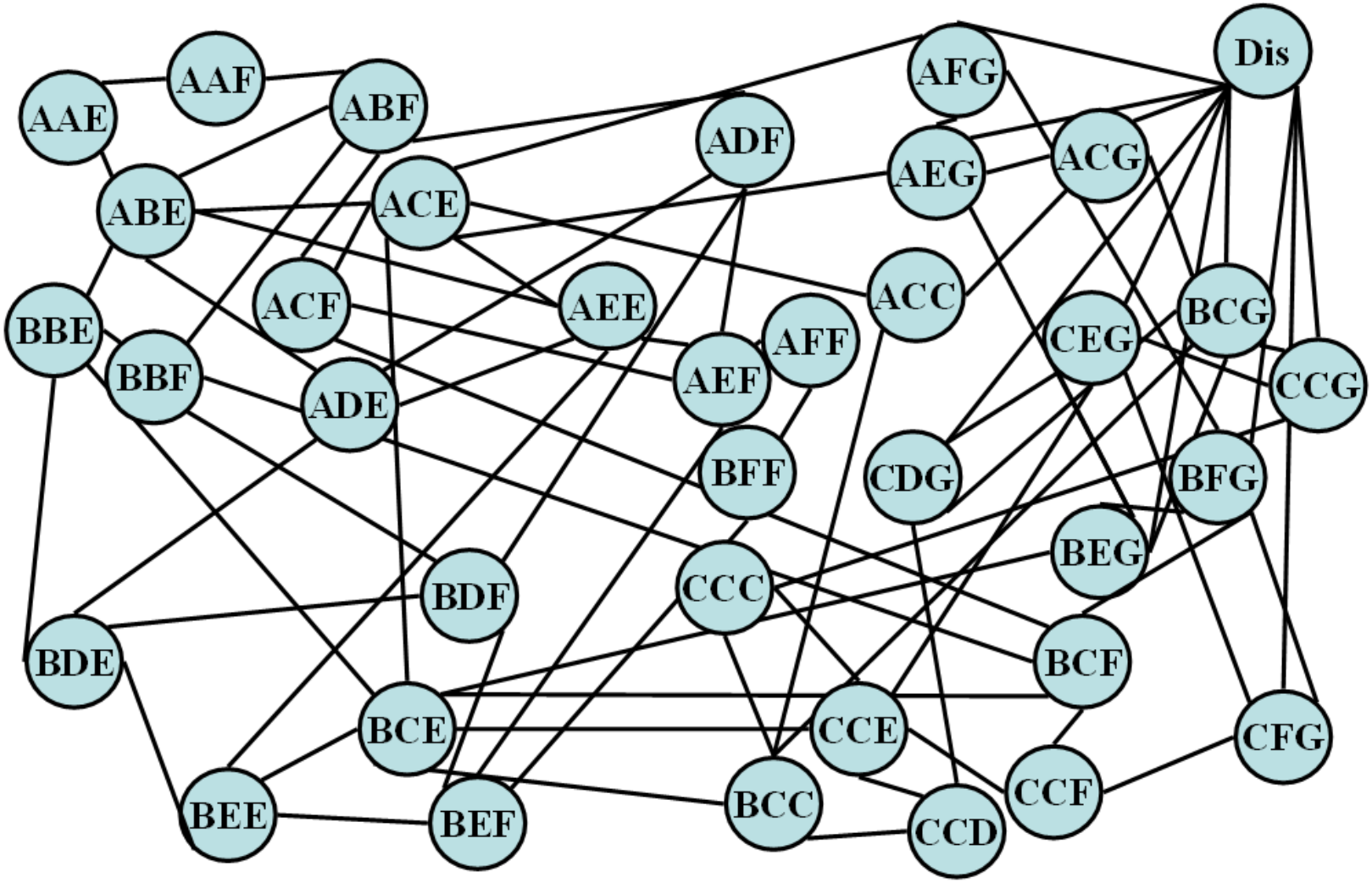
	Pocket	Hole	Blind Slot	Slot	Blind Step	Step
Pocket	0	1	1	2	2	3
Hole	1	0	2	1	3	2
Blind Slot	1	2	0	1	1	2
Slot	2	1	1	0	2	1
Blind Step	2	3	1	2	0	1
Step	3	2	2	1	1	0

$$\begin{bmatrix} & pocket & slot & step \\ hole & 1 & 1 & 2 \\ slot & 2 & 0 & 1 \\ step & 3 & 1 & 0 \end{bmatrix}$$

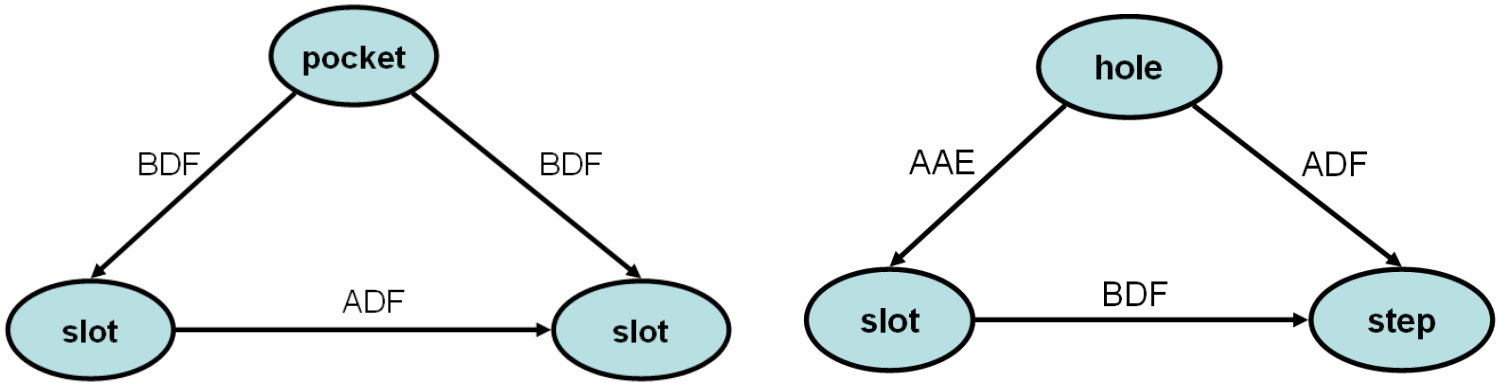
Arc Cost Matrix

- For every node assignment construct an arc cost matrix
- It is a $p \times p$ matrix
- p is the max of (# of arcs adj to $v_{i,k}$, # of arcs adj to $v_{j,l}$)
- Suppose
  - Number of adj arcs to $v_{i,k} = p$
  - Number of adj arcs to $v_{j,l} = q$
  - and $p > q$ then
  - the rest of $p - q$ elements are arcs of type `dis`
- There is a cost for converting any interaction to `dis`

Arc Cost Matrix (cont.)



Arc Cost Matrix (cont.)



$$
\begin{bmatrix}
 & pocket & slot & step \\
hole & \begin{bmatrix} & BDF & BDF \\ AAE & 4 & 4 \\ ADF & 1 & 1 \end{bmatrix} & \begin{bmatrix} & BDF & ADF \\ AAE & 4 & 3 \\ ADF & 1 & 0 \end{bmatrix} & \begin{bmatrix} & BDF & ADF \\ AAE & 4 & 3 \\ ADF & 1 & 0 \end{bmatrix} \\
slot & \begin{bmatrix} & BDF & BDF \\ AAE & 4 & 4 \\ BDF & 0 & 0 \end{bmatrix} & \begin{bmatrix} & BDF & ADF \\ AAE & 4 & 3 \\ BDF & 0 & 1 \end{bmatrix} & \begin{bmatrix} & BDF & ADF \\ AAE & 4 & 3 \\ BDF & 0 & 1 \end{bmatrix} \\
slot & \begin{bmatrix} & BDF & BDF \\ BDF & 0 & 0 \\ ADF & 1 & 1 \end{bmatrix} & \begin{bmatrix} & BDF & ADF \\ BDF & 0 & 1 \\ ADF & 1 & 0 \end{bmatrix} & \begin{bmatrix} & BDF & ADF \\ BDF & 0 & 1 \\ ADF & 1 & 0 \end{bmatrix}
\end{bmatrix}
$$

Arc Cost Matrix (cont.)

- Cost of converting adj nodes of $v_{i,k}$ to the adj nodes of $v_{j,l}$

$$
\begin{array}{c}
\begin{array}{ccc} & pocket & \end{array} \\
\begin{array}{cc} hole & \end{array}
\begin{bmatrix}
 & slot & step \\
slot & 0 & 1 \\
slot & 0 & 1
\end{bmatrix}
\end{array}
\quad
\begin{array}{c}
slot \\
\begin{bmatrix}
 & pocket & step \\
slot & 2 & 1 \\
slot & 2 & 1
\end{bmatrix}
\end{array}
\quad
\begin{array}{c}
step \\
\begin{bmatrix}
 & pocket & slot \\
slot & 2 & 1 \\
slot & 2 & 1
\end{bmatrix}
\end{array}
$$

$$
\begin{array}{cc} slot & \end{array}
\begin{bmatrix}
 & slot & step \\
hole & 1 & 2 \\
slot & 0 & 1
\end{bmatrix}
\quad
\begin{bmatrix}
 & pocket & step \\
hole & 1 & 2 \\
slot & 2 & 1
\end{bmatrix}
\quad
\begin{bmatrix}
 & pocket & slot \\
hole & 1 & 2 \\
slot & 2 & 1
\end{bmatrix}
$$

$$
\begin{array}{cc} slot & \end{array}
\begin{bmatrix}
 & slot & step \\
slot & 0 & 1 \\
hole & 1 & 2
\end{bmatrix}
\quad
\begin{bmatrix}
 & pocket & step \\
slot & 2 & 1 \\
hole & 1 & 2
\end{bmatrix}
\quad
\begin{bmatrix}
 & pocket & slot \\
slot & 2 & 1 \\
hole & 1 & 2
\end{bmatrix}
$$

Arc Cost Matrix (cont.)

- Add the two matrices

$$
\begin{array}{l}
 \\
hole \\
 \\
 \\
slot \\
 \\
 \\
slot \\
 \\

\end{array}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (slot) \end{array} & \begin{array}{c} BDF \\ (step) \end{array} \\
\begin{array}{c} AAE \\ (slot) \end{array} & 4 & 5 \\
\begin{array}{c} ADF \\ (slot) \end{array} & 1 & 2
\end{bmatrix}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (pocket) \end{array} & \begin{array}{c} ADF \\ (step) \end{array} \\
\begin{array}{c} AAE \\ (slot) \end{array} & 6 & 4 \\
\begin{array}{c} ADF \\ (slot) \end{array} & 3 & 1
\end{bmatrix}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (pocket) \end{array} & \begin{array}{c} ADF \\ (slot) \end{array} \\
\begin{array}{c} AAE \\ (slot) \end{array} & 6 & 4 \\
\begin{array}{c} ADF \\ (slot) \end{array} & 3 & 1
\end{bmatrix}
$$

$$
\begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\

\end{array}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (slot) \end{array} & \begin{array}{c} BDF \\ (step) \end{array} \\
\begin{array}{c} AAE \\ (hole) \end{array} & 5 & 6 \\
\begin{array}{c} BDF \\ (slot) \end{array} & 0 & 1
\end{bmatrix}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (pocket) \end{array} & \begin{array}{c} ADF \\ (step) \end{array} \\
\begin{array}{c} AAE \\ (hole) \end{array} & 5 & 5 \\
\begin{array}{c} BDF \\ (slot) \end{array} & 2 & 2
\end{bmatrix}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (pocket) \end{array} & \begin{array}{c} ADF \\ (slot) \end{array} \\
\begin{array}{c} AAE \\ (hole) \end{array} & 5 & 5 \\
\begin{array}{c} BDF \\ (slot) \end{array} & 2 & 2
\end{bmatrix}
$$

$$
\begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\

\end{array}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (slot) \end{array} & \begin{array}{c} BDF \\ (step) \end{array} \\
\begin{array}{c} BDF \\ (slot) \end{array} & 0 & 1 \\
\begin{array}{c} ADF \\ (hole) \end{array} & 2 & 3
\end{bmatrix}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (pocket) \end{array} & \begin{array}{c} ADF \\ (step) \end{array} \\
\begin{array}{c} BDF \\ (slot) \end{array} & 2 & 2 \\
\begin{array}{c} ADF \\ (hole) \end{array} & 2 & 2
\end{bmatrix}
\begin{bmatrix}
 & \begin{array}{c} BDF \\ (pocket) \end{array} & \begin{array}{c} ADF \\ (slot) \end{array} \\
\begin{array}{c} BDF \\ (slot) \end{array} & 2 & 2 \\
\begin{array}{c} ADF \\ (hole) \end{array} & 2 & 2
\end{bmatrix}
$$

Minimize Relaxations

- Assign each adj arc of $v_{i,k}$ to an adj arc of $v_{j,l}$ such that the overall cost (relaxation) is reduced
- NP-Complete problem
- Use greedy approach
  - Find the element $e_{x,y}$, with minimum cost in the matrix
  - Assign arc in row x to the arc in column y
  - Delete the assigned row and column and reiterate
- Running time of the greedy algorithm is $O(n^3)$

Greedy Algorithm - Example

- Consider the assignment of pocket to slot

$$\begin{bmatrix} & BDF(slot) & BDF(step) \\ AAE(slot) & 4 & 5 \\ ADF(slot) & 1 & 2 \end{bmatrix}$$

- Lowest element is 1, assigned to element $e_{2,1}$.
- Assigns arc ADF (slot) to arc BDF (slot)
- Remove the row and column

$$\begin{bmatrix} & BDF(step) \\ AAE(slot) & 5 \end{bmatrix}$$

Total Cost Matrix

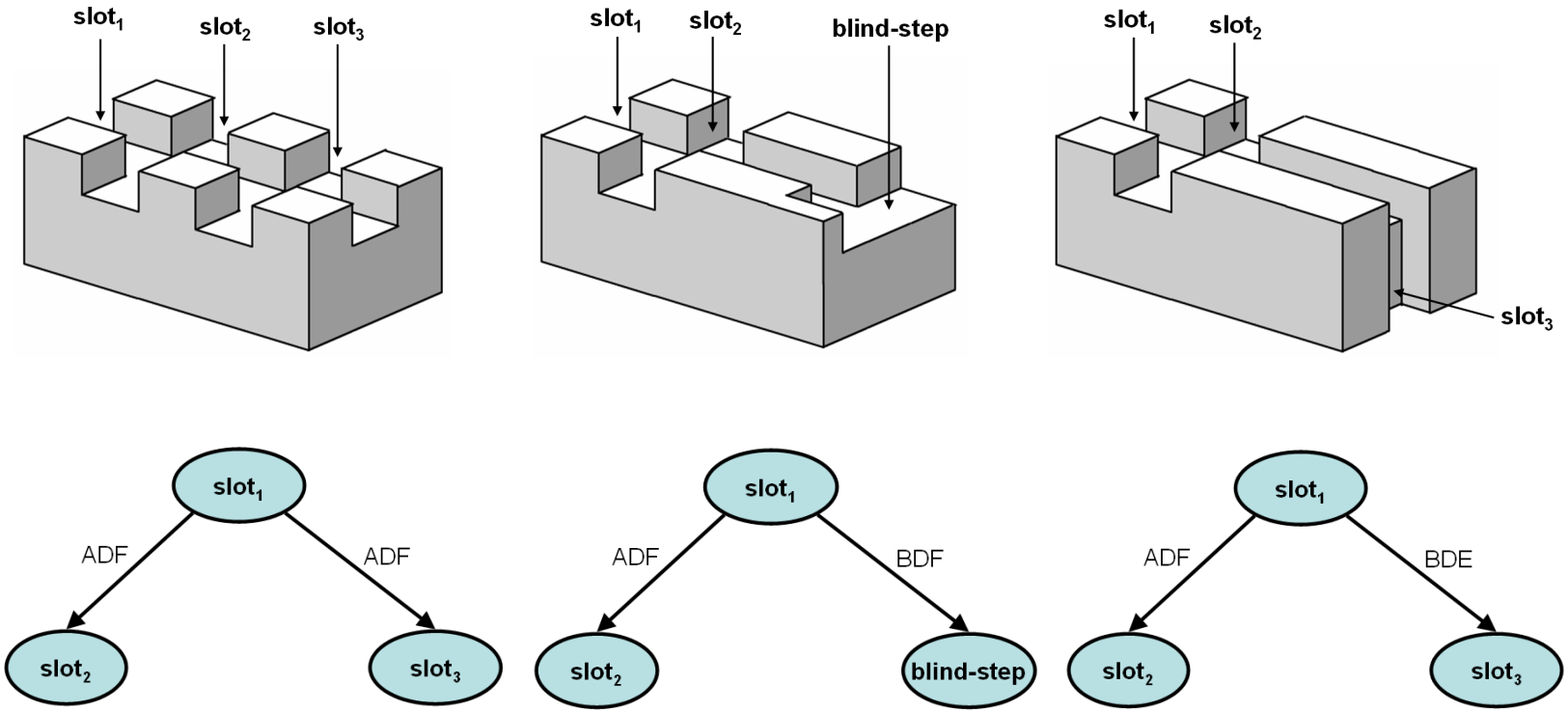
- Arc cost matrix after applying greedy algorithm

$$\begin{bmatrix} & pocket & slot & step \\ hole & 6 & 7 & 7 \\ slot & 6 & 7 & 7 \\ step & 3 & 4 & 4 \end{bmatrix}$$

- Total Cost Matrix = Minimized Arc Cost Matric + Node Cost Matrix

$$\begin{bmatrix} & pocket & slot & step \\ hole & 7 & 8 & 9 \\ slot & 8 & 7 & 8 \\ step & 6 & 5 & 4 \end{bmatrix}$$

Alignment Cost



- Total Cost for $oldPart_1 = 2 + 1 = 3$
- Total Cost for $oldPart_2 = 0 + 2 = 2$

Alignment Cost (cont.)

- Algorithm
  1. For each feature f_i in the new part
  2. Align the access direction of f_i and it's matched feature
  3. For each 90° rotation of new part about access dir. of f_i,
  4. Count the number of mismatched access directions
  5. Choose configuration that provides least # of mismatchs
- Cost of 1 mismatch is 3
- Previous Example
  - Total Cost for $oldPart_1 = 2 + 1 + 0 = 3$
  - Total Cost for $oldPart_2 = 0 + 2 + 3 = 5$

Total Cost

- Apply Greedy approach to Total Cost Matrix

$$\begin{bmatrix} & pocket & slot & step \\ hole & 7 & 8 & 9 \\ slot & 8 & 7 & 8 \\ step & 6 & 5 & 4 \end{bmatrix}$$

- Minimum cost is 4, and step is assigned to step

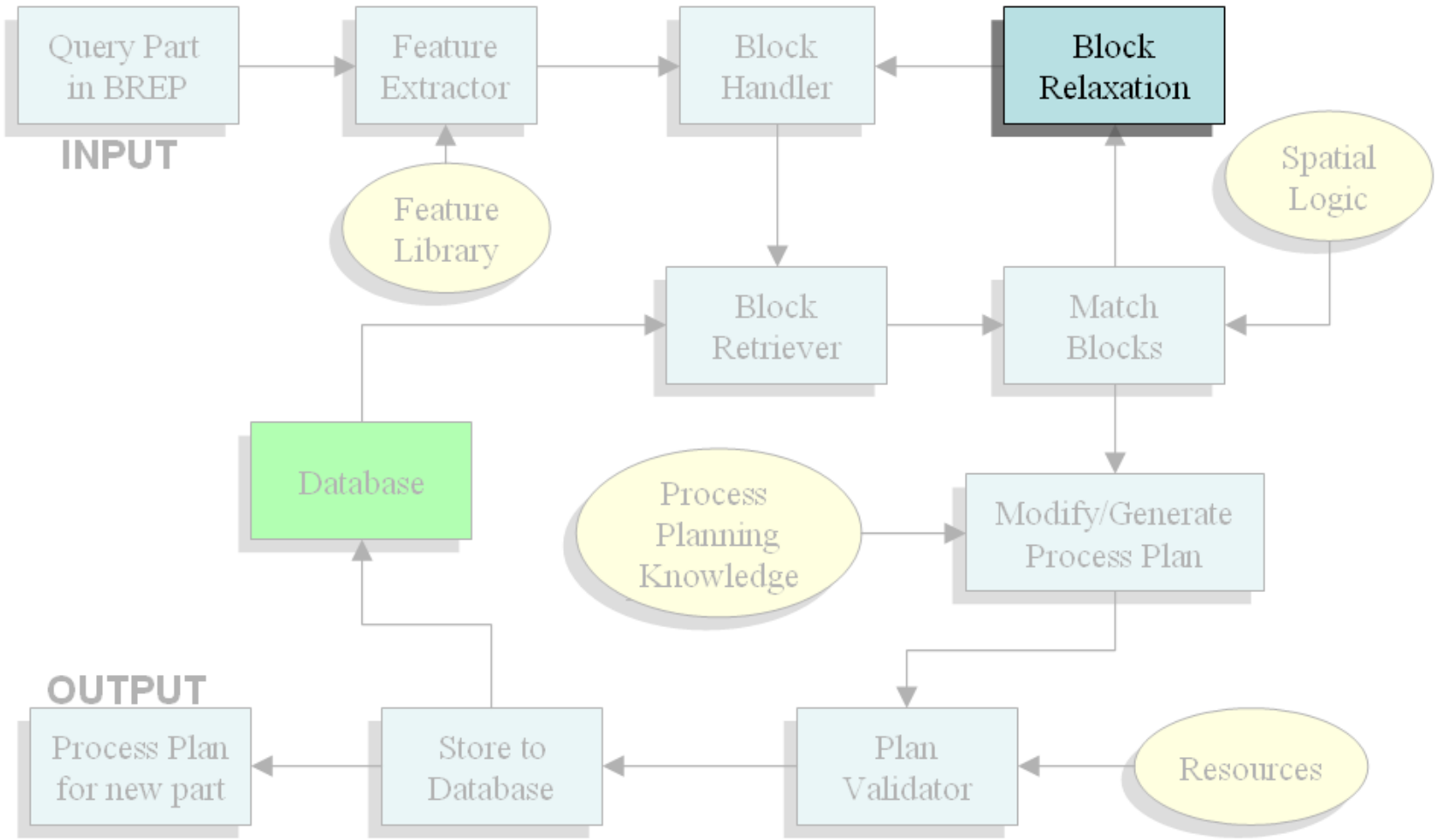
$$\begin{bmatrix} & pocket & slot \\ hole & 7 & 8 \\ slot & 8 & 7 \end{bmatrix}$$

- Arbitrarily select pocket, hole match (cost 7) as minimum
- Finally cost of slot, slot match = 7
- Total cost of all matches = 4 + 7 + 7 = 18

Run Time Analysis

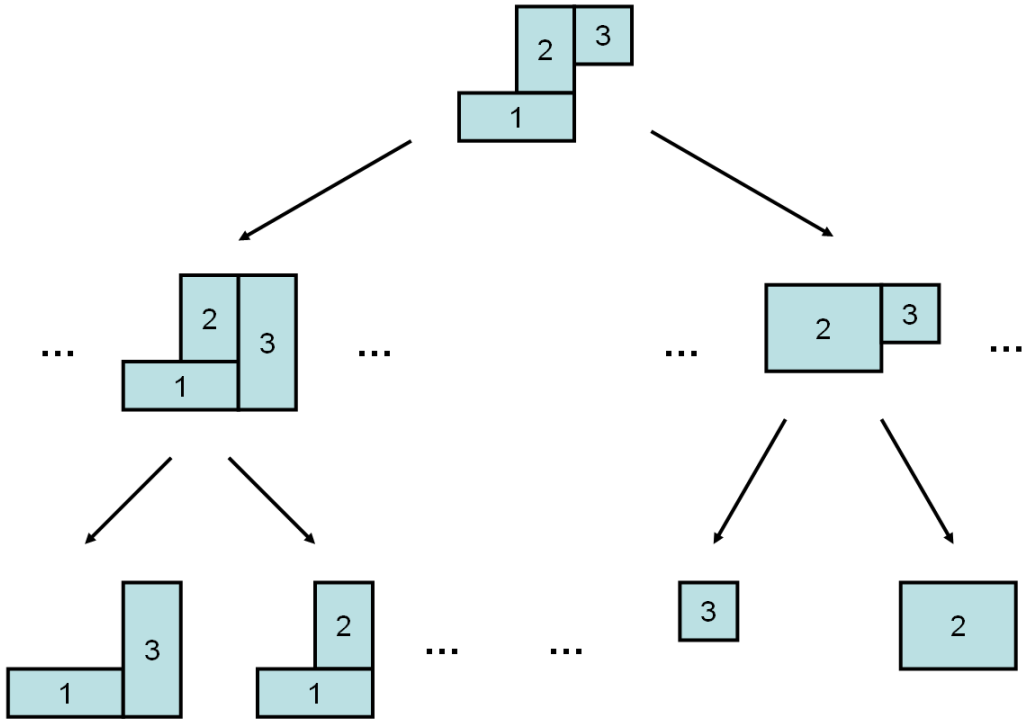
```
1:  procedure MatchBlocks(p, b, P, B)
2:      for (i = 0; i < number_of_blocks_in(b); i++) do
3:          OldBlocks[i] = BlockRetriever (b[i], P, B)
4:      end for
5:      for (i = 0; i < number_of_blocks_in(b); i++) do
6:          for (j = 0; j < number_of_blocks_in(OldBlocks[i]); j++) do
7:              CostMat = CreateCostMatrix (b[i], OldBlocks[i][j])
8:              for (k = 0; k < number_of_rows_in(CostMat); k++) do
9:                  for (l = 0; l < number_of_columns_in(CostMat); l++) do
10:                     ArcMat = MakeArcMat (b[i], CostMat[k][l], OldBlocks[i][j])
11:                     BestArcCost[k][l] = GreedyAlgorithm (ArcMat)
12:                 end for
13:             end for
14:             TotalCostMat = AddMat (CostMat, BestArcCost)
15:             AccessDirCost = FindAccessDirCost (b[i], OldBlocks[i][j])
16:             TotalCost[i][j] = GreedyAlgorithm (TotalCostMat) + AccessDirCost
17:         end for
18:     end for
19: end procedure
```

Block Relaxation



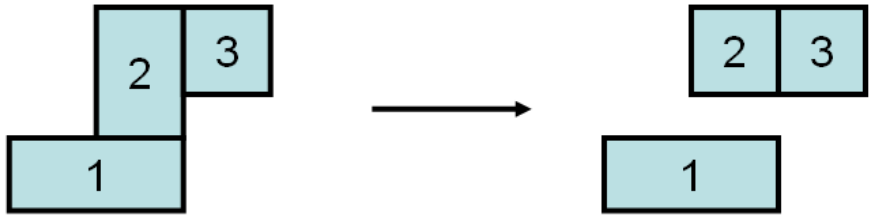
Block Relaxation (cont.)

- Move face of a feature by 1 qualitative step = 1 relaxation
- A qualitative step is the minimum distace a face can be moved to change the number of interactions



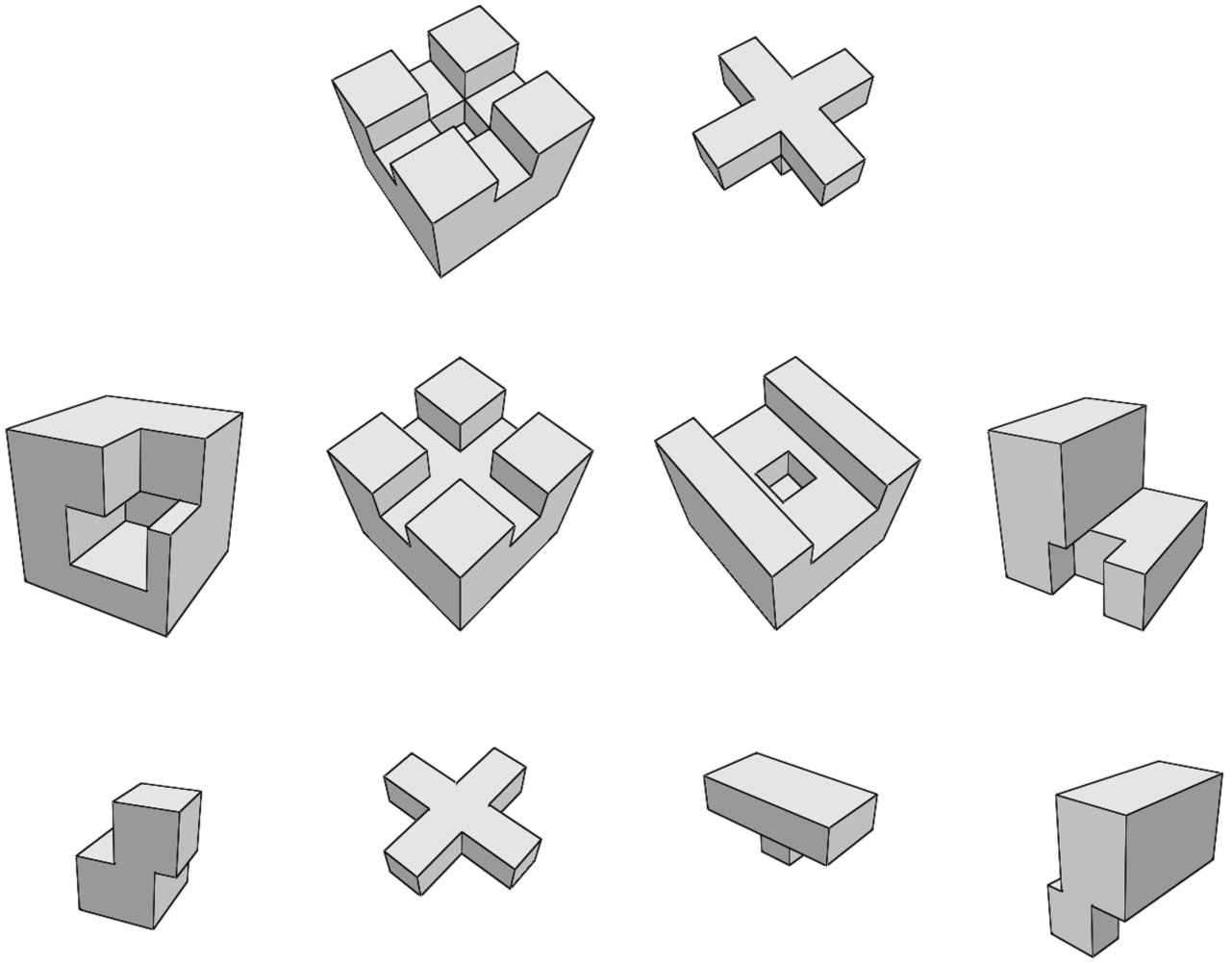
Block Relaxation (cont.)

- Relaxations not allowed to break the block into two or more blocks



- Actual implementation uses 3D relaxations
- The cost of losing 1 interaction is 9
- The cost of losing 1 feature is considered 1
- Values can be changed by the user
- Cost added to total cost before sorting is performed

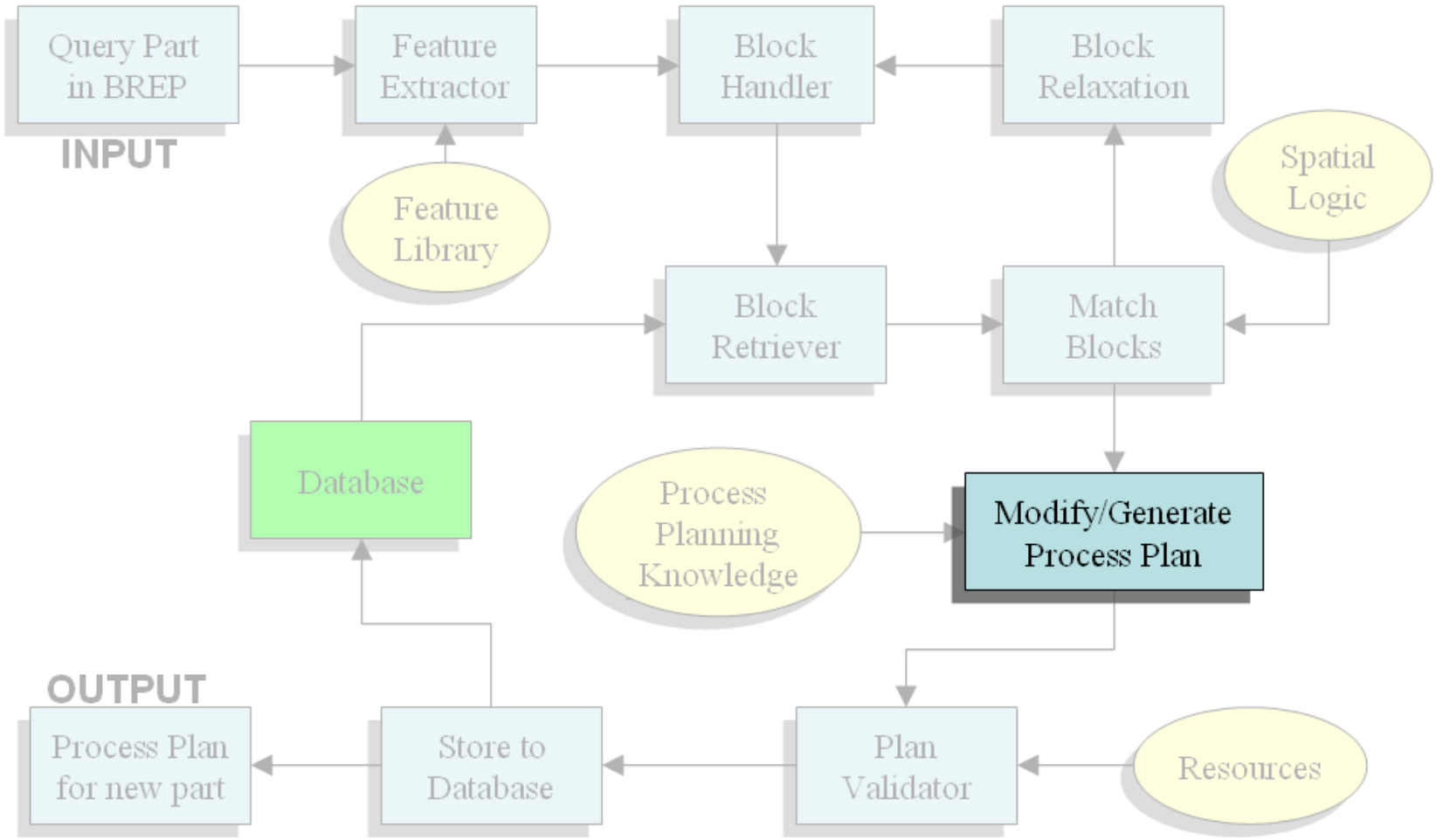
Example



Example (cont.)

- No objects found with 0 relaxations
- All four returned with 1 relaxation
- Calculated cost for each block
  - $part_1$: Cost 51
  - $part_2$: Cost 39
  - $part_3$: Cost 41
  - $part_4$: Cost 53
- Sorted order is $part_2$, $part_3$, $part_1$, $part_4$

Modify/Generate Process Plans



Variant Process Planning

- Reuse old process plans performed in hierarchical manner
- Plans refined at each level by enforcing stronger constraints

Level	Properties of Feature Used	Examples of Processes
1	Hardness	-
2	Width and height of feature	End milling, planing
3	Tolerance	Fine end milling, fine shaping

Variant Process Planning

```
name:                part2
part_id:             1
hardness:            180
blocks:              1

process2:
    block_id:        1
    feature_id:      2
    type:            slot
    length:          60
    width:           20
    height:          20
    tolerance:       0.1
    minX:            -20
    minY:            10
    minZ:            -5
    maxX:            20
    maxY:            20
    maxZ:            5
    process:         select tool1
                     milling
                     select tool2
                     end milling
```

```
name:                newpart
part_id:             50
hardness:            185
blocks:              1

process1:
    block_id:        1
    feature_id:      1
    type:            blind-slot
    length:          60
    width:           20
    height:          20
    tolerance:       0.001
    minX:            -20
    minY:            10
    minZ:            -5
    maxX:            20
    maxY:            25
    maxZ:            5
```

Generative Process Planning

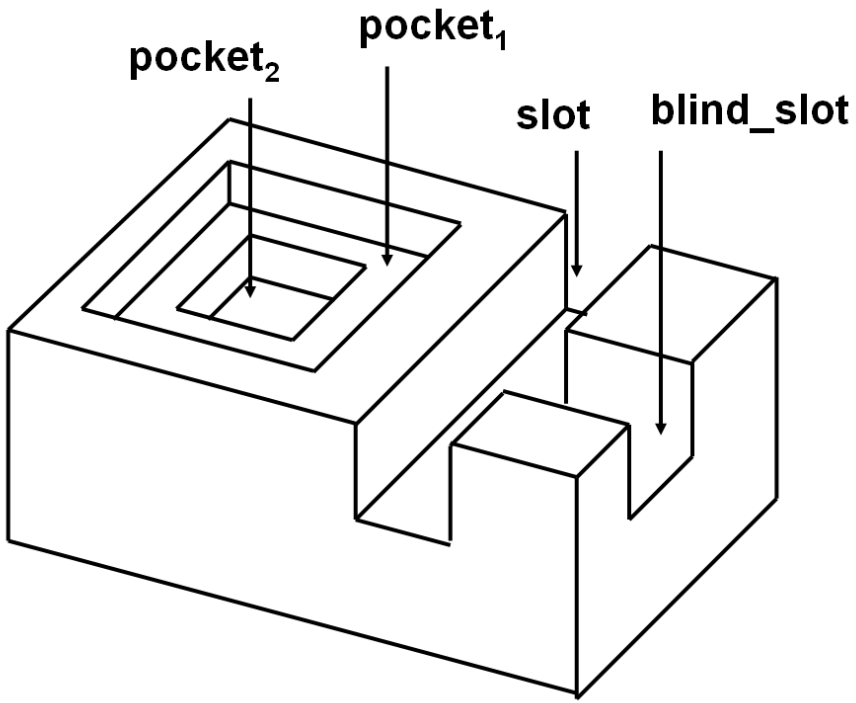
- Synthesize process information to create process plan
- Rule based system: Production rules expressed as if-then rules
- Mutliple processes may be required

```
if diameter $\leq$ 0.5 in.:
    if true position $\geq$ 0.01:
        if tolerance $>$ 0.01:
            Drill hole
        else if tolerence $\leq$ 0.01:
            Drill and ream
    else if true position $<$ 0.01:
        if tolerance $\leq$ 0.01:
            Drill and finish bore
        else if tolerance $\leq$ 0.002
            Drill, semi-finish bore and finish bore
```

Precedence Constraints

- Process plan operations cannot be performed in any order
- Geometrical constraints require that some processes be performed before others
- Group together a set of features with the same access direction
- The features within each group are again grouped based on interacting features
- Within each of these groups the features are sorted based on the top face
- Groups can be processed in any order

Precedence Constraints (cont.)



- Group 1: $\{pocket_1, pocket_2\}$
- Group 2: $\{slot, blindslot\}$

Valid Sequences

$\{pocket_1, pocket_2, slot, blindslot\}$

$\{pocket_1, slot, pocket_2, blindslot\}$

$\{pocket_1, slot, blindslot, pocket_2\}$

$\{slot, pocket_1, blindslot, pocket_2\}$

$\{slot, blindslot, pocket_1, pocket_2\}$

$\{pocket_1, pocket_2, blindslot, slot\}$

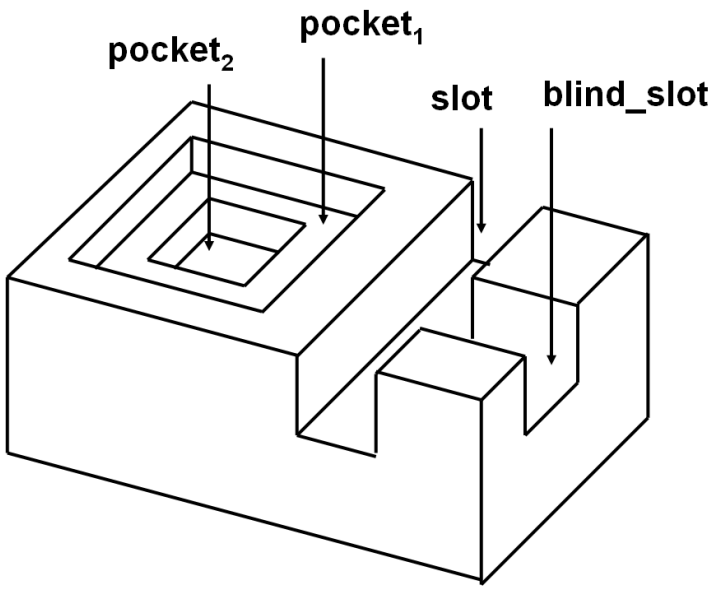
$\{pocket_1, blindslot, pocket_2, slot\}$

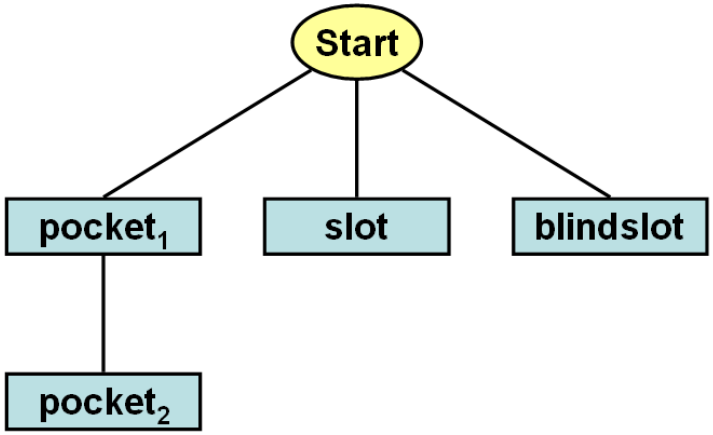
$\{pocket_1, blindslot, slot, pocket_2\}$

$\{blindslot, pocket_1, slot, pocket_2\}$

$\{blindslot, slot, pocket_1, pocket_2\}$

Precedence Constraint Graph





- Group 1: $\{pocket_1, pocket_2\}$
- Group 2: $\{slot, blindslot\}$

Plan Merging

- Optimal sequencing
- NP-Hard
- We use hierarchical plan merging
  - Fixture level optimization
  - Tooling level optimization

Fixture Level Optimization

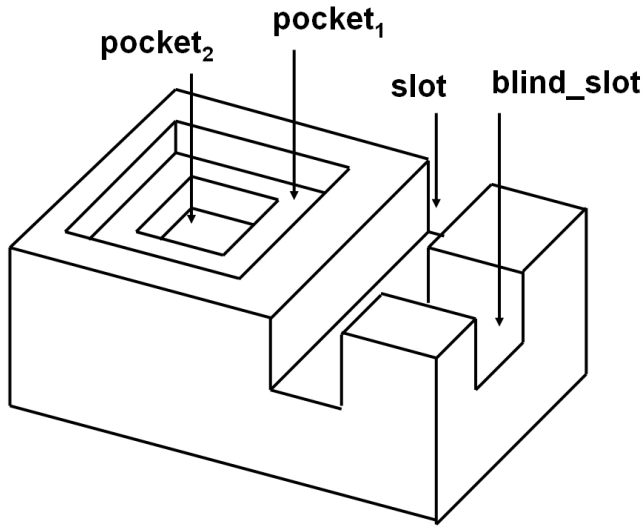
- Greedy approach: select fixture with most # of features
- Remove all selected features and reapply greedy method
- Continue until no features are left
- Example

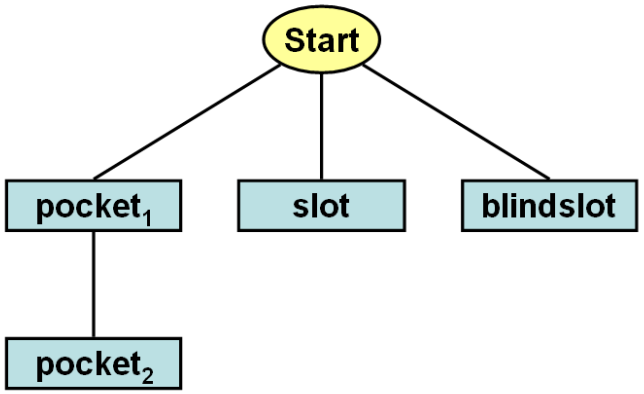
$\{\}$	$\{F_1(x,y),\ F_2(x,-y),\ F_3(y,z),\ F_4(x,y),\ F_5(x,y),\ F_6(x,-y)\}$
$\{< x,y >\}$	$\{F_2(x,-y),\ F_3(y,z),\ F_6(x,-y)\}$
$\{< x,y >, < x,-y >\}$	$\{F_3(y,z)\}$
$\{< x,y >, < x,-y >, < y,z >\}$	$\{\}$

Tool Level Optimization

- Choose tool with most number of operations in first level of precedence graph
- Remove the operations and repeat this procedure for the rest of the processes in first level
- The above two steps are applied to each level

Tool Level Optimization (cont.)





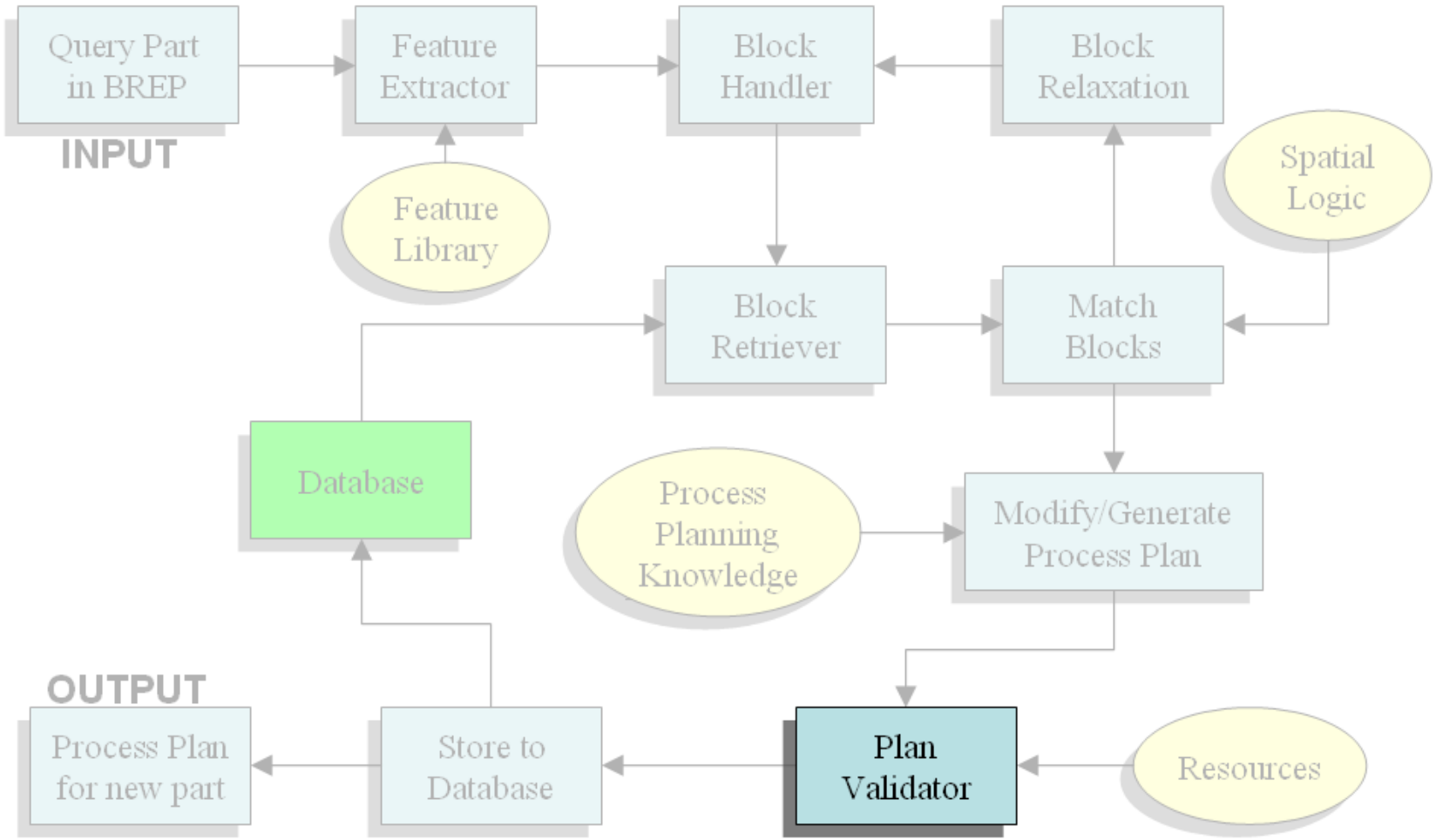

```
name:                pocket1
process:             select tool1
                     mill
                     select tool2
                     end mill

name:                pocket2
process:             select tool1
                     mill
                     select tool2
                     end mill

name:                slot
process:             select tool3
                     mill
                     select tool2
                     end mill

name:                blindslot
process:             select tool3
                     mill
                     select tool2
                     end mill
```

Plan Validator

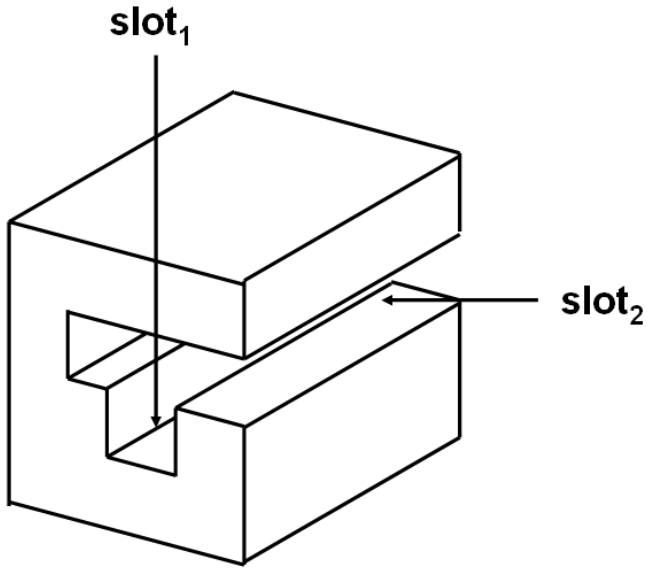


Plan Validation (cont.)

- Some of validations that can be performed are
  - Shape of finished part
  - feature interaction
  - feature interference
  - feature manufacturability
  - Flexibility for SFC etc
- We perform
  - Shape validation
  - Tool accessibility validation
- Plugable system: users can plug in new validation methods

Plan Validation (cont.)

- Shape Validation
  - Over-cut
  - Under-cut
  - Simulate using
    - feed
    - feed rate
    - cutting speed
    - depth of cut
- Tool accessibility validation
- Produces a detailed error report: error type, cause of error, where it occured



Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Conclusions

No conclusions yet

Outline

- Introduction and Motivation
- Related Work
- Importance
- Contributions
- Problem Formulation and Statement
- Solution Approach
- Details
- Conclusions
- Future Plans

Plan of Work

- Debug and test variant process planner
- Implement a Plan validator

Additions

- Block Alignment
- Generative process planner
- Variant process planner

QUESTIONS