Information Management in Production Systems

CAPP

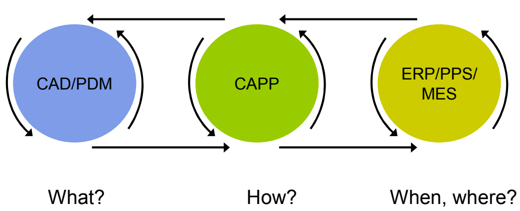
Computer-Aided Process Planning

József Váncza
MTA SZTAKI
BME GIT
vancza@sztaki.hu

Overview

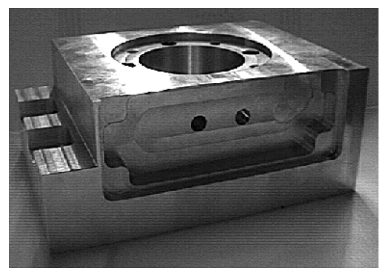
- **The role of CAPP in Production Informatics**
- **Process Planning (without computers)**
  - Problem statement
  - Example domains
    - Machining of prismatic parts
    - Sheet metal bending
  - Key concepts: features and setups
- **Computer-aided methods**
  - Approaches
    - Variant method
    - Generative CAPP
  - Representation issues
  - CAPP as the weakest link in manufacturing automation
- **A generic CAPP model**
  - Constraint-based planning
  - CAD-CAPP integration
- **Summary**

The role of CAPP in Production Informatics



Question

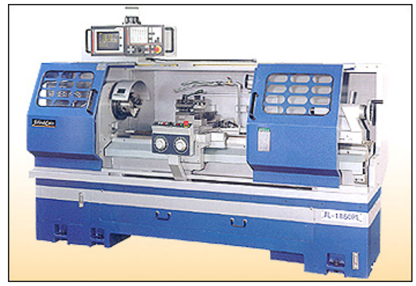
- **How to produce this part?**



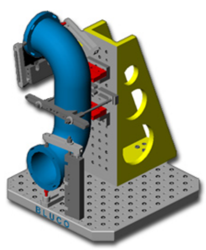
Answer: by machining operations …

… with specific tools,…

… on (some) machine(s), …

… in appropriate fixture(s)

Result of process planning: routing

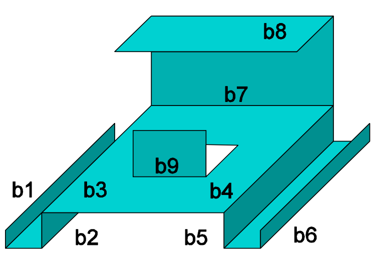
OPERATION SHEET

Part No. *CLP023456-4-92-028* Material *steel 1040*
Part Name *Widget*
Orig. *H.Jack* Changes
Checked *W.H.ElMaraghy* Approved *D.Corrin*

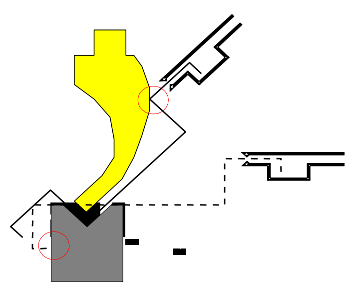
No.	Operation	Machine	Setup	Time (hrs.)
0010	*Saw off and slug 1.75 dia. hole*	*Dept 12 Saw 3*		*.3*
0015	*R'Turn 6.00 dia. stock to 5.210/5.190* *R'Bore 1.75 dia to 2.00* *F'Bore 2.00 to 2.005/2.015*	*G.E. Turn Lathe*	*Hold in counter centrifugal Chuck*	*1.2*
0025	*Deburr all edges*			*5 mins.*

Question (2)

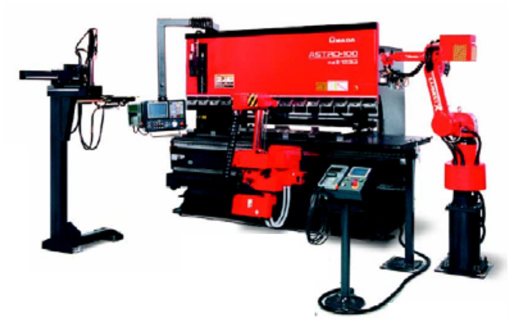
- How to fold this part?



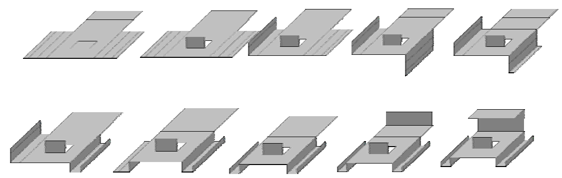
Answer: by bending operations …



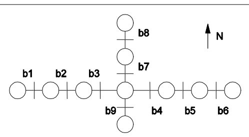
… with appropriate tools, grippers and machine

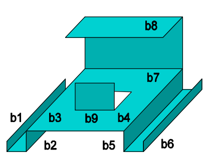


Result: bending sequence …



… with tool and gripping specifications





op	tprof	tlen	ori
b9	leftgn	10	N
b2	leftgn	50	E
b4	leftgn	50	E
b6	leftgn	50	E
b5	leftgn	50	E
b3	leftgn	50	E
b1	leftgn	50	E
b8	leftgn	50	S
b7	leftgn	50	S

Process Planning

- **What is planning in general?**
  - Given
    - Initial state and a goal state of the world,
    - Set of actions that can change the state of the world
      - Preconditions and effects
    - Optimization criteria
  - Find
    - The best sequence of actions that transform the world from the initial state to the goal state.
- **Worlds (domains) of Process Planning**
  - Manufacturing
    - Machining of rotational and prismatic parts
    - Sheet metal bending
  - Assembly/Disassembly
  - Inspection planning

Process Planning: problem statement

- **Given:**
  - Specification of the blank and the finished part
    - Geometry -- dimensions and tolerances
    - Materials
    - Surface finishes
  - Description of available resources
    - Machines
    - Fixtures
    - Tools
  - Relevant technological knowledge
  - Criteria for best solutions
- **Find:**
  - Executable,
  - If possible, (close-to) optimal plans

Phases of Process Planning

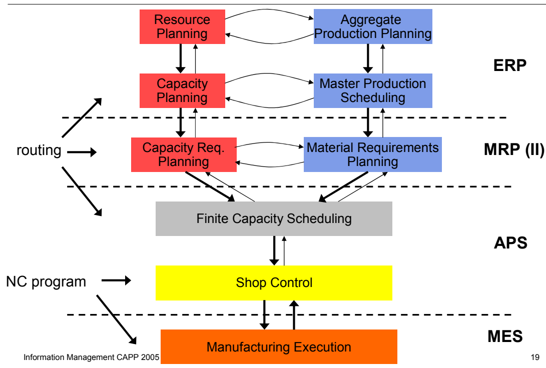
- **Part and process analysis**
  - Technological interpretation of design specifications
  - Determination of the entities to be produced – so-called *features*
  - Production alternatives

  → **feature-based part model**
- **High-level (macro) planning**
  - Selection resources
  - Specifying the actual operators
  - Determining the sequence of operations
  - Determining *setups*

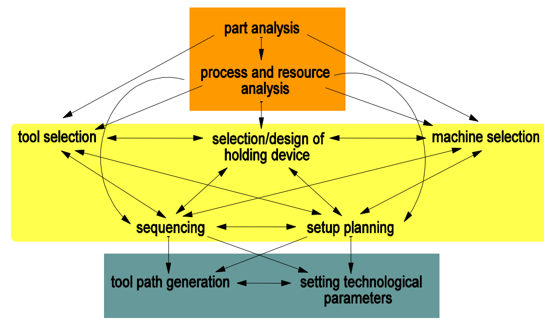
  → **routing**
- **Low-level (micro) planning**
  - Determining the paths for moving the tools
  - Parameter for operations (cutting speed, etc.)

  → **NC program**

Where are the results used?



Planning tasks

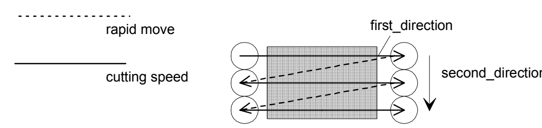


Additional task: fixture design

- **Of modular kits**
- **Configuration design problem**
  - Free tool paths
  - No collision

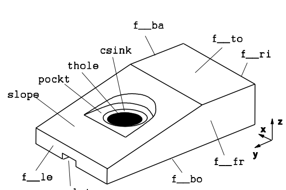
Micro planning

- **Setting technological parameters**
  - Speed
  - Angle
  - Feed-rate
- **Path planning**



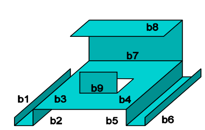
Example: machining

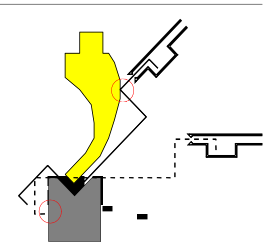
- **Operations**
  - Produce feature states
- **Resources**
  - Tools
  - Setups
  - Machine tools
  - →Plan alternatives
- **Criteria**
  - Minimal setup changes
  - Minimal tool changes



- **Domain knowledge**
  - Sequencing requirements
  - Resource usage requirements (strict tolerances → common setup)
  - Good practice
    - *Make first the large surfaces!*

Example: bending

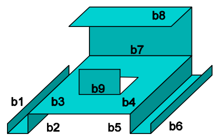




- **Operations**
  - Make bends
- **Resources**
  - Tools (length, profile)
  - Holdings (orientations)
  - One resource → several bends
- **Criteria**
  - Minimal changeovers (tool, holding)
  - Maximal balance

Example: bending (2)

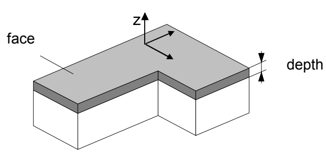
- **Problem**
  - Avoid collisions
  - Conflicting criteria
  - Local bends → global changes
- **Typical pieces of advice**
  - *Make outside bends first to avoid "rolling up" the part.*
  - *Produce channels (b1-b2-b3) from inside toward outside.*
  - *Postpone tall flanges as far as possible.*
  - *Make bends determining the shape of the part as late as possible.*
  - *Make strictly toleranced bends without changing resources.*
  - *After making b3 and b4, b7 can be made with exact length tool only.*
- **Properties**
  - Conditional rules
  - Contradictions between rules

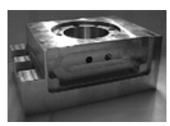


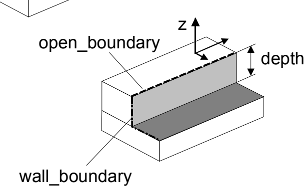
Manufacturing features

- **"Micro" worlds of planning**
  - Atomic tasks and solution patterns
  - Mediating between design and production
  - Mapping of local geometry, tolerances to manufacturing processes
  - Engineering interpretation of design drawings
- **Design vs. manufacturing features**
- **Ambiguity**
  - *"One man's hole is another's bore."*
- **Feature relations**
  - Workpiece as a set of interrelated features
  - Multiple interpretations
- **Feature interactions**

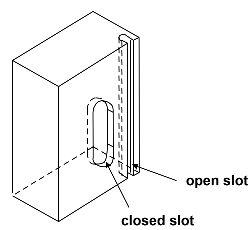
Features in machining: face and step

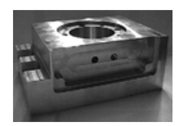


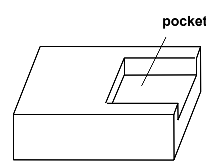




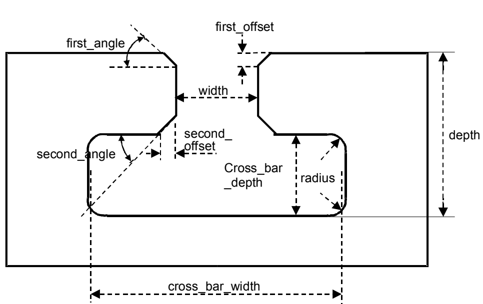
Pocket and slot



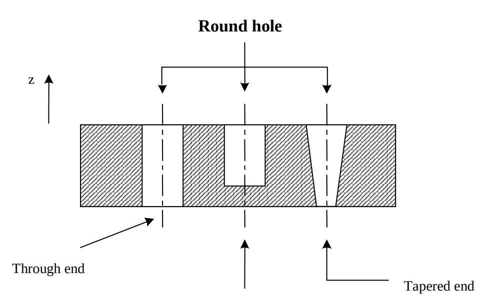




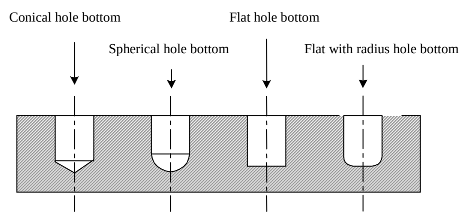
T-slot



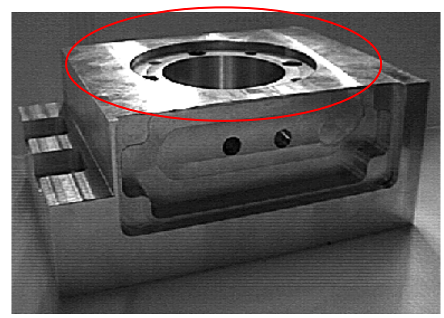
Holes



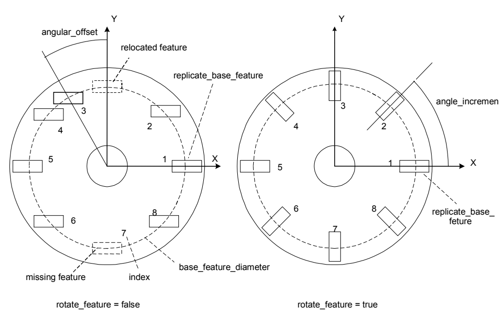
Holes: bottom conditions



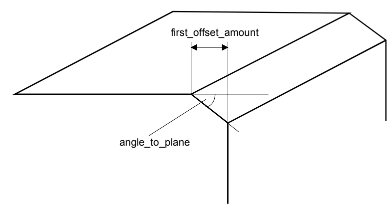
Compound features



Feature patterns

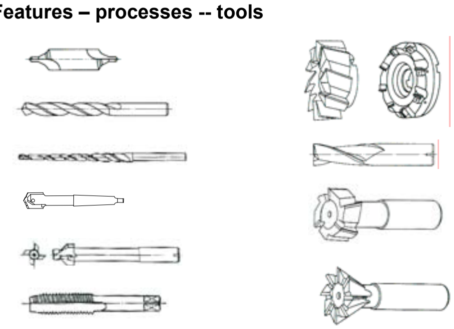


Subfeatures: e.g. chamfer



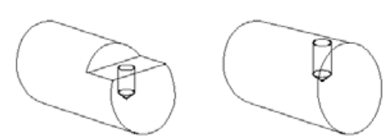
Tools

- Features – processes -- tools



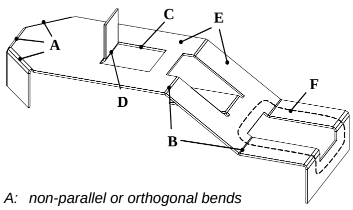
Interaction of features

- Crossing holes
- Strict tolerances
- Holes on non-orthogonal surfaces



- Much more …

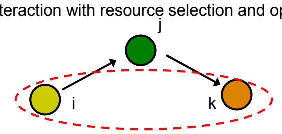
Features in bending



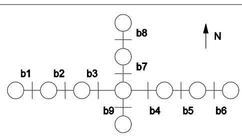
A: non-parallel or orthogonal bends
B: non 90°-bends
C: internal contour
D: internal bend
E: double connected flanges
F: closed loops of flanges

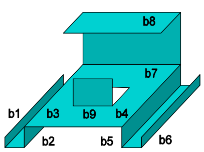
Setups

- **Set of operations executed without changing a basic resource**
  - Machining – same fixturing and part orientation
  - Bending – same tool
- **Rationale**
  - Tolerance requirements
  - Cost
- **Setup planning**
  - Grouping operations
  - Strong interaction with resource selection and operation sequencing



Setup example: bending





op	tprof	tlen	ori
b9	leftgn	10	N
b2	leftgn	50	E
b4	leftgn	50	E
b6	leftgn	50	E
b5	leftgn	50	E
b3	leftgn	50	E
b1	leftgn	50	E
b8	leftgn	50	S
b7	leftgn	50	S

Computer-Aided Process Planning

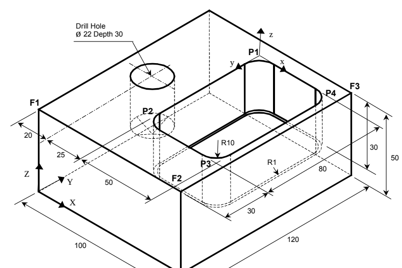
- **Aim: supporting engineering problem solving**
  - Efficient use of deficient knowledge
  - Finding a match between
    - available domain knowledge, its
    - appropriate representation, and
    - efficient utilization
- **Representation issues**
- **Solution: reasoning and search**
  - Even in face of *inconsistency, incompleteness* and
  - In the presence of *complexity*
- **Optimization**
  - Finding the best solutions
- **Integrating different sources of knowledge**
  - Interactive modeling and solution generation

Variant approach to CAPP

- **Similarity-based problem solving**
- **From the pre-computer era**
- **Method**
  - Previously prepared process plans are stored in a database
    - Group technology
    - Complex parts
    - Complicated coding and classification system
    - Indexing
  - Retrieve and adapt – manually
- **Problems**
  - Considers only workpiece similarity
  - Optimization cannot be performed
  - Conservative

Generative CAPP

- **How to represent what we know?**
- **How to exploit the represented knowledge efficiently?**



Representation issues

- **Part model**
  - General attributes + features + feature relations
- **Features**
  - Geometric attributes + technological parameters
  - Primary/sub; compound; pattern
  - Sequence of feature states (raw, fine, superfine, …)
- **Resources**
  - Machines
  - Setups
  - Tools
- **Machining operations**
  - Produce feature states
  - With resources specified
- **Process plans**
  - Sequence/network of operations

Representing engineering knowledge

- **Interpretation of the design**
  - Part analysis
  - Feature recognition
- **Use of resources**
  - Mapping feature states and operations
  - Determining admissible resource combinations
- **Elaboration of alternatives**
- **Partial ordering of operations**
  - Generating precedence relations between operations
  - Prohibiting change of resource

Representation techniques (AI)

- **Conceptual hierarchies**
  - Class structure
    - Classes and instances
    - Inheritance
  - So-called frame systems
  - Appropriate for describing objects
    - Features, tools, setups, machines
- **Rule-based representation methods**
  - Condition-action pairs
    - Condition patterns
    - Several applications of the same rule
  - Intuitive for engineers
  - For capturing (fragments) of
- **Both methods rooted in logic**

Representation of features (ISO)

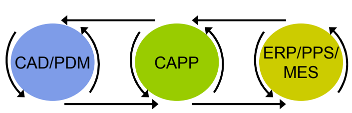
```
ENTITY workpiece;
    its_id:                identifier;
    its_material:          OPTIONAL material;
    global_tolerance:      OPTIONAL shape_tolerance;
    its_rawpiece:          OPTIONAL workpiece;
    its_geometry:          OPTIONAL advanced_brep_shape_representation;
    its_bounding_geometry: OPTIONAL bounding_geometry_select;
    clamping_positions:    SET [0:?] OF cartesian_point;
END_ENTITY;

ENTITY machining_feature
    ABSTRACT SUPERTYPE OF (ONEOF(planar_face, pocket, slot, step,
         round_hole, generic_feature, general_outside_profile, compound_feature))
    SUBTYPE OF (two5D_manufacturing_feature);
    depth:                    toleranced_length_measure;
    planar_radius:            OPTIONAL toleranced_length_measure;
    orthogonal_radius:        OPTIONAL toleranced_length_measure;
    surface_roughness_bottom: OPTIONAL property_parameter;
    surface_roughness_sides:  OPTIONAL property_parameter;
END_ENTITY;
```


Why is CAPP so hard?

- **Contradictions**
  - Conflicts of local pieces of advice in the global context of a particular planning problem

  → Logic cannot help
- **Complexity**
  - Of large constrained optimization problems

  → Search cannot help
- **Failure of pure logical approaches**
- **Features divide but not conquer the problem**
- **CAPP as the weakest link in automation**



A generic CAPP model

- **Main requirements**
  - For all kinds of relevant domain knowledge
  - Applicable on several fields
  - With engineer-friendly representation
  - Coming from either human or machine source
- **Efficient plan generation and optimization**
  - Consolidation of conflicting pieces of knowledge
  - Multi-criteria optimization
  - Handling incomplete models → CAD integration
- **Decision support**
  - Interaction
  - Mixed-initiative problem solving
  - Supporting both variant and generative strategies

Constraint programming

- **Variables**
  $X=\{x_1,\ldots,x_n\}$
- **Finite domains**
  - For all x_i the set of possible values $\mathbf{D}_i$
- **Constraints**
  - Logical relations on variables that restrict the possible values.
  - Properties
    - Express partial information
    - Not directed
    - Declarative
    - Additive
    - Common variables → bind together constraints
- **Solution**
  - A value for all x_i such that all the constraints are satisfied
  - Several solutions → optimization

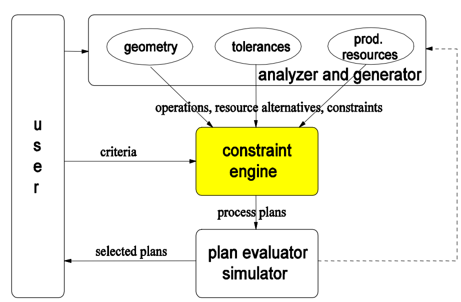
Constraint-based CAPP model

- **Variables, domains and constraints**
- **Operations** `op_i, i ∈ 1,…,N`
  - Position of operations `pos[op_i]: [1,…,N]`
  - Each executed only once `all pos[op_i] different`
- **Resources**
  - Alternatives for each operation
    - Part orientation `orient[op_i] : ORIENT_i ⊆ O`
    - Tool `tool[op_i] : T_i ⊆ T`
    - Tool length `tlen[op_i] : TLEN_i ⊆ TL`
    - Tool profile `tprof[op_i] : TPROF_i ⊆ TP`
  - Allowed combinations
- **Process plan**
  - Completely specified sequence of operations
  - Each operation with specific position and resources

Constraint-based model (2)

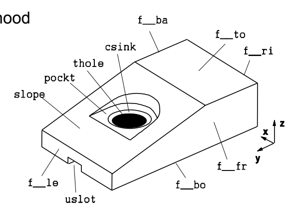
- **Constraints**
  - Express properties of executable plans
  - Represent *results* of engineering reasoning
  - Natural to understand, generate and modify
- **Model is generated from part and tool geometry**
  - By hand (machining)
  - Automatically (bending)
- **Optimization criteria**
  - Minimal changeover of resources
    - Setups in machining
    - Tool in bending

Constraint-based CAPP



Constraint types

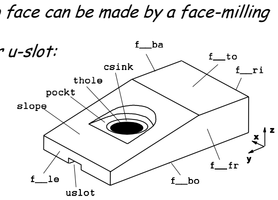
- **Ordering**
  - operation precedence
    - *Centering a hole must be done before drilling it.*
    - *Make large faces first.*
    - *If through-hole is made from the bottom face, make it before u-slot.*
  - operation neighborhood



Constraint types (2)

- **Resource assignment**
  - *In this orientation the top face can be made by a face-milling tool.*
  - *Resource combinations for u-slot:*

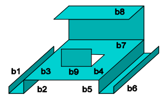
Operation	Tool	Holding
uslot	e_mill	+x−y−z
	s_mill	−y−x−z
		−x+y−z
		+y+x−z
		−z−y−x
		+y−z−x
		−z+y+x
		−y−z+x



- **Resource sharing (setup formation)**
  - Common resource
  - Common resource sequence
    - *Make final states of strictly tolerances features in one setup.*

Constraint types (3)

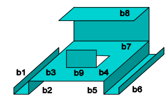
- **Ordering**
  - operation precedence
    - *- Make small tabs first.*
    - *- Make outside bends before inside bends.*
    - *- Make internal tabs first.*
    - *- Produce channels from inside toward outside.*



- **Resource assignment**
    - *- After bending 6 and 7, bends 2 and 3 need exact size tool.*
- **Resource sharing (setup formation)**
  - Common resource sequence
    - *- If possible, perform two collinear bends in one operation.*
    - *- Make parallel bends without changing the tool.*
    - *- Try to execute parallel bends of the same lengths immediately after each other, using the same resources.*

Hard and soft constraints

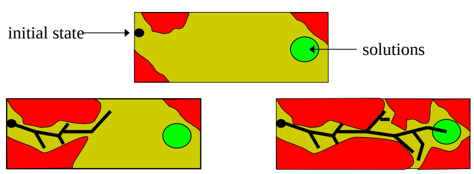
- **Hard constraints**
  - Always to be observed
- **Soft constraints**
  - Weight for each constraint
  - Total weight e_G and limit l_G limit for each *G* constraint type
  - $e_G \geq l_G$
  - Express properties of good plans
- **Conditional constraints**
  - Hard only
  - Bound to logical condition



If b3 and b4 is made before b7, b7 needs exact length tool.

```
(pos[b3] < pos[b7]) & (pos[b4] < pos[b7])
     =>  (tlen[b7] = tl50);
```

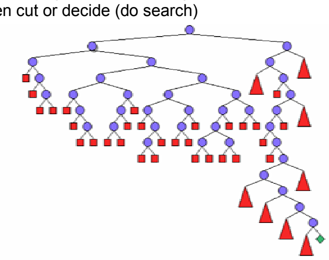

Solution: reasoning + search



- **Reasoning: constraint propagation**
  - Reducing the domains of variables
  - Proving infeasibility (empty domain)
  - Incomplete
- **Search**
  - When reasoning does not help more
  - Making a try with adding a new constraints C or its negation ¬C

Solution: example

- **Tree search**
- **At the nodes**
  - First propagate (do reasoning)
  - Then cut or decide (do search)



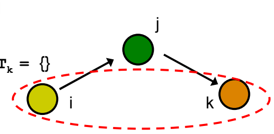
Solution of the CAPP model

- **Key issues**
  - Complexity
    - Sequencing and resource assignment decisions are *not* separated
    - Even relaxed problem is NP-complete
  - Inconsistencies: handling soft constraints
    - Interaction of sequencing and
    - Resource assignment decisions
- **Example**

```
pos[i] < pos[j] < pos[k]
commonOriSeq[i,k]
```

$\mathbf{ORIENT}_i \cap \mathbf{ORIENT}_j \cap \mathbf{ORIENT}_k = \{\}$

→ no solution exists



Solution of the CAPP model (2)

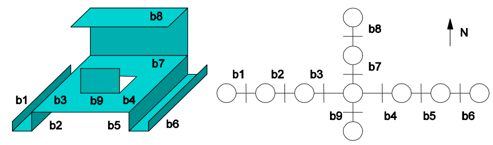
- **Requirements towards the solver**
  - Satisfy soft constraints above a threshold
    - Or: maximize accepted domain knowledge
  - Find the best solution
  - Work with short response time
- **Looking for the Pareto set of solutions**
  - P is a subset of all solutions S
  - All solutions in S/{P} are dominated by P
  - No solution in P is dominated by another solution in P
- **Several solution strategies**
  - First consolidate soft constraints, then optimize plan
    - Efficient, but not complete
  - Extensions of *branch-and-bound*
    - Recent solutions bound search
    - Complete but expensive
    - Soft constraints make it much better (*best practice*)

Geometry

- **At the bottom of plan correctness**
- **Constraints**
  - Capture only the results of geometric considerations
- **Issues**
  - Automatic generation of constraint-based models
  - Checking of (partial) solutions
  - Augmenting incomplete models during problem solving
- **Answers**
  - Narrowed scope -- bending (2,5D)
  - Geometric modell, preprocessor and evaluator
  - Solver working on “open” model

Geometric model in bending

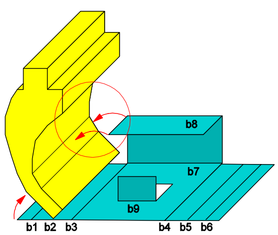
- **Part geometry**
  - Connectivity graph with pane and bend nodes
  - Rectangular solids on panes



- **Tool geometry**
  - Tool length
  - Front and back profile of punch
- **Holding: part orientation**
- **Machine: forbidden areas**

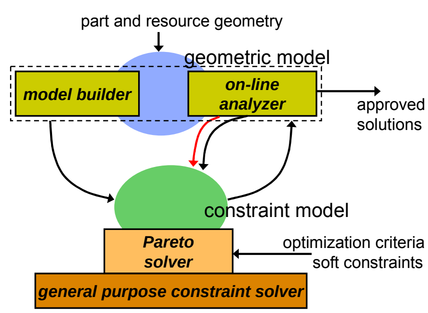
Collision test and constraint feedback

- **Collision test by simulating the bending process**
  - Each time one more operation has been specified
  - Sweep volume collision test
- **Reasons of failure**




```
(pos[b7] < pos[b2]) &
(pos[b8] < pos[b2]) &
(pos[b2] < pos[b3])
 =>
(tprof[b2] ≠ leftgn)
```


System structure



Implementation experiments (1)

- **General purpose CAPP system**
  - Domain language and compiler
  - Hard and soft constraint solver
- **Experiments**
  - Prismatic parts

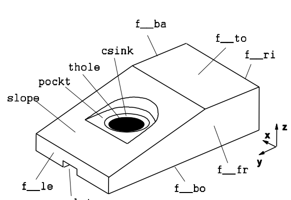
Problem

operations	13
setups	20
tools	7
hard const.	126
soft const.	10

Model and solver

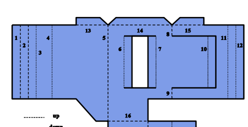
variables	10^5
propagators	10^6
size of the tree	10^4

all soft constraints satisfied
3 solutions in 80 sec

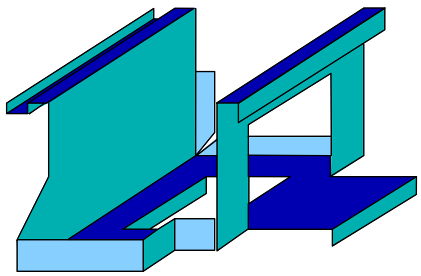


Implementation experiments (2)

- **CAPP system with design feedback**
  - Dedicated model builder and analyzer in 2.5D
  - Interactive Pareto solver
    - on top of ILOG Solver
- **Experiments**
  - Bending problems up to 18 bends
    - Response time: ~ 0.1 – 10 min
    - Best practice is useful to solve complex problems
  - Incomplete models
    - Different sets of initial hard constraints
    - No soft constraints
    - Let the geometric analyzer to detect necessary constraints
    - Learning from failure



Solution of the bending example (1)



Solution of the bending example (2)

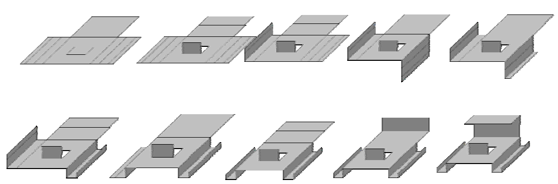
- **Experiments with incomplete models**

	Full	Slightly incom.	Highly incom.	Empty
Constraints				
Evident	28	28	28	28
Further initial	23	17	10	0
Feedback recognized	2	9	19	27
Solutions				
Found totally	7	14	24	32
Correct	5	5	5	5
Pareto optimal	2	2	2	2
Solution time (s)	16	46	337	1030

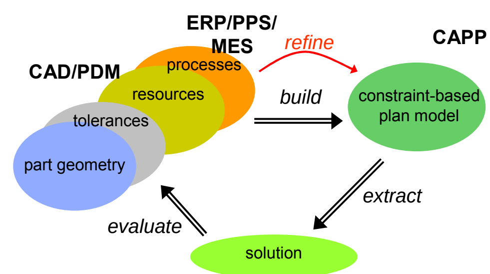
op	tprof	tlen	ori
b9	leftgn	10	N
b2	leftgn	50	E
b4	leftgn	50	E
b6	leftgn	50	E
b5	leftgn	50	E
b3	leftgn	50	E
b1	leftgn	50	E
b8	leftgn	50	S
b7	leftgn	50	S

Solution of the bending example (2)

- **One Pareto-optimal plan**
  - 1 tool change
  - 2 holding changes



Role of CAPP revisited



Conclusions

- **Constraint programming provides an efficient framework for CAPP**
  - Declarative domain description language
  - Model and algorithm separation
  - Open to mixed-initiative problem solving
- **No crisp borderline between hard and soft domain knowledge**
- **Complementary CAPP and CAD models**
  - Relieve modeling efforts in CAPP
  - Interaction via constraint feedback
- **Current issues**
  - How to make the solution process more efficient?
  - How to generalize?

* * *