Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-92-49

Heuristic Classifcation for Automated CAPP

Christoph Klauck, Ralf Legleitner,
Ansgar Bernardi

December 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Heuristic Classification for Automated CAPP

Christoph Klauck, Ralf Legleitner,
Ansgar Bernardi

DFKI-RR-92-49

A short version of this paper will be published in the Proceedings of the Conference 1963: Eleventh Applications of Artificial Intelligence 1993 (AAI-XI) - Knowledge-Based Systems in Aerospace and Industry, SPIE.

This work has been supported ba a grant from The Federal Ministry for Research and Technology (ITWM-8902 C4).

Heuristic Classification for Automated CAPP

Christoph Klauck, Ralf Legleitner and Ansgar Bernardi

German Research Center for Artificial Intelligence Inc.
ARC-TEC Project
Mailing address: P. O. Box 2080, D-6750 Kaiserslautern
Street address: Erwin-Schroedinger-Str., D-6750 Kaiserslautern
Telephone: ++49-631-205-3477, *FAX:* ++49-631-205-3210
E-mail: klauck@dfki.uni-kl.de

ABSTRACT

In order to create a process plan from a workpiece description, a human expert thinks in a special terminology with respect to the given workpiece. The steps of human thinking during the generation process of a process plan are following the principles of heuristic classification: First using feature recognition an abstraction process is realized yielding a high level (qualitative) description of the current workpiece in terms of features. To these features certain (more or less) abstract (partial) process plans – the so-called skeletal plans – are associated. In the refinement step these skeletal plans are merged together to one complete process plan.

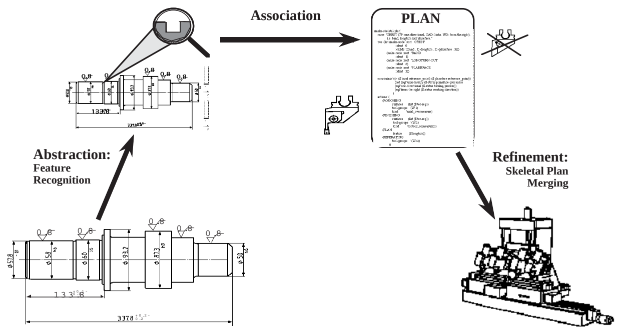


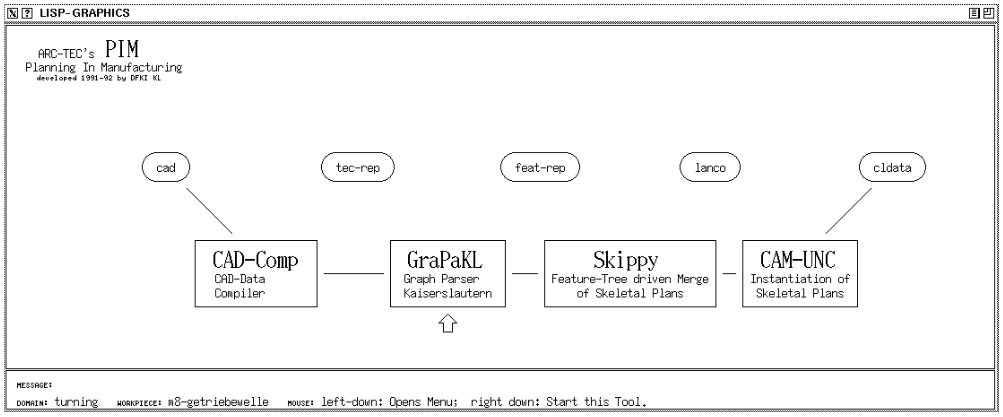
Figure 1: General model of process planning

In this paper we present a set of domain-oriented higher level representation formalisms for features and skeletal plans suitable for the modeling of this approach. When an expert's (process planner's) knowledge has been represented using these formalisms, the generation of a process plan can be achieved by heuristic classification. This is demonstrated in the CAPP-system PIM, which is currently implemented as a prototype.

1 MOTIVATION

Systems to support Computer Aided Process Planning (CAPP) encounter a particular problem which arises out of the structure of this domain: In order to create a successful process plan it is not sufficient to rely on theoretically founded rules – which normally can be coded into a program quite easily – but it is also necessary to consider particularities of the production environment and special experience of human process planners. The latter is very often difficult to formalize and consequently hard to be coded, the former may change radically between different working environments and leads to a high maintenance workload.

Under these circumstances the knowledge-based programming methodology may be used: If it is possible to define domain-oriented higher level representation languages facilitating the explicit representation of the knowledge in question and to provide adequate execution mechanisms for these languages the problem becomes treatable. Domain-oriented higher level representation languages lead to easier representation of the experience and environment-dependent knowledge. The execution mechanisms are more independent of the environment and probably useful in a broader domain, thereby reducing the general maintenance workload.


LISP-GRAPHICS
ARC-TEC's PIM
Planning In Manufacturing
developed 1991-92 by DFKI KL
cad
tec-rep
feat-rep
lanco
cldata
CAD-Comp
CAD-Data Compiler
GraPaKL
Graph Parser Kaiserslautern
Skippy
Feature-Tree driven Merge of Skeletal Plans
CAM-UNC
Instantiation of Skeletal Plans

```
MESSAGE:
DOMAIN: turning   WORKPIECE: m8-getriebewelle   MOUSE: left-down: Opens Menu;  right down: Start this Tool.
```

Figure 2: User interface of PIM: current state abstraction

In summary, knowledge-based programming using domain-oriented higher level representation languages may lead to higher flexibility of the system, easier adaptation to changing environments, reduced maintenance costs, and successful solutions of some problems which can't be tackled efficiently with standard techniques.

In order to create a knowledge-based CAPP-system, we examine the way a human expert solves the task in question. This leads to the general model of a process planning method shown in figure 1.

A process planner (expert) is given the description of the workpiece by a designer. This description consists of all geometrical and technological data which are necessary for the generation of a process plan. In this description the expert identifies characteristic parts or areas of interest, the so-called features. The exact form of these features is influenced by his manufacturing environment and by his personal experience. The expert associates features with generalized process plans, the so-called skeletal plans. The combination of features and associated skeletal plans represent the experience of this expert. Having build the description of the current workpiece in terms of features, the expert selects associated skeletal plans out of his memory (or out of an existing plan library). By combining these partial plans, bearing in mind some general and some specific principles, and by adapting them to the concrete workpiece, he creates in a refinement step the complete production plan.

It is important to realize that this observation implies that the features and the skeletal plans depend on the concrete expert as well as on the concrete working environment and may be different for another expert or another environment.

The described model of process planning was implemented in a first prototype called *Planning In Manufacturing* (PIM, see figure 2). The main parts of PIM, Graph Parser KaisersLautern (GraPaKL) and the skeletal planning module Skippy, are explained in the next sections. CAD-Compiler (CAD-Comp) and CAM-Compiler (CAM-UNC) are simple compilers connecting PIM to the outer world. The approach pursued in the PIM system closely mimics the expert's problem solving behavior. This system is especially tailored to a concrete manufacturing environment and uses the expert's knowledge optimally. This should lead to good quality of produced plans and to high acceptance of the system.

2 ABSTRACTION: Feature Recognition

Features are domain- and company-specific description elements based on the geometrical and technological data of a workpiece that an expert in a domain associates with certain informations [2]. Features are build upon a *syntax* (shape description, given here by productions of a graph grammar) and a *semantics* (description of related informations, e.g. skeletal plans in manufacturing or functional relations in design). They provide an abstraction mechanism to facilitate e.g. the creation or manufacturing of workpieces or more general to bridge the gap between several systems in the world of Computer Integrated Manufacturing (CIM).

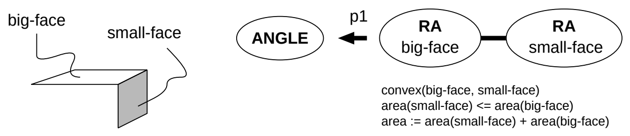

big-face
small-face
ANGLE
p1
RA big-face
RA small-face
convex(big-face, small-face)
area(small-face) <= area(big-face)
area := area(small-face) + area(big-face)

Figure 3: A sample feature definition

From the point of view of structural pattern recognition features may be characterized by the following points:

Interaction: Areas of features can overlap (see e.g. $p2$ in figure 4).

Dependence of Dimensions: In dependence of dimensions, the same structures may be recognized as different features.

Hierarchy: A feature description of a workpiece is hierarchical structured via a has-parts relation.

Ambiguity: A feature may be derivable in many syntactic different but semantic equivalent ways. Often only one pithy description is needed.

Similar Features: Features may be similar with respect to a has-parts or an is-a hierarchy.

As feature recognizer to realize the step of abstraction we developed a 'made-to-measure' grammar formalism, called FEATure REPresentation language (FEAT-REP) [8], and a heuristic driven chart-based bottom-up graph parser, called Graph Parser KaisersLautern (GraPaKL) [9], with the above characteristics.

Workpieces are represented by attributed graphs. The nodes of a workpiece graph represent geometric primitive surfaces, the node label decode the type of the surface, the attributes carry detailed geometric and technologic informations and the edges decode the topology of the workpiece, i.e. two nodes are adjacent if the corresponding surfaces touch each other.

Features are given here by productions of the graph grammar FEAT-REP and feature recognition is treated as a parsing problem. Our parser analyzes attributed node labeled graphs (representing workpieces) with a feature-specific attributed node labeled graph grammar (representing the feature definitions) yielding a high level (qualitative) description of the workpiece in terms of features.

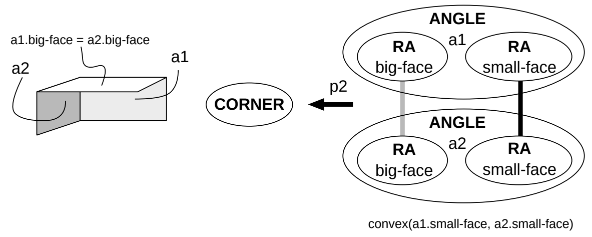


Figure 4: A feature definition with a deep connection and an overlap

In figure 3 a sample feature definition and its associated production is shown. An ANGLE consists of two connected atomic features of sort RectAngular (RA) surfaces, called *big-face* and *small-face* within the feature ANGLE. Several constraints are specified, like *convex(big-face, small-face)* and a function is used to define the attribute *area* of the feature ANGLE .

In figure 4 the (complex) feature CORNER is defined by two overlapping features ANGLE. The two ANGLEs (a1 and a2) overlap with their *big-face*, say they share the same surface as *big-face*. Additional their *small-face*'s must be adjacent. Our grammar formalism enables us to express two kinds of so-called deep relations between the components of features: neighboring (black edges) and overlapping (grey edges).

The application of a production is used by the parser only from right to left (i.e. for analysis) and the grammar's embedding rule states in this case, that the instance of the left-hand side (lhs) is adjacent to all nodes that the nodes on the right-hand side (rhs) were adjacent to. Geometrical interpreted it states, that two complex features touch each other, if at least one of their components do so. So in our formalism, edges are rather inherited than created.

The grammar formalism uses hierarchies of sorts to support a clear distinction between is-a and hast-parts hierarchies. Furthermore, hierarchies of sorts reduce the number of productions needed to describe the domain. This idea follows the idea of the *Knowledge Language ONE* (KL-ONE) based systems[0]. In figure 6 a sample has-parts hierarchy is shown. Not illustrated in this figure is the starting (workpiece) graph of surfaces.

The parser is an extension to the one introduced by Lutz [10]. It consists of three rules **initialize**, **choose** and **combine** that operate on two sets *agenda* and *chart* of complete (cp) and partial (pp) instances of productions, where a complete instance of production is one where every node on the rhs is instantiated and a partial instance of production is one where some nodes on the rhs are instantiated.

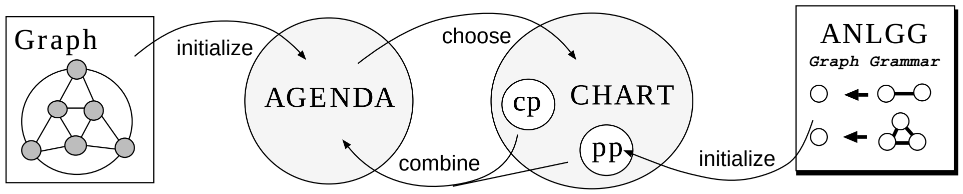


Figure 5: Architecture of the parser

The agenda is initialized with one cp for each of the (starting) graph's nodes (regarding as rule with an empty rhs) and the chart is initialized with one pp for each production (where no node of the rhs is instantiated). Then choose and combine are applied alternately until the agenda runs empty (see figure 5). If this happens the chart contains all possible parses. In practise the parser will be stopped after the first parse is found. Figure 6 illustrate a partial result of our parser.

Combine merges a chosen cp with all pp in chart respectively a chosen pp with all cp in chart such that cp fills one of the uninstantiated nodes in pp's rhs, (if cp satisfys the constraints given for that node) and creates a new pp (or probably cp) for every such successful combination. Since the application of a production does not introduce explicit new edges, the number of edges remains unchanged and hence edges can easily be used to index and quickly retrieve the instances in the chart, that can combine with a chosen instance to build such more complete instances.

The ability to incorporate search-guiding heuristics (by ordering the agenda and by ordering the production's nodes, i.e. by determining the order, that the parser uses to build an instance of the production) and the use of sorts helps to reduce the parser's search space and thereby to avoid the combinatoric explosion inherent to the graph parsing problem.

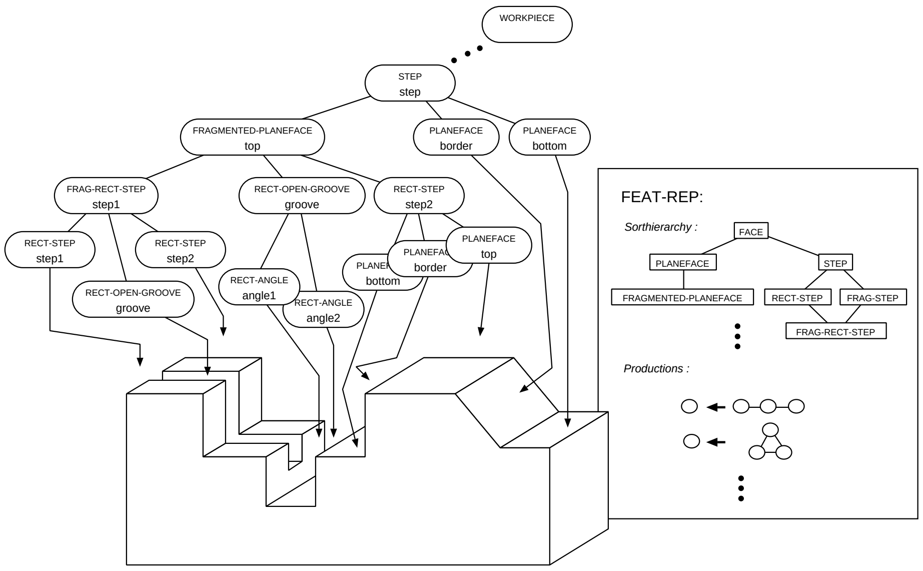


Figure 6: A workpiece and a partial feature structure

3 ASSOCIATION: Selecting of Skeletal Plans

To combine the expert's knowledge about the manufacturing process with feature structures as shown above we use skeletal plans [6]. Our skeletal plan representation formalism, called SKEletal Plan REPresentation language (SKEP-REP), allows the expert to write down his knowledge about the process plans necessary for the manufacturing of workpieces and for special parts of workpieces – the features. It also allows the expert to define how skeletal plans for features should be merged to complete process plans. Every skeletal plan contains concrete and/or abstract information about parts of the process plan (handling features) or about the whole process plan (handling the complete feature structure). Every skeletal plan is associated to a feature respectively a feature structure. Feature definition and skeletal plan together represent the expert's knowledge about a certain planning task.

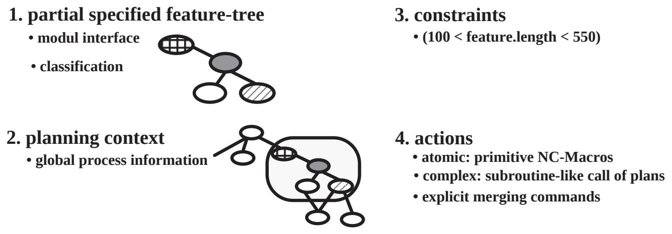

1. partial specified feature-tree
• modul interface
• classification
2. planning context
• global process information
3. constraints
• (100 < feature.length < 550)
4. actions
• atomic: primitive NC-Macros
• complex: subroutine-like call of plans
• explicit merging commands

Figure 7: Skeletal plan representation

In order to provide a suitable formalism for the representation of skeletal plans, the language SKEP-REP was developed. We currently use a formalism which describes skeletal plans as shown in figure 7. A skeletal plan contains the feature or feature structure it is associated to. It also contains some context information which relates to other skeletal plans forming preconditions for the application of this particular plan. It may also contain some applicability constraints which are not expressed by the features or the context of skeletal plans. Then it contains a sequence of operations, which may result in the subroutine-like call of other skeletal plans or in the generation of concrete planning steps. In figure 8 a sample skeletal plan for the feature *CREST* is given. The atomic actions are *ROUGHING* and *FINISHING* whereas *PLAN* specifies a complex action.

The idea of skeletal plans is straight forward: Represent knowledge about the intended solution in some (skeletal) plans. Find a first skeletal plan for the given feature (goal). The plan contains some steps which are subgoals respectively plans associated to parts of the feature to be reached in order to solve the whole problem. For every subgoal repeat the process until only elementary steps remain. The sequence of elementary steps is the intended plan [6].

In figure 9 the selection of the skeletal plans is illustrated. Our system always chooses the most specified plan, whose feature structure matches with the structure recognized by our parser. SKEP-REP as plan reuse framework uses restricted conceptual graphs as representation formalism and the selection of the skeletal plans as reusing is approached by retrieving the most specific general candidate and effectively modifying [15]. A similarity metrics about restricted conceptual graphs is given for guarding the effective retrieval.

The proposed restricted Conceptual Graph (RCG) is a special kind of conceptual graph, which is to restrict the representation mechanism of Sowa's conceptual graph [14] such as the number of relations, referent mapping, type function and formation rules. Additionally a taxonomic hierarchy of RCG's is used where the partial ordering is defined by generalization. To perform the selection of the skeletal plans, a prototypical RCG was implemented as a part of our system PIM.

```
(make-skeletal-plan
  :name "CREST (TP: one-directional, CAD: links, WD: from-the-right),
           i.e. band, longturn and planeface."
  :tree (list (make-node :sort   'CREST
                     :ident   0
                     :childs '((band . 1) (longturn . 2) (planeface . 3)))
           (make-node :sort   'BAND
                     :ident   1)
           (make-node :sort   'LONGTURN-OUT
                     :ident   2)
           (make-node :sort   'PLANEFACE
                     :ident   3))

  :constraints '((> ($ band reference_point) ($ planeface reference_point))
                    (not (eql 'unnecessary ($-status planeface-process)))
                    (eql 'one-directional ($-status turning-process))
                    (eql 'from-the-right ($-status working-direction))
                  )
  :actions '(
     (ROUGHING
            surfaces     (list ($ tec-rep))
            tool-groups   '(W.1)
            kind         'axial_overmeasure)
     (FINISHING
            surfaces     (list ($ tec-rep))
             tool-groups   '(W.2)
             kind         'contour_onmeasure)))
     (PLAN
             feature       ($ longturn))
     (SEPERATING
            tool-groups    '(W.4))
            ))
```

Figure 8: Skeletal plan for Feature *CREST*

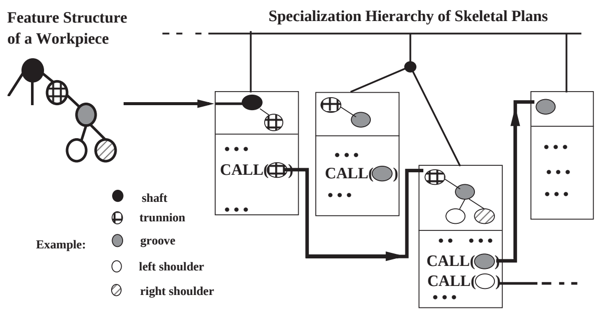


Figure 9: Selection of skeletal plans

4 REFINEMENT: Merging of Skeletal Plans

When a given workpiece description has been transformed in a feature structure according to the methods outlined in the chapter above, the skeletal plans associated with these features are found and selected according to the constraints embedded in the plans. The resulting set of skeletal plans is then merged to one final plan, and abstract variables are replaced by the concrete data of the workpiece in question.

The merging of the skeletal plans is oriented on several topics: Operations using the same tool should be performed consecutively (minimalization of tool change operations). Operations in one chucking context must be performed together, minimizing the changes of chucking. Different tools belonging to a common group may be exchanged against a more general tool of this group, such that several operations using slightly different tools can be merged to one operation using only one tool. Different surfaces of a workpiece which are treated with similar operations should be grouped together. These general rules will be used as default rules additionally to the rules given by the expert in the skeletal plans. The expert's rules will overwrite these general rules. This is the advantage of an expert: using efficient hard-and-fast (heuristic) rules.

The merging operations are supported by a hierarchical ordering of the available tools and a

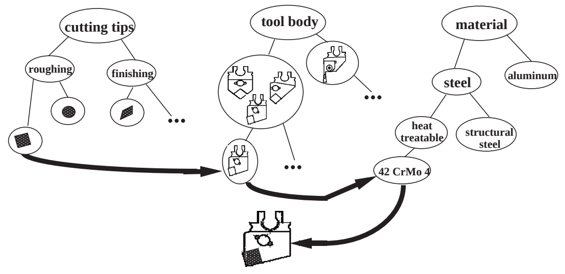


Figure 10: Hierarchies of tools, material and cutting tips

hierarchical grouping of the possible operations (see e.g. figure 10). Some heuristical approaches to skeletal plan merging are under investigation.

Figure 11 illustrate the output of the Skippy system. Without merging the output will be a sequence of operations for every surface. Without skeletal plans of the expert the output will be one operation using the universal (copying) tool. To generate a good plan both is needed: merging and know-how of the expert.

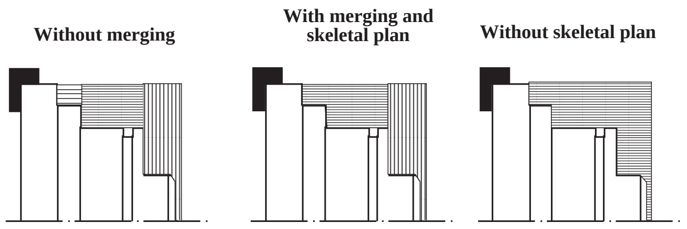


Figure 11: Different qualities of process plans

5 CONNECTIONS TO THE OUTER WORLD

Every CAPP-System serving any useful purpose must be embedded in the existing world of Computer Integrated Manufacturing (CIM). It has to accept the workpiece description from Computer Aided Design (CAD) and its output will be fed into Numerically-Controlled (NC) machines from Computer Aided Manufacturing (CAM). The system PIM fulfills this requirement using interfaces to CAD- and CAM-systems (see figure 2).

One interface to the CAD-world transforms the necessary geometrical/technological information about the workpiece from the CAD data into our internal representation formalism using the forthcoming ISO-standard *STandard for the Exchange of Product model data* (STEP) [7]. This standard promises to provide a data exchange format covering all necessary information and providing a system-independent interface. Another usable interface (CAD-Comp) connects the CAD-system *Konstruktionssystem Fertigungsgerecht* of Prof. Meerkamm, University of Erlangen [11] with our system PIM.

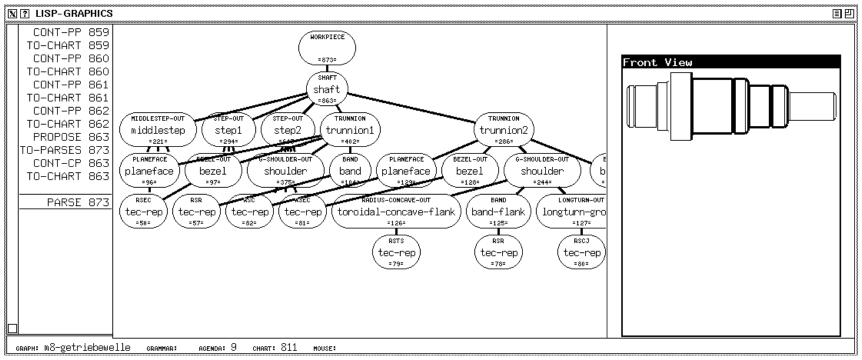
To get connected to the NC-machines we rely on components of CAM-systems available on the market today. In our case we choose the system UNC. The output of the system PIM is compiled (CAM-UNC) into the command language of the UNC CAM-system. Then, this system can generate

Cutter Location DATA (CLDATA) Code as well as machine specific NC-Code, without any further human interaction.

6 CONCLUSION

The observation of human expert's problem-solving behavior resulted in a model of process planning which supports a knowledge-based approach to CAPP. Based on technological/geometrical information about the workpiece, higher-level features are defined and associated with skeletal plans.

Special languages for a adequate representation of the necessary knowledge on the different abstraction levels are presented. The transformation and interpretation steps between the different languages have been implemented and form the planning system PIM. This system creates a production plan to a given workpiece by performing a sequence of abstraction, selection (association) and refinement following the principles of heuristic classification. Note that for the presented representation languages also editors have been developed. They are used currently by mechanical engineers to extend the knowledge base of features and skeletal plans. Besides milling and turning the GraPaKL-tool is used to recognize design features for mechanical parts classification.



```
  CONT-PP 859
 TO-CHART 859
  CONT-PP 860
 TO-CHART 860
  CONT-PP 861
 TO-CHART 861
  CONT-PP 862
 TO-CHART 862
  PROPOSE 863
TO-PARSES 873
  CONT-CP 863
 TO-CHART 863

    PARSE 873
```

Figure 12: User interface of PIM: intermediate result of GraPaKL

Every component of the system PIM is implemented in COMMON-Lisp using a compatible window system. The average runtime is about 2 sec. CPU depending on the domain (milling or turning) and the chosen workpiece. In figure 12 one possible intermediate state of PIM is illustrated: To support the user of PIM in the left window the feature structure is shown and in the right window the chosen workpiece. Clicking on one node of the tree the associated feature is highlighted on the workpiece.

In future work the objective of a new project *Intelligent Manager for Comprehensive Design* (IMCOD) is to make engineering systems more perceptive: Solutions of individual problems are abstracted into larger common contexts to obtain deeper structural insights and to achieve higher functionality and performance. This project aims at extending the competence of a design manager in his background knowledge by integrating the expertise of several (local) experts in order to achieve a feasible solution from an overall (global) point of view. Each local expert (e.g. PIM), is asked to contribute its professional judgement, taking into account each others' available assessment, globally reflected by cooperative negotiation.

7 ACKNOWLEDGEMENTS

This research was performed during the ARC-TEC project (Acquisition, Compilation and Representation of TEChnical knowledge) at the German Research Center for Artificial Intelligence Inc. (DFKI). It has been supported by grant from The Federal Ministry for Research and Technology (FKZ ITW-8902 C4).

8 REFERENCES

0 Baader, F. et al: *A Scheme for Integrating Concrete Domains into Concept Languages*, in: 12th IJCAI, pp. 452-457, 1991

1 Becker, A.: *Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung fuer die Arbeitsplanung.* Master Thesis, University of Kaiserslautern, 1991.

2 Chang, T.-C.: *Expert Process Planning for Manufacturing.* 1990.

3 Chuang, S.-H. et al: *Compound Feature Recognition by Web Grammar Parsing.* in: Research in Engineering Design, 1991, pp. 147-158.

4 Ehrig, H. et al: *Graph Grammars and Their Application to Computer Science.* LNCS 73, 153, 291, 532, 1979-1990.

5 Finger, S. et al: *Parsing Features in Solid Geometric Models.* in: ECAI'90, pp. 566-572.

6 Friedland, P.E. et al: *The Concept and Implementation of Skeletal Plans.* in: Journal of Automated Reasoning, 1985, pp. 161-208.

7 Grabowski, H. et al: *STEP - Entwicklung einer Schnittstelle zum Produktdatenaustausch.* VDI-Z 131, Nr. 9, 1989, pp. 68-76.

8 Klauck, Ch. et al: *FEAT-REP: Representing Features in CAD/CAM.* in: IV ISAI: Applications in Informatics, 1991, pp. 158-168.

9 Klauck, Ch. et al: *A Heuristic Driven Parser Based on Graph Grammars for Feature Recognition in CIM.* in: IAPR'92, 1992, pp. 200-210.

10 Lutz, R.: *Chart Parsing of Flowgraphs.* in: 11th IJCAI, 1989, pp. 116-121.

11 Meerkamm, H. et al: *Konstruktionssystem mfk - Integration von Bauteilsynthese und -analyse.* in: VDI Berichte 903, 1991, pp. 231-249.

12 Mullins, S. et al: *Grammatical Approaches to Engineering Design, Part I.* in: Research in Engineering Design, Springer-Verlag, 1991, pp. 121-135.

13 Rinderle, R.: *Grammatical Approaches to Engineering Design, Part II.* in: Research in Engineering Design, Springer-Verlag, 1991, pp. 137-146.

14 Sowa, J.F.: *Conceptual Structures: Information Processing in Man and Machine.* Addison-Wesley Publishing, 1984.

15 Wu, Z. et al: *Skeletal Plans Reuse: A Restricted Conceptual Graph Classification Approach.* in: Seventh AWCG, 1992, pp. 142-152.