μCAD2NC:  
A Declarative Lathe-Workplanning Model  
Transforming CAD-like Geometries  
into Abstract NC Programs  

Harold Boley, Philipp Hanschke, Martin Harm,  
Knut Hinkelmann, Thomas Labisch, Manfred Meyer,  
Jörg Müller, Thomas Oltzen, Michael Sintek,  
Werner Stein, Frank Steinle  

November 1991
Deutsches Forschungszentrum
für
Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Krupp-Atlas, Mannesmann-Kienzle, Philips, Sema Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director
μCAD2NC:
A Declarative Lathe-Workplanning Model Transforming
CAD-like Geometries into Abstract NC Programs

Harold Boley, Philipp Hanschke, Martin Harm, Knut Hinkelmann,
Thomas Labisch, Manfred Meyer, Jörg Müller, Thomas Oltzen,
Michael Sintek, Werner Stein, Frank Steinle

DFKI-D-91-15
This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8902 C4).

© Deutsches Forschungszentrum für Künstliche Intelligenz 1991

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.
μCAD2NC:
A Declarative Lathe-Workplanning Model
Transforming CAD-like Geometries into
Abstract NC Programs*

Harold Boley Philipp Hanschke
Martin Harm Knut Hinkelmann Thomas Labisch
Manfred Meyer Jörg Müller Thomas Oltzen
Michael Sintek Werner Stein Frank Steinle

DFKI Kaiserslautern
W-6750 Kaiserslautern, F.R. Germany

November 1991

Abstract
μCAD2NC is a knowledge-based system generating workplans for idealized lathe CNC machines. It transforms CAD-like geometries of rotational-symmetric workpieces into abstract NC programs, using declarative term representations for all processing steps. The system has been developed using COLAB, a hybrid-knowledge compilation laboratory which integrates the power of forward and backward reasoning (incl. functional programming), constraint propagation, and taxonomic classification. The focus of this work is on exemplifying techniques of the hybrid, declarative COLAB formalisms for the central subtasks of CAD-to-NC transformations.
5.1 Demo Workpiece

(attrterm (ring rng42 (tup (tup center1 110)
(tup center2 110)
(tup radius1 30)
(tup radius2 20))))

(attrterm (cylinder cyl43 (tup (tup center1 110)
(tup center2 120)
(tup radius1 20)
(tup radius2 20))))

(attrterm (ring rng44 (tup (tup center1 120)
(tup center2 120)
(tup radius1 20)
(tup radius2 25))))

(attrterm (cylinder cyl45 (tup (tup center1 120)
(tup center2 150)
(tup radius1 25)
(tup radius2 25))))

(attrterm (circle circ46 (tup (tup center1 150)
(tup center2 150)
(tup radius1 25)
(tup radius2 0))))

(fact (neighbor circ33 cyl34))
(fact (neighbor cyl34 rng35))
(fact (neighbor rng35 cyl36))
(fact (neighbor cyl36 rng37))
(fact (neighbor rng37 tr38))
(fact (neighbor tr38 cyl39))
(fact (neighbor cyl39 rng40))
(fact (neighbor rng40 cyl41))
(fact (neighbor cyl41 rng42))
(fact (neighbor rng42 cyl43))
(fact (neighbor cyl43 rng44))
(fact (neighbor rng44 cyl45))
(fact (neighbor cyl45 circ46))
5.2 Detailed Trace

Here we give a complete trace of the $\mu$CAD2NC sample session showing the "operational semantics" of our CAD-to-NC transformation model. Except from the LaTeXpage formatting, the trace is reproduced verbatim from an actual COLAB demo run.

colab,fw> (demo/f)

derived-fact-asserted-into-taxon-abox
  (add-data
    shoulder
    shoulder-rng35-lts-cyl36-cyl136
    (tup
     (tup ground lts-cyl36-cyl136)
     (tup flank rng35)
     (tup leftmost rng35)
     (tup rightmost cyl136) ) )

derived-fact-asserted-into-taxon-abox
  (add-data
    longturningsurface
    lts-cyl136-cyl136
    (tup (tup radius 20) (tup leftmost cyl136) (tup rightmost cyl136)) )

derived-fact-asserted-into-taxon-abox
  (add-data
    shoulder
    shoulder-lts-cyl136-cyl136-rng37
    (tup
     (tup ground lts-cyl136-cyl136)
     (tup flank rng37)
     (tup leftmost cyl136)
     (tup rightmost rng37) ) )

derived-fact-asserted-into-taxon-abox
  (add-data
    shoulder
    shoulder-rng40-lts-cyl41-circ46
    (tup
     (tup ground lts-cyl41-circ46)
     (tup flank rng40)
     (tup leftmost rng40)
     (tup rightmost circ46) ) )

\footnote{Of course, also selective traces are possible and everything except the final ANC program would be hidden to end users.}
5.2 Detailed Trace

derived-fact-asserted-into-taxon-abox
(add-data
douturningsurface
lts-cyl41-circ46
(tup (tup radius 30) (tup leftmost cyl41) (tup rightmost circ46)) )

derived-fact-asserted-into-taxon-abox
(add-data
shoulder
shoulder-rng42-lts-cyl43-cyl43
(tup
(tup ground lts-cyl43-cyl43)
(tup flank rng42)
(tup leftmost rng42)
(tup rightmost cyl43) ) )

derived-fact-asserted-into-taxon-abox
(add-data
shoulder
shoulder-rng42-lts-cyl43-circ46
(tup
(tup ground lts-cyl43-circ46)
(tup flank rng42)
(tup leftmost rng42)
(tup rightmost circ46) ) )

derived-fact-asserted-into-taxon-abox
(add-data
louturningsurface
lts-cyl43-circ46
(tup (tup radius 25) (tup leftmost cyl43) (tup rightmost circ46)) )

derived-fact-asserted-into-taxon-abox
(add-data
shoulder
shoulder-lts-cyl43-cyl43-rng44
(tup
(tup ground lts-cyl43-cyl43)
(tup flank rng44)
(tup leftmost cyl43)
(tup rightmost rng44) ) )
derived-fact-asserted-into-taxon-abox
(add-data
  longturningsurface
  lts-cyl43-cyl43
  (tup (tup radius 20) (tup leftmost cyl43) (tup rightmost cyl43)) )

derived-fact-asserted-into-taxon-abox
(add-data
  shoulder
  shoulder-lts-circ33-rng37-tr38
  (tup
    (tup ground lts-circ33-rng37)
    (tup flank tr38)
    (tup leftmost circ33)
    (tup rightmost tr38)) )

derived-fact-asserted-into-taxon-abox
(add-data
  longturningsurface
  lts-circ33-rng37
  (tup (tup radius 25) (tup leftmost circ33) (tup rightmost rng37)) )

derived-fact-asserted-into-taxon-abox
(add-data
  groove
  groove-rng35-lts-cyl36-cyl36-rng37
  (tup
    (tup leftflank rng35)
    (tup ground lts-cyl36-cyl36)
    (tup rightflank rng37)
    (tup leftmost rng35)
    (tup rightmost rng37)) )

derived-fact-asserted-into-taxon-abox
(add-data
  groove
  groove-rng42-lts-cyl43-cyl43-rng44
  (tup
    (tup leftflank rng42)
    (tup ground lts-cyl43-cyl43)
    (tup rightflank rng44)
    (tup leftmost rng42)
    (tup rightmost rng44)) )
5.2 Detailed Trace

features:

(tup
desc-tc
go35
(tup (tup center1 40) (tup center2 40) (tup radius1 25) (tup radius2 20))
(asc-tc
go37
(tup (tup center1 50) (tup center2 50) (tup radius1 20) (tup radius2 20))
(desc-tc
go40
(tup (tup center1 70) (tup center2 70) (tup radius1 40) (tup radius2 30))
(desc-tc
go42
(tup (tup center1 110) (tup center2 110) (tup radius1 30) (tup radius2 20))
(asc-tc
go44
(tup (tup center1 120) (tup center2 120) (tup radius1 20) (tup radius2 20))
(desc-tc
go46
(tup (tup center1 150) (tup center2 150) (tup radius1 25) (tup radius2 0))
(asc-tc
go33
(tup (tup center1 0) (tup center2 0) (tup radius1 0) (tup radius2 25))
(asc-tc
go38
(tup (tup center1 50) (tup center2 60) (tup radius1 25) (tup radius2 40))
(data
shoulder
shoulder-rng35-lts-cyl36-cyl36
(tup
(tup ground lts-cyl36-cyl36)
(tup flank rng35)
(tup leftmost rng35)
(tup rightmost cycyl36))
(data
longturningsurface
lts-cyl36-cyl36
(tup (tup radius 20) (tup leftmost cycyl36) (tup rightmost cycyl36))
(data
shoulder
shoulder-lts-cyl36-cyl36-rng37
(tup
(tup ground lts-cyl36-cyl36)
(tup flank rng37)
(tup leftmost cycyl36)
(tup rightmost rng37))


(data
shoulder
shoulder-rng40-lts-cyl41-circ46
(tup
(tup ground lts-cyl41-circ46)
(tup flank rng40)
(tup leftmost rng40)
(tup rightmost circ46) ) )

(data
longturningsurface
lts-cyl41-circ46
(tup (tup radius 30) (tup leftmost cyl41) (tup rightmost circ46)) )

(data
shoulder
shoulder-rng42-lts-cyl43-cyl43
(tup
(tup ground lts-cyl43-cyl43)
(tup flank rng42)
(tup leftmost rng42)
(tup rightmost cyl43) ) )

(data
shoulder
shoulder-rng42-lts-cyl43-circ46
(tup
(tup ground lts-cyl43-circ46)
(tup flank rng42)
(tup leftmost rng42)
(tup rightmost circ46) ) )

(data
longturningsurface
lts-cyl43-circ46
(tup (tup radius 25) (tup leftmost cyl43) (tup rightmost circ46)) )

(data
shoulder
shoulder-lts-cyl43-cy143-rng44
(tup
(tup ground lts-cyl43-cy143)
(tup flank rng44)
(tup leftmost cyl43)
(tup rightmost rng44) ) )

(data
longturningsurface
lts-cyl43-cy143
(tup (tup radius 20) (tup leftmost cyl43) (tup rightmost cyl43)) )

(data
shoulder
shoulder-lts-circ33-rng37-tr38
(tup
(tup ground lts-circ33-rng37)
(tup flank tr38)
(tup leftmost circ33)
(tup rightmost tr38) ) )
5.2 Detailed Trace

(data longturningsurface lts-circ33-rng37
tup (tup radius 25) (tup leftmost circ33) (tup rightmost rng37))
(data groove groove-rng35-lts-cyl36-cyl36-rng37
tup
tup leftflank rng36
(tup ground lts-cyl36-cyl36)
(tup rightflank rng37)
(tup leftmost rng35)
(tup rightmost rng37)
(data groove groove-rng42-lts-cyl43-cyl43-rng44
tup
tup leftflank rng42
(tup ground lts-cyl43-cyl43)
(tup rightflank rng44)
(tup leftmost rng42)
(tup rightmost rng44))}
classified workpiece:

(cwp
40
(tup
(nft
(1sh (flk (tup (p 70 40) (p 70 30))) (grd (tup (p 70 30) (p 150 30)))))
(tup
(nft
(1sh (flk (tup (p 110 30) (p 110 25))) (grd (tup (p 110 25) (p 150 25)))))
(tup
(nft
(grv
(flk (tup (p 110 25) (p 110 20)))
(grd (tup (p 110 20) (p 120 20)))
(flk (tup (p 120 20) (p 120 25)))))
(tup))))
(nft
(rsh (grd (tup (p 0 25) (p 50 25))) (flk (tup (p 50 25) (p 60 40))))
(tup
(nft
(grv
(flk (tup (p 40 25) (p 40 20)))
(grd (tup (p 40 20) (p 50 20)))
(flk (tup (p 50 20) (p 50 25)))))
(tup))))
)
5.2 Detailed Trace

contax tool selection:
arguments: roughing high-alloy-steel normal 0 90 left
propagating...
results: (tup (tup dnmm-71 tmaxp-pd193) (tup rcmx tmaxp-prl40)
         (tup rcmx tmaxp-prl130) (tup tnmm-71 tmaxp-ptr90))

contax tool selection:
arguments: roughing high-alloy-steel normal 0 90 left
propagating...
results: (tup (tup dnmm-71 tmaxp-pd193) (tup rcmx tmaxp-prl40)
         (tup rcmx tmaxp-prl130) (tup tnmm-71 tmaxp-ptr90))
          (grd (tup (p 40 20) (p 50 20)))
          (flk (tup (p 50 20) (p 50 25)))

contax tool selection:
arguments: roughing high-alloy-steel normal 90 90 right
propagating...
results: (tup (tup dnmm-71 tmaxp-pdr93) (tup rcmx tmaxp-prr30)
          (tup tnmm-71 tmaxp-prt90))

contax tool selection:
arguments: roughing high-alloy-steel normal 90 90 left
propagating...
results: (tup (tup dnmm-71 tmaxp-pd193) (tup rcmx tmaxp-prl40)
         (tup rcmx tmaxp-prl130) (tup tnmm-71 tmaxp-ptr90))

contax tool selection:
arguments: roughing high-alloy-steel normal 90 90 left
propagating...
results: (tup (tup dnmm-71 tmaxp-pd193) (tup rcmx tmaxp-prl40)
         (tup rcmx tmaxp-prl130) (tup tnmm-71 tmaxp-ptr90))

contax tool selection:
arguments: roughing high-alloy-steel normal 0 60 right
propagating...
results: (tup (tup dnmm-71 tmaxp-pdr93) (tup tnmm-71 tmaxp-ptn60)
         (tup rcmx tmaxp-prr30))

contax tool selection:
arguments: roughing high-alloy-steel normal 90 90 right
propagating...
results: (tup (tup dnmm-71 tmaxp-pdr93) (tup rcmx tmaxp-prr30)
         (tup tnmm-71 tmaxp-prt90))

contax tool selection:
arguments: roughing high-alloy-steel normal 90 90 left
propagating...
results: (tup (tup dnmm-71 tmaxp-pd193) (tup rcmx tmaxp-prl40)
         (tup rcmx tmaxp-prl130) (tup tnmm-71 tmaxp-ptr90))
skeletal plan:

```
(skp
40
(com
(tup
(seq
(tup
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pd193)
left
(geo (tup (p 70 40) (p 70 30) (p 150 30)))
(roughing
(tool rcmx tmaxp-pr140)
left
(geo (tup (p 70 40) (p 70 30) (p 150 30)))
(roughing
(tool rcmx tmaxp-pr130)
left
(geo (tup (p 70 40) (p 70 30) (p 150 30)))
(roughing
(tool tnmm-71 tmaxp-pt190)
left
(geo (tup (p 70 40) (p 70 30) (p 150 30))))
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pd193)
left
(geo (tup (p 110 30) (p 110 25) (p 150 25)))
(roughing
(tool rcmx tmaxp-pr140)
left
(geo (tup (p 110 30) (p 110 25) (p 150 25)))
(roughing
(tool rcmx tmaxp-pr130)
left
(geo (tup (p 110 30) (p 110 25) (p 150 25)))
(roughing
(tool tnmm-71 tmaxp-pt190)
left
(geo (tup (p 110 30) (p 110 25) (p 150 25))))
(alt
(tup
(seq
(tup
(alt
(tup
(roughing
```

5 SAMPLE SESSION
5.2 Detailed Trace

(tool dnmm-71 tmaxp-pd193)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool rcmx tmaxp-prl40)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool rcmx tmaxp-prl30)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool tnmm-71 tmaxp-ptr190)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)) ) )
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pdr93)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool rcmx tmaxp-prr30)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool tnmm-71 tmaxp.ptr90)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))) ) )
(seq
(tup
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pdr93)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool rcmx tmaxp-prr30)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool tnmm-71 tmaxp-ptr90)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)) ) ) )
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pd193)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing

(tool rcmx tmaxp-prl40)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool rcmx tmaxp-prl30)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
(roughing
(tool tnmm-71 tmaxp-ptl90)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
)
(seq
(tup
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pd193)
right
(geo (tup (p 0 25) (p 50 25) (p 60 40)))
(roughing
(tool tnmm-71 tmaxp.ptr90)
right
(geo (tup (p 0 25) (p 50 25) (p 60 40)))
(roughing
(tool tnmm-71 tmaxp-tn160)
right
(geo (tup (p 0 25) (p 50 25) (p 60 40)))
(roughing
(tool rcmx tmaxp-prr30)
right
(geo (tup (p 0 25) (p 50 25) (p 60 40)))
)
(alt
(tup
(seq
(tup
(alt
(tup
(roughing
(tool dnmm-71 tmaxp-pd193)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool rcmx tmaxp-prl40)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool rcmx tmaxp-prl30)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool tnmm-71 tmaxp-ptl90)
left
5.2 Detailed Trace

(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(alt
(tup
(roughing
(tool dnmn-71 tmaxp-pdr93)
right
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool rcmx tmaxp-prr30)
right
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool tnmn-71 tmaxp-prr90)
right
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
)
)
)
(seq
(tup
(alt
(tup
(roughing
(tool dnmn-71 tmaxp-pdr93)
right
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool rcmx tmaxp-prr30)
right
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool tnmn-71 tmaxp-prr90)
right
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
)
)
(alt
(tup
(roughing
(tool dnmn-71 tmaxp-pdl93)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool rcmx tmaxp-prl40)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool rcmx tmaxp-prl30)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
(roughing
(tool tnmn-71 tmaxp-pnl90)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
)
)
)
)
)
)
)
qualitative simulation:

(tup
  (roughing
    (tool dnmm-71 tmaxp-pdl93)
    left
    (geo (tup (p 70 40) (p 70 30) (p 150 30)))
  )
  (roughing
    (tool dnmm-71 tmaxp-pdl93)
    left
    (geo (tup (p 110 30) (p 110 25) (p 150 25)))
  )
  (roughing
    (tool dnmm-71 tmaxp-pdl93)
    left
    (geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
  )
)

4.4->
(skp
40
(com
(tup
  (alt
    (tup
      (roughing
        (tool dnmm-71 tmaxp-pdr93)
        right
        (geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
      )
      (roughing
        (tool rcax tmaxp-prr30)
        right
        (geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
      )
      (roughing
        (tool tnmm-71 tmaxp-ptr90)
        right
        (geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
      )
    )
  )
  (seq
    (tup
      (alt
        (tup
          (roughing
            (tool dnmm-71 tmaxp-pdr93)
            right
            (geo (tup (p 0 25) (p 50 25) (p 60 40)))
          )
          (roughing
            (tool tnmm-71 tmaxp-ptr90)
            right
            (geo (tup (p 0 25) (p 50 25) (p 60 40)))
          )
          (roughing
            (tool tnmm-71 tmaxp-tpn60)
            right
            (geo (tup (p 0 25) (p 50 25) (p 60 40)))
          )
        )
      )
    )
  )
)
5.2 Detailed Trace

(roughing
  (tool rcmx tmaxp-prr30)
  right
  (geo (tup (p 0 25) (p 50 25) (p 60 40))) )

(alt
  (tup
    (seq
      (tup
        (alt
          (tup
            (roughing
              (tool dnmm-71 tmaxp-pd193)
              left
              (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
            (roughing
              (tool rcmx tmaxp-prl40)
              left
              (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
            (roughing
              (tool rcmx tmaxp-prl30)
              left
              (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
            (roughing
              (tool tnnm-71 tmaxp-ptl90)
              left
              (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
          )
          (seq
            (tup
              (alt
                (tup
                  (roughing
                    (tool dnmm-71 tmaxp-pd93)
                    right
                    (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
                  (roughing
                    (tool rcmx tmaxp-prr30)
                    right
                    (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
                  (roughing
                    (tool tnnm-71 tmaxp-prr90)
                    right
                    (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
                )
            )
          )
        )
      )
    )
  )
)
(roughing
  (tool rcmx tmaxp-prl30)
  left
  (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)) ) )
(roughing
  (tool tnmm-71 tmaxp-pt190)
  left
  (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)) ) ) )

(tup
 (roughing
  (tool dnmm-71 tmaxp-pd193)
  left
  (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)) ) )
1.0->
(skp 40 (tup))
anc-program:

(tup
  (roughing
    (tool dnmm-71 tmaxp-pd193)
    left
    (geo (tup (p 70 40) (p 70 30) (p 150 30))))
  (roughing
    (tool dnmm-71 tmaxp-pd193)
    left
    (geo (tup (p 110 30) (p 110 25) (p 150 25)))
  (roughing
    (tool dnmm-71 tmaxp-pd193)
    left
    (geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
  (roughing
    (tool dnmm-71 tmaxp-pdr93)
    right
    (geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
  (roughing
    (tool dnmm-71 tmaxp-pdr93)
    right
    (geo (tup (p 0 25) (p 50 25) (p 60 40)))
  (roughing
    (tool dnmm-71 tmaxp-pdr93)
    right
    (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
  (roughing
    (tool dnmm-71 tmaxp-pd193)
    left
    (geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
) )
Acknowledgements

Besides the authors several other people have contributed, directly or indirectly, to the COLAB and/or μCAD2NC systems. In particular, our thanks go to Hans-Günter Hein, Klaus Elsbernd and Bernd Reuther. Also, we want to express our gratitude to Prof. Dr. Michael M. Richter for inspiring our AI involvement in mechanical engineering with the ARC-TEC project, and for critically reading an earlier draft of this paper.

References


A FORWARD Sources

A.1 Feature Aggregation

<table>
<thead>
<tr>
<th>Feature Aggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>microCAD2MC</td>
</tr>
<tr>
<td>FORWARD part</td>
</tr>
<tr>
<td>Feature Aggregation Rules</td>
</tr>
<tr>
<td>(c) Knut Emkelmann</td>
</tr>
<tr>
<td>Martin Hars September 1991</td>
</tr>
</tbody>
</table>

---

Features derived by these rules are asserted into the ABox:
- Attributes common to all features are the leftmost and rightmost surface. This is necessary to check neighborhood of surfaces and features.

- A shoulder is a feature consisting of two components:
  - The ground is a longturning surface.
  - The flank is an either a descending surface (on the left) or an ascending surface (on the right) of the longturning surface. An additional condition is that the radius of the longturning surface is not greater than that of the descending or ascending surface, respectively.

Examples:

A groove is an aggregation of two shoulders:
- A left shoulder and a right shoulder with common ground:

Examples:

There are three definitions for a longturning surface:
1. Each cylinder is a longturning surface
2. Starting from a descending surface a longturning surface extends to the right until either the workpiece ends or...
A.1 Feature Aggregation

```
(< _rada _rad-limit)
(is _new-max (max _rad-max _rada))
(sub-lts-from-left
  "(tup _rad-max _new-max) ; bishieres maximum"
  "(tup _rad-limit _rad-limit) ; obere Schranke( darf nicht! erreicht werden"
  "(tup _leftmost _1)"
  "(tup _rightmost _next-tc)"
)

(ft (sub-lts-from-left (tup _rad-max _the-max)
  (tup _rad-limit _rad-limit)
  (tup _leftmost _tc)
  (tup _rightmost _tc))
_the-max)

; up add-data
(up (add-data longturningsurface _featid
  (tup (tup _radius _rad)
        (tup _leftmost _leftm)
        (tup _rightmost _rightm)))
  (desc-tc _left-end
    "(tup _center1 _zl)
      (tup _center2 _zr)
      (tup _radius1 _rad-limit)
      (tup _radius2 _rad-first))"
  (neighbor _left-end _leftm)
  (is _rad (once
    "(sub-lts-from-right (tup _rad-max _rad-first)
      (tup _rad-limit _rad-limit)
      (tup _rightmost _r)
      (tup _leftmost _leftm))))
  (different-tc _r _leftm)
  (is _featid (make-instance-name lts _leftm _r)))
)

(ft (sub-lts-from-right
  (tup _rad-max _rad-max) ;; bishieres maximum
  (tup _rad-limit _rad-limit) ;; obere Schranke( darf nicht! erreicht werden"
  (tup _rightmost _r)
  (tup _leftmost _tc)
  (neighbor _tc _next-tc)
  (trunccone _next-tc
    "(tup _center1 _zl)
      (tup _center2 _zr)
      (tup _radius1 _rada)
      (tup _radius2 _radb))"
  (< _rada _rad-limit)
  (is _new-max (max _rad-max _rada))
  (sub-lts-from-right "(tup _rad-max _new-max"
    "(tup _rad-limit _rad-limit"
    "(tup _rightmost _r)"
)

; the radius of a surface exceeds the radius of the descending surface.
; The radius of such a longturning surface is the radius of the highest surface covered by the longturning surface.
;
\|--------------------(the dashed line shows the longturning surface)n
\| 1__
\| 1
;
3. Similar as 2. starting at an ascending surface and going to the left.

(up (add-data longturningsurface _cyl
  (tup (tup _radius _rad)
        (tup _leftmost _cyl)
        (tup _rightmost _cyl))
  (trunccone _cyl (tup _center1 _zl)
    (tup _center2 _zr)
    (tup _radius1 _rad)
    (tup _radius2 _rad)))
)

(up (add-data longturningsurface _featid
  (tup (tup _radius _rad)
        (tup _leftmost _1)
        (tup _rightmost _rightm))
  (asc-tc _right-end
    "(tup _center1 _zl)
      (tup _center2 _zr)
      (tup _radius1 _rad-first)
      (tup _radius2 _rad-limit))"
  (neighbor _rightm _right-end)
  (is _rad (once
    "(sub-lts-from-left (tup _rad-max _rad-first)
      (tup _rad-limit _rad-limit)
      (tup _leftmost _1)
      (tup _rightmost _rightm))))
  (different-tc _1 _rightm)
  (is _featid (make-instance-name lts _1 _rightm)))
)

(ft (sub-lts-from-left
  (tup _rad-max _rad-max) ;; bishieres maximum
  (tup _rad-limit _rad-limit) ;; obere Schranke( darf nicht! erreicht werden"
  (tup _leftmost _1)
  (tup _rightmost _tc)
  (neighbor _next-tc _tc)
  (trunccone _next-tc
    "(tup _center1 _zl)
      (tup _center2 _zr)
      (tup _radius1 _rada)
      (tup _radius2 _radb))"
)
```
A FORWARD SOURCES

A.1 Feature Aggregation

; Descending truncated cone: First radius is smaller than second radius
(rl (desc-tc _x (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(truncone _x ' (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(< _rada _radb))

; Truncated cone is the general form of a rotation symmetric surface.
; Therefore every surface is a truncated cone.
(rl (truncone _x (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb))))
(ring _x ' (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(r1 (truncone _x (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(cylinder _x ' (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(r1 (truncone _x (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb))))
(cIRCLE _x ' (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))

; Ascending truncated cone: First radius is greater than second radius
(rl (asc-tc _x (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(truncone _x ' (tup (tup center1 _zl)
               (tup center2 _zr)
               (tup radius1 _rada)
               (tup radius2 _radb)))
(> _rada _radb)
A.2 TAXON Access Functions and Reasoning Strategies

The definition of DATA retrieves attribute values of a concept's instances from the TAXON ABox. It applies the access function TX with specifier INSTANCE to get all instances of the given concept and retrieves all the attributes for this instance.

```
(hn (data _ concept _ instance _ attr-terms)
  (member _ instance (tx instances _ concept))
  (is _ attr-terms (data-attributes _ instance _ attr-terms))
  (rf-terpri)
  (is _ dummy (rf-terpri))
  (rf-print "Retrieval from TAXON ABox: ")
  (rf-print '(data _ concept _ instance _ attr-terms)))
)
```

The function DATA-ATTRIBUTES retrieves all attribute values for a given instance.

```
(ft (data-attributes _ instance))
(ft (data-attributes _ instance (tup (tup _ attr _ val))))
  (is _ val (tx attr-filler _ attr _ instance))
  '(tup (tup _ attr _ val))
(ft (data-attributes _ instance (tup (tup _ attr _ val) (tup _ attr2 _ val2) _ attr-terms))
  (is _ val (tx attr-filler _ attr _ instance))
  (is _ rest-terms (data-attributes _ instance _ attr-terms))
  '(tup (tup _ attr _ val) _ rest-terms))
```
A.2 TAXON Access Functions and Reasoning Strategies

; FORWARD-Reasoning Strategies:
; Bottom-up reasoning as applied in MARDGC is simulated in the
; HELFUN system. Therefore the bidirectional rules are transformed
; into HELFUN un-rules and compiled into an extended AFM with
; special forward code area and retain stack.
; As reasoning strategies only breadth-first search is applied
; in our example.

; Breadth-first Search:
; (bf-all (tup _fact | _rest) _inference-pattern) as its value
; the list of consequences for its second argument derived
; in depth-first search.

(hn (bf-all (tup _fact | _rest) _inference-pattern)
  (fc-initialize)
  (satisfied (tup _fact | _rest))
  (bf-alist (tup _fact | _rest) _inference-pattern))
(hn (bf-all _fact _inference-pattern)
  (sou _fact ' (tup _x))
  (fc-initialize)
  ;_Fact
  (forward _fact _conclusion)
  false)

(fit (bf-all _fact _inference-pattern)
  (forward-all)
  (is _inference-list (filter _inference-pattern _inferences))
  (reset-retain)
  _inference-list)

(hn (bf-alist (tup _fact | _rest) _inference-pattern)
  (forward _fact _inference)
  false)

(fit (bf-alist (tup _fact | _rest) _inference-pattern)
  (bf-alist _rest _inference-pattern))

(hn (forward-all)
  (open-node _fact)
  (forward _fact _conclusion)
  false)

(hn (forward-all))
; The initial facts of forward reasoning must be satisfied, so that
; they can be used for proving premises of rules.
; Instead of simply testing it would be possible to assert them if
; they are not already satisfied.

(hn (satisfied (tup)))
(hn (satisfied (tup _Fact | _Rest)))

(satisfied _Fact)

(hn (not-reached _Conclusion))

(is t (subsumes-value _Conclusion)))

B TAXON Sources

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; microCAD2DC1
;;; TAXON part
;;; Features
;;; (c) Philipp Hansche September 1991
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Interesting features defined in this file
(hierarchy FEATURE

COMPOSED ATOMIC
; DESCENDING ASCENDING
; HOLLOW FILLED
RSHOULDER LSHOULDER
LONGTURNINGSURFACE

lue-from-right
lue-from-left
GROOVE
TRUNCONE CYLINDER COKE RING CIRCLE
ASC-CONE DESC-CONE
ASC-RING DESC-RING
ASC-tc DESC-tc
DESC-CIRCLE ASC-CIRCLE
SCORE S mber SHOULDER)

; The concrete domain of rational numbers with comparison
; operators and boolean connectives is assigned to the tag RA.
; (domain RA edom-real-ord)

; Some simple predicates of the concrete domain.

(pred >0 (RA (x) (> x 0)))
(pred <0 (RA (x) (< x 0)))
(pred >=0 (RA (x) (>= x 0)))
(pred <=0 (RA (x) (<= x 0)))
(pred =0 (RA (x) (= x 0)))
(pred < (RA (x y) (< x y)))
(pred > (RA (x y) (> x y)))
(pred <= (RA (x y) (<= x y)))
(pred > (RA (x y) (> x y)))
(pred = (RA (x y) (= x y)))

; Separate atomic and composed objects

(atom atomic)
(conc composed (not atomic))
A truncated cone is given by two centers and two radii.

\textbf{Attr: }\text{center1}
\text{center2}
\text{radius1}
\text{radius2}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ center1} \\
\text{ra} \text{ center2} \\
\text{>0 radius1} \\
\text{>0 radius2} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ radius1 radius2 center1 center2} \\
\text{and} \text{ (>0 radius1)} \\
\text{ (>0 radius2)} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ center1} \\
\text{ra} \text{ center2} \\
\text{>0 radius1} \\
\text{>0 radius2} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ radius1 radius2 center1 center2} \\
\text{and} \text{ (>0 radius1)} \\
\text{ (>0 radius2)} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ center1} \\
\text{ra} \text{ center2} \\
\text{>0 radius1} \\
\text{>0 radius2} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ center1} \\
\text{ra} \text{ center2} \\
\text{>0 radius1} \\
\text{>0 radius2} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

\textbf{ TC-Condition: }
\begin{align*}
\text{ra} \text{ radius1 radius2 center1 center2} \\
\text{and} \text{ (>0 radius1)} \\
\text{ (>0 radius2)} \\
\text{or} \text{ (= center1 center2)} \\
\text{and} \text{ (= radius1 radius2)} \\
\text{and} \text{ (= center1 center2)} \\
\text{or} \text{ (>0 radius1)} \\
\text{ (>0 radius2)}
\end{align*}

A cylinder
\begin{align*}
\text{conc cylinder} \\
\text{and} \text{ (tc-Condition = radius1 radius2)}
\end{align*}

A circle is a ring where one radius is 0
\begin{align*}
\text{conc circle} \\
\text{and} \text{ (ring (or (=0 radius1) (=0 radius2)))}
\end{align*}

A cone
\begin{align*}
\text{conc cone} \\
\text{and} \text{ (tc-Condition (or (=0 radius1) (=0 radius2)))}
\end{align*}

Applying some adjectives to ring, circle, and cone.
\begin{align*}
\text{conc asc-rc \label{asc-ring} \text{and} \text{ (ring ascending)}} \\
\text{conc desc-rc \label{desc-ring} \text{and} \text{ (ring descending)}} \\
\text{conc asc-circ \label{asc-circle} \text{and} \text{ (ring (=0 radius1))}} \\
\text{conc desc-circ \label{desc-circle} \text{and} \text{ (ring (=0 radius2))}} \\
\text{conc asc-cne \label{asc-cone} \text{and} \text{ (tc-Condition (=0 radius1))}} \\
\text{conc desc-cne \label{desc-cone} \text{and} \text{ (tc-Condition (=0 radius2))}}
\end{align*}

A ring is a very flat surface
\begin{align*}
\text{conc ring} \\
\text{and} \text{ (tc-Condition (= center1 center2))}
\end{align*}

Two adjectives, filled and hollow, suitable for non rings. They replace the in/out resp. left/right resp. +/- flags determining the orientation of the surface.
\begin{align*}
\text{conc filled} \label{filled} \text{and} \text{ (= center1 center2)} \\
\text{conc hollow} \text{and} \text{ (> center1 center2)}
\end{align*}

Two adjectives, ascending and descending, suitable for filled truncated cones.
\begin{align*}
\text{conc ascending} \text{and} \text{ (= radius1 radius2)} \\
\text{conc descending} \text{and} \text{ (> radius1 radius2)}
\end{align*}

A ring is a very flat surface
\begin{align*}
\text{conc ring} \\
\text{and} \text{ (tc-Condition (= center1 center2))}
\end{align*}

Two adjectives, filled and hollow, suitable for non rings. They replace the in/out resp. left/right resp. +/- flags determining the orientation of the surface.
\begin{align*}
\text{conc filled} \text{and} \text{ (= center1 center2)} \\
\text{conc hollow} \text{and} \text{ (> center1 center2)}
\end{align*}

Two adjectives, ascending and descending, suitable for filled truncated cones.
\begin{align*}
\text{conc ascending} \text{and} \text{ (= radius1 radius2)} \\
\text{conc descending} \text{and} \text{ (> radius1 radius2)}
\end{align*}

Neighbourhood is tested by the embedding system.
\begin{align*}
\text{prim neighbouring} \\
\text{role neighbour}
\end{align*}

The attributes leftmost and rightmost are used to check neighbourhood.
\begin{align*}
\text{attr leftmost rightmost} \\
\text{conc feature (and (some leftmost trunccone) (some rightmost trunccone))}
\end{align*}

Left and right
\begin{align*}
\text{prim left}
\end{align*}
For long turning surface only necessary conditions can be expressed

\( \text{prim} \text{ lts-sufficient} \)

\( \text{attr radius} \)

\( \text{conc longturningsurface} \)
\( \quad \text{and feature} \)
\( \quad (\geq 0 \text{ radius}) \)
\( \quad (\text{lts-sufficient}) \)

\( \text{conc lts-from-right} \)
\( \quad (\text{and right longturningsurface}) \)

\( \text{conc lts-from-left} \)
\( \quad (\text{and left longturningsurface}) \)

Attributes for long turning surfaces and shoulders

\( \text{attr ground} \)
\( \text{flank} \)
\( \text{depth\#width} \)
\( \text{width} \)

The flank is left to the ground

\( \text{conc flankleft2ground} \)
\( \quad (\text{and} (\leq (\text{flank leftmost center}1)) \)
\( \quad (\text{ground leftmost center}2))) \)

\( \text{conc flankright2ground} \)
\( \quad (\text{and} (\geq (\text{flank leftmost center}1)) \)
\( \quad (\text{ground leftmost center}2))) \)

There are two kinds of shoulders

\( \text{conc shoulder-aux} \)
\( \quad (\text{and composed} \)
\( \quad \text{shoulder-class} \)
\( \quad \text{feature; that leftmost/rightmost are related to} \)
\( \quad \text{leftmost/rightmost of the ground and the flank} \)
\( \quad \text{has to be tested by the embedding system} \)
\( \quad \text{neighbouring} \)
\( \quad \text{some ground longturningsurface} \)
\( \quad (\geq 0 \text{ depth}) \)

\( \text{conc lshoulder} \)
\( \quad (\text{and shoulder-aux} \)
\( \quad \text{flankleft2ground} \)
\( \quad (\text{some flank descending})) \)

\( \text{conc rshoulder} \)
\( \quad (\text{and shoulder-aux} \)
\( \quad \text{flankright2ground} \)
\( \quad (\text{some flank ascending})) \)

\( \text{conc shoulder} \)
\( \quad (\text{and shoulder-aux} \)

\( \text{groove} \)
\( \quad (\text{attr leftflank} \)
\( \quad \text{rightflank}) \)

\( \text{conc groove} \)
\( \quad (\text{and feature} \)
\( \quad \text{groove-class} \)
\( \quad (\text{some leftflank descending}) \)
\( \quad (\text{some rightflank ascending}) \)
\( \quad (\text{some ground longturningsurface}) \)
\( \quad (\geq 0 \text{ depth}\#\text{width}) \)
\( \quad (\geq 0 \text{ depth}) \)
\( \quad (\geq 0 \text{ width}) \)

\( \text{neighbouring} \)

\( \text{pred insertion-condition} \)
\( \quad (\text{ra} (d2w \ d) \ (\text{and} (\geq d2w \ 0 \ 25) (\lt d \ 30))) \)

\( \text{conc insertion} \)
\( \quad (\text{and groove} (\text{insertion-condition} \text{depth}\#\text{width} \text{depth})) \)

\( \text{technological properties of a surface} \)
\( \text{role has-finish} \)
\( \text{attr kind} \)
\( \text{value} \)

\( \text{conc finish} \)
\( \quad (\text{and} (\text{or} (\text{some kind top}) (\text{front-end:u-pred kind})) \)
\( \quad (\text{or} (\text{some value top}) (\text{front-end:u-pred value})))) \)
C RELFUN Sources

C.1 Library Functions

(c) MicroCAD2NC RELFUN Library Functions

C.2 Transforming Aggregated Features to Classified Workpieces

(c) Thomas Giltzen September 1991

变换聚合特征至分类工作件
(leftmost _lm)  (rightmost _rm)

| rest _new-ret
  (compacting _rest _zw-erg)
  (is _new-ret 'tup (groove _fn)
  (leftflank _lf)
  (ground _g)
  (rightflank _rf)
  (leftmost _lm)
  (rightmost _rm))

| rest _new-ret
  (compacting _rest _zw-erg)
  (is _new-ret 'tup (longturningsurface _fn)
  (radius _rad)
  (leftmost _lm)
  (rightmost _rm))

| rest _new-ret
  (compacting _rest _zw-erg)
  (is _new-ret 'tup (longturningsurface _fn)
  (radius _rad)
  (leftmost _lm)
  (rightmost _rm))
C.3 Recursive Workpiece Classification

The program mainly uses RELFUN functions defined by ft clauses which are all deterministic.

classify wp

-----------

(ft (class-feat _wp-rng) ; _wp-rng must be in rng notation
 (is _rad (max-rad _wp-rng))
 (is _class-feat (class-sec-rad _wp-rng))
 ('(tup _rad _class-feat)))

classify section (class-sec <max> <wp-rng>)

-------------

; end at (tup) or (tup _x)
(ft (class-sec_max (tup)) !
 'tup))

(ft (class-sec_max (tup _x)) ! ; _x thrown out because a single supporting
 'tup)) ; point cannot become a feature

; ignore max height cylinders
(ft (class-sec_max (tup (rng _r1 _r1_max) (rng _r2 _r2_max _ro)) ! _rest)) !
 (class-sec_max 'tup (rng _r2 _r2_max _ro) ! _rest))))

; groove: begins with max and split-wp-rng succeeds
(ft (class-sec_max (tup (rng _r1 _r1_max) _rest))
 ; split-wp-rng_max (tup _left-wp-rng) ; check for and return part
 (is (tup _left-wp-rng) ! _rest-wp-rng) ; up to maximum
 (tup _left-wp-rng))
 ;((rf-print groove)
 ;(rf-print '(tup (rng _r1 _r1_max) _rest))
 ;(rf-print '(tup (rng _r1 _r1_max) _left-wp-rng))
 (tup _left-wp-rng))
 (class-sec_max 'tup (rng _r2 _r2_max _ro) ! _left-wp-rng))

; right shoulder: doesn't start with max and split-wp-rng succeeds
(ft (class-sec_max (tup (rng _r1 _r1_max) _rest))
 (is (tup _left-wp-rng) ! _rest-wp-rng)
 (split-wp-rng_max 'tup (rng _r1 _r1_max) ! _rest)))) !

;((rf-print right-shoulder)
 ;(rf-print '(tup (rng _r1 _r1_max) _rest)))
C RELFUN SOURCES

5/S

; (rf-print _left-vp-rng)
(tup (class-rah _left-vp-rng) | (class-sec _max _rest-vp-rng))

; left shoulder: no more maxima
(ft (class-sec _max _vp-rng)
 ; (rf-print left-shoulder)
 ; (rf-print _wp-rng)
(tup (class-lah _wp-rng)))

; classify grooves
----------------------

(ft (class-gv _wp-rng)
 (is (tup _test-flank1 | _rest-wp) (find-flank _wp-rng 0))
 (is (tup _test-flank2 | _rest-wp2) (find-flank (reverse-vp-rng _rest-wp1) 0))
 (is (rng _tz _tri _test-flank1-min) (rf-last _test-flank1))
 (is _ground-max (max _test-flank1-min (max-rad _rest-wp2)))
 (is (tup _l-flank | _rest) (find-flank _wp-rng _ground-max))
 (is (tup _r-flank | _ground) (find-flank (reverse-vp-rng _rest _ground-max))
 (is _r-flank-rev (reverse-vp-rng _r-flank))
 (asf (gv (flk _l-flank) _grd (flk _r-flank-rev)) _refined-ground)

; classify left shoulder
----------------------

(ft (class-lsh _wp-rng)
 (is (tup _test-flank | _rest-wp) (find-flank _wp-rng 0))
 (is (rng _tz _tri _test-flank-min) (rf-last _test-flank))
 (is _ground-max (max _test-flank-min (max-rad _rest-wp2)))
 (is (tup _flank | _ground) (find-flank _wp-rng _ground-max))
 (is _r-flank-rev (reverse-vp-rng _r-flank))
 (asf (lsh (flk _flank) _grd _refined-ground))

; classify right shoulder
----------------------

(ft (class-rah _wp-rng)
 (reverse-feature (class-lsh (reverse-vp-rng _wp-rng))))

; classify ground
----------------------

(hn (class-grd (tup) _ground-max (grd (tup)) (tup) !)

(hn (class-grd (tup) _ground-max (grd (tup)) (tup) !)

(hn (class-grd _ground _ground-max

C 3 Recursive Workspace Classification

; find flank (find-flank <vp-rng> <min>)
----------------------

; (a) sec. ring
(ft (find-flank (tup _z _r1 ro) _rest-wp) _min)
 ; (rf-print a)
 (> _ro _r1) !
 ( (tup (rng _z _r1 ro)) | (tup (rng _z _ro _ro) | _rest-wp))

; (b) desc. ring passing or down to min
(ft (find-flank (tup _z _r1 ro) _rest-wp) _min)
 ; (rf-print b)
 (> _min _ro) !
 ( (tup (rng _z _r1 ro) | (tup (rng _z _min _ro) | _rest-wp))

; (c) desc. last ring above min
(ft (find-flank (tup _z _r1 _ro) _min) !
 ; (rf-print c)
 (> _ro _r1) (> _ro _min) holds
 (tup (rng _z _r1 _ro) | _rest-wp))

; (d) sec. trunc. cone
(ft (find-flank (tup _z1 _r11 _rol) (rng _z2 _r12 _ro2) _rest-wp) _min)
 ; (rf-print d)
 (> _r12 _ro1) !
 ( (tup (rng _z1 _r11 _rol) | (tup (rng _z2 _r12 _ro2) | _rest-wp))

; (e) desc. trunc. cone down to min
(ft (find-flank (tup _z1 _r11 _rol) (rng _z2 _r12 _ro2) _rest-wp) _min)
 ; (rf-print e1)
 (> _rol _r12) !
 ( (tup (rng _z1 _r11 _rol) | (rng _z2 _rol _ro2) | _rest-wp))

; (f) cylinder or desc. trunc. cone above min

; (f) cylinder or desc. trunc. cone above min

(ft (find-flank (tup (rng _zi _ri1 _ro1) (rng _zi2 _ri2 _ro2) | rest) min)
 ;(rf-print f)
 ; fail through case: (=> _ri1 _ro1 _ri2) holds
 (is (tup _rest-flank | _rest-wp)
  (find-flank (tup (rng _zi _ri1 _ro1) | _rest-wp)
  '(tup (tup (rng _zi _ri1 _ro1) | _rest-flank) | _rest-wp))
 mystery

 ; reverse feature & reverse feature list
 ; -----------------------------

 (ft (reverse-feature (nft (lab (flk_flank) (grd_ground)) _refined-ground))
  (is _flank_rev (reverse-wp-rng_flank))
  (is _ground_rev (reverse-wp-rng_ground))
  (is _refined-ground_rev (reverse-feature-list _refined-ground))
  '(nft (lsh (grd_ground-rev) (flk_flank-rev)) _refined-ground-rev))

 C.3 Recursive Workpiece Classification

 (ft (reverse-feature (nft (grv (flk_flank)) (grd_ground) (flk_flank2))
  (is _flank1_rev (reverse-wp-rng_flank1))
  (is _ground1_rev (reverse-wp-rng_ground1))
  (is _flank2_rev (reverse-wp-rng_flank2))
  (is _refined-ground_rev (reverse-feature-list _refined-ground))
  '(nft (grv (flk_flank2-rev) (grd_ground-rev) (flk_flank1-rev))
   _refined-ground-rev))

 (ft (reverse-feature-list (tup))
  '(tup))
 (ft (reverse-feature-list (tup _h _t))
  (appfun (reverse-feature-list _t) (tup (reverse-feature _h))))

 ; auxiliary functions
 ; -------------------

 (ft (split-wp-rng_max (tup (rng _zi _ri _ro) | rest))
  '(tup (tup (tup (rng _zi _ri _ro) | rest))
   (tup (rng _zi _ri _ro) | rest)))
 (ft (split-wp-rng_max (tup (rng _zi _ri _ro) | rest))
  '(tup (tup (rng _zi _ri _ro) | rest)))
 (ft (split-wp-rng_max (tup (rng _zi _ri _ro) | rest))
  '(tup (tup (rng _zi _ri _ro) | _left-wp-rng)
   (split-wp-rng_max_rest)))
 (ft (split-wp-rng_max (tup (rng _zi _ri _ro) | rest))
  '(tup (tup (rng _zi _ri _ro) | _left-wp-rng)
   (split-wp-rng)))

 (ft (split-truncmax _zi _ri _ro _min)
  (is_h (_ _ _ _))
  (is_h (_ _ _ _)
   (+ _zi (*/ (- (neg _z2 _zi) (_ _ _ _))) _h))
 (ft (max-rad (tup)) 0)
 (ft (max-rad (tup (rng _zi _ri _ro) | rest-rng-list))
  (max _ri _ro (max-rad _rest-rng-list)))
 (ft (reverse-pre-wp_r (tup))
  'tup))
 (ft (reverse-pre-wp (tup (rng _zi _ri _ro) | rest))
  (is _mirr-z (_ 0 _z))
  (appfun (reverse-pre-wp _rest) 'tup (neg _mirr-z _ri)))
C.4 Transforming CWP’s from rng-Notation to p-Notation

```lisp
(ft (cwp-rng2p (cwp_globals _class-feat-rng))
   (is _class-feat-p (rng2p-feature-list _class-feat-rng))
   '(_cwp_globals _class-feat-p))

(ft (rng2p-feature (nft (lsh (flk _flank) (grd _ground)) _refined-ground))
   (is _flank-p (rng2p _flank))
   (is _ground-p (rng2p _ground))
   (is _refined-ground-p (rng2p-feature-list _refined-ground))
   '(_nft (lsh (flk _flank-p) (grd _ground-p)) _refined-ground-p))

(ft (rng2p-feature (nft (rsh (grd _ground) (flk _flank)) _refined-ground))
   (is _flank-p (rng2p _flank))
   (is _ground-p (rng2p _ground))
   (is _refined-ground-p (rng2p-feature-list _refined-ground))
   '(_nft (rsh (grd _ground-p) (flk _flank-p)) _refined-ground-p))

(ft (rng2p-feature (nft (grv (flk _flank) (grd _ground) (flk _flank2))
                           _refined-ground))
   (is _flank1-p (rng2p _flank1))
   (is _flank2-p (rng2p _flank2))
   (is _ground-p (rng2p _ground))
   (is _refined-ground-p (rng2p-feature-list _refined-ground))
   '(_nft (grv (flk _flank1-p) (grd _ground-p) (flk _flank2-p))
          _refined-ground-p))

(ft (rng2p-feature-list (tup))
   '(tup))

(ft (rng2p-feature-list (tup _h _t))
   '(tup (rng2p-feature-list _t))
   (tup (rng2p-feature-list _h) (rng2p-feature-list _t)))

(ft (rng2p (tup)) '('(tup)))

(ft (rng2p (tup _uri _ro) _rest))
   (tup '(p _x _ri) _uri _ro)))

(ft (rng2p (tup _uri _ro) _rest))
   (tup '(p _x _ro) _uri _ro)))
```

C.5 Skeletal Plans

```lisp
# Input: classified workpieces

class-wp ::= (cwp <globals> <feature-list>)

globals ::= "set of global annotations; momentarily the max radius; should additionally contain the material and the surface quality"

feature-list ::= (tup { <nested-feature> })

nested-feature ::= (nft <feature> <feature-list>)

feature ::= <leftshoulder> | <rightshoulder> | <groove> |

leftshoulder ::= (lsh <flank> <ground>)

rightshoulder ::= (rsh <ground> <flank>)

groove ::= (grv <flank> <ground> <flank>)

flank ::= (flk <coordinates>)

ground ::= (grd <coordinates>)

coordinates ::= (tup { <point> })

point ::= (p <num> <num>) ; z-coordinate and radius
```
Output: Skeletal Plans

---

```
(skeletal-plan := (skp <globals> <sac-plan>)
     ; sac = sequential/alternative/commutative

plan ::= (tup) ; empty plan
    | <action> ; atomic action
    | (seq <plan-list>) ; sequential
    | (com <plan-list>) ; commutative
    | (alt <plan-list>) ; alternative

plan-list ::= (tup { <plan> })

action ::= (roughing) | (finishing) | ...

roughing ::= (roughing <tool> <way> <geometry>)

tool ::= (tool <plate> <holder>)

way ::= left | right ; means: working into this direction

gometry ::= (geo <coordinates>)

---

1. create skeletal plan

---

(ft (skeletal-plan <wp _globals _features>)
    (is _skp (skp/com _features))
    '(skp <globals> _skp)

; skp/com

---

(ft (skp/com (tup)) !
    '(tup))

(ft (skp/com (tup _h _t))
    (norm-com (skp/cfeat _h) (skp/com _t)))

; skp/cfeat (complex features like HST = nested feature)
```

---

C.5 Skeletal Plans

---

```
(ft (skp/cfeat (nft . feat _refined-feat))
    (norm-seq (skp/cfeat _feat) (skp/com _refined-feat)))

; additional complex/nested features ...

; skp/feat (for unnested features like ish, rsh, grw)

---

(ft (skp/feat (ish (flk _flank) (grd _ground))))
    (skp/ish _flank _ground)

(ft (skp/feat (rsh (grd _ground) (flk _flank)))
    (skp/rsh _ground _flank))

(ft (skp/feat (grw (flk _flank) (grd _ground) (flk _flank2)))
    (skp/grw _flank _ground _flank2)))

; skeletal plans for unnested features

---

; global constants for demo version:

(ft (wp-material) high-alloy-steel)

(ft (quality) normal)

---

; skp/ish

---

(ft (skp/ish _flank _ground)
    (is _fl-gr (append-coord _flank _ground))
    (tool-sel roughing (wp-material) (quality)
        0 (rf-max (compute-flank-angles _flank))
        left _tools)
    (check-alternatives
        (geo-roughing-alternatives _tools left '(geo _fl-gr))))

; skp/rsh

---

(ft (skp/rsh _ground _flank)
    (is _gr-fl (append-coord _ground _flank))
    (tool-sel roughing (wp-material) (quality)
        0 (rf-max (compute-flank-angles (reverse-coord _flank))
            right _tools))
```
(check-alternatives
  (gen-roughing-alternatives tools right 'geo _gr-f1)))

; exp/grv
; -------

(ft (exp/grv _flankl _ground _flank2)
  (is _fl-gr-f1 (append-coord _flank1 (append-coord _ground _flank2)))
  (is _max1 (rf-max (compute-flank-angles _flank1)))
  (is _max2 (rf-max (compute-flank-angles (reverse-coord _flank2)))
  (tool-sel roughing (wp-material) (quality)
    _max1_max2 right _tools)
  (tool-sel roughing (wp-material) (quality)
    _max2_max1 left _tools2)
  (is _right-alt (check-alternatives (gen-roughing-alternatives
    _tools) right 'geo _fl-gr-f1)))
  (is _left-alt (check-alternatives (gen-roughing-alternatives
    _tools) left 'geo _fl-gr-f1)))
  (norm-alt (norm-seq _left-alt _right-alt)
    (norm-seq _right-alt _left-alt)))

; additional skeletal plans ...

C.5 Skeletal Plans

; tool-sel (tool selection)
; ----------------------------------------------

; format:
; (tool-sel <process> <wp-material> <quality> <alpha> <beta> <way>
;             (tup { (tup plate <holder>) }+ )

; place precomputed tool selections here ...

; CONTAl tool selection:

  (hn (tool-sel _process wp-material _quality _alpha _beta _direction
    _tools)
    (rf-print context\ tool\ selection:\)
    (rf-print \ arguments\)
    (rf-print _process\)
    (rf-print wp-material\)
    (rf-print _quality\)
    (rf-print _alpha\)
    (rf-print _beta\)
    (rf-print _direction\)
    (rf-print _tools)
    (rf-print \ propagating\)
    (hn tool-sel (get-cn-tools)
      (cn global hard
        (tup quality _quality)
        (tup wp-material wp-material
          (tup process _process)
        (tup alpha _alpha)
        (tup beta _beta)
        (tup direction _direction)
        (tup tool lathe-tools)
        (tup holder holders)
        (tup plate plate-geometries)
        (tup edge-angle edge-angles)
        (tup tc-edge-angle tc-edge-angles))
      tool holder))
    (rf-terpri)
    (rf-print \ results:\)
    (rf-print _tools)
    (rf-terpri) (rf-terpri))
(an (init-tool-awl))

(rf-print const\ initialization:\)

(rf-print \ \ creating\ constraint\ variables\)
; (cn cv quality hcl (tup qualities))
(cn cv wp-material hcl (tup wp-materials))
(cn cv process hcl (tup processes))
(cn cv tool hcl (tup lathe-tools))
(cn cv holder hcl (tup holders))
(cn cv alpha hcl (tup acute-angles))
(cn cv beta hcl (tup acute-angles))
(cn cv direction hcl (tup directions))
(cn cv plate hcl (tup plate-geometries))
(cn cv edge-angle hcl (tup edge-angles))
(cn cv tc-edge-angle hcl (tup tc-edge-angles))

(rf-print \ \ creating\ constraint\ instances\)
(cn ci inst-ho-to hard holder-tool holder tool)
(cn ci inst-pr-ho hard process-holder process holder)
(cn ci inst-ho-de1 hard holder-desc1 holder direction)
(cn ci inst-ho-de2 hard holder-desc2 holder tc-edge-angle)
(cn ci inst-ho-de3 hard holder-desc3 holder plate)
(cn ci inst-pr-ma-to hard process-material-tool
 process wp-material tool)
(cn ci inst-pl-ea hard plate-angle plate edge-angle)
(cn ci inst-pr-ea hard process-angle process edge-angle)
; (cn ci inst-te-hata hard tc-te-aal tc-edge-angle
 ; edge-angle alpha)
(cn ci inst-te-hata hard tc-te-aal tc-edge-angle hata)

(rf-print ok.)
(rf-terminate)
2. skp access/transform functions

;=============================================

; a) get-skp-top
;--------------

(ft (get-skp-top (skp-globals _sac-plan))
 (skp-top/plan _sac-plan))

(ft (skp-top/plan (tup))
 '('(tup))

(ft (skp-top/com (com _com))
 (skp-top/com _com))

(ft (skp-top/seq (seq _seq))
 (skp-top/seq _seq))

(ft (skp-top/alt (alt _alt))
 (skp-top/alt _alt))

(ft (skp-top/plan _action)
 _action)

(ft (skp-top/com (tup))
 '('(tup))

(ft (skp-top/com (tup _h _t))
 (norm-com (skp-top/plan _h) (skp-top/com _t)))

(ft (skp-top/seq (tup))
 '('(tup))

(ft (skp-top/seq (tup _h _t))
 (skp-top/plan _h))

(ft (skp-top/alt (tup))
 '('(tup))

(ft (skp-top/alt (tup _h _t))
 (norm-alt (skp-top/plan _h) (skp-top/alt _t)))

; b) execute-skp-action
;---------------------

; skp-exec-action (skp-exec-action <skp> <ac:spec>)
(skp-exec-action (skp-globals _actspec))
  (is (tup _actions _rest-plan _costs) (skp-exec/plan _actspec))
    (tup _actions (skp-globals _rest-plan _costs))

(ft (skp-exec/plan (tup) _actspec)
  ('(tup (tup) (tup) 0))

(ft (skp-exec/plan (com _com) _actspec)
  (skp-exec/com _com _actspec))

(ft (skp-exec/plan (seq _seq) _actspec)
  (skp-exec/seq _seq _actspec))

(ft (skp-exec/plan (alt _alt) _actspec)
  (skp-exec/alt _alt _alt _actspec))

(ft (skp-exec/plan _action _actspec)
  (skp-exec/act _action _actspec))

(ft (skp-exec/com (tup) _actspec)
  ('(tup (tup) (tup) 0))

(ft (skp-exec/com (tup) _actspec)
  (is (tup _actions _rest-plan _costs) (skp-exec/plan _actspec))
    (is (tup _actions _t-rest-plan _t-costs) (skp-exec/com _t _actspec))
      (tup (appfun _h-actions _t-actions)
        (norm-com _h-rest-plan _h-costs)
        (+ _h-costs _t-costs))))

;

; -> (tup <action-list> <rest-plan> <costs>)
; -------------------------------

(ft (skp-exec/seq (tup) _actspec)
  ('(tup (tup) (tup) 0))

(ft (skp-exec/seq (tup _h) _actspec)
  (skp-exec/seqi (skp-exec/plan _actspec) _t _actspec))

(ft (skp-exec/seqi (tup _h-actions (tup) _h-costs) _t _actspec)
  (is (tup _t-actions _t-rest-plan _t-costs) (skp-exec/com _t _actspec))
    (tup (appfun _h-actions _t-actions)
      _t-rest-plan
      (+ _h-costs _t-costs))

(ft (skp-exec/seqi (tup _h-actions _h-rest-plan _h-costs) _t _actspec)
  (tup _h-actions
    (norm-seq _h-rest-plan (norm-rest-seq _t))
    _h-costs))

(ft (skp-exec/alt _alt (tup) _actspec) ! ; so more alternatives to check
  ('(tup (tup) (tup) (alt _alt) 0))

(ft (skp-exec/alt (tup _h) _t _actspec)
  (skp-exec/alti _alt (skp-exec/plan _h _actspec) _b _t _actspec))

(ft (skp-exec/alti _alt (tup _h-actions _h-costs) _b _t _actspec)
  (skp-exec/alti _alt _t _actspec)); try remaining alternatives

(ft (skp-exec/alti _alt (tup _h-actions _b-rest-plan _b-costs) _h _t _actspec)
  (tup _h-actions _h-rest-plan _h-costs)); alternative found

(ft (skp-exec/act (roughing_tool _way _geo) _tool)
  ('(tup (tup (roughing_tool _way _geo)) (tup) 1)); costs: 1

(ft (skp-exec/act _action _actspec)
  ('(tup (tup) _action 0))
(c) extract-actions (extract-actions <skp>)

; extract-actions (extract-actions <skp>)

(ft (extract-actions (tup)) !

'(tup))

(ft (extract-actions (com _com)) !

(extract-actions/list _com))

(ft (extract-actions (seq _seq)) !

(extract-actions/list _seq))

(ft (extract-actions (alt _alt)) !

(extract-actions/list _alt))

(ft (extract-actions _action)

'(tup _action))

(ft (extract-actions/list (tup _t))

'(tup))

(ft (extract-actions/list (tup _h | _t))

(appfun (extract-actions _h) (extract-actions/list _t)))

(d) get-tools (from action-list; removing duplicates)

; get-tools (from action-list; removing duplicates)

(ft (get-tools (tup)) !

'(tup))

(ft (get-tools (tup (roughing _tool _way _geo) | _rest))

(is _rest-tools (get-tools _rest))

(add-if-new _tool _rest-tools))

; (e) count-com-actions

; count-com-actions

(ft (count-com-actions (skp _globals _skp-plan))

(cca/plan _skp-plan))

(ft (cca/plan (tup)) !

0)

(ft (cca/plan (com _com)) !

(cca/com _com))

(ft (cca/plan (seq (tup _h | _t)) !

(cca/plan _h))

(ft (cca/plan (alt (tup _h | _t))) !

; only consider first alternative!

(cca/plan _h))

(ft (cca/plan _action)

1)

(ft (cca/com (tup)) !

0)

(ft (cca/com (tup _h | _t))

(+ (cca/plan _h) (cca/com _t)))
; 3. exp normalizations
; -------------------
; norm-com (norm-com (plan1) (plan2))
; -------------------------------

(ft (norm-com (tup) _plan) ! _plan)

(ft (norm-com _plan (tup)) !
 _plan)

(ft (norm-com (com _com1) (com _com2)) !
(is _com12 (appfun _com1 _com2))
'(com _com12))

(ft (norm-com _plan (com _com)) !
'(com (tup _plan _com)))

(ft (norm-com (com _com)) !
'(com _plan _com))

(ft (norm-com _plan _plan2) !
'(com (tup _plan _plan2)))

; norm-seq (norm-seq (plan1) (plan2)) & (norm-rest-seq <tup>)
; -------------------

(ft (norm-seq (tup) _plan) ! _plan)

(ft (norm-seq _plan (tup)) !
 _plan)

:(ft (norm-seq (seq _seq1) (seq _seq2)) !
(is _seq12 (appfun _seq1 _seq2))
'(seq _seq12))

(ft (norm-seq _plan (seq _seq)) !
'(seq (tup _plan _seq)))

(ft (norm-seq (seq _seq) _plan) !
(is _sp (appfun _seq '(tup _plan)))
'(seq _sp))

(ft (norm-seq _plan _plan2) !
'(seq (tup _plan _plan2)))

; norm-alt (norm-alt (plan1) (plan2))
; -------------------------------

(ft (norm-alt (tup) _plan) ! _plan)

(ft (norm-alt _plan (tup)) !
 _plan)

(ft (norm-alt (alt _alt1) (alt _alt2)) !
(is _alt12 (appfun _alt1 _alt2))
'(alt _alt12))

(ft (norm-alt _plan (alt _alt)) !
'(alt _alt))

(ft (norm-alt (alt _alt) _plan) !
(is _ap (appfun _alt '(tup _plan)))
'(alt _ap))

(ft (norm-alt _plan _plan2) !
'(alt (tup _plan _plan2)))

; (norm-rest-seq <tup>) : <tup> is expected to be the rest of a
; normalized sequence!

(ft (norm-rest-seq (tup)) !
'(tup))

(ft (norm-rest-seq (tup _plan)) !
 _plan)

(ft (norm-rest-seq _tup) !
'(seq _tup))

; norm-alt (norm-alt (plan1) (plan2))
; -------------------------------

(ft (norm-alt (tup) _plan) ! _plan)

(ft (norm-alt _plan (tup)) !
 _plan)

(ft (norm-alt (alt _alt1) (alt _alt2)) !
(is _alt12 (appfun _alt1 _alt2))
'(alt _alt12))

(ft (norm-alt _plan (alt _alt)) !
'(alt (tup _plan _alt)))

(ft (norm-alt (alt _alt) _plan) !
(is _ap (appfun _alt '(tup _plan)))
'(alt _ap))

(ft (norm-alt _plan _plan2) !
'(alt (tup _plan _plan2)))
C.6 ANC-Plan Generation

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; microCADMC ;;;
;;; BELFUN part ;;;
;;; ANC-Plan Generation ;;;
;;; (c) Michael Sintek September 1991 ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Input: skeletal-plans

; Output: abstract NC programs:

anc-program ::= (tup {<action>})

(ft (gen-anc-plan (skp _globals (tup)))
 ' (tup))

(ft (gen-anc-plan _skp)
 (get-promising-tool
 no-tool 0 no-actions no-rest-skp
 (get-tools (extract-actions (get-skp-top_skp))) _skp
 _best-tool _best-costs _best-actions _best-rest-skp
 (rf-pprint _best-actions)
 (rf-pprint _best-costs)
 (rf-pprint =>)
 (rf-terpri)
 (rf-pprint _best-rest-skp)
 (rf-terpri) (rf-terpri)
 (appfun _best-actions (gen-anc-plan _best-rest-skp)))
C.6 ANG-Plan Generation

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp))

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)

(get-promising-tool
  _best-tool-so-far _best-costs-so-far
  _best-actions-so-far _best-rest-skp-so-far
  _rest-tools _skp
  _best-tool _best-costs _best-actions _best-rest-skp)
D  CONTAX  Sources

D.1 Tools Database

; Part 1: domain definitions

; definitions of domains for tools (cutting plates)

(dd lathe-tools ( finishing-tools roughturn-tools ))

(dd roughturn-tools ( universal-tools ma71 nma m61 ))

(dd finishing-tools ( universal-tools ma53 cma ))

(dd universal-tools ( nmg mm52 BCRM ))

(dd ma71 ( DMMM-71 THMM-71 SHMM-71 CHMM-71 ))

(dd m61 ( THMM-41 SHMM-41 DMMM-41 CHMM-41 ))

(dd ma53 ( TCM-63 DCMN-63 SCMN-63 VBMN-63 ))

(dd mm53 ( TCMN-63 DCM-63 SCM-63 VBM-63 ))

(dd cma ( TCMN DCMN SCMN CMN ))

(dd nma ( TNNM DNNM SNMN CNMN ))

(dd nmg ( RMG THMG SHMG DHMG CHMG ))
(dd pa
  TMAIP-PSL76
  TMAIP-PSL46
  TMAIP-PSR86
  TMAIP-PSR76
  TMAIP-PSR46
))

(dd pc
  TMAIP-PCL96
  TMAIP-PCL76
  TMAIP-PCK66
  TMAIP-PCK86
  TMAIP-PCR76
  TMAIP-PCR86
))

(dd pr
  TMAIP-PRL30
  TMAIP-PRL40
  TMAIP-PRM
  TMAIP-PRR30
))

(dd pd
  TMAIP-PDL93
  TMAIP-PDR93
))

;; TMAI-U holders

;;(dd tmax-u ( at ss av sr TMAIU-SCM95 TMAIU-SDL93 ))
#]
(dd tmax-u ( at ss sr TMAIU-SCM95 TMAIU-SDL93 ))

(dd st
  TMAIU-STL90
  TMAIU-STL75
  TMAIU-STN60
  TMAIU-STN46
  TMAIU-STR90
  TMAIU-STR75
))

(dd ss
  TMAIU-SRM75
  TMAIU-SRM45
  TMAIU-SDL76
  TMAIU-SDL46
))

(dd sv
  TMAIU-SVL93
  TMAIU-SVW72
  TMAIU-SVR93
))

(dd sr
  TMAIU-SRM
  TMAIU-SRJ30
))

[; definition of work processes]

(dd processes (roughing finishing))

;; definition of required workpiece qualities

(dd qualities ( normal critical ))

;; domain definitions for workpiece materials

(dd wp-materials ( steel cast alu ))

(dd steel ( building-steel alloy-steel stainless-steel ))

(dd cast ( GG GGG ))

(dd alloy-steel ( low-alloy-steel high-alloy-steel ))

;; domain definitions of relevant angles

(dd acute-angles ( 0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57
  60 63 66 69 72 75 78 81 84 87 90 )
)

;; definition of tool cutting edge angles

(dd tc-edge-angles ( 0 small-tc-angles big-tc-angles ))
(dd small-te-angles (45 60 72 75))
(dd big-te-angles (90 93 96))
;; edge-angle of the cutting plate

(dd edge-angles (0 small-edge-angles medium-edge-angles big-edge-angles))
(dd small-edge-angles (0 36))
(dd medium-edge-angles (56 60))
(dd big-edge-angles (0 90 90))
;; domain definitions of cutting-plate geometry

;;(dd plate-geometries (R S T D V C K))
(dd plate-geometries (R S T D C K))
;; definition of cutting directions, which can be to left or to right

(dd directions (left right))

;; Part 2: constraint definitions
;; holder-tool describes the cutting-plates fitting to several holders for
;; reasons of geometry

(pc holder-tool (ho to) (holders lathe-tools))
(pt TMM-71)
(pt TMM-41)
(pt TMMG)

(pc SMM-71)
(pc SMM-41)
(pc SMMG)

(pc CMM-71)
(pc CMM-41)
(pc CMMG)
(pc CMA)

(pr DCM)
(pr DMMG)

(pr RCMG)

(pd DMMG)

(pd DMM-71)

(pd DMM-41)

(at TCMM-53)

(at TCMM-62)

(at TCMG)

(as CHMM-53)

(as CHMM-52)

(as CHMG)

(as CHM)

(ar CMM-62)

(ar VMM-63)

(TMAI-U-SCRM CCM-52)

(TMAI-U-SCRM CCM-53)

(TMAI-U-SCRM CCMG)

(TMAI-U-SL93 DCM-52)

(TMAI-U-SL93 DCM-53)

(TMAI-U-SL93 DCMA )

;; process-holder describes the fact that TMAI-P holders are favourable for
;; roughing, whereas TMAI-U holders should be preferred for finishing. If
quality restrictions are not so critical, for finishing, MA1-S holders may be used.

(pc process-holder (pr ho) (processes holders) (roughing tmx-p) (finishing tmx-u ))

(pc holder-desc1 (ho di) (holders directions) (MAIP-PL90 left) (MAIP-PR90 right) (MAIP-PL80 left) (MAIP-PR80 right) (MAIP-P860 directions) (MAIP-PL45 left) (MAIP-PR45 right) (MAIP-P876 left) (MAIP-PS76 right) (MAIP-P846 right) (MAIP-PS46 directions) (MAIP-PCL96 left) (MAIP-PCL86 right) (MAIP-PL76 left) (MAIP-PS76 right) (MAIP-P765 right) (MAIP-P456 right) (MAIP-PL93 left) (MAIP-PR93 right) (MAIP-PL30 left) (MAIP-PR30 right) (MAIP-PL40 left) (MAIP-P460 right) (MAIP-PL46 left) (MAIP-PR46 right) (MAIP-PL45 left) (MAIP-PR45 right) (MAIP-PS45 directions) (MAIP-PCL93 left) (MAIP-PSL76 left) (MAIP-P456 right) (MAIP-P983 left) (MAIP-P456 right) (MAIP-P20 0) (MAIP-P830 0) (MAIP-P90 0) (MAIP-P460 0) (MAIP-P93 0) (MAIP-P93 0)

(pc holder-desc2 (ho tc) (holders tc-edge-angles) (MAIP-PL90 90) (MAIP-PR90 90) (MAIP-PL80 80) (MAIP-PR80 80) (MAIP-PL60 60) (MAIP-PR60 60) (MAIP-P45 45) (MAIP-P45 45) (MAIP-P85 95) (MAIP-P85 95) (MAIP-P876 76) (MAIP-P456 66) (MAIP-P456 66) (MAIP-P876 76) (MAIP-P456 66) (MAIP-P876 76) (MAIP-P456 66) (MAIP-P983 93) (MAIP-P983 93) (MAIP-P90 0) (MAIP-P830 0) (MAIP-P90 0) (MAIP-P460 0) (MAIP-P93 0) (MAIP-P93 0)

;; (MAIP-PR directions)

;; (MAIU-STR60 left 90 T) (MAIU-STR60 right 90 T) (MAIU-STR75 left 75 T) (MAIU-STR75 right 75 T) (MAIU-STR60 directions 60 T) (MAIU-STR45 directions 45 T) (MAIU-STR76 left 75 S) (MAIU-STR76 right 75 S) (MAIU-STR45 directions 45 S) (MAIU-STR90 directions 90 S) (MAIU-STR90 directions 90 C) (MAIU-STR90 directions 90 S) (MAIU-STR90 directions 90 C) (MAIU-STR90 directions 90 S) (MAIU-STR90 directions 90 C) (MAIU-STR90 directions 90 S) (MAIU-STR90 directions 90 C) (MAIU-STR90 directions 90 S) (MAIU-STR90 directions 90 C)
| (TMAU-SVL93 left 93 V)                  | (TMAU-SVR93 right 93 V)                  |
| (TMAU-SVW72 directions 72 V)          | (TMAU-SWH directions 0 R)                |
| (TMAU-SNH directions 0 R)             | (TMAU-SNH30 directions 0 R)              |
| (pc holder-desc3 (no pl) (holders plate-geometries) |
| (TMAIP-PTL90 T)                       | (TMAIP-PTR90 T)                         |
| (TMAIP-PTO90 T)                       | (TMAIP-PTR90 T)                         |
| (TMAIP-PTW60 T)                       | (TMAIP-PTL45 T)                         |
| (TMAIP-PTL45 T)                       | (TMAIP-PTR45 T)                         |
| (TMAIP-PSL76 S)                       | (TMAIP-PSR76 S)                         |
| (TMAIP-PSL45 S)                       | (TMAIP-PSR45 S)                         |
| (TMAIP-PSH45 S)                       | (TMAIP-PSR45 S)                         |
| (TMAIP-PCL96 C)                       | (TMAIP-PCL96 C)                         |
| (TMAIP-PCL76 C)                       | (TMAIP-PCL76 C)                         |
| (TMAIP-PCL66 C)                       | (TMAIP-PCL66 C)                         |
| (TMAIP-PDL93 D)                       | (TMAIP-PDR93 D)                         |
| (TMAIP-PRL40 R)                       | (TMAIP-PRL40 R)                         |
| (TMAIP-RN R)                          | (TMAIP-RN R)                            |
| (TMAU-STR90 right 90 T)               | (TMAU-STR75 left 75 T)                  |
| (TMAU-STR75 left 75 T)                | (TMAU-STR75 left 75 T)                  |
| (TMAU-STR460 directions 60 T)         | (TMAU-STR46 directions 60 T)             |
| (TMAU-SSL76 left 75 S)                | (TMAU-SSL46 left 75 S)                  |
| (TMAU-SSL46 left 75 S)                | (TMAU-SSL45 directions 45 S)             |
| (TMAU-SSL45 directions 45 S)          | (TMAU-SSL95 directions 95 C)             |
| (TMAU-SDL93 left 93 D)                | (TMAU-SDL93 left 93 D)                  |
process-material-tool specifies the usability of cutting-plates w.r.t. the
working-process to be done and the properties of materials.

The constraint reflects the suitability of the cutting-plate materials
which are implicitly contained in their names) for certain workplace
materials, e.g. short-cutting, long-cutting, stainless, warmfast, 777-hard,hard.

(process-material-tool (pr ma to) (processes vp-materials lathe-tools)
(roughing steel ma71)
(roughing cast ma71)
(roughing stainless-steel ma61)
(roughing low-alloy-steel ma62)
(finishing low-alloy-steel ma53)
(finishing steel ma52)
(finishing cast ma52)
(processes alu umg)
(processes steel RCHX)
(processes cast RCHX))

(process-material-tool specifies the usability of cutting-plates w.r.t. the
working-process to be done and the properties of materials.

The constraint plate-angle maps the cutting-plates to their edge-angles

(plate-angle (to pl) (plate-geometries edge-angles)
(R 0)
(S 90)
(T 80)
(D 60)
(V 36)
(C 60)
(R 55))

The constraint process-angle gives expression to the fact that roughing
should be performed using a big edge angle whereas for finishing, a small
edge angle is appropriate. Round plates (with edge-angle 0) can be used
both for roughing and finishing.

(process-process-angle (pr ea) (processтs edge-angles)
(roughing small-edge-angles)
(roughing medium-edge-angles)
(roughing big-edge-angles)
(finishing medium-edge-angles)
(finishing big-edge-angles))

The constraint tc-ea-al gives expression to the requirement that
the sum of the tool-cutting-edge angle, the tool-edge-angle and the
angle alpha must be ≤ 180 degrees

(tc tc-ea-al tc ea al) (tc-edge-angles edge-angles acute-angles) le180)
(tc tc-beta tc bt) (tc-edge-angles acute-angles) ge-or-zero)
D.2 CONTAX Lisp Functions

(defun first-char (atom res)
  (cond
   ((null atom) nil)
   (t (eq res (intern (substring (string atom) 0 1))))))

(defun le180 (W1 W2 W3)
  (<= (+ (+ W1 W2) W3) 180))

(defun ge-plus-6 (W1 W2)
  (>= W1 (+ W2 6)))

(defun ge-or-zero (W1 W2)
  (or
   (>= W1 W2)
   (equal W1 0)
  ))
E Figures

μCAD2NC: Where and How COLAB is used in a Workplanning Model

COLAB Subsystems cooperating in μCAD2NC
Feature Recognition in TAXON and FORWARD
Constraint Net used for tool selection in μCAD2NC

Feature Abstraction with Forward Reasoning and Terminological Reasoning over Concrete Domains
ANC-Program
(Roughing Phase)

```
tup
roughing
(tool dnmm-71 tmxar-pdr93)
right
(geo (tup (p 0 25) (p 50 25) (p 60 40)))
roughing
(tool dnmm-71 tmxar-pdr93)
left
(geo (tup (p 40 25) (p 40 20) (p 50 20) (p 50 25)))
roughing
(tool dnmm-71 tmxar-pdl93)
left
(geo (tup (p 70 40) (p 70 30) (p 110 30) (p 110 25) (p 150 25)))
roughing
(tool dnmm-71 tmxar-pdl93)
left
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
roughing
(tool dnmm-71 tmxar-pdr93)
right
(geo (tup (p 110 25) (p 110 20) (p 120 20) (p 120 25)))
```

OUTPUT from µCAD2NC
DFKI Publications

The following DFKI publications or the list of all published papers so far can be ordered from the above address. The reports are distributed free of charge except if otherwise indicated.

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann: Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist: Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harald Boley: Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmidthofer, Otto Kühn, Gabriele Schmidt: Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost: The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

RR-90-16
Franz Baader, Werner Nutt: Adding Homomorphisms to Commutative/Monomial Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-90-17
Stephan Busemann: Generalisierte Phasenstrukturgrammatiken und ihre Verwendung zur maschinelten Sprachverarbeitung
114 Seiten
RR-91-01
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt: The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
Bernhard Hollunder, Franz Baader: Qualifying Number Restrictions in Concept Languages
34 pages

RR-91-04
Harald Trost: X2MORF: A Morphological Component Based on Augmented Two-Level Morphology
19 pages

RR-91-05
17 pages

RR-91-06
Elisabeth André, Thomas Rist: Synthesizing Illustrated Documents A Plan-Based Approach
11 pages

RR-91-07
Günter Neumann, Wolfgang Finkler: A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures
13 pages

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist: WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schlippe: RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke: A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel: Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-12
J. Mark Gawron, John Nerbonne, Stanley Peters: The Absorption Principle and E-Type Anaphora
33 pages

RR-91-13
Gert Smolka: Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-14
Peter Breuer, Jürgen Müller: A Two Level Representation for Spatial Relations, Part I
27 pages

RR-91-15
Bernhard Nebel, Gert Smolka: Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Bussemann: Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

RR-91-17
Andreas Dengel, Nelson M. Mattos: The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-18
John Nerbonne, Klaus Netter, Abdel Kader Diagne, Ludwig Dickmann, Judith Klein: A Diagnostic Tool for German Syntax
20 pages

RR-91-19
Munindar P. Singh: On the Commitments and Precommitments of Limited Agents
15 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-21
Klaus Netter: Clause Union and Verb Raising Phenomena in German
38 pages
TM-91-10
Béla Buschauer, Peter Poller, Anne Schauder, Karin Harbusch: Tree Adjoining Grammars mit
Unifikation
149 pages

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des
D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

DFKI Documents

D-91-01
Werner Stein, Michael Sintek: Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-02
Jörg P. Müller: Design and Implementation of a Finite Domain Constraint Logic Programming
System based on PROLOG with Coroutining
127 pages

D-91-03
Harold Boley, Klaus Elbernd, Hans-Günther Hein, 
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht
1990
93 Seiten

D-91-06
Gerd Kamp: Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems
für Drehleit
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause: Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen
Sprache RELFUN
137 pages

D-91-09
David Powers and Lary Reeker (Eds): Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds): MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds”
246 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-11
Thilo C. Horstmann: Distributed Truth Maintenance
61 pages

D-91-12
Bernd Bachmann:
H²rAcOn - a Knowledge Representation System with Typed Hierarchies and Constraints
75 pages

D-91-13
International Workshop on Terminological Logics Organizers: Bernhard Nebel, Christoph Pellason, 
Kai von Luck
131 pages

D-91-14
Erich Achilles, Bernhard Hollunder, Armin Laux, 
Jörg-Peter Mohren: KRIS: Knowledge Representation and Inference System
- Benutzerhandbuch -
28 Seiten

D-91-15
Harold Boley, Philipp Hanschke, Martin Harm, 
Knut Hinkelmann, Thomas Labisch, Manfred Meyer, Jörg Müller, Thomas Otzen, Michael Sintek, Werner Stein, Frank Steinele:
μCAD2NC: A Declarative Lathe-Worplaning Model Transforming CAD-like Geometries into
Abstract NC Programs
100 pages
μCAD2NC: A Declarative Lathe-Workplanning Model Transforming CAD-like Geometries into Abstract NC Programs

Harold Boley, Philipp Hanschke, Martin Harm, Knut Hinkelmann, Thomas Labisch, Manfred Meyer, Jörg Müller, Thomas Oltzen, Michael Sintek, Werner Stein, Frank Steinle