29. Architecture of a knowledge-based-system for the detailing of reinforced concrete columns

Christoph Meinecke, Raimar J. Scherer

Institut für Massivbau und Baustofftechnologie
Universität Karlsruhe
Am Fasangarten
D-W-7500 Karlsruhe 1, Germany

This paper presents the hypothesis part of an expert-system for detailing reinforced concrete structures. The structural members on which the work is focused are columns. To generate a hypothesis - that means to configurate the reinforcement for a given structural member with an almost fixed geometry - needs different kinds of information, i.e. knowledge and a strategy to apply this knowledge. Therefore a hybrid system is chosen which combines object oriented organization to represent the fixed knowledge and a rule base to model the strategy and the dynamic knowledge.

Introduction

It is the aim to use CAD-systems for the construction of reinforced concrete structures and not only for the purpose of the graphical drawing. To reach this aim a CAD-system is connected with an expert-system creating an intelligent construction system. An exclusively rulebased attempt seems to be not sufficient since the description of a reinforced concrete structure is too extensive. A rulebased system combined with an object oriented data base is a tool to solve such a distinct problem.

Unit construction system

A reinforced concrete unit is not only substructured in building components (e.g. continuous beams or frames) but on a second level these building components itselves are substructured in fundamental components (e.g. columns, frame corners or corbels) (Scherer 1990 a). This classification is made

- to generate an easier solution for the reinforcement layout (hypothesis)
- to simplify users control of this 'machine-made' solution
- to reduce the time for the verification made by the expert-system
- to increase the grade of repetition of the fundamental components for reducing
- the quantity of components.

The basic idea of these fundamental components are the D- (disturbance) and B- (bending) areas corresponding to the framework analogy (Schlaich 1987). Each fundamental component has typical reinforcement layouts. It is possible to describe in a parametric form its geometry and function by a rule base.

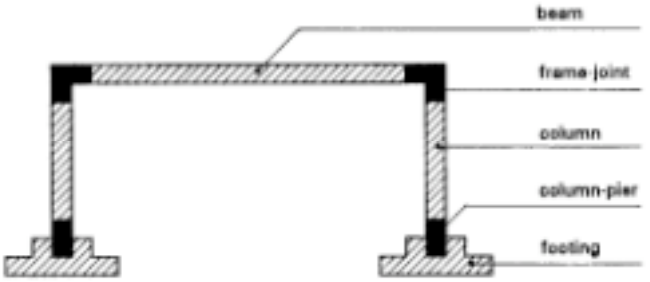

Figure 1. The building component 'frame' with its fundamental components.

The geometric description in parameters enables to model the whole range of variation for a fundamental component in one element. The primitive of the geometric description in parameters is the prisma. This description is comparable to the representation of the building components used in (Zlajpah 1990).

The functional description in parameters enables to model in one set of rules all possible reinforcement layouts as a function of the requirements of the building component. The rules are substructured in accordance to the function of the reinforcement (compression, tension, shear, construction).

The advantage of this procedure is a finite number of fundamental components. Out of these components all possible building components and units can be assembled (see figure 2). The amount of these units however is infinite.

To use this unit construction system the solution finding mechanism is subdivided in three steps. In the first step the unit is disassembled in building components and in fundamental components. In a second step the reinforcement of the fundamental component is generated and in the third step these fundamental components are reassembled to the building component and at last to the unit.

A full construction expert-system should include two different parts (Scherer 1990 b), (Garret 1989): a hypothesis generating and a verification system. A hypothesis generating system constructs a solution - a verification system proves a solution. In this paper a hypothesis generating system for the detailing process of reinforced columns is presented. It is described how mechanisms which are specific for expert-systems are used to solve the detailing problem.

Requirements for the AI tool

The AI tool (artificial intelligence) which is going to be used for such an attempt should provide the following features (Meinecke 1990):

- based on a symbol oriented programing language
- object oriented database
- rulebased knowledge representation
- allocation of expert-system typical mechanisms like rule matching algorithm
- possible connection of foreign programs with the shell to use existing numerical programs.

The tool that seems to be satisfactory to these requirements is Knowledge-Craft (KC) (Carnegie 1989). It provides the following components: an object oriented data base CRL (Carnegie Representation Language), a forward-chaining rule based knowledge representation CRL-OPS (an implementation of OPS-5) (Krickhahn 1987), a backward-chaining rule based knowledge representation CRL-PROLOG. KC is embedded in the LISP (Steele 1984).

KC could be characterized as an open expert-system-shell. That means the user has all opportunities which are typical for expert-systems. He also has the possibility to introduce his own ideas into the system. This aspect was very important while choosing the shell. For example in KC there are following opportunities:

For the conflict resolution strategy there are two OPS-specific alternatives (LEX and MEA), but there is still the possibility to integrate a self defined solution strategy in the system. The CRL data base offers the opportunity to define relations between the classes in addition to the CRL-given relations. By this a special inheritance mechanism is created.

It is important to distinguish the knowledge which is required for the design problem in fixed knowledge, dynamic knowledge and in strategy knowledge. The fixed knowledge is represented in the CRL data base. Dynamic and strategy knowledge is modelled in CRL-OPS rules.

Object oriented data base

The object oriented data base - furtheron called the taxonomy - is subclassified in three subtaxonomies:

- geometry
- material
- load.

Each of them stores facts to describe reinforced concrete columns. This classification is carried out for the sake of distinctness. Information about the geometry of the building component is stored in the geometry taxonomy. Information due to the material is accumulated in the material taxonomy. The load taxonomy serves for the storage of facts to describe the load of columns.

The three fundamental ideas of object oriented knowledge representation are (Puppe 1988): inheritance hierarchy, attributive procedures and default values. The realization of these ideas is shown in the following chapters:

Inheritance Hierarchy

An object oriented database - with implicit inheritance hierarchies - is due to an economic data structure (Puppe 1988). Only individual properties of a class are stored at the class itself (Caspar 1990). Properties and occasionally their values with common character are stored in superordinated superclasses. Properties and their values are left from superordinated to subordinated classes. The same properties and values of different classes are stored in the data base only once at a superordinated class.

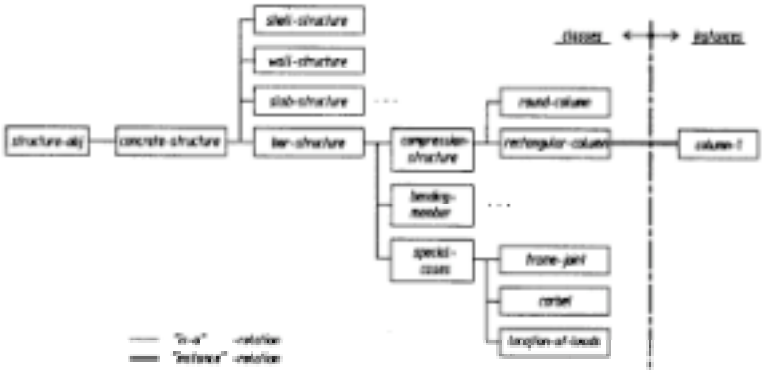

Figure 2. Extract of the geometry taxonomy.

```
(defschema concrete-structure
        (is-a structure-obj)
        (material-concrete )
        (material-steel )
        (environment-class-1 closed-rooms permanent-dry)
        ...
        (environment-class-4 susceptible-to-corrosion strong-chemical-attack)
        (environment-class )
        (ch-ds )                                    ; chosen-diameter long.rein.
        (c )                                        ; concrete-cover
        ...)
```

Figure 3. The class "concrete-structure".

Doing this there is no redundancy in a well shaped taxonomy. The use of this mechanism is shown in the figures 2 and 3. Globale properties like "material-concrete" and "material-steel" or "environment-class-_" are stored at the class "concrete-structure". The information is left to all subclasses of "concrete-structure" that are all concrete structures but they are stored only once in the taxonomy.

Attributive Procedures

This mechanism - or in other words: demons - serves also for an economic data structure. A demon is a program which is allocated to a certain property of a class. It determines the value of its property based on existing values of other properties. The demon is connected to the data base by special CRL commands.

The class "steel-bar" (see figure 4) contains the knowledge how the area of a single bar is calculated. It leaves this ability to its subclasses and their instances, e.g. to "diameter-20". The class "concrete-structure" (see figures 2 and 3) contains the information how the concrete cover for the reinforcement is provided. In German standards this concrete cover is dependent on the diameter of the chosen longitudinal reinforcement bars and the existing environment condition (see figure 5).

Figure 4. The material taxonomy.

The table shown in figure 5 is programed as a demon which is guarding the property "c" (concrete-cover) of all concrete structures. This property and its demon are arranged at the class "concrete-structure". The demon is getting active each time the CRL-command "get-value...." is attaching the value of the property "c". If the diameter and/or the environment-conditions are not yet known the demon gives an answer with secure assumptions for the unknown details. Naturally this ability is left to all subclasses and their instances, that means to all concrete structures.

	1	2	3	4
	environmental conditions	diameter of the reinforcement d_s [mm]	minimum value for ≥ B25 min c [cm]	characteristic value for ≥ B25 nom c [cm]
1	structures in closed rooms e.g. in flats (incl. kit-chen, bath room and laundry) offices, schools, hospitals commercial stores. - unless otherwise specified structures that are permanent dry	to 12 14, 16 20 25 28	1.0 1.5 2.0 2.5 3.0	2.0 2.5 3.0 3.5 4.0
..	....	....	....	....
4	structures susceptible to corrosion. E.g. frequent exposure of aggressive gases, deising salts or strong chemical attack	to 28	4.0	5.0

Figure 5. Extract of table 10 of the DIN 1045 (DIN1045 1988).

Default values

Often it seems to be useful to have default values of certain properties in specific classes. During a session the user can accept these default values but naturally he can also change them. The default value for the arrangement of the reinforcement in rectangle columns with biaxial-load is the concentration at the corners (see figure 5). This design always has the larger lever arm to the neutral axis.

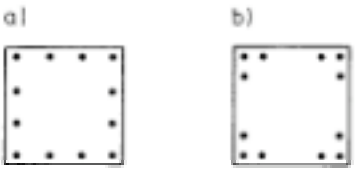

Figure 6. Longitudinal reinforcement distributed along the sides (a) and concentrated at the corners of a column (b) (DAfStb 1979).

Relations

For the design problem there is the necessity to leave knowledge under classes which do not have an original inheritance relationship. The classes representing the different forms of concrete (B-5 to B-55), the kinds of reinforcement steel or the reinforcement itself do not have such an original relation to all subclasses of "concrete-structure". A relation mechanism provides the chance to leave knowledge outside of the classes. In the class "concrete-structure" (see figure 3) the properties "material-concrete" and "material-steel" and their inverse relation "is-used-in" utilized in the classes "b-5" to "b-55" and "bst-500-s" are not only properties but also self defined relations.

Having chosen the materials the instance "column-1" e.g. has the value "b-35" for the property "material-concrete" and the value "bst-500-s" for the property "material-steel". All information stored in the classes "b-35" and "bst-500-s" is available for the instance "column-1". Relations are - like demons - connected to the data base by CRL mechanisms. The relations are properties in the classes. The values of these properties are the classes to which the relation is connected to.

```
{{ COLUMN-1
        INSTANCE: RECTANGULAR-COLUMN
        MATERIAL-CONCRETE: B-35
        MATERIAL-STEEL: BST-500-S
...}}
```

```
{{ B-35
        IS-A: B-II
        IS-USED-IN: COLUMN-1
        ...
}}
```

Figure 7. Extract of the instance "column-1" and the class "b-35".

Example for encoding the German standard DIN 1045

In standards knowledge is often represented in tables. In the following chapter it will be shown how this is reproduced. Generally only such knowledge is stored in the taxonomy which is due to one class (fixed knowledge). Knowledge concerning more than one class is represented as a rule. Figure 8 shows table 1 of the German standard DIN 1045 (DIN1045-1988). It is evident that the CRL-code of the class "b-35" shown in figure 9 is very close to the language of the standard.

	1	2	3	4	5	6
	concrete group	strength class of the concrete	characteristic strength β_{WN} (minimum value for the compression strength β_{W28}) [N/mm^2]	mean strength β_{WS} (minimum value for the average compression strength β_{Wm}) [N/mm^2]	mix	use
1	concrete B I	B 5	5	8	paragraph 6.5.5	unreinforced concrete
2		B 10	10	15		
3		B 15	15	20		un-
4		B 25	25	30		reinforced
5	concrete B II	B 35	35	40	paragraph 6.5.6	and reinforced concrete
6		B 45	45	50		
7		B 55	55	60		

Figure 8. Table 1 of the DIN 1045.

```
(defschema b-35
        (is-a b-II)
        (is-used-in )
        (characteristic-strength-n/mm**2 35)
        (mean-strength-n/mm**2 40)
        (mix din-1045-6.5.6)
        (use unreinforced-concrete reinforced-concrete ))
```

Figure 9. The class "b-35".

Rule base

There are two kinds of rules in the rule base. One kind of rule describes the construction knowledge that is necessary for the solution of the design problem. The other kind of rule represents strategy knowledge. They are used to describe the way how the solution of the design problem can be achieved.

Strategy knowledge

The solution finding mechanism (recognize-act cycle) in a rulebased system is only controlled by the data status of the active classes and their instances. It uses the inference engine (Winston 1984)

- to select the rules, which match the instances (matching phase)
- to choose one of those rules (conflict resolution phase)
- to execute the actions specified in the selected rule (execution phase).

So it is not required to subdivide the rule base. But it became obvious that a lot of the data requests in the rules condition part do only have control character. A large part of the aspired clarity would be lost. To model the strategy knowledge the first phase of the recognize-act cycle is interfered by introducing a "context-object" into the data base (see figure 10) (Krickhahn 1987).

```
(defschema context-obj
        (is-a obj )
        (stack geometry forces long-rein tie-rein)
        (context ))
```

Figure 10. The "context-object".

The property "stack" contains all necessary contexts to become active during the session. Each of them must become active to generate a satisfactory result. The rule "control-ini" puts the first value of the property "stack" into the property "context", and deletes this special value in the "stack" property. The essential condition for this action is: the property "context" must not have any value.

```
(p control-ini
(context-obj
        ^schema-name <name>
        ^context ())
-(context-obj
        ^schema-name <name>
        ^stack ())
-->
(new-value <name> 'context (get-value <name> 'stack))
(delete-value <name> 'stack (get-value <name> 'context)))
```

Figure 11. The rule "control-ini".

All the other rules - those rules which represent the construction knowledge - have a context-object request in their condition part to prove the rule against the active context. To leave an actual context, there must be a precise defined instance data status. If this data status is reached the value of the context-object's property "context" must be deleted to bring up the next context in the "stack". In each context there is a rule to manage this changeover. For the context "long-rein" this rule is shown in figure 12.

```
(p end-context-long-rein
(context-obj
        ^context 'long-rein)
-(rectangular-column
        ^instance 'rectangular-column
        ^long-bars ())
```

```
–>
(delete-any-values 'context-obj 'context) )
```

Figure 12. The rule "end-context-long-rein".

Construction knowledge

The basic knowledge to describe the construction of reinforced columns is formulated in standards, recommandations and textbooks (DIN1045 1988, DAfStb 1979). How this kind of knowledge is transferred into a shape the computer is able to understand is the purpose of this chapter.

In standards knowledge is often represented as a table. Above it is shown how such a table is represented in the taxonomy. If a table is concerning more than one class it should be represented in rules. Such a table is not containing fixed but a kind of dynamic knowledge. The data status of one class is not only altering its own properties and their values but also properties and values of different classes (dynamic or interactive knowledge). Knowledge - not only stored in tables - structured as just described is represented in the rule base.

Table 32 (DIN1045 1988) presents the permissible diameters for the longitudinal reinforcement bars in columns (see figure 13). How the second line of this table is performed is shown in figure 14.

	1	2
	smallest dimension of the cross section of the compressive member [cm]	smallest diameter d_{sl} [mm]
1	< 10	8
2	≥ 10 to < 20	10
3	≥ 20	12

Figure 13. Table 32 of the DIN 1045.

The condition part or the LHS (Left-Hand-Side) of the rule (shown in figure 14) checks first whether the active context is "long-rein" (longitudinal reinforcement). If the rule should fire the instance of the class "rectangluar column" must have the minimum of width and depth ("min-d") between 10 and 20 cm. The class' property "has-diameter" may not have any values. Beyond of this on the variable <name> the name of the instance is stored locally (e.g. "column-1").

```
(p 1045.tab.32-2
(context-obj
        ^context 'long-rein)
(rectangular-column
        ^instance 'rectangular-column
        ^schema-name <name>
        ^min-d { >= 10 < 20 }
        ^has-diameter ())
```

```
->
(new-values <name> 'has-diameter (get-values-if 'steel-bar 'diameter #'(lambda (x) (>= x 10))))
)
```

Figure 14. The rule "1045.tab.32-2".

In the conclusion part or the RHS (Right-Hand-Side) of the rule the value of the property "has-diameter" is determined. In the class "steel-bar" all available diameters are stored in its property "diameter". For the value of the column's property "has-diameter" only those diameters which are greater or equal than 10 mm will be taken. Afterwards the diameter for the longitudinal reinforcement will be chosen out of the values of this property.

To secure the stability of the reinforcement cage and to prevent wide cracks the maximal distance of the reinforcement bars is set to 30 cm (DIN1045-1988). This is only effective for columns with larger dimensions than 40 cm. If the longitudinal reinforcement - which is necessary to ensure the structural analysis - is laid out in larger distances than 30 cm, constructive reinforcement must be build in to regard the maximal distances between the reinforcement bars. Each bar has got the information about the distance to its next bar to the right ("delta-r") and about the name of this right neighbor ("bar-r").

```
(p constructive-reinforcement-b
(context-obj
        ^context 'long-rein)
(bar
        ^instance 'bar
        ^schema-name <name>
        ^is-used-in <structure>
        ^delta-r > 30)
(rectangular-column
        ^instance 'rectangular-column
        ^schema-name <structure>
        ^b > 40 )
-->
(const-rein <name>) )
```

Figure 15. The rule "constructive-reinforcement-b".

The rule "constructive-reinforcement-b" fires each time a bar has a larger distance to its right neighbor than 30 cm. Precondition is that the column's dimension "b" is greater than 40 cm. The layout of the new constructive bar or eventually bars (that depends on the distance "delta-r") is managed by the LISP-function "const-rein". This function creates the new bar or bars, calculates and determines the appropriate coordinates and values of the other necessary properties.

At last the function enters the new instance (s) into the data base. Of course the values of the properties "delta-r" and "bar-r" of the original longitudinal reinforcement bar are updated too. - Often it is very helpful to use such functions in the RHS to achieve a better performance.

Representation of the results

There are at least two different ways to describe the results of the expert-system session: All information about the column, the longitudinal reinforcement bars and the ties is stored in one instance. The other possibility is all information is stored in different instances: one for the concrete body of the column and one for each reinforcement bar and tie.

Representation as a single instance

This first proposal would have the advantage that all information due to the column is stored at one place and the system has to handle only one result-instance. The shortcoming of this suggestion is that all information due to a single bar or a single tie should be stored as one property of the result-instance. But such a property with a list value is not easy to handle. CRL and CRL-OPS do not have the possibility to work with a distinct element of a list value. CRL only has the possibility to work with the first element of a list value (get-value...) or to work with all of them (get-values...). So for each kind of data request it is necessary to produce a selfdefined request function. This seems to be not very useful.

A way out of this could be to create a certain property for each important feature of a reinforcement bar or a tie at the result-instance. Using this solution all CRL mechanisms can be used. But the clarity of the result-instance is going to be lost because of the huge amount of properties.

Representation in several instances

We chose the solution to create an instance of an adequate class not only for the concrete body of the column but also for each bar and tie. This solution has the decisive advantage that all CRL and OPS commands can be used on this data structure. Each bar or tie (represented as an instance) can be matched by CRL-OPS rules and can be manipulated by CRL-commands.

```
{{ COLUMN-1
        INSTANCE: RECTANGULAR-COLUMN
        MATERIAL-CONCRETE: B-35                 ; material-concrete
        MATERIAL-STEEL: BST-500-S       ; material-steel
        LAYOUT: CORNER                          ; layout corner-concentrated
        H: 420                                  ; hight [cm]
        D: 40                                   ; width [cm]
        B: 40                                   ; depth [cm]
        MIN-D: 40                               ; minimum of depth and width [cm]
        HAS-DIAMETER: 12 14 16 20 25 28         ; has diameters (long. rein.) [mm]
        CH-DS: 20                               ; chosen diameter (long.rein.) [mm]
        CH-DTIE: 8                              ; chosen diameter (tie) [mm]
        AS: 35.8                                ; necess. area long. rein. [cm**2]
        CH-AS: 37.7                             ; chosen area long. rein. [cm**2]
        ENVIROMENT-CLAS: 2              ; actual environment
        C: 3                                    ; concrete cover [m]
        ....
        TIE: 22                                 ; quantity of ties
        LONG-BARS: 12}}                         ; quantity of long. bars
```

Figure 16. The instance "column-1".

```
{{ TIE-12
        INSTANCE: RECTANGULAR-TIE       ; rectangular tie
        IS-USED-IN: COLUMN1             ; belongs to "stuetze-1"
        DTIE-U: 20                              ; distance to the next tie up [cm]
        NTIE-U: 13                              ; name of the next tie up
        STIE: 8                                 ; diameter of the tie [mm]
        HOOK: 10                                ; end hook [cm]
        DBR: 5                                  ; bend radius[cm]
        X: 180                                  ; coordinates of the tie (x,y,z) [cm]
        Y1: 16.6
        Z1: 16.6
```

```
....
Y4: -16.6
Z4: 16.6
N: 12}}                                    ; tie's number
```

Figure 17. The instance "tie-12".

Interface between the expert-system and a CAD-system

The representation of the solution found by the expert-system takes place on a CAD-system. To increase the acceptance of this solution the user must always have the full control of the construction process. To realize this the system must work interactively. In other words the progress of a partial solution of the whole construction problem must be open for the ideas of the user. Not giving preference to a single CAD-system it is necessary to work out an interface which is able to connect the system with different CAD-systems. So the interface must have two parts: one part which is expert-system specific and the other one that is CAD-system specific.

The first part of the interface converts the information stored in the instances, which is described above into a datafile. This is realized as a LISP program. It is started as soon as a partial solution that should be proved by the user is worked out. The second part of the interface must transfer the datafile made by the first part in a form that the CAD-system will understand. For example at the UNICAD system (Hochtief 1989) these are extensions at the CAD-system's interface encoded in FORTRAN.

It is the aim to implement the interface on a standardized exchange format. The most promising one seems to be STEP, because it has an object oriented concept. Currently the 2D draw exchange format is validated and will be a standard in Germany in the near future (DIN-NAM 1989). To use such a standardized drawing exchange format will have the great advantage that an interface made for one CAD-system can be made easier suitable to other CAD-systems.

The interactive interface

In the system there are two kinds of interactive processes. The first one is used if the system needs an information which is obtainable neither in the taxonomy nor in the rule base. The system will ask the user to supply it with that information. The other kind of interactivity is used to increase the acceptance of the system's solution: The entire process of constructing the reinforcement must be subdivided in partial solutions which the user can prove more easily. This process must be managed by the interface, its function is a hypothesis / verification mechanism, what is described in the next paragraphs.

The system constructs a partial solution which is in conform with the standards (hypothesis) and offers it on the CAD platform to the user. The user can accept this solution or change it. If he changes the solution, the system will prove this user made proposal whether it is in conform with the standards (verification).

If the system proves the user made solution to be not acceptable the hypothesis/verification mechanism is starting again. The system will work out a new solution which is in conform with the standards. To generate this solution the system will change the user's unconform proposal until it is sufficient to the standards. This new proposal is offered again and the user can accept it or he naturally can change it once again.

If the user's proposal is in conform with the standards or the user does not want to change it the system will adapt it and will go on with the next step in generating a solution for the entire problem.

Future developments

In the future our work will have the following focal points: It will be very interesting to integrate the verification part into the system (Scherer 1990 b). That will achieve the full possibility of interactive work. The realization of the interface between CAD and expert-system based on a standardized data format will be a second emphasis. A third point will be to integrate knowledge into the taxonomy and the rule base that is representing different schools of construction of concrete structures (Franz 1988) (Leonhard 1977). Than the user will have the opportunity to decide which school will be relevant for his construction.

Conclusions

The hybrid realisation of a hypothesis generating system for the detailing of reinforced concrete columns by an object oriented and a rule based representation of the design knowledge and the design strategy is an encouraging attempt for the implementation of one part of a general expert-system which is under development.

The hypothesis generating system is outlined as a production system not considering nonmonotonic reasoning and contradictive solutions. It generates - based on strategy selected by the user - a unique solution. The structural member described in this paper covers only a specific design region which is not very complex.

This allows to be concentrated on the object oriented presentation of the fixed knowledge and the rule based representation of the dynamic knowledge. In other design regions it will be impossible to generate one unique and satisfactory solution. For example the connection process in which two seperately designed fundamental components (e.g. column and frame-joint) are assembled to a building component (frame) can force the system to take back its solutions made for the reinforcement layout of one of the fundamental components (backtracking). Some ideas to manage this problems are given in (Mehrafza 1991). A redesign of one single fundamental component will be necessary. Nonmonotonic reasoning technics have to be applied to make the process efficient.

Nevertheless our simple structural member proved the applicability of the combined object oriented and rule based implementation of the problem. Especially the object oriented representation of the fixed knowledge and the fixed strategy in a taxonomic form proved their advantages to a pure rule based representation. This result encourages to use the hybrid object oriented and rule based implementation as an outline for the whole expert-system.

References

Carnegie Group Inc. 1989. "Manual of Knowledge Craft 3.2." Pittsburgh, PA

Caspar, E. 1990. "Ingenieurgerechte Wissensrepräsentation am Beispiel eines objektorientierten Expertensystems für Stahlbau-Nachweise." (knowledge representation for engineers at the example of an object oriented expert-system for the prove of steel constructions) In: J.Gauchel (ed.), *KI-Forschung im Baubereich.*, Berlin: Verlag Ernst & Sohn

Deutscher Ausschuß für Stahlbeton (DAfStb). 1979. "Heft 220." (Heft 220: guideline for the construction of reinforced concrete members) Berlin: Verlag Ernst & Sohn

Deutsches Institut für Normung e.V. 1988. "DIN 1045." (DIN 1045: German standard for reinforced concrete structures) Berlin: Beuth

Franz, G., and K. Schäfer. 1988. "Konstruktionslehre des Stahlbetons." (construction of the reinforced concrete) Berlin: Springer.

Garret, J.H., and S.J. Fenves. 1989. "Knowledge-based standard-independent member design." *Journal of Structural Engineering* 115, no. 6: 1396-1411

DIN-NAM 96.4.2. 1989 "STEP-2DBS." (STEP-2D Building Subset, version 1.0, standardized exchange format - going to be standard in Germany) RIB e.V. Stuttgart

Hochtief AG. 1989. "UNICAD-B Programmierhandbuch." (UNICAD-B programers handbook) Franfurt/Main, Essen

Krickhahn, R., and B. Radig. 1987. "Die Wissensrepräsentationssprache OPS 5." (the knowledge representation language OPS-5) Braunschweig: Vieweg

Leonhardt, F. 1977. "Vorlesungen über Massivbau." (lectures on reinforced concrete) Berlin: Springer

Mehrafza, M.J., and R.J. Scherer. 1991. "Regelbasierte Formulierung von Variantenelementen in der Stahlbetondetaillierung." (rule based formulation of parameterized elements in detailing reinforced concrete buildings) In: *Tagungsberichte der Dortmunder Expertensystemtage*. Dortmund

Meinecke, Ch., and R.J. Scherer. 1990. "Expertensystem-Techniken im Vergleich zum normalen Programm am Beispiel der Stahlbetondetaillierung." (expert-system technics compared to the normal program at the example of detailing reinforced concrete structures) In: *DAfStb-Forschungskolloquium*. Karlsruhe

Puppe, F. 1988. "Einführung in Expertensysteme." (introduction to expert-systems) Heidelberg: Springer

Scherer, R.J. 1990-a. "Formen der Wissensrepräsentation in einem Expertensystem für die Stahlbetondetaillierung." (different kinds of knowledge representation in an expert-system for detailing reinforced concrete structures) In: J.Gauchel (ed.), *KI-Forschung im Baubereich.*, Berlin: Verlag Ernst & Sohn

Scherer, R.J. 1990-b. "Expertensystem für die Planung und Bemessung der Bewehrung von Stahlbetonbauteilen." (expert-systems for the design of reinforced concrete structures) In: R. Anderl (ed.) *CAD/CAM-Technologie*. Berlin: Springer

Schlaich, J., K. Schäfer, and M. Jennewein. 1987. "Toward a Consistent Design of Structural Concrete" In *PCI Journal,* May-June 1987

Steele, G.L. 1984 "COMMON-LISP the language." Digital Press

Winston, P.H. 1984. "Artifical Intelligence." Addison-Wesley

Zlajpah ,D., and J. Duhovnik. 1990. "AR_CAD: Regelbasiertes System für die Bewehrungsführung im Stahlbetonbau." (AR_CAD: a rule based system to model the reinforcement for concrete structures) In: J.Gauchel (ed.), *KI-Forschung im Baubereich.*, Berlin: Verlag Ernst & Sohn