SUPPORTING THE CONFIGURATION OF TECHNICAL SYSTEMS

Hans Kleine Büning
FB 17 – Praktische Informatik
Universität-GH Paderborn
Warburger Str. 100, 33095 Paderborn
Germany

Benno Stein
FB 17 – Praktische Informatik
Universität-GH Paderborn
Warburger Str. 100, 33095 Paderborn
Germany

ABSTRACT

Configuration is a process of creating complex systems out of a finite set of predefined objects. This process may enclose the selection, the assemblage and the adaptation of these objects. In this paper we want to describe how the configuration of hydraulic plants can be supported.

A hydraulic plant is a complex system that consists of electronic, mechanical, and hydraulic components. Such a system has to be configured according to particular customers' demands. Before installing and setting a hydraulic plant into operation, it must be checked whether it contains faults and whether it fulfills the required demands.

Since such a configuration process is rather creative, it cannot be automated completely. But, it is possible to support the process of checking, detecting faults and designing a hydraulic circuit in an *appropriate* way. Our idea of "appropriate" orients itself by the usual working documents of experts within the hydraulic domain. Since the central working document is the drawing of a hydraulic plant, one key idea is the *graphical description of hydraulic problems* using knowledge based techniques.

For this, we have been developing the prototypic system $^{\mathrm{art}}$*deco* that enables a user to examine hydraulic systems as follows: While a hydraulic circuit is drawn, basic checks are performed and associated knowledge bases are built automatically in order to reproduce the physical model of the system. Upon the base of this model arbitrary parts of the circuit can be investigated easily.[1]

SUPPORTING THE CONFIGURATION OF HYDRAULIC CIRCUITS

A hydraulic circuit consists of mechanical, hydraulic, and—increasingly—electronic components. Configuring a hydraulic system is a complex process. Within this process that starts with first design concepts and ends with setting the readily installed system into operation, there exist a lot of tasks that can be supported by a computer. Figure 1 depicts a circuit and some constraints which may result from some customer's demands:

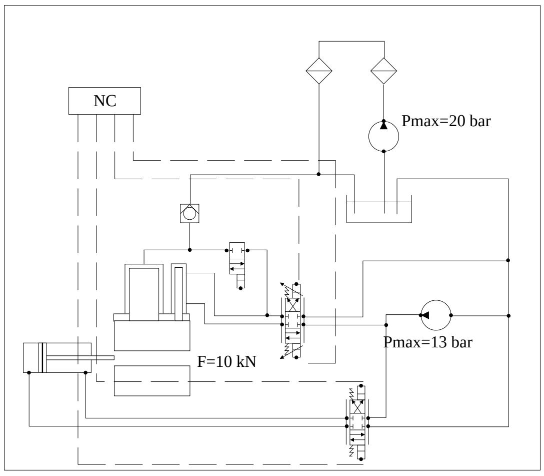


Figure 1: Hydraulic circuit.

[1]This work is part of a cooperative project together with the institute "Meß-, Steuer- & Regelungstechnik", Universität Duisburg, Prof. H. Schwarz. It is supported by the DFG (Project-No. Kl 529/3-1 "WITA").

Typical problems within this connection are:

- Often, the checking of a circuit's functions and its configuration happens during the installation. Any fault that is only detected at this stage, will be both difficult to correct and expensive.
- Until now, there exists no diagnosis support when setting a hydraulic circuit into operation.
- Since the design of a hydraulic circuit is supported only marginally, components might be dimensioned incorrectly.

In order to guarantee a circuit design in unison with customers' demands, the checking of a circuit's function has to be scheduled earlier. For this, the design of a hydraulic system should be supported and its checking should be automated widely.

Basic Concepts

A central idea of ^{art}*deco* is the following: The support of tasks within the configuration process should be oriented by the typical working documents of the domain experts. That means here, it must be possible to specify a checking problem graphically. I.e., when manipulating a graphic object, the associated physical and topological knowledge has to be processed also.

To achieve such a deep connection between the graphical representation and the technical specification, a graphical as well as a technical description language is provided by ^{art}*deco*. These two languages can be used to define complex objects which are characterized by a picture structure and a technical behavior description. Furthermore, it is possible (and necessary) to specify dependencies between the syntactical information, i.e. the picture structures and the semantics, i.e. the technical description of an object.

Since all technical information of an object is described locally, it is easy for a user to define new objects. This leads us to the second major concept of our approach: Deriving the global behavior of a technical system from its local component descriptions. This idea may sound clearly and easily, but entails inherent problems. The following example may illustrate this: Consider an electric resistor whose behavior can be described using Ohm's Law: $i = \frac{\phi_1 - \phi_2}{R}$. This description can be evaluated to calculate any unknown parameter. By contrast, if two resistors are connected in parallel the local description remains correct but cannot be evaluated *locally*. Moreover, such a parallel structure has to be detected and reduced. Here, in the case of hydraulic resistors, the calculation of non-linear equations is not trivial.

Another point to to be aimed at is a knowledge acquisition concept that is both adequate to the problem and user-friendly. Therefore, the interaction with a knowledge acquisition tool has to be realized at the same level as the "normal" user interaction takes place. I.e., there should not be a need for an AI/computer expert to maintain and extend the system.

Distinguishing Different Configuration Steps

It is not possible to give a general set of rules according to a hydraulic system can be configured. However, Lemmen (cf. [10]) identifies several problems that always have to be solved and whose extent depends on the particular circuit. The problems can be divided into checking problems, dimensioning problems, and diagnosis.[2]

Within the tasks described at the beginning of this section the checking problems play the most important role and will be introduced in more detail here. It is useful to distinguish between the checking of

1. connections,
2. stationary behavior, and
3. dynamic behavior.

According to information of manufactures and the personnel installing hydraulic circuits it would be possible to find about 90% of all faults during the first two checking steps.

ad 1) When checking connections, it has to be examined if all environmental demands of a component can be satisfied. Within this connection mechanical, hydraulic, and electric connections are distinguished. Typical parameters are dimensions of pipes, loading capacities, electric and mechanical constraints (cf. [10], [3]).

ad 2) The checking of the stationary behavior bases upon step one. Within this step, the interplay of components is investigated with respect to a customer's stationary demands. Among other things, following points have to be settled:

- Which maximum pressure values occur?
- Will the flow velocity exceed the range of allowed values?
- Does the switching logic comply to the customer's demands?

Since all demands are known at configuration time of a hydraulic system, two possibilities exist to check the system's stationary behavior:

[2]Until now, parts of the checking tasks are supported by ^{art}*deco*. The checking and dimensioning of components with respect to their dynamic behavior and the automation of the diagnosis are subjects of the current work. These topics will be introduced and discussed in more detail on page 5 and page 7 respectively.

- destructive approach: Test, whether a circuit's behavior model and the output parameters given (= demands of a customer) contradict each other.

- constructive approach: Try to derive the output parameters from both the behavior model of the circuit and the input parameters given.

After the stationary check has been carried out successfully the dynamic checking step has to be invoked.

MODELING TOPOLOGICAL AND STATIONARY KNOWLEDGE

As formerly mentioned, the philosophy is to represent hydraulic knowledge as locally as possible, i.e. component-oriented. In particular, such a modeling approach is distinguished by following concepts:

- causality through locality
- easy modifiability and extensibility
- transparent knowledge acquisition
- reusability of knowledge

Our approach distinguishes between elements, gates, connections and sources (or sinks resp.). These elements will be used to describe a systems topology, causal dependencies, and the flow of information within a technical system. A more detailed description can be found in [8].

The medium to be transported (e.g. electric current, fluids) is specified by sources/sinks of the appropriate type. A component of a real system will be described by an element e that defines constraints between all *accessible* sources/sinks of the medium. In contrast to this, the "classical" approaches[3] model the properties of a medium *component-inherently*. E.g., an electric resistor can be defined through an input flow and an output flow of current. Within the approach presented here, a resistor is defined having two gates that are connected to sources/sinks of electric current. The following figure illustrates the outlined concepts:

A concrete technical system is modeled according to the following philosophy:

(*i*) All physical parameters that have no direct manifestation within a technical component will be represented as sources of the associated medium.

(*ii*) For each technical component, a building block will be defined that contains a set of constraints which reproduce the behavior of this component.

[3]There exist a lot of "classical" approaches for component-oriented descriptions of technical systems (cf. [1], [9], [11]).

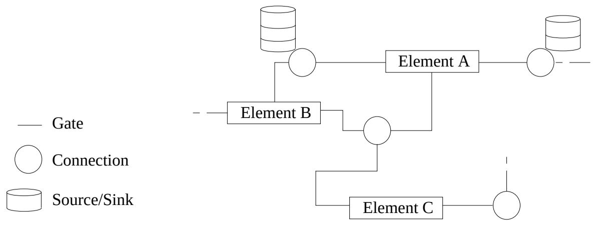


Figure 2: Basic elements of the data structure.

A hydraulic system contains electric, hydraulic and mechanical components. Therefore, we distinguish different types of gates. This predefined semantics within our data structures can be interpreted and processed while a user specifies a hydraulic system.

Until now, we left open to what extent the specification of a component's functions and behavior should be of qualitative or quantitative nature respectively. Within the domain here, both levels of description have to be provided: On the one hand, we need exact quantitative values to model the hydraulic behavior. On the other hand, e.g. the behavior of electrical components will only be modeled on a qualitative level.

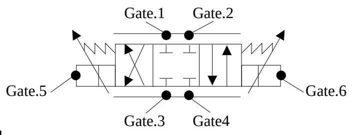
Furthermore, we need mechanisms to formulate propositions like the following one:

IF Position of valve.1 is *crossed* AND
Position of valve.2 is *blocked* THEN
Cylinder.3 extends

To describe the technical behavior of components, a formal language had been developed that provides—among other things—the following concepts:

- numerical and symbolic description of behavior
- embedding user-defined functions
- non-continuous description of behavior

The figure below depicts a valve and a part of its behavior description:



```
flow[GATE.1] =
  (CASE
   WHEN position[SELF] IS CROSSED : - flow[GATE.4]
   WHEN position[SELF] IS PARALLEL : - flow[GATE.3])
   OTHERWISE : 0)
```

pressure[GATE.4] = pressure[GATE.1] - SIGNUM(flow[GATE.1]) $\cdot$ flow[GATE.1]^2 $\cdot$ rh[SELF]

Based on such behavior specifications and the connection-gate-concept as described above, each component of a technical system can be defined as follows:

$$\begin{array}{lll}\langle\text{component}\rangle & \longrightarrow & \langle\text{name}\rangle\ \{\langle\text{gate}\rangle\}_1^*\\ & & \{\langle\text{functionality}\rangle\}_0^*\\ & & \{\langle\text{behavior description}\rangle\}_0^*\end{array}$$

A more detailed description of the underlying concepts is given in [8].

PROCESSING TOPOLOGICAL AND STATIONARY KNOWLEDGE

In order to check the components' connections, the stationary behavior, and a customer's demands of a hydraulic system the following types of constraints have to be processed:

1. topological constraints
2. connection constraints
3. functional constraints

ad 1) Topological constraints have to be derived from the drawing of a circuit and define its physical structure. The correct reproduction of this structure is achieved by (*i*) introducing global variables within the local behavior descriptions, and (*ii*) unification of variables within the connections.

ad 2) When the connection constraints are checked, the set of incident components is determined for each connection. The connection constraints of a component are defined through a set of $\langle functionality\ value\rangle$ - pairs. By means of a functionality-dependent test predicate, the correctness of the components connections can be determined.

ad 3) Within the class of functional constraints we identify three different subtypes:

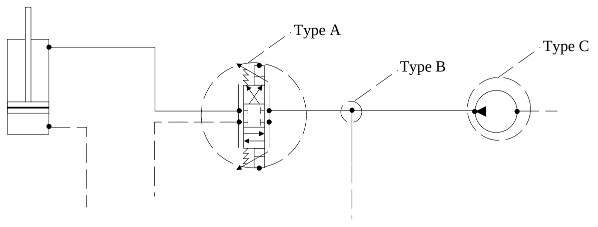


Figure 3: Different functional constraints.

A. These constraints reproduce a component's behavior and make up the major part of all functional constraints. The behavior description of the valve, presented within the former section is of this type. It should be mentioned that not all functional constraints are already known when constraint processing starts.

B. Constraints of this type are called *internal*. They also reproduce physical behavior but can neither be accessed nor changed by a user. Typical examples of such constraints are the balance of forces within a mechanical connection or the continuity condition within a hydraulic connection:

```
flow[gate.1] + flow[gate.2] + flow[gate.3] = 0.
```

C. These constraints enable a user to specify certain demands like: *"The output flow of pump.2 must be less than 120 l/min"*

Functional constraints are processed using the method of constraint propagation. We realized a constraint propagation approach to keep the processing of behavior flexible and causal: Dependencies can be traced back and explanation that base upon local derivations can be generated. If local propagation comes to an end, other mechanisms are involved in order to derive further information:

- Reducing structural complexity. This module investigates a circuit and introduces for series-parallel structures additional constraints. This results in a new network that can be procesed using local propagation. The next subsection takes a closer view to this module.
- Numerical methods. Linear equations can be extracted from constraints and processed using algebraic transformation. The efficient solution of non-linear equation systems is part of the current work.
- User inquiry. The user can be requested to specify further information. This module is part of the current work and will employ knowledge-based techniques in order to implement heuristics for parameter selection.

Reducing Network Structures

A hydraulic system can be seen as a network composed of non-linear resistors where the task is to calculate all pressure and flow values. This can be done using a global approach: All associated equations are set up and solved as a whole.

As discussed earlier, the local description and *processing* of knowledge exhibits advantages with respect to explanation and modification. But how can local propagation take place on a network of resistors?

The idea is to identify parts of the network that have a particular structure[4] —e.g., some parallel resistors. Since we know the resistance of each resistor, we can (*i*) introduce a virtual substitute resistor, and (*ii*) determine the

[4]The subnetwork must have exactly one source and one sink.

complete flow distribution. This additional information will supplement the original continuity condition.

After such a “recompilation”-process, the propagation mechanism can be used to distribute all pressure and flow information. Thus, a local dependency network can be set up and values can be explained as well as drawn back. Since this recompilation process has to be performed only once, one gets a speed up when calculating such a network for different input information (compared to the global approach).

Investigating an arbitrary hydraulic system with respect to a structure simplification is a rather complex task and encloses the following steps:

- Identification of all sources and sinks within a hydraulic system. E.g., a cylinder that extends represents one sink and one source that have to be associated with the connections, considering the direction of the piston’s velocity.
- Determination of all subgraphs that can be reduced.
- Calculation of (non-linear) substitute resistors and installation of additional constraints.
- Processing alternatives for those components whose resistance has to be suggested in order to find a globally consistent solution (throttling valves, pressure relief valves etc.).

A detailed description of concepts and algorithms concerning the problems above is given in [7].

DYNAMIC EXAMINATION OF FUNCTION

In $^{\text{art}}$*deco*, the static examination of a mechatronic system is followed by a dynamic examination of function. The description of the dynamic behavior of a system involves quantities known from statics, in particular the information responsible for the dynamic behavior.

The dynamic examination of function constitutes an independent module in $^{\text{art}}$*deco* and is currently being developed. It can be seen as a systematical extension of the model based approach in $^{\text{art}}$*deco*. The actual dynamic examination process follows the approach of an expert. In order to reduce the problem specification (to a minimal extent from the user’s view), the knowledge that is already present about the domain is used extensively. In the simplest case, it is sufficient to specify the relevant time intervals and subnetworks. The most complicated case additionally requires specifying the components’ dynamic behavior description. The basics of this conception will be outlined in the following subsections.

Modules of the Dynamic Function Examination

Figure 4 shows the module structure and the interaction of the different modules. Parallel arrows represent alternatives; nested modules correspond to an iterated execution of the embedded modules:

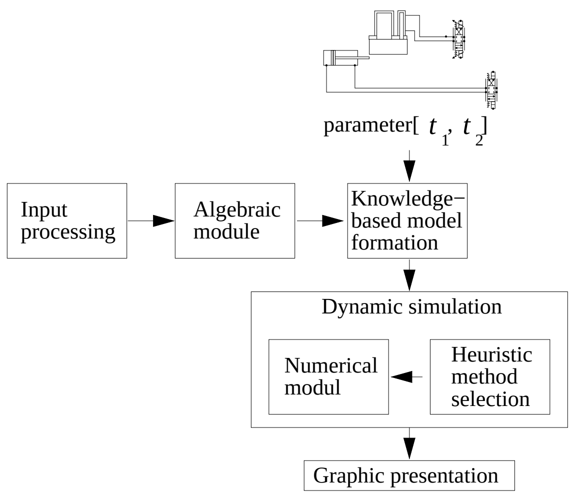


Figure 4: The dynamic checking module.

In the following the function of each module is described briefly.

- Processing the inputs takes for every component not yet known to the system a description of the dynamic behavior in form of order n differential equations which are to be provided by the user. The syntax of the differential equations is subject to a formal language definition, as for which the expressions have to be checked. A general, formal definition of differential equations can be found in [4].
- The algebraic module takes the output from processing the inputs and reduces for each new dynamically specified component the order n differential equations to a system of differential equations which is of order 1. This transformation yields normalized expressions of the form

  $$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t)),$$

  where x and f are vector-valued functions. The transformation is necessary because most of the single-step-methods are applicable for differential equation only of order 1. For a detailed discussion on single-step-methods see [6], [5].
- The module of model formation has as additional input-parameters the user selected subnetwork and the time interval needed for the dynamic simulation. Along with the known, different levels of model

depths of the dynamic component description, a knowledge driven graph algorithm yields a subcircuit of the system. This circuit always presents a superset of the components pre-chosen by the user which also have to be considered for the dynamic simulation. Each component in this superset is modeled at a different level of detail, according to the component's significance for the overall dynamic behavior of the subcircuit.

- The module of dynamic simulation carries out the dynamic function examination for that part of the system which was determined by the model formation process. The result of this examination is kept as value tables

t	t_1	t_2	$\dots$	t_n
$x_k(t)$	$x_k(t_1)$	$x_k(t_2)$	$\dots$	$x_k(t_n)$

  for the quantities x_k for every single component of the subcircuit. These methods reside inside the numerical module. The selection of the appropriate numerical method is controlled by the module of method selection. The respective choice takes place in a context sensitive manner, i.e. for each phase of integration the method that suits best according to heuristic criteria is determined and applied. Doing this, an integration phase can also comprise a set of step-widths-intervals.

- The module of graphic representation is responsible for the graphical output of the dynamic simulation. The value-tables provided by the module dynamic simulation serve as a data base. In general, the set of pairs required for good quality in the presentation of the function graph will be a subset of the pairs available. This subset depends on the size of the time interval and an appropriate scaling factor. The output procedure considers both parameters, in order to yield an optimal presentation.

Heuristic and Mathematical Domain Knowledge

The knowledge needed for the dynamic description of mechatronic network can be roughly divided into two categories. The first category comprises the mathematical knowledge that is necessary for problem specification and its solution. The second one contains the heuristic knowledge that controls the application of the mathematical knowledge.

Merely the mathematical knowledge suffices to complete describe the problem of dynamic function checking. This knowledge reduces the problem of dynamic behavior to solution of an initial value problem. The mathematical modeling of the dynamic behavior of components is presented locally for every component as ordinary differential equations at the different levels of model depths. Different levels arise from differing levels of details at which the components are being modeled. Which model depth of a component has to be chosen during model formation of a network is in the realm of heuristic knowledge.

The mathematical knowledge for solving initial value problems is independent of the system's components and is therefore represented globally. In $^{\text{art}}$*deco*, an initial value problem shall be solved numerically by means of so-called single-step-methods. The global knowledge base consists of one implicit and two explicit Runge-Kutta methods with an error of order $O(h^5)$ $(O(h^3), O(h^5)$ respectively. With each of these methods, the problem can be solved, however, the above mentioned Runge-Kutta method is more or less appropriate in specific situations. At this point, again knowledge from experience is helpful.

The entire heuristic knowledge is global, that is, it is not represented on the level of components. It falls into two parts, one that influences model formation of the components, and one which controls the selection of appropriate Runge-Kutta-methods. In both parts, the knowledge is represented in form of rules.

For a subnetwork that has to be specified by the user a graph algorithm systematically finds all paths from working elements to controlling elements to service components in the network. If there are paths leading out of the prespecified subnetwork then the heuristic knowledge gives a clue as to whether the additional components require further consideration or whether they can be discarded. In the first case, the heuristic determines the proper depth of the given dynamic component description.

Heuristic knowledge helps choosing among several Runge-Kutta methods. For one thing, it provides for switching to higher order methods during time critical periods of initial impulses in order to limit the local truncation error. After evaluating the critical time steps one can resume the computation with lower order methods. Thereby, the computational effort of dynamic simulation can be substantially reduced.

For another thing, with stiff differential equations systems being used, heuristic knowledge has to ensure that the implicit Runge-Kutta method is employed when calculation proceeds with larger step sizes. Having a larger stability range, this method prevents an accumulation of local truncation errors, thus yielding a less falsified simulation result.

The applications of heuristical knowledge mentioned above indicate only a few of the aspects that have to be considered.

DIAGNOSING FAULT BEHAVIOR

Diagosing faults in hydraulic circuits is also part of the current research. This section does not go into diagnostic details but outlines how diagnosis will be realized within ^{ar}*deco*.

Roughly speaking, diagnosing a hydraulic system means to explain observations of faulty behavior. Let's consider the following part of a hydraulic circuit:

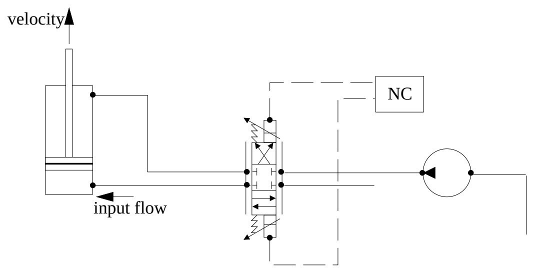


Figure 5: A simple hydraulic circuit.

If an expert tries to explain the observation that the cylinder's piston drives out too slowly, he starts his diagnosis ignoring the exact relationships between the piston plane, the piston velocity and the flow value. Rather, he follows up the piping's structure in order to determine a set of components that contains the misbehaving one. Since he knows about the components' failure probability, he is able to make a well-founded assumption as to which component out of the determined set could be defect. At this point, the expert will refine the diagnose using his knowledge about the malfunctioning *inside* of components.

When analyzing the scenario above, we identify three kinds of diagnosis knowledge:

1. Knowledge at the circuit's component level which helps to find out candidates of faulty components.

2. Discriminatory knowledge to differentiate between multiple candidates. In our example, a probability model of the components' failures is employed to perform the differentiating job.

3. Refining knowledge at the subcomponent level.

These different kinds of knowledge are not different steps of *one* diagnosis strategy like "Establish-and-Refine" or "Cover-and-Differentiate". Moreover, the distinction between three kinds of knowledge is our approach to tackle the complex diagnosis problem: Each portion of knowledge contains a diagnosis problem that needs to be solved in its own manner. The idea is, rather than looking for an overall strategy, to identify different diagnosis problems that can be solved independently. The following figure depicts the concept graphically:

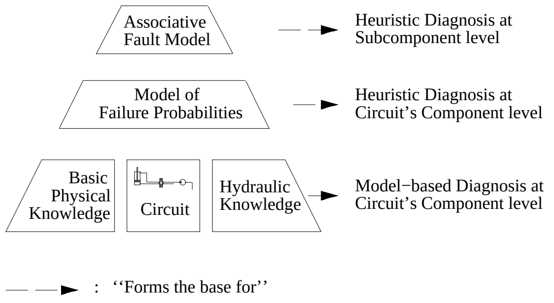

Associative Fault Model
Heuristic Diagnosis at Subcomponent level
Model of Failure Probabilities
Heuristic Diagnosis at Circuit's Component level
Basic Physical Knowledge
Circuit
Hydraulic Knowledge
Model−based Diagnosis at Circuit's Component level
: ''Forms the base for''

Figure 6: Different levels of diagnosis.

Each kind of knowledge identified above will base upon its own model in ^{ar}*deco*. The major difference between the first model and the two other is the following: The first model is founded on functional descriptions of the circuit and needs deep understanding of the underlying physics. In contrast, the second and the third model are heuristic and much easier to operationalize. In ^{ar}*deco*, a module will be provided that enables an expert to specify his heuristics for investigating components that might be broken. This knowledge will be processed, if the diagnoses at the component level of the circuit is solved.

The realization of the model-based diagnosis is the more crucial problem when diagnosing a hydraulic circuit. The idea (of model-based diagnosis in general) is to exchange parts of the correct model for parts of the malfunctioning model in a way, that the resulting model is compatible with all observations. In order to realize the diagnosis process efficiently, the exact technical behavior description of the components must be reduced to qualitative ones as in the expert's approach. Instead of using exact numerical values, symbols like "high", "too low" etc. have to be processed. A qualitative description of the components' behavior and the fluid physics has to satisfy following aspects:

- The global behavior has to be derived by local component descriptions, because no fixed circuit structure can be assumed.

- Multiple faults have to be detected, since in reality a malfunctioning can be caused by two components that are broken at the same time.

- The description must stay extensible with regard to future extensions.

Based upon such a qualitative description, powerful diagnosis mechanisms like GDE[5] can be used to perform the model-based diagnosis. Hypotheses which are generated with the GDE will form the input for the subsequent heuristic models.

[5] GDE stands for General Diagnostic Engine and is introduced in [2].

WORKING WITH $^{\text{art}}$*deco*

$^{\text{art}}$*deco* distinguishes between an application mode and an acquisition mode. A general idea is to provide equivalent concepts for "normal" user interaction as well as for acquiring additional knowledge.

The Application Mode

Within the application mode, a user selects objects from a catalog and arranges them in order to build a new hydraulic system. When drawing lines between the components' gates, the appropriate types of connection lines will be selected and instantiated. Within this connection, it will be checked if both gates incident to a connection line are of the same type. All necessary information concerning the topology is generated as well. Figure 7 shows $^{\text{art}}$*deco*'s application screen:

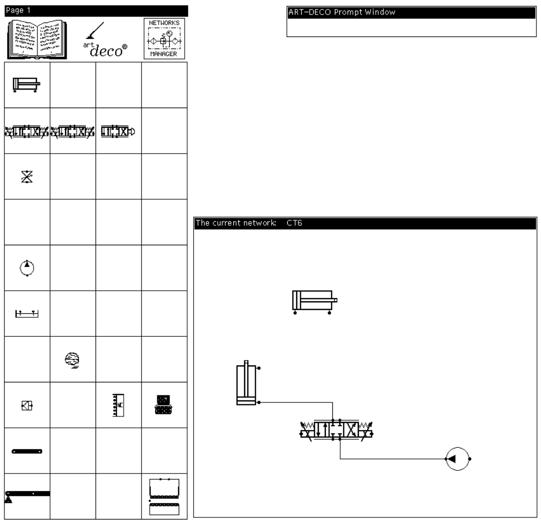


Figure 7: $^{\text{art}}$*deco*application screen.

Within a hydraulic system, all parameters belonging to components and connections like hydraulic resistances, valve positions, pressure values, or forces can be predefined or changed.

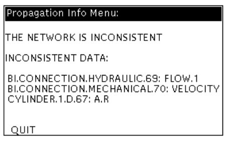
For arbitrary parts of a network, constraint propagation can be invoked any time. The derived values and possible faults can be displayed within the circuit.

While checking the circuit illustrated above, an inconsistency has been detected by the local propagation mechanism: The position of one valve contradicts some demands specified at a cylinder (cf. Figure 8).

The Acquisition Mode

When the acquisition mode is activated, new classes of components can be defined where existing classes might serve as templates. $^{\text{art}}$*deco* provides a module that enables (*i*) the generation of structured pictures and (*ii*) the specification of dependencies between picture hotspots and a component's behavior description. Interpreting this information, $^{\text{art}}$*deco* automatically generates the according data structures and constraint information for both the knowledge processing and the user interaction.

Introducing new properties within such a behavior description will cause $^{\text{art}}$*deco* to update the according component definition automatically. Figure 9 shows the $^{\text{art}}$*deco*-screen while defining a new component.



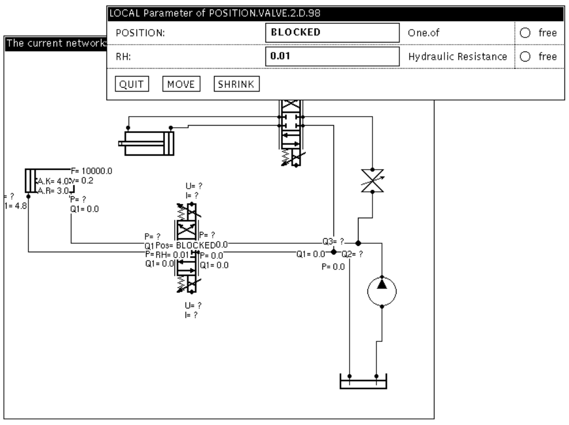
Propagation Info Menu:

THE NETWORK IS INCONSISTENT

INCONSISTENT DATA:

BI.CONNECTION.HYDRAULIC.69: FLOW.1
BI.CONNECTION.MECHANICAL.70: VELOCITY
CYLINDER.1.D.67: A.R

QUIT

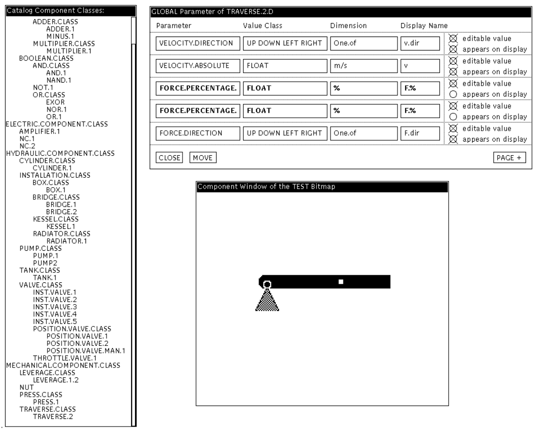


LOCAL Parameter of POSITION.VALVE.2.D.98

POSITION:	BLOCKED	One.of	○ free
RH:	0.01	Hydraulic Resistance	○ free

QUIT MOVE SHRINK

Figure 8: The value assignment after the simulation.



Catalog Component Classes:

```
ADDER.CLASS
    ADDER.1
    MINUS.1
  MULTIPLIER.CLASS
    MULTIPLIER.1
BOOLEAN.CLASS
  AND.CLASS
    AND.1
    NAND.1
  NOT.1
  OR.CLASS
    EXOR
    NOR.1
    OR.1
ELECTRIC.COMPONENT.CLASS
  AMPLIFIER.1
  NC.1
  NC.2
HYDRAULIC.COMPONENT.CLASS
  CYLINDER.CLASS
    CYLINDER.1
  INSTALLATION.CLASS
    BOX.CLASS
      BOX.1
    BRIDGE.CLASS
      BRIDGE.1
      BRIDGE.2
    KESSEL.CLASS
      KESSEL.1
    RADIATOR.CLASS
      RADIATOR.1
  PUMP.CLASS
    PUMP.1
    PUMP2
  TANK.CLASS
    TANK.1
  VALVE.CLASS
    INST.VALVE.1
    INST.VALVE.2
    INST.VALVE.3
    INST.VALVE.4
    INST.VALVE.5
    POSITION.VALVE.CLASS
      POSITION.VALVE.1
      POSITION.VALVE.2
      POSITION.VALVE.MAN.1
    THROTTLE.VALVE.1
MECHANICAL.COMPONENT.CLASS
  LEVERAGE.CLASS
    LEVERAGE.1.2
  NUT
  PRESS.CLASS
    PRESS.1
  TRAVERSE.CLASS
    TRAVERSE.2
```

GLOBAL Parameter of TRAVERSE.2.D

Parameter	Value Class	Dimension	Display Name	
VELOCITY.DIRECTION	UP DOWN LEFT RIGHT	One.of	v.dir	⊠ editable value ⊠ appears on display
VELOCITY.ABSOLUTE	FLOAT	m/s	v	⊠ editable value ⊠ appears on display
FORCE.PERCENTAGE.	**FLOAT**	**%**	**F.%**	⊠ editable value ○ appears on display
FORCE.PERCENTAGE.	**FLOAT**	**%**	**F.%**	⊠ editable value ○ appears on display
FORCE.DIRECTION	UP DOWN LEFT RIGHT	One.of	F.dir	⊠ editable value ⊠ appears on display

CLOSE MOVE PAGE +

Component Window of the TEST Bitmap

Figure 9: Component definition in $^{\text{art}}$*deco*.

SUMMARY

When configuring a hydraulic system, an engineer is faced with a lot of problems. The most important ones are: checking of the stationary behavior, dynamic examination of function, and detecting faults. Each of this problems needs a deep model to become operationalized on a computer. Within this connection, we have developed concepts for the representation and the processing of hydraulic knowledge. We have been realizing our ideas in the prototypic system ^{art}*deco* that bases upon the following key ideas:

- local description of a component's properties and behavior
- graphical specification of hydraulic problems

Rather then solving very particular hydraulic problems, our philosophy is to support the processing of standard tasks in an adequate way. This can be achieved by modeling the typical working documents of the domain experts on a computer.

REFERENCES

[1] J. de Kleer, J. S. Brown: *A Qualitative Physics Based on Confluences*. Artificial Intelligence 24, 1984.

[2] J. de Kleer, B.C. Williams: *Diagnoses with Behavioral Models*. Proc. IJCAI 1989, p.1324.

[3] H. Faatz et al.: *Der Hydrauliktrainer Band 3*. Mannesmann Rexroth GmbH, Lohr a. Main, 1988.

[4] O. Forster: *Analysis 2, Differentialgleichungen im* $\mathbf{R}^n$. Vieweg, Braunschweig, 1984.

[5] C. W. Gear: *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Inc., 1971.

[6] P. Henrici: *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley & Sons, Inc., 1962.

[7] M. Hoffmann: *Algorithmen zur Verarbeitung von topologischen Informationen in Netzwerken*. Diploma thesis, Universität-GH Duisburg, FB11 Praktische Informatik, 1993.

[8] H. Kleine Büning, B. Stein: *Ein wissensbasiertes System zur Inbetriebnahmeunterstützung*. Internal report, Universität-GH Paderborn, FB17 Praktische Informatik, 1993.

[9] B. Kuipers: *Commonsense Reasoning about Causality: Deriving Behavior from Structure*. Artificial Intelligence 24, 1984.

[10] R. Lemmen: *Akquisition und Analyse von Wissen zur Inbetriebnahme von hydraulischen, translatorischen Anlagen*. Diploma thesis, Universität-GH Duisburg, Inst. f. Meß-, Steuer-, und Regelungstechnik, 1991.

[11] P. Struss: *Assumption-Based Reasoning about Device Models*. In H. W. Fruechtenicht (ed.): *Technische Expertensysteme: Wissensrepräsentation und Schlußfolgerungsverfahren*. R. Oldenbourg Verlag München Wien, 1988.