Appeared in: Sixteenth International Workshop on Qualitative Reasoning, Sitges, Catalonia, Spain, 2002.

Automated Abstraction of Numerical Simulation Models - Theory and Practical Experience

P. Struss

Technical University of Munich and OCC'M Software
struss@in.tum.de, struss@occm.de

Abstract

This paper presents and discusses work on the automated generation of qualitative (diagnostic) models from simulation models that have been developed for (control) engineering purposes. This work is motivated by an attempt to build model-based tools that support a closer integration of diagnostic considerations in early design phases of on-board systems for vehicles and based on the insight that such an attempt has to limit the required modeling efforts. We present the mathematical foundations and the implementation of the abstraction process and discuss the various difficulties and problems encountered when we applied the software to real automotive subsystems. These difficulties include complexity and methodological issues, and what should be, but has not been, a major concern of research on qualitative reasoning: How to obtain adequate qualitative domains.

1. Introduction

Each attempt to build a model-based problem solver for some industrial application faces the problem of creating the appropriate model. It does not suffice to finally deliver such a model, it is important that there exist methodologies and tools that allow to calculate and restrict the costs for model generation. Moreover, it does not suffice to develop arbitrary methodologies and tools, it is necessary that they are related to the current practice, education, work processes, and tools in the respective engineering domain.

Automotive industry has become one of the major areas in which model-based solutions, especially for diagnosis-related tasks, are being developed at a broad scale. Although the goal is usually still prototype development for feasibility studies, the perspective of integrating the solutions into the work processes during the product life cycle has to guide this work, and, hence, the above issues have to be addressed. After a successful demonstration of the utility of model-based solutions for on-board diagnosis of vehicles in the VMBD project ([Cascio et al. 99], [Sachenbacher-Struss-Weber 00]), the European Fifth Framework project „Integrated Design Process for on-board Diagnosis" (IDD) pursues the goal to formalize and standardize the diagnostic design process and to enable the introduction of diagnosis early in the chain. This methodological goal has to be combined with another important objective: *giving to the designers a set of model-based tools that can help them in evaluating and understanding the effects of each choice on the system being designed.*

In order to achieve a close link between the different work processes, such as control design, failure-modes-and-effects analysis (FMEA), and diagnosability analysis, we decided to explore the possibilities for automatically transforming the simulation models used for control purposes into qualitative diagnostic models.

In this attempt, we developed some mathematical foundations and an implemented solution and applied it to examples of vehicle subsystems. We describe the approach in section 3. More than presenting what we solved, we are interested to discuss the difficulties and unsolved problems, because we feel that they deserve more attention and research efforts (section 4). We continue by summarizing the framework for this work, the IDD project (for more details see [R. Brignolo et al. 01]).

2. Integrating Diagnosis into the Design Process

The importance of diagnosis in on-board automotive systems is constantly growing together with the complexity of the systems. The average dimension of the diagnostic code inside a modern electronic control unit (ECU) is now more than 50% of the whole code. At present, there is no correspondence between such an important role of diagnosis in on-board systems and a similar role that diagnosis could play in the design process chain.

The correct way of dealing with this situation is *to re-organise the design and development chain so that the diagnosis is no longer the last task in the design chain.* This goal provides an opportunity and challenge to model-based systems technology for several reasons. First, in early design stages, when physical prototypes of the designed system are not existing, diagnostic reasoning can only be based on a model. Second, since the design is subject to revisions, the adaptation of diagnostics and fault analysis to such revisions has to happen automatically or, at least, without major efforts. Finally, the existence and use of (simulation) models for the development and validation of control design can

provide a basis for the application model-based diagnosis technology.

The IDD project was started February 2000 with a duration of three years and involves both industrial and academic partners: Fiat CRF (Torino), Magneti-Marelli SpA (Torino), PSA, Peugeot Citroen (Paris), Renault (Paris), DaimlerChrysler AG (Stuttgart), OCC'M Software GmbH (München), Universita di Torino, Université de Paris Nord, XIII, and Technische Universität München.

The current processes of each industrial partners have been investigated with a focus on the integration of the diagnostic process and diagnosis-related processes into the whole design process of mechatronic subsystems, and a model of a new design process has been developed.

In summary, the framework for a new process has to satisfy the requirement that the designers (the different experts involved in the design) should be supported in performing different activities, such as design of control algorithms, and their simulation, generation of the FMEA, and analysis of the diagnosability, in an interleaved way.

In representing the current stage of the design decisions, the model(s) of the current system design(s) has to mediate between the processes. Because the models currently developed for the control design are mostly numerical simulation models, a transformation into qualitative models suited for the diagnosis-related tasks becomes necessary. One way to achieve this would be through a library of qualitative component models with the disadvantage that its creation requires additional efforts. Therefore, we decided to also explore ways to automate this transformation. This results in the framework for a new process shown in Figure 1.

The foundations for the diagnosability analysis tool have been outlined in [Struss 02]. Our work on the model transformation module is described in the following section.

3. Automated Model Abstraction – Theory and Implementation

3.1 The Goal

There are several reasons for transforming the given numerical model into a qualitative diagnosis model. A fundamental reason is that the distinction between correct behavior and a fault by its very nature is a qualitative one. Secondly, a finite representation promises to provide compact models (e.g. for on-board diagnosis) and efficient consistency checks. When creating qualitative models, the key question to be answered is, "*what are the distinctions in the domains of the system variables that are both necessary and sufficient to achieve a particular **goal** in a certain **context** and under given **conditions**?*".

In previous work, we addressed this goal by developing a theory and implementation of task-dependent model abstraction, called AQUA ([Sachenbacher-Struss 01]) whose basic ideas we summarize in the following subsection.

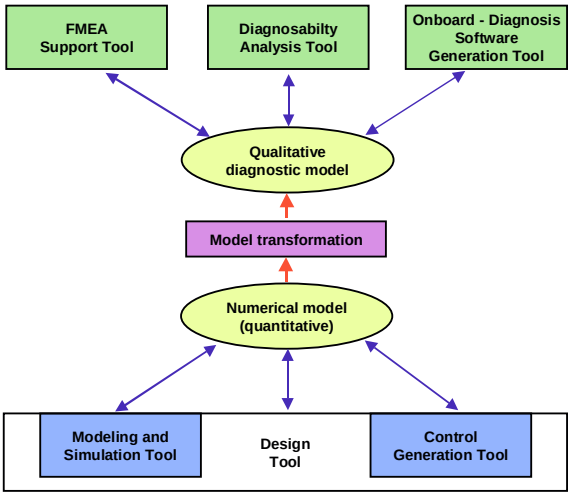

Figure 1 An Integration of model-based tools based on model transformation

3.2 Task-Dependent Model Abstraction

In AQUA, the **goal** of using a model is characterized by a set of **target partitions** of the domains of selected variables (e.g. output variables), the **context** is given by the **structure** of the modeled system, and the **conditions** are represented by a set of initial variables and their **possible distinctions** (e.g. possible observations). Then the goal of abstracting a fine-grained model can be described as finding variable domains that maintain only those distinctions that are necessary to determine the required distinctions of the target variables. In other words, we want to drop distinctions in the domains of the fine-grained model without losing its inferences concerning the target partitions.

We developed a formalization of this goal and an algorithm and implementation that takes a fine-grained model as an input and generates the task-dependent abstraction by eliminating distinctions that do not contribute to a distinction between target partitions. Usually, the partitions of the domain can be given by (finite) sets of landmarks that define qualitative values as intervals between adjacent landmarks. The abstract model will then contain a subset of the landmarks of the original model but maintain the predictive power w.r.t. qualitative values of the target variables. This solution worked quite well for a number of examples including on-board diagnosis of a turbo charger system on a real test vehicle. However, it does not provide the full solution to the problem we are addressing here.

3.3 Generation of a Finite Relational Model from an Equation Model

AQUA needs a fine-grained, but **finite** base relation as a starting point. When we are given a numerical simulation model, as in our application, or simply a set of ordinary differential equations on paper, we first have to **generate such a finite relation**.

Let us assume first, we are given

- a numerical (simulation) model that computes one output variable y as a function of n input variables, x_i:
  $y = f(x_1, ..., x_n)$
  (This is general enough, because if there is more than one output, we simply have to consider several functions),
- a set of landmark values for all input variables and the output variable
- two continuous functions, $\varepsilon_-(x_1, ..., x_n)$, $\varepsilon_+(x_1, ...x_n)$, that characterize the precision of the model, i.e. the base model is given by the envelope of f:
  $R_0(f, \varepsilon_-, \varepsilon_+) =$
  $\{ (x_1, ..., x_n, y) \mid f(x_1, ..., x_n) - \varepsilon_-(x_1, ...x_n) < y < f(x_1, ..., x_n) + \varepsilon_+(x_1, ..., x_n) \}.$

For monotonic (sections of) functions, it is straightforward to define and compute the model abstraction for a given sets of landmarks. For each tuple of qualitative input values, $(q_{1,j1}, ... , q_{n,jn})$, we have to compute the qualitative values of y that are consistent with this tuple. We define a qualitative value as an interval between two adjacent landmarks: $q_{i,j} := (l_{i,j}, l_{i,j+1})$. Then a tuple of qualitative values is the cross product of such intervals, i.e. an n-dimensional rectangle. The **corners** of such a rectangle are given by the tuples that combine the bounding landmarks:

$Corners(q_{1,j1}, ... , q_{n,jn}) :=$
$\{ (l_{1,k1}, ... l_{n,kn}) \mid q_{i,j} := (l_{i,j}, l_{i,j+1}) \wedge k_i \in \{j, j+1\} \}.$

If f is a continuous function, the consistent qualitative values of y are those that have a non-empty intersection with the interval between the minimal and the maximal value that $f-\varepsilon_-$, $f+\varepsilon_+$ take on in the rectangle. If these envelope functions are also monotonic, then these extreme points are taken on at some corners of the rectangle. Hence, we have to compute only the function values at the corners in order to obtain the (minimal) abstract relation that covers the envelope of f, as stated by the following theorem.

Theorem (Abstraction of $R_0(f, \varepsilon_-, \varepsilon_+)$ for monotonic functions)

Let $f-\varepsilon_-$, $f+\varepsilon_+$ be continuous and monotonic for each x_i, $\{q_{i,j}\}$ the qualitative values for x_i, and $\{q_{y,j}\}$ the ones for y. For each tuple of qualitative input values, we define the range of the output y as

$int_y\{ (q_{1,j1}, ... , q_{n,jn}) :=$
$[\min \{(f-\varepsilon_-)(cor) \mid cor \in Corners(q_{1,j1}, ... , q_{n,jn}),$
$\max \{(f+\varepsilon_+)(cor) \mid cor \in Corners(q_{1,j1}, ... , q_{n,jn})].$

Then

$R_{abstr}(f, \varepsilon_-, \varepsilon_+) :=$
$\{(q_{1,j1}, ... , q_{n,jn}, q_{y,j}) \mid q_{y,j} \cap int_y\{(q_{1,j1}, ... , q_{n,jn}) \neq \emptyset\}$

is an abstraction of $R_0(f, \varepsilon_-, \varepsilon_+)$, i.e.

$R_0(f, \varepsilon_-, \varepsilon_+) \subseteq R_{abstr}(f, \varepsilon_-, \varepsilon_+)$,

and it is minimal, i.e. any proper subset $R'_{abstr}(f, \varepsilon_-, \varepsilon_+)$ of $R_{abstr}(f, \varepsilon_-, \varepsilon_+)$ is not an abstraction of $R_0(f, \varepsilon_-, \varepsilon_+)$.

The example in Figure 2 shows the abstract relation as a set of shaded rectangles covering the envelope around the function - for the monotonic sections. It also illustrates that $R_{abstr}(f, \varepsilon_-, \varepsilon_+)$ may fail in regions where the envelope has a maximum (or minimum).

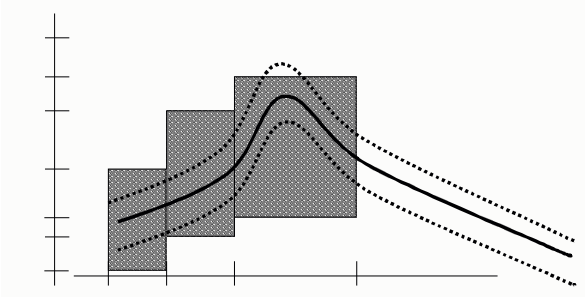

Figure 2 Abstraction of the envelope of a function

To avoid this, landmarks have to be chosen in an appropriate way, i.e., intuitively, there have to be landmark tuples of the input variables „close enough“ to the location of the maximum, such that the respective value of the function lies in the qualitative output value that covers the maximum value, as illustrated in **Figure 3**. Without giving details, we state that, if $f-\varepsilon_-$, $f+\varepsilon_+$ are differentiable, it is possible to compute what „close enough“ means, based on bounds of the derivatives around the extreme points. However, we also point out that this requires an analysis of the details of the function which either involves the user or requires some automated approach.

This is not the only caveat to be considered.

The procedure based on the theorem gives us a way to generate a finite relation that covers the numerical base relation **for a given set of landmarks**. However, we have to make sure that the chosen landmark sets still maintain the relevant distinctions the numerical model could derive.

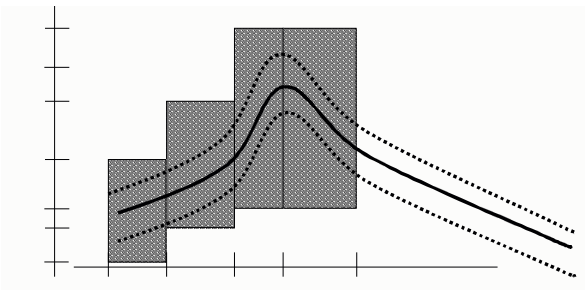

Figure 3 Abstraction covering the maximum

There are only two factors that influence and provide boundaries to the choice of the model granularity represented by the landmark sets:

- For observable variables, it must not be below the level of observable distinctions.
- There is no need to introduce more than the target distinctions to the respective variables (unless they contribute to deriving target distinctions of other variables).

For all other variables, we have no a priori criteria for selecting appropriate landmarks. They can only be derived from the above restrictions and the model. This constitutes a first major problem. Of course, a second one can be that the precision of observations is high, and so is the number of possible initial landmarks for observables.

3.4 Abstraction of Ordinary Differential Equations

If we have to create an abstract model of a dynamic system, some of the variables are derivatives of other variables. They do not require any special treatment in the computation. Ordinary differential equations are treated just like algebraic equations and transformed into qualitative constraints involving variables and derivatives.

3.5 Automated Abstraction of a Numerical Simulation Model

In order to implement the model abstraction, we need to compute the output values of landmark tuples (as corners of qualitative value tuples). But a means for this computation was the starting point for our work: the numerical simulation model. In our implementation, the numerical model is a MATLAB/Simulink model. Such a model consists of a set of subsystems that have a number of interconnected input and output values and are possibly organized in a hierarchy. These subsystems, at some appropriate level, will be the entities that are subject to the abstraction procedure that was outlined above.

However, this model is a **simulation** model and computes system behavior over time. In particular, it contains integration steps, and, hence, the value of some variables may refer to a later time point than the inputs to the system. We have to make sure that the qualitative constraints link only values that occur at the same time and therefore, we have to apply some surgery to the subsystem structure: any integration block (or, more generally, any block that involves a delay) is eliminated from the respective subsystem, its input (the derivative) becomes an output of the modified subsystem, and its output (the integrated value) is treated as the input to the remainder of the computation, if there is any. Figure 4 illustrates this procedure.

Simulation freaks are usually horrified by this change in the model and suspect that it eliminates the dynamics of the system. However, this fear is not justified and results from a procedural view on dynamic systems as it is represented by the Simulink model. Nobody would claim that the set of differential equations

d/dt x = a*y
d/dt y = b*x

does not capture the dynamics of the system, just because it does not mention integration.

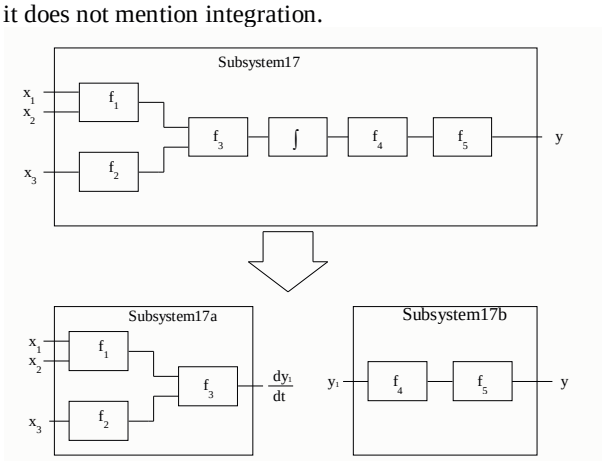

Figure 4 Eliminating integration steps

From the simulation model that corresponds to the equations, we obtain two blocks for abstraction as illustrated in Figure 5.

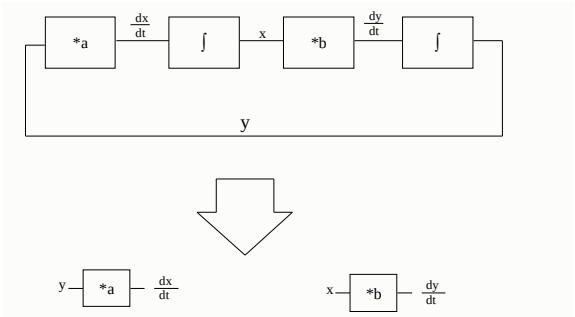

Figure 5 The restructuring of the simulation model yields the differential equations as blocks

They correspond to the equations to be abstracted, and, hence, exactly the intended result will be produced. Stated differently: the integration blocks in the simulation model do not capture anything that is specific to the modeled system, but only (the implementation of) the general simulation algorithm which is totally independent of this system. For the abstract model, they would have to be replaced by a qualitative simulation algorithm rather than a model fragment.

Parameters that occur in the subsystems, such as the coefficients a and b in the example, are treated as subsystem inputs. After these modifications to the Simulink model, we can apply the abstraction procedure to the resulting subsystems (which possibly miss some connections via integration blocks) and obtain the

abstract relation of the entire Simulink model as the join of the abstract relations of the various subsystems. In our implementation, we use MATLAB/Simulink to compute the tuples of the abstract relation (i.e. the output values at the corners, their minima and maxima and the consistent qualitative output values) and store these tuples as a constraint in the modeling framework of RAZ'R ([RAZ'R 02] which is based on ordered multiple decision diagrams (OMDD).

This implementation has been applied to examples of car subsystems including a model of the air intake of a vehicle that had been developed for control purposes. In these exercises, a lot of difficulties had to be overcome, and we think that some of the problems, although quite fundamental to qualitative modeling, have not sufficiently or not at all been addressed by the research community. In the following, we discuss some of the major problems that occurred in order stimulate discussion and work on these issues.

Our goal is to automatically generate a qualitative model for diagnosis, and we distinguish two sets of problems, namely related to

- the generation of a **qualitative** model and to
- the generation of a **diagnostic** model.

4. Practical Obstacles to Automated Model Abstraction

4.1 Computational Complexity

The first obstacle one faces is a fairly obvious one: the combinatorial explosion that lurks in the algorithm. In the air intake example, the model contains 9 subsystems at the top level (corresponding to components). Four of them have 7 inputs, two others have 5 and 6, respectively. Even if each variable domain had only 10 landmarks, The computation of the output for 10^7 landmark combinations is not feasible. Furthermore, the output value of each landmark combination is needed for qualitative tuple generation of up to 2^n qualitative input value combinations, where n is the number of input variables. Caching of these computed outputs has its limitation, and beyond this, re-computation cannot be avoided.

In response to this, subsystems with many inputs had to be split into a number of subsystems with less inputs. In our experiments, we handled blocks with up to three inputs and around 60 landmarks. This lead to runtimes of several hours, which does not necessarily constitute a serious obstacle when compared to the time spent on writing diagnostics by hand. However, other effects matter: in general, „smaller subsystems" implies longer computational chains of subsystems connected by intermediate variables.

4.2 The Curse of Intermediate Variables

This situation raises two problems.

First, due to the finite granularity of the domains of intermediate variables, „**rounding errors**" occur at the interface of two subsystems. This happens because the output for a landmark combination does, in general, not hit a landmark of the output. The rounding error on average amounts to one qualitative value (interval) of the output, and it is guaranteed. The errors of several inputs of a subsystem combine, and they accumulate along the computational chain. If we assume such a chain consisting of k subsystems whose (single) output is a linear combination of its inputs, the rounding error amounts to the sum of the width of k intervals. When a subsystem (component) with n inputs is decomposed into smaller subsystems with, say two inputs each, the number of these subsystems, k, may well be in the order of n. If $|d|$ denotes the cardinality of the output domain, then the overall result of the linear combination with the rounding error included will cover the entire domain with a ratio of $k/|d|$. For the case of non-linear subsystems, it can be worse. For instance, the decomposed components in our examples had up to 13 subsystems which means that even with, say, 30 landmarks, the abstracted model will hardly yield any restrictions. Hence, we need to increase the number of landmarks.

This leads to the second problem: determining appropriate sets of landmarks.

4.3 Fundamental: How to Determine Initial Landmarks?

Here, the first question to answer is what **range to cover** with the landmarks. While the input and output variables of components usually have some physical meaning that helps to guess the possible range, this is not the case for the intermediate results of the computation. This makes it extremely difficult to guess the range of the values that can occur. This holds even more, if we want to derive a model for diagnostic purposes, since we have to make sure that this range does cover all possible conditions, including fault situations. Actually, this also applies to many component inputs and outputs: for instance, how can we estimate what values the derivative of some unmeasured pressure at the interface of two components can take on under the sudden occurrence of a leakage?

To some extent, the answer lies in the model: together with the ranges of input variables it determines the possible range of the output. For monotonic functions, we can easily compute the extreme points, and this is what we actually partly did in our experiments. However, the Simulink model can (and, in practice, usually will) contain not only nonlinear analytic functions, but also tables with empirical data and even black-box model fragments with C code which makes it extremely difficult or impossible to compute the extreme points in a deterministic way. One will have to run simulations on real data and determine the extreme points occurring, and also this we did. Obviously, there is no guarantee for covering of the range of behaviors, and this holds even

more when the model has to cover all possible fault situations (see also the discussion in section 4.6).
Under these circumstances, the second problem is even harder to solve: the **selection of landmarks** within the decided range. For physical variables, domain experts may be able to propose some important distinctions as candidates for landmarks. But these are unlikely to suffice, and for intermediate variables, there is hardly any alternative to choosing landmarks by an equidistant partitioning of the range. Unfortunately, this enforces again the introduction of many landmarks in order to avoid large rounding errors, because under this goal, the number of landmarks is determined by smallest distinctions required.
Another incarnation of the landmark selection problem occurs in the frequent case where there are certain given landmarks for the output and for some inputs of a subsystem, and appropriate landmarks for the other inputs should be determined. While for AQUA, this is a non-problem due to the relational representation, we become a victim of the directionality of the Simulink model which does not allow to compute inputs from outputs. Again, the only compensation is to select a fine-grained partitioning of the input domain.

4.4 Discussion of the Dilemma

In summary, there is a „positive feedback" loop in the interdependencies among the problems discussed which can be summarized as follows:

1 If we have no „informed method" for selecting landmarks, we have to select equidistant ones.
2 If we choose equidistant ones, in order to avoid rounding errors, we have to choose many.
3 If we choose many landmarks, we have to reduce the number of inputs due to the combinatorial complexity.
4 A reduction of the number of inputs leads to more subsystems and more intermediate variables.
5 For the intermediate variables, there are no clues for the landmarks ($\rightarrow$ 1.).
6 More intermediate variables introduce more locations for rounding errors, hence more landmarks are necessary ($\rightarrow$ 2.).

As a result, we have a positive feedback influence on complexity, and for some of the examples we considered, this rendered a solution infeasible, so far. In addition, the model becomes more complex both in terms of its structure and the landmark sets. This turned the steps that required our intervention (such as determining ranges) more cumbersome and error prone.
But perhaps a solution lies in the opposite direction: work with few landmarks, abstract larger blocks with more inputs and, thus, avoid rounding errors. We did not find much evidence that this is possible for interesting examples, because any starting point in terms of chosen landmarks raises the third issue and enters the cycle. Main reasons for this lie in the fact that, for many variables, no a priori selection of good landmarks seems possible, and that the form of the simulation model, especially its directed computation, prevents to obtain them.

4.5 Refinement instead of Abstraction – A Solution?

The problems discussed above are significant. They are not related to the automated abstraction procedure AQUA itself, but to the step of creating its basis, a finite relation which is fine-grained enough to preserve the required distinctions. These difficulties could be avoided if we go the opposite way: start with a coarse model and refine it where necessary (see [Sachenbacher 01]).
The basic idea can be described as follows: for each variable of a subsystem, we determine some (small) initial set of landmarks and generate the respective abstract relation as described above. If a qualitative value of some variable occurs in many tuples, it is identified as a candidate for refinement and split into two or more intervals by the introduction of additional landmarks. Then relation abstraction is applied using the extended landmark set, and this is repeated until we end up with a satisfactory model granularity.
As an illustration, consider the function envelope displayed in Figure 6. As the abstract relation shows, qx_3 occurs in 5 tuples and, hence, will lead only to weak predictions of the value of y.
A closer look at this approach reveals a number of problems and caveats:

- The outcome strongly depends on the **choice of initial landmark sets**.
- The question arises **which variables are subject to refinement**. For instance, qy_6 occurs in 6 tuples. If it is refined, this might also trigger a further refinement of qx_3 ... qx_8 which may lead to more landmarks of y and so forth.

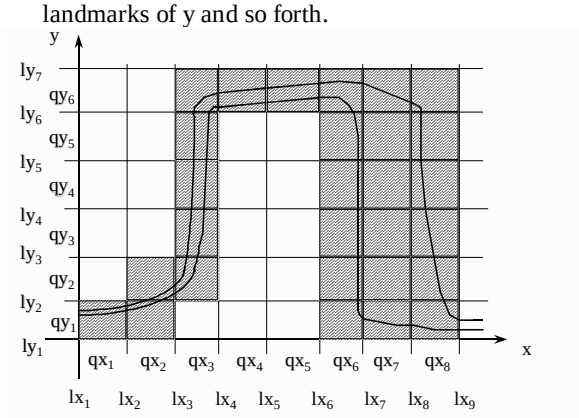

Figure 6 Abstract relation (shaded rectangles) covering the envelope of the numerical function

- It is not obvious whether the process **terminates** before reaching the granularity of the original relation and which termination criteria are appropriate.

- qy_2 occurs in 5 tuples, but a refinement of this qualitative value may not be necessary, because **not all of them are adjacent**.
- Finally, without a detailed analysis of the original relation, some **useless refinement** steps might be attempted. For instance, qx_7 would be divided although this will not provide any benefit. The problem is how to detect this. That a first split of the interval results in qualitative values that are consistent with the same qualitative values of y does not suffice: This criterion would also stop the refinement of qx_3, although this will eventually yield a useful refinement.

The key concepts for addressing these issues are the same as for task-oriented qualitative abstraction: target distinctions and observable distinctions, since both limit the refinement to be performed: refinement below observable distinctions does not make sense, and also the refinement of target distinctions is excluded unless required by some other target distinction.

Thus, the refinement strategy could be stated informally as follows:

1. For target variables, choose the target distinctions, for other landmarks, choose few landmarks well above the observable distinctions and make this **the current landmark set L_{curr}**.
2. **Generate the abstract relation**, $R(L_{curr})$ for the current set of landmarks.
3. For all tuples $q_{obs}=(q_{obs,1}, \ldots, q_{obs,k})$:
   Compute the target index, i.e. the maximal number of adjacent qualitative target tuples that are consistent with q_{obs}.
   Compute also the target indexes of the landmark tuples that are corners of q_{obs}.
   If the target index of q_{obs} *differs significantly* from the indexes of its corners,
   Then For all j
   If $q_{obs,j}$ has a refinement above the observable distinctions,
   Then *add l landmarks* for $q_{obs,j}$ to L_{curr}.
4. If L_{curr} has changed, go to 2
5. Perform **task-dependent model abstraction** on $R(L_{curr})$.

There are some issues and problems in this solution that need explanation and discussion:

- The **index** of qualitative values and landmark tuples is a measure of the imprecision of the predictions that can be derived from them (namely the size of the entailed disjunction). For instance, the index of lx_4 in Figure 6 is 1 (because it is consistent only with qy_6), while the one of lx_7 is 6. The idea is that the index of landmark tuples reflects the inherent imprecision of the model (because landmarks represent „exact input“) and that the indexes of the landmark tuples that are corners of a qualitative value can be used as a reference for its own index. It cannot be smaller than the minimum of the „corner indexes“, but should not be significantly greater than the maximum. This idea supplies a set of possible heuristics for deciding whether or not splitting a qualitative value promises to create a model that improves the determination of target distinctions. Such a heuristic would suggest to split qx_3 (its index 5 is significantly greater than the index 1 of its corners) but not qx_7 whose index of 6 is equal to its corner indexes.
- The deviation of the index of a qualitative value from its corner indexes could also be used for determining the **number of landmarks** to be introduced in one step.
- However, there remains the problem **where to place** the additional landmarks. The example of qx_6 shows that a half-split approach or, more generally, the introduction of equidistant landmarks may generate many useless landmarks and many iterations. Again, a more informed choice of the additional landmarks would require a more detailed analysis of the functional interdependency and is either in conflict with the goal of automation or bears a significant increase in the computational efforts.
- **Running abstraction** (step 5) is necessary in order to eliminate both ineffective distinctions that might have been introduced in the refinement phase and previous landmarks that may have become obsolete. One might be tempted to perform this after each refinement step in order to avoid computations in subsequent steps. But this comes at the price of some bookkeeping that prevents the algorithm from re-introducing the discarded landmarks. This does not apply to the second kind of elimination, though.

In summary, there are serious problems and complexity traps lurking also in a realization of the refinement approach sketched above as an alternative to abstraction. A closer and more formal analysis and experiments are required in order to determine its feasibility and practical value.

Finally, it is worth noting that, in a sense, the algorithm outlined above, in omitting a refinement of target distinctions is based on an implicit assumption about the completeness of the predictor that uses the resulting model. As an illustration of this fact, consider the trivial example of a system that is composed of two equality constraints: x=y and y=z where x is observable with the integers as a landmark set, y has a target distinction $\{[-\infty, 0], [0, \infty]\}$, and z's target distinction is given by the landmark set {-1, 0, 1}. If the predictor for the abstract model uses local propagation via y (or the join of the individual abstracted relations) to determine z, it is necessary to introduce the landmarks -1 and 1 for y, while this is not necessary if we operate on the abstraction of the join of the fine-grained equality relations.

4.6 Problems in Generating a Qualitative Diagnostic Model

The difficulties discussed above are independent of the particular purpose of the model and its abstraction. In our application, the abstract model is meant to support diagnosis-related tasks. This creates additional requirements, and we met further difficulties in the attempt to satisfy these model requirements based on simulation models that were originally developed for purposes of control. We list the most important problems we encountered.

1 **Needed: a component-oriented model.** For achieving the simulation of the system behavior, the component structure of the respective device is fairly irrelevant. As a result, the subsystem structure of the model does not necessarily reflect the component structure of the device. For instance, a certain pipe might not occur at all in this model. But if diagnosis has to consider the possibility of a leakage or clogging, the component has to represented and modeled.

2 **Needed: Preservation of the physical structure.** A typical example of a violation of this requirement is that input and output flow of an aggregate device were identified by an equations which, again, is based on the assumption of normal behavior (no leakage occurring).

3 **Needed: models of faulty behavior**. They are required if we are not only interested in fault detection, but fault (class) identification (as in on-board diagnostics), diagnosability analysis, and FMEA. As long as control is considered as controlling the device under normal conditions, faults are not considered in the development of the control algorithms. As a result, they are not part of the respective simulation model. Extending this to include fault models is not always trivial. If faults correspond to deviating parameters, it is fairly straightforward, but in the general case, faults may change the structure of the model radically. For instance, introducing a pipe with the potential to have a leakage means introducing another state variable and affecting the possibility to simulate the model.

4 **Needed: a physically correct simulation model**. Since the diagnosis approach is based on identifying discrepancies between a certain behavior mode (OK mode or some fault) and the respective model, it is crucial that the real physical behavior is actually covered by (the envelope of) the model. If this is not the case, e.g. because the error functions ε-, ε+ are difficult to estimate, diagnosis runs the risk of detecting model faults rather than component faults and, hence, of generating wrong diagnoses. While we may assume that the normal behavior is properly covered if the model satisfies the needs of control, it also has to correctly model the behavior if a fault occurs. We found many examples where the models of components were based on an implicit assumption about overall normal behavior. This may be addressed by an appropriate modeling methodology. However, there is a serious limitation: in particular for complex components, we may lack first principles models, and the simulation model contains characteristic maps that contain empirical data. In this case, the conditions under which these date were obtained (typically normal conditions) are compiled into the model in a way that is hard or impossible to detect.

We should note that only few of these difficulties really stem from an inappropriate modeling process or modeling faults. Rather, it is the purpose of the simulation models, namely simulating correct behavior for control purposes, that is in conflict with the diagnostic requirements. Without integrating the views and the work processes concerning system development for control and diagnosis, this will be difficult to change.

5. Conclusions

The exploitation of model-based systems in industry will greatly depend on the (additional) modeling efforts they require. This lead us to the attempt of reducing these efforts by automated conversion of existing simulation models into abstract models suited for model-based problem solvers. The problems we encountered and described in this paper are significant, concerning both the automated model abstraction and the generation of a diagnostic model. One origin lies in unsolved theoretical and technical problems, and one purpose of this paper is to stimulate research into these problems. But one also has to realize that another class of problems is due to „cultural“, educational, and organizational issues which are, at least, equally difficult to overcome.

Acknowledgements

Many thanks to all partners in the IDD project and, in particular, to Oskar Dressler, Michael Esser, and Alessandro Fraracci for their contributions to this experiment. The reviewers provided useful comments. This work is supported by the Commission of the European Union (Project no. G3RD - CT199-00058).

References

[Brignolo et a. 01] R. Brignolo, F. Cascio, L. Console, P. Dague, P. Dubois, O. Dressler, D. Millet, B. Rehfus, P. Struss. Integration of Design and Diagnosis into a Common Process. In: Electronic Systems for Vehicles, pp. 53-73. VDI Verlag, Duesseldorf, 2001.

[Cascio et al. 99] F. Cascio, L. Console, M. Guagliumi, M. Osella, A. Panati, S. Sottano, D. Theseider-Dupré: Strategies for on-board diagnostics of dynamic automotive systems using qualitative models, AI Communications, June 1999.

[RAZ'R 02] Raz'r Version 1.6, Occ'm Software GmbH, see http://www.occm.de/

[Sachenbacher-Struss-Weber 00] M. Sachenbacher, P. Struss, R. Weber: Advances in Design and Implementation of OBD Functions for Diesel Injection Systems based on a Qualitative Approach to Diagnosis, SAE World Congress, Detroit, USA, 2000.

[Sachenbacher 01] M. Sachenbacher: Automated Qualitative Abstraction and its Application to Automotive Subsystems. PhD thesis, Technical Univ. of Munich, Computer Science Department, 2001

[Sachenbacher-Struss 01] M. Sachenbacher, P. Struss: AQUA: A Framework for Automated Qualitative Abstraction. In: Working Papers of the 15th International Workshop on Qualitative Reasoning (QR-01), San Antonio, USA, 2001

[Struss 02] P. Struss: Model-based Tools for the Integration of Design and Diagnosis into a Common Process - A Project Report. In: Working Papers of the 13th International Workshop on Principles of Diagnosis (DX02), Semmering, Austria, 2002.