Constraint-based Models for Qualitative Reasoning on Dynamic Systems

Andrea Panati
Dipartimento di Informatica
Università di Torino
Torino, Italy
panati@di.unito.it

Daniele Theseider Dupré
Dipartimento di Scienze e Tecnologie Avanzate
Università del Piemonte Orientale
Alessandria, Italy
dtd@mfn.unipmn.it

Abstract

In this paper we describe qualitative constraint-based modeling and reasoning strategies for simulation and diagnosis of continuous dynamic systems.

We introduce a Qualitative Deviations Equations language for modeling the correct and faulty behavior of dynamic physical systems. We then address the task of implementing the language in a constraint processing framework, using heuristics (cycle cutset decomposition and variable ordering) to exploit the structure of the model. We also describe a causal semantics for our models, and a causal-directed efficient qualitative simulation algorithm.

We report results relative to the application of the methodologies to a fuel injection system in the automotive domain.

1 Introduction

Diagnosis of dynamic controlled systems is an important subarea of model-based reasoning and diagnosis. In this paper we address the task of simulation and diagnosis of continuous dynamic systems given a qualitative deviation equation model [12] (introduced in section 2) which is a constraint-based model.

We therefore show (section 3) how to efficiently implement constraint processing for such a language using heuristics (cycle cutset decomposition and variable ordering) to exploit the structure of the model. A greedy approach to compute an approximation of the minimum cycle cutset for a bipartite graph with preferred side is presented and compared with related work in the CP and OR communities.

We then describe (section 4) a causal semantics for our models, and a causal-directed efficient qualitative simulation algorithm.

Section 5 discusses the application of the modeling methodology and implementation architecture to the diagnosis of a real-life automotive subsystem, with significant efficiency results.

2 Qualitative Deviations modeling

In recent years, there was an increasing interest in the automated design, control, simulation and diagnosis of dynamic physical systems, given that most technical devices that require automated control are dynamic [12; 3; 4; 14]. One of the most challenging issues when dealing with such systems is the choice of adequate modeling languages. Qualitative modeling and reasoning techniques are often used in order to cope with the complexity of the tasks.

Qualitative Reasoning (QR) about continuous systems requires domain *abstraction* over a finite set of values. One such abstraction is the sign algebra with domain $S = \{-, 0, +\}$. Operations, such as addition and multiplication, are defined on this sign domain as relations ($\oplus$, $\otimes$) rather than functions, sincein some cases (e.g adding a positive and a negative number: $+ \oplus -$) the result is ambiguous.

In many applications (such as model-based diagnosis, system analysis, qualitative simulation) a system is modeled in terms of *differential equations*, possibly including appropriate *parameters* for components, whose values correspond to different (correct or faulty) *behavior modes* of the component. For example, the equation: $\frac{d}{dt}p = f_{in} - f_{out} - f_{leak}$ describes the behavior of the pressure in a container: the pressure derivative depends on incoming and outcoming flows, including possible leaks.

Moreover, in controlled dynamic systems, the system variables are often varying according to the system state and a number of inputs; therefore a *normal range* of values cannot be easily given. In these cases it is necessary or convenient to reason in terms of relative (instead of absolute) behaviors; for this reason *deviations* are introduced. Each variable x has a *reference value*, and the deviation of x is defined as the difference between its actual and its reference values, i.e. $\Delta x = x - x_{ref}$ [12]. From an ordinary differential equation we derive the corresponding qualitative deviations equation simply applying the deviation (Δ) and the sign operators ([.]) to both sides of the equation. For the previous example equation we obtain:

$$[\Delta \frac{d}{dt} p] = [\Delta f_{in}] \ominus [\Delta f_{out}] \ominus [\Delta f_{leak}] \quad (1)$$

Qualitative reasoning across time [9] involves two fundamental types of reasoning:

- *intra-state* reasoning, which in this case means solving the CSP corresponding to a set of equations and assigning a qualitative value to variables and their derivatives;
- *inter-state* reasoning, which involves reasoning on constraints relating values of variables (and their derivatives) in a qualitative state and its successor.

For diagnostic purposes, in addition to the model of general behavior, each component has an associated set of fault modes which assign a value to a component parameter, thus providing an additional constraint.

In the following we restrict our attention to the case where each qualitative equation can be written in the form:

$$s = (f_{11} \otimes \ldots f_{1k_1}) \oplus \ldots \oplus (f_{p1} \otimes \ldots \otimes f_{pk_p}) \qquad (2)$$

where s and f_{ij} are qualitative variables over the sign domain S, and $\oplus$ and $\otimes$ are the qualitative addition and subtraction operations. We call this language L_{QDE} for brevity.

3 Constraint Programming heuristics for QDEs

Let us now discuss the efficient implementation of L_{QDE} using constraint processing over finite domains.

Qualitative operators are ternary relations over S. The translation from qualitative equations to a set of ternary constraints is easily done automatically. We already observed that sign addition ($\oplus$) is sometimes ambiguous; the relation is satisfied by 13 combinations of values out of the 27 possible ones. Recent contributions to the field of phase transition [5] show that hardest CSPs are often characterized by constraints being satisfied by about 50% of the possible value assignments. In the following, we introduce two heuristics that proved to be effective on the class of CSPs considered here, relying on the structure of the constraint graph.

3.1 Cycle-cutset heuristics

One useful heuristic is *cycle cutset* decomposition [7]. The basic idea of the *cycle cutset* decomposition technique is the following: any given constraint graph can be made a tree after deleting certain vertices such that all the cycles from the graph are removed. This set of vertices is called the *cycle cutset*. If a small cycle cutset can be easily found, a good heuristic is to first instantiate all the variables in the cycle cutset and then solve the resulting tree-structured CSPs. In fact [10], tree-structured constraint graphs can be efficiently solved without backtracking simply at the cost of achieving arc consistency. [1] Moreover, the resulting subproblems can be potentially solved independently (e.g. in a parallel and/or distributed environment) because they do not share variables.

The problem of computing a minimal cycle cutset is however NP-hard, so we can usually conveniently rely on an approximate solution. In particular, in order to apply this heuristic to CSPs derived from qualitative equations systems, it should be noted that:

- In most cases, the number of variables appearing in more than one qualitative equation is a small fraction (less than 1/3) of the total number of variables;
- Each equation of the form (2) (if it does not contain more than one reference to the same variable) gives rise to a tree-structured constraint graph.

Given a system of qualitative equations of the form specified in (2), a naive solution as a non-optimal cycle cutset is the set C of variables appearing in more than one equation (and thus potentially responsible for introducing cycles in the constraint network).

3.2 Greedy cutset reduction heuristic

Since the cardinality of C could still be too large, so that the number of labelings for variables in C blows up, we might be interested in:

- reducig the cardinality of C, but at a low computational cost;
- label the remaining variables in a "convenient" order.

To address these issues we propose the following cutset-reduction heuristic:

1. Construct the bipartite graph $(E \cup C, A)$. where E is the set of qualitative equations, C is the set of variables which occur in more than one equation (initial cutset, as discussed above), and A is the set of edges representing the relation of "occurrence" of a variable into an equation;
2. Sort nodes in C by decreasing degree, resolving ties with some efficient heuristic such as a fixed ordering[2];
3. Let $C' = \emptyset$ be the reduced cutset we are going to compute, and $X = E$ the set of equations we should check;
4. Until $|C| < 2$ or $|E| < 2$ do:
   - If a node $e \in X$ with $deg(e) < 2$ (i.e. less than 2 incident arcs) exists then call $check\_node(e, (E \cup C, A))$;
   - else choose the first node $v \in C$ using the $\leq$ order computed in step 2; let $C' = C' \cup \{v\}$, $X = \{e \in E | (v, e) \in A\}$ (i.e. X is the set of nodes connected to v); delete node v and all the arcs incident on v from the graph;
5. Return C' as reduced cutset.

The problem of finding a cycle cutset in the constraint graph reduces to finding a cycle cutset in the bipartite

[1] If the size of a cycle cutset of an n-variables CSP is m, then the original CSP can be solved in $O(d^m + (n-m)d^2)$ steps, where d is the size of the largest domain.

[2] In other words, define a total order $\leq$ on the set C such that $\forall v_i, v_j \in C : v_i \leq v_j \leftrightarrow (deg(v_i) \geq deg(v_j) \vee (deg(v_i) = deg(v_j) \wedge (i < j)))$, where $deg(v)$ denotes the degree of node v.

```
procedure check_node (node e, graph (E ∪ C, A))
    d_e = deg(e);
    if d_e < 2 then
        remove node e from E;
        if d_e ≠ 0 then
            remove arc (e, v) from A;
            d_v = deg(v);
            if d_v < 2 then
                remove node v from C;
                if d_v ≠ 0 then
                    remove arc (v, e') from A;
                    check_node (e', (E ∪ C, A))
                endif
            endif
        endif
    endif
endproc
```

graph $(E \cup C, A)$. Notice that we use the a priori knowledge that all original qualitative equations (and all corresponding sets of constraints in the CSP model) are tree-structured, thus we reason on clusters of constraints (one for each qualitative equation) and not on single constraints; this drastically reduces the complexity of the algorithm. This heuristic reduces the size of the initial cutset, and provides a variable ordering for labeling variables in C' (the $\leq$ total order).

The algorithm is essentially based on a greedy approach. The worst case time complexity of the algorithm is $O(|C| \cdot |E|)$, since we consider (i.e. remove) each node $v \in C$ (except the last one, in the worst case) and then check its adjacent nodes; this requires $O(|E|)$ steps[3].

An important remark: the problem of finding a minimum cycle cutset on an undirected bipartite graph *with a preferred side*[4] is not a subproblem of the minimum cycle cutset problem on generic graphs (see e.g. [1]). Some work in progress has already determined that this minimum cycle cutset problem on undirected bipartite graphs with a preferred side:

- remains an NP-hard problem;
- can be usually efficiently solved using a branch and bound approach and special heuristics;
- presents however some really hard instances, especially when $|A|$ is of the same order of $|C| + |E|$.

4 Adding causality to QDEs

A qualitative equation of the form $a = b \oplus c$ is normally understood as a constraint (i.e. non-directional relation) between the three qualitative entities a, b, and c. However, when modeling a real-life system, we usually have some additional information: a causal understanding of the system.

For instance, by adding causal semantics to the following QDE which models the pressure in a pipe:

$$\frac{d}{dt}\Delta p \Longleftarrow \Delta f_{in} \ominus \Delta f_{out} \tag{3}$$

we mean that the (deviation of the) pressure derivative $\frac{d}{dt}\Delta p$ can change its value only if at least one of Δf_{in} (incoming flow), or Δf_{out} (outcoming flow) change, but a variable on the right hand side of the equation will not change, at least as a direct result of this relation, if one of the other variables changes (it can only change due to a feedback loop involving other equations).

This, like the use of causality in reasoning about actions [11], provides a stronger semantics to qualitative equations, restricting the set of solutions for each constraint, given the way the constraint was satisfied in the previous state. This greatly improves inter-state reasoning; the causal structure is also used to develop a linear time backtrack-free constraint propagation for intra-state reasoning.

In the following we assume that equations are of the form $v \Longleftarrow expr$, meaning that v causally depends on variables in $expr$. Such an equation can then be seen as "defining" v, with all the variables influencing v on the right hand side. Each variable should have only one defining equation.

A directed graph (the *causal graph*) is derived from the set of equations as follows: there is an edge from a to b if and only if a occurs on the right hand side of an equation where b is on the left hand side, i.e. a influences b. We impose that the causal graph does not contain directed cycles.

Several approaches [6; 8] exist for introducing causal information. Our causal graph is similar to temporal causal graphs, derived from *bond graphs*, in [13] except that temporal *integrating* edges are not made explicit, so that feedback loops do not appear.

4.1 Causal-directed Constraint Propagation

The causal semantics for QDEs induces a partial "causal" order on the set of model variables, and also on the set of constraints. Here we show how to compute this order, and how it can be used to efficiently reason on the model using a backtrack-free constraint propagator.

Let C be the set of ternary constraints (with causal meaning) of the causal QDE model; as mentioned above, complex expressions are easily reduced to a set of ternary constraints. Let $c_i : x \Longleftarrow y \; OP \; z$ be the generic constraint, where OP denotes a qualitative operator. Assign a degree to each $c_i \in C$ as follows:

$$deg(c_i) = 1 + max(deg(c_j)) \qquad \forall c_j \in C_i$$

where C_i is the set of constraints in which y or z appear on the left side. Then, sort constraints in C in "causal order", i.e. non-decreasing order of $deg(c_i)$; let C_{causal} be this sorted list of constraints.

---

[3] Moreover, $O(|C| \cdot \log |C|)$ steps are required for sorting nodes in C; in all practical cases, $\log |C| << |E|$, so this initial cost is negligible.

[4] Because we can only delete nodes from one side of the bipartite graph (the set C of shared variables), not from both sides. In other words, we are interested in finding a (minimal) subset of the shared variables, but we are not interested in identifying equation nodes which can remove cycles.

We obtain a tree-structured network whose leaves are all the possible "primary changes" of our causal model (i.e. state variables and component mode parameters). Such primary changes will be used as a starting point for intra-state propagation, which computes instantaneous side effects of such primary changes; a set of primary changes is computed by inter-state reasoning.

If constraints are evaluated according to the C_{causal} order, the model can be processed (finding a consistent state or proving the inconsistency of the CSP) without backtracking, then in $O(|C_{causal}|)$ (linear time in the number of constraints). In the following we further elaborate on this idea.

4.2 Causal Simulation Algorithm

Let S_i be the current qualitative state. The core of the simulation algorithm is to compute the set of successor states of S_i. Let S_{i+1} be the generic element of this set.

Let V_D be the subset of variables for which a qualitative derivative is explicitly represented in the model.

The CD relation (continuity and derivative constraints) shown in table 1, relates the values of (v, v') in state S_i with the value of v in the successor state S_{i+1}. In other words, it states that changes of continuous functions are driven by their derivatives (if they exist and are qualitatively known).

v_i	v'_i	v_{i+1}
-	-	-
-	0	-
-	+	-, 0
0	-	-
0	0	0
0	+	+
+	-	0, +
+	0	+
+	+	+

Table 1: The CD relation

The algorithm for computing the successor state of a qualitative state is given below.

1. Consider all "primary changes" given by each pair of variables (v, v'), i.e. changes imposed to system variables by their respective derivatives (explicitly modeled).
2. Generate all possible *partial* states for each primary change, and then compute the cartesian product of these partial solutions. Then the set $PARTIAL_i$ of all partial successors of S_i is given by:
$$PARTIAL_i = \{S_{i+1} | v_{i+1} \in CD(v_i, v'_i)\ \forall v \in V_D\}$$
3. Let $c_i : x \longleftarrow y\ OP\ z$ be the generic constraint. Construct a list of constraints C_{list} containing all constraints for which a right-hand variable is a primary change and its value has really changed with respect to the previous state:
$$C_{list} = \{(c_i : x \longleftarrow y\ OP\ z) \in C \text{ s.t. } (y \in V_D \wedge y_{i+1} \neq y_i) \vee (z \in V_D \wedge z_{i+1} \neq z_i)\}$$
Constraints in the list are considered in "causal order", i.e. according to the C_{causal} partial order defined above. This keeps time complexity linear, no sort procedure is needed.
4. While C_{list} is not empty do the following:
   - Extract the first constraint in C_{list}, let it be $c_i : x \longleftarrow y\ OP\ z$.
   - If variables y and z have not yet been assigned a value, we let $y_{i+1} = y_i, z_{i+1} = z_i$, i.e. retain the same values they have in the previous state S_i. This corresponds to assuming no change for which there is no cause.
   - Evaluate the constraint c_i, which is done very efficiently given that variables y and z are now necessarily already instantiated.
   - If c_i is satisfiable, use the derivative of x in state S_i (i.e. x'_i) to filter admissible values for x_{i+1} (those satisfying the c_i constraint).
   $$x'_i = 0 \rightarrow x_{i+1} = x_i$$
   $$x'_i = + \rightarrow x_{i+1} \geq x_i$$
   $$x'_i = - \rightarrow x_{i+1} \leq x_i$$
   - If the derivative of x is not explicitly modeled as a variable, then try to estimate its value using y' and z' (e.g. using the rule for the derivative of a sum).
   - If $x_{i+1} \neq x_i$ then add to C_{list} constraints in which x appears in the rhs.
   - If more than one value is admissible for x_{i+1} then select non deterministically a single value[5].

Notice that the complexity of generating a successor state is $O(|C|)$; no backtracking is required for constraint evaluation, i.e. no constraint is ever evaluated twice. Moreover, using derivative values allows an efficient and correct pruning of the search space according to continuity constraints.

4.3 Diagnosis

Causal simulation is applied to diagnosis of dynamic systems as follows [14]:

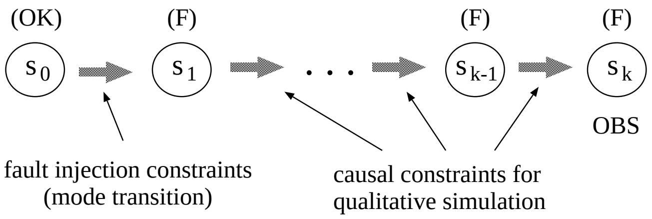


Figure 1: Diagnosis using causal constraints

[5] Different selections lead to different successor states.

The initial (qualitative) state S_0 is a state with no deviations, and consistent with the correct fault modes for all components.

State S_1 is obtained from S_0 setting as primary changes the values for component mode parameters for a specific fault F_i, and computing side effects using the causal structure, but relaxing the CD continuity constraints (we intend that F_i occurs abruptly, then introducing a discontinuity).

Subsequent states are computed using causal simulation as described in the previous section searching a state matching observations OBS. If such a state is found, then fault F_i is a candidate diagnosis. The search space is pruned when a state with detectable deviations inconsistent with OBS is reached.

4.4 Related Work

As mentioned before, the modeling approach is similar to temporal causal graphs in [13]. Algorithms in that approach are efficient because they do not reason on complete states of the system, but just on chaining deviations caused by each other (within a single state or across states). However, the approach in [13] does not deal with interaction of causes, which can give rise to ambiguous results: e.g. a cause for a deviating negatively and one for it deviating positively.

5 Applications and Experimental Results

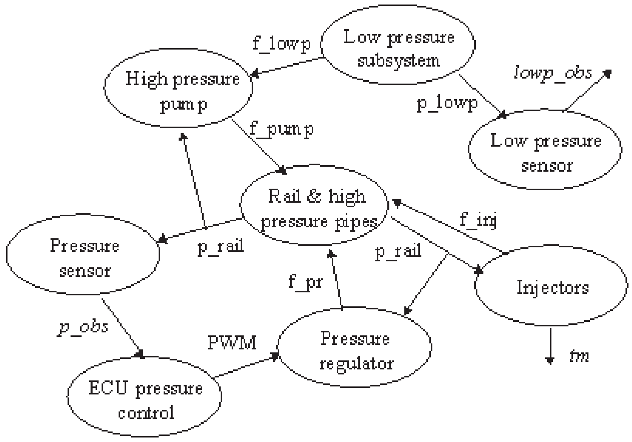
As a case study for the modeling and constraint processing methodologies, we present a continuous controlled dynamic system, one of the guiding applications within VMBD, a Brite-Euram project for the application of model-based diagnostic techniques in the automotive domain.

The Common Rail is a fuel injection system for diesel engines [15] designed in order to be able to control the injection pressure, as well as injection amount and timing, to allows for better engine performance and lower noise and emissions. To this end, pressurized fuel is stored in a container, the rail and its pressure is controlled by an Electronic Control Unit (ECU) through a pressure regulator (figure 2; see also [3] for more details).

Qualitative deviations have been used for modeling this system especially because the fuel pressure in the rail is rapidly varying, according to the position of the accelerator pedal and a number of other inputs, therefore a *normal range* of values cannot be given. The model used for our experiments is shown in the block diagram in figure 3. Variable names on the arcs correspond to pairs of interface variables of components, which are imposed to be equal by the connection. Observable variables are in italic.

The qualitative deviations model for the general behavior of the example system is shown in figure 5.[6]

[6]We omit the sign operator [.] on all qualitative variables for brevity.



– f_{lowp}: flow from low pressure system;
– p_{lowp}: pressure in low pressure system;
– $lowp_{obs}$: sensor for low pressure system;
– f_{pump}: outflow of high pressure pump;
– p_{rail}: pressure in the rail;
– p_{obs}: pressure sensor in the rail;
– pwm: command to pressure regulator;
– f_{pr}: flow through pressure regulator;
– f_{inj}: flow through the injectors;
– tm: torque measurement.

Figure 3: Block diagram of the Common Rail and meaning of interface variables.

In addition to the model of general behavior, we considered 13 fault modes for the components; each fault mode assigns a value to the component parameters. For example, the "ok" mode for the rail component assigns $\Delta f_{leak} = 0$ (i.e., there are no leaks), while the fault mode "leaking" assigns $\Delta f_{leak} = +$.

Note that only 7 of the 24 variables of the model appear in more than one equation. Thus, the greedy heuristic algorithm works on a bipartite graph with only 7 variable nodes (fig. 5) and computes a cutset which is of minimal cardinality in this case, containing only the variable Δp_{rail}.

In solving the example CSP for diagnostic purposes (i.e. checking consistency with the 13 considered fault modes), The results with and without applying the cutset heuristic are 512 and 2954 variable instantiations during the labeling phase, respectively (i.e. 83% reduction of search when the heuristic decomposition approach is used), with a similar ratio for execution times; then the cost of the heuristic algorithm is negligible with respect to the overall execution time.

As regards the causal approach, figure 6 contains the causal graph of the model augmented with temporal edges (shown as dotted edges).

The results with and without applying the causal approach are 5669 and 28802 total states generated for checking all the 27 possible combinations of observable values, respectively, i.e. there is a (more than) 80% reduction of search when the causal approach is used. In

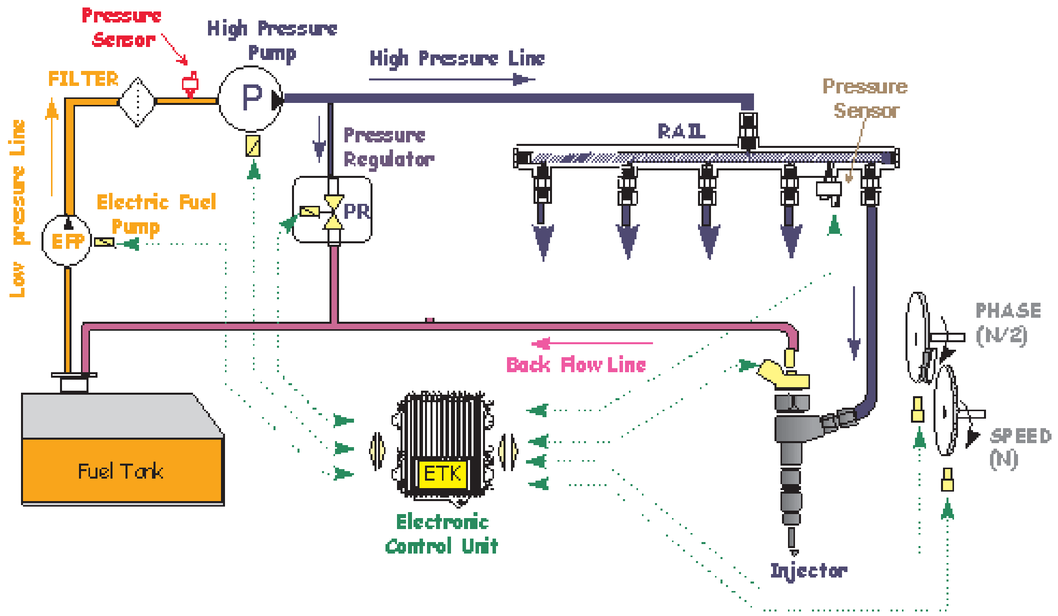


Figure 2: The Common Rail fuel injection system

High pressure pump	$\Delta f_{pump} \Leftarrow= c_p \otimes \Delta f_{lowp} \oplus \Delta c_p \ominus \Delta p_{rail}$	(4)
High pressure pipes and rail	$\frac{d}{dt}\Delta p_{rail} \Leftarrow= \Delta f_{pump} \ominus \Delta f_{pr} \ominus \Delta f_{inj} \ominus \Delta f_{leak}$	(5)
Pressure regulator	$\Delta f_{pr} \Leftarrow= open_{pr} \otimes \Delta p_{rail} \oplus \Delta open_{pr}$	(6)
	$\Delta open_{pr} \Leftarrow= c_{reg} \ominus \Delta pwm \otimes k_{reg}$	(7)
Pressure sensor	$\Delta p_{obs} \Leftarrow= \Delta k_s \oplus \Delta p_{rail} \oplus \Delta c_s$	(8)
Injectors	$\Delta f_{inj} \Leftarrow= i \otimes \Delta p_{rail} \oplus \Delta i \oplus \Delta f_{inj\_rec}$	(9)
Low pressure system	$\Delta f_{lowp} \Leftarrow= \Delta p_{lowp}$	(10)
Electronic Control Unit	$\frac{d}{dt}\Delta pwm \Leftarrow= c_{ecu} - \Delta p_{obs}$	(11)

Figure 4: The qualitative deviations model for the general behavior of the Common Rail

other words, several solutions which do not respect the causal semantics of the model are pruned. Moreover, there is an even better ratio for execution times, since partial states are extended to complete solutions without backtracking using causal-directed constraint propagation, i.e. improved intra-state reasoning, rather than a standard constraint propagation algorithm.

The constraint-based models and algorithms described in this paper have been initially implemented using a $CLP(FD)$ language (ECL^iPS^e) for rapid prototyping, and then using C++ for our running system.

6 Conclusions

From the modeling point of view, the layered modeling approach (L_{QDE} for model specification and a constraint language for model implementation) proves to be essential even on simple case studies in order to develop a model with limited effort and automatically derive an equivalent CSP and associated heuristics. Maintaining model specification and implementation separate allows better model reusability from the user perspective and easier (i.e. cheaper) software evolution from the software developer point of view.

Our experiments on the Common Rail and other real-life models show that the greedy algorithm for cycle cutset reduction gives good results on the class of CSPs considered here (those derived from L_{QDE}) on practical applications such as consistency-based diagnosis: labeling is reduced on the average of 80% when looking for all solutions of the CSP (which is often required, e.g. when performing simulation). Additional investigation of the cycle cutset problem on bipartite graphs with a preferred side is our current work.

Moreover, we have shown how structuring the model can provide efficient ways of reasoning on physical systems; in particular, a causal structure allows reasoning

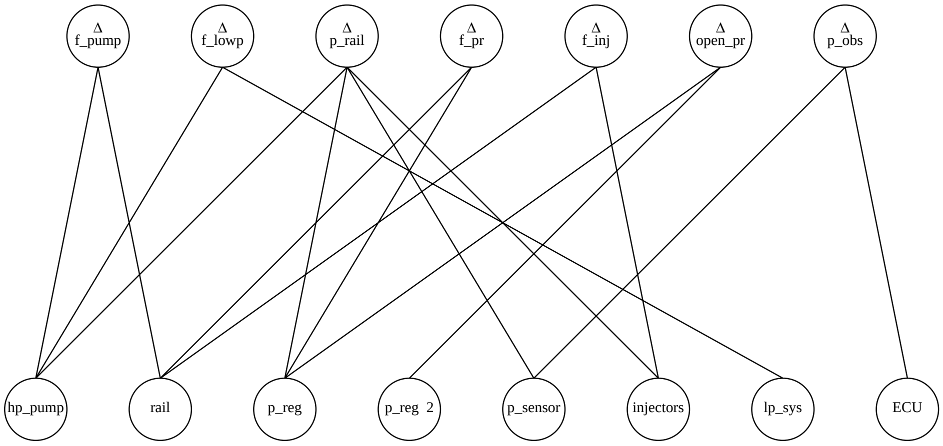


Figure 5: Relation between variables and equations as a bipartite graph.

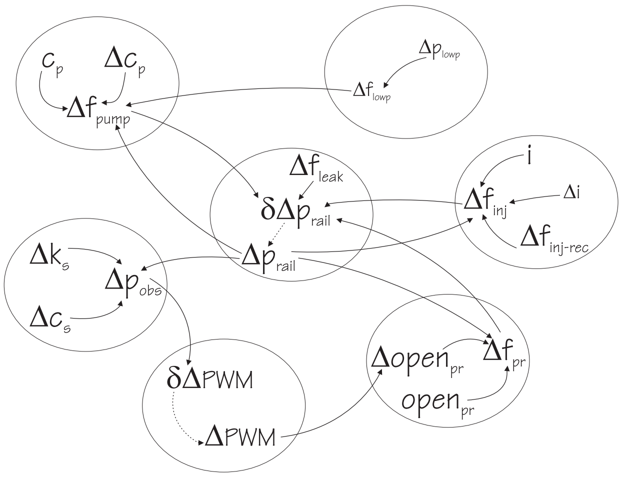


Figure 6: Causal graph for the Common Rail model

efficiently on the dynamics of the system, yet maintaining the ability of dealing with interaction of causes and with the ambiguities associated with qualitative abstractions.

Similar results have been obtained in a component-based model decomposition approach for qualitative simulation, which has been proposed in [4] to cope with complexity and scalability issues in QSIM-like systems.

A different approach for solving CSPs derived from systems of equations using a bipartite graphs decomposition was presented in [2]. However, their work (which combines numerical solvers with backtracking). focuses on solving *continuous* CSPs derived from systems of (non linear) equations, thus targeting a different domain and taking as input a set of numeric equations instead of constraints on a qualitative (i.e. finite) domain.

7 Acknowledgements

We would like to thank Marco Locatelli for useful discussions on graph problems and heuristics. Wa also thank Luca Console, Fulvio Cascio and other VMBD partners for providing real case studies and suggestions, and the anonymous reviewers for their helpful comments and remarks.

References

[1] Ann Becker and Dan Geiger. Approximation algorithms for the loop cutset problem. *Uncertainty in Artificial Intelligence*, 1994.

[2] C. Bliek, B. Neveu, and G. Trombettoni. Using graph decomposition for solving continuous CSPs. In *Proc. 4th Internationa Conference on Principles and Practice of Constraint Programming*, Pisa (Italy), 1998. Lecture Notes in Computer Science 1520.

[3] F. Cascio, L. Console, M. Guagliumi, M. Osella, A. Panati, S. Sottano, and D. Theseider Dupré. Generating on-board diagnostics of dynamic automotive systems based on qualitative models. *AI Communications*, 12(1,2):33–43, June 1999. Special issue on Model-Based Reasoning.

[4] D.J. Clancy and B.J. Kuipers. Qualitative simulation as a temporally extended constraint satisfaction problem. In *Proc. AAAI 98*, 1998.

[5] A. Davenport and E. Tsang. An empirical investigation into the exceptionally hard problems. *Proc. Workshop on Constraint-based Reasoning*, pages 46–53, 1995.

[6] Johan de Kleer and John Seely Brown. Qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984. Also in *Readings in Knowledge Representation*, Brachman and Levesque, editors, Morgan Kaufmann, 1985, pages 88-126.

[7] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[8] Yumi Iwasaki and Herbert A. Simon. Causality in device behavior. *Artificial Intelligence*, 29(1):3–32, 1986.

[9] B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.

[10] V. Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[11] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In C.S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1985–1991, 1995.

[12] A. Malik and P. Struss. Diagnosis of dynamic systems does not necessarily require simulation. *Proc. 7th Int. Workshop on Principles of Diagnosis*, 1996.

[13] P.J. Mosterman and G. Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Trans. on Systems, Man and Cybernetics*, 29(6):554–565, 1999.

[14] A. Panati and D. Theseider Dupré. State-based vs simulation-based diagnosis of dynamic systems. In *Proc. European Conference on Artificial Intelligence*, Berlin (Germany), August 2000.

[15] Gerhard Stumpp and Mario Ricco. Common rail - an attractive fuel injection system for passenger car DI diesel engines. *J. Society of Automotive Engineers*, 1996.