On the use of qualitative deviation models for diagnosis

Franz Wotawa
Graz University of Technology, Institute for Software Technology,
Inffeldgasse 16b/2, 8010 Graz, Austria
e-mail: wotawa@ist.tugraz.at

Abstract

Detecting and even locating faults in systems is an important but also very much resource consuming task, which is especially true for finding and fixing bugs in programs. In literature someone finds different approaches for supporting the fault localization task for programs including statistical methods like spectrum-based fault localization, methods based on control and data dependences like slicing, and even model-based diagnosis relying on a logical or constraint representation of a program for computing diagnosis candidates. One issue that hampers the use of model-based diagnosis for debugging is its computational requirements especially when relying on a more or less one-to-one representation of the underlying source code. In order to decrease computational requirements abstract models have to be used. In this paper, we discuss the use of deviation models and provide a framework for comparing such models making use of an abstraction function. First experimental results indicate that some abstract models behave similar to concrete models for diagnosis but come with a much lower computational footprint enabling their use in practice even for larger programs.

Introduction

When a program exhibits an unexpected behavior, the identification of its corresponding root cause can be a very laborious and time consuming task. This is due to several reasons including: (1) The interactions and data communicated with the program leading to the unexpected also contains a lot of information that is not needed for bug localization. (2) The chain of computations from a root cause to its effect, i.e., the failure, might be very long. (3) And it might also be very difficult to state an expected value. The latter occurs for example in cases of complicated computations where we know that the value should be within a range but nothing more.

Let us illustrate this third case using an example from the Spreadsheet domain taken from (Hofer et al. 2015). The spreadsheet given in Figure 1 computes the cardiac index of a person using the diastolic and systolic volume, the heart rate, and the body surface area as inputs. For illustrative purposes we added the cell's formulae directly beside the

	A	B	C	D
1	**Cardiogenic Shock Estimator**			
2	**End Diastolic Volume**	120 mL		
3	**End Systolic Volume**	60 mL		
4	**Heart Rate**	72 bpm		
5	**Body Surface Area**	2 m2	**Formulas**	**Fault: "/" instead of "-"**
6	Stroke Volume	2 mL	=B2/B3	
7	Cardiac Output	144 mL/min	=B6*B4	
8	**Cardiac Index**	72 mL/min/m2	=B7/B5	
9				
10				

Figure 1: Spreadsheet "Cardiogenic Shock Estimator"

spreadsheet where we introduced a fault in cell C6, which should be B2-B3. Because of this bug the resulting cardiac index is 72 instead of 2,160. Someone experienced in estimating the cardiac index may easily detect this far too low value but may not be able to specify the real expected output value. Hence, means for specifying deviations from expected values in a qualitative way would be very valuable for automated debugging.

In this paper, we follow this idea of using qualitative representations for fault localization instead of real values. Someone should also bear in mind that using values from domains like integer or even reals in models for diagnosis might not be feasible. For example, (Hofer et al. 2015) reported that computing single faults took 25.1 seconds even for smaller spreadsheets having up to 70 non-empty cells. Hence, for larger spreadsheets representations used for diagnosis may hardly use quantitative models. Instead qualitative models that are able to handle deviations, i.e., differences between the expected and the observed value should be used providing that such models come with a smaller computational footprint.

In the following, we define diagnosis based on constraint solving and introduce different models including a value-based variant considering integer values, a dependency model capturing information about the correctness (or incorrectness) of certain values, and a model where we are able to state whether a value is smaller, equivalent, or larger than expected. The latter model we refer as comparison model. In addition to these models we discuss a framework where we are able to compare models with respect to diagnosis accuracy, which we define as the ability of a model for reducing the diagnosis search space. Moreover, we introduce a definition of abstraction that allows for compar-

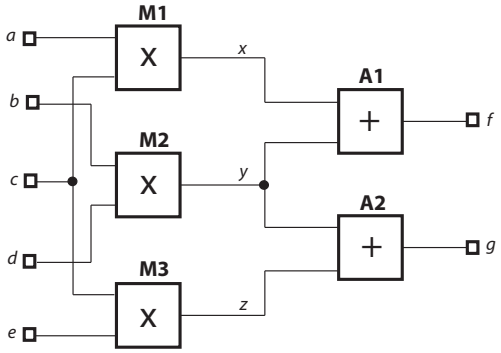

Figure 2: The d74 circuit

ing models. Afterwards, we present the first experimental results when using the different models for diagnosis. The results indicate that the comparison model, which we later plan to use for debugging spreadsheets, has a good running time performance and good diagnosis capabilities.

Basic definitions

In the following we introduce the basic definitions where we rely on the classical definitions of model-based diagnosis (Reiter 1987; de Kleer and Williams 1987) but adapt them to fit to the underlying constraint-based representation of models. For illustration purposes we make use of the famous d74 circuit example depicted in Figure 2.

We first start defining constraints systems and their corresponding constraint satisfaction problem. We define a constraint system as a tuple $(VARS, DOM, CONS)$ where $VARS$ is a finite set of variables, DOM is a function mapping each variable to its domain comprising at least one element, and $CONS$ a finite set of constraints. Without restricting generality we define a constraint c as a pair $((v_1, \ldots, v_k), tl)$ where $(v_1, \ldots, v_k)$ is a tuple of variables from $VARS$, and tl a set of tuples $(x_1, \ldots, x_k)$ of values where for each $i \in \{1, \ldots, k\}$: $x_i \in DOM(v_i)$. The set of tuples tl represents allowed variable value combination. For simplicity, we assume a function $scope(c)$ for a constraint c returning the tuple $(v_1, \ldots, v_k)$, and a similar function $tl(c)$ returning the set of tuples tl of c.

For example, the constraint representation M_{VB} of the d74 circuit to be used for diagnosis purposes has the following variables:

$$VARS = \left\{ \begin{array}{c} a, b, c, d, e, f, g, x, y, z, \\ ab\_m1, ab\_m2, ab\_m3, ab\_a1, ab\_a2 \end{array} \right\}$$

In this set the variables ab_X represent the fault status of a component X, i.e., ab_X is true if component X is said to be abnormal and false, otherwise. The domain for the variables representing connection are integers: $\forall w \in \{a, b, c, d, e, f, g, x, y, z\} : DOM(w) = \mathbb{W}$ and for the fault status we use Boolean values, i.e.: $\forall w \in \{ab\_m1, ab\_m2, ab\_m3, ab\_a1, ab\_a2\} : DOM(w) = \{T, F\}$.

For each component, we have to introduce a constraint. For the multiplication components M1, M2, M3, we have the following constraints:

$((ab\_M_{1|2|3}, a, c, x), \{(F, u, v, u \cdot b) | u, v, w \in \mathbb{W}\} \cup \{(T, u, v, w) | u, v, w \in \mathbb{W}\})$

For the adders A1 and A2 we have similar constraints:

$((ab\_A_{1|2}, a, c, x), \{(F, u, v, u + b) | u, v, w \in \mathbb{W}\} \cup \{(T, u, v, w) | u, v, w \in \mathbb{W}\})$

Note that both types of constraints indicate that in case of a fault all possible value combinations may be observed whereas for the correct behavior the respective restricting relationship among the connections have to be fulfilled.

In order to define a constraint satisfaction problem, we first introduce the concept of value assignments for variables. Given a constraint system $(VARS, DOM, CONS)$, and variable $v \in VARS$, then $v = x$ with $x \in DOM(v)$ is a single assignment of a value x to the variable v. We further say that a set of single assignments where there at the maximum one single assignment for a variable as value assignment. A constraint c with scope $(v_1, \ldots, v_k)$ fulfills a value assignment $\{\ldots, v_1 = x_1, \ldots, v_k = x_k, \ldots\}$, if there exists a tuple $(x_1, \ldots, x_k)$ in $tl(c)$. Otherwise, we say that such a value assignment contradicts the constraint.

A constraint satisfaction problem for a given constraint system is the question whether a value assignment exists that fulfills all given constraints. If there is such a value assignment, then the constraint satisfaction problem is said to be itself fulfilled.

For example, the value assignment $a = 2, b = 2, c = 3, d = 3, e = 2, x = 6, y = 6, z = 6, f = 12, g = 12, ab\_M1 = F, ab\_M2 = F, ab\_M3 = F, ab\_A1 = F, ab\_A1 = F$ fulfills all constraints of the d74 circuit constraint system, whereas $a = 2, b = 2, c = 3, d = 3, e = 2, x = 6, y = 6, z = 6, f = 10, g = 12, ab\_M1 = F, ab\_M2 = F, ab\_M3 = F, ab\_A1 = F, ab\_A1 = F$ does not. Hence, the d74 constraint satisfaction system can be fulfilled.

Solving a constraint satisfaction problem is basically a search procedure for a value assignment that fulfills all constraints. This search for constraint systems having only constraints with finite tuple lists is well known to be exponential and its corresponding problem is well known to be NP-complete. For details about algorithms and heuristics we refer to interested reader to (Dechter 2003).

In the following we discuss the diagnosis problem and show how constraint solving can be used to solve the classical diagnosis problem. According to (Reiter 1987) a diagnosis problem is a tuple $(COMP, SD, OBS)$ where $COMP$ is a set of components, SD a logical sentence describing the behavior of the system, i.e., the system description, and OBS a set of observations. In out constraint based representation of the diagnosis problem, we assume a constraint representation of the system and additional constraints specifying the observations. The constraint representation of a diagnosis problem (or the diagnosis problem for short) is a tuple $(VARS, DOM, CONS \cup COBS)$ where $(VARS, DOM, CONS)$ is a constraint representation of a system comprising variables ab_C for every component C of the system, and $COBS$ is the constraint representation of

all observations OBS.

For our d74 circuit, the constraint representation M_{VB} together with the constraint representation $COBS = \{(a, b, c, d, e, f, g), \{(2, 2, 3, 3, 2, 10, 12)\}\}$ specifying observations forms a diagnosis problem.

The results of a diagnosis problem, i.e., the diagnoses, are subsets of the set of components $COMP$. We obtain these subsets from the solutions of the corresponding constraint problem via taking one value assignment that is a solution, and putting all components C for which the corresponding variable ab_C is set to T into a set, i.e., if s is a solution of the constraint representation of a diagnosis problem, then its corresponding diagnosis is $\Delta_s = \{C|ab\_C = T \in s\}$. When computing all solutions from the constraint representation, we obtain all possible diagnosis. As usual, we define a diagnosis to be minimal if there exists no subset, which is itself a diagnosis. Of course, we are always interested in only computing minimal diagnosis in the most efficient way.

In the following we discuss briefly a diagnosis algorithm that computes minimal diagnosis of increasing size. This can be achieved via restricting the number of ab_C variables to be set to true using constraints. In this way we are able to compute diagnoses up to a pre-specified size. The necessary additional constraints are added during diagnosis computation in diagnosis algorithms like ConDiag. (Nica and Wotawa 2012) introduced the ConDiag algorithm that computes minimal diagnoses up to a predefined size using a constraint representation of the diagnosis problem. (Nica et al. 2013) compared ConDiag with other diagnosis algorithms showing a good overall runtime. In order to be self-contained we briefly discuss the ConDiag algorithm, which is given in Algorithm 1.

Algorithm 1 ConDiag$((VARS, DOM, CONS \cup COBS), COMP, n)$

Input: A constraint model $(VARS, DOM, CONS \cup COBS)$ of a system having components $COMP$ and the desired diagnosis cardinality n

Output: All minimal diagnoses up to the predefined cardinality n

1: Let DS be $\{\}$
2: Let M be $CONS \cup COBS$
3: **for** $i = 0$ **to** n **do**
4: $CM = M \cup \{|\{ab_C | C \in COMP \wedge ab_C = T\}| = i\}$
5: $S = \mathcal{P}(\mathbf{CSolver}(VARS, DOM, CM))$
6: **if** i is 0 **and** S is $\{\{\}\}$ **then**
7: **return** S
8: **end if**
9: Let DS be $DS \cup S$.
10: $M = M \cup \{\neg(\mathcal{C}(S))\}$
11: **end for**
12: **return** DS

The ConDiag algorithm computes diagnoses starting with cardinality 0 to the predefined size n that has to be provided as parameter. In each step, we are searching for solutions that have exactly a size of i (Step 4). All these solutions are added to the set of solutions in Step 9. In order to prevent the computation of non-minimal diagnoses additional constraints saying that we are not interested in superset diagnoses are added (see Step 10). ConDiag returns all minimal diagnoses up to size n and the empty diagnosis if the system works as expected.

When using ConDiag on the M_{VB} model of the d74 circuit, we obtain two single fault diagnoses $\{M1\}$ and $\{A1\}$ and two double fault diagnoses $\{M2, A2\}$ and $\{M2, M3\}$.

Qualitative models for diagnosis

In the previous section we illustrated the basic definitions using the quantitative model of the d74 circuit M_{VB} based on constraints over integer values. In order to speed up the diagnosis computation especially for large systems comprising thousands of components, we have to use appropriate abstractions. In software debugging data and control dependencies can be used for this purpose like in program slicing (Weiser 1982). Based on static slices (Friedrich, Stumptner, and Wotawa 1999) developed a model that could be easily integrated into model-based diagnosis, and which was later proved to be equivalent to program slicing (Wotawa 2002).

All these abstractions are not abstractions in the sense of homomorphic functions applied to a quantitative space in order to obtain a qualitative representation. Instead these abstractions introduce the idea of classifying variables or values of connection between components to be either correct or incorrect in a particular diagnosis problem. Hence, instead of using particular values, e.g., from the integer domain, the dependency-based models use classifications, which are based on deviations between the actual and the expected behavior. For a very detailed analysis of such deviation models in the context of diagnosis we recommend to consult (Struss 2004).

In the following, we discuss two dependency-based models, and show how they can be represented using constraints. We start with the dependency-based model of (Friedrich, Stumptner, and Wotawa 1999) we call M_D^{orig}. There the authors introduce a model of a component C having m inputs and one output. This models states that the output can only be correct, if the component is correct and all inputs have a correct value, i.e.: $\neg ab_C \rightarrow (\bigwedge_i^m in_i = ok \rightarrow out = ok)$. In the case the component is correct, but one input is not, the output may be correct or not correct. In the case of a faulty component, there is no way of determining the correctness status of the output from any correctness information of the input. When taking this thoughts into consideration, then we obtain the following table constraint for a component C with $n = 2$ inputs.

ab_C	in_1	in_2	out
F	ok	ok	ok
F	$\neg ok$	ok	ok
F	ok	$\neg ok$	ok
F	$\neg ok$	$\neg ok$	ok
F	$\neg ok$	ok	$\neg ok$
F	ok	$\neg ok$	$\neg ok$
F	$\neg ok$	$\neg ok$	$\neg ok$
T	.	.	.

In this table and also the following ones a $'.'$ stands for any possible value. Hence a row with $'.'$ represents multiple rows when changing the placeholder $'.'$ with possible values.

If we use this component model for all components of the d74 circuit from Figure 2 and further set the observations as follows $COBS = \{(a, b, c, d, e, f, g), \{(ok, ok, ok, ok, ok, \neg ok, ok\}\}$, which represent the observations used for diagnosing the d74 in the previous section, then we obtain the following three minimal diagnoses: $\{M1\}$, $\{M2\}$, and $\{A1\}$. When comparing this result with the previous one we see that when considering integers, we have two diagnoses which are supersets of $\{M2\}$. Hence, when considering dependencies only we lose some information, which leads to a larger search space of potential diagnoses including all their supersets.

The underlying reason for this decrease in precision of diagnosis is that the model does not consider the case where a faulty value does not propagate through the whole system. For example, if we consider a logical and gate and we know that one input is false, then the other input does no longer determine the value of the output. Hence, any faulty value occurring will never be visible on sider of the output. This behavior is named coincidental correctness in software debugging and always influences the fault localization capabilities.

In order to handle coincidental correctness using a dependency-based model, we have to distinguish two cases: (1)There are component where coincidental correctness may occur, e.g., for logical gates. (2) There are other cases, where coincidental correctness is at least very unlikely, e.g., when considering a function for adding two integers. In the latter case, we can state that a correct output value for a working component also implies that all inputs are working, i.e., $\neg ab_C \rightarrow (\bigwedge_i^m in_i = ok \leftrightarrow out = ok)$. (Hofer and Wotawa 2014) introduced this improved model for debugging Spreadsheet programs handling coincidental correctness we call M_D^{CC}. The constraint representation of a two inputs component takes care of the bi-implication used in the component model, which is only allowed to be used if no coincidental correctness may occur.

ab_C	in_1	in_2	out
F	ok	ok	ok
F	$\neg ok$	ok	$\neg ok$
F	ok	$\neg ok$	$\neg ok$
F	$\neg ok$	$\neg ok$	$\neg ok$
T	.	.	.

When using the model M_D^{CC} for diagnosing the d74 circuit and the previously used set of observations, we obtain again two single fault diagnoses $\{M1\}$, $\{M2\}$ and also one double fault diagnosis $\{M1, A2\}$. The other double fault diagnosis $\{M2, M3\}$ is missing. The reason here is that this model is not able to handle the case that two faulty inputs may lead to a correct output, which might happen even for operations on integer domains. Adding the tuple $(F, \neg ok, \neg ok, ok)$ to the table solves this issue. In the following we refer to the model M_D^{CC} extended with the tuple as M_D.

The dependency-based models discussed can be seen as the most abstract form of deviation model only considering values to be either correct (i.e., ok) or incorrect (i.e. $\neg ok$). A less abstract model may allows to distinguish cases where a value is smaller, equivalent, or larger than an expected value. In the following we discuss such a model and introduce abstraction formally in order to allow comparing such models with others.

When dealing with comparisons like smaller $<$, equivalent $=$, or larger $>$, we have to introduce tabular constraints for the different operators. In case of multiplication and addition, the constraints are the same but for others like subtraction adaptation for capturing the different semantics are necessary. In the following table we summarize the constraints handling the behavior of addition and multiplication components. There we state that in case of equivalent inputs we also obtain an output value with no deviation. In case one input is smaller (or larger) and the other is equivalent, the output also is expected to be smaller (or larger respectively). In case we have one smaller and one larger input value, we cannot say anything about the output. Hence, in such a case all output values may occur. If the operator (or component) is said to be faulty, all combinations of values are possible.

ab_C	in_1	in_2	out
F	$=$	$=$	$=$
F	$<$	$=$	$<$
F	$=$	$<$	$<$
F	$<$	$<$	$<$
F	$>$	$=$	$>$
F	$=$	$>$	$>$
F	$>$	$>$	$>$
F	$<$	$>$	$=$
F	$<$	$>$	$<$
F	$<$	$>$	$>$
F	$>$	$<$	$=$
F	$>$	$<$	$<$
F	$>$	$<$	$>$
T	.	.	.

We call the comparison model based on such a table M_C.

Obviously, the deviation model based on comparison gives additional information for the diagnosis process. However, the question is whether there is an improvement of the accuracy of the obtained diagnosis.

Let us start with continuing the d74 example. From the value-based model we obtain the following observations:

a	b	c	d	e	f	g
$=$	$=$	$=$	$=$	$=$	$<$	$=$

This together with the model for the components $M1, M2, M3, A1, A2$ allows for computing again 2 single fault diagnoses $\{M1\}$ and $\{A1\}$, and the 2 double fault diagnoses $\{M2, M3\}$ and $\{M2, A2\}$. Hence, there is no improvement in accuracy for this example.

If we change the observations, i.e., assuming $g = 10$ and $f = 14$ the situation changes. The value-based model allows for computing no single fault but 8 double fault diagnoses: $\{M1, M2\}$, $\{M1, M3\}$, $\{M1, A2\}$, $\{M2, M3\}$,

$\{M2, A1\}$, $\{M2, A2\}$, $\{M3, A1\}$, and $\{A1, A2\}$. The same diagnoses can be obtained when using the more accurate deviation model and the observations:

a	b	c	d	e	f	g
$=$	$=$	$=$	$=$	$=$	$<$	$>$

In case of the both the original and the improved dependency-based model we obtain one single fault diagnosis $\{M2\}$, and 4 double fault diagnoses $\{M1, M3\}$, $\{M1, A2\}$, $\{M3, A1\}$, and $\{A1, A2\}$. Hence, we see that the more abstract deviation models lead to the computation of less accurate diagnoses in some cases. We depict the diagnosis search space that includes the minimal diagnoses as well as all of their supersets in Figure 3.

In the next section we introduce abstraction and diagnosis accuracy formally, and further more discuss their relationship in detail.

Domain abstraction

We start with defining abstraction formally. Given two constraint models for diagnosis $M_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $M_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. We say that M_1 is more abstract than M_2, i.e., $M_1 \prec M_2$, if there exists a function $h : DOM_2 \mapsto DOM_1$ that makes the constraints equivalent, i.e., $\forall c_1 \in CONS_1 \cup COBS_1$ and $c_2 \in CONS_2 \cup COBS_2$ having the same scope, $h(c_2) = c_1$.

For this definition of abstraction, we use the following definition of the application of a function f on a constraint $((v_1, \ldots, v_k), tl)$: $f(((v_1, \ldots, v_k), tl)) = ((v_1, \ldots, v_k), f(tl))$ where f is defined on tuple list as follows: $f(tl) = \{(f(x_1), \ldots, f(x_k)) | (x_1, \ldots, x_k) \in tl\}$.

From our running d74 example using the function h defined as $h(=) = ok$, $h(<) = \neg ok$, and $h(>) = \neg ok$ we can easily check that the improved dependency-based model considering coincidental correctness M_D is more abstract than the comparison-based model M_C, i.e., $M_D \prec M_C$. It is worth noting that there is no such function h for the original dependency-based model and the original model handling coincidental correctness.

From the definition of $\prec$ the definition of model equivalence follows immediately. Let M_1 and M_2 are constraint models for diagnosis. M_1 and M_2 are equivalent, i.e., $M_1 \equiv M_2$, if and only if $M_1 \prec M_2$ and $M_2 \prec M_1$. Obviously, if the same function h can be used to show that $M_1 \prec M_2$ and $M_2 \prec M_1$, then h has to be a bijective function.

In the following we define diagnosis accuracy. For this purpose, we bear in mind that in case of pure consistency-based diagnosis, all supersets of minimal diagnoses are also diagnoses. This is ensured in all cases where we only be aware of the behavior of a correct component but do not know a component's incorrect behavior. If we use models of the faulty behavior minimal diagnoses are not characterizing all possible diagnoses anymore. See (de Kleer, Mackworth, and Reiter 1992) for a detailed discussion on this topic. In this paper we assume models that capture the correct behavior only. Hence, we know that all possible diagnosis can be characterized as follows. Let Δ-MIN be the set of all minimal diagnoses obtained from a constraint model $M = (VARS, DOM, CONS \cup COBS)$. The set of all diagnoses comprises the minimal diagnoses and all of their supersets, i.e., $\Delta\text{-SET}^M = \{\Delta | \exists \Delta' \in \Delta\text{-MIN} : \Delta \supseteq \Delta'\}$. Thus $\Delta\text{-SET}^M$ spans the whole search space of diagnosis. Using this definition we are able to define accuracy as the ability of a model M to come up with the smallest possible set $\Delta\text{-SET}^M$. To compare two constrain models used for diagnosis, we only need to compare their search spaces.

Given two constraint models for diagnosis $M_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $M_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. We say that M_1 is less accurate than M_2, i.e., $M_1 \prec_A M_2$, iff $\Delta\text{-SET}^{M_1} \supset \Delta\text{-SET}^{M_2}$. M_1 is as accurate as M_2, i.e., $M_1 \eqsim M_2$, iff $\Delta\text{-SET}^{M_1} = \Delta\text{-SET}^{M_2}$. M_1 is less or equal accurate M_2, i.e., $M_1 \preceq_A M_2$, iff $M_1 \eqsim M_2$ or $M_1 \prec_A M_2$.

When considering the search spaces for diagnosing the d74 circuit given in Figure 3 we see that the model M_D is less accurate than M_C, i.e., $M_D \prec_A M_C$.

In the following theorem we manifest the relationship between abstraction and diagnosis accuracy.

Theorem 1. *Given two constraint models for diagnosis $M_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $M_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. If M_1 is more abstract than M_2, then M_1 is less or equal accurate than M_2, i.e., $M_1 \prec M_2 \rightarrow M_1 \preceq_A M_2$.*

Proof. To prove the theorem we first assume that we have two models M_1 and M_2 where $M_1 \prec M_2$. From this follows that there exists a function h, which maps the elements of DOM_2 to elements of DOM_1 such that the tuple sets for each component becomes equivalent using only elements from DOM_1. What we have to proof is that $\Delta\text{-SET}^{M_1} \supseteq \Delta\text{-SET}^{M_2}$, i.e., for all $\Delta \in \Delta\text{-SET}^{M_2}$ it follows that $\Delta \in \Delta\text{-SET}^{M_1}$. We prove this by contradiction. Let Δ be in$\Delta\text{-SET}^{M_2}$ but $\Delta \notin \Delta\text{-SET}^{M_1}$. Because $\Delta \in \Delta\text{-SET}^{M_2}$ we know that the constraint model $(VARS, DOM_2, CONS_2 \cup COBS_2 \cup ((\Delta), \{(T, \ldots, T)\}))$ is satisfiable. Because of the definition of $\prec$ we would get a corresponding constraint model $(VARS, DOM_1, CONS_1 \cup COBS_1 \cup ((\Delta), \{(T, \ldots, T)\}))$ when applying h to the constraints. Note that h has no effect on the constraint $((\Delta), \{(T, \ldots, T)\})$. But this constraint system has to be also satisfiable because of the construction of h. Hence, Δ is also element of $\Delta\text{-SET}^{M_1}$ contradiction our assumption, and the theorem hold. □

Note that a more abstract model does not cause less accurate diagnoses in all cases. For the d74 example, we saw that depending on the observations we obtain the same or a less accurate diagnosis for the more abstract model M_D when compared to M_C. It is also worth noting that the definition of more or less accurate can also be used independently from abstraction.

From Theorem 1 we obtain the following lemma, which states that equivalent models also have an equivalent diagnosis accuracy.

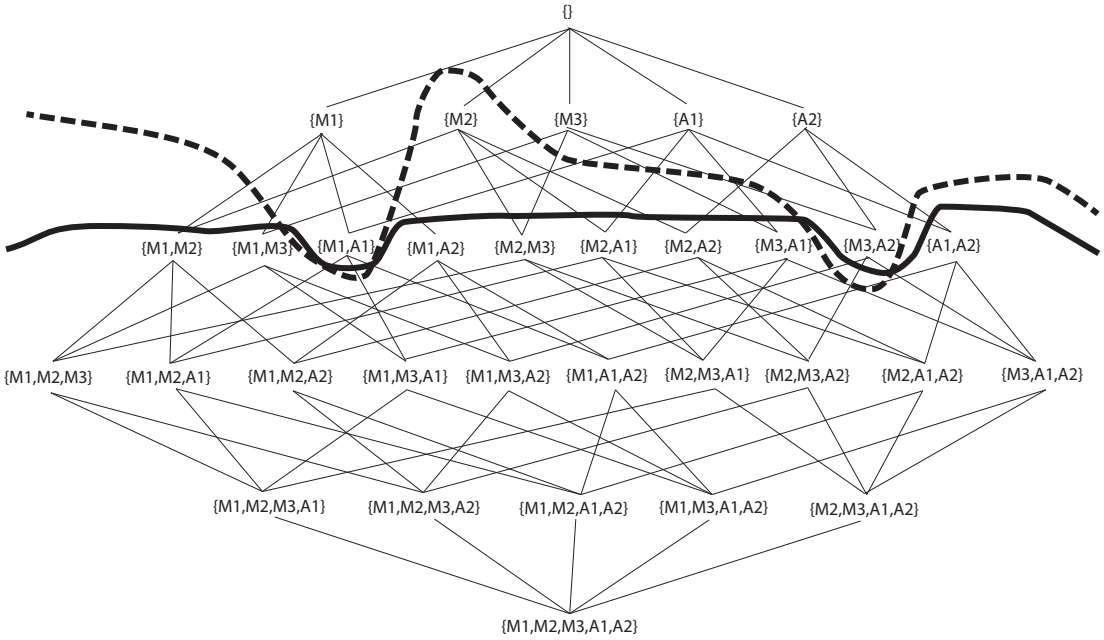

Figure 3: The diagnosis search space for the d74 circuit and assuming both outputs to behave incorrectly, i.e., $f = 10$ and $g = 14$. The dotted line shows the search space for the improved dependency-based model handling coincidental correctness whereas the solid line indicates the search space for both the value-based and the comparison-based model.

Lemma 1. *Let M_1 and M_2 be equivalent models, i.e., $M_1 \equiv M_2$, then M_1 and M_2 have the same diagnosis accuracy, i.e., $M_1 \equiv M_2 \rightarrow M_1 \eqsim M_2$.*

Proof. The lemma follows directly from Theorem 1 and the definition of equal diagnosis accuracy. □

Obviously, there might be cases where two models have the same accuracy but there is no mapping between model elements. Hence, we are not allowed to conclude model equivalence from equal diagnosis accuracy.

Experimental results

In order to motivate the use of qualitative models for diagnosis we carried out some experiments based on a parametrizable circuit comprising components for adding and multiplying integers. Our underlying research questions are: (1) whether the discussed qualitative models decrease the running time of diagnosis compared to a model based on integer values, and (2) whether the accuracy of diagnosis does not decrease substantially.

For generating a parametrizable circuit we implemented a circuit generator having 2 parameters: (1) the number of components directly connected to the inputs, and (2) the number of outputs. The generator constructs the circuit level by level, where in each level the number of components is reduced by 1. We stop at a level where the number of components is equivalent to the wanted number of outputs. We further assume that each component is either a component for adding two integers or multiplying two integers. The functionality of each component changes at every level. Components from level i are only connected to components from level $i + 1$ where two components of $i + 1$ share one output of a component of level i. For example, in Fig-

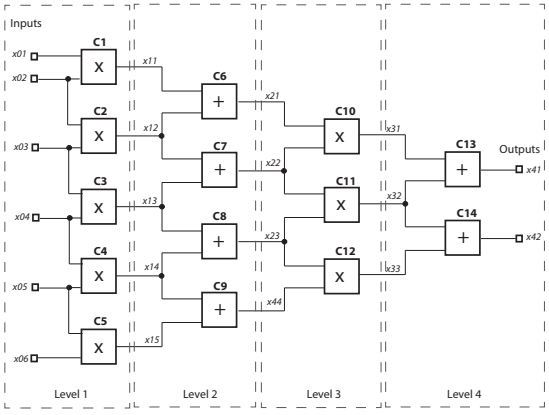

Figure 4: A generated circuit having 5 components directly connected to the inputs and 2 outputs.

ure 4 we depict the circuit, which can be generated using 5 components in level 1 and 2 outputs. Obviously, the number of inputs is always $n + 1$ if n is the number of components in the first level, and the number of wanted outputs k has to fulfill equation $n \geq k \geq 1$. For the experiment, we use the values 2 and 3 in an alternating way and computed the expected outputs when constructing the circuit. The implementation of the circuit generator return a value-based, an improved functional dependency model, and a comparison model of the circuit using the parameters. All the models can be executed using the Minion (Gent, Jefferson, and Miguel 2006) constraint solver. For the experiments we used the latest Minion Version 1.8 (available at `http://constraintmodelling.org/`).

In the experiment we generated 6 smaller circuits with $2 \ldots 7$ components in the first level and exactly two outputs. We named the circuit c22, c32, ..., c72. The purpose of this experiment was to compare the diagnosis results and the running time for computing all single fault diagnosis using our 3 models, i.e.: the value-based model M_{VB}, the improved functional dependency based one M_D, and the comparison model M_C. Besides the predefined inputs, which are either 2 or 3 for M_{VB} and ok for the other models, we assumed every output except the last one at the bottom of the circuit to be correct. For the last output we set its value to 0 (for M_{VB}), to $\neg ok$ for M_D, and to $<$ for M_C. In Table 1 we summarize the obtained results when running the search for single fault diagnoses on a MacBook Pro, 2,8 GHz Intel Core i7, 16 GB memory, and OS X version 10.11.3.

Note that for the value based model we used a integer domain ranging from -300 to 300 in order to compute the diagnoses. From Table 1 we see that even for small circuit there is a substantially larger running time when using M_{VB} compared to the other models. It is worth noting that with the given integer domain range the circuits c62 and c72 cannot be solved using M_{VB}. Moreover, the functional dependency model M_D produces many more diagnoses than both other models. The reason here is the introduction of a tuple in the constraint table that allows for masking faults in case of two wrong input values and the highly interconnected structure of the circuits. The best model in terms of running time and diagnosis accuracy is M_C. It is also interesting to see that for the slightly larger circuits computing diagnoses within 10 seconds was not possible when using M_{VB}. In order to complete the first experiments we further studied the influence of the used integer domain to the running time of diagnosis when using the value-based model. See Figure 5 for the results. When considering the logarithmic scale we see an exponential increase of running time when doubling the space of integers. Hence, for larger numbers diagnoses using M_{VB} becomes infeasible and that even for very small systems.

The results show that qualitative models for expressing the propagation of deviations from expected values are very valuable for diagnosis purposes. The introduced comparison model M_C provides a good diagnosis running time and accuracy especially when compared with the functional dependency model M_D.

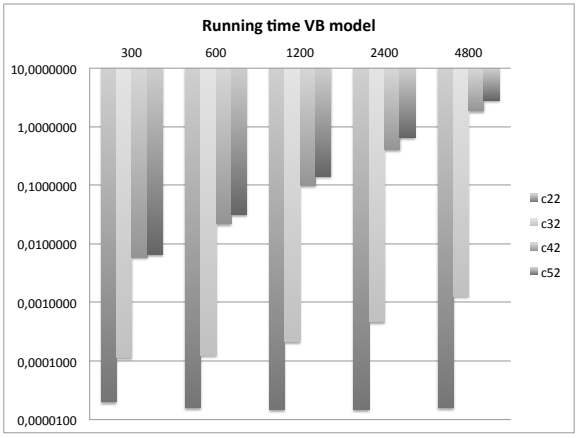

Figure 5: Minimum running time in seconds as a function of the size of the used integer domain.

Related research

The idea of using abstraction for diagnosis and in particular model-based diagnosis (Reiter 1987; de Kleer and Williams 1987) is not new. Initial work including (Mozetič 1991) and later (Autio and Reiter 1998) discussed the concept of structural abstraction, where sets of interconnected components are mapped to one component. The behavior of such a component is given using the sets of interconnected components. When using such an abstraction, we obtain a hierarchical model, where a component model in one level is given using the structure and behavior of the corresponding interconnected components. (Autio and Reiter 1998) discussed the resulting properties of such an approach in detail.

(Struss 1992) discussed modeling including abstraction and refinement in very much detail. (Sachenbacher and Struss 2003; 2005) introduced a different abstraction approach where quantitative domains are mapped to qualitative ones considering value boundaries influencing the behavior of the system. Such boundaries depend on the given diagnosis problem. Therefore, the authors suggested to use an automated abstraction approach for solving this issue.

In contrast, to these previous papers, we focus on deviation models for diagnosis and state a theory allowing to compare them using the introduced definition of abstraction, which is close to (Struss 1992). Although, we used example from classical hardware diagnosis to illustrate the concepts, we are driven by the idea of coming up with automated debuggers for programs. There qualitative representations seems to be very useful and appropriate.

It is also worth mentioning other work related to abstraction of programs. (Cousot and Cousot 1977) introduced the concept of program abstraction providing a theoretical framework. The focus of (Cousot and Cousot 1977) was on the execution part and there the consequences of introducing abstraction. In our work, the focus is on fault localization and deviation models.

Circuit		M_{VB}				M_D				M_C			
name	comps	singl. f.	min. T	avg. T	max T	singl. f.	min. T	avg. T	max T	singl. f.	min. T	avg. T	max T
c22	2	1	0.000020	0.000026	0.000035	1	0.000018	0.000024	0.000031	1	0.000018	0.000024	0.000033
c32	5	2	0.000113	0.000172	0.000258	2	0.000051	0.000060	0.000075	2	0.000051	0.000066	0.000081
c42	9	3	0.005931	0.006842	0.009888	4	0.000093	0.000121	0.000144	3	0.000084	0.000119	0.000147
c52	14	4	0.006591	0.006940	0.007364	7	0.000163	0.000200	0.000241	4	0.000169	0.000194	0.000213
c62	20	-	-	-	-	16	0.000313	0.000417	0.000479	5	0.000272	0.000306	0.000344
c72	27	-	-	-	-	42	0.000679	0.000882	0.000966	6	0.000411	0.000462	0.000561

Table 1: Empirical diagnosis results obtained for the different models. Besides the number of diagnoses, the minimum, average, and maximum running time in seconds for every model is given.

Conclusions

In this paper we formalized diagnosis as constraint satisfaction problem and introduced deviation models for fault localization. In addition, we discussed a framework that allows for comparing different models and to state whether one model is an abstraction of another model. Moreover, we present first empirical results showing that a deviation model based on the qualitative values *smaller*, *equivalent*, or *larger* behaves similar to a representation based on concrete values. The obtained running time for computing single fault diagnosis is also very much promising and may raise its usability for fault localization in programs.

The empirical evaluation is of course limited and has to be extended in the future. Moreover, it is planned to use the comparison model M_C in the domain of fault localization of spreadsheets, where a fast response time is required even in cases of large spreadsheets comprising hundreds of non-empty cells. There we expect that the use of qualitative deviation models improves fault localization substantially.

Acknowledgments

The work described in this paper has been been funded by the Austrian Science Fund (FWF) project DEbugging Of Spreadsheet programs (DEOS) under contract number I2144 and the Deutsche Forschungsgemeinschaft (DFG) under contract number JA 2095/4-1.

References

Autio, K., and Reiter, R. 1998. Structural abstraction in model-based diagnosis. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*.

Cousot, P., and Cousot, R. 1977. Abstract interpreation: A unified lattice model for static analysis of programs by construction of approximation of fixpoints. In *in Proc. POPL'77*, 238–252. Los Angeles: ACM.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.

de Kleer, J.; Mackworth, A. K.; and Reiter, R. 1992. Characterizing diagnosis and systems. *Artificial Intelligence* 56.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Friedrich, G.; Stumptner, M.; and Wotawa, F. 1999. Model-based diagnosis of hardware designs. *Artificial Intelligence* 111(2):3–39.

Gent, I. P.; Jefferson, C.; and Miguel, I. 2006. Minion: A fast, scalable, constraint solver. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*.

Hofer, B., and Wotawa, F. 2014. Why does my spreadsheet compute wrong values? In *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*, volume 25, 112 –121.

Hofer, B.; Wotawa, F.; Abreu, R.; and Außerlechner, S. 2015. Testing for distinguishing repair candidates in spreadsheets - the mussco approach. In *27th International Conference on Testing Software and Systems (ICTSS)*, 124–140.

Mozetič, I. 1991. Hierarchical model-based diagnosis. *International Journal of Man-Machine Studies* 35:329–362.

Nica, I., and Wotawa, F. 2012. Condiag – computing minimal diagnoses using a constraint solver. In *Proceedings of the International Workshop on Principles of Diagnosis (DX)*, 185–191.

Nica, I.; Pill, I.; Quaritsch, T.; and Wotawa, F. 2013. A route to success – a performance comparison of diagnosis algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–95.

Sachenbacher, M., and Struss, P. 2003. Automated qualitative domain abstraction. In Gottlob, G., and Walsh, T., eds., *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, 382–387. Morgan Kaufmann.

Sachenbacher, M., and Struss, P. 2005. Task-dependent qualitative domain abstraction. *Artif. Intell.* 162(1-2):121–143.

Struss, P. 1992. What's in sd? towards a theory of modeling for diagnosis. *Readings in model-based diagnosis* 419–449.

Struss, P. 2004. Deviation models revisited. In *Working Papers of the 15th International Workshop on Principles of Diagnosis (DX-04)*.

Weiser, M. 1982. Programmers use slices when debugging. *Communications of the ACM* 25(7):446–452.

Wotawa, F. 2002. On the Relationship between Model-Based Debugging and Program Slicing. *Artificial Intelligence* 135(1–2):124–143.