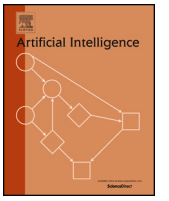
Artificial Intelligence 348 (2025) 104399

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Choosing abstraction levels for model-based software debugging: A theoretical and empirical analysis for spreadsheet programs

Patrick Rodler [a,*], Birgit Hofer [b], Dietmar Jannach [a], Iulia Nica [b], Franz Wotawa [b]

[a] University of Klagenfurt, Austria
[b] Graz University of Technology, Austria

ARTICLE INFO

Keywords:
Model-based diagnosis
Modeling for diagnosis
Model extraction
Abstract models
Software fault localization
Spreadsheet debugging
Automated debugging
Diagnostic reasoning
Qualitative reasoning
Deviation models
Diagnostic accuracy
Diagnostic performance
Diagnosis computation
Constraint programming

ABSTRACT

Model-based diagnosis is a generally applicable, principled approach to the systematic debugging of a wide range of system types such as circuits, knowledge bases, physical devices, or software. Based on a formal description of the system, it enables precise and deterministic reasoning about potential faults responsible for observed misbehavior. In software, such a formal system description can often even be extracted from the buggy program fully automatically. As logical reasoning is central to diagnosis, the performance of model-based debuggers is largely influenced by reasoning efficiency, which in turn depends on the complexity and expressivity of the system description. Since highly detailed models capturing exact semantics often exceed the capabilities of current reasoning tools, researchers have proposed more abstract representations.

In this work, we thoroughly analyze system modeling techniques with a focus on fault localization in spreadsheets—one of the most widely used end-user programming paradigms. Specifically, we present three constraint model types characterizing spreadsheets at different abstraction levels, show how to extract them automatically from faulty spreadsheets, and provide theoretical and empirical investigations of the impact of abstraction on both diagnostic output and computational performance. Our main conclusions are that *(i)* for the model types, there is a trade-off between the conciseness of generated fault candidates and computation time, *(ii)* the exact model is often impractical, and *(iii)* a new model based on qualitative reasoning yields the same solutions as the exact one in up to more than half the cases while being orders of magnitude faster.

Due to their ability to restrict the solution space in a sound way, the explored model-based techniques, rather than being used as standalone approaches, are expected to realize their full potential in combination with iterative sequential diagnosis or indeterministic but more performant statistical debugging methods.

* Corresponding author.
E-mail addresses: patrick.rodler@aau.at (P. Rodler), bhofer@tugraz.at (B. Hofer), dietmar.jannach@aau.at (D. Jannach), inica@ist.tugraz.at (I. Nica), wotawa@ist.tugraz.at (F. Wotawa).

https://doi.org/10.1016/j.artint.2025.104399
Received 31 March 2023; Received in revised form 31 July 2025; Accepted 2 August 2025

Available online 20 August 2025

	A	B	C	D	E	F	G
1	**Sales "BSC"**						
2	**1. half 2017**						
3		**Q1**	**Q2**	**Sum**	**Percentage**	**Bonus Payment**	
4	Mrs. Mayer	10000	15000	25000	1%	250	
5	Mrs. Smith	20000	18000	38000	2%	380	
6	Total			**63000**		**630**	
7							

(a) Regular (data-oriented) view on the spreadsheet content.

	A	B	C	D	E	F	G
1	**Sales "BSC"**						
2	**1. half 2017**						
3		**Q1**	**Q2**	**Sum**	**Percentage**	**Bonus Payment**	
4	Mrs. Mayer	10000	15000	=B4+C4	0.01	=D4*E4	
5	Mrs. Smith	20000	18000	=B5+C5	0.02	=D5*E4	
6	Total			**=D4+D5**		**=F4+F5**	
7							

(b) View on the formulas used in the spreadsheet.

Fig. 1. A simple spreadsheet computing bonus payments (a) and its used formulas (b).

1. Introduction

Model-based diagnosis [58] is a general, domain-independent and principled approach to a systematic, semi-automatic debugging of malfunctioning systems. It has over the last decades found widespread adoption for troubleshooting systems in a broad range of application areas including circuits, hardware, knowledge bases, recommender systems, spreadsheets, schedules, robots, distributed systems, aircrafts, cars, and software [16,25,60,78,24,40,64,80,59,28,74,49]. The idea underlying model-based diagnosis is to exploit a formal logic-based description of the system to be debugged, which characterizes all relevant system components (e.g., gates in a circuit, lines of code of a program) along with their functionality, their interrelations as well as other relevant knowledge about the system. This system description, along with the assumption that all components exhibit their modeled behavior, can be used to draw conclusions about the system behavior by means of automated reasoning techniques. Given observations of the system (e.g., unexpected outputs of a circuit, or asserted outputs of the intended program) that are inconsistent with the model-based predictions about the system behavior, some components must deviate from their modeled functionality. The goal is then to determine sets of components whose malfunction explains the discrepancy between the predicted and observed system behavior. We call such component sets *diagnoses*.

Consider as an example the simple spreadsheet given in Fig. 1(a) which computes the sales values and bonuses for two employees of a company. The components in this case are the spreadsheet cells, and the system description, under the assumption that no components are faulty, is given by the contents (i.e., constants and formulas) of the cells, shown in Fig. 1(b), e.g., $\texttt{B4} = 10\,000$, $\texttt{C4} = 15\,000$, $\texttt{D4} = \texttt{B4} + \texttt{C4}$, and so on. Technically, one could use a constraint knowledge base as a straightforward representation formalism to express this [39]. Let us now assume that the payroll accountant of the company realizes that the overall amount of bonuses to be paid is too low, and should in fact come to 1010. That is, she asserts that $\texttt{F6} = 1010$. Hence, in this case, the system description characterizes the actual behavior of the (buggy) spreadsheet, and the observation gives information about the intended output of some spreadsheet cell. Since, for the cell F6, the value derived by means of the model (630) differs from the intended value (1010), some cell in the spreadsheet must contain a bug. One possible explanation (i.e., a diagnosis) for this discrepancy is, e.g., that cell F5 is faulty (whereas all others are correct); this can be checked by verifying the consistency of the model along with the observation after retracting the assumption that $\texttt{F5} = \texttt{D5} * \texttt{E4}$ (where consistency is now given because $\texttt{F5} = 760$ can be assumed).

Since automated logical inference is a core ingredient to model-based debuggers, their performance depends decisively on the efficiency of the used reasoning engine, which in turn is largely affected by the complexity and expressivity of the system description. For instance, when using an exact representation of the spreadsheet semantics, the mere computation of all single-fault diagnoses (each of which assume a single cell to be faulty) for the very small toy example from Fig. 1 already takes 58 milliseconds on today's hardware. Thus, for sophisticated systems of considerable size, as we have observed, using models characterizing the system at high levels of detail often exceeds the capacity of state-of-the-art reasoning algorithms and can thus render model-based debugging impractical or even infeasible. As a remedy, exploiting the general fact that different models can be employed to express a problem of interest, researchers have proposed less detailed abstract types of models as an attempt to overcome the performance bottleneck caused by exact models (see [10]). The price to pay for the increased computational performance of abstract models is a potentially less succinct set of generated diagnoses, i.e., a higher number of spurious fault explanations. Hence, the degree of detail captured by the system description can be viewed as a means to trade a tendentially lower solution accuracy for a tendentially higher computational efficiency. While this general intuitive understanding is certainly valuable, current research literature lacks studies that investigate the relationship between modeling detail and diagnostic usefulness of different types of system descriptions on a more precise, theoretical as well as on a quantitative, empirical level.

The contribution of this work is a first step towards closing this gap. In particular, we focus on the domain of spreadsheet debugging. Spreadsheets constitute one of the most prominent and influential end-user programming paradigms (see [44]) providing users with an intuitive yet powerful development and execution environment, but only rudimentary debugging support. A more advanced debugging assistance is however a vital requirement for the successful application of spreadsheets since *(1)* spreadsheets are used for a myriad of tasks in a multitude of application areas where critical decisions often rely on spreadsheet calculations, *(2)* spreadsheets are adopted by a heterogeneous user community including both experts on the one end and people with little IT training on the other, which is why a significant human error rate of 3-5 % when writing formulas was ascertained [55], and *(3)* spreadsheet users often find it difficult to identify the source of wrong calculations and to make appropriate corrections [40].

The particular **contributions** of this work are:

- We examine three types of constraint-based modeling paradigms for spreadsheets which capture the semantics of the spreadsheet at different abstraction levels. More specifically, we consider a value-based model which represents the exact semantics of the spreadsheet, a functional dependency model which abstracts from describing numerical values and allows a reasoning at the level of cell (in)correctness, and a newly proposed qualitative deviation model which extends the functional dependency model by considering two manifestations of incorrectness for cells, i.e., “too high” and “too low”. Besides the advantage of enabling more performant reasoning, a pivotal plus of the latter two models is that they do not require users to specify precise numerical values as observations (intended cell outputs), but can deal with vaguer information. In our example from Fig. 1, it would thus suffice to state that “the value of cell `F6` is too low” (qualitative deviation model) or “the value of cell `F6` is incorrect” (functional dependency model).
- We provide theoretical analyses that precisely capture and relate the notions of degree of abstraction, diagnostic accuracy, and inferential power for model types. Moreover, we theoretically prove accuracy relationships between each pair of the analyzed model types.
- We conduct comprehensive empirical evaluations on hundreds of debugging problems from both a real-world and a synthetic corpus of spreadsheets and quantitatively assess the trade-off between diagnostic accuracy and computational performance for the different model types.

The **main conclusions** from our studies are:

- Through our *theoretical analysis*, we show that the set of fault explanations (diagnoses) output by the value-based model is always a subset of or equal to the set of explanations generated by the qualitative deviation model, which in turn is at least as concise as the set of explanations identified by means of the functional dependency model. In other words, all models are complete w.r.t. diagnosis finding, i.e., facilitate the localization of the actually faulty cells, but models applying abstractions are generally unsound and might give rise to more spurious diagnoses.
- In our *empirical study*, we find that the exact value-based model produces significantly fewer diagnoses than the abstract models on average, albeit the difference barely exceeds one order of magnitude. On the other hand, the value-based model is often not (efficiently) applicable, which is why the proposal of more abstract models is well motivated. Further, we observe that diagnosis computation performance is very similar for both abstract models. They are at least three times and by up to more than four orders of magnitude faster than the value-based model. This, combined with the insight that the qualitative deviation model is as good regarding diagnostic accuracy as the value-based model in almost one third of the cases, reveals that abstract models can be a powerful surrogate for their exact and detailed counterparts in scenarios where computational efficiency is a challenge.

Finally, we note that the discussion in the present paper is not only relevant to spreadsheets. Instead, the basic principles of the shown modeling techniques are applicable to other types of programs as well. However, due to certain restrictions of the expressiveness of spreadsheets (e.g., no loops, simple underlying computational model) and the consequential less sophisticated modeling considerations required make spreadsheets an ideal testbed to introduce, study and evaluate automated fault localization methods. In addition, although the model-based diagnosis paradigm is highly appealing in that it allows to precisely and deterministically distinguish between possible and proven impossible fault explanations (diagnoses vs. non-diagnoses), a potential shortcoming is the computational efficiency for large problems. This might necessitate the combination of the suggested methods with complementary techniques such as less precise and more performant spectrum-based fault localization (SFL) approaches that help to rank the diagnoses [3], or with iterative sequential diagnosis approaches [16,71] that enable the acquisition of additional information to successively eliminate spurious fault explanations.

The **organization** of this work is as follows. In Sec. 2, we give preliminaries including a definition of spreadsheets, constraint satisfaction problems, and model-based diagnosis, which forms the basis of our debugging approach. Then, we introduce and explain the different model types in Sec. 3, and show in Sec. 4 how they can be automatically extracted from a given spreadsheet. In Sec. 5, we then study these model types from a theoretical viewpoint based on formal definitions of the notions of model abstractness, diagnostic accuracy, inferential power, soundness, and completeness, and we derive results that relate all model types based on these concepts. To provide quantitative insights into how the proven qualitative superiority relationships between the models regarding diagnostic accuracy find expression in terms of inferior computation times, we report, analyze, and critically discuss the results of comprehensive experiments in Sec. 6. Finally, we review related works in Sec. 7, and summarize our overall findings in Sec. 8.

2. Preliminaries

We now formalize spreadsheets, review basic concepts from the domain of constraint solving, and give the foundations of model-based diagnosis. This will allow us to model a spreadsheet diagnosis problem in terms of a constraint satisfaction problem, and to develop according solution techniques.

2.1. Spreadsheets

In the following, we review the syntax and semantics of spreadsheets, based on the work of Abreu et al. [2]. To simplify the discussion throughout the paper, we put the focus on some core functionality of spreadsheets and omit some advanced features of modern spreadsheet implementations; however, our approach can be extended in a straightforward way to capture additional features as well.

Definition 1 *(Spreadsheet).* A *spreadsheet* Σ is a finite set of cells $\{c_1, \ldots, c_n\}$, where each cell

- can be uniquely identified using its column index C, its row index R, and its worksheet index W, and
- has a content $\chi(c)$, which might be empty (i.e., ϵ), a number, or a string representing a spreadsheet formula.

Remark 1. Regarding the cell identifiers, we omit the worksheet index in this paper, and assume that the first cell in the first row has the unique identifier, `A1`, the first element in the second row, `A2`, the second element in the first row, `B1`, and so on. Because of the uniqueness of the cell identifiers, we do not distinguish between a cell itself and its identifier. As to the cell types, we restrict the debugging focus to only cells of type number. Moreover, we assume that references to empty cells are checked, detected, and corrected prior to the debugging process, i.e., we do not further consider empty cells in what follows. □

Definition 2 *(Syntax of a spreadsheet formula).* A spreadsheet formula (in our restricted core formula language) starts with an `=` followed by either

- a number n,
- a cell reference, i.e., the unique identifier of a cell,
- a numerical operation $c_1 \odot c_2$ with $\odot \in \{+, -, *, /\}$ and cell references c_1 and c_2, or
- a conditional $\texttt{if}(c_1 \square c_2, c_3, c_4)$ with $\square \in \{<, =, >\}$ and cell references $c_1, \ldots, c_4$.

Example 1. In our salary spreadsheet from Fig. 1, twelve cells contain non-empty formulas. E.g., cell `B4` has the value 10 000, and cell `D6` holds a formula "`= D4 + D5`", which means that $\chi(\texttt{B4}) = 10000$ and $\chi(\texttt{D6}) =$ "`= D4 + D5`".

The purpose of a spreadsheet Σ is to compute the value for every cell $c \in \Sigma$, which is determined by the content of c, as defined next:

Definition 3 *(Value of a spreadsheet cell).* Let Σ be a spreadsheet and $c \in \Sigma$ be one of its cells. The *value* of c, denoted by $cv(c)$, is defined as follows:

- If $\chi(c) = n$ and n is a number, then $cv(c) = n$.
- If $\chi(c) = f$ and f is a spreadsheet formula, then $cv(c) = eval(f)$, where $eval(f)$ is the result of the evaluation of f, defined in Definition 4 below.

Definition 4 *(Semantics of a spreadsheet formula).* Let f be a spreadsheet formula. The semantics of f, i.e., $eval(f)$, is defined as follows:

- If f is of the form "`=` n" and n is a number, then $eval(f) = n$.
- If f is of the form "`=` c" and c is a cell, then $eval(f) = cv(c)$.
- If f is of the form "`=` $c_1 \odot c_2$" for $\odot \in \{+, -, *, /\}$, then $eval(f) = cv(c_1) \odot cv(c_2)$.
- If f is of the form "`=` $\texttt{if}(c_1 \square c_2, c_3, c_4)$" for $\square \in \{<, =, >\}$, then $eval(f) = cv(c_3)$ if $cv(c_1) \square cv(c_2)$ is true, and $eval(f) = cv(c_4)$ otherwise.

We next define input and output cells of a spreadsheet. Intuitively, input cells are used in some spreadsheet computations, but are themselves not the result of any spreadsheet computations, whereas output cells are the result of some spreadsheet computations, but are themselves not used in any spreadsheet computations.

Definition 5 *(Input/Output cell).* Let Σ be a spreadsheet. A cell $c \in \Sigma$ is

- an *input cell* iff c is used in the formula of some other cell in Σ and c itself does not reference any cell in its formula, and
- an *output cell* iff c is not used in the formula of any other cell, but c itself references some other cell.

Example 2. For the salary spreadsheet in Fig. 1, all input cells are given by {`B4`, `B5`, `C4`, `C5`, `E4`, `E5`}, and all output cells by {`D6`, `F6`}.

Remark 2. Although the expounded theory of spreadsheets, cells, and formulas does not reflect the entire range of functionality found in today's spreadsheet applications, more complicated expressions not covered by our definitions can often be handled by using corresponding cell references. For instance, the expression "`= D2 + (D3 * D4)`" can be represented by a cell that specifies the addition and refers to two other cells, i.e., `D2` and a cell containing the formula "`= D3 * D4`". A spreadsheet comprising such (indirectly expressible) formulas can be automatically preprocessed by introducing corresponding auxiliary cells, in a way that *(i)* all resulting cells contain only formulas matching the syntax specified in Definition 2, and *(ii)* all formulas in cells originally present in the spreadsheet remain semantically (as per Definition 4) equivalent.

In general, albeit not allowing to represent all possible spreadsheets, the basic principles and constraint techniques used for debugging outlined in this paper are in line with existing research (e.g., [39]) and serve as a basis for future extensions. □

2.2. Constraint solving

In this section, we briefly recap the basic definitions relevant to constraint solving, based on [32,34]. For more in-depth information on the topic, we refer the interested reader to [18].

The first step towards approaching a problem by means of constraint solving techniques is to model the problem in terms of a *constraint system*:

Definition 6 *(Constraint system).* A *constraint system* is a tuple $(VARS, DOM, CONS)$, where

- $VARS$ is a finite set of variables,
- DOM is a function mapping each variable $x \in VARS$ to its non-empty domain $DOM(x)$, and
- $CONS$ is a finite set of constraints.

A *constraint* $c \in CONS$ is a pair $(scope(c), tl(c))$ where

- $scope(c)$ is a tuple $(x_1, \ldots, x_k)$ of variables $x_1, \ldots, x_k \in VARS$, and
- $tl(c) \subseteq DOM(x_1) \times \cdots \times DOM(x_k)$ is a relation over the domains of the variables in $scope(c)$, i.e., $tl(c)$ is a set of tuples $(v_1, \ldots, v_k)$ of values $v_i \in DOM(x_i)$ for $1 \leq i \leq k$.[1]

Remark 3. Intuitively, $scope(c)$ refers to the variables involved in the constraint c, and $tl(c)$ declares all value combinations $(v_1, \ldots, v_k)$ which satisfy the constraint c when assigned to the variables $(x_1, \ldots, x_k)$ in $scope(c)$. In general, variables in constraint systems might have infinite domains. However, usually in constraint solving practice, and in this work in particular, the focus is on finite variable domains. That is, in case of spreadsheet cells with an infinite range of possible values, we assume that the modeled domains are large enough, i.e., that all potentially needed value combinations are covered by the specified constraint system. □

Initially, no variable of a constraint system has an allocated value. When, throughout the solving process, some (all) variables are allocated a value of their respective domain, we speak of a *(complete) value assignment*:

Definition 7 *(Value assignment).* Let $CS = (VARS, DOM, CONS)$ be a constraint system and $VARS' \subseteq VARS$. Then, a *value assignment* VA for CS assigns to each variable x from $VARS'$ a value from its domain $DOM(x)$. Formally:

- $VA \subseteq \{(x, v) \mid x \in VARS', v \in DOM(x)\}$, and
- for each variable $x \in VARS'$ there is one, and only one, $v \in DOM(x)$ such that $(x, v) \in VA$.

We call a value assignment VA for CS *complete* iff $VARS' = VARS$, i.e., VA assigns to each variable x from $VARS$ a value from its domain $DOM(x)$.

Each constraint imposes a restriction on the allowed value assignments to variables of a constraint system. Hence, a particular value assignment can either comply with a particular constraint (*satisfaction*) or not (*violation*):

[1] For presentation purposes (cf. Sec. 3), we use throughout this work an *explicit representation* of constraints c in the form of a relation $tl(c)$ that enumerates all value-tuples that make the constraint true. A more concise (and often also computationally favorable) alternative is to draw on an *implicit representation* where each constraint constitutes a predicate that can be either true or false depending on a given variable assignment. For further details on this aspect, we refer the reader to [18].

Definition 8 *(Constraint satisfaction/violation).* Let $CS = (VARS, DOM, CONS)$ be a constraint system, c a constraint in $CONS$ with $scope(c) = (x_1, \ldots, x_k)$ and $VA \supseteq \{(x_1, v_1), \ldots, (x_k, v_k)\}$ a value assignment for CS. Then, *VA satisfies c* iff $(v_1, \ldots, v_k) \in rl(c)$; otherwise, *VA violates c*.

A *valid assignment* (or: *solution*) for a given constraint system is an assignment of values to all variables such that all constraints are satisfied:

Definition 9 *(Valid assignment/Solution).* Let $CS = (VARS, DOM, CONS)$ be a constraint system. We call a value assignment VA for CS *valid* (or: *a solution*) for CS iff VA is complete and satisfies all constraints $c \in CONS$.

The problem of finding a solution for a constraint system is called the *constraint satisfaction problem (CSP)*:

Definition 10 *(Constraint satisfaction problem (CSP)).*
Given: A constraint system $CS = (VARS, DOM, CONS)$.
Find: A valid assignment (solution) for CS.

Definition 11 *((Un)Satisfiable constraint system).* We call a constraint system *(un)satisfiable* iff it has some (no) solution.

A naive approach to solving CSPs, which involves generating all possible complete value assignments and checking the satisfaction of all constraints for each assignment, is exponential in the number of variables and thus impractical. Modern constraint solvers (e.g., Minion [27] and Choco [56]) overcome this issue by exploiting heuristics or the inherent structural information of constraint systems to achieve a high solving performance for many practical problems.[2] In the following, we assume a function ConstrSolver that implements a constraint solver, which, given a constraint system as input, returns a set of valid assignments for it.

2.3. Model-based diagnosis

Given a system, such as a circuit [16], a program [49], a spreadsheet [38], or a knowledge base [78], that does not behave as expected, Reiter [58] defines a *diagnosis problem* as a tuple $(COMP, SD, OBS)$. In this context,

- $COMP$ (*system components*) is a set of possibly faulty system constituents relevant to the diagnostic task (e.g., gates, lines of code, spreadsheet cells, or logical axioms),
- SD (*system description*) is a formal characterization of the system in terms of a monotonic knowledge representation language (e.g., propositional logic, first-order logic, or some constraint language), and
- OBS (*system observations*) is a formal description of the observed system behavior (e.g., measured system outputs) in the same language.

Besides other relevant system knowledge, the system description SD in particular comprises the description of the normal behavior of each system component. To this end, for each $c \in COMP$, a special abnormality predicate ab_c is used, and the nominal functionality of c (under the assumption $\neg ab_c$, i.e., "c is not abnormal") is specified in SD. Given that the system is described using propositional logic, e.g., this could be accomplished by adding for each $c \in COMP$ a sentence $\neg ab_c \rightarrow beh_c$ to SD where beh_c models the normal behavior of c in propositional logic terms. If SD does not include any descriptions of abnormal component behaviors, which is our assumption throughout this work, SD is commonly referred to as a *weak fault model* (as opposed to strong fault models, cf., e.g., [81,15]).

When the expected system behavior, logically derived from the system description SD under the assumption that all components in $COMP$ are normal, is inconsistent with the given evidence about the system in terms of OBS, the goal is to find a diagnosis. A *diagnosis* is a setting of the abnormality predicate for each system component, where a subset of the components are assumed abnormal in a way that the given inconsistency between prediction and observation is resolved. Intuitively, a diagnosis effectuates the retraction of the original normality assumption for a set of components whose faultiness provides one possible explanation for the system's misbehavior.

In this work, the type of system of interest is spreadsheets, the components relevant to the diagnosis task are spreadsheet cells, the used system description language are constraints, and the type of observations assumed are statements about values of spreadsheet cells (e.g., specifications of the correct values or assertions such as "is incorrect" or "too high"). To characterize a spreadsheet in terms of the system description SD, we will draw on three constraint *model types* with different degrees of abstraction regarding the representation of cell values (as discussed later in Sec. 3). Applying any of these model types will result in a specification SD of a given spreadsheet as a constraint system $(VARS, DOM, CONS)$ (cf. Sec. 2.2). Leaving the particular model type as well as the specifics of modeling a spreadsheet as a constraint system open for now, we define a *spreadsheet-constraint diagnosis problem* as follows:

[2] See also http://www.constraint.org/en/tools/ for further constraint languages, tools, and libraries.

Definition 12 *(Spreadsheet-constraint diagnosis problem)*. Let MT be a constraint model type, Σ be a spreadsheet and $\Sigma' \subseteq \Sigma$ be the possibly faulty cells in Σ.[3] Let

- $COMP := \Sigma'$
- SD be the constraint system $(VARS, DOM, CONS)$ that models Σ as per MT, where, for each $c \in \Sigma \setminus COMP$,
  - $VARS$ includes a variable c' (cell value) with $DOM(c')$ as per MT
  - $CONS$ contains a set of constraints that model in terms of MT the semantics of c (i.e., the relationship between the cell value c' and the values of the cells referenced in the formula of c)

  and, for each $c \in COMP$,
  - $VARS$ includes a variable c' (cell value) with $DOM(c')$ as per MT, and a variable ab_c (abnormality of cell content) with $DOM(ab_c) = \{\mathsf{true}, \mathsf{false}\}$,
  - $CONS$ contains a set of constraints that model in terms of MT *(i)* the normal semantics of c (i.e., the relationship between the cell value c' and the values of the cells referenced in the formula of c) if ab_c is set to false, and *(ii)* an undefined semantics of c if ab_c is set to true (weak fault model)
- OBS be a set of constraints $COBS$ represented in terms of MT and describing the given observations about some cells in Σ'.

Now, let CS be the constraint system $(VARS, DOM, CONS \cup COBS)$ resulting from the combination of SD and OBS. Then, the tuple $(CS, COMP)$ is called a *spreadsheet-constraint diagnosis problem*.

If, for a spreadsheet-constraint diagnosis problem $(CS, COMP)$, the observations in terms of $COBS$ contradict the predictions in terms of $CONS$ under the normality assumption for all cells in $COMP$, the constraint system $CS = (VARS, DOM, CONS \cup COBS)$ is unsatisfiable. A *diagnosis* for $(CS, COMP)$ is then a set Δ of spreadsheet cells from $COMP$ that, when assumed abnormal while all other cells in $COMP \setminus \Delta$ are assumed normal, explains the observed faults in the spreadsheet, i.e., makes CS satisfiable:

Definition 13 *(Diagnosis)*. Let $(CS, COMP)$ be a spreadsheet-constraint diagnosis problem with $CS = (VARS, DOM, CONS \cup COBS)$, let $\Delta \subseteq COMP$, and let $CONS_\Delta$ denote the set of constraints $\{((ab_c), \{(\mathsf{true})\}) \mid c \in \Delta\} \cup \{((ab_c), \{(\mathsf{false})\}) \mid c \in COMP \setminus \Delta\}$ specifying the assumption that all $c \in \Delta$ are abnormal and all $c \in COMP \setminus \Delta$ are normal. Then, Δ is a *diagnosis* for $(CS, COMP)$ iff the constraint system $(VARS, DOM, CONS \cup COBS \cup CONS_\Delta)$ is satisfiable. We refer to a diagnosis Δ for $(CS, COMP)$ as

- *minimal* iff there is no diagnosis $\Delta' \subset \Delta$ for $(CS, COMP)$, and
- *minimum-cardinality* iff there is no diagnosis Δ' with $|\Delta'| < |\Delta|$ for $(CS, COMP)$.

In practice, there are often multiple diagnoses for a spreadsheet-constraint diagnosis problem. In order to rank diagnoses, diagnosis methods usually focus on minimal diagnoses only (which are representative of all diagnoses under the weak fault model [15]). In several scenarios, e.g., when components are much more likely to be normal than faulty, it can even be desired and reasonable to concentrate solely on minimum-cardinality diagnoses. Another practical strategy, which we will draw on in this work, is the systematic enumeration of diagnoses in the order of their cardinality, by starting with the (likely) most preferred minimum-cardinality ones before gradually extending the attention to further (likely) less preferred ones of higher cardinality until the diagnosis pinpointing the actually faulty cells is located.

We now illustrate the modeling of a spreadsheet as a spreadsheet-constraint diagnosis problem as well as the concept of a diagnosis by means of our running example:

Example 3. Reconsider our sales example given in Fig. 1. For simplicity, and since the purpose of the following explications is the demonstration of the principle, we only focus on the cells in the area between `D4` and `F5`, and on cell `F6`, while assuming that the numbers in `D4` and `D5` are given as fixed values 25 000 and 38 000, respectively. That is, $\Sigma = \{\mathtt{D4}, \mathtt{E4}, \mathtt{F4}, \mathtt{D5}, \mathtt{E5}, \mathtt{F5}, \mathtt{F6}\}$. Let us assume that the possibly faulty cells are all non-input cells, i.e., $\Sigma' = \{\mathtt{F4}, \mathtt{F5}, \mathtt{F6}\}$ (cf. Definition 12), and let the observations consist of the specification of the expected values 760 and 1010, respectively, for the cells `F5` and `F6`. Finally, let the used constraint model type MT (cf. Definition 12) be the value-based model (detailed later in Sec. 3) which involves one-to-one translations of cell formulas to constraints (based on Definition 4).

Based on these premises, a spreadsheet-constraint diagnosis problem $(CS, COMP)$ with $CS = (VARS, DOM, CONS \cup COBS)$ can be specified in terms of Definition 12 as follows. First, $COMP = \Sigma'$. Second, $VARS = \{\mathtt{D4}', \mathtt{E4}', \mathtt{F4}', \mathtt{D5}', \mathtt{E5}', \mathtt{F5}', \mathtt{F6}', ab_{\mathtt{F4}}, ab_{\mathtt{F5}}, ab_{\mathtt{F6}}\}$, $DOM(x') = \mathbb{R}$ for all variables x' modeling cells x, and $DOM(ab_x) = \{\mathsf{true}, \mathsf{false}\}$ for all ab_x variables, $COBS = \{((\mathtt{F5}'), \{(760)\}), ((\mathtt{F6}'), \{(1010)\})\}$, and $CONS = \{c_{\mathtt{D4}}, c_{\mathtt{E4}}, c_{\mathtt{F4}}, c_{\mathtt{D5}}, c_{\mathtt{E5}}, c_{\mathtt{F5}}, c_{\mathtt{F6}}\}$ where the constraints c_x represent the cells x as follows:

$$c_{\mathtt{D4}} = ((\mathtt{D4}'), \{(25\,000)\}) \qquad c_{\mathtt{D5}} = ((\mathtt{D5}'), \{(38\,000)\})$$
$$c_{\mathtt{E4}} = ((\mathtt{E4}'), \{(0.01)\}) \qquad c_{\mathtt{E5}} = ((\mathtt{E5}'), \{(0.02)\})$$

[3] Not necessarily all spreadsheet cells are possibly faulty. In fact, a user might specify some cells as correct, e.g., after verifying them manually. The effect of such a restriction of the possible fault locations is a smaller search space for diagnoses, which can result in a higher computational efficiency of the diagnostic process. See also [61, Sec. 3.2].

$$c_{\texttt{F4}} = ((ab_{\texttt{F4}}, \texttt{D4}', \texttt{E4}', \texttt{F4}'),$$
$$\{(\mathsf{false}, x, y, x * y) \mid x \in \mathrm{DOM}(\texttt{D4}'),\ y \in \mathrm{DOM}(\texttt{E4}')\} \cup$$
$$\{(\mathsf{true}, x, y, z) \mid x \in \mathrm{DOM}(\texttt{D4}'),\ y \in \mathrm{DOM}(\texttt{E4}'),\ z \in \mathrm{DOM}(\texttt{F4}')\})$$
$$c_{\texttt{F5}} = ((ab_{\texttt{F5}}, \texttt{D5}', \texttt{E4}', \texttt{F5}'),$$
$$\{(\mathsf{false}, x, y, x * y) \mid x \in \mathrm{DOM}(\texttt{D5}'),\ y \in \mathrm{DOM}(\texttt{E4}')\} \cup$$
$$\{(\mathsf{true}, x, y, z) \mid x \in \mathrm{DOM}(\texttt{D5}'),\ y \in \mathrm{DOM}(\texttt{E4}'),\ z \in \mathrm{DOM}(\texttt{F5}')\})$$
$$c_{\texttt{F6}} = ((ab_{\texttt{F6}}, \texttt{F4}', \texttt{F5}', \texttt{F6}'),$$
$$\{(\mathsf{false}, x, y, x + y) \mid x \in \mathrm{DOM}(\texttt{F4}'),\ y \in \mathrm{DOM}(\texttt{F5}')\} \cup$$
$$\{(\mathsf{true}, x, y, z) \mid x \in \mathrm{DOM}(\texttt{F4}'),\ y \in \mathrm{DOM}(\texttt{F5}'),\ z \in \mathrm{DOM}(\texttt{F6}')\})$$

Recall that we assume a weak fault model, where only the normal behavior of cells and included formulas is modeled (case: ab_x variable set to false), and any behavior is possible if the cell is abnormal (case: ab_x variable set to true). Hence, a constraint c_i above does not restrict the value of a cell if the cell is assumed abnormal, which is reflected by allowing all possible value combinations in this case. Also note, although we model all number domains to be real-valued and thus infinite in this example, we usually deal with only finite domains in practice (cf. Remark 3).

When assuming all cells in this example to be correct, by adding the constraints $CONS_\Delta$ for $\Delta = \emptyset$ (cf. Definition 13), the resulting constraint system $(VARS, DOM, CONS \cup COBS \cup CONS_\Delta)$ is unsatisfiable. The reason is that, e.g., the normality assumption for cell F5 allows us to derive that the value of this cell must be 380, which contradicts the first constraint in *COBS*, which allows only the value 760 for F5. As it turns out, assuming $ab_{\texttt{F5}} = \mathsf{true}$, i.e., setting $\Delta = \{\texttt{F5}\}$, already explains all discrepancies (values of cells F5 and F6) between the model-based prediction (infers that values of F5 and F6 are correct) and the given observations of a spreadsheet user (states that values of F5 and F6 are wrong). To see this, observe that, given $\Delta = \{\texttt{F5}\}$, the constraint solver would set the variable F5′ to 760, which together with the value 250 deduced for variable F4′ leads to the derivation of the value 1010 for variable F6′, i.e., no contradiction arises. Consequently, {F5} is both a minimal and a minimum-cardinality diagnosis for the given spreadsheet-constraint diagnosis problem.

There is a wide landscape of algorithms for diagnosis computation in the literature, including the HS-tree algorithm [58], along with its corrected version HS-DAG [29] and parallelized versions [41], the GDE [16], stochastic search algorithms (e.g., SAFARI [21]), Boolean strategies (e.g., BHS-tree [47]), divide-and-conquer methods for high-cardinality diagnoses (e.g., Inv-HS-tree [79,43,67]), memory-efficient techniques (e.g., RBF-HS and HBF-HS [65]), iterative methods [70,62], or algorithms using constraints directly (e.g., ConDiag [54]). Given a suitable constraint solver for satisfiability checking, all of these available diagnosis algorithms can be used in the context of fault localization in spreadsheets. Apart from constraint solvers, it is also possible to use Max-CSP techniques (see, e.g., [11]), SAT-based or SAT-related solvers (see [50,22,46]), SMT solvers like Z3 [17], or quantum SAT solvers [51]. For an analysis of different diagnosis algorithms and underlying reasoning methods we refer the interested reader to [53,66] for empirical studies and to [70, Sec. 4] as well as [69] for a comprehensive survey and theoretical comparison.

Algorithm 1 ConDiag(P, n).

Input: A spreadsheet-constraint diagnosis problem $P = (CS, COMP)$, the maximal cardinality n of diagnoses to be returned
Output: All minimal diagnoses for P up to cardinality n

1: $D = \{\}$
2: **for** $i = 0$ **to** n **do**
3: $c_{card} = \textsc{GetConstr}\left(\sum_{c \in COMP} \textsc{isTrue}(ab_c) = i\right)$
4: $CS_{card} = \textsc{addConstr}(CS, c_{card})$
5: $Solutions = \textsc{ConstrSolver}(CS_{card})$
6: $Diags_{card} = \textsc{extractDiags}(Solutions)$
7: **if** $i = 0$ **and** $Diags_{card} = \{\emptyset\}$ **then**
8: **return** $Diags_{card}$
9: $D = D \cup Diags_{card}$
10: $c_{min} = \textsc{GetConstr}(\text{“no supersets of } Diags_{card}\text{”})$
11: $CS = \textsc{addConstr}(CS, c_{min})$
12: **return** D

For the sake of completeness, we briefly review the **ConDiag** algorithm [54], given by Algorithm 1, which we will use in our empirical evaluations in Sec. 6. It gets a spreadsheet-constraint diagnosis problem P and a natural number n as input, and outputs all minimal diagnoses up to size n for P.

To this end, the algorithm implements a for-loop, where all minimal diagnoses of cardinality i are computed in iteration i, starting with $i = 0$. In the very first iteration ($i = 0$), the algorithm directly returns (line 8) if the diagnosis $\emptyset$ is found. The reason is that all other diagnoses must in this case be supersets of $\emptyset$, which is why they must be non-minimal (cf. Definition 13). Otherwise, if $\emptyset$ is not a diagnosis, the initially empty solution set D is successively filled with minimal diagnoses of increasing cardinality by iterating the for-loop. Per iteration, the algorithm proceeds as follows:

First, in line 3, a constraint c_{card} is created (function GETCONSTR) which specifies that diagnoses of exactly cardinality i are to be computed in iteration i, i.e., the number of ab_c variables set to true is fixed to i (where the function ISTRUE returns 1 if its Boolean argument is true, and 0 else). In line 4, c_{card} is then added to the constraint system CS (function ADDCONSTR), before the solver CONSTRSOLVER is called for the resulting constraint system CS_{card} with the configuration to compute all solutions (line 5). These are then (line 6) postprocessed by the EXTRACTDIAGS function which extracts, for each solution, exactly the components $c \in COMP$ for which the ab_c variables are set to true, and returns all resulting component sets, stored in $Diags_{card}$. These are precisely the diagnoses of cardinality i and are added to the solution set D in line 9. Finally, in order for subsequent iterations to not produce non-minimal diagnoses (i.e., supersets of already found ones), the algorithm builds a new constraint c_{min} in line 10 which is violated whenever the ab_c variables for all components of some $\Delta \in Diags_{card}$ are set to true. This new constraint is eventually added to the original constraint system CS of the problem P (line 11) and will take effect in every subsequent iteration.

We assume that diagnoses do exist for the input problem P when calling **ConDiag**, which can be checked beforehand by verifying the satisfiability of the constraint system $(VARS, DOM, CONS \cup COBS \cup CONS_{\Delta})$ for $\Delta = COMP$. From the analytical perspective, **ConDiag** is guaranteed to terminate (assuming that the used constraint solver CONSTRSOLVER terminates). The worst-case runtime complexity of **ConDiag** is exponential in the number of ab_c variables, i.e., in $O(|\Sigma'|^n)$ where Σ' is the set of possibly faulty spreadsheet cells (cf. Definition 12). Since n is a parameter of the algorithm, however, **ConDiag** is in fact fixed-parameter tractable. A similar argument holds for the fixed-parameter tractable worst-case space complexity of **ConDiag** (note that all $O(|\Sigma'|^n)$ computed diagnoses of cardinality up to n must be stored by the algorithm). Thus, when searching for diagnoses of small cardinality (one to three), the algorithm's runtime and memory behavior is acceptable for many practical applications, as our experimental results in Sec. 6 shall testify. Note also, in case a given problem case turns out to be particularly demanding, a wide range of alternative (more time- and/or space-efficient) strategies can be adopted as well, as discussed above.

3. Modeling paradigms for spreadsheet diagnosis

In this section, we review different constraint model types that can be used to represent a given spreadsheet along with observations about wrong cell values as a spreadsheet-constraint diagnosis problem. Stated in terms of Definition 12, we thus discuss different model types MT and how each of these types maps a spreadsheet Σ to a constraint system $(VARS, DOM, CONS)$ and how it describes given evidence about some spreadsheet cells as a set of constraints $COBS$. In particular, we detail three model types:

- A *value-based model*, which represents the semantics of spreadsheet formulas one-to-one as constraints and in this vein very closely captures the spreadsheet structure (as already sketched in Example 3),
- a *functional dependency model*, which, instead of specific cell values, models only whether cell values are "correct" or "incorrect", and
- a *qualitative deviation model*, which refines the functional dependency model by allowing the expression of two specific types of cell value incorrectness, i.e., "too high" or "too low".

Depending on the type of model adopted for spreadsheet representation, different types of observations are assumed. E.g., consider cell `F5` in the sales example from Fig. 1, whose value computes to 380 using the formulas in the spreadsheet. Now, to state that this value deviates from the expected value, the value-based model assumes a (set of) numbers(s) to be expressed which covers the expected value of `F5` (in this case, 760), whereas it suffices for the functional dependency and qualitative deviation models to specify that the output of the cell `F5` is "incorrect" and "too low", respectively.

Importantly, for all discussed model types, a spreadsheet-constraint diagnosis problem can be *automatically* extracted from a given spreadsheet and corresponding observations, without any additional effort for the programmer or spreadsheet user, as we will demonstrate in Sec. 4. In order to make the formulation of automated model extraction algorithm in Sec. 4 precise and unambiguous, we explain each of the three models in exact formal terms in what follows.

3.1. Modeling aspects common to all model types

All models share a common specification of the variables $VARS$ in that

- the value (as per the used model type) of each cell c in the spreadsheet Σ is represented by one variable $c' \in VARS$ (cf. Definition 12),
- the abnormality (incorrectness) of the formula in each possibly faulty cell c in Σ' is modeled by a variable $ab_c \in VARS$ (cf. Definition 12), and
- the outcome of evaluating the condition $c_1 \square c_2$ (as per the used model type) for each cell c including a conditional $\texttt{if}(c_1 \square c_2, c_3, c_4)$ is described by a variable $C_c \in VARS$.

Thus, the characterizations of the models in Secs. 3.2–3.4 involve the specification of the remaining aspects DOM, $CONS$ and $COBS$ of the spreadsheet-constraint diagnosis problem (cf. Definition 12), where we assume

- a function $\text{CONSTR}_{MT}(c)$ that returns a set of (one or two) constraint(s) for cells $c \in \Sigma$ based on
  - the formula $f(c)$ in c (cf. Definition 2),

  - the semantics of spreadsheet formulas (cf. Definition 4),
  - the used model type MT, and
  - whether c is possibly faulty, i.e., $c \in \Sigma'$, or known/assumed correct, i.e., $c \in \Sigma \setminus \Sigma'$ (cf. Definition 12), and
- a function $\textsc{ObsConstr}_{MT}(c, S)$ that returns a constraint for a given observation of the form (c, S) where $c \in \Sigma$ is a cell and S is a set of values (over a particular domain defined by the used model type MT), and the tuple (c, S) reflects the expression that the value of c must be one of the values in S.

Finally, note that, for all described types of models, we assume that faults *(i)* might influence the computed values for cells, but *(ii)* do not lead to exceptions such as division by zero. The reason for this assumption is our focus on *comparing* different model types while keeping them as simple as possible.

3.2. Value-based model (VBM)

3.2.1. Idea

A value-based model [2,39] represents formulas stored in spreadsheet cells *one-to-one* as constraints, e.g., a cell `D4` including the formula "`= B4 + C4`" is modeled by a constraint formalizing the equation $\texttt{D4}' = \texttt{B4}' + \texttt{C4}'$ for the variables $\texttt{D4}'$, $\texttt{B4}'$, $\texttt{C4}'$ describing the respective cell values. The intention of this model is to express the exact semantics of the spreadsheet so that the diagnostic reasoning operates on the level of concrete numerical values.

3.2.2. Variable domains (DOM)

In the value-based model, the domain $DOM(x)$ of a variable related to a cell $c \in \Sigma$ is specified as the range of values the respective spreadsheet cell c might attain in case x is a variable modeling a cell value, and as either `true` or `false` in case x is an abnormality variable or a variable modeling a condition. Formally:

$$DOM(x) = \begin{cases} \mathbb{R} & \text{if } x = c' \\ \{\mathsf{true}, \mathsf{false}\} & \text{if } x = ab_c \text{ or } x = C_c \end{cases}$$

As discussed in Remark 3, infinite domains are modeled as sufficiently large finite domains in constraint solving practice.

3.2.3. Cell constraints (CONS)

The function $\textsc{cConstr}_{VBM}(c)$ maps spreadsheet cells $c \in \Sigma$ directly to constraints and returns one or two resulting constraints per cell depending on the type and content of the cell. We describe the function output separately based on whether the input cell c is *(i)* possibly faulty ($c \in \Sigma'$) or *(ii)* known/assumed fault-free ($c \in \Sigma \setminus \Sigma'$), see Definition 12:

1. Let $c \in \Sigma'$ (c *is possibly faulty*). Then the output of $\textsc{cConstr}_{VBM}(c)$ for the different cases regarding the formula $f(c)$ in c is:
   - If $f(c)$ is a number n:
     $$\textsc{cConstr}_{VBM}(c) = \left\{ \begin{array}{l} ((ab_c, c'), \\ \{(\mathsf{false}, n)\} \cup \\ \{(\mathsf{true}, v) \mid v \in DOM(c')\}) \end{array} \right\}$$
     (*Explanation:* If c is assumed normal, then it can only include the specified value n, otherwise it might contain any value.)
   - If $f(c)$ is a reference to cell $\tilde{c}$:
     $$\textsc{cConstr}_{VBM}(c) = \left\{ \begin{array}{l} ((ab_c, c', \tilde{c}'), \\ \{(\mathsf{false}, v, v) \mid v \in DOM(\tilde{c}')\} \cup \\ \{(\mathsf{true}, v, \tilde{v}) \mid v \in DOM(c'), \tilde{v} \in DOM(\tilde{c}')\}) \end{array} \right\}$$
     (*Explanation:* If c is assumed normal, then it must include the same value as $\tilde{c}$, otherwise it might contain any value.)
   - If $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$:
     $$\textsc{cConstr}_{VBM}(c) = \left\{ \begin{array}{l} ((ab_c, c_1', c_2', c'), \\ \{(\mathsf{false}, v_1, v_2, v_1 \odot v_2) \mid \\ v_1 \in DOM(c_1'), v_2 \in DOM(c_2')\} \cup \\ \{(\mathsf{true}, v_1, v_2, v) \mid v \in DOM(c'), \\ v_1 \in DOM(c_1'), v_2 \in DOM(c_2')\}) \end{array} \right\}$$
     (*Explanation:* If c is assumed normal, then it must contain exactly the result of the operation $c_1 \odot c_2$, else it might contain any value.)
   - If $f(c)$ is of the form $\texttt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<, =, >\}$:
     $\textsc{cConstr}_{VBM}(c) = \{cons_C, cons_c\}$ where
     $$cons_C := \left(\begin{array}{l} (c_1', c_2', C_c), \\ \{(v_1, v_2, \mathsf{true}) \mid v_1 \in DOM(c_1'), v_2 \in DOM(c_2'), \\ v_1 \square v_2 \text{ is } \mathsf{true}\} \cup \\ \{(v_1, v_2, \mathsf{false}) \mid v_1 \in DOM(c_1'), v_2 \in DOM(c_2'), \\ v_1 \square v_2 \text{ is } \mathsf{false}\} \end{array} \right)$$

(*Explanation:* The "auxiliary" constraint $cons_C$ models the condition $c_1 \square c_2$ in the `if` expression. It is required in the constraint $cons_c$ below, which models the semantics of the `if` formula by choosing the respective cell value depending on the evaluation of the condition.)

$$cons_c := \left(\begin{array}{l} (ab_c, C_c, c'_3, c'_4, c'), \\ \{(\mathsf{false}, \mathsf{true}, v_3, v_4, v_3) \mid \\ v_3 \in DOM(c'_3), v_4 \in DOM(c'_4)\} \cup \\ \{(\mathsf{false}, \mathsf{false}, v_3, v_4, v_4) \mid \\ v_3 \in DOM(c'_3), v_4 \in DOM(c'_4)\} \cup \\ \{(\mathsf{true}, b, v_3, v_4, v) \mid b \in \{\mathsf{true}, \mathsf{false}\}, \\ v_3 \in DOM(c'_3), v_4 \in DOM(c'_4), v \in DOM(c')\} \end{array} \right) \tag{1}$$

(*Explanation:* If c is assumed normal, then it attains the value of cell c_3 (second entry of the `if` expression) given that the condition evaluates to `true`, and the value of cell c_4 (third entry of the `if` expression) if the condition evaluates to `false`. If c is assumed abnormal, the cell can attain any value, irrespective of the evaluation of the condition and of the contents of cells c_3, c_4.)

2. Let $c \in \Sigma \setminus \Sigma'$ (*c is known/assumed correct*). Then the output of $\text{cConstr}_{VBM}(c)$ for the different cases regarding the formula $f(c)$ in c is analogue to the ones for the case $c \in \Sigma'$ above with ab_c set to `false`, with the difference that there is no ab_c variable here since the cells are known correct. The explicit constraints can be found in Appendix B.1.

3.2.4. Observation constraints (COBS)

The value-based model assumes an observation to be given in the form of (a set of) possible values S for a particular cell c. In the best case, a sufficiently knowledgeable user will be able to come up with a single expected value for the cell; however, cases may arise where users might have only a less exact idea of a cell's expected value, which they might express in terms of a range of possible values. For example, given a spreadsheet computing the salaries of the employees of a company, the human resources staff might only know that the salary of any employee must be larger than, say, 1 000. Hence, a computed value lower than this amount for a respective cell is a symptom of a fault, without the user being able to specify the exact expected value.

That is, given an observation (c, S), the function obsConstr_{VBM} is defined as follows:
$\text{obsConstr}_{VBM}(c, S) = ((c'), \{(n) \mid n \in S\})$

3.3. Functional dependency model (FDM)

3.3.1. Idea

A functional dependency model, originally introduced by [25], *takes only two states of spreadsheet cells into account*, "correct" and "incorrect". These are modeled by the values $\top$ and $\bot$, respectively, for the variables of the spreadsheet-constraint diagnosis problem. For example, consider the spreadsheet cell `D4` comprising the formula "`= B4 + C4`". If the formula itself is correct and the values of both the cells `B4` and `C4` are correct, then the computed value for `D4` must be correct as well, whereas, e.g., the incorrectness of (exactly) one of the referenced cells `B4` or `C4` implies the incorrectness of `D4`. However, in general, the incorrectness of some cell(s) referenced in the formula $f(c)$ of another cell c does not necessarily need to entail the incorrectness of c, e.g., if `B4` is by 1 too high whereas `C4` is by 1 too low, the value of `D4` is still correct despite the incorrectness of both cells `B4` and `C4` occurring in the formula of `D4`. This phenomenon, which is frequently referred to as *coincidental correctness*, must be carefully taken into account in the functional dependency model.

The intention underlying the functional dependency model is the abstraction from the exact spreadsheet semantics and the concrete numerical values. As a consequence of this, the diagnostic reasoning operates only on the level of value (in)correctness which implies (often significantly) smaller variable domains which in turn can boost the efficiency of the constraint solving process.

3.3.2. Variable domains (DOM)

In the functional dependency model, the domain $DOM(x)$ of a variable related to a cell $c \in \Sigma$ is specified as $\{\top, \bot\}$ in case x is a variable modeling a cell value or a variable modeling a condition, and as either `true` or `false` in case x is an abnormality variable. Formally:

$$DOM(x) = \begin{cases} \{\top, \bot\} & \text{if } x = c' \text{ or } x = C_c \\ \{\mathsf{true}, \mathsf{false}\} & \text{if } x = ab_c \end{cases}$$

3.3.3. Cell constraints (CONS)

The function $\text{cConstr}_{FDM}(c)$ maps spreadsheet cells $c \in \Sigma$ to constraints in a way that the constraint specification takes into account only the (in)correctness of cell values. The function returns one or two resulting constraints per cell depending on the type and content of the cell. We describe the function output separately based on whether the input cell c is *(i)* possibly faulty ($c \in \Sigma'$) or *(ii)* known/assumed fault-free ($c \in \Sigma \setminus \Sigma'$), see Definition 12:

1. Let $c \in \Sigma'$ (*c is possibly faulty*). Then the output of $\text{cConstr}_{FDM}(c)$ for the different cases regarding the formula $f(c)$ in c is:

$\textsc{val}(+, v_1, v_2)$		v_2 = $\top$	v_2 = $\bot$
v_1	$\top$	$\{\top\}$	$\{\bot\}$
	$\bot$	$\{\bot\}$	$\{\top, \bot\}$

$\textsc{val}(-, v_1, v_2)$		v_2 = $\top$	v_2 = $\bot$
v_1	$\top$	$\{\top\}$	$\{\bot\}$
	$\bot$	$\{\bot\}$	$\{\top, \bot\}$

$\textsc{val}(*, v_1, v_2)$		v_2 = $\top$	v_2 = $\bot$
v_1	$\top$	$\{\top\}$	$\{\top, \bot\}$
	$\bot$	$\{\top, \bot\}$	$\{\top, \bot\}$

$\textsc{val}(/, v_1, v_2)$		v_2 = $\top$	v_2 = $\bot$
v_1	$\top$	$\{\top\}$	$\{\top, \bot\}$
	$\bot$	$\{\bot\}$	$\{\top, \bot\}$

Fig. 2. Definition of the function $\textsc{val}(\odot, v_1, v_2)$ for the functional dependency model for numerical operators $\odot \in \{+, -, *, /\}$. Rows indicate the value of v_1, columns the value of v_2, and table entries the resulting output of $\textsc{val}(\odot, v_1, v_2)$.

$\textsc{val}(\square, v_1, v_2)$		v_2 = $\top$	v_2 = $\bot$
v_1	$\top$	$\{\top\}$	$\{\top, \bot\}$
	$\bot$	$\{\top, \bot\}$	$\{\top, \bot\}$

Fig. 3. Definition of the function $\textsc{val}(\square, v_1, v_2)$ for the functional dependency model for relational operators ($\square \in \{<, =, >\}$. Rows indicate the value of v_1, columns the value of v_2, and table entries the resulting output of $\textsc{val}(\square, v_1, v_2)$.

- If $f(c)$ is a number n:
$$\textsc{cConstr}_{FDM}(c) = \left\{ \begin{array}{l} ((ab_c, c'), \\ \{(\mathsf{false}, \top)\} \cup \\ \{(\mathsf{true}, v) \mid v \in \{\top, \bot\}\}) \end{array} \right\}$$
(*Explanation:* If c is assumed normal, then the number it specifies must be correct, otherwise any correctness value is possible.)
- If $f(c)$ is a reference to cell $\tilde{c}$:
$$\textsc{cConstr}_{FDM}(c) = \left\{ \begin{array}{l} ((ab_c, c', \tilde{c}'), \\ \{(\mathsf{false}, v, v) | v \in DOM(\tilde{c}')\} \cup \\ \{(\mathsf{true}, v, \tilde{v}) | v \in DOM(c'), \tilde{v} \in DOM(\tilde{c}')\}) \end{array} \right\}$$
(*Explanation:* If c is assumed normal, then its (in)correctness must be the same as that of $\tilde{c}$, otherwise any combination of (in)correctness of cells c and $\tilde{c}$ is possible.)
- If $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$:
$$\textsc{cConstr}_{FDM}(c) = \left\{ \begin{array}{l} ((ab_c, c'_1, c'_2, c'), \\ \{(\mathsf{false}, v_1, v_2, v) \mid v \in \textsc{val}(\odot, v_1, v_2), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\} \cup \\ \{(\mathsf{true}, v_1, v_2, v) | v \in DOM(c'), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\}) \end{array} \right\}$$
where $\textsc{val}(\odot, v_1, v_2)$ returns the (non-empty) set of possible values $V \subseteq \{\top, \bot\}$ for cell c given the semantics of the operator $\odot$ and (in)correctness values $v_1, v_2 \in \{\top, \bot\}$ for the cells c_1, c_2 referenced in the formula $f(c)$. The function $\textsc{val}$ is defined in Fig. 2.
(*Explanation:* If c is assumed abnormal, any combination of values for the cells c_1, c_2 and c are possible. Otherwise, if c is assumed normal, then a simple (but tedious) case analysis iterating over all possible combinations of (non-)deviations of the values of cells c_1 and c_2 allows to infer the sets of possible values for cell c depicted in Fig. 2. This case analysis is given in Appendix A.1.1.)
- If $f(c)$ is of the form $\texttt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<, =, >\}$:
$\textsc{cConstr}_{FDM}(c) = \{cons_C, cons_c\}$ where
$$cons_C := \left(\begin{array}{l} (c'_1, c'_2, C_c), \\ \{(v_1, v_2, v) \mid v \in \textsc{val}(\square, v_1, v_2), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\} \end{array} \right)$$
and $\textsc{val}(\square, v_1, v_2)$ returns the (non-empty) set of possible correctness values $V \subseteq \{\top, \bot\}$ for the condition $c_1 \square c_2$ given the semantics of the operator $\square$ and (in)correctness values $v_1, v_2 \in \{\top, \bot\}$ for the cells c_1, c_2 involved in the condition. The function $\textsc{val}$ is defined in Fig. 3.
(*Explanation:* The "auxiliary" constraint $cons_C$ models the condition $c_1 \square c_2$ in the $\texttt{if}$ expression regarding the (in)correctness of its output. This information is required by the constraint $cons_c$ below, which models whether the value of the cell c is (in)correct. A simple case analysis for all operators $\square \in \{<, =, >\}$ given in Appendix A.1.2 yields the value sets depicted in Fig. 3.)

$$cons_c := \left(\begin{array}{l} (ab_c, C_c, c'_3, c'_4, c'), \\ \{(\mathsf{false}, \top, v, v, v) \mid v \in \{\top, \bot\}\} \cup \\ \{(\mathsf{false}, \top, v_3, v_4, v) \mid \\ v_3, v_4, v \in \{\top, \bot\}, v_3 \neq v_4\} \cup \\ \{(\mathsf{false}, \bot, v_3, v_4, v) \mid v_3, v_4, v \in \{\top, \bot\}\} \cup \\ \{(\mathsf{true}, b, v_3, v_4, v) \mid b, v_3, v_4, v \in \{\top, \bot\}\} \end{array} \right) \tag{2}$$

(*Explanation:* If c is assumed abnormal, both correctness values ($\top$,$\bot$) are possible for the cell c, regardless of the (in)correctness of cells c_3, c_4 and of the condition $c_1 \square c_2$. If c is assumed normal, then the case-based argumentation given in Appendix A.1.3 explains the constraint $cons_c$.)

2. Let $c \in \Sigma \setminus \Sigma'$ (*c is known/assumed correct*). Then the output of $\text{cCONSTR}_{FDM}(c)$ for the different cases regarding the formula $f(c)$ in c is analogue to the ones for the case $c \in \Sigma'$ above with ab_c set to false, with the difference that there is no ab_c variable here since the cells are known correct. The explicit constraints can be found in Appendix B.2.

3.3.4. Observation constraints (COBS)

The functional dependency model assumes an observation to be given in the form of the expression that a particular cell has a correct or an incorrect value. That is, an observation is specified by one specific cell $c \in \Sigma$ along with a non-empty set $S \subseteq \{\top, \bot\}$, usually including only a single correctness value (either $\top$ or $\bot$) for c. That is, given an observation (c, S), the function OBSCONSTR_{FDM} is defined as follows: $\text{OBSCONSTR}_{FDM}(c, S) = ((c'), \{(v) \mid v \in S\})$

3.4. Qualitative deviation model (QDM)

3.4.1. Idea

The qualitative deviation model is a refinement of the functional dependency model and an extension of the one presented in [88]. Instead of only two states, “correct” and “incorrect”, it allows to differentiate between two states of incorrectness for cells of numeric type, whose incorrect value can be either “too low” or “too high”. Boolean expressions such as conditions in `if` statements are still modeled dichotomously as in the functional dependency model since the more fine-grained characterization of incorrectness does not make sense for Boolean statements. Hence, the proposed qualitative deviation model *takes three states of spreadsheet cells into account, “correct”, “too low” and “too high”*. These are modeled by the values =, < and >, respectively, for the variables of the spreadsheet-constraint diagnosis problem.

For example, consider the spreadsheet cell `D4` comprising the formula “`= B4 + C4`”. If the formula itself is correct and the values of both the cells `B4` and `C4` are correct (=), then the computed value for `D4` must be correct (=) as well, whereas it can be derived to be too high (>) given that some cell among `B4`, `C4` is correct (=) whereas the other is too large (>). Just as in the functional dependency model, we have to deal with coincidental correctness also in case of the qualitative deviation model, e.g., in case `B4` is by 1 too high (>) whereas `C4` is by 1 too low (<), the value of cell `D4` will be correct in spite of the incorrectness of both cells referenced in its formula.

The intention underlying the qualitative deviation model is, as for the functional dependency model, the abstraction from the exact spreadsheet semantics and from the concrete numerical values, but in a slightly more nuanced manner. As a consequence of this, the diagnostic reasoning operates only on the level of the three qualitative classifications of cell values which implies (often significantly) smaller variable domains than when relying on a value-based model, and only a minor increase in terms of domain sizes compared to the functional dependency model. The goal is thus to facilitate a more refined reasoning than in the functional dependency model at a comparable computational performance.

3.4.2. Variable domains (DOM)

In the qualitative deviation model, the domain $DOM(x)$ of a variable related to a cell $c \in \Sigma$ is specified as $\{<, =, >\}$ (too low, correct, too high) in case x is a variable modeling a cell value, as $\{\top, \bot\}$ (correct, incorrect) for a variable modeling a condition, and as either true or false in case x is an abnormality variable. Formally:

$$DOM(x) = \begin{cases} \{<, =, >\} & \text{if } x = c' \\ \{\top, \bot\} & \text{if } x = C_c \\ \{\mathsf{true}, \mathsf{false}\} & \text{if } x = ab_c \end{cases}$$

3.4.3. Cell constraints (CONS)

The function $\text{cCONSTR}_{QDM}(c)$ maps spreadsheet cells $c \in \Sigma$ to constraints in a way that the constraint specification takes into account only whether the cell values are correct (=), too high (>), or too low (<). The function returns one or two resulting constraints per cell depending on the type and content of the cell. We describe the function output separately based on whether the input cell c is *(i)* possibly faulty ($c \in \Sigma'$) or *(ii)* known/assumed fault-free ($c \in \Sigma \setminus \Sigma'$), see Definition 12:

1. Let $c \in \Sigma'$ (*c is possibly faulty*). Then the output of $\text{cCONSTR}_{QDM}(c)$ for the different cases regarding the formula $f(c)$ in c is:

$\text{VAL}(+, v_1, v_2)$		v_2 $<$	$=$	$>$
v_1	$<$	$\{<\}$	$\{<\}$	$\{<,=,>\}$
	$=$	$\{<\}$	$\{=\}$	$\{>\}$
	$>$	$\{<,=,>\}$	$\{>\}$	$\{>\}$

$\text{VAL}(*, v_1, v_2)$		v_2 $<$	$=$	$>$
v_1	$<$	$\{<,=,>\}$	$\{<,=,>\}$	$\{<,=,>\}$
	$=$	$\{<,=,>\}$	$\{=\}$	$\{<,=,>\}$
	$>$	$\{<,=,>\}$	$\{<,=,>\}$	$\{<,=,>\}$

$\text{VAL}(-, v_1, v_2)$		v_2 $<$	$=$	$>$
v_1	$<$	$\{<,=,>\}$	$\{<\}$	$\{<\}$
	$=$	$\{>\}$	$\{=\}$	$\{<\}$
	$>$	$\{>\}$	$\{>\}$	$\{<,=,>\}$

$\text{VAL}(/, v_1, v_2)$		v_2 $<$	$=$	$>$
v_1	$<$	$\{<,=,>\}$	$\{<,>\}$	$\{<,=,>\}$
	$=$	$\{<,=,>\}$	$\{=\}$	$\{<,=,>\}$
	$>$	$\{<,=,>\}$	$\{<,>\}$	$\{<,=,>\}$

Fig. 4. Definition of the function $\text{VAL}(\odot, v_1, v_2)$ for the qualitative deviation model for numerical operators $\odot \in \{+, -, *, /\}$. Table rows indicate the value of v_1, table columns the value of v_2, and table entries the resulting output of $\text{VAL}(\odot, v_1, v_2)$.

$op \in \{<,=,>\}$: $\text{VAL}(op, v_1, v_2)$		v_2 $<$	$=$	$>$
v_1	$<$	$\{\top, \bot\}$	$\{\top, \bot\}$	$\{\top, \bot\}$
	$=$	$\{\top, \bot\}$	$\{\top\}$	$\{\top, \bot\}$
	$>$	$\{\top, \bot\}$	$\{\top, \bot\}$	$\{\top, \bot\}$

Fig. 5. Definition of the function $\text{VAL}(\square, v_1, v_2)$ for the qualitative deviation model for relational operators ($\square \in \{<,=,>\}$. Rows indicate the value of v_1, columns the value of v_2, and table entries the resulting output of $\text{VAL}(\square, v_1, v_2)$.

- If $f(c)$ is a number n:
$$\text{CONSTR}_{QDM}(c) = \left\{ \begin{array}{l} ((ab_c, c'), \\ \{(\mathsf{false}, =)\} \cup \\ \{(\mathsf{true}, v) \mid v \in \{<,=,>\}\}) \end{array} \right\}$$
(*Explanation:* If c is assumed normal, then the number it specifies must be correct, else correctness or any deviation are possible.)
- If $f(c)$ is a reference to cell $\tilde{c}$:
$$\text{CONSTR}_{QDM}(c) = \left\{ \begin{array}{l} ((ab_c, c', \tilde{c}'), \\ \{(\mathsf{false}, v, v) | v \in DOM(\tilde{c}')\} \cup \\ \{(\mathsf{true}, v, \tilde{v}) | v \in DOM(c'), \tilde{v} \in DOM(\tilde{c}')\}) \end{array} \right\}$$
(*Explanation:* If c is assumed normal, then its deviation must be the same as that of $\tilde{c}$, otherwise any combination of cell value deviations for cells c and $\tilde{c}$ is possible.)
- If $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$:
$$\text{CONSTR}_{QDM}(c) = \left\{ \begin{array}{l} ((ab_c, c'_1, c'_2, c'), \\ \{(\mathsf{false}, v_1, v_2, v) \mid v \in \text{VAL}(\odot, v_1, v_2), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\} \cup \\ \{(\mathsf{true}, v_1, v_2, v) \mid v \in DOM(c'), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\}) \end{array} \right\}$$
where $\text{VAL}(\odot, v_1, v_2)$ returns the (non-empty) set of possible values $V \subseteq \{<,=,>\}$ for cell c given the semantics of the operator $\odot$ and deviation values $v_1, v_2 \in \{<,=,>\}$ for the cells c_1, c_2 referenced in the formula $f(c)$. The function VAL is defined in Fig. 4.
(*Explanation:* If c is assumed abnormal, any combination of values for the cells c_1, c_2 and c are possible. Otherwise, if c is assumed normal, then a simple (but tedious) case analysis iterating over all possible combinations of (non-)deviations of the values of cells c_1 and c_2 allows to infer the sets of possible values for cell c depicted in Fig. 4. This case analysis is given in Appendix A.2.1.)
- If $f(c)$ is of the form $\texttt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<,=,>\}$:
$\text{CONSTR}_{QDM}(c) = \{cons_C, cons_c\}$ where
$$cons_C := \left(\begin{array}{l} (c'_1, c'_2, C_c), \\ \{(v_1, v_2, v) \mid v \in \text{VAL}(\square, v_1, v_2), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\} \end{array} \right)$$
and $\text{VAL}(\square, v_1, v_2)$ returns the (non-empty) set of possible values $V \subseteq \{\top, \bot\}$ for the condition $c_1 \square c_2$ given the semantics of the operator $\square$ and deviation values $v_1, v_2 \in \{<,=,>\}$ for the cells c_1, c_2 involved in the condition, where $\top/\bot$ means that the truth value of the condition is correct/incorrect. The function VAL is defined in Fig. 5.
(*Explanation:* The "auxiliary" constraint $cons_C$ models the condition $c_1 \square c_2$ in the $\texttt{if}$ expression regarding the (in)correctness of its output. This information is required by the constraint $cons_c$ below, which models the deviation of the cell c. A simple case analysis for all operators $\square \in \{<,=,>\}$ given in Appendix A.2.2 yields the value sets depicted in Fig. 5.)

$$cons_c := \left(\begin{array}{l} (ab_c, C_c, c'_3, c'_4, c'), \\ \{(\mathsf{false}, \top, v_3, v_4, v) \mid v_3, v_4 \in \{<,=,>\}, \\ v = v_3 \vee v = v_4\} \cup \\ \{(\mathsf{false}, \bot, v_3, v_4, v) \mid v_3, v_4, v \in \{<,=,>\}\} \cup \\ \{(\mathsf{true}, b, v_3, v_4, v) \mid v_3, v_4, v \in \{<,=,>\}, \\ b \in \{\top, \bot\}\} \end{array} \right) \tag{3}$$

(*Explanation:* If c is assumed abnormal, both correctness values $\top$ and $\bot$ are possible for the cell c, regardless of the deviation of the cells c_3, c_4 and of the condition $c_1 \square c_2$. If c is assumed normal, then the case-based argumentation given in Appendix A.2.3 explains the constraint $cons_c$.)

2. Let $c \in \Sigma \setminus \Sigma'$ (*c is known/assumed correct*). Then the output of $\textsc{cConstr}_{QDM}(c)$ for the different cases regarding the formula $f(c)$ in c is analogue to the ones for the case $c \in \Sigma'$ above with ab_c set to false, with the difference that there is no ab_c variable here since the cells are known correct. The explicit constraint can be found in Appendix B.3.

3.4.4. Observation constraints (COBS)

The qualitative deviation model assumes an observation to be given in the form of a classification that either the value of a particular cell is correct (class $=$) or of how the value of a particular cell deviates from its expected value, with possible classes $<$ (too low) or $>$ (too high). That is, an observation is defined by a cell $c \in \Sigma$ along with a non-empty set $S \subseteq \{<,=,>\}$ that expresses the possible deviations of the value of cell c. In many cases, an observation will specify either $S = \{<\}$ or $S = \{>\}$, corresponding to the assertion of some *particular* type of deviation for the respective cell. However, also combinations of multiple values are possible, such as $S = \{<,=\}$, expressing that the value of the cell, if incorrect, can only be too low. Also, if a user does know that a cell comprises a wrong value, but cannot discern whether the value is too high or too low, then they would specify S to be equal to $\{<,>\}$, meaning that "c is incorrect", which corresponds to the specification $S = \{\bot\}$ in the functional dependency model. That is, given an observation (c, S), the function $\textsc{obsConstr}_{QDM}$ is defined as follows: $\textsc{obsConstr}_{QDM}(c, S) = \{(c'), \{(v) \mid v \in S\}\}$

4. Automated model extraction

In this section, we describe the conversion of a spreadsheet into a spreadsheet-constraint diagnosis problem. This conversion, shown by Algorithm 2, allows for automated spreadsheet debugging based on any of the three model types—the value-based model (*VBM*), the functional dependency model (*FDM*), or the qualitative deviation model (*QDM*)—discussed in Sec. 3.

The algorithm gets as inputs a spreadsheet (set of cells) Σ, a set of possibly faulty cells $\Sigma' \subseteq \Sigma$ (where all cells $\Sigma \setminus \Sigma'$ are assumed fault-free), a model type $MT \in \{VBM, FDM, QDM\}$, and a set of observations O, each element of which is of the form (c, S) where c is some cell in Σ' and S a set of values covering the value as per MT of c. It finally outputs a spreadsheet-constraint diagnosis problem $P = (CS, COMP)$ for the spreadsheet Σ and the observations O based on the model type MT.

The workflow is as follows: First, the components *COMP* are instantiated to the set of possibly faulty cells Σ'. Second, the variables (*VARS*) and their domains (function *DOM*) are instantiated, according to the descriptions in Secs. 3.2–3.4. Third, a set of constraints is extracted by the function $\textsc{cConstr}_{MT}$ for each spreadsheet cell in Σ based on the given model type MT, and added to the set of constraints *CONS*. Fourth, a corresponding constraint is created for each observation in O by means of the function $\textsc{obsConstr}_{MT}$ based on the given model type MT, and added to the set of constraints *COBS*. The functions $\textsc{cConstr}_{MT}$ and $\textsc{obsConstr}_{MT}$ for the three model types $MT \in \{VBM, FDM, QDM\}$ are detailed in Secs. 3.2–3.4; note, the former returns a set of constraints whereas the latter returns a single constraint. Finally, the tuple $((VARS, DOM, CONS \cup COBS), COMP)$, specifying the generated spreadsheet-constraint diagnosis problem for the given inputs, is returned.

The algorithm terminates because all executed loops involve iterating through Σ or O and $|\Sigma|$ and $|O|$ are finite. The time complexity is obtained by adding the time for the three for-loops. In the first, a maximum of $6 \in O(1)$ steps (variable as well as domain instantiation for the three variables c', C_c, and ab_c) are required per cell $c \in \Sigma$. The time complexity is thus in $O(|\Sigma|)$. The second for-loop involves the generation of at most two constraints per $c \in \Sigma$. The time for the creation of a constraint depends on the number of value combinations the variables involved in the constraint might attain, i.e., on the number of tuples in the constraint. For the *FDM* and *QDM* model types, the constraint comprising the most tuples each is $cons_c$ in Eq. (2) and $cons_c$ in Eq. (3), respectively, where the number of tuples is in $O(2+4+8+16) = O(30) = O(1)$ in case of the *FDM* and in $O(15+27+54) = O(96) = O(1)$ in case of the *QDM* (cf. also Fig. C.17 in Appendix C, which enumerates these tuples explicitly). For the *VBM*, the number of tuples in the largest constraint, given by $cons_c$ on page 11, is in $O(|D|^3)$ where $|D|$ is the size of the largest domain D of a variable occurring in the constraint (and $|D|$ is finite, as discussed in Sec. 3.2). Consequently, the second for-loop has a time complexity in $O(|\Sigma|)$ for $MT \in \{FDM, QDM\}$, and in $O(|\Sigma||D|^3)$ for $MT = VBM$. The third for-loop performs one iteration through all observations in O and the number of tuples per constraint is in $O(|S|)$, which is in $O(1)$ for all three model types. Thus, the complexity of this for-loop is in $O(|O|)$. So, the overall time complexity of Algorithm 2 is in $O(|\Sigma| + |O|)$ for $MT \in \{FDM, QDM\}$ and in $O(|\Sigma||D|^3 + |O|)$ for $MT = VBM$.[4]

[4] Whereas we rely on a representation of constraint systems based on explicit constraints (enumeration of tuples that satisfy the constraint) in this work for presentation purposes, one might use more compact implicit constraint representations in practice which are well supported by modern constraint solvers. Such implicit representations will imply a similar complexity of Algorithm 2 for $MT = VBM$ as for $MT \in \{FDM, QDM\}$.

Algorithm 2 ExtractModel(Σ, Σ', MT, O).

Input: A spreadsheet Σ, a set of possibly faulty cells $\Sigma' \subseteq \Sigma$, a model type $MT \in \{VBM, FDM, QDM\}$, and a set of observations O
Output: A spreadsheet-constraint diagnosis problem $P = (CS, COMP)$ for Σ and O

$COMP = \Sigma'$ # instantiation of components
$VARS, CONS, COBS = \{\}$
for $c \in \Sigma$ **do** # instantiation of variables and domains
  $VARS \leftarrow VARS \cup \{c'\}$
  if $MT = VBM$ **then**
    $DOM(c') = \mathbb{R}$
  if $MT = FDM$ **then**
    $DOM(c') = \{\mathsf{T}, \bot\}$
  if $MT = QDM$ **then**
    $DOM(c') = \{<, =, >\}$
  if $\text{TYPE}(c) = \texttt{if}$ **then**
    $VARS \leftarrow VARS \cup \{C_c\}$
    if $MT = VBM$ **then**
      $DOM(C_c) = \{\mathsf{true}, \mathsf{false}\}$
    if $MT \in \{FDM, QDM\}$ **then**
      $DOM(C_c) = \{\mathsf{T}, \bot\}$
  if $c \in COMP$ **then**
    $VARS \leftarrow VARS \cup \{ab_c\}$
    $DOM(ab_c) = \{\mathsf{true}, \mathsf{false}\}$
for $c \in \Sigma$ **do** # specification of constraints for spreadsheet cells
  $CONS \leftarrow CONS \cup \text{CCONSTR}_{MT}(c)$
for $(c, S) \in O$ **do** # specification of constraints for observations
  $COBS \leftarrow COBS \cup \{\text{OBSCONSTR}_{MT}(c, S)\}$
return $((VARS, DOM, CONS \cup COBS), COMP)$

5. Theoretical analysis of model types

In this section, we theoretically analyze the three model types discussed in Sec. 3 concerning their degree of abstractness, inferential power and diagnostic accuracy. We find that the former two notions have implications on the latter. That is, we show relationships between pairs of model types w.r.t. abstractness and inferential power, respectively, which in turn allow us to derive relationships between the model types regarding diagnostic accuracy.[5]

First, we define what we understand by model abstractness, based on the characterization in [88]. Given two spreadsheet-constraint diagnosis problems modeling one and the same spreadsheet, we call the model P_2 more abstract than the model P_1 if and only if P_1 can be "translated" to P_2 by a single function that maps elements from the domain of each variable in P_1 to the same variable's domain in P_2. Intuitively, this means that P_2 can be obtained from P_1 by a simple renaming of domain values. Formally:

Definition 14 *(Model abstractness).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_1 = (CS_1, COMP)$ as well as $P_2 = (CS_2, COMP)$ be two spreadsheet-constraint diagnosis problems modeling the spreadsheet Σ, where $CS_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $CS_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. Then, we call P_2 a *more abstract* model for Σ than P_1 (denoted by $P_1 \prec_{abs} P_2$) iff

- for each variable $x \in VARS$ there is a function $h_x : DOM_1(x) \rightarrow DOM_2(x)$ such that
  - for all $x, x' \in VARS \setminus \{ab_c \mid c \in COMP\}$ and for all $v \in DOM_1(x), v' \in DOM_1(x')$ it holds that $v = v' \rightarrow h_x(v) = h_{x'}(v')$ (i.e., one and the same value is mapped to the same image by all functions h_x where x is not an ab_c variable),
  - $h_x(v) = v$ for all $x \in \{ab_c \mid c \in COMP\}$ (i.e., the truth value of ab_c variables remains unaffected by the application of the function h_{ab_c}), and
- for each pair of constraints (c_1, c_2) where $c_1 = (scope(c_1), tl(c_1)) \in CONS_1 \cup COBS_1$ and $c_2 = (scope(c_2), tl(c_2)) \in CONS_2 \cup COBS_2$ with $scope(c_1) = scope(c_2) = (x_1, \ldots, x_k)$ it holds that $h(c_1) = c_2$ where h is defined as
  - $h(c_1) = h((scope(c_1), tl(c_1))) := (scope(c_1), h(tl(c_1)))$ and
  - $h(tl(c_1)) := \{(h_{x_1}(v_1), \ldots, h_{x_k}(v_k)) \mid (v_1, \ldots, v_k) \in tl(c_1)\}$.

Remark 4. Due to the way a model (spreadsheet-constraint diagnosis problem) is extracted from a given spreadsheet (see Secs. 3 and 4), it holds that, *(1)* any two constraints in $CONS_1 \cup COBS_1$ have a different scope, and *(2)* for each constraint $c_1 \in CONS_1 \cup COBS_1$, there is one and only one constraint $c_2 \in CONS_2 \cup COBS_2$ with the same scope. The reason is that, per cell $c \in \Sigma$, and irrespective of the used model type, there is exactly one constraint involving the variable c' (and possibly one additional constraint involving the variable C_c), and if a cell c is possibly faulty (associated constraint involves the ab_c variable) in one model, it is possibly faulty in the other model as well (Σ' is the same for both CS_1 and CS_2). As a consequence, there is a bijective function $\text{SAMESCOPE} : (CONS_1 \cup COBS_1) \rightarrow (CONS_2 \cup COBS_2)$ which maps each $c_1 \in CONS_1 \cup COBS_1$ to $\text{SAMESCOPE}(c_1) = c_2$ where c_2 is the unique constraint in $CONS_2 \cup COBS_2$

[5] Please see Appendix C for the proofs of all theorems and corollaries in this section.

with $scope(c_1) = scope(c_2)$. Hence, in particular, the number of constraint pairs relevant to the last but one bullet point of Definition 14 equals the number of constraints in CS_1 (and CS_2).

The function h intuitively maps each value v_i from the domain $DOM_1(x_i)$ occurring in a tuple $(v_1, \ldots, v_k)$ of possible values of constraint c_1 to a corresponding value $h_{x_i}(v_i)$ from the domain $DOM_2(x_i)$. For example, if $DOM_1(x_i) = \{<, =, >\}$ (cf. *QDM*) and $DOM_2(x_i) = \{\top, \bot\}$ (cf. *FDM*), a reasonable mapping would be to assign $\bot$ to each value from $DOM_1(x_i)$ modeling a deviation (i.e., to $<$ and $>$) and to assign $\top$ to the value indicating no deviation (i.e., to $=$). This would result in the following function: $h_{x_i}(<) = \bot$, $h_{x_i}(>) = \bot$, $h_{x_i}(=) = \top$. □

The definition of diagnostic accuracy is based on the conciseness of the solution space of diagnoses for a given model of a spreadsheet. We next introduce a formal notation for the set of all diagnoses (as per Definition 13) for a spreadsheet-constraint diagnosis problem:

Definition 15 *(Set of all diagnoses w.r.t. a model).* Let $P = (CS, COMP)$ be a spreadsheet-constraint diagnosis problem. Then, we refer to the set of all diagnoses for P (as per Definition 13) as $\mathbf{AllD}_P$.

Using this notation, the next definition characterizes one model of a spreadsheet as less/equally accurate than another model if and only if the former gives rise to more/equally many diagnoses than the latter.

Definition 16 *(Diagnostic accuracy).* Let $P_1 = (CS_1, COMP)$ and $P_2 = (CS_2, COMP)$ be two spreadsheet-constraint diagnosis problems modeling the same spreadsheet Σ. Then, we call

- P_1 a *less accurate* model for Σ than P_2 (denoted by $P_1 \prec_{acc} P_2$) iff $\mathbf{AllD}_{P_1} \supset \mathbf{AllD}_{P_2}$,
- P_1 an *equally accurate* model for Σ as P_2 (denoted by $P_1 \equiv_{acc} P_2$) iff $\mathbf{AllD}_{P_1} = \mathbf{AllD}_{P_2}$,
- P_1 a *less or equally accurate* model for Σ as P_2 (denoted by $P_1 \preceq_{acc} P_2$) iff $\mathbf{AllD}_{P_1} \supseteq \mathbf{AllD}_{P_2}$.

As we will later see, a spreadsheet model of any of the three types leads to a set of diagnoses comprising one that pinpoints actually faulty spreadsheet cells. Thus, a higher model accuracy intuitively leads to less effort in finding such a "correct" diagnosis among all diagnoses generated by the model. As discerning "correct" from "spurious" solutions among a large set of diagnoses can be tedious, this task is often assisted, e.g., by the use of sequential diagnosis methods [16,71] or by exploiting a ranking of the diagnoses (e.g., based on cardinality or likelihood) [63], where all these approaches benefit from a smaller number of relevant diagnoses to be investigated.

Now, having at hand the formal notions of model abstractness and diagnostic accuracy, we find that a higher degree of abstractness implies a lower or equal diagnostic accuracy:

Theorem 1 *(Connection between model abstractness and accuracy). Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_1 = (CS_1, COMP)$ as well as $P_2 = (CS_2, COMP)$ be two spreadsheet-constraint diagnosis problems modeling the spreadsheet Σ, where $CS_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $CS_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. Then, $P_2 \prec_{abs} P_1$ implies $P_1 \preceq_{acc} P_2$, i.e., if P_1 is more abstract than P_2, then P_1 is less or equally accurate as P_2.*

Proof. *(Idea)* We prove this theorem by contradiction, assuming that *(1)* $P_2 \prec_{abs} P_1$ and *(2)* $P_1 \preceq_{acc} P_2$ do not hold. By Definition 16, (2) is equivalent to $\mathbf{AllD}_{P_1} \not\supseteq \mathbf{AllD}_{P_2}$. We then assume Δ to be an arbitrary diagnosis for P_2, which is subsequently shown to be a diagnosis for P_1 as well using the mapping that holds between P_2 and P_1 as per Definition 14 due to (1). This is a contradiction and concludes the proof. (See the full proof in Appendix C.1.) □

Theorem 2 below investigates the relationships between the three model types regarding their abstractness. This analysis aims to leverage Theorem 1 to deduce corresponding relationships w.r.t. diagnostic accuracy. Indeed, it turns out that the *FDM* is more abstract than the *QDM*, and that no further abstractness relationships between pairs of model types can be derived.

Before stating the theorem, we formalize the notion of consistent observations. Intuitively, in order for two model types to be reasonably comparable, we will have to assume that the observations provided by the spreadsheet user(s) are compatible for both analyzed model types. That is, we assume that, whenever an observation for some cell is given for one model, then a corresponding observation for the same cell is given for the other model as well. Moreover, we stipulate that two such corresponding observations are in line regarding their content. For instance, if an observation for the *QDM* states that the value of some cell is too low, we postulate that there is an observation for the *FDM* which declares the value of the same cell as wrong. Definition 19 below phrases this idea in formal terms; but before, Definitions 17 and 18 as well as Lemma 1 discuss some notation relevant to Definition 19:

Definition 17 *(Spreadsheet constraint model under fault-free assumption).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, and let $((VARS, DOM, CONS), COMP)$ be the spreadsheet-constraint diagnosis problem output by Algorithm 2 given the inputs Σ, Σ' as well as $MT := VBM$ and $O := \emptyset$. We then define $CS^*_{VBM}(\Sigma, \Sigma') := (VARS, DOM, CONS \cup \{((ab_c), \{(\mathsf{false})\}) \mid c \in COMP\})$.

Simply put, $CS^*_{VBM}(\Sigma, \Sigma')$ is the description of a spreadsheet Σ by means of the value-based model without taking any observations into account, under the additional assumption that all possibly faulty cells in Σ' are correct.

Lemma 1. *There is a single valid value assignment for the constraint system* $CS^*_{VBM}(\Sigma, \Sigma')$.

Definition 18 *(Spreadsheet cell values under fault-free assumption).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, and $CS^*_{VBM}(\Sigma, \Sigma')$ be as specified in Definition 17. Then, for any cell $c \in \Sigma$, the unique value assigned to the variable c' (and, if applicable, C_c) by the single valid value assignment for $CS^*_{VBM}(\Sigma, \Sigma')$ (cf. Lemma 1) is referred to as c^* (and C^*_c). We call c^* (and C^*_c) the *cell value(s) of cell c under the fault-free assumption*.

Definition 19 *(Consistent observations).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_{MT} = (CS_{MT}, COMP)$ with $CS_{MT} = (VARS, DOM_{MT}, CONS_{MT} \cup COBS_{MT})$ for $MT \in \{VBM, QDM, FDM\}$ be three spreadsheet-constraint diagnosis problems modeling Σ where P_{MT} is extracted from Σ using the model type MT. Further, let c^* (and C^*_c) be the cell value(s) of cells $c \in \Sigma$ under the fault-free assumption as per Definition 18.

Then, we say that the observations for spreadsheet-constraint diagnosis problems

- P_{QDM} and P_{FDM} are *consistent* iff Statement 1 below holds,
- P_{VBM} and P_{FDM} are *consistent* iff Statement 2 below holds, and
- P_{VBM} and P_{QDM} are *consistent* iff Statement 3 below holds.

We moreover say that the observations for all three spreadsheet-constraint diagnosis problems P_{VBM}, P_{QDM} and P_{FDM} are *consistent* iff all Statements 1–3 below hold:

1. An observation (c, S_{QDM}) for a cell $c \in \Sigma$ is specified for P_{QDM} iff an observation (c, S_{FDM}) is specified for P_{FDM} where $\bot \in S_{FDM}$ iff $> \in S_{QDM} \vee < \in S_{QDM}$ and $\top \in S_{FDM}$ iff $= \in S_{QDM}$.
2. An observation (c, S_{VBM}) for a cell $c \in \Sigma$ is specified for P_{VBM} iff an observation (c, S_{FDM}) is specified for P_{FDM} where $\bot \in S_{FDM}$ iff some $\hat{c} \in S_{VBM}$ where $\hat{c} \neq c^*$ and $\top \in S_{FDM}$ iff $c^* \in S_{VBM}$.
3. An observation (c, S_{VBM}) for a cell $c \in \Sigma$ is specified for P_{VBM} iff an observation (c, S_{QDM}) is specified for P_{QDM} where $< \in S_{QDM}$ iff some $\hat{c} \in S_{VBM}$ where $\hat{c} < c^*$, $> \in S_{QDM}$ iff some $\hat{c} \in S_{VBM}$ where $\hat{c} > c^*$, and $= \in S_{QDM}$ iff $c^* \in S_{VBM}$.

Theorem 2 *(Abstractness relationships between model types). Let* Σ *be a spreadsheet,* $\Sigma' \subseteq \Sigma$ *be the set of possibly faulty cells,* $COMP = \Sigma'$*, and* $P_1 = (CS_1, COMP)$ *and* $P_2 = (CS_2, COMP)$ *be two spreadsheet-constraint diagnosis problems modeling* Σ *where* $CS_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ *and* $CS_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$*. Further, let* P_i *be extracted from* Σ *using the model type* $MT_i \in \{QDM, FDM, VBM\}$ *for* $i \in \{1, 2\}$*, and let the observations for both models* P_1 *and* P_2 *be consistent (cf. Definition 19). Then, the following propositions hold regarding the relation* $\prec_{abs}$*:*

1. $P_1 \prec_{abs} P_2$ *if* $(MT_1, MT_2) = (QDM, FDM)$.
2. *In general,* $P_1 \not\prec_{abs} P_2$ *for all pairs* $(MT_1, MT_2) \neq (QDM, FDM)$ *with* $MT_1 \neq MT_2$.

Proof. *(Idea)* To prove *Statement 1*, we define a function h_{x_i} for all variables $x_i \in VARS$, as defined in Remark 4, that maps elements from all domains of variables in P_1 to domains of the associated variables in P_2, and subsequently verify that the conditions of Definition 14 are satisfied.

Statement 2 is shown by providing reasons why there cannot be a function from variable domains in P_1 to domains of associated variables in P_2 that satisfies Definition 14, for all pairs $(MT_1, MT_2) \in \{(FDM, QDM), (FDM, VBM), (QDM, VBM), (VBM, FDM), (VBM, QDM)\}$. For the first three pairs, in order to define such a function, we observe that smaller variable domains would need to be mapped to larger variable domains, which generally does not allow to transform the tuple lists of the constraints in P_1 to those of the constraints in P_2, i.e., such a function cannot exist. For the last two pairs, we assume such a function exists, derive necessary properties of this function, and then show that these properties lead to a contradiction, which proves that such a function cannot exist. (See the full proof in Appendix C.3.) □

The next corollary combines the general relationship between model abstractness and diagnostic accuracy (Theorem 1) with the specific result that the *FDM* is more abstract than the *QDM* (Theorem 2, Statement 1) to conclude that the *QDM* is at least as accurate as the *FDM*:

Corollary 1 *(Accuracy relationship between FDM and QDM). Let* Σ *be a spreadsheet,* $\Sigma' \subseteq \Sigma$ *be the set of possibly faulty cells,* $COMP = \Sigma'$*, and* $P_1 = (CS_1, COMP)$ *and* $P_2 = (CS_2, COMP)$ *be two spreadsheet-constraint diagnosis problems modeling* Σ*, where* P_1 *and* P_2 *are extracted from* Σ *according to the model type FDM and QDM, respectively. Further, let the observations for both models* P_1 *and* P_2 *be consistent (cf. Definition 19). Then,* $P_1 \preceq_{acc} P_2$.

As the concept of model abstractness is not helpful in interrelating the *VBM* with the other two models, as evidenced by Statement 2 of Theorem 2, we conduct a further analysis of the model types regarding their inferential power. We can infer a statement α from a

constraint system $CS = (VARS, DOM, CONS)$ if and only if α is true in every solution (valid value assignment, cf. Definition 9) of CS; in this case, we also say that CS *entails* α, written as $CS \models \alpha$. Note that $CS \models \alpha$ if and only if $CS_{\neg\alpha} := (VARS, DOM, CONS \cup \{\neg\alpha\})$ is unsatisfiable. Given two spreadsheet-constraint diagnosis problems P_1, P_2 modeling one and the same spreadsheet, we say that the model P_2 has a higher inferential power than the model P_1 if and only if, for one and the same assumption about the (ab)normality of all spreadsheet cells (cf. $CONS_\Delta$ in Definition 13), the constraint system of P_1 has a solution whenever the constraint system of P_2 has one. Intuitively, P_2 allows "more inferences" as, whenever the constraint system of P_1 is unsatisfiable, the one of P_2 is unsatisfiable as well. Regarding the task of diagnosis computation, this means that, whenever Δ is a diagnosis for P_2, then Δ is a diagnosis for P_1 as well. Formally:

Definition 20 *(Inferential power of a model).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_1 = (CS_1, COMP)$ as well as $P_2 = (CS_2, COMP)$ be two spreadsheet-constraint diagnosis problems modeling the spreadsheet Σ, where $CS_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $CS_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. Further, let $\Delta \subseteq COMP$, and let $CONS_\Delta$ denote the set of constraints $\{((ab_c), \{(\mathsf{true})\}) \mid c \in \Delta\} \cup \{((ab_c), \{(\mathsf{false})\}) \mid c \in COMP \setminus \Delta\}$ specifying the assumption that all $c \in \Delta$ are abnormal and all $c \in COMP \setminus \Delta$ are normal, and let $CS_i^\Delta := (VARS, DOM_i, CONS_i \cup COBS_i \cup CONS_\Delta)$ for $i \in \{1, 2\}$. Then, we say that model P_2 has a *higher or equal inferential power* than P_1 (denoted by $P_1 \preceq_{inf} P_2$) iff CS_1^Δ is satisfiable whenever CS_2^Δ is satisfiable (or, equivalently: CS_2^Δ is unsatisfiable whenever CS_1^Δ is unsatisfiable).

The next theorem testifies that a higher or equal inferential power implies a higher or equal diagnostic accuracy:

Theorem 3 *(Connection between inferential power and accuracy). Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_1 = (CS_1, COMP)$ as well as $P_2 = (CS_2, COMP)$ be two spreadsheet-constraint diagnosis problems modeling the spreadsheet Σ. Then, $P_1 \preceq_{inf} P_2$ implies $P_1 \preceq_{acc} P_2$, i.e., if P_2 has a higher or equal inferential power as P_1, then P_1 is less or equally accurate as P_2.*

Proof. *(Idea)* The theorem follows directly from Definitions 13 and 20 (see Appendix C.5). □

We next show that the *VBM* has a higher or equal inferential power as both the *FDM* and the *QDM*:

Theorem 4 *(Inferential power relationships between VBM and FDM/QDM). Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_1 = (CS_1, COMP)$ and $P_2 = (CS_2, COMP)$ be two spreadsheet-constraint diagnosis problems modeling Σ where $CS_1 = (VARS, DOM_1, CONS_1 \cup COBS_1)$ and $CS_2 = (VARS, DOM_2, CONS_2 \cup COBS_2)$. Let P_1 be extracted from Σ using some model type $MT_1 \in \{QDM, FDM\}$, and P_2 be extracted from Σ using the model type $MT_2 = VBM$. Moreover, let $CS^*_{VBM}(\Sigma, \Sigma')$ as well as c^* and C^*_c for cells $c \in \Sigma$ be defined as in Definitions 17 and 18 and let the observations for all models P_i be consistent (cf. Definition 19) regardless of the used model type $\{QDM, FDM, VBM\}$. Then, $P_1 \preceq_{inf} P_2$.*

Proof. *(Idea)* First, we assume an arbitrary $\Delta \in COMP$ and an arbitrary valid value assignment M_{VBM} for CS_2. The proof considers the two cases $MT_1 \in \{QDM, FDM\}$ in turn. For each case, we specify a value assignment M_1 for CS_1 based on M_{VBM}, where all components in Δ are assumed to be abnormal and all components in $COMP \setminus \Delta$ are assumed to be normal for both CS_1 and CS_2. We then show by contradiction that M_1 is a valid value assignment for CS_1. Hence, whenever CS_2 is satisfiable given Δ, CS_1 is also satisfiable given Δ, which concludes the proof by Definition 20. (See the full proof in Appendix C.6.) □

The next corollary uses Theorems 3 and 4 to establish accuracy relationships between the *VBM* and the other two models types. Specifically, we obtain that the *VBM* is at least as accurate as each of the other model types, i.e., as the *FDM* and as the *QDM*.

Corollary 2 *(Accuracy relationships between VBM and FDM/QDM). Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, $COMP = \Sigma'$, and $P_{MT} = (CS_{MT}, COMP)$ be a spreadsheet-constraint diagnosis problem modeling Σ and extracted from Σ using the model type $MT \in \{QDM, FDM, VBM\}$. Moreover, let the observations for all models P_{MT} be consistent (cf. Definition 19) regardless of the used model type $MT \in \{QDM, FDM, VBM\}$. Then, $P_{FDM} \preceq_{acc} P_{VBM}$ and $P_{QDM} \preceq_{acc} P_{VBM}$.*

Now, putting it all together, we learn that the *VBM* is at least as accurate as the *QDM*, which in turn is at least as accurate as the *FDM*:

Corollary 3 *(Accuracy relationships between all model types). Let the preconditions of Corollary 2 hold. Then, $P_{FDM} \preceq_{acc} P_{QDM} \preceq_{acc} P_{VBM}$.*

Given a spreadsheet-constraint diagnosis problem P_{VBM} modeled by the "exact" *VBM*, every (minimal) diagnosis for the problem constitutes a potential (minimal) explanation for the spreadsheet's observed faultiness. In fault localization, however, the ultimate goal is not only to compute diagnoses, but to find (what we will call) a correct diagnosis, all of whose components are actually faulty[6]

[6] In general, we cannot be sure that the set of actually faulty cells in the spreadsheet corresponds to a *minimal* diagnosis of the spreadsheet-constraint diagnosis problem at hand; we might just not (yet) have enough observations to this end. Hence, we refer by a correct diagnosis to one which pinpoints *only* actually faulty cells, but *not necessarily all* of them.

and need to be repaired or replaced. Every diagnosis in $\mathbf{AllD}_{P_{VBM}}$ which is not a correct diagnosis thus contains (i.e., assumes faulty) at least one correct cell, which is why we will refer to these as false-positive spurious diagnoses. When contrasting the set of diagnoses $\mathbf{AllD}_{P_{VBM}}$ with the set of diagnoses $\mathbf{AllD}_{P'}$ identified for a problem P' modeled by an abstract model in $\{QDM, FDM\}$, another class of component sets of interest is given by $\mathbf{AllD}_{P'} \setminus \mathbf{AllD}_{P_{VBM}}$. This collection comprises all component sets which are diagnoses for the abstract model, but not for the exact model. Each of these component sets does not contain (i.e., assumes correct) at least one faulty cell, which is why we term them false-negative spurious diagnoses:

Definition 21 *(Correct/Spurious diagnosis).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, and P_{VBM} be a spreadsheet-constraint diagnosis problem modeling Σ by means of the model type *VBM*. Let further $\Sigma_{nok} \subseteq \Sigma'$ be the set of all cells in Σ that are actually faulty, and $\Sigma_{ok} := \Sigma \setminus \Sigma_{nok}$ be the set of all cells in Σ that are actually correct. Then, we call a set of components $\Delta \subseteq \Sigma'$

- *a correct diagnosis for* Σ iff $\Delta \in \mathbf{AllD}_{P_{VBM}}$ and $\Delta \subseteq \Sigma_{nok}$,
- an *FP-spurious diagnosis for* Σ iff $\Delta \in \mathbf{AllD}_{P_{VBM}}$ and $\Delta \cap \Sigma_{ok} \neq \emptyset$, and
- an *FN-spurious diagnosis for* Σ iff $\Delta \notin \mathbf{AllD}_{P_{VBM}}$.

Note, for a given set of observations, there might be multiple correct diagnoses in the above sense. However, given a sufficiently knowledgeable user or expert able to answer questions about the correctness of spreadsheet cells, the set of observations can be extended in a way that only one correct minimal diagnosis for the new problem including the additional observations exists (whereas all other minimal diagnoses for earlier problems have been proven spurious by the new observations) [61]. Regardless of whether interactive techniques, asking a user for additional information, or ranking techniques, imposing an order on the computed diagnoses using some preference criterion, are used to facilitate the detection of a correct diagnosis, these techniques will tend to be more effective and lead to a more efficient fault localization if *(1)* all correct diagnoses are among the diagnosis set $\mathbf{AllD}_P$ for a spreadsheet-constraint diagnosis problem P, and if *(2)* $\mathbf{AllD}_P$ comprises as few spurious diagnoses as possible.

Regarding (1) and (2), the next theorem shows that *(i)* irrespective of the model type among $\{VBM, QDM, FDM\}$ used to describe a spreadsheet-constraint diagnosis problem P, all correct diagnoses will be among $\mathbf{AllD}_P$, *(ii)* all model types give rise to the same FP-spurious diagnoses, and *(iii)* the model types differ only in terms of the generated FN-spurious diagnoses.

Theorem 5 *(Relationship between model types w.r.t. correct and spurious diagnoses). Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, and P_{MT} be a spreadsheet-constraint diagnosis problem modeling Σ by means of the model type MT. Let moreover $Corr(\Sigma)$, $FPsp(\Sigma)$ and $FNsp(\Sigma)$ denote the correct, FP-spurious and FN-spurious diagnoses for Σ, respectively. Then,*

1. $\mathbf{AllD}_{P_{MT}} \supseteq Corr(\Sigma)$ *for all* $MT \in \{VBM, QDM, FDM\}$
   (i.e., for all model types, the diagnosis set includes all correct diagnoses),
2. *the set* $\mathbf{AllD}_{P_{MT}} \cap FPsp(\Sigma)$ *is equal for all* $MT \in \{VBM, QDM, FDM\}$
   (i.e., for all model types, the diagnosis set includes the same set of FP-spurious diagnoses), and
3. $\mathbf{AllD}_{P_{FDM}} \cap FNsp(\Sigma) \supseteq \mathbf{AllD}_{P_{QDM}} \cap FNsp(\Sigma) \supseteq \mathbf{AllD}_{P_{VBM}} \cap FNsp(\Sigma)$
   (i.e., the set of FN-spurious diagnoses w.r.t. FDM dominates the same set w.r.t. QDM which in turn dominates the same set w.r.t. VBM).

Proof. *(Idea)* All three statements of the theorem follow in a straightforward way from Definitions 16 and 21 as well as Corollary 3 (see Appendix C.9). □

This result guarantees that the actually faulty spreadsheet cells can be localized not only by the exact *VBM*, but also by the two abstract model types. In addition, the theorem ensures that the *VBM* can never yield more (FN-)spurious diagnoses than the *QDM* which in turn can never give rise to more (FN-)spurious diagnoses than the *FDM*.

From a slightly different perspective, we might also compare model types regarding their soundness and completeness w.r.t. diagnosis computation:

Definition 22 *(Soundness/Completeness of model types).* Let Σ be a spreadsheet, $\Sigma' \subseteq \Sigma$ be the set of possibly faulty cells, P be a spreadsheet-constraint diagnosis problem modeling Σ by means of the model type *VBM*, and P' be a spreadsheet-constraint diagnosis problem modeling Σ by means of any model type $MT' \in \{VBM, QDM, FDM\}$. Then, we call MT'

- *diagnostically complete for* Σ iff every diagnosis for P is also a diagnosis for P'
  (i.e., iff $\mathbf{AllD}_P \subseteq \mathbf{AllD}_{P'}$), and
- *diagnostically sound for* Σ iff every diagnosis for P' is also a diagnosis for P
  (i.e., iff $\mathbf{AllD}_{P'} \subseteq \mathbf{AllD}_P$).

Given this definition, we have that:

Theorem 6 *(Soundness/Completeness of model types).*

- *All model types in* $\{VBM, QDM, FDM\}$ *are diagnostically complete for any spreadsheet* Σ.
- *The two abstract models, QDM and FDM, are diagnostically unsound in general, i.e., there are spreadsheets* Σ *for which these two models are unsound.*

Proof. *(Idea)* The first bullet point follows directly from Corollary 3 as well as Definitions 21 and 22. To show the second bullet point, we refer to the respective counterexamples given in Sec. 6. (See full proof in Appendix C.10.) □

Given an unsound model, it appears reasonable to quantify its degree of unsoundness, which we can accomplish using the notion of precision. The precision of a model type MT for a spreadsheet Σ describes which ratio of the diagnoses produced for Σ using MT are also diagnoses for Σ using the exact model type VBM.[7] Since it is generally intractable to calculate the set of all diagnoses $\mathbf{AllD}_P$ for a spreadsheet-constraint diagnosis problem P, we characterize the concept of precision with regard to all diagnoses $\mathbf{AllD}_P^{(k)}$ of a specific size k for P:

Definition 23 *(Set of all diagnoses of cardinality k w.r.t. a model).* Let $P = (CS, COMP)$ be a spreadsheet-constraint diagnosis problem. Then, we refer to the set of all diagnoses with cardinality k for P (as per Definition 13) as $\mathbf{AllD}_P^{(k)}$.

Definition 24 *(Precision of model types w.r.t. a diagnosis cardinality k).* Let Σ be a spreadsheet, P be a spreadsheet-constraint diagnosis problem modeling Σ by means of the model type VBM, and P' be a spreadsheet-constraint diagnosis problem modeling Σ by means of any model type $MT' \in \{VBM, QDM, FDM\}$. Then, the *precision of* MT' *for* Σ *w.r.t. the diagnosis cardinality* k is defined as

- $|\mathbf{AllD}_{P'}^{(k)} \cap \mathbf{AllD}_P^{(k)}| / |\mathbf{AllD}_{P'}^{(k)}|$ if $\mathbf{AllD}_{P'}^{(k)} \neq \emptyset$, and
- 1 if $\mathbf{AllD}_{P'}^{(k)} = \emptyset$.

Remark 5. First, by Corollary 3 and Definition 16, $\mathbf{AllD}_{P'}^{(k)} \supseteq \mathbf{AllD}_P^{(k)}$ holds for all k. Hence, the second bullet of Definition 24 considers the case where both $\mathbf{AllD}_{P'}^{(k)} = \emptyset$ and $\mathbf{AllD}_P^{(k)} = \emptyset$. The precision is defined to be 1 in this case since none of the model types MT' and VBM produces any diagnoses of size k, and thus "all" diagnoses generated for MT' are also diagnoses for the VBM. Simply put, the model type MT' leads to a correct output w.r.t. the diagnoses of size k. Second, note that the precision of the model type VBM is 1 by definition for all k. □

The results we derived in this section are insightful to compare the three model types from a theoretical viewpoint regarding their diagnostic accuracy, soundness, and completeness, from which the VBM is superior to the QDM which in turn is superior to the FDM. However, to obtain a clearer picture of which model type to prefer for practical use cases, we need to take into consideration *(1)* the magnitude and significance of the differences between the model types w.r.t. their diagnostic accuracy, as well as *(2)* the magnitude and significance of the differences in terms of their computational performance w.r.t. diagnosis generation. This is what we investigate in Sec. 6.

6. Empirical analysis of model types

In this section, we compare the three model types empirically with regard to their efficiency (diagnostic time performance) and accuracy (number of generated diagnoses), as well as other aspects derived from these properties. In what follows, we first describe the datasets used for our experiments, then we outline the experiment environment and settings, and finally we discuss the obtained experimental results.

6.1. Evaluation datasets

For our evaluation, we used both real-world and artificially generated faulty spreadsheets.

6.1.1. Real-world spreadsheets

The first part of our evaluation considers a subset of the publicly available *Integer Spreadsheet Corpus* [4].[8] This corpus contains 231 spreadsheets, each with up to three artificially seeded faults, i.e., up to three faulty cells. The spreadsheets use arithmetical and logical expressions as well as the functions `SUM`, `MAX` and `IF` in their formulas.[9] The size of the spreadsheets in the corpus ranges from 7 to 241 formulas (avg: 41), and their number of input and output cells varies between 4 and 176 (avg: 44) as well as 1 and 13 (avg: 4), respectively.

[7] Analogously, we could at this stage also define the recall of a model type MT for a spreadsheet Σ, i.e., the fraction of the diagnoses produced for Σ using the exact model type VBM that are also diagnoses for Σ using MT. However, since due to Theorem 6 we have that all discussed model types are diagnostically complete, it is theoretically guaranteed that all model types have a recall of one for arbitrary spreadsheets. Hence, we do not explicitly discuss recall.

[8] http://spreadsheets.ist.tugraz.at/index.php/corpora-for-benchmarking/integer-corpus/.

[9] Functions not directly included in our theory (cf. Sec. 2.1) were represented using auxiliary cells, as sketched in Remark 2.

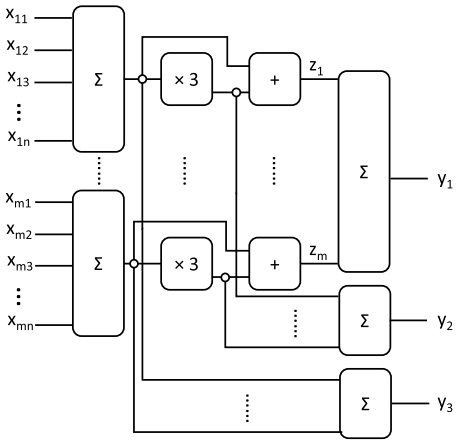


Fig. 6. Parameterizable calculation scheme with $m * n$ inputs x_{ij} and three outputs y_1, y_2, y_3.

For three reasons, some of the spreadsheets in the corpus had to be excluded from our experiments. First, since our prototype currently supports only integer values, we could not analyze spreadsheets containing real values (22 files) at this stage. Second, in line with earlier works [36,33,34], we focus on the task of finding faults in spreadsheet formulas rather than in input cells. The assumption behind this is that users can straightforwardly verify the correctness of input cells (intuitively: the constants in the spreadsheet) before starting a debugging session.[10] Thus, we did not consider spreadsheets with faults seeded in the input cells (17 files). Third, for some spreadsheets we faced computational issues when determining all (minimal and non-minimal) diagnoses, a statistic needed for our evaluations as discussed in Sec. 6.3 (8 files). As a result, we obtained a collection of 184 spreadsheets to be analyzed in our experiments.

6.1.2. *Artificial spreadsheets*

The second part of our evaluation considers a parameterizable calculation scheme, depicted by Fig. 6, which performs addition and multiplication operations on a set of $m * n$ input integers $x_{11}, \ldots, x_{mn}$, and outputs three integer values y_1, y_2, y_3. Using various settings of m and n, and randomly assigning integer values between 1 and 10 to the input cells, we generated 15 spreadsheets modeling this scheme. The sizes of these spreadsheets range from 27 to 875 in terms of input cells (avg: 249) and from 9 to 63 w.r.t. formula cells (avg: 34), whereas all include 3 output cells.[11]

To generate faulty versions of these spreadsheets, we automatically seeded faults into them by randomly mutating formulas. In particular, we replaced *(a)* references with constants or other references, *(b)* constants with other constants, or *(c)* operators with other operators (e.g., interchanging '+' with '*' and vice versa), and we *(d)* shrank areas of referenced cells. Per baseline spreadsheet, we created 15 faulty versions, five with a single fault (one faulty cell), five with a double fault (two faulty cells), and five with a triple fault (three faulty cells). We used the baseline spreadsheets to determine the expected values for the output variables y_1, y_2, and y_3. Again, assuming users to make a pass over the input cells while verifying their correctness before entering the debugging process, we used in our experiments the 186 of the 225 generated files with faults not included in the input cells.

6.2. *Evaluation environment and settings*

The evaluation was performed on an Intel(R) Xeon(R) W-2150B CPU @ 3.00 GHz with 128 GB RAM, and Windows 10 Enterprise (64 bit) as operating system.

We used Apache POI[12] for parsing the spreadsheets, Java for converting them into constraints, and Minion v1.8 [27] as a constraint solver.

For each faulty spreadsheet Σ in the experiment dataset and for each model type MT among $\{VBM, FDM, QDM\}$ (cf. Sec. 3), we generated a spreadsheet-constraint diagnosis problem $P_{\Sigma,MT}$ by means of Algorithm 2. For each of the resulting problems $P_{\Sigma,MT}$, we called **ConDiag**$(P_{\Sigma,MT}, 3)$ to compute all minimal diagnoses up to cardinality three. Every such call of Algorithm 1 involves three consecutive computation phases: the generation of *(i)* all minimal diagnoses of cardinality one, *(ii)* all minimal diagnoses of cardinality two, and *(iii)* all minimal diagnoses of cardinality three. For each of the phases (i)–(iii), we recorded the (number

[10] Such an initial check is also reasonable with regard to the accuracy and usefulness of the diagnostic output. Because the assumption of the faultiness of input cells can, in general, provide explanations for a wide range of possible observed spreadsheet faults. Hence, forgoing such an initial check of the constants might significantly blow up the number of the computed diagnoses.

[11] The number of formula and output cells amounts to $3 * m + 3$ and 3, respectively, which can be directly seen from Fig. 6. The number of input cells includes the $m * n$ inputs of the scheme, along with cells including strings to designate other cells, and a cell including the constant 3 used in the multiplication operations of the scheme.

[12] https://poi.apache.org/.

of) returned diagnoses, and computed the solving time (averaged over 5 runs of the algorithm). In addition, we used a timeout of 20 minutes for each phase to stop overlong computations, i.e., when the diagnosis computation task for a model type cannot be finished before this deadline, the case is considered unsuccessful for this model type and categorized as a "timeout" case in our results.

Remark 6. Consider the following comments on the choice of the constraint solver and the used variable domains for the value-based model:

- *Choice of constraint solver:* Our approach in general, and the spreadsheet-constraint diagnosis problems $P_{\Sigma,MT}$ used in our experiments in particular, are basically independent of the adopted constraint solving procedure. The only stipulated property of the used solver is that it constitutes a sound and complete procedure for deciding the satisfiability (cf. Definition 11) of constraint systems expressed by the constraint language L that is used to model $P_{\Sigma,MT}$. For instance, constraint solvers such as Minion [27] and Choco[13] or SMT solvers such as Z3[14] can be employed for our problems. These solvers have the same accuracy, i.e., they compute the same diagnoses, but they might exhibit differences w.r.t. their solving times (see [4] for a comparison). Since we are mainly interested in the *relative differences* regarding computational efficiency *between the models* rather than the absolute solving times, we decided to use Minion v1.8, an out-of-the-box, open source tool that supports arithmetic, relational, and logical constraints over Booleans and integers. The use of an alternative solver and/or processor might produce lower absolute computation times. Note, however, that the evaluation of which computation environment or tool framework is optimally efficient *in absolute terms* for the studied benchmark problems is beyond the scope of this work.
- *Variable domains for the value-based model:* The variables that describe the cells' values in the value-based model are integers. Minion requires a range for all integers. The larger this range is, the higher is the worst-case solving time. However, a range that is too small means that diagnoses may not be found. We set the domain size of the integer variables to {-2000..50000}. This range is large enough to represent the values computed by the spreadsheets used in the evaluation, but it is small enough to keep the solving times feasible. □

6.3. Evaluation criteria

For each model type in $\{VBM, FDM, QDM\}$ and each diagnosis cardinality in $\{1, 2, 3\}$, we investigate

(i) the number of generated diagnoses,
(ii) the computation time required to compute all minimal diagnoses,
(iii) the precision (as per Definition 24),
(iv) the ratio of diagnosis candidates ruled out (i.e., the ratio of non-diagnoses), and
(v) the trade-off between the required computation time and the achieved diagnostic accuracy.

The trade-off in *(v)* represents the ratio of the factor of more diagnosis computation time required by *VBM* (vs. a model type *MT*) and the factor of more diagnoses generated by *MT* (vs. *VBM*):

Definition 25 *(Trade-off between time savings and diagnostic accuracy).* Let Σ be a spreadsheet, $MT \in \{VBM, QDM, FDM\}$, and $P_{\Sigma,MT}$ be a spreadsheet-constraint diagnosis problem for Σ described by model type MT, as characterized in Sec. 6.2. Further, let $\mathsf{time}(MT)$ be the computation time for all minimal diagnoses (of a given size) for $P_{\Sigma,MT}$, and $\mathsf{diag}(MT)$ be the number of all diagnoses (of a given size) produced for $P_{\Sigma,MT}$. Then, we define (for diagnoses of a given size) the *trade-off between the required computation time and the achieved diagnostic accuracy for* Σ *and MT* by $f^+_{\mathsf{time}}(VBM, MT)/f^+_{\mathsf{diag}}(MT, VBM)$, where $f^+_{\mathsf{time}}(VBM, MT) := \mathsf{time}(VBM)/\mathsf{time}(MT)$ and $f^+_{\mathsf{diag}}(MT, VBM) := \mathsf{diag}(MT)/\mathsf{diag}(VBM)$.

Some comments on and intuitions behind the examined aspects *(i)–(v)* are given in the following remark.

Remark 7. For *(i)*, we consider the number of *all*, i.e., minimal and non-minimal, diagnoses of a given size. The cause for doing so is the definition of model accuracy (cf. Definition 16), which is based on all diagnoses, rather than on the minimal ones—for good reason. Because non-exact models might give rise to spurious diagnoses, i.e., sets of components that in fact do not constitute valid fault explanations, or, in other words, are no diagnoses w.r.t. an exact model (cf. Sec. 5). Hence, if one model type, e.g., leads to the generation of many spurious minimal diagnoses of low cardinality, then there might be few or even no minimal diagnoses of larger cardinalities for this model type since many or all of them might be supersets of the spurious low-cardinality diagnoses. However, exactly these supersets might be where the valid fault explanations can actually be found. So, simply put, we have to generally take into account also non-minimal diagnoses when relying on non-exact models. Consequently, we also compare the model types based on all diagnoses. Nevertheless, for *(ii)*, we measure the computation times required when using the various model types for the task of determining only the *minimal* diagnoses, since all non-minimal ones are already uniquely characterized by the minimal ones in the

[13] https://choco-solver.org/.
[14] https://github.com/Z3Prover/z3.

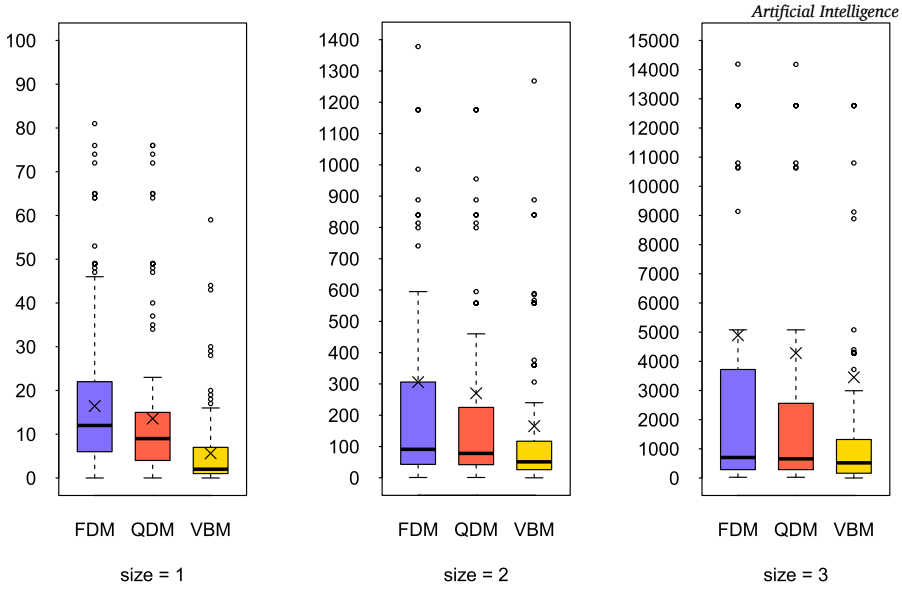


Fig. 7. Real-World Spreadsheets: Boxplot summaries of the **number of returned diagnoses** of different cardinalities (sizes 1, 2, 3) w.r.t. each model type in $\{VBM, QDM, FDM\}$. Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Medians are indicated by a horizontal line, means by a cross. Note, for clarity, in the middle and right plot, some outliers were omitted from the chart. Specifically, in the middle plot (size 2), 5 / 4 / 3 outliers spanning a range of $[2016, 3240]$ / $[3204, 3230]$ / $[1863, 3009]$ are not shown for *FDM* / *QDM* / *VBM*, whereas in the right plot (size 3), 8 / 6 / 6 outliers spanning a range of $[23240, 85320]$ / $[23240, 85310]$ / $[23240, 83780]$ are not shown for *FDM* / *QDM* / *VBM*. Observe that the differences between *QDM* and *FDM* are not statistically significant for sizes 2 and 3 (cf. Table 1).

weak fault model setting we are considering (cf. Sec. 2.3). The idea underlying *(iii)* is to ascertain the ratio of valid fault explanations (i.e., diagnoses w.r.t. the *VBM*) in the set of diagnoses (of a particular size) produced by a model type. The feature *(iv)* is important for scenarios where a model-based approach is not used in a standalone manner, but in combination with other techniques, such as the statistics-based Spectrum-based Fault Localization (SFL) [3]. While such strategies can be highly performant, they might give rise to a large number of possible diagnosis candidates. In such scenarios, using model-based techniques can be very fruitful in that they can allow to *provably* exonerate significant fractions of diagnosis candidates and hence can help streamline both the efficiency and effectivity of other approaches. Thus, *(iv)* measures the fraction of (theoretically possible) diagnosis candidates that are proven to be no diagnoses by a model type. Note, by Corollary 3, whenever a set of components is not a diagnosis w.r.t. a model type (regardless if exact or abstract), then this set is guaranteed to be not a valid fault explanation (i.e., not a diagnosis w.r.t. the *VBM*). Finally, *(v)* is an attempt to quantitatively gauge the trade-off between the required computation time and the usefulness of the generated diagnostic output. More specifically, *(v)* aims at bringing light to questions such as "when switching to a more/less abstract model, how does the cost/gain in terms of additionally/fewer generated spurious diagnoses compare to the saved/incurred time costs?", or "which model type constitutes the overall best trade-off between the generated number of diagnoses and the diagnosis computation time?". □

6.4. Evaluation results

The results[15] of the conducted experiments are presented by Figs. 7–16 and Tables 1–4.[16] In the following, we discuss our findings for the two datasets in turn. A bottomline of the main insights, intended for readers seeking a quick impression, is given in Appendix D. Moreover, detailed results for all individual spreadsheets can be found in Appendix E.

6.4.1. Results for the real-world spreadsheets

Number of generated diagnoses (Fig. 7, Table 1, Table 2) [17] While we have already learned from Corollary 3 that the *VBM* can never give rise to more diagnoses than the *QDM*, and the latter can never produce more diagnoses than the *FDM*, we are still in the dark about *(a)* whether there are differences at all between the model types in this regard, and, if so, *(b)* how often equal diagnosis sets might be generated for two model types, and *(c)* how many more (spurious) diagnoses might result from the usage of a more abstract model. Based on our results, regarding (a), in terms of the number of diagnoses computed, we observe statistically significant differences among all pairs of model types, except for the comparisons between both abstract model types for sizes 2 and 3 (cf. Table 1). That is, over the entire dataset, we can conclude that the outputs of the three model types are generally different for single-faults, and different

[15] Please find the raw results obtained from our experiments as well as the repository including the program code we used for our experiments at https://github.com/bhoferTU/Abstraction-Levels-for-Model-based-Software-Debugging.git.

[16] For interpretation of the colors in the figure(s) and in the table(s), the reader is referred to the web version of this article.

[17] See Figs. E.18–E.20 in Appendix E for the detailed results for all single spreadsheets.

Table 1
Real-World Spreadsheets: **Statistical significance results** for pairwise t-tests (with Bonferroni adjustment) among the model types in $\{VBM, FDM, QDM\}$ for the different cardinalities (sizes 1, 2, 3) of the computed diagnoses (*columns*) and for the different aspects (cf. Figs. 7–11) analyzed in our evaluations (*rows*). The symbols $***$ / $**$ / $*$ and $\times$ mean statistical significance w.r.t. the level α for $\alpha = 0.001/0.01/0.05$ and 'no statistical significance', respectively.

		size = 1		size = 2		size = 3	
		VBM	*FDM*	*VBM*	*FDM*	*VBM*	*FDM*
Number of returned diagnoses	*VBM*		***		***		**
	QDM	***	***	***	×	**	×
Computation time	*VBM*		***		***		***
	QDM	***	***	***	***	***	×
Precision	*VBM*		***		***		***
	QDM	***	***	***	*	***	*
Ratio of candidates ruled out	*VBM*		***		***		***
	QDM	***	***	***	*	***	*
Time-accuracy ratio	*VBM*		***		***		***
	QDM	***	×	***	*	***	×

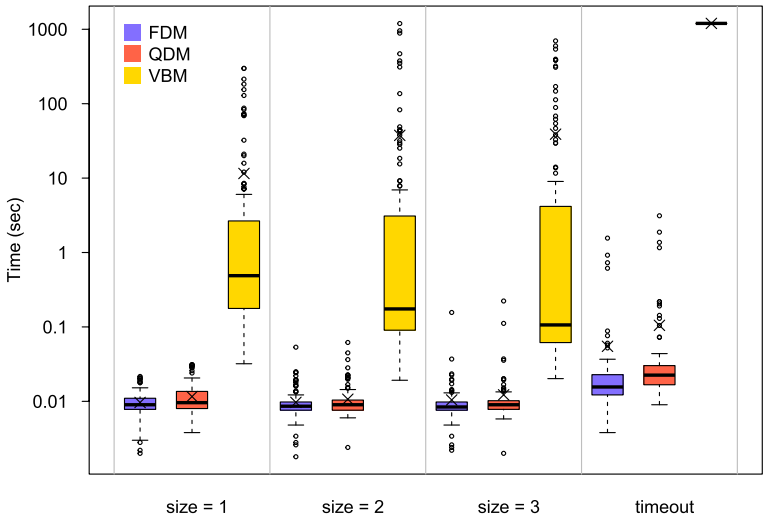


Fig. 8. Real-World Spreadsheets: Boxplot summaries of the **computation times (in seconds)** required when using the model types in $\{VBM, QDM, FDM\}$ to compute all minimal diagnoses of various cardinalities. The *three left sectors of the plot (size 1, 2, 3)* illustrate the cases where no timeout was triggered, i.e., all minimal diagnoses of the given size could be computed w.r.t. all model types. The *rightmost sector of the plot (timeout)* presents the times required by *FDM* and *QDM* for all cases where a timeout was triggered for *VBM*; the values for *VBM* shown in this plot are lower bounds (corresponding to the timeout of 1200 seconds) of the computation time needed when using *VBM*. Medians are indicated by a horizontal line, means by a cross. Note the logarithmic scale of the y-axis. Observe that the differences between *QDM* and *FDM* are not statistically significant for size 3 (cf. Table 1).

between the exact model and any abstract model type for double- and triple-faults, i.e., changing the model type has a measurable effect in these cases. As regards (b), Table 2 provides evidence that the *QDM* is equally accurate, i.e., generates the same solutions, as the *VBM* for 40 % / 42 % / 52 % of the spreadsheets when diagnoses of cardinality 1 / 2 / 3 are sought. These are decent values in the face of the substantial advantages of the *QDM* compared to the *VBM* in terms of time performance (see below). Particularly striking is also that the *QDM* matches up to the *VBM* in more than half of the cases (for cardinality 3), despite its much simpler nature. A little lower but still fair ratios of 29 % / 37 % / 48 % could be recorded for the *FDM*, demonstrating that also the most abstract model type under test can be an expedient tool in many cases. Moreover, we find that the two abstract models lead to the same output in 76 % / 89 % / 91 % of the cases, which shows that the additional modeling considerations incorporated into the *QDM* compared to the *FDM* can, but in the majority of cases do not, translate into more concise solution sets. Addressing (c), Fig. 7 shows median numbers of computed diagnoses for *FDM* / *QDM* / *VBM* of 12 / 9 / 2 for diagnosis size 1, of 91 / 78 / 51 for diagnosis size 2, and of 703 / 658 / 519 for diagnosis size 3. This tells us that, in absolute terms, all three model types are reasonable in case the actual diagnosis is assumed among the single-fault diagnoses. In relative terms, the difference between the medians however is largest for single-fault diagnoses, and appears to get gradually smaller as the size of the computed diagnoses is increased. In other words, we observe a trend towards a better relative accuracy of the abstract model types for growing diagnosis size compared to the *VBM*. Studying Fig. 7, we also detect that there are some "outlier" spreadsheets for all model types which lead to very large sets of diagnoses. Hence, when the actual diagnosis is conjectured to be a multiple-fault diagnosis, it can for efficiency reasons make sense to complement this mere diagnosis computation approach by other techniques that either collect additional information to prune the diagnosis space, or that allow to impose a ranking on the diagnoses based on useful meta information (e.g., estimated fault probabilities).

Table 2

Real-World and Artificial Spreadsheets: The table shows **for how many and for which ratio (in percent) of the spreadsheets certain predicates are true**. Investigated are the main aspects under evaluation, i.e., the number of returned diagnoses (*first and third quarter of the table* viewed from the top) as well as the computation time (*second and fourth quarter*), for different diagnosis cardinalities (sizes 1, 2, 3; *row sectors*), for all pairs of model types in $\{VBM, QDM, FDM\}$ (*column sectors*), and for both examined spreadsheet corpora (*top and bottom half of the table*). Note, for the "Number of returned diagnoses" category, the values for one and the same pair of model types (the two values in one and the same row and in one and the same column sector, i.e., between two bold vertical lines) add up to the overall number of cases for the "Number of cases" row, and to 100 % for the "Ratio of cases" row, respectively. For the "Computation time" category, this is often the case as well; if not, this means that the difference between the sum of the two values and the overall number of cases (100 %) is the number (ratio) of cases where computation times were equal for both model types. The greater value for one and the same pair of model types is highlighted by boldface. E.g., the top left entry means that in 70 from overall 177 cases, the number of returned diagnoses was equal for *VBM* and *QDM*, which amounts to 40 % of all cases; hence, it was more often (107 cases, 60 %) the case that *VBM* yielded fewer diagnoses than *QDM*, which is why the values 107 and 60 are emphasized.

Real-World Spreadsheets	**Number of returned diagnoses**		**VBM=QDM**	**VBM<QDM**	**VBM=FDM**	**VBM<FDM**	**QDM=FDM**	**QDM<FDM**
	size = 1	Number of cases (from 177)	70	**107**	51	**126**	**135**	42
		Ratio of cases (in %)	40	**60**	29	**71**	**76**	24
	size = 2	Number of cases (from 142)	59	**83**	53	**89**	**126**	16
		Ratio of cases (in %)	42	**58**	37	**63**	**89**	11
	size = 3	Number of cases (from 129)	**67**	62	62	**67**	**117**	12
		Ratio of cases (in %)	**52**	48	48	**52**	**91**	9
	Computation time		**VBM>QDM**	**VBM<QDM**	**VBM>FDM**	**VBM<FDM**	**QDM>FDM**	**QDM<FDM**
	size = 1	Number of cases (from 177)	**177**	0	**177**	0	**126**	38
		Ratio of cases (in %)	**100**	0	**100**	0	**71**	21
	size = 2	Number of cases (from 142)	**142**	0	**142**	0	**92**	29
		Ratio of cases (in %)	**100**	0	**100**	0	**65**	20
	size = 3	Number of cases (from 129)	**128**	1	**129**	0	**86**	30
		Ratio of cases (in %)	**99**	1	**100**	0	**67**	23
Artificial Spreadsheets	**Number of returned diagnoses**		**VBM=QDM**	**VBM<QDM**	**VBM=FDM**	**VBM<FDM**	**QDM=FDM**	**QDM<FDM**
	size = 1	Number of cases (from 186)	63	**123**	55	**131**	**168**	18
		Ratio of cases (in %)	34	**66**	30	**70**	**90**	10
	size = 2	Number of cases (from 184)	2	**182**	0	**184**	**165**	19
		Ratio of cases (in %)	1	**99**	0	**100**	**90**	10
	size = 3	Number of cases (from 91)	2	**89**	1	**90**	**86**	5
		Ratio of cases (in %)	2	**98**	1	**99**	**95**	5
	Computation time		**VBM>QDM**	**VBM<QDM**	**VBM>FDM**	**VBM<FDM**	**QDM>FDM**	**QDM<FDM**
	size = 1	Number of cases (from 186)	**186**	0	**186**	0	**143**	31
		Ratio of cases (in %)	**100**	0	**100**	0	**77**	17
	size = 2	Number of cases (from 184)	**184**	0	**184**	0	**148**	26
		Ratio of cases (in %)	**100**	0	**100**	0	**80**	14
	size = 3	Number of cases (from 91)	**91**	0	**91**	0	**64**	16
		Ratio of cases (in %)	**100**	0	**100**	0	**70**	18

Computation time (Fig. 8, Table 1, Table 2, Table 3)[18] The main questions of interest concerning the efficiency of the model types are *(a)* whether there is generally a difference between the model types in terms of their time performance, and, if so, *(b)* what the magnitude of the performance differences is, and *(c)* whether all model types are reasonably applicable in all cases. As to (a), Table 1 indicates that the difference in computation time is statistically significant for all pairs of model types and all diagnosis sizes, except for the pair (*QDM*,*FDM*) for diagnosis cardinality 3. That is, the *VBM* leads to a measurably (and consistently, cf. Table 2) inferior time performance for diagnosis computation than the two abstract models, and the *QDM* exhibits a measurably worse time performance than the *FDM* for single- and double-faults. However, none of the abstract models consistently outperforms the other. This is shown in Table 2, where we observe that the *FDM* manifests time improvements over the *QDM* in around two thirds of the cases, whereas the latter allows an at least as fast diagnosis computation in the remaining cases. This similar performance of the abstract model types can also be well observed in Fig. 8, which informs us about (b). It shows that the median computation times for the two abstract models are always close to each other and very low in absolute terms. Their runtime exceeds 1 second only in very few cases and reaches maxima of merely 1.6 (*FDM*) and 3.1 (*QDM*) seconds over all cases. By absolute numbers, when considering the median, also the exact *VBM* exhibits a reasonable time behavior of less than a second for all three diagnosis sizes. However, this is not the end of the story; in fact, there are a few caveats to bear in mind. First, note that the *VBM* is by at least an order of magnitude inferior than both abstract models for all diagnosis sizes in terms of the median, and already more than two orders of magnitude worse as concerns the third quartile (upper end of the boxes in the boxplots). Second, there is a substantially larger number of "outlier" cases (above the upper whisker of the boxplots) for the *VBM* than for the abstract models. Third, regarding these extreme values, the difference between the *VBM* and the abstract model types amounts to more than three orders of magnitude for all diagnosis sizes (e.g., almost 20 minutes for *VBM* vs. 0.01 seconds for *FDM* and *QDM* for one and the same spreadsheet). This also leads the discussion to question (c), which must be answered negatively, as summarized by Table 3. It reveals that, when using the

[18] See Figs. E.21–E.23 in Appendix E for the detailed results for all single spreadsheets.

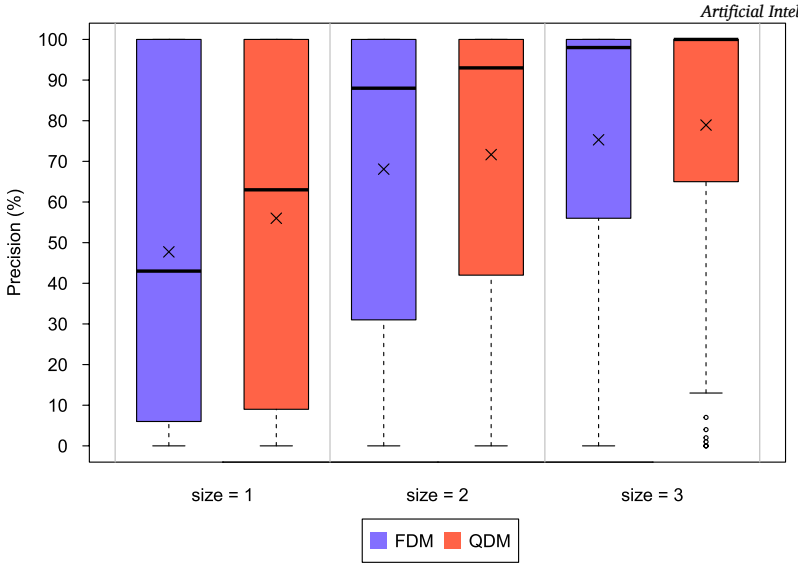


Fig. 9. Real-World Spreadsheets: Boxplot summaries of the **precision** (cf. Definition 24) achieved by the model types in $\{QDM, FDM\}$ w.r.t. diagnoses of various cardinalities (sizes 1, 2, 3). Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Values for *VBM* are omitted since they always amount to 100 % (exact model). Medians are indicated by a horizontal line, means by a cross.

VBM, there are cases for all diagnosis cardinalities where it was not possible to compute all minimal diagnoses of the given size before the timeout (cf. Sec. 6.2) was triggered. In numbers, we already have 7 timeout cases for diagnosis size 1 where the combinatorial explosion (large set of diagnosis candidates) is not an issue. This suggests that the reasoning is very time-consuming and thus a main source of complexity for the *VBM*. For double- and triple-faults, we even register almost a quarter and, respectively, a third of all cases to be unsuccessful when relying on the *VBM*. This motivates the general necessity of abstract models in model-based diagnosis, and for spreadsheets in particular.

Precision (Fig. 9, Table 1) When opting for the use of a non-exact model, we are interested in the degree of unsoundness w.r.t. the computed diagnoses this will potentially cause. Simply put, we want to know which fraction of the diagnoses (of a particular size) in the solution set are valid fault explanations (i.e., diagnoses w.r.t. the *VBM*). This intuition is formalized by the notion of precision (cf. Definition 24). Since the precision for the *VBM* is trivially always 100 %, we report only the results for the abstract model types in Fig. 9. It shows that the expected precision is statistically significantly (cf. Table 1) different for all model types and all diagnosis sizes. Also, the effect size is notable, especially for single-fault diagnoses, which can be seen by studying both the medians (43 vs. 63, 88 vs. 93, and 98 vs. 100 percent for *FDM* vs. *QDM* and sizes 1, 2, and 3) and the means (48 vs. 56, 68 vs. 72, and 75 vs. 79 percent). What these values also reveal is that the fraction of generated (FN-)spurious diagnoses (cf. Definition 21) decreases for both abstract model types when the cardinality of the computed diagnoses is incremented. In other words, model types tend to be more precise for multiple-fault diagnoses. However, on the other hand, we also encountered cases where the precision was 0 %, for both the *QDM* (in 16 % / 2 % / 1 % of the cases for size 1 / 2 / 3) and the *FDM* (in 17 % / 2 % / 1 % of the cases for size 1/ 2 / 3). This situation occurs when there are no valid fault explanations of a given size while the abstract models still produce some diagnoses of this cardinality (but the exact *VBM* does not). In such a scenario, all output diagnoses of this cardinality are spurious. On the positive side, however, we need to emphasize that the cases where the precision amounts to 100 % are much more numerous (e.g., 40 % / 59 % / 67 % of the cases for the *QDM* and size 1 / 2 / 3). This can also be seen from the "VBM = QDM" and "VBM = FDM" columns in the top quarter of Table 2. Overall, we regard these precision results as fairly reasonable given the drastic superiority of the abstract models to the *VBM* as to the runtime performance.

Ratio of ruled out diagnosis candidates (Fig. 10, Table 1) Let us now assume that model-based diagnosis is used in an application scenario where its main purpose is the restriction of the diagnosis search space, e.g., to make other techniques that rely or operate on a set of diagnosis candidates more effective or efficient. With this use case in mind, our focus shifts from the analysis of the diagnoses to the investigation of the non-diagnoses under the use of the various model types. That is, in contrast to evaluating how few non-diagnoses are (incorrectly) found by the model types, as precision does, we now illuminate how many non-diagnoses are (correctly) not output. Fig. 10 gets to the bottom of this aspect. It indicates that the median ratio of non-diagnoses correctly classified as such for *FDM* / *QDM* / *VBM* amounts to 31 / 53 / 91 percent for size 1, to 8 / 23 / 58 percent for size 2, and to 2 / 11 / 43 percent for size 3. This means that, e.g., for prominent alternative fault localization techniques such as SFL approaches [3], which mainly address the single-fault case, their combination with model-based reasoning can bring a *provable* reduction of the theoretical search space of up to more than 90 % while requiring a median of less than 1 second time (see above) to accomplish this in our experiments. This points to the potential of synergizing various diagnostic methodologies, such as logics-based and statistics-based ones. The statistical significance results presented in Table 1 also suggest that the model types actually differ as regards their ability

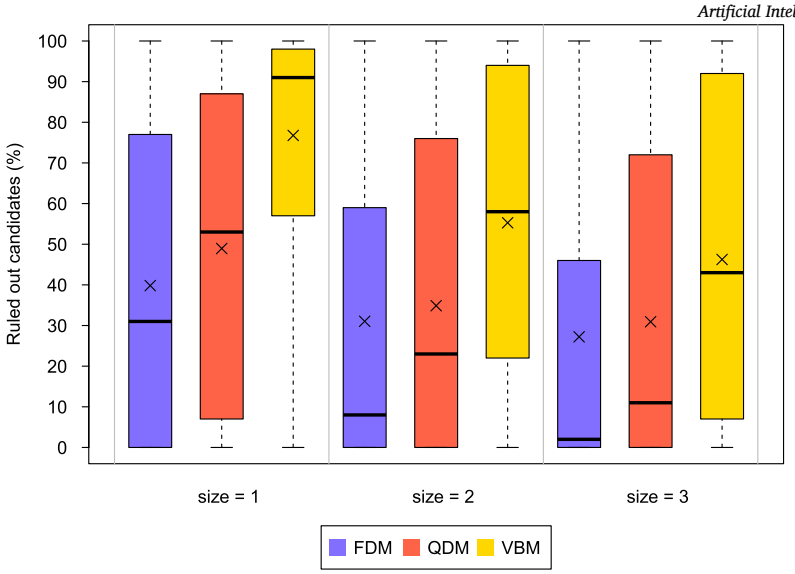


Fig. 10. Real-World Spreadsheets: Boxplot summaries of the **ratio of diagnosis candidates ruled out** by the model types for different cardinalities (sizes 1, 2, 3). In other words, the plots show the ratio of all sets of cells (of sizes 1, 2, 3) that are proven to be non-diagnoses by the model types. Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Medians are indicated by a horizontal line, means by a cross.

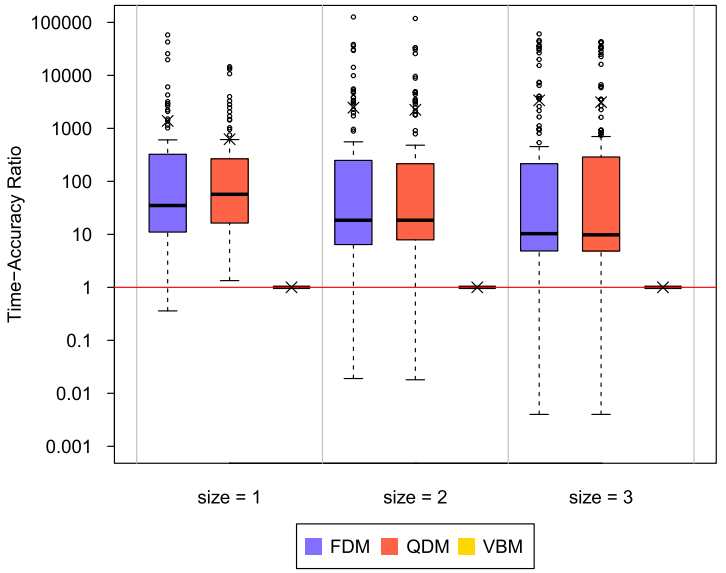


Fig. 11. Real-World Spreadsheets: Boxplot summaries of the **trade-off between time savings and diagnostic accuracy** (cf. Definition 25) of the model types compared to the *VBM* for different cardinalities (sizes 1, 2, 3). The values for *VBM*, which are trivially always 1, are included for comparison. The red horizontal line highlights the areas of a favorable (above the line) vs. an unfavorable (below the line) time-accuracy trade-off achieved by a model type. Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Medians are indicated by a horizontal line, means by a cross. Note the logarithmic scale on the y-axis. Observe that the differences between *QDM* and *FDM* are not statistically significant for sizes 1 and 3 (cf. Table 1).

to disprove diagnoses. Thus, for all examined sizes of diagnoses, the *VBM* is the best choice to this end, and the *QDM* is a better option than the *FDM*. Notably, for single-fault diagnoses, also the comparably simple abstract model types can be powerful search space pruning tools, where the *QDM* could eliminate a median of more than half of the diagnosis candidates, and the *FDM* more than a quarter in our experiments.

Trade-off between efficiency and accuracy (Fig. 11, Table 1) Having shed light on the relationship between model types in terms of both the number of generated diagnoses and the required computation time separately, where the order of favorability of the model types is inverse for both aspects (*VBM* dominates *QDM* dominates *FDM* vs. *FDM* dominates *QDM* dominates *VBM*), it now stands

Table 3
Real-World and Artificial Spreadsheets: The table shows **for how many and for which ratio (in percent) of the spreadsheets the diagnosis computations for *VBM* could not be finished (timeout was triggered)**, for different diagnosis cardinalities (sizes 1, 2, 3; *columns*), and for both examined spreadsheet corpora (*top and bottom half of the table*). E.g., the top-right entry means that in 55 from overall 184 cases, a timeout occurred for *VBM*, which amounts to 30 % of all cases.

	Timeout for VBM	size = 1	size = 2	size = 3
Real-World Spreadsheets	Number of spreadsheets (from 184)	7	42	55
	Ratio (in %) of spreadsheets	4	23	30
Artificial Spreadsheets	Number of spreadsheets (from 186)	0	2	95
	Ratio (in %) of spreadsheets	0	1	51

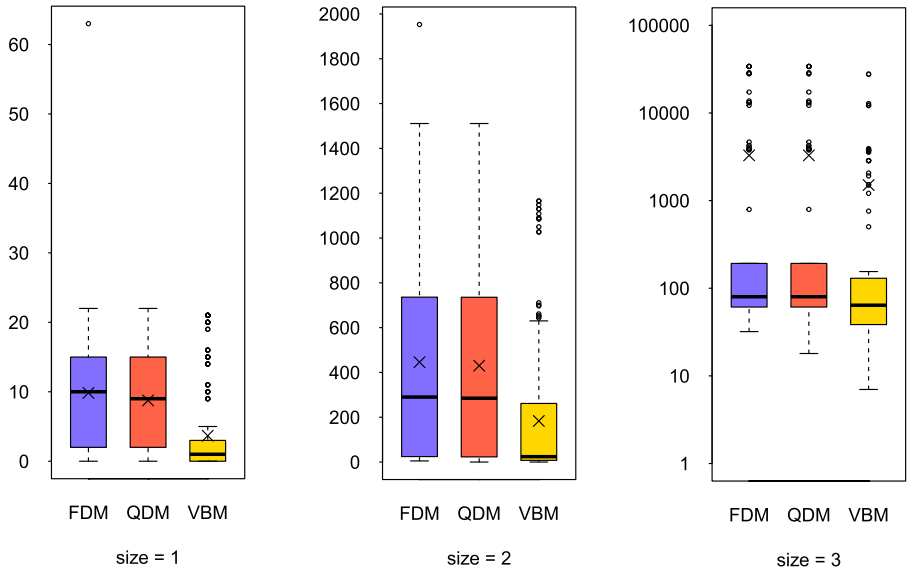


Fig. 12. Artificial Spreadsheets: Boxplot summaries of the **number of returned diagnoses** of different cardinalities (sizes 1, 2, 3) w.r.t. each model type in $\{VBM, QDM, FDM\}$. Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Medians are indicated by a horizontal line, means by a cross. Note the logarithmic scale of the y-axis in the rightmost plot. Observe that the differences between *QDM* and *FDM* are not statistically significant for size 3 (cf. Table 4).

to reason to inquire into the question which model type might constitute the best trade-off between efficiency and accuracy. To this end, taking the *VBM* as a baseline, we evaluate model types *MT* based on the ratio between the factor of more time required by the *VBM* compared to *MT*, and the factor of more diagnoses generated by *MT* compared to the *VBM*. Simply said, we intend to figure out whether the gain in diagnostic accuracy achieved by switching to a less abstract model is worthwhile in the sense of not implying a greater loss in time performance in exchange, and vice versa. Fig. 11 illustrates the results, where larger values for a model mean a better achieved trade-off, and the red horizontal line marks the threshold between a favorable trade-off above and an unfavorable one below the line in comparison to the *VBM*. Note that the values for the *VBM* are trivially always 1 by definition. We can see that the abstract models fare pretty similarly for the considered multiple-fault cases (exactly equal medians of 18 and 10 for size 2 and 3, respectively), while the *QDM* produces the best results (median 57 vs. 35 for *FDM*) in the single-fault scenario (difference is however not statistically significant, cf. Table 1). The difference between each of the abstract model types and the *VBM* is, on the contrary, always statistically significant. For these comparisons, also the effect sizes are substantial, as Fig. 11 clearly shows. In concrete terms, the average ratio for *QDM* for diagnosis sizes 1 / 2 / 3 comes to roundly 620 / 2240 / 3120. These factors describe the extent by which the time reduction when making use of *QDM* instead of *VBM* is expected to outweigh the increase in the number of produced diagnoses. That is, using *QDM* instead of *VBM*, on average, brings hundreds of times more time savings than the set of generated diagnostic solutions is inflated. However, as the boxplots also evince, there are as well some cases where the ratio is below 1 for the abstract models, meaning that they do not always trade less time for more diagnoses in a favorable way. Another observation is that the variability of the ratios over our dataset is relatively high, ranging from more than five orders of magnitude in the favorable direction (maximum: greater than 100 000) to almost three orders of magnitude in the unfavorable direction (minimum: 0.004). Consequently, the achieved time savings do not allow to draw very reliable and precise conclusions about the resulting accuracy losses, and vice versa.

Table 4
Artificial Spreadsheets: **Statistical significance results** for pairwise t-tests (with Bonferroni adjustment) among the model types in $\{VBM, FDM, QDM\}$ for the different cardinalities (sizes 1, 2, 3) of the computed diagnoses (*columns*) and for the different aspects (cf. Figs. 12–16) analyzed in our evaluations (*rows*). The symbols $***$ / $**$ / $*$ and $\times$ mean statistical significance w.r.t. the level α for $\alpha = 0.001/0.01/0.05$ and 'no statistical significance', respectively.

		size = 1		size = 2		size = 3	
		VBM	*FDM*	VBM	*FDM*	VBM	*FDM*
Number of returned diagnoses	*VBM*		***		***		*
	QDM	***	**	***	**	*	×
Computation time	*VBM*		***		***		**
	QDM	***	***	***	***	**	*
Precision	*VBM*		***		***		***
	QDM	***	**	***	*	***	×
Ratio of candidates ruled out	*VBM*		***		***		***
	QDM	***	***	***	***	***	×
Time-accuracy ratio	*VBM*		***		***		***
	QDM	***	***	***	***	**	**

6.4.2. Results for the artificial spreadsheets

Number of generated diagnoses (Fig. 12, Table 2, Table 4)[19] As witnessed by Table 4, all pairs of model types are in fact different w.r.t. the number of generated diagnoses for all three investigated diagnosis cardinalities, except for the pair (*FDM*,*QDM*) for diagnosis size 3. With medians of 10 / 9 / 1 generated diagnoses for the model types *FDM* / *QDM* / *VBM* (see left plot in Fig. 12), the results for the artificial spreadsheets for the single-fault case are very similar to the ones for the real-world spreadsheets regarding the relative differences between the three model types. For diagnosis size 2, however, we see a pretty different picture. Whereas the relationship between the median of the abstract models *FDM* / *QDM* compared with the median of the exact model type *VBM* corresponds to a factor of 1.8 / 1.5 (number of diagnoses less than doubled) for the real-world cases, we see factors of 12.1 / 11.9 (one order of magnitude more diagnoses) in case of the artificial ones. That is, comparing abstract models against the exact one, there are (much) higher relative numbers of spurious diagnoses for the artificial spreadsheets than for the real-world ones when double-faults are considered. This is also reflected by the numbers presented in Table 2, where we recognize that the high fractions of the cases where the *QDM* / *FDM* yielded the same solutions as the *VBM* for the real-world spreadsheets drop from 42 to 1 / 37 to 0 percent for the artificial spreadsheets. Moreover, there are much fewer cases where *VBM* led to a timeout. In short, for double-fault diagnoses, the relative performance of *VBM* is substantially better here than for the real-world cases. Further analyses are needed to better understand the reasons for this. When studying the results for cardinality 3, we must carefully take into account that (the hardest) more than half of the cases are not included in Fig. 12 because they triggered a timeout for *VBM* (cf. Table 3).[20] For the remaining cases, where all model types worked successfully, we perceive only slight differences between the abstract model types and the exact one, with medians of 80 / 80 / 64 for *FDM* / *QDM* / *VBM* (cf. right plot in Fig. 12). That is, for triple-fault diagnoses, whenever the *VBM* was reasonably applicable, both abstract models produced comparably accurate diagnostic results. However, remarkably and in contrast to what we have seen for the real-world spreadsheets, we hardly register cases (2 % for the *QDM*, 1 % for the *FDM*) where one of the abstract model types could achieve an exactly equal accuracy as the *VBM* (cf. Table 2). So, overall, our insights are that *(1)* all model types are reasonable for single-fault diagnoses, whereas *(2)* for double-faults only the exact model appears to be practical, and *(3)* for triple-faults, more than half of the cases lead to a timeout for the exact model, while abstract models fare almost equally well as the exact one for the remaining (easier) cases. In general, however, due to the sometimes considerable numbers of generated diagnoses of sizes 2 and 3, it appears reasonable to combine the discussed model-based diagnosis techniques with other methods that aim at restricting or ordering the returned set of diagnoses whenever the actual diagnosis is conjectured to be a multiple-fault.

Computation time (Fig. 13, Table 2, Table 3, Table 4)[21] Consulting Table 4, we first observe that the difference between all pairs of model types w.r.t. the diagnosis computation efficiency is statistically significant for all diagnosis sizes. In other words, all model types generally imply a measurably different time performance. Taking a look at Fig. 13, it becomes evident that median computation times using the *FDM* are faster than for the *QDM* for all diagnosis sizes (*QDM* requires 37 / 25 / 3 percent more median time than *FDM* for sizes 1 / 2 / 3). However, absolute computation times for both abstract models are always below 1 second and thus negligible (even for the cases where the *VBM* triggers the timeout). Hence, in our experiments, the abstract models exhibited very favorable and practically indistinguishable behavior. Very clear-cut and consistent is also the performance gain of each abstract model type over the exact one (cf. Fig. 13 and Table 2). Specifically, we find (median-based) time overhead factors of *VBM* versus *FDM* and *QDM* of more than one order of magnitude for size 1, and more than two orders of magnitude for sizes 2 and 3. For the cases where *VBM* led to a timeout, we even record savings of (at least) four orders of magnitude achieved by the abstract models over the *VBM*. Interestingly, and similarly as for the number of diagnoses criterion investigated above, the time differences between the exact and the abstract models are (much) more substantial for the artificial spreadsheets than for the real-world ones for diagnosis size 2. It is important to note, however, that the changes take reverse directions for both aspects, i.e., for the artificial spreadsheets the *VBM* is

[19] See Figs. E.24–E.26 in Appendix E for the detailed results for all single spreadsheets.

[20] See Fig. E.26 in Appendix E for the results for the abstract model types in those cases where *VBM* failed.

[21] See Figs. E.27–E.29 in Appendix E for the detailed results for all single spreadsheets.

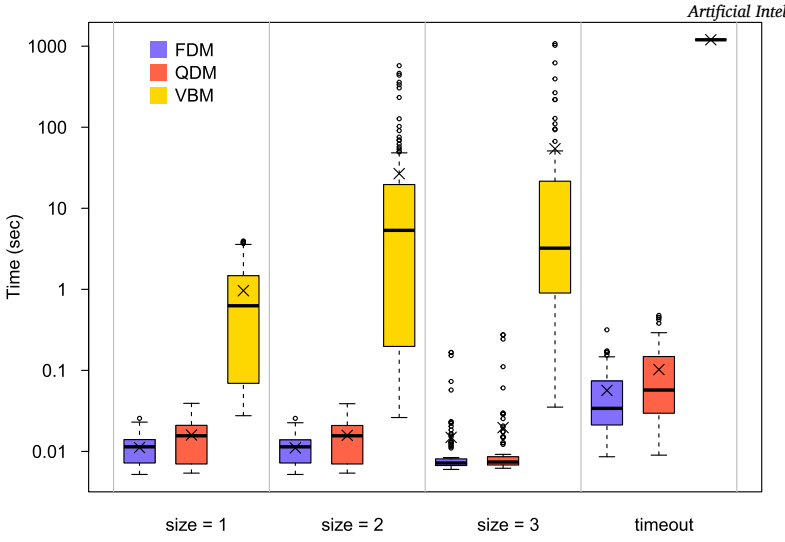


Fig. 13. Artificial Spreadsheets: Boxplot summaries of the **computation times (in seconds)** required when using the model types in $\{VBM, QDM, FDM\}$ to compute all minimal diagnoses of various cardinalities. The *three left sectors of the plot (size 1, 2, 3)* illustrate the cases where no timeout was triggered, i.e., all minimal diagnoses of the given size could be computed w.r.t. all model types. The *rightmost sector of the plot (timeout)* presents the times required by *FDM* and *QDM* for all cases where a timeout was triggered for *VBM*; the values for *VBM* shown in this plot are lower bounds (corresponding to the timeout of 1200 seconds) of the computation time needed when using *VBM*. Medians are indicated by a horizontal line, means by a cross. Note the logarithmic scale of the y-axis. Observe that the differences between *QDM* and *FDM* are not statistically significant for size 3 (cf. Table 4).

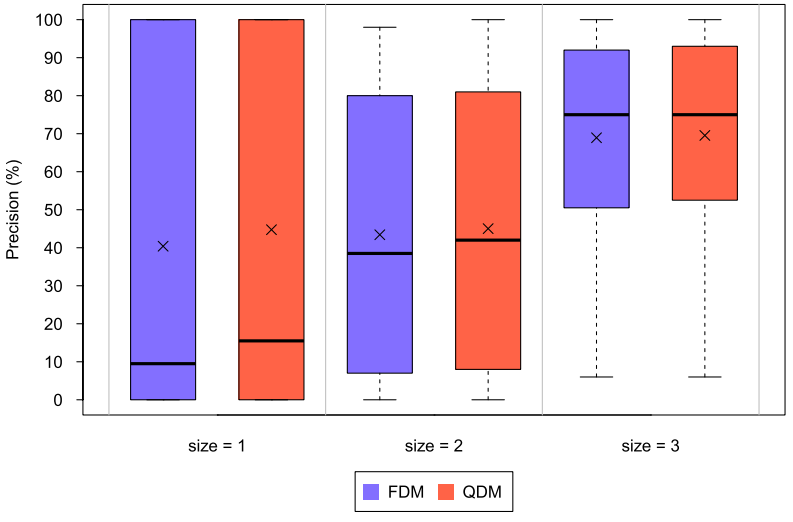


Fig. 14. Artificial Spreadsheets: Boxplot summaries of the **precision** (cf. Definition 24) achieved by the model types in $\{QDM, FDM\}$ w.r.t. diagnoses of various cardinalities (sizes 1, 2, 3). Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Values for *VBM* are omitted since they always amount to 100 % (exact model). Medians are indicated by a horizontal line, means by a cross. Observe that the differences between *QDM* and *FDM* are not statistically significant for size 3 (cf. Table 4).

much *better* w.r.t. the number of diagnoses whereas it is much *worse* w.r.t. the computation time in relation to the abstract model types. This suggests that a higher relative improvement in efficiency over another model type tends to simultaneously involve a higher deterioration w.r.t. accuracy, and vice versa. In absolute terms, the computation times for both abstract models *never* exceeded 0.5 seconds in our experiments, not even for the "hardest" cases where *VBM* triggered the timeout. The latter, on the other hand, required an average / median of 1.0 / 0.6 seconds for size 1, 27 / 5 seconds for size 2, 54 / 3 seconds for size 3, and more than 1200 / 1200 seconds for all timeout-cases. Even though drastically higher than for the abstract model types, we note that these times—each for the computation of *all* minimal diagnoses of a given size—are still practical, given that no timeout occurs. Timeouts for the *VBM* were observed for 1 % of the spreadsheets when computing diagnoses of size 2, in 51 % of the cases for size 3, and never for single-fault diagnoses (cf. Table 3).

Precision (Fig. 14, Table 4) Considering Table 4, we first note that the difference in terms of precision between all pairs of model types is statistically significant, except for the pair (QDM, FDM) for diagnosis size 3 where both model types lead to the same median

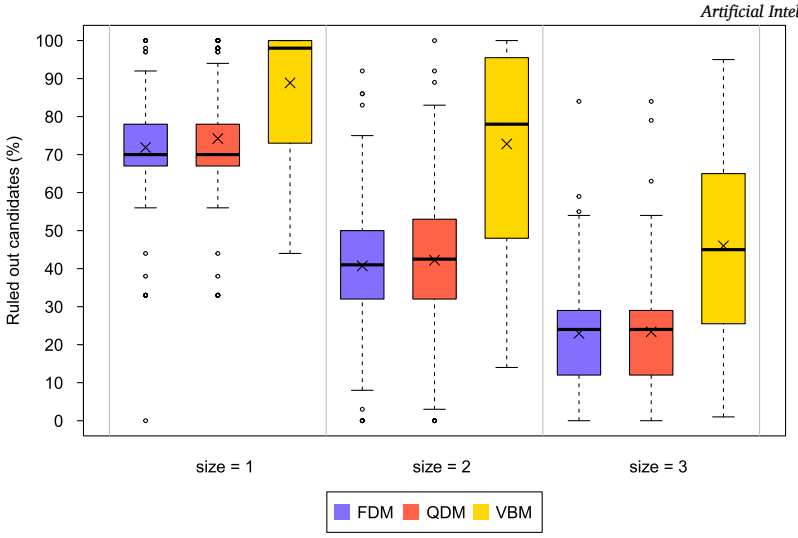


Fig. 15. Artificial Spreadsheets: Boxplot summaries of the **ratio of diagnosis candidates ruled out** by the model types for different cardinalities (sizes 1, 2, 3). In other words, the plots show the ratio of all sets of cells (of sizes 1, 2, 3) that are proven to be non-diagnoses by the model types. Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Medians are indicated by a horizontal line, means by a cross. Observe that the differences between *QDM* and *FDM* are not statistically significant for size 3 (cf. Table 4).

precision. In concrete numbers, for *QDM* / *FDM*, the achieved precision attains medians of 15 / 9 for size 1, 42 / 38 for size 2, and 75 / 75 for size 3. Thus, whereas for both abstract models an acceptable median of 3 out of 4 generated diagnoses are valid fault explanations for diagnosis cardinality 3, we see a drop to around 4 out of 10 for cardinality 2, and to about 1 out of 7 (*QDM*) and 1 out of 11 (*FDM*), respectively, for cardinality 1. In other words, for sizes 1 and 2, most of the diagnoses produced by the abstract model types are spurious. So, a rational way of proceeding would be to adopt the *VBM* (with 100 % precision) for single- and double-fault cases, as it is reasonably fast and barely runs into timeouts (see above), and switch to one of the abstract models for triple-faults as they are both highly efficient and reasonably precise in this scenario. When contrasting these results with the ones for the real-world cases above, we observe a similar relative precision between *QDM* and *FDM*, i.e., for sizes 1 / 2 / 3, the former yields a by 31 / 5 / 2 percent better precision for the real-world cases, and fares by 39 / 8 / 0 percent better considering the artificial cases. Nevertheless, we recognize that the absolute precision of both abstract models proves to be considerably and consistently lower for the artificial cases, which is in line with the insights we gained when discussing the number of generated diagnoses above.

Ratio of ruled out diagnosis candidates (Fig. 15, Table 4) Assume again we intend to use model-based reasoning based on the three model types to provably rule out diagnosis candidates. Fig. 15 indicates fairly practical stats w.r.t. the correct classification of non-diagnoses for all model types and all diagnosis cardinalities. In numbers, we observe medians for the median percentage of eliminated diagnosis candidates for *FDM* / *QDM* / *VBM* of 70 / 70 / 98 for size 1, of 41 / 43 / 78 for size 2, and of 24 / 24 / 45 for size 3. As per Table 4, the differences between all pairs of model types (except for the pair (*QDM*, *FDM*) for size 3) are statistically significant, albeit the effect sizes between the abstract models are only minor (size 2) and negligible (size 1). So, even the comparably simple abstract models could, on average, reduce the search space for diagnoses by almost one quarter to almost three quarters for the different diagnosis cardinalities, a fair result. Depending on the diagnostic strategies relying on this search space restriction, the employment of the discussed models can thus imply efficiency improvements of up to almost 100 % (*VBM*, single-faults) and almost 80 % (*VBM*, double-faults), while taking a median of 1 second and a maximum of less than 10 minutes (see above), times one would probably often be ready to invest for the substantial pruning of the search space achieved. As a result, we see model-based diagnosis to be a potentially very powerful tool to boost other techniques that can profit from a sound and efficient limitation of the (theoretical) diagnostic hypotheses space, such as SFL.

Trade-off between efficiency and accuracy (Fig. 16, Table 4) Studying Fig. 16, we note very favorable time-accuracy trade-offs for both abstract model types. The medians for *FDM* / *QDM* amount to 17 / 16 for size 1, to 11 / 9 for size 2, and to 264 / 266 for size 3, where all differences are statistically significant. However, the effect sizes are very small, so that it is reasonable to say that both model types provide more or less the same efficiency-accuracy trade-off. On the other hand, the differences between the trade-off manifested by any abstract model and the exact one is statistically highly significant for all diagnosis sizes. That also the effect sizes are considerable, is well visible in Fig. 16. Remarkably, we do not even observe a single case with an unfavorable time-accuracy trade-off for *QDM* and *FDM* for diagnosis cardinality 3, and only very few for cardinality 1. That is, for these cases, switching from an abstract model to the *VBM* nearly always brings a smaller relative reduction of the set of generated diagnoses than the relative time overhead it involves. In comparison with the real-world spreadsheets, where a higher diagnosis cardinality meant a worse trade-off, we now find that the (by far) best trade-off is achieved for size 3. This suggests that we cannot generally relate the cardinality with the goodness of the efficiency-accuracy ratio. What is largely consistent with the findings for the real-world spreadsheets is the close similarity between

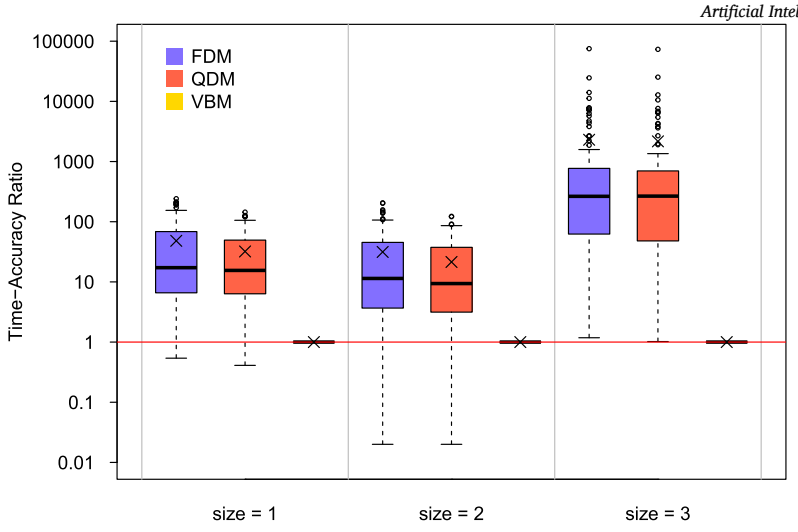


Fig. 16. Artificial Spreadsheets: Boxplot summaries of the **trade-off between time savings and diagnostic accuracy** (cf. Definition 25) of the model types compared to the *VBM* for different cardinalities (sizes 1, 2, 3). The values for *VBM*, which are trivially always 1, are included for comparison. The red horizontal line highlights the areas of a favorable (above the line) vs. an unfavorable (below the line) time-accuracy trade-off achieved by a model type. Included in the analysis are only those cases where no timeout was triggered for *VBM*, i.e., where an output was successfully generated for all model types. Medians are indicated by a horizontal line, means by a cross. Note the logarithmic scale on the y-axis.

the median trade-offs achieved by both abstract model types; however, the variability in the values is notably lower for the artificial spreadsheets. In short, both abstract model types overall lead to a significantly better trade-off between computation time and output quality than the exact model, and both are a more or less equally good choice based on this criterion.

6.5. Threats to validity and research limitations

We identify the following potential threats to the validity of the experimental findings reported in this section. In terms of *internal validity*, one possible danger is that the procedures to extract the models from the spreadsheets may contain programming errors. Given that these procedures are non-trivial, we thoroughly tested our code during and after the development and corrected any errors we detected. Furthermore, we see the fact that the experimental results align with the presented and proven theoretical foundations as another strong indicator of the correctness of our software implementation. We also note that we used the same constraint solver for consistency checking, relied on the same implementation of the diagnosis algorithm for all examined models, and used the same type of constraints for implementing *FDM* and *QDM*.[22] This rules out the possibility that the observed performance differences are only the result of model-specific implementations of the underlying reasoning components. Finally, it is important to note that all experiments were conducted on the same computer. Thus, the results are all based on an identical hardware and software configuration, ruling out the risk that differences in the observed results are due to the execution environment.

Considering the *external validity* of our findings, one main potential threat may be the choice of the set of spreadsheets that we used in our experiments. To minimize the risk that our results are specific to a certain class of spreadsheets, we conducted experiments with two corpora of spreadsheets. One corpus is a pre-existing and publicly available one containing real-world spreadsheets; the other contains artificially-created spreadsheets with a structure commonly found in real-world spreadsheets. Since the obtained results for these sufficiently different corpora are by and large aligned, we consider the risk that the reported findings strongly depend on the underlying characteristics of the spreadsheets to be low.

In terms of *research limitations*, we recall that the modeling approaches presented and evaluated in this work so far focus on spreadsheets that use integer computations. Furthermore, we iterate that the theory of spreadsheets, cells, and formulas that we use as a basis for our research (cf. Section 2) focuses on a restricted, yet relevant, part of the common functionality of modern spreadsheets. Future research is therefore required to examine the properties of different modeling approaches in the presence of a further enhanced underlying spreadsheet "language". A more specific technical limitation of our experimental evaluations is that we had to implement *timeouts* to ensure that the time needed to complete the experiments remains within manageable bounds. The specific choices of the timeouts may, to a certain extent, influence both the observed average and median values and the reported ratios. However, we see no indications that different timeouts would largely impact the general findings of our experiments.

[22] The programming language provided by the used constraint solver Minion v1.8 allows for implementing integer operations directly. For the abstract models, *FDM* and *QDM*, we relied on table constraints.

7. Related research

Automated debugging, which is the focus of our work, mainly began in the early 1980s. In a seminal work of that time, Shapiro [77], for example, introduced an algorithm that guides the user when searching for bugs in Prolog programs. Around the same time, Weiser proposed the concept of *program slicing* [85,86] and corresponding algorithms for computing *slices*, which are subsets of programs that lead to unexpected values for certain variables. In our own research, we follow an alternative approach to automated debugging based on model-based diagnosis techniques, which was originally developed—also in the 1980s—for locating faults in hardware, see [14] and [58]. Console et al. [12] were the first authors to apply model-based diagnosis to software, particularly logic programs. Their results showed that the model-based approach is superior to the earlier method by Shapiro, and Bond [8] subsequently further improved the work of Console et al. [12]. Later on, given these positive results, researchers applied the same principles to locate faults in programs written using sequential, concurrent, and functional programming languages [25,84,48].

An early approach to applying model-based diagnosis techniques for spreadsheet debugging was proposed by Jannach and Engler [38], who, like our present work, relied on constraint solving as an underlying reasoning technique. Later, Jannach and Schmitz [39] extended this work by developing an add-on for Microsoft Excel called the EXQUISITE debugging tool. A related constraint-based approach was also developed by Abreu et al. [2]. A particularity of their work is that their approach relies on a single test case at a time and directly encodes the reasoning about the correctness of the cells into a CSP. An alternative, dependency-based model for diagnosis-based spreadsheet debugging was later proposed in [36] and [31]. In these works, it was also shown how the performance of such models could be further improved.

Besides model-based diagnosis techniques, various other fault localization and debugging techniques were proposed for the case of spreadsheet programs, cf. [40] for a survey on the topic. In many cases, ideas and concepts previously applied to debug general software (e.g., programs written in imperative languages) were adapted to the particularities of spreadsheets.[23] Reichwein et al. [57], for example, adapted program slicing to be used for fault localization in spreadsheets, and Ruthruff et al. [73] proposed to use Tarantula, a Spectrum-based Fault Localization (SFL) technique, for debugging spreadsheets. Later on, Hofer et al. [35] empirically evaluated the performance of different similarity coefficients when applying SFL on spreadsheets.

While SFL-based techniques are commonly evaluated in an "offline" manner, i.e., with the help of computational experiments and metrics and without involving users, there are also several research works on spreadsheet debugging that address also the user interaction perspective, such as the above-mentioned EXQUISITE add-on to Microsoft Excel. Notable works in this context include *GoalDebug*, a debugger that not only locates faults but also computes repair suggestions for faulty spreadsheets [1]. In another work, Ruthruff et al. [72] introduced a technique for localizing faults in spreadsheet environments *in an interactive way*, which is of particular interest because of the necessity to reduce the number of generated diagnosis candidates. Also, general sequential diagnosis methods, such as [68,71], can be adopted for interactive spreadsheet debugging.

In the context of tools, it is worth mentioning that several existing approaches to locate faults in spreadsheets rather rely on static analyses of a given spreadsheet than on evaluating its functions as we do. These static analyses frequently target at the identification of *spreadsheet smells*, which can be seen as structural peculiarities of a spreadsheet program that are assumed to often lead to a faulty behavior, see [19,37,42,45]. Specific tools that use such analysis or smells include CheckCell ([6]), CUSTODES ([9]), CACheck ([20]), and ExceLint ([7]). In contrast to these works, we do not rely on such structural peculiarities but focus on models that represent the structure and the behavior of spreadsheets directly.

Going back to the more general principles of model-based debugging, we note that the idea of using abstraction for model-based diagnosis is not new. Early foundations in that area were established, for example, in [52], and Autio and Reiter [5] discussed the concept of structural abstraction, where sets of interconnected components are mapped to one component. When using such an abstraction, the behavior of such a component is then defined based on the behavior of the components it represents. Questions of modeling for diagnosis using abstractions and refinement are discussed in detail in [5] and [82], and an iterative refinement approach for diagnosing larger knowledge bases was proposed in [23]. Similarly, Mayer [49] utilized abstract interpretation (see [13]) for model-based fault localization in programs.

An alternative abstraction approach was introduced in [75,76], where quantitative domains are mapped to qualitative ones considering value boundaries that influence the behavior of the system. These boundaries however depend on the given diagnosis problem. Therefore, the authors suggested using an automated abstraction approach for solving this issue. In yet another approach, Struss [83] discussed the use of deviation models for the diagnosis of hardware. In contrast to these previous works, we focus on deviation models for the diagnosis of spreadsheets, and we furthermore develop a theory that allows us to compare modeling approaches using different levels of abstraction.

In the context of abstraction levels, note also that exact value-based models are not necessarily the least abstract model type one might think of. In fact, instead of defining the spreadsheet cells to be the atomic components of interest to a debugging task, one might pursue the more ambitious goal of not only locating which cells are faulty, but of additionally determining which parts of the cell contents are wrong, and how to fix them. To this end, a similar technique as the one presented in [26] could be adopted for spreadsheets as well.

Finally, note that this paper extends our previous work [33] by *(1)* describing, analyzing and contrasting in detail *three* different model types for spreadsheet debugging, *(2)* presenting a revised conversion algorithm that allows to compile spreadsheets into all

[23] A survey on general software fault localization techniques can be found in [87]. Note also that spreadsheets can be seen as a particular form of software, which means that many general software engineering approaches can be applied to them as well [30].

three model types, *(3)* introducing a formal framework that enables the derivation of diagnostic accuracy relationships between the model types, and by *(4)* substantially extending the experimental evaluation.

8. Conclusions

In this work, we theoretically and empirically analyze three types of (constraint) models for model-based spreadsheet diagnosis, one exact model (value-based model–*VBM*) capturing the precise semantics of the spreadsheet, and two abstract models. One of the latter allows to reason on the level of cell (in)correctness instead of actual cell values (functional dependency model–*FDM*), whereas the second additionally captures two types of incorrectness for numerical cells in terms of the predicates "too high" and "too low" (qualitative deviation model–*QDM*). We also present an algorithm to automatically extract models of any of the discussed types directly from a given buggy spreadsheet.

Apart from the pragmatic attractiveness of abstract models insofar as they can incorporate intuitive and less precise qualitative information for debugging purposes and do not depend on the ability of the user(s) to specify observations (i.e., intended values for selected cells) on a very fine-grained quantitative level, we show that they are complete by always allowing to localize the true diagnosis. However, we also find that a higher degree of abstraction implies a lower or at best an equal diagnostic accuracy. In other words, abstract models are generally unsound w.r.t. diagnosis computation; the more the semantics of the used model type deviates from the precise semantics, the more spurious diagnoses can be the result. On the other hand, the price to pay for a sound and complete diagnostic output when using the exact model is a generally degraded computational performance due to the higher expressiveness of the spreadsheet representation which can largely affect the efficiency of the model-based reasoning.

In comprehensive empirical evaluations on both a real-world and a synthetic dataset comprising 370 spreadsheets and involving more than 5 500 performed diagnosis computation tasks, we investigate the significance and scale of the superiority relationships between the three model types in terms of their diagnostic accuracy and computational efficiency.

Our main insights concerning efficiency are: *(1)* All model types manifest reasonable absolute median and average computation times, below one minute, to generate all single-, double- and triple-fault diagnoses, respectively, which proves that model-based diagnosis can be a practical approach for spreadsheet debugging. *(2)* Due to the more sophisticated reasoning it involves, the exact model is between three times and up to more than four orders of magnitude slower than the two abstract models, manifests impractical times in a number of cases, and cannot successfully complete about 1 000 (almost 20 %) of the run diagnosis generation tasks due to a computation timeout. This clearly demonstrates that, in order to leverage the benefits of model-based diagnosis in terms of the deterministic, precise and provable diagnostic conclusions it allows, we cannot always draw on fully detailed models, but *have to* exploit more abstract models. *(3)* The abstract model types are always highly performant, and never require more than 3.1 seconds to determine all diagnoses of a given cardinality, even for the hardest cases where the exact model is not applicable.

In terms of accuracy, our study provides the following insights: *(1)* All model types yield a reasonably small (median) number of cardinality-one diagnoses (no more than a dozen) which even allows for a manual inspection of the fault hypotheses in case a single-fault is conjectured. *(2)* In terms of double- and triple-fault diagnoses, we observe significantly higher numbers of computed diagnoses for all model types. That is, overall, in more than 7 of 10 cases for double-faults, and in 98 of 100 cases for triple-faults, the number of diagnoses exceeds the reasonable load (a dozen) for manual inspection even for the *VBM* (ground truth), and much worse for the abstract model types. This indicates that, in case a multiple-fault is surmised, none of the model types can be expected to produce an acceptably small output, so that it can be well-advised to combine the mere model-based diagnosis computation by additional information acquisition or ranking techniques that enable to reduce the number of output fault hypotheses or point the user to the most plausible ones. *(3)* In terms of the differences between the models, we reveal that the (median) number of produced diagnoses is always within one order of magnitude for all model types, where the exact model type generates between 8 % and 92 % fewer diagnoses than the more accurate abstract model type *QDM* for the executed diagnosis computation tasks, and the latter leads to a reduction in the number of output diagnoses between 0 % and 25 % compared to the least sophisticated model type *FDM*. That is, whenever a more detailed model is affordable from the computational perspective, it is worthwhile using it as a notable improvement in diagnostic accuracy can be expected. *(4)* The exact model type, however, is not always (strictly) preferable to the abstract ones in terms of diagnostic accuracy. In fact, we record that the *QDM* is as accurate as the exact model in a significant fraction of up to 52 % of the cases per computation task, whereas also the simplest model type under test, the *FDM*, reaches very good percentages of up to 48 % in this regard; notably, while both abstract model types outperform the exact one by orders of magnitude as regards efficiency. The two abstract model types lead to the same diagnostic output in at least 76 % and up to 95 % of the cases for the various computation tasks run in our experiments. These findings show that abstract model types can be practical and powerful surrogates of an exact model in scenarios where the latter is not (efficiently) applicable.

In addition, our theory and results indicate that model-based reasoning can be an impactful mechanism to *provably* prune the search space for alternative diagnostic techniques such as SFL, with achieved median search space reductions of up to 98 % achieved by the exact model, and even up to 70 % for both abstract model types. Finally, an assessment of the abstract model types regarding the ratio between the factor of less time required than the exact model and the factor of more diagnoses generated than the exact model, brings to light that abstract types almost always lead to a favorable trade-off between the achieved savings in computation time and the incurred expansion of the solution set, and that up to an average of more than 3 000 times more time is saved than there are more diagnoses generated when using an abstract model type instead of the exact one.

Future work could explore several avenues, including analyzing abstraction techniques within a more comprehensive spreadsheet language and investigating their applicability to other software types and model-based diagnosis domains. Further research should also study the benefits of integrating these methods with statistical debugging and sequential diagnosis techniques. Develop-

ing approaches that combine abstract and exact models to achieve a balance between diagnostic efficiency and accuracy is another promising direction. Finally, incorporating probabilistic information into *FDM* and *QDM* to prioritize the most statistically plausible diagnoses—for example, by accounting for the low likelihood of multiple faulty inputs resulting in a correct output—warrants investigation.

CRediT authorship contribution statement

Patrick Rodler: Writing – original draft, Writing – review & editing, Visualization, Validation, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Birgit Hofer:** Validation, Software, Resources, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Dietmar Jannach:** Supervision, Project administration, Funding acquisition. **Iulia Nica:** Validation, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Franz Wotawa:** Writing – original draft, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/P32445, and by the Austrian Research Promotion Agency (FFG), contract number FO999910235. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission.

Appendix A. Additional explanations

A.1. Functional dependency model (FDM)

A.1.1. Detailed explanation of Fig. 2

Consider the cell constraint $\text{cCONSTR}_{FDM}(c)$ given that $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$ in Sec. 3.3.3, and assume that c is normal. Then:

- If $\odot = +$:
  - If both operands v_1, v_2 have the correct value ($\top$), then $v_1 + v_2$ must also have the correct value ($\top$).
  - If exactly one of the operands v_1, v_2 is incorrect ($\bot$), then $v_1 + v_2$ must also be incorrect ($\bot$).
  - If both operands v_1, v_2 are incorrect ($\bot$), then $v_1 + v_2$ will generally be incorrect ($\bot$), but will be correct ($\top$) in cases where v_1 deviates positively/negatively from its correct value by the same amount as v_2 deviates negatively/positively from its correct value, e.g., v_1 is by 1 too large whereas v_2 is by 1 too small.
- If $\odot = -$:
  - If both operands v_1, v_2 have the correct value ($\top$), then $v_1 - v_2$ must also have the correct value ($\top$).
  - If exactly one of the operands v_1, v_2 is incorrect ($\bot$), then $v_1 - v_2$ must also be incorrect ($\bot$).
  - If both operands v_1, v_2 are incorrect ($\bot$), then $v_1 - v_2$ will generally be incorrect ($\bot$), but will be correct ($\top$) in cases where v_1 deviates positively/negatively from its correct value by the same amount as v_2 deviates positively/negatively from its correct value, e.g., both v_1 and v_2 are by 1 too large.
- If $\odot = *$:
  - If both operands v_1, v_2 have the correct value ($\top$), then $v_1 * v_2$ must also have the correct value ($\top$).
  - If exactly one of the operands v_1, v_2 is incorrect ($\bot$), then $v_1 * v_2$ will generally be incorrect ($\bot$), but will be correct ($\top$) in case the correct ($\top$) operand is equal to 0.
  - If both operands v_1, v_2 are incorrect ($\bot$), then $v_1 * v_2$ will generally be incorrect ($\bot$), but will be correct ($\top$) in cases where v_1 deviates by a factor x from its correct value whereas v_2 deviates by a factor $1/x$ from its correct value, e.g., v_1 is double its correct value whereas v_2 is half its correct value.
- If $\odot = /$:
  - If both operands v_1, v_2 have the correct value ($\top$), then v_1/v_2 must also have the correct value ($\top$).
  - If the dividend v_1 is incorrect ($\bot$) and the divisor v_2 is correct ($\top$), then v_1/v_2 must be incorrect ($\bot$). That is, for any fixed value of v_2, for any variation of v_1, the quotient will change.[24]
  - If the dividend v_1 is correct ($\top$) and the divisor v_2 is incorrect ($\bot$), then v_1/v_2 will generally be incorrect ($\bot$), but will be correct ($\top$) in case the dividend v_1 is equal to 0.

[24] Note, for simplicity we do not cover the case of error values resulting from exceptions like 'division by zero'.

– If both operands v_1, v_2 are incorrect ($\bot$), then v_1/v_2 will generally be incorrect ($\bot$), but will be correct ($\top$) in cases where v_1 deviates by the same factor x from its correct value as v_2 does from its correct value, e.g., both v_1 and v_2 are double their correct values.

A.1.2. Detailed explanation of Fig. 3

Consider the constraint $cons_C$ in Sec. 3.3.3. Then:

- If both operands (cells c_1, c_2) are correct ($v_1, v_2 = \top$), then the truth value of $c_1 \square c_2$ must be correct as well ($C_c = \top$).
- If at least one operand (cell among c_1, c_2) is incorrect ($v_1 = \bot \vee v_2 = \bot$), then the truth value of $c_1 \square c_2$ can be correct or incorrect ($C_c \in \{\top, \bot\}$). E.g., for $\square$ equal to $<$, given that the truth value is `false` in case both operands are correct (i.e., $c_1 \geq c_2$ holds), then the truth value remains correct ($C_c = \top$) in case, e.g.,
  – only c_1 is increased (case: v_1 is $\bot$, v_2 is $\top$),
  – only c_2 is decreased (case: v_1 is $\top$, v_2 is $\bot$), or
  – both c_1 is increased and c_2 is decreased (case: both v_1 and v_2 are $\bot$).
  but the truth value becomes incorrect ($C_c = \bot$) in case, e.g.,
  – only c_2 is increased to become larger than c_1 (case: v_1 is $\top$, v_2 is $\bot$),
  – only c_1 is decreased to become smaller than c_2 (case: v_1 is $\bot$, v_2 is $\top$), or
  – both c_1 is decreased and c_2 is increased such that $c_1 < c_2$ (case: both v_1 and v_2 are $\bot$).
  A similar argumentation applies to the other operands $>, =$.

A.1.3. Detailed explanation of Eq. (2)

Consider the constraint $cons_c$ in Eq. (2) in Sec. 3.3.3, and assume that c is normal. Then:

- If the condition is correct ($C_c = \top$), then c takes the value of the designated cell among c_3, c_4:
  – If both c_3 and c_4 have the same (in)correctness value ($\top/\bot$), then the designated cell, whichever it is, and thus c, must have the same value ($\top/\bot$).
  – If some cell among c_3, c_4 is correct ($\top$) and the other is incorrect ($\bot$), then the correctness of c depends on the (in)correctness value ($\top$ or $\bot$) of the designated cell, i.e., c can be $\top$ or $\bot$.
- If the condition is incorrect ($C_c = \bot$), then c takes the value of the undesignated cell among c_3, c_4. Then, regardless of the values ($\top/\bot$) of the cells c_3, c_4, both values $\top$ and $\bot$ are possible for c. Specifically, if the value, whether correct ($\top$) or incorrect ($\bot$), of the undesignated cell among c_3, c_4 happens to coincide with the correct value of the designated cell among c_3, c_4, then c is correct ($\top$), else c is incorrect ($\bot$).

A.2. Qualitative deviation model (QDM)

A.2.1. Detailed explanation of Fig. 4

Consider the cell constraint $\textsc{cConstr}_{QDM}(c)$ given that $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$ in Sec. 3.4.3, and assume that c is normal. Then:

- If $\odot = +$:
  – If one operand among v_1, v_2 is correct ($=$), then the deviation of cell c corresponds to the deviation of the other operand, i.e., if the other operand is too high ($>$) / correct ($=$) / too low ($<$), then the value of c will be too high ($>$) / correct ($=$) / too low ($<$).
  – If both operands v_1, v_2 deviate from their correct value, then the value of c is too high ($>$) if both operands are too high ($>$), too low ($<$) if both operands are too low ($<$), and unknown, i.e., an element of $\{<, =, >\}$, if one operand is too high ($>$) whereas the other is too low ($<$).
- If $\odot = -$:
  – If v_2 is correct ($=$), then the deviation of cell c corresponds to the deviation of v_1, i.e., if v_1 is too high ($>$) / correct ($=$) / too low ($<$), then the value of c will be too high ($>$) / correct ($=$) / too low ($<$).
  – If v_1 is correct ($=$), then the deviation of cell c is inverse to the deviation of v_2, i.e., if v_2 is too high ($>$) / correct ($=$) / too low ($<$), then the value of c will be too low ($<$) / correct ($=$) / too high ($>$).
  – If both operands v_1, v_2 deviate from their correct value, then the value of c is *(i)* too high ($>$) if both v_1 is too high ($>$) and v_2 is too low ($<$), *(ii)* too low ($<$) if both v_1 is too low ($<$) and v_2 is too high ($>$), and *(iii)* unknown, i.e., an element of $\{<, =, >\}$, if the values v_1, v_2 are both too high ($>$) or both too low ($<$).
- If $\odot = *$: If one or both of the operands v_1, v_2 are incorrect ($<$ or $>$), then the deviation of cell c can be any among $\{<, =, >\}$. This can be seen by the following observations:
  – Let one operand, say w.l.o.g. v_1, be correct ($=$) and
    * positive. Then, the value of c will be *(i)* too high ($>$) if the value of the other operand (v_2) is too high ($>$), and *(ii)* too low ($<$) if the value of the other operand (v_2) is too low ($<$),
    * zero. Then, the value of c will be correct regardless of the deviation of the other operand (v_2).

  - $*$ negative. Then, the value of c will be *(i)* too low ($<$) if the value of the other operand (v_2) is too high ($>$), and *(ii)* too high ($>$) if the value of the other operand (v_2) is too low ($<$),
  - Let both operands be too high ($>$). Then, the value of c will be *(i)* too high ($>$) if both the correct and the incorrect values of both operands are positive, *(ii)* correct ($=$) if, e.g., the correct value of one operand as well as the incorrect value of the other operand are 0, and *(iii)* too low ($<$) if, e.g., both correct operands are negative, one of the incorrect operands is positive while the other is negative.
  - Let both operands be too low ($<$). Then, the value of c will be *(i)* too low ($<$) if both the correct and the incorrect values of both operands are positive, *(ii)* correct ($=$) if, e.g., the correct value of one operand as well as the incorrect value of the other operand are 0, and *(iii)* too high ($>$) if, e.g., one of the correct operands is positive while the other is negative, and both incorrect operands are negative.
  - Let one operator be too low ($<$) and the other be too high ($>$). Then, no general conclusions can be drawn about the value of c, i.e., its deviation can be any among $\{<,=,>\}$. Whereas the possibility of too low ($<$) and too high ($>$) values of c are obvious, a correct value will be the result, e.g., if the correct values of the operands are x and $-x$, and their incorrect values are $-x$ and x, respectively.
- If $\odot = /$: Here, $v_2 \neq 0$ holds by assumption (cf. Sec. 3.1) and the argumentation is analogous to the case $\odot = *$ (by viewing v_1/v_2 as $v_1 * \frac{1}{v_2}$), except for the cases where v_1 is incorrect ($<$ or $>$) and v_2 is correct ($=$). Unlike for $\odot = *$, the value of c cannot be correct ($=$) in this scenario. To see this, assume that $v_1/v_2 = v_1^*/v_2$ where v_1^* denotes the correct value of the operand v_1 and $v_1 \neq v_1^*$. Then, a rearrangement yields $v_2 * (v_1^* - v_1) = 0$ which implies that $v_2 = 0$, a contradiction.

A.2.2. Detailed explanation of Fig. 5

Consider the constraint $cons_C$ in Sec. 3.4.3. Then: For all operators $\square \in \{<,=,>\}$, we have that *(i)* the condition necessarily evaluates to the correct truth value ($\top$) only if both operands v_1, v_2 are correct ($=$), and *(ii)* can evaluate to both the correct ($\top$) or an incorrect ($\bot$) truth value in all other cases. The reason for this is that, for any operator $\square \in \{<,=,>\}$ and any assumed deviation of cells c_1, c_2 (i.e., $v_1, v_2 \in \{<,=,>\}$), there are possible cases where the truth value of the condition is changed through the deviation, and ones where the truth value remains constant despite the deviation. E.g., let $\square = >$, $v_1 = <$ and $v_2 = >$, then, assuming the correct cell values of 4 and 1 for cells c_1 and c_2, a deviation resulting in the values 3 and 2 will not change the correct truth value of the condition $c_1 > c_2$, whereas a deviation resulting in the values 1 and 2 will do so; i.e., both values $\top$ (truth value unchanged) and $\bot$ (truth value changed) are possible for the condition (variable C_c). Similar examples can be found for all other combinations of operator and cell value deviations.

A.2.3. Detailed explanation of Eq. (3)

Consider the constraint $cons_c$ in Eq. (3) in Sec. 3.4.3, and assume that c is normal. Then:

- If the condition is correct ($C_c = \top$), then c takes the value of the designated cell among c_3, c_4. Its deviation value thus equals the deviation value of either c_3 or c_4. In the special case where both c_3 and c_4 have the same deviation value ($v_3 = v_4$), this will be also the deviation value of cell c.
- If the condition is incorrect ($C_c = \bot$), then c takes the value of the undesignated cell among c_3, c_4 and, regardless of the deviation ($</=/>$) of each of the cells c_3, c_4, all deviations among $<,=,>$ are possible for c. Specifically, if the value of the undesignated cell among c_3, c_4, regardless of its deviation ($</=/>$), happens to coincide with the correct value of the designated cell among c_3, c_4, then c is correct ($=$), else the deviation of c is either $<$ or $>$.

Appendix B. Constraints for known/assumed correct cells

B.1. Value-based model (VBM)

Let $c \in \Sigma \setminus \Sigma'$ (*c is known/assumed correct*). Then the output of $\text{CONSTR}_{VBM}(c)$ for the different cases regarding the formula $f(c)$ in c is:
(*Explanation:* The following constraints are the same as the ones for the case $c \in \Sigma'$ delineated in Sec. 3.2.3 with ab_c set to `false`, with the difference that there is no ab_c variable here since the cells are known correct.)

- If $f(c)$ is a number n:
  $\text{CONSTR}_{VBM}(c) = \{((c'), \{(n)\})\}$
- If $f(c)$ is a reference to cell $\tilde{c}$:
  $\text{CONSTR}_{VBM}(c) = \{((c', \tilde{c}'), \{(v, v) | v \in DOM(\tilde{c}')\})\}$
- If $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$:
  $$\text{CONSTR}_{VBM}(c) = \left\{ \begin{array}{l} ((c_1', c_2', c'), \\ \{(v_1, v_2, v_1 \odot v_2) \mid \\ v_1 \in DOM(c_1'), v_2 \in DOM(c_2')\})\} \end{array} \right\}$$
- If $f(c)$ is of the form $\texttt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<,=,>\}$:
  $\text{CONSTR}_{VBM}(c) = \{cons_C, cons_c\}$ where $cons_C$ is as described in Sec. 3.2.3 and

$$cons_c := \left(\begin{array}{l} (C_c, c'_3, c'_4, c'), \\ \{(\mathsf{true}, v_3, v_4, v_3) \mid v_3, v_4 \in DOM(c')\} \cup \\ \{(\mathsf{false}, v_3, v_4, v_4) \mid v_3, v_4 \in DOM(c')\} \end{array} \right)$$

B.2. Functional dependency model (FDM)

Let $c \in \Sigma \setminus \Sigma'$ (*c is known/assumed correct*). Then the output of $\textsc{cConstr}_{FDM}(c)$ for the different cases regarding the formula $f(c)$ in c is:
(*Explanation:* The following constraints are the same as the ones for the case $c \in \Sigma'$ delineated in Sec. 3.3.3 with ab_c set to false, with the difference that there is no ab_c variable here since the cells are known correct.)

- If $f(c)$ is a number n:
  $\textsc{cConstr}_{FDM}(c) = \{((c'), \{(\top)\})\}$
- If $f(c)$ is a reference to cell $\tilde{c}$:
  $\textsc{cConstr}_{FDM}(c) = \{((c', \tilde{c}'), \{(v, v) \mid v \in \{\top, \bot\}\})\}$
- If $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$:
  $$\textsc{cConstr}_{FDM}(c) = \left\{ \begin{array}{l} ((c'_1, c'_2, c'), \\ \{(v_1, v_2, v) \mid v \in \textsc{val}(\odot, v_1, v_2) \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\})\} \end{array} \right\}$$
  where $\textsc{val}(\square, v_1, v_2)$ is defined in Fig. 3.
- If $f(c)$ is of the form $\mathtt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<, =, >\}$:
  $\textsc{cConstr}_{FDM}(c) = \{cons_C, cons_c\}$ where $cons_C$ is as described in Sec. 3.3.3 and

$$cons_c := \left(\begin{array}{l} (C_c, c'_3, c'_4, c'), \\ \{(\top, v, v, v) \mid v \in \{\top, \bot\}\} \cup \\ \{(\top, v_3, v_4, v) \mid v_3, v_4 \in \{\top, \bot\}, v_3 \neq v_4\} \cup \\ \{(\bot, v_3, v_4, v) \mid v_3, v_4, v \in \{\top, \bot\}\} \end{array} \right)$$

B.3. Qualitative deviation model (QDM)

Let $c \in \Sigma \setminus \Sigma'$ (*c is known/assumed correct*). Then the output of $\textsc{cConstr}_{QDM}(c)$ for the different cases regarding the formula $f(c)$ in c is:
(*Explanation:* The following constraints are the same as the ones for the case $c \in \Sigma'$ delineated in Sec. 3.4.3 with ab_c set to false, with the difference that there is no ab_c variable here since the cells are known correct.)

- If $f(c)$ is a number n:
  $\textsc{cConstr}_{QDM}(c) = \{((c'), \{(=)\})\}$
- If $f(c)$ is a reference to cell $\tilde{c}$:
  $\textsc{cConstr}_{QDM}(c) = \{((c', \tilde{c}'), \{(v, v) | v \in DOM(\tilde{c}')\})\}$
- If $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+, -, *, /\}$:
  $$\textsc{cConstr}_{QDM}(c) = \left\{ \begin{array}{l} ((c'_1, c'_2, c'), \\ \{(v_1, v_2, v) \mid v \in \textsc{val}(\odot, v_1, v_2), \\ v_1 \in DOM(c'_1), v_2 \in DOM(c'_2)\} \end{array} \right\}$$
  where $\textsc{val}(\odot, v_1, v_2)$ is defined in Fig. 4.
- If $f(c)$ is of the form $\mathtt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<, =, >\}$:
  $\textsc{cConstr}_{QDM}(c) = \{cons_C, cons_c\}$ where $cons_C$ is as described in Sec. 3.4.3 and

$$cons_c := \left(\begin{array}{l} (C_c, c'_3, c'_4, c'), \\ \{(\top, v_3, v_4, v) \mid v_3, v_4 \in \{<, =, >\}, v = v_3 \vee v = v_4\} \cup \\ \{(\bot, v_3, v_4, v) \mid v_3, v_4, v \in \{<, =, >\}\} \end{array} \right)$$

Appendix C. Proofs

C.1. Proof of Theorem 1

Proof. We prove the theorem by contradiction. To this end, we assume that *(1)* $P_2 \prec_{abs} P_1$ is true, but *(2)* $P_1 \preceq_{acc} P_2$ does not hold. By Definition 16, (2) is equivalent to $\mathbf{AllD}_{P_1} \not\supseteq \mathbf{AllD}_{P_2}$. That is, there is some diagnosis $\Delta \in \mathbf{AllD}_{P_2}$ such that $\Delta \notin \mathbf{AllD}_{P_1}$. Hence, by Definition 13, given $CONS_\Delta = \{((ab_c), \{(\mathsf{true})\}) \mid c \in \Delta\} \cup \{((ab_c), \{(\mathsf{false})\}) \mid c \in COMP \setminus \Delta\}$, the constraint system $(VARS, DOM_2, CONS_2 \cup COBS_2 \cup CONS_\Delta)$ is satisfiable. This implies by Definition 11 that there is a valid assignment VA for the constraint system $(VARS, DOM_2, CONS_2 \cup COBS_2 \cup CONS_\Delta)$. Due to Definitions 7 and 9, this means that VA includes one pair (x, v) for

each $x \in VARS$, assigning x to some element of $DOM_2(x)$ such that each constraint in $CONS_2 \cup COBS_2 \cup CONS_\Delta$ is satisfied. According to Definition 8, this means that, for any constraint $c = (scope(c), tl(c)) \in CONS_2 \cup COBS_2 \cup CONS_\Delta$ with $scope(c) = (x_1, \ldots, x_k)$, it holds that $tl(c)$ includes the tuple of values $(v_1, \ldots, v_k)$ where v_i is the value assigned to the variable x_i by VA.

Now, consider an arbitrary constraint $c \in CONS_2 \cup COBS_2$ and the unique constraint $c' = \text{SAMESCOPE}(c)$ from $CONS_1 \cup COBS_1$ with the same scope as c (cf. Remark 4). Then, due to (1), i.e., $P_2 \prec_{abs} P_1$, and Definition 14, there is a function h_x for each variable $x \in VARS$ which maps each value from $DOM_2(x)$ to $DOM_1(x)$ in a way that $c' = h(c) = h(scope(c), tl(c)) = (scope(c), h(tl(c))) = (scope(c), \{(h_{x_1}(u_1), \ldots, h_{x_k}(u_k)) \mid (u_1, \ldots, u_k) \in tl(c)\})$. Due to the already derived fact that $(v_1, \ldots, v_k) \in tl(c)$, we deduce that $(h_{x_1}(v_1), \ldots, h_{x_k}(v_k)) \in tl(c')$. Because c was chosen arbitrarily, the same holds true for any constraint $c \in CONS_2 \cup COBS_2$ and its associated constraint $c' = \text{SAMESCOPE}(c) \in CONS_1 \cup COBS_1$. Since SAMESCOPE is a bijective mapping from $(CONS_2 \cup COBS_2)$ to $(CONS_1 \cup COBS_1)$, we obtain that the value assignment $VA_h := \{(x, h_x(v)) \mid (x, v) \in VA\}$ is a valid assignment for the constraint system $(VARS, DOM_1, CONS_1 \cup COBS_1)$. However, as by Definition 14, $h_x(v) = v$ for all $x \in \{ab_c \mid c \in COMP\}$, and as $COMP$ is the same for P_1 and P_2, we obtain that VA_h is also a valid assignment for the constraint system $(VARS, DOM_1, CONS_1 \cup COBS_1 \cup CONS_\Delta)$. Consequently, $\Delta \in \mathbf{AllD}_{P_1}$, which is a contradiction to our assumption. This concludes the proof. □

C.2. Proof of Lemma 1

Proof. $CS^*_{VBM}(\Sigma, \Sigma')$ is a constraint representation of the spreadsheet Σ by means of the value-based model, as specified in Sec. 3.2.3. On this basis, a simple inductive argument allows us to prove that the value of the variable $c' \in VARS$ (and, if applicable, the value of the variable $C_c \in VARS$) is unique for each cell $c \in \Sigma$. To this end, let us define the precedence index of a cell variable c' (or C_c) for a cell $c \in \Sigma$ as $pre(c') := 0$ if c is an input cell (i.e., $\chi(c) = n$ for a number n), and as $pre(c') := \max_{\{c_i \mid c_i \text{ is referenced by } c\}}(pre(c'_i)) + 1$ (or $pre(C_c) := \max_{\{c_i \mid c_i \text{ occurs in the first entry of the } \texttt{if}\text{-expression in } c\}}(pre(c'_i)) + 1$) for all non-input cells (i.e., $\chi(c) = f$ for a spreadsheet formula f), cf. Sec. 2.1. Intuitively, this means that all input cell variables have a precedence index of zero, all cell variables that can be evaluated using only the values of input cell variables have a precedence index of one, all the cell variables that can be evaluated using the values of cell variables with precedence indices one or zero have a precedence index of two, and so on. Note that the precedence index is uniquely defined since circular references (where the value of one cell directly or indirectly depends on itself) are forbidden in spreadsheets. The following induction proof is based on the precedence index of cell variables:
(Induction Base:) Assume a variable $c' \in VARS$ with $pre(c') = 0$. Then, it can be immediately seen from the tuple definitions in Sec. 3.2.3 that there is one and only one possible value for c'.
(Induction Assumption:) Assume that the value of all variables c' and C_c in $VARS$ with $pre(c') \leq k$ and $pre(C_c) \leq k$ is unique.
(Induction Step:) Consider any variable c' with $pre(c') = k + 1$ (or C_c with $pre(C_c) = k + 1$). By the definition of the precedence index pre and by the Induction Assumption, the values of all cells referenced in the formula included in the cell c (referenced in the `if`-expression in c) are uniquely defined. Considering the tuple definitions for the various cell types in Sec. 3.2.3 along with the fact that in $CS^*_{VBM}(\Sigma, \Sigma')$ all cells are either *(i)* assumed correct or *(ii)* possibly faulty and non-abnormal, we immediately see that there is one and only one possible value for c' (for C_c).

This concludes the proof. □

C.3. Proof of Theorem 2

Proof. *Statement 1:* We have to verify that the conditions of Definition 14 are fulfilled. To this end, let us define the functions

- $h_{c'} : \{<, =, >\} \to \{\top, \bot\}$ as $h_{c'}(<) = h_{c'}(>) = \bot$, $h_{c'}(=) = \top$ for all variables $c' \in VARS$ for cells $c \in \Sigma$ (cf. Secs. 3.3.2 and 3.4.2), and
- $h_{C_c} : \{\top, \bot\} \to \{\top, \bot\}$ as $h_{C_c}(\top) = \top$, $h_{C_c}(\bot) = \bot$ for all variables $C_c \in VARS$ for cells $c \in \Sigma$ (cf. Secs. 3.3.2 and 3.4.2).

Note that *(1)* these specified functions h_x meet the requirement of Definition 14 that for all $x, x' \in VARS \setminus \{ab_c \mid c \in COMP\}$ and for all $v \in DOM_1(x), v' \in DOM_1(x')$ it holds that $v = v' \to h_x(v) = h_{x'}(v')$, and that *(2)* $h_{ab_c}(\mathsf{true}) = \mathsf{true}$ and $h_{ab_c}(\mathsf{false}) = \mathsf{false}$ for all variables $ab_c \in VARS$ for cells $c \in \Sigma'$ is prespecified by Definition 14.

Now, we have to show that $h(c_1) = c_2$ (as specified in Definition 14) for each pair of constraints (c_1, c_2) where $c_1 \in CONS_1 \cup COBS_1$ and $c_2 \in CONS_2 \cup COBS_2$ with $scope(c_1) = scope(c_2)$. While this is straightforward to verify, it is somewhat laborious. For this reason, we demonstrate the principle for one example pair of constraints given by $c_1 = cons_c \in CONS_1$ from Eq. (3) and $c_2 = cons_c \in CONS_2$ from Eq. (2). The tuple lists $tl(cons_c)$ for the two constraints are explicated in Fig. C.17. We now have to check if the application of $h_{c'}$, h_{C_c} and h_{ab_c}, respectively, as defined above to the variables in all the tuples of c_1 (left table in the figure) yield *exactly* the tuples of c_2 (right table in the figure). Identifying the respective tuples by their row number (first column in the tables), we observe that

- tuples 1, 4, 5, 11, 12 and 15 (left table) map to tuple 2 (right table),
- tuples 2 and 13 (left table) map to tuple 6 (right table),
- tuples 3 and 14 (left table) map to tuple 5 (right table),
- tuples 6 and 9 (left table) map to tuple 3 (right table),
- tuples 7 and 10 (left table) map to tuple 4 (right table), and
- tuple 8 (left table) maps to tuple 1 (right table).

Moreover, using our definition of $h_{c'}$ above, it is easy to see that

- the tuples in lines 16–42 (left table), which cover *all* the $3^3 = 27$ *possible combinations* of three values from $\{<, =, >\}$ for the variables c'_3, c'_4, c', *map exactly to all* the $2^3 = 8$ *possible combinations* of three values from $\{\top, \bot\}$ for the variables c'_3, c'_4, c' in lines 7–14 (right table), and
- the tuples in lines 43–96 (left table), which cover *all* the $2 * 3^3 = 54$ *possible tuples* in $\{\top, \bot\} \times \{<, =, >\}^3$ for the variables c'_3, c'_4, c', *map exactly to all* the $2^4 = 16$ *possible tuples* in $\{\top, \bot\}^4$ for the variables c'_3, c'_4, c' in lines 15–30 (right table).

So, every tuple in the right table occurs as an image of some tuple in the left table, and no other tuples except for those listed in the right table are generated by the mapping. This reveals that $h(tl(cons_c))$, i.e., the set of tuples resulting from $tl(cons_c)$ for $cons_c$ in Eq. (3) by replacing variables as prescribed by the functions $h_{c'}$ and h_{C_c}, indeed corresponds to the set of tuples $tl(cons_c)$ for $cons_c$ in Eq. (2). Hence, $h(c_1) = c_2$ holds for this pair of constraints (c_1, c_2). Analogously, this can be proven for all other respective pairs of constraints (c_1, c_2) where $c_1 \in CONS_1$ and $c_2 \in CONS_2$ with $scope(c_1) = scope(c_2)$. For any pair of constraints (c_1, c_2) where $c_1 \in COBS_1$ and $c_2 \in COBS_2$ with $scope(c_1) = scope(c_2)$, we also have $h(c_1) = c_2$ since observations are consistent by assumption. That is, $c_1 = ((c'), \{v \mid v \in S_1\})$ and $c_2 = ((c'), \{v \mid v \in S_2\})$ are obtained from observations (c, S_1) for P_1 and (c, S_2) for P_2, respectively, and $h_{c'}(>) = h_{c'}(<) = \bot \in S_2$ iff $> \in S_1 \vee < \in S_1$ and $h_{c'}(=) = \top \in S_2$ iff $= \in S_1$ due to Definition 19.

Statement 2: We have to prove that $P_1 \not\prec_{abs} P_2$ for all pairs $(MT_1, MT_2) \in \{(FDM, QDM), (FDM, VBM), (QDM, VBM), (VBM, FDM), (VBM, QDM)\}$.

First, to show $P_1 \prec_{abs} P_2$ for a tuple among (FDM, QDM), (FDM, VBM) and (QDM, VBM), we would need to find a function $h_{c'}$ which maps elements from a smaller domain (i.e., $\{\top, \bot\}$ for *FDM* and $\{<, =, >\}$ for *QDM*) to a larger domain (i.e., $\{<, =, >\}$ for *QDM* and a set of numbers for *VBM*) such that $h(c_1) = c_2$ for any pair of constraints (c_1, c_2) where $c_1 \in CONS_1 \cup COBS_1$ and $c_2 \in CONS_2 \cup COBS_2$ with $scope(c_1) = scope(c_2)$. We consider the constraint for a cell c where the formula $f(c)$ in c is a number n. This constraint is given by

$$c_{FDM} := ((ab_c, c'), \{(\mathsf{false}, \top), (\mathsf{true}, \top), (\mathsf{true}, \bot)\})$$

$$c_{QDM} := ((ab_c, c'), \{(\mathsf{false}, =), (\mathsf{true}, <), (\mathsf{true}, =), (\mathsf{true}, >)\})$$

$$c_{VBM} := ((ab_c, c'), \{(\mathsf{false}, n)\} \cup \{(\mathsf{true}, v) \mid v \in DOM(c')\})$$

for *FDM*, *QDM* and *VBM*, respectively. Since, by Definition 14, the function h_x, for any variable $x \in VARS$, maps values from the domain $DOM_1(x)$ to the domain $DOM_2(x)$, there cannot be a function $h_{c'} : \{\top, \bot\} \to \{<, =, >\}$ that transforms the tuples $\{(\mathsf{true}, \top), (\mathsf{true}, \bot)\}$ to the tuples $\{(\mathsf{true}, <), (\mathsf{true}, =), (\mathsf{true}, >)\}$. Because regardless how $\top$ and $\bot$ are mapped by the *function* $h_{c'}$, these *two* values can never result in *three* different images. Hence, $P_1 \not\prec_{abs} P_2$ for $(MT_1, MT_2) = (FDM, QDM)$. Analogously, we argue that there cannot be a function $h_{c'}$ which maps the (two or three) values in the domain of c' for the *FDM* and *QDM* model types, respectively, to a number domain of higher cardinality for the *VBM* model type in a way the application of the function $h_{c'}$ transforms the tuple lists of the constraints c_{FDM} and c_{QDM}, respectively, to the tuple list of the constraint c_{VBM}. Consequently, in general, $P_1 \not\prec_{abs} P_2$ for $(MT_1, MT_2) \in \{(FDM, VBM), (QDM, VBM)\}$.

Second, to show $P_1 \prec_{abs} P_2$ for a tuple among (VBM, QDM) and (VBM, FDM), we would need to find a function $h_{c'}$ which maps numbers onto either $\{<, =, >\}$ for *QDM* or onto $\{\top, \bot\}$ for *FDM*, and a function h_{C_c} which maps $\{\mathsf{true}, \mathsf{false}\}$ onto $\{\top, \bot\}$, such that $h(c_1) = c_2$ for any pair of constraints (c_1, c_2) where $c_1 \in CONS_1 \cup COBS_1$ and $c_2 \in CONS_2 \cup COBS_2$ with $scope(c_1) = scope(c_2)$. Let us assume that such functions exist. We will now show that this assumption, in general, yields a contradiction. First, assume that $h_{c'}$ maps all numbers to $\bot$ for *FDM* and maps all numbers to either $<$ or $>$ for *QDM*. To see why this is impossible, consider the constraint for a cell c where $f(c)$ is equal to a number n, which is given by c_{VBM}, c_{FDM} as well as c_{QDM} stated above. Since $h_{c'}$ maps no number to $\top$ (*FDM*) and no number to $=$ (*QDM*), the tuple $(\mathsf{true}, \top) \in tl(c_{FDM})$ is not an element of $h(tl(c_{VBM}))$ and the tuple $(\mathsf{true}, =) \in tl(c_{QDM})$ is not an element of $h(tl(c_{VBM}))$, respectively. Consequently, $h_{c'}$ must map at least one number to $\top$ for *FDM* and at least one number to $=$ for *QDM*. Let us refer to this number as k_1. The fact that $h_{c'}(k_1)$ equals $\top$ (*FDM*) and $h_{c'}(k_1)$ equals $=$ (*QDM*) however implies that either $k_1 = 0$ or that there must be a second number k_2 with $k_1 \neq k_2$ and $h_{c'}(k_2)$ equals $\top$ (*FDM*) and $h_{c'}(k_2)$ equals $=$ (*QDM*). This is a consequence of *(1)* the fact that the constraint for cells c including a formula $f(c)$ of the form $c_1 \odot c_2$ for cells c_1, c_2 for both *FDM* and *QDM* (see Secs. 3.3.3 and 3.4.3) does not include any tuple of the form $(\top, \top, \bot)$ and $(=, =, x)$ for $x \in \{<, >\}$, respectively, and *(2)* $k_1 + k_1 = 2k_1$ which is why $2k_1$ must map to $\top$ and $=$, respectively, and *(3)* $k_2 := 2k_1 \neq k_1$ for any $k_1 \neq 0$ or $k_2 = 0$ for $k_1 = 0$. Now, since k_1 and k_2 are mapped to $\top$ or $=$, respectively, we obtain due to the VAL function for a relational operator $\square \in \{<, =, >\}$ defined in Fig. 3 (*FDM*) and in Fig. 5 (*QDM*) that the output value of the comparison must be mapped to $\top$ and $=$, respectively, as well. In other words, the constraint $cons_C$ for *FDM* and the constraint $cons_C$ for *QDM* (cf. Secs. 3.3.3 and 3.4.3) do not include the tuple $(\top, \top, \bot)$ and $(=, =, x)$ for $x \in \{<, >\}$. Now, consider the two comparisons $k_1 < k_2$ and $k_1 = k_2$ and observe that one evaluates to true whereas the other evaluates to false, regardless of whether $k_1 = 0$ or $k_1 \neq 0$. This however means that the function $h_{C_c} : \{\mathsf{true}, \mathsf{false}\} \to \{\top, \bot\}$ must be defined as $h_{C_c}(\mathsf{true}) = h_{C_c}(\mathsf{false}) = \top$. As a consequence, the result $h(tl(cons_C))$ of applying h to the tuple list $tl(cons_C)$ of the constraint $cons_C$ for *VBM* (cf. Sec. 3.2.3) yields only tuples of the form $(\cdot, \cdot, \top)$. This in turn entails that $h(tl(cons_C))$ for the constraint $cons_C$ for *VBM* necessarily differs from $tl(cons_C)$ for both the constraint $cons_C$ for *FDM* and the constraint $cons_C$ for *QDM* (cf. Secs. 3.3.3 and 3.4.3). The reason is that the latter two include, e.g., the tuple $(\bot, \bot, \bot)$ and $(>, >, \bot)$, respectively. This is a contradiction to the assumption of the existence of functions $h_{c'}$ and h_{C_c} satisfying the conditions of Definition 14. Hence, in general $P_1 \not\prec_{abs} P_2$ for $(MT_1, MT_2) \in \{(VBM, FDM), (VBM, QDM)\}$. □

$tl(cons_c)$ for $cons_c$ from Eq. 3

	ab_c	C_c	c'_3	c'_4	c'
1	**false**	$\top$	$<$	$<$	$<$
2	**false**	$\top$	$<$	$=$	$<$
3	**false**	$\top$	$<$	$=$	$=$
4	**false**	$\top$	$<$	$>$	$<$
5	**false**	$\top$	$<$	$>$	$>$
6	**false**	$\top$	$=$	$<$	$=$
7	**false**	$\top$	$=$	$<$	$<$
8	**false**	$\top$	$=$	$=$	$=$
9	**false**	$\top$	$=$	$>$	$=$
10	**false**	$\top$	$=$	$>$	$>$
11	**false**	$\top$	$>$	$<$	$<$
12	**false**	$\top$	$>$	$<$	$>$
13	**false**	$\top$	$>$	$=$	$>$
14	**false**	$\top$	$>$	$=$	$=$
15	**false**	$\top$	$>$	$>$	$>$
$(16-42)$	**false**	$\bot$	(all comb.)		
$(43-96)$	**true**	(all comb.)			

$tl(cons_c)$ for $cons_c$ from Eq. 2

	ab_c	C_c	c'_3	c'_4	c'
1	**false**	$\top$	$\top$	$\top$	$\top$
2	**false**	$\top$	$\bot$	$\bot$	$\bot$
3	**false**	$\top$	$\top$	$\bot$	$\top$
4	**false**	$\top$	$\top$	$\bot$	$\bot$
5	**false**	$\top$	$\bot$	$\top$	$\top$
6	**false**	$\top$	$\bot$	$\top$	$\bot$
$(7-14)$	**false**	$\bot$	(all comb.)		
$(15-30)$	**true**	(all comb.)			

Fig. C.17. Tuple lists for two constraints, used in the proof of Theorem 2.

C.4. Proof of Corollary 1

Proof. The statement of the corollary is a direct consequence of Theorem 1 and of the first statement of Theorem 2. □

C.5. Proof of Theorem 3

Proof. Let $\Delta \subseteq COMP$, and let $CONS_\Delta$ denote the set of constraints given by $\{((ab_c), \{(\mathsf{true})\}) \mid c \in \Delta\} \cup \{((ab_c), \{(\mathsf{false})\}) \mid c \in COMP \setminus \Delta\}$. Now, assume that $P_1 \preceq_{inf} P_2$ holds. This implies by Definition 20 that CS_1^Δ is satisfiable whenever CS_2^Δ is. By Definition 13, we can directly conclude from this that Δ is a diagnosis for P_1 whenever it is a diagnosis for P_2. As Δ is an arbitrary subset of $COMP$, we have that $\mathbf{AllD}_{P_2} \subseteq \mathbf{AllD}_{P_1}$, which means that $P_1 \preceq_{acc} P_2$. □

C.6. Proof of Theorem 4

Proof. Let $\Delta \in COMP$. Assume there is a solution (valid value assignment, cf. Definition 7) M_{VBM} for CS_2.

First, assume that *(I)* $MT_1 = FDM$. Let a variable assignment M_{FDM} for CS_1 be defined based on M_{VBM} as follows: Given a cell $c \in \Sigma$, for the associated variable c' (and the associated variable C_c if c includes an if-statement, cf. Sec. 3), we specify $(c', \top) \in M_{FDM}$ if $(c', c^*) \in M_{VBM}$ and $(c', \bot) \in M_{FDM}$ else (and $(C_c, \top) \in M_{FDM}$ if $(C_c, C_c^*) \in M_{VBM}$ and $(C_c, \bot) \in M_{FDM}$ else). Note that all ab_c variables are set equally in both M_{FDM} and M_{VBM} since the same constraints $CONS_\Delta$, which uniquely specify all ab_c variables, are used for both P_1 and P_2.

We now show that M_{FDM} is a valid value assignment for CS_1. To this end, assume that M_{FDM} is not a valid value assignment for CS_1. This means that there must be some cell $c \in \Sigma$ for which an associated constraint, denoted by x_c, is not satisfied (recall, there is exactly one constraint for all discussed cell types, except for cells including an if-statement, which have two associated constraints $cons_C$ and $cons_c$, cf. Sec. 3); note that all observation constraints $COBS_1$ are satisfied for FDM due to Definition 19, the assumption that the observations for all models are consistent, and the specification of M_{FDM}. Thus, assume that some constraint x_c for some cell $c \in \Sigma$ is violated. First, x_c cannot be any constraint including a variable c' where $c \in \Delta$, since for all these constraints the variable ab_c is set to true, which means these constraints are satisfied for arbitrary value assignments (weak fault model, cf. the constraint specifications in Sec. 3). Hence, we only need to consider those constraints including a variable c' for which ab_c is set to false or for which the cell is not possibly faulty, i.e., $c \in \Sigma \setminus \Sigma'$ (note, for any cell c, the constraint including the variable c' has the same effect, whether ab_c is set to false or $c \in \Sigma \setminus \Sigma'$, cf. Sec. 3), and those constraints including a variable C_c. Let us now consider the constraint x_c for the different types of cells c based on their included formula $f(c)$:

- $f(c)$ is a number n: Since x_c is satisfied for M_{VBM}, it must hold that $(c', n) \in M_{VBM}$ (cf. Sec. 3.2.3). However, also for $CS^*_{VBM}(\Sigma, \Sigma')$, it must hold that c' is set to n. Hence, $c^* = n$ and $(c', c^*) \in M_{VBM}$, which is why $(c', \top) \in M_{FDM}$. This in turn means that x_c is satisfied for M_{FDM} (cf. Sec. 3.3.3). This is a contradiction to the assumption that x_c is violated for M_{FDM}.

- $f(c)$ is a reference to a cell $\tilde{c}$: Since x_c is satisfied for M_{VBM}, it must hold that c' and $\tilde{c}'$ are set equally in M_{VBM} (cf. Sec. 3.2.3). Hence, either $(c',\top),(\tilde{c}',\top) \in M_{FDM}$ or $(c',\bot),(\tilde{c}',\bot) \in M_{FDM}$. In both cases, x_c is satisfied for M_{FDM} (cf. Sec. 3.3.3). This is a contradiction to the assumption that x_c is violated for M_{FDM}.
- $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+,-,*,/\}$: We have four possible cases, i.e., in M_{VBM}, c'_1 can be set to c^*_1 or another value $x \neq c^*_1$, and c'_2 can be set to c^*_2 or another value $x \neq c^*_2$. If c' is not equal to c^* in M_{VBM} for some cell c, this means that c has a value that differs from the value computed for c in the original spreadsheet (without taking any observations into account). So, for instance, assume $\odot = +$ as well as $c'_1 = c^*_1$ and $c'_2 \neq c^*_2$ in M_{VBM} (which means $(c'_1,\top),(c'_2,\bot) \in M_{FDM}$). Then, $c' \neq c^*$ must hold in M_{VBM} due to the facts that *(i)* constraint x_c is satisfied for M_{VBM}, *(ii)* x_c enforces the value of c' to be equal to $c'_1 \odot c'_2$, and *(iii)* the sum of a "correct" value ($\top$) with an "incorrect" value ($\bot$) must yield an "incorrect" result ($\bot$). Now, observe that, in general, for any setting of $\odot$, (i) holds by assumption, (ii) holds due to the specification of the constraints in Sec. 3.2.3, and (iii) is exactly the argumentation we used in Sec. 3.3.3 to specify the constraint for *FDM* where $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+,-,*,/\}$. Consequently, the setting $(c',\top) \in M_{FDM}$ satisfies x_c whenever $(c',c^*) \in M_{VBM}$, and the setting $(c',\bot) \in M_{FDM}$ satisfies x_c whenever $(c',y) \in M_{VBM}$ for some $y \neq c^*$. As a result, we have that x_c cannot be violated for M_{FDM}, which is a contradiction to our assumption.
- $f(c)$ is of the form $\texttt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \ldots, c_4$ and $\square \in \{<,=,>\}$: For this cell type we have two constraints, $cons_C$ and $cons_c$ (cf. Sec. 3), and $c'_i = c^*_i$ or $c'_i \neq c^*_i$ can hold for M_{VBM} for $i \in \{1,\ldots,4\}$. Let us first assume that *(A)* x_c corresponds to the constraint $cons_C$ including the variables C_c as well as c'_1 and c'_2. First, assume *(i)* $c'_1 = c^*_1$ and $c'_2 = c^*_2$ in M_{VBM}, which is why $(c'_1,\top),(c'_2,\top) \in M_{FDM}$. That is, two "correct" values ($\top$) are compared with one another, which implies that the result will be "correct" ($\top$) as well, i.e., $C_c = C^*_c$ by the specification of the constraint $cons_C$ for *VBM* (cf. Sec. 3.2.3). Thus, $(C_c,\top) \in M_{FDM}$ holds, which satisfies the constraint $cons_C$ for *FDM* (Sec. 3.3.3). Now, assume *(ii)* that $c'_1 \neq c^*_1$ or $c'_2 \neq c^*_2$ or both are in M_{VBM}, which is why $(c'_1,\bot) \in M_{FDM}$ or $(c'_2,\bot) \in M_{FDM}$ or both. In this case, regardless of the setting of $C_c \in \{\top,\bot\}$ in M_{VBM}, the constraint $cons_C$ will be satisfied for M_{FDM} (cf. Sec. 3.3.3). Now, let *(B)* x_c correspond to the constraint $cons_c$ including the variables c', C_c, c'_3, and c'_4. First, assume *(i)* $c'_3 = c^*_3$, $c'_4 = c^*_4$, and $C_c = C^*_c$ in M_{VBM}, which is why $(c'_3,\top),(c'_4,\top),(C_c,\top) \in M_{FDM}$. That is, the comparison outcome is correct ($\top$), and for both results of the comparison in the if-statement, the respective chosen value is "correct" ($\top$), which implies that the value c' of the cell will be "correct" ($\top$) as well, i.e., $c' = c^*$ by the specification of the constraint $cons_c$ for *VBM* (cf. Sec. 3.2.3). Thus, $(c',\top) \in M_{FDM}$ holds, which satisfies the constraint $cons_c$ for *FDM* (cf. Sec. 3.3.3). Now, assume *(ii)* at least one of $c'_3 \neq c^*_3$ or $c'_4 \neq c^*_4$ or $C_c \neq C^*_c$ is in M_{VBM}, which is why one of $(c'_3,\bot) \in M_{FDM}$ or $(c'_4,\bot) \in M_{FDM}$ or $(C_c,\bot) \in M_{FDM}$ must hold. In this case, regardless of the setting of $c' \in \{\top,\bot\}$ in M_{VBM}, the constraint $cons_c$ will be satisfied for M_{FDM} (cf. Sec. 3.3.3). Overall, we obtain a contradiction to our assumption that some constraint x_c for c is violated.

Hence, we have shown that M_{FDM} is indeed a valid value assignment for CS_1. This proves that CS_1 is satisfiable. By Definition 20, this concludes the proof for the case (I) where $MT_1 = FDM$.

Second, assume that *(II)* $MT_1 = QDM$. The proof for this case is analogous to the one for case (I), just that we now specify a variable assignment M_{QDM} for CS_1 based on M_{VBM} as follows: Given a cell $c \in \Sigma$, for the associated variable c' (and the associated variable C_c if c includes an if-statement, cf. Sec. 3), we specify $(c',=) \in M_{QDM}$ if $(c',c^*) \in M_{VBM}$, $(c',<) \in M_{QDM}$ if $(c',y) \in M_{VBM}$ for some $y < c^*$, and $(c',>) \in M_{QDM}$ if $(c',y) \in M_{VBM}$ for some $y > c^*$ (and $(C_c,\top) \in M_{QDM}$ if $(C_c,C^*_c) \in M_{VBM}$ and $(C_c,\bot) \in M_{QDM}$ else).

The argumentation to show that M_{QDM} is a valid value assignment for CS_1 again involves a differentiation between different cell types based on the cell formula $f(c)$ in the cell c:

- $f(c)$ is a number n: Since x_c is satisfied for M_{VBM}, it must hold that $(c',n) \in M_{VBM}$ (cf. Sec. 3.2.3). However, also for $CS^*_{VBM}(\Sigma,\Sigma')$, it must hold that c' is set to n. Hence, $c^* = n$ and $(c',c^*) \in M_{VBM}$, which is why $(c',=) \in M_{QDM}$. This in turn means that x_c is satisfied for M_{QDM} (cf. Sec. 3.4.3). This is a contradiction to the assumption that x_c is violated for M_{QDM}.
- $f(c)$ is a reference to a cell $\tilde{c}$: Since x_c is satisfied for M_{VBM}, it must hold that c' and $\tilde{c}'$ are set equally in M_{VBM} (cf. Sec. 3.2.3). Hence, either $(c',=),(\tilde{c}',=) \in M_{QDM}$ or $(c',<),(\tilde{c}',<) \in M_{QDM}$ or $(c',>),(\tilde{c}',>) \in M_{QDM}$. In all cases, x_c is satisfied for M_{QDM} (cf. Sec. 3.4.3). This is a contradiction to the assumption that x_c is violated for M_{QDM}.
- $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+,-,*,/\}$: We have four possible cases, i.e., in M_{VBM}, c'_1 can be set to c^*_1 or another value $x \neq c^*_1$, and c'_2 can be set to c^*_2 or another value $y \neq c^*_2$. If c' is not equal to c^* in M_{VBM} for some cell c, this means that c has a value that differs from the value computed for c in the original spreadsheet (without taking any observations into account). So, for instance, assume $\odot = +$ as well as $c'_1 = c^*_1$ and $c'_2 < c^*_2$ in M_{VBM} (which means $(c'_1,=),(c'_2,<) \in M_{QDM}$). Then, $c' < c^*$ must hold in M_{VBM} due to the facts that *(i)* constraint x_c is satisfied for M_{VBM}, *(ii)* x_c enforces the value of c' to be equal to $c'_1 \odot c'_2$, and *(iii)* the sum of a "correct" value (=) with a "too low" value (<) must yield a "too low" result (<). Now, observe that, in general, for any setting of $\odot$, (i) holds by assumption, (ii) holds due to the specification of the constraints in Sec. 3.2.3, and (iii) is exactly the argumentation we used in Sec. 3.4.3 to specify the constraint for *QDM* where $f(c)$ is of the form $c_1 \odot c_2$ for cells c_1, c_2 and $\odot \in \{+,-,*,/\}$. Consequently, the setting $(c',=) \in M_{QDM}$ satisfies x_c whenever $(c',c^*) \in M_{VBM}$, the setting $(c',<) \in M_{QDM}$ satisfies x_c whenever $(c',y) \in M_{VBM}$ for some $y < c^*$, and the setting $(c',>) \in M_{QDM}$ satisfies x_c whenever $(c',y) \in M_{VBM}$ for some $y > c^*$. As a result, we have that x_c cannot be violated for M_{QDM}, which is a contradiction to our assumption.

- $f(c)$ is of the form $\texttt{if}(c_1 \square c_2, c_3, c_4)$ for cells $c_1, \dots, c_4$ and $\square \in \{<, =, >\}$: For this cell type we have two constraints, $cons_C$ and $cons_c$ (cf. Sec. 3), and $c_i' = c_i^*$ or $c_i' \neq c_i^*$ can hold for M_{VBM} for $i \in \{1, \dots, 4\}$.
  Let us first assume that *(A)* x_c corresponds to the constraint $cons_C$ including the variables C_c as well as c_1' and c_2'. First, assume *(i)* $c_1' = c_1^*$ and $c_2' = c_2^*$ in M_{VBM}, which is why $(c_1', =), (c_2', =) \in M_{QDM}$. That is, two "correct" values ($=$) are compared with one another, which implies that the result will be "correct" ($\top$) as well, i.e., $C_c = C_c^*$ by the specification of the constraint $cons_C$ for *VBM* (cf. Sec. 3.2.3). Thus, $(C_c, \top) \in M_{QDM}$ holds, which satisfies the constraint $cons_C$ for *QDM* (cf. Sec. 3.4.3). Now, assume *(ii)* $c_1' = x$ for some $x \neq c_1^*$ or $c_2' = y$ for some $y \neq c_2^*$ or both are in M_{VBM}, which is why at least one of $(c_1', <)$, $(c_1', >)$, $(c_2', <)$, or $(c_2', >)$ must be in M_{QDM}. In this case, however, the constraint $cons_C$ is satisfied for any setting of C_c in M_{QDM} (cf. Sec. 3.4.3).
  Now, let *(B)* x_c correspond to the constraint $cons_c$ including the variables c', C_c, c_3', and c_4'. First, assume *(i)* $C_c = C_c^*$ in M_{VBM}, which is why $(C_c, \top) \in M_{QDM}$. That is, the comparison outcome is correct ($\top$). Hence, the designated cell value among c_3', c_4' will be chosen as the value of c', which is why, by the specification of the constraint $cons_c$ for *VBM* (cf. Sec. 3.2.3), c' must be equal to c_3' (given the comparison outcome is `true`) or equal to c_4' (given the comparison value is `false`). Thus, $(c', v) \in M_{QDM}$ where $v = v_3 \vee v = v_4$, $v_3, v_4 \in \{<, =, >\}$, and $(c_3', v_3) \in M_{QDM}$ and $(c_4', v_4) \in M_{QDM}$. This however implies that the constraint $cons_c$ (cf. Sec. 3.4.3) is satisfied for M_{QDM}. Now, assume *(ii)* $C_c \neq C_c^*$ in M_{VBM}, which is why $(C_c, \bot) \in M_{QDM}$. In this case, the constraint $cons_c$ in *QDM* is satisfied for any setting of the variables c_3', c_4', c' (cf. Sec. 3.4.3) and thus cannot be violated. Overall, we obtain a contradiction to our assumption that some constraint x_c for c is violated.

As a result, we have demonstrated that M_{QDM} is indeed a valid value assignment for CS_1. This proves that CS_1 is satisfiable. By Definition 20, this concludes the proof for the case (II) where $MT_1 = QDM$. □

C.7. Proof of Corollary 2

Proof. The statement of the corollary is a direct consequence of Theorems 3 and 4. □

C.8. Proof of Corollary 3

Proof. The statement of the corollary is a direct consequence of Corollaries 1 and 2. □

C.9. Proof of Theorem 5

Proof. *Statement 1:* By Definition 21, $\mathbf{AllD}_{P_{VBM}}$ comprises all correct diagnoses. Now, the statement follows directly from Corollary 3 and Definition 16.

Statement 2: The statement follows from the fact that $FPsp(\Sigma) \subseteq \mathbf{AllD}_{P_{VBM}}$ (cf. Definition 21) along with Corollary 3 and Definition 16.

Statement 3: From Statement 1, Corollary 3, and Definition 21 we directly obtain that *(a)* $\mathbf{AllD}_{P_{FDM}} \cap Corr(\Sigma) = \mathbf{AllD}_{P_{QDM}} \cap Corr(\Sigma) = \mathbf{AllD}_{P_{VBM}} \cap Corr(\Sigma)$. Moreover, Definition 21 implies that *(b)* every set of components $\Delta \subseteq \Sigma'$ belongs to exactly one of the sets $Corr(\Sigma)$, $FPsp(\Sigma)$, or $FNsp(\Sigma)$. Statement 2 is equivalent to $\mathbf{AllD}_{P_{FDM}} \cap FPsp(\Sigma) = \mathbf{AllD}_{P_{QDM}} \cap FPsp(\Sigma) = \mathbf{AllD}_{P_{VBM}} \cap FPsp(\Sigma)$. By Corollary 3 and Definition 16 we have that *(d)* $\mathbf{AllD}_{P_{FDM}} \supseteq \mathbf{AllD}_{P_{QDM}} \supseteq \mathbf{AllD}_{P_{VBM}}$. Now, putting (a)–(d) together, the statement follows. □

C.10. Proof of Theorem 6

Proof. The first statement follows directly from Corollary 3 as well as Definitions 21 and 22. Regarding the second statement, we give in Sec. 6 counterexamples showing that $\mathbf{AllD}_{P'} \subseteq \mathbf{AllD}_P$ does not generally hold. In particular, we will investigate spreadsheets Σ such that for a spreadsheet-constraint diagnosis problem P modeling Σ by means of the *VBM* and for a spreadsheet-constraint diagnosis problem P' modeling Σ by means of any model type among $\{QDM, FDM\}$, we will observe that $\mathbf{AllD}_{P'} \supset \mathbf{AllD}_P$. □

Appendix D. Bottomline of the experiment results

We summarize our main findings as follows:

Computation time

- On average, all model types allow for an efficient computation of all minimal diagnoses of any cardinality in $\{1, 2, 3\}$ (median / average computation times of less than 6 seconds / 1 minute).
- There are statistically significant and sizeable performance differences between the abstract model types (*FDM*, *QDM*) and the exact one (*VBM*).

- The *VBM* requires between three times as much and more than four orders of magnitude more computation time for the same tasks than both the *FDM* and the *QDM*.
- The *VBM* could not successfully finish the computations (before a timeout was triggered) for almost one out of five cases overall, and for up to more than half of the spreadsheets per computation task (triple-fault diagnoses, artificial dataset).
- For diagnosis cardinalities 2 and 3, there is a significant number of cases where the *VBM* manifests impractical computation times; for cardinality 1, times for *VBM* are mostly reasonable (less than 5 minutes), but 7 timeouts were still registered.
- The abstract model types are always highly performant, and never require more than 0.5 seconds (artificial dataset) / 3.1 seconds (real-world dataset) for the computation of all minimal diagnoses of a particular cardinality, even for the hardest cases where the *VBM* leads to a timeout.

Number of generated diagnoses

- Considering only non-timeout cases, the (median) number of generated single-fault diagnoses is reasonable (no more than a dozen) for all model types and both experiment datasets.
- Considering only non-timeout cases, the median number of generated double-fault diagnoses for $\langle FDM, QDM, VBM\rangle$ is $\langle 91, 78, 51\rangle$ / $\langle 290, 285, 24\rangle$ for the real-world / artificial dataset. There is
  - a reasonable number (less than a dozen) of double-fault diagnoses for the *VBM* (ground truth) in 9 % / 32 % of the cases for the real-world / artificial dataset. Among those cases, $\langle FDM, QDM\rangle$ also generate a reasonable number of double-fault diagnoses in $\langle 17, 17\rangle$ / $\langle 20, 25\rangle$ percent of the spreadsheets for the real-world / artificial dataset.
  - a still acceptable number (less than 50) of double-fault diagnoses for the *VBM* (ground truth) in 49 % / 60 % of the cases for the real-world / artificial dataset. Among those cases, $\langle FDM, QDM\rangle$ also generate an acceptable number of double-fault diagnoses in $\langle 54, 60\rangle$ / $\langle 69, 69\rangle$ percent of the spreadsheets for the real-world / artificial dataset.
- Considering only non-timeout cases, the median number of generated triple-fault diagnoses for $\langle FDM, QDM, VBM\rangle$ is roundly $\langle 700, 660, 520\rangle$ / $\langle 80, 80, 64\rangle$ for the real-world / artificial dataset. There is
  - a reasonable number (less than a dozen) of triple-fault diagnoses for the *VBM* (ground truth) in 2 % / 2 % of the cases for the real-world / artificial dataset. Among those cases, $\langle FDM, QDM\rangle$ also generate a reasonable number of triple-fault diagnoses in $\langle 0, 0\rangle$ / $\langle 0, 0\rangle$ percent of the spreadsheets for the real-world / artificial dataset.
  - a still acceptable number (less than 50) of triple-fault diagnoses for the *VBM* (ground truth) in 14 % / 36 % of the cases for the real-world / artificial dataset. Among those cases, $\langle FDM, QDM\rangle$ also generate an acceptable number of triple-fault diagnoses in $\langle 61, 61\rangle$ / $\langle 52, 52\rangle$ percent of the spreadsheets for the real-world / artificial dataset.
- For all scenarios where the generated number of diagnoses is large, it is recommendable to combine the mere diagnosis computation with complementary approaches that allow to prune the diagnosis space by gathering additional information (e.g., sequential diagnosis techniques), or that impose an expedient ranking on the returned diagnoses (e.g., SFL techniques).
- *VBM* generates statistically significantly fewer diagnoses than both abstract model types for all investigated diagnosis sizes and datasets, but the difference in the (median) number of diagnoses between *VBM* and any of the abstract model types barely exceeds one order of magnitude.
- The reductions in the (median) number of generated diagnoses when using the exact *VBM* model instead of the more accurate *QDM* among the abstract model types range from 20 % (artificial dataset, diagnosis cardinality 3) to 92 % (artificial dataset, diagnosis cardinality 2).
- The reductions in the (median) number of generated diagnoses when using the *QDM* instead of the *FDM* range from 0 % (artificial dataset, diagnosis cardinality 3) to 25 % (real-world dataset, diagnosis cardinality 1).
- The fraction of spreadsheets for which the abstract model types lead to the same diagnostic solutions as the exact model ranges from 1 % (artificial dataset, diagnosis size 2) to fairly impressive 52 % (real-world dataset, diagnosis size 3) for the *QDM*, and from 0 % (artificial dataset, diagnosis size 2) to remarkable 48 % (real-world dataset, diagnosis size 3) for the *FDM*.

Precision

- While the exact model type *VBM* always generates all diagnoses (of a particular cardinality) in a sound and complete way, both abstract model types are complete (do not miss diagnoses) as well, but unsound in general (can produce solutions that are not diagnoses).
- The precision (ratio of diagnoses in the generated solution set) of the *VBM* is always 100 %; for all diagnosis sizes and both datasets, it is statistically significantly and on average substantially larger than for both abstract model types.
- Also, the differences in precision between the two abstract model types are statistically significant and notable for all diagnosis sizes and both datasets, except for diagnosis cardinality 3 considering the artificial dataset.
- For both *QDM* and *FDM*, the (median) precision values are very good (over 90 %) for diagnosis sizes 2 and 3 for the real-world dataset, and reasonable (at least 75 %) and for diagnosis cardinality 3 for the artificial dataset. The worst recorded value for *QDM* / *FDM* is 15 % / 9 % (diagnosis size 1, artificial dataset).

Ratio of ruled out diagnosis candidates

- Model-based reasoning can be used in combination with other diagnostic approaches (such as SFL), where its main aim is the detection of as many *proven* non-diagnoses as possible in order to reduce the search space of potential diagnosis candidates for these other approaches.

- For all model types and both datasets, the ratio of eliminated potential diagnosis candidates is highest for single-fault diagnoses, and decreases for each increment of the diagnosis size.
- Apart from (QDM, FDM) for size 3 for the artificial dataset, all model types are statistically significantly different w.r.t. their ability to rule out potential diagnosis candidates, where the *VBM* is superior to the *QDM* which in turn is superior to the *FDM*.
- The (median) ratio of potential diagnosis candidates provably ruled out is always substantial for the exact *VBM* model type, with values ranging from 45 % (diagnosis size 3, artificial dataset) to 98 % (diagnosis size 1, artificial dataset), which demonstrates the power of model-based reasoning as a means for search space pruning.
- Although considerably lower than for the exact model type, the (median) ratio of potential diagnosis candidates provably ruled out by the *QDM* is reasonable (larger than 20 %) in most investigated scenarios, with values ranging from 23 % (diagnosis size 2, real-world dataset) to 70 % (diagnosis size 1, artificial dataset). The lowest value we measured for *QDM* was 11 % (size 3, real-world dataset).
- Even the simplest model type *FDM* can be a fair tool to prune the diagnostic search space, with its achieved (median) ratio of potential diagnosis candidates provably ruled out ranging from 24 % to respectable 70 % for the artificial dataset. It was not so powerful for the real-world dataset, however, where we observed values between mere 2 % and 31 %.

Time-accuracy trade-off

- Since the superiority rankings among the model types are inverse when comparing their efficiency and their accuracy, where *FDM* dominates *QDM* dominates *VBM* w.r.t. a lower computation time and *VBM* dominates *QDM* dominates *FDM* w.r.t. a lower number of generated diagnoses, we assessed the model types based on their time-accuracy trade-off, i.e., the ratio between the factor of less time required than the *VBM* and the factor of more diagnoses generated than the *VBM*.
- While the time-accuracy trade-off is by definition always 1 for the *VBM*, we find that the (median) trade-off is statistically significantly and substantially larger than 1 for both abstract model types for both datasets and all diagnosis sizes, meaning that both abstract types achieve a favorable trade-off, i.e., they save more time than they generate more diagnoses compared with the *VBM*.
- Both abstract model types roughly provide an equally good trade-off, i.e., their differences are mostly minor or negligible.
- The median trade-off values achieved by the abstract model types for the different diagnosis cardinalities range from 9 to 266 over both datasets, the averages from 21 to 3372, i.e., between one and up to more than three orders of magnitude more time is saved on average than there are more diagnoses generated when using one of the abstract model types instead of the exact *VBM*.

Appendix E. Experiment results for individual spreadsheets

Complementing our analyses in Sec. 6, Figs. E.18–E.29 expound the detailed results for all the individual spreadsheets under test.

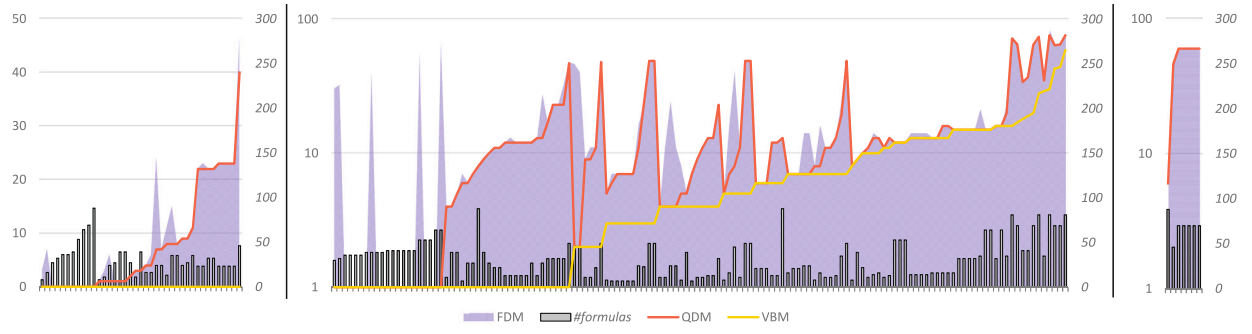


Fig. E.18. Real-World Spreadsheets: Results for the single spreadsheets in terms of the **number of returned diagnoses of cardinality 1** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where there are no diagnoses of cardinality 1 for the *VBM*, whereas the *middle plot* describes all cases where at least one diagnosis of cardinality 1 exists for *VBM*. While the left and middle plots relate to the cases where the computations for all model types could be successfully finished, the *right plot* depicts all cases where computations for *VBM* triggered a timeout. The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *QDM* (left and right plots) and *VBM* (middle plot). Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis in each plot.

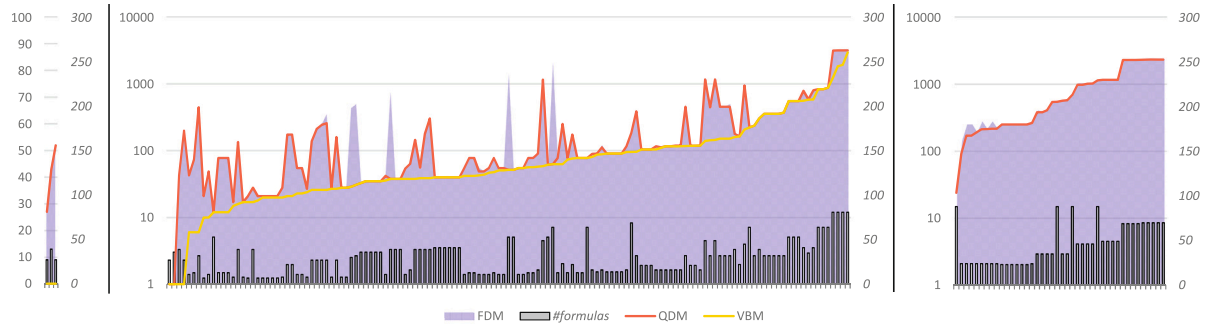


Fig. E.19. Real-World Spreadsheets: Results for the single spreadsheets in terms of the **number of returned diagnoses of cardinality 2** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where there are no diagnoses of cardinality 2 for the *VBM*, whereas the *middle plot* describes all cases where at least one diagnosis of cardinality 2 exists for *VBM*. While the left and middle plots relate to the cases where the computations for all model types could be successfully finished, the *right plot* depicts all cases where computations for *VBM* triggered a timeout. The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *QDM* (left and right plots) and *VBM* (middle plot). Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis in each plot.

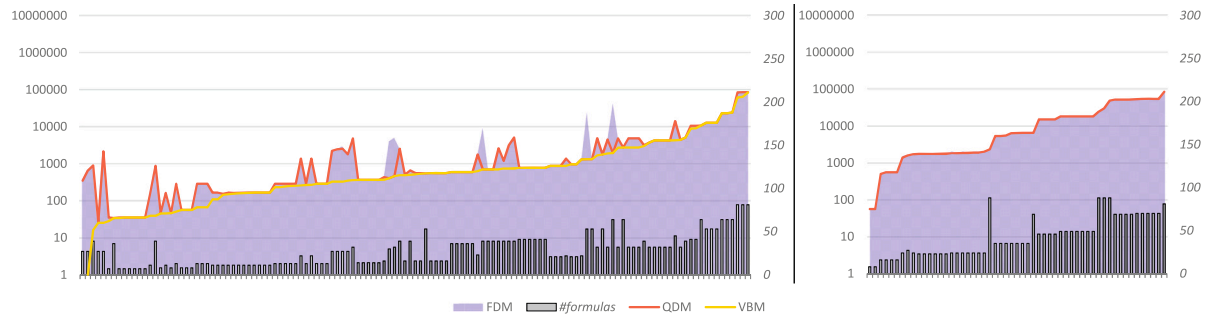


Fig. E.20. Real-World Spreadsheets: Results for the single spreadsheets in terms of the **number of returned diagnoses of cardinality 3** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where the computations for all model types finished successfully (*no timeout triggered*), whereas the *right plot* describes all cases where computations for *VBM* could not be finished (*timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Note, for the leftmost case in the left plot, the value of *VBM* equals zero and is thus not shown (logarithmic scale). Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis in each plot.

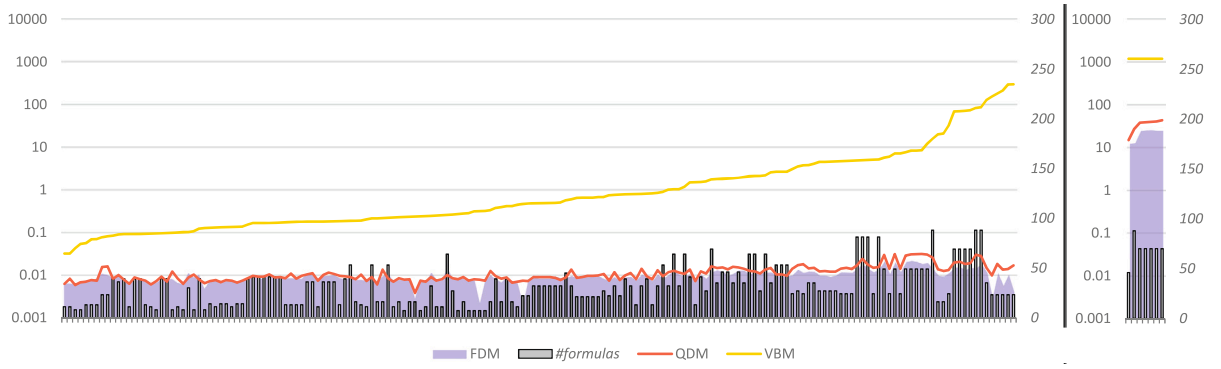


Fig. E.21. Real-World Spreadsheets: Results for the single spreadsheets in terms of the **time (in seconds) required for the computation of all minimal diagnoses of cardinality 1** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where computations finished successfully for all model types (*no timeout was triggered*), whereas the *right plot* depicts the cases where computations for *VBM* could not be finished (*timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Note, the displayed times (corresponding to the timeout of 1200 seconds) for *VBM* in the right plot are lower bounds of the actual times. Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis.

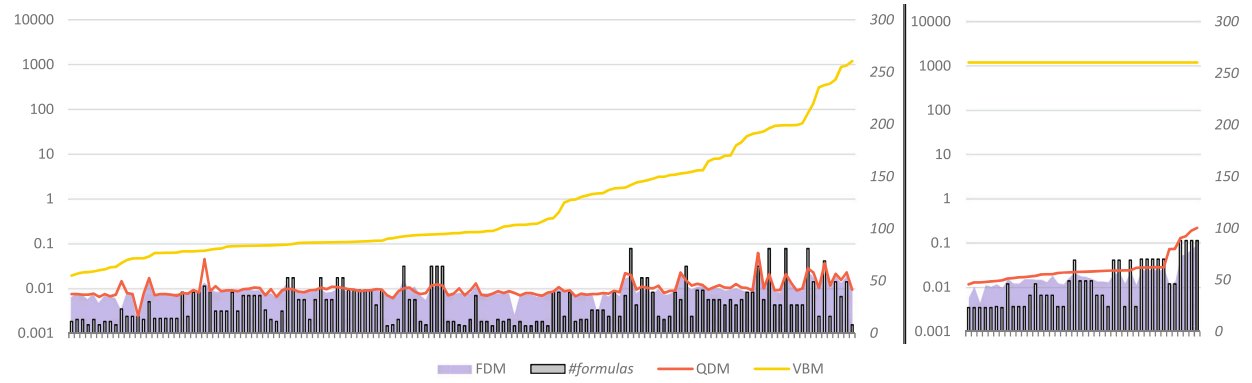


Fig. E.22. Real-World Spreadsheets: Results for the single spreadsheets in terms of the **time (in seconds) required for the computation of all minimal diagnoses of cardinality 2** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where computations finished successfully for all model types (*no timeout was triggered*), whereas the *right plot* depicts the cases where computations for *VBM* could not be finished (*timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Note, the displayed times (corresponding to the timeout of 1200 seconds) for *VBM* in the right plot are lower bounds of the actual times. Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis.

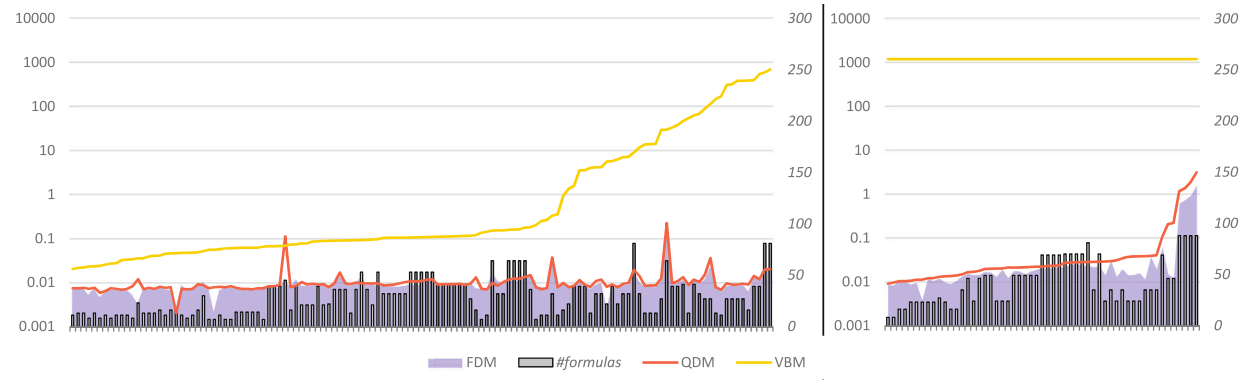


Fig. E.23. Real-World Spreadsheets: Results for the single spreadsheets in terms of the **time (in seconds) required for the computation of all minimal diagnoses of cardinality 3** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where computations could be finished for all model types (*no timeout was triggered*), whereas the *right plot* depicts the cases where computations for *VBM* could not be finished (*timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Note, the displayed times (corresponding to the timeout of 1200 seconds) for *VBM* in the right plot are lower bounds of the actual times. Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis.

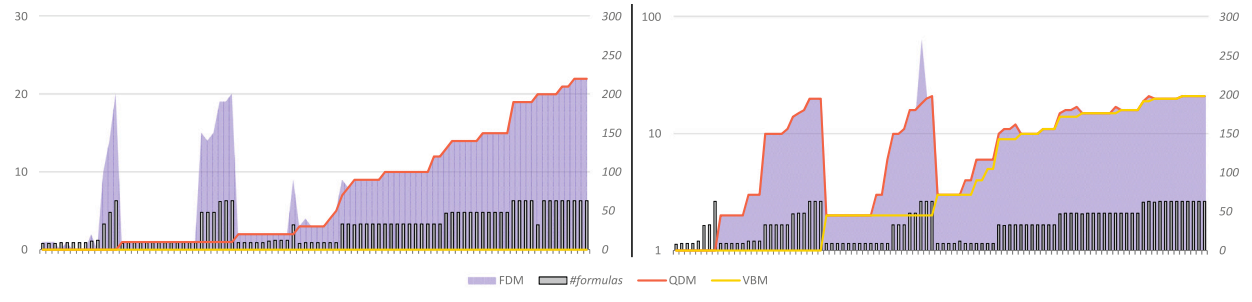


Fig. E.24. Artificial Spreadsheets: Results for the single spreadsheets in terms of the **number of returned diagnoses of cardinality 1** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where there were no diagnoses of cardinality 1 for the *VBM* (normal scale of the left y-axis), whereas the *right plot* describes all other cases (logarithmic scale of the left y-axis). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *QDM* (left plot) and *VBM* (right plot). Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis in each plot.

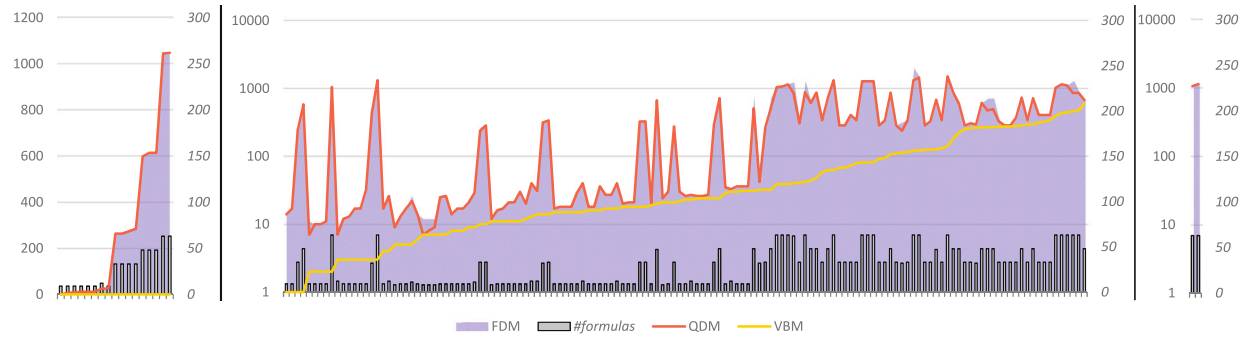


Fig. E.25. Artificial Spreadsheets: Results for the single spreadsheets in terms of the **number of returned diagnoses of cardinality 2** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where there are no diagnoses of cardinality 2 for the *VBM*, whereas the *middle plot* describes all cases where at least one diagnosis of cardinality 2 exists for *VBM*. While the left and middle plots relate to the cases where the computations for all model types could be successfully finished, the *right plot* depicts all cases where computations for *VBM* triggered a timeout. The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *QDM* (left and right plots) and *VBM* (middle plot). Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis in each plot.

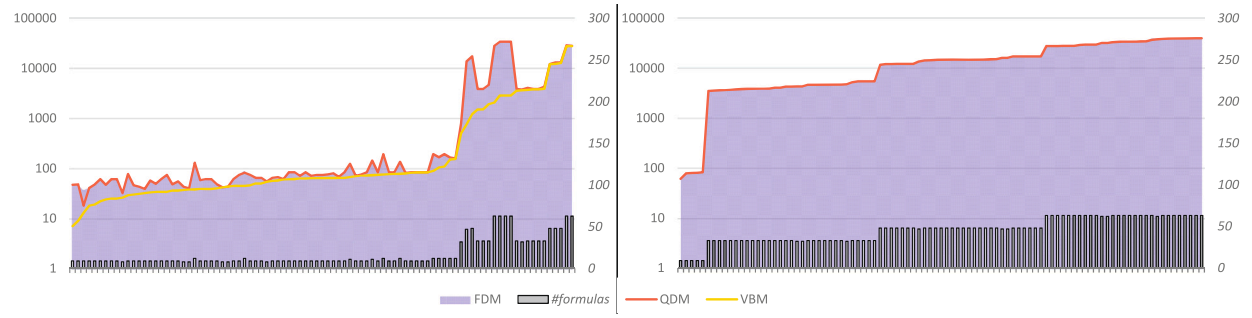


Fig. E.26. Artificial Spreadsheets: Results for the single spreadsheets in terms of the **number of returned diagnoses of cardinality 3** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where the computations for all model types finished successfully (*no timeout triggered*), whereas the *right plot* describes all cases where computations for *VBM* could not be finished (*timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis in each plot.

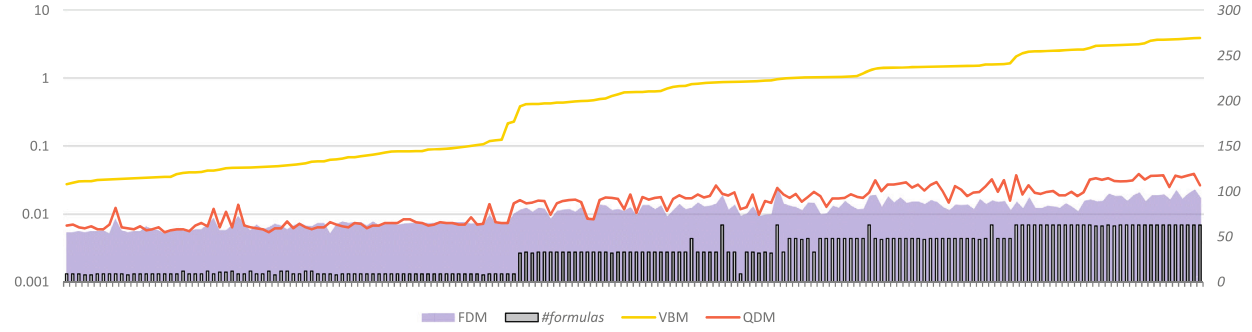


Fig. E.27. Artificial Spreadsheets: Results for the single spreadsheets in terms of the **time (in seconds) required for the computation of all minimal diagnoses of cardinality 1** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM*. Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis.

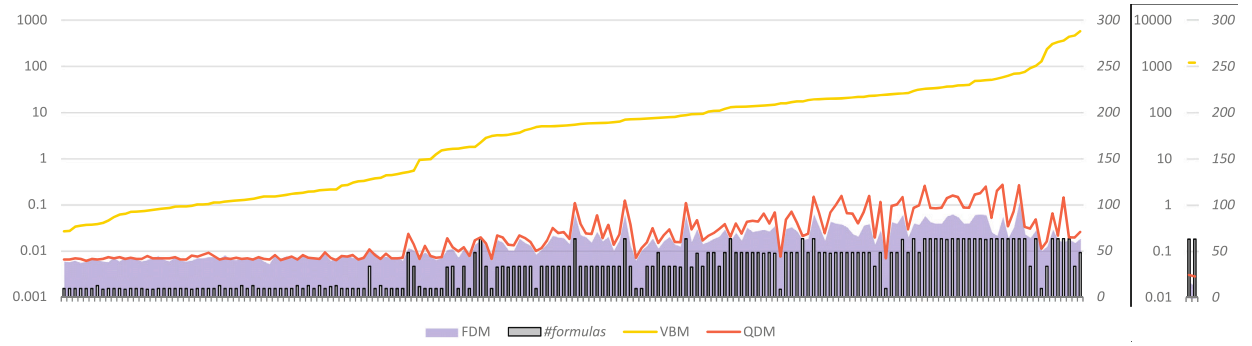


Fig. E.28. Artificial Spreadsheets: Results for the single spreadsheets in terms of the **time (in seconds) required for the computation of all minimal diagnoses of cardinality 2** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where computations finished successfully for all model types (*no timeout was triggered*), whereas the *right plot* depicts the cases where computations for *VBM* could not be finished (*timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Note, the displayed times (corresponding to the timeout of 1200 seconds) for *VBM* in the right plot are lower bounds of the actual times. Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis.

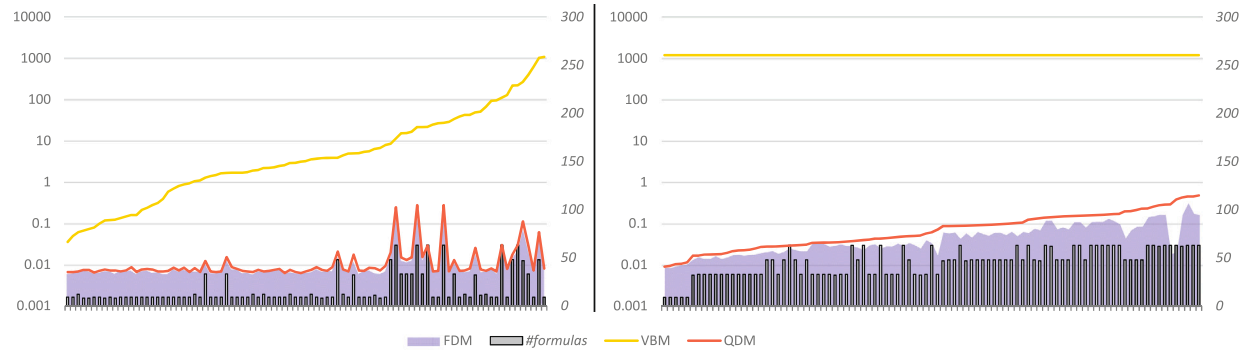


Fig. E.29. Artificial Spreadsheets: Results for the single spreadsheets in terms of the **time (in seconds) required for the computation of all minimal diagnoses of cardinality 3** w.r.t. each model type in $\{VBM, QDM, FDM\}$ (see lines and shaded area). The *left plot* shows only the cases where computations could be finished for all model types (*no timeout was triggered*), whereas the *right plot* depicts the cases where computations for *VBM* could not be finished (*a timeout was triggered*). The grey bars (# formulas) indicate the number of formulas in the spreadsheet. The area between each two tick marks on the x-axis refers to one spreadsheet. Cases along the x-axis are sorted in ascending order w.r.t. the values for *VBM* (left plot) and *QDM* (right plot). Note, the displayed times (corresponding to the timeout of 1200 seconds) for *VBM* in the right plot are lower bounds of the actual times. Line charts (instead of dots) were used for better clarity. The values for the model type *FDM* are depicted by a shaded area to better visualize the slight differences between *FDM* and *QDM*. Features written in normal/italic font in the legend are plotted w.r.t. the left/right y-axis.

Data availability

We share a link to the used software and datasets as well as to the raw results obtained from our experiments in the paper.

References

[1] R. Abraham, M. Erwig, GoalDebug: a spreadsheet debugger for end users, in: International Conf. on Software Engineering (ICSE), 2007.
[2] R. Abreu, B. Hofer, A. Perez, F. Wotawa, Using constraints to diagnose faulty spreadsheets, Softw. Qual. J. 23 (2) (2015) 297–322.
[3] R. Abreu, P. Zoeteweij, A. Van Gemund, On the accuracy of spectrum-based fault localization, in: Testing: Academic and Industrial Conference Practice and Research Techniques (TAICPART), 2007.
[4] S. Ausserlechner, S. Frühmann, W. Wieser, B. Hofer, R. Spork, C. Mühlbacher, F. Wotawa, The right choice matters! SMT solving substantially improves model-based debugging of spreadsheets, in: International Conf. on Quality Software (SWQD)., 2013.
[5] K. Autio, R. Reiter, Structural abstraction in model-based diagnosis, in: European Conf. on Artificial Intelligence (ECAI), 1998.
[6] D. Barowy, D. Gochev, E. Berger, CheckCell: data debugging for spreadsheets, in: International Conf. on Object Oriented Programming Systems Languages & Applications (OOPSLA), 2014.
[7] D. Barowy, E. Berger, B. Zorn, ExceLint: automatically finding spreadsheet formula errors, in: International Conf. on Object Oriented Programming Systems Languages & Applications (OOPSLA), 2018.
[8] G.W. Bond, Logic Programs for Consistency-Based Diagnosis, Ph.D. thesis, Carleton University, Faculty of Engineering, Ottawa, Canada, 1994.
[9] S.C. Cheung, W. Chen, Y. Liu, C. Xu, CUSTODES: automatic spreadsheet cell clustering and smell detection using strong and weak features, in: International Conf. on Software Engineering (ICSE), 2016.
[10] L. Chittaro, R. Ranon, Hierarchical model-based diagnosis based on structural abstraction, Artif. Intell. 155 (1–2) (2004) 147–182.

[11] R. Ceballos, R. Gasca, C. Del Valle, M. Toro, Max-CSP approach for software diagnosis, in: Advances in Artificial Intelligence: 8th Ibero-American Conference on AI (IBERAMIA), 2002.
[12] L. Console, G. Friedrich, D.T. Dupré, Model-based diagnosis meets error diagnosis in logic programs, in: International Joint Conf. on Artificial Intelligence (IJCAI), 1993.
[13] P. Cousot, R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction of approximation of fixpoints, in: Symposium on Principles of Programming Languages (POPL), 1977.
[14] R. Davis, Diagnostic reasoning based on structure and behavior, Artif. Intell. 24 (1984) 347–410.
[15] J. De Kleer, A.K. Mackworth, R. Reiter, Characterizing diagnosis and systems, Artif. Intell. 56 (1992).
[16] J. De Kleer, B.C. Williams, Diagnosing multiple faults, Artif. Intell. 32 (1) (1987) 97–130.
[17] L. De Moura, N. Bjørner, Z3: an efficient SMT solver, in: Theory and Practice of Software, International Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS/ETAPS), 2008.
[18] R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.
[19] W. Dou, S.-C. Cheung, J. Wei, Is spreadsheet ambiguity harmful? Detecting and repairing spreadsheet smells due to ambiguous computation, in: International Conf. on Software Engineering (ICSE), 2014.
[20] W. Dou, C. Xu, S.C. Cheung, J. Wei, CACheck: detecting and repairing cell arrays in spreadsheets, IEEE Trans. Softw. Eng. 43 (3) (2017) 226–251.
[21] A. Feldman, G. Provan, A. Van Gemund, Approximate model-based diagnosis using greedy stochastic search, J. Artif. Intell. Res. 38 (2010) 371–413.
[22] A. Feldman, G. Provan, J. De Kleer, S. Robert, A. Van Gemund, Solving model-based diagnosis problems with Max-SAT solvers and vice versa, in: International Workshop on Principles of Diagnosis (DX), 2010.
[23] A. Felfernig, G. Friedrich, D. Jannach, M. Stumptner, M. Zanker, Hierarchical diagnosis of large configurator knowledge bases, in: Proceedings of the 2001 Joint German/Austrian Conference on Artificial Intelligence (KI 2001), 2001, pp. 185–197.
[24] A. Felfernig, E. Teppan, G. Friedrich, K. Isak, Intelligent debugging and repair of utility constraint sets in knowledge-based recommender applications, in: International Conference on Intelligent User Interfaces (IUI), 2008.
[25] G. Friedrich, M. Stumptner, F. Wotawa, Model-based diagnosis of hardware designs, Artif. Intell. 111 (2) (1999) 3–39.
[26] G. Friedrich, S. Rass, K. Shchekotykhin, A general method for diagnosing axioms, in: International Workshop on Principles of Diagnosis (DX), 2006.
[27] I.P. Gent, C. Jefferson, I. Miguel, Minion: a fast, scalable, constraint solver, in: European Conference on Artificial Intelligence (ECAI), 2006.
[28] D. Gorinevsky, K. Dittmar, D. Mylaraswamy, E. Nwadiogbu, Model-based diagnostics for an aircraft auxiliary power unit, in: International Conference on Control Applications (CCA), 2002.
[29] R. Greiner, B.A. Smith, R.W. Wilkerson, A correction to the algorithm in Reiter's theory of diagnosis, Artif. Intell. 41 (1) (1989) 79–88.
[30] F. Hermans, B. Jansen, S. Roy, E. Aivaloglou, A. Swidan, D. Hoepelman, Spreadsheets are code: an overview of software engineering approaches applied to spreadsheets, in: International Conf. on Software Analysis, Evolution, and Reengineering (SANER), 2016.
[31] B. Hofer, A. Hoefler, F. Wotawa, Combining models for improved fault localization in spreadsheets, IEEE Trans. Reliab. 66 (1) (2017) 38–53.
[32] B. Hofer, I. Nica, F. Wotawa, AI for localizing faults in spreadsheets, in: International Conf. on Testing Software and Systems (ICTSS), 2017.
[33] B. Hofer, I. Nica, F. Wotawa, Qualitative deviation models for spreadsheet debugging, in: International Workshop on Program Debugging (IWPD) - ISSRE Workshops, 2017.
[34] B. Hofer, I. Nica, F. Wotawa, Qualitative deviation models vs. quantitative models for fault localization in spreadsheets, in: International Workshop on Qualitative Reasoning (QR), 2017.
[35] B. Hofer, A. Perez, R. Abreu, F. Wotawa, On the empirical evaluation of similarity coefficients for spreadsheets fault localization, Autom. Softw. Eng. 22 (1) (2015) 47–74.
[36] B. Hofer, F. Wotawa, Why does my spreadsheet compute wrong values?, in: International Symposium on Software Reliability Engineering (ISSRE), 2014.
[37] B. Hofer, D. Jannach, P. Koch, K. Schekotihin, F. Wotawa, Product metrics for spreadsheets - a systematic review, J. Syst. Softw. 175 (2021) 110910.
[38] D. Jannach, U. Engler, Toward model-based debugging of spreadsheet programs, in: Joint Conf. on Knowledge-Based Software Engineering (JCKBSE), 2010.
[39] D. Jannach, T. Schmitz, Model-based diagnosis of spreadsheet programs - a constraint-based debugging approach, Autom. Softw. Eng. 23 (1) (2014) 105–144.
[40] D. Jannach, T. Schmitz, B. Hofer, F. Wotawa, Avoiding, finding and fixing spreadsheet errors - a survey of automated approaches for spreadsheet QA, J. Syst. Softw. 94 (2014) 129–150.
[41] D. Jannach, T. Schmitz, K. Shchekotykhin, Parallel model-based diagnosis on multi-core computers, J. Artif. Intell. Res. 55 (2016) 835–887.
[42] B. Jansen, F. Hermans, Code Smells in Spreadsheet Formulas Revisited on an Industrial Dataset, International Conf. on Software Maintenance and Evolution (ICSME), 2015.
[43] U. Junker, Preferred explanations and relaxations for over-constrained problems, in: AAAI Conf. on Artificial Intelligence (AAAI), 2004.
[44] A. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. Rosson, G. Rothermel, M. Shaw, S. Wiedenbeck, The state of the art in end-user software engineering, ACM Comput. Surv. 43 (3) (2011) 21.
[45] P. Koch, K. Schekotihin, D. Jannach, B. Hofer, F. Wotawa, Metric-based fault prediction for spreadsheets, IEEE Trans. Softw. Eng. 47 (10) (2019).
[46] M. Liffiton, M. Moffitt, M. Pollack, K. Sakallah, Identifying conflicts in overconstrained temporal problems, in: International Joint Conf. on Artificial Intelligence (IJCAI), 2005.
[47] L. Lin, Y. Jiang, The computation of hitting sets: review and new algorithms, Inf. Process. Lett. 86 (4) (2003) 177–184.
[48] C. Mateis, M. Stumptner, F. Wotawa, Modeling Java programs for diagnosis, in: European Conf. on Artificial Intelligence (ECAI), 2000.
[49] W. Mayer, Static and hybrid analysis in model-based debugging, Ph.D. thesis, University of South Australia, Australia, 2007.
[50] A. Metodi, R. Stern, M. Kalech, M. Codish, A novel SAT-based approach to model-based diagnosis, J. Artif. Intell. Res. 51 (1) (2014) 377–411.
[51] M. Mosca, J.M. Basso, S.R. Verschoor, On speeding up factoring with quantum SAT solvers, Sci. Rep. 10 (1) (2020) 1–8.
[52] I. Mozetič, Hierarchical model-based diagnosis, Int. J. Man-Mach. Stud. 35 (1991) 329–362.
[53] I. Nica, I. Pill, T. Quaritsch, F. Wotawa, A route to success – a performance comparison of diagnosis algorithms, in: International Joint Conf. on Artificial Intelligence (IJCAI), 2013.
[54] I. Nica, F. Wotawa, ConDiag – computing minimal diagnoses using a constraint solver, in: International Workshop on Principles of Diagnosis (DX), 2012.
[55] R.R. Panko, Thinking is bad: implications of human error research for spreadsheet research and practice, arXiv:0801.3114, 2008.
[56] C. Prud'homme, J.-G. Fages, Choco-solver: a Java library for constraint programming, J. Open Source Softw. 78 (7) (2022) 4708.
[57] J. Reichwein, G. Rothermel, M. Burnett, Slicing spreadsheets: an integrated methodology for spreadsheet testing and debugging, in: Conf. on Domain-Specific Languages (DSL), 1999.
[58] R. Reiter, A theory of diagnosis from first principles, Artif. Intell. 32 (1) (1987) 57–95.
[59] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, K. Hernandez, Adaptive diagnosis in distributed systems, IEEE Trans. Neural Netw. 16 (5) (2005) 1088–1109.
[60] P. Rodler, K. Shchekotykhin, P. Fleiss, G. Friedrich, RIO: minimizing user interaction in ontology debugging, in: International Conference on Web Reasoning and Rule Systems (RR), 2013.
[61] P. Rodler, Interactive Debugging of Knowledge Bases, Ph.D. thesis, University of Klagenfurt, 2015, arXiv:1605.05950.
[62] P. Rodler, M. Herold, StaticHS: a variant of Reiter's hitting set tree for efficient sequential diagnosis, in: International Symposium on Combinatorial Search (SoCS), 2018.

[63] P. Rodler, D. Jannach, K. Schekotihin, P. Fleiss, Are query-based ontology debuggers really helping knowledge engineers?, Knowl.-Based Syst. 179 (2019) 92–107.
[64] P. Rodler, E. Teppan, D. Jannach, Randomized problem-relaxation solving for over-constrained schedules, in: International Conference on Principles of Knowledge Representation and Reasoning (KR), 2021.
[65] P. Rodler, Memory-limited model-based diagnosis, Artif. Intell. 305 (2022) 103681.
[66] P. Rodler, Random vs. Best-first: impact of sampling strategies on decision making in model-based diagnosis, in: AAAI Conf. on Artificial Intelligence (AAAI), 2022.
[67] P. Rodler, A formal proof and simple explanation of the QuickXplain algorithm, Artif. Intell. Rev. 55 (2022) 6185–6206.
[68] P. Rodler, One step at a time: an efficient approach to query-based ontology debugging, Knowl.-Based Syst. 251 (2022) 108987.
[69] P. Rodler, How should I compute my candidates? A taxonomy and classification of diagnosis computation algorithms, in: European Conference on Artificial Intelligence (ECAI), 2023.
[70] P. Rodler, DynamicHS: streamlining Reiter's hitting-set tree for sequential diagnosis, Inf. Sci. 627 (2023) 251–279.
[71] P. Rodler, Sequential model-based diagnosis by systematic search, Artif. Intell. 323 (2023) 103988.
[72] J. Ruthruff, M. Burnett, G. Rothermel, Interactive fault localization techniques in a spreadsheet environment, IEEE Trans. Softw. Eng. 32 (4) (2006) 213–239.
[73] J. Ruthruff, E. Creswick, M. Burnett, C. Cook, S. Prabhakararao, M. Fisher, M. Main, End-user software visualizations for fault localization, in: Symposium on Software Visualization (SoftVis), 2003.
[74] M. Sachenbacher, A. Malik, P. Struss, From electrics to emissions: experiences in applying model-based diagnosis to real problems in real cars, in: International Workshop on Principles of Diagnosis (DX), 1998.
[75] M. Sachenbacher, P. Struss, Automated qualitative domain abstraction, in: International Joint Conf. on Artificial Intelligence (IJCAI), 2003.
[76] M. Sachenbacher, P. Struss, Task-dependent qualitative domain abstraction, Artif. Intell. 162 (1–2) (2005) 121–143.
[77] E. Shapiro, Algorithmic Program Debugging, MIT Press, 1983.
[78] K. Shchektotykhin, G. Friedrich, P. Fleiss, P. Rodler, Interactive ontology debugging: two query strategies for efficient fault localization, J. Web Semant. 12 (2012) 88–103.
[79] K. Shchektotykhin, G. Friedrich, P. Rodler, P. Fleiss, Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation, in: European Conf. on Artificial Intelligence (ECAI), 2014.
[80] S. Zaman, G. Steinbauer, J. Maurer, P. Lepej, S. Uran, An integrated model-based diagnosis and repair architecture for ROS-based robot systems, in: International Conference on Robotics and Automation (ICRA), 2013.
[81] P. Struss, O. Dressler, "Physical negation" – integrating fault models into the general diagnostic engine, in: International Joint Conf. on Artificial Intelligence (IJCAI), 1989.
[82] P. Struss, What's in SD? Towards a theory of modeling for diagnosis, in: Readings in Model-Based Diagnosis, 1992, pp. 419–449.
[83] P. Struss, Deviation models revisited, in: International Workshop on Principles of Diagnosis (DX), 2004.
[84] M. Stumptner, F. Wotawa, Debugging functional programs, in: International Joint Conf. on Artificial Intelligence (IJCAI), 1999.
[85] M. Weiser, Programmers use slices when debugging, Commun. ACM 25 (7) (1982) 446–452.
[86] M. Weiser, Program slicing, IEEE Trans. Softw. Eng. 10 (4) (1984) 352–357.
[87] W.E. Wong, R. Gao, Y. Li, R. Abreu, F. Wotawa, A survey on software fault localization, IEEE Trans. Softw. Eng. 42 (8) (2016) 707–740.
[88] F. Wotawa, On the use of qualitative deviation models for diagnosis, in: International Workshop on Qualitative Reasoning (QR), 2016.