On the Usage of Value- and Dependency-based Models for Spreadsheet Debugging with SMT Solvers

Andrea Höfler

Structure

- Motivation
- Overview of my Master’s Thesis
- Model-based Spreadsheet Debugging
- Models
- Examples
- Value-based Verification Method
- Framework
- Evaluation

Motivation

- Millions of people use spreadsheet programs
  - 95% of the U.S. companies use spreadsheets
- Most of the time very complex
  - the average business spreadsheet has 60.000 cells
- Often contain errors
  - ~88% of spreadsheets investigated during 1995 – 2007 were erroneous
- Even “experts” make errors
  - ~63% of the spreadsheets contained errors

Panko: http://panko.shidler.hawaii.edu/SSR/
Brown & Gould: Experimental study of people creating spreadsheets

Overview of my Master's Thesis

- Overview of basic functionality of constraint-, SAT-, and SMT solvers
- SMT solver comparison
- 2 dependency-based models for Z3 (simple, sophisticated)
- Value-based verification method
- Integration of additional spreadsheet functions
- Further cases of coincidental correctness
- Comparison of value-based and dependency-based models
  - Runtime
  - Diagnoses quality
  - Faulty cell's distribution

Coincidental Correctness

- Cases where formulas might evaluate to the correct value, even though the formula is faulty.

Spreadsheet Functions
IF, SUMIF, COUNTIF, …
MIN, MAX, COUNT, SMALL, LARGE, …
Boolean
PRODUCT, SUMPRODUCT, POWER, MOD
ROUND, FLOOR, ABS, …
SIN, COS, …
…

Model-based Spreadsheet Debugging (MBSD)

- Needed:
  - Spreadsheet $\prod$
  - Failing test case T
  - Model
    - Constraint representation of $\prod$ and T
    - Not-abnormal variables to represent the cells' health state
- Uses:
  - SMT solver to find contradictions in the model
- Returns:
  - Possible faulty cells (diagnoses)

Models

Value-based Models	Dependency-based Models
Formulas	Dependencies
Equivalence (==)	Simple: implication (→), Sophisticated: bi-implication (↔, ==)
Values (integer, real,…)	Truth values (Boolean)
No coincidental correctness	Coincidental correctness (sophisticated)
NIRA problem	SAT problem
Currently only solvable with Z3	Solvable with any SAT- or SMT solver
More accurate	Faster

A1 == 1 + C2 * B2

A1 == 4

B2■C2 → A1

B2■C2 ↔ A1

A1 == true

Example

	A	B	C	D	E
1		Constant Acceleration	Constant Velocity	Constant Deceleration	Final State
2	Initial Velocity [m/s]	0,0	20,0	20,0	0,0
3	Acceleration [m/s^2]	2,0	0,0	-4,0	
4	Duration [s]	10,0	10.000,0	5,0	
5	Distance [m]	100,0	0,0	50,0	
6	Accumulated Distance [m]	100,0	100,0	150,0	

	A	B	C	D	E
1		Constant Acceleration	Constant Velocity	Constant Deceleration	Final State
2	Initial Velocity [m/s]	0	=B2+B3*B4	=C2+C3*C4	=D2+D3*D4
3	Acceleration [m/s^2]	2	0	-4	
4	Duration [s]	10	10000	5	
5	Distance [m]	=B2*B4+B3*B4*B4/2	=B2*C4+C3*C4*C4/2	=D2*D4+D3*D4*D4/2	
6	Accumulated Distance [m]	=B5	=B5+C5	=B5+C5+D5	

should be:
C2*C4+C3*C4*C4/2

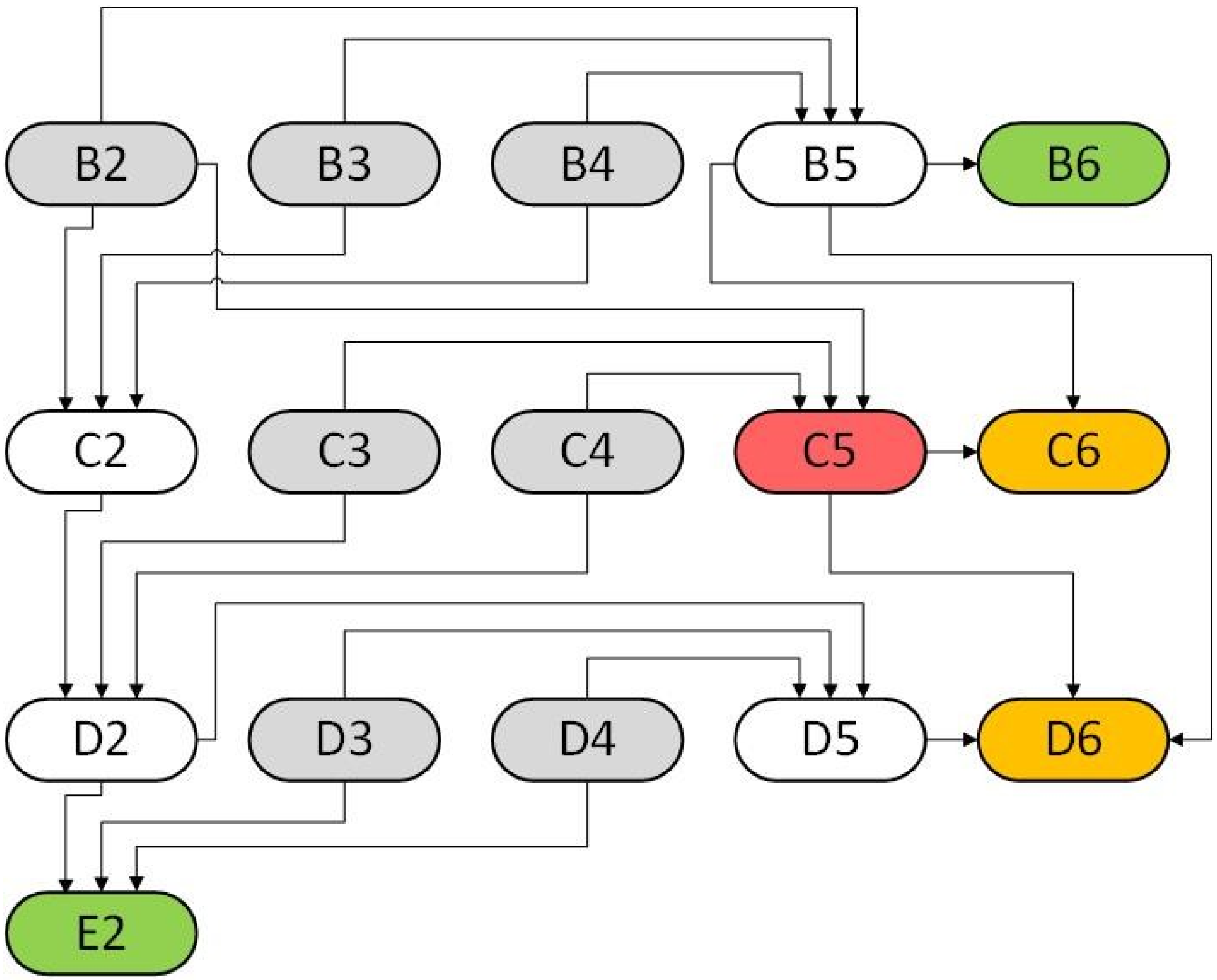
Example (Value-based)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	NAB(B5) → B5 == B2 * B4 + B3 * B4 * B4 / 2
E2 == D2 + D3 * D4	NAB(C2) → C2 == B2 + B3 * B4
	NAB(C5) → C5 == B2 * C4 + C3 * C4 * C4 / 2
	…

Result: (C5)

Dependency Graph



Example (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == true	B6 == true	C6 == false
B3 == true	E2 == true	D6 == false
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B5 → B6	NAB(B5) → [B2■B3■B4 → B5]
D2■D3■D4 → E2	NAB(C2) → [B2■B3■B4 → C2]
	NAB(C5) → [B2■C3■C4 → C5]
	…
Result: (B5), (C5)	

Example (Sophisticated)

Input cells	Correct output cells	Incorrect output cells
B2 == true	B6 == true	C6 == false
B3 == true	E2 == true	D6 == false
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B5 ↔ B6	NAB(B5) → [B2■B3■B4 → B5]
D2■D3■D4 ↔ E2	NAB(C2) → [B2■B3■B4 ↔ C2]
	NAB(C5) → [B2■C3■C4 → C5]
	…

Result: (C5)

Value-based Verification (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	NAB(B5) → B5 == B2 * B4 + B3 * B4 * B4 / 2
E2 == D2 + D3 * D4	NAB(C2) → C2 == B2 + B3 * B4
	NAB(C5) → C5 == B2 * C4 + C3 * C4 * C4 / 2
	…

Value-based Verification (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	NAB(B5) → B5 == B2 * B4 + B3 * B4 * B4 / 2
E2 == D2 + D3 * D4	NAB(C2) → C2 == B2 + B3 * B4
	NAB(C5) → C5 == B2 * C4 + C3 * C4 * C4 / 2
	…

Value-based Verification (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	B5 == B2 * B4 + B3 * B4 * B4 / 2
E2 == D2 + D3 * D4	C2 == B2 + B3 * B4
	C5 == B2 * C4 + C3 * C4 * C4 / 2
	…

Value-based Verification (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	B5 == B2 * B4 + B3 * B4 * B4 / 2
E2 == D2 + D3 * D4	C2 == B2 + B3 * B4
	C5 == B2 * C4 + C3 * C4 * C4 / 2
	…

Value-based Verification (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	
E2 == D2 + D3 * D4	C2 == B2 + B3 * B4
	C5 == B2 * C4 + C3 * C4 * C4 / 2
	…
Result: SAT→ (C5) = High priority diagnosis (HPD)	

Value-based Verification (Simple)

Input cells	Correct output cells	Incorrect output cells
B2 == 0	B6 == 100	C6 == 200,100
B3 == 2	E2 == 0	D6 == 200,150
…		

Cells NOT connected to incorrect output cells	Cells connected to incorrect output cells
B6 == B5	B5 == B2 * B4 + B3 * B4 * B4 / 2
E2 == D2 + D3 * D4	C2 == B2 + B3 * B4
	…

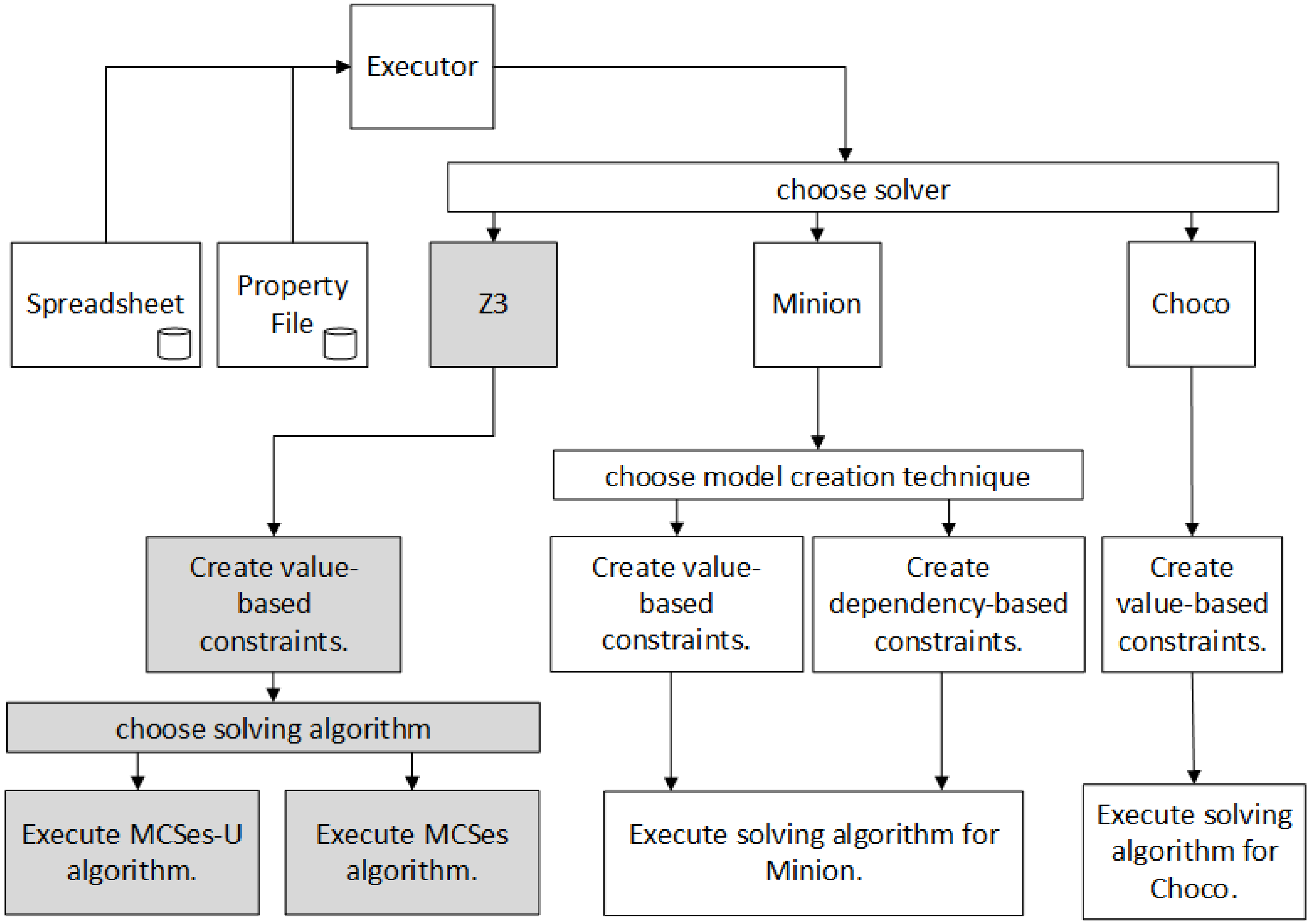
Result: UNSAT→ (B5) = Low priority diagnosis (LPD)

Value-based Verification

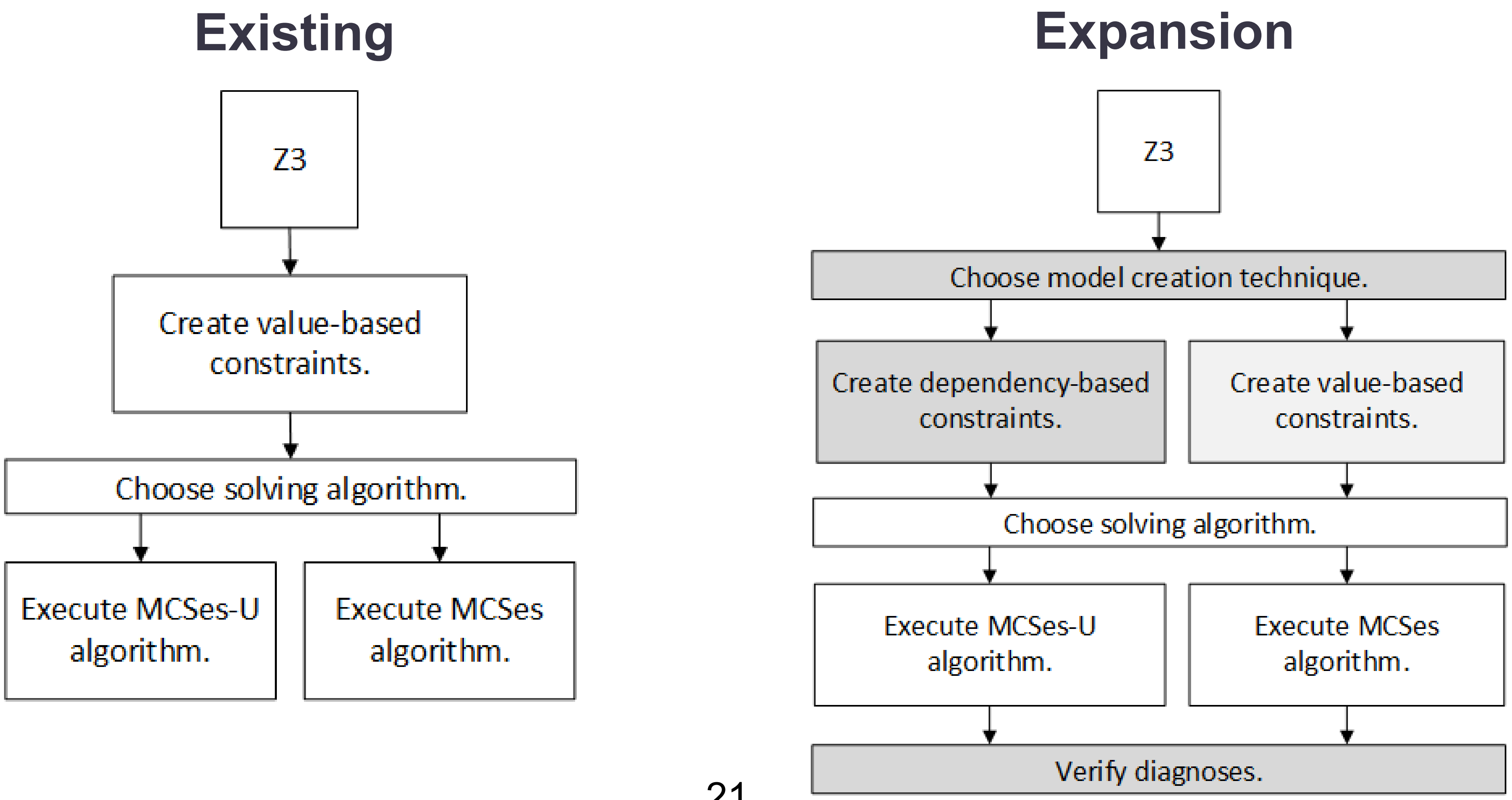
- Improves the quality of the dependency-based diagnoses
- Not useful for value-based approach
  - Already models the cells‘ formulas
  - Therefore, no further improvement possible

- Currently only applicable with Z3

Framework


Executor
choose solver
Spreadsheet
Property File
Z3
Minion
Choco
choose model creation technique
Create value-based constraints.
choose solving algorithm
Execute MCSes-U algorithm.
Execute MCSes algorithm.
Create value-based constraints.
Create dependency-based constraints.
Create value-based constraints.
Execute solving algorithm for Minion.
Execute solving algorithm for Choco.

Framework


Existing
Z3
Create value-based constraints.
Choose solving algorithm.
Execute MCSes-U algorithm.
Execute MCSes algorithm.
Expansion
Z3
Choose model creation technique.
Create dependency-based constraints.
Create value-based constraints.
Choose solving algorithm.
Execute MCSes-U algorithm.
Execute MCSes algorithm.
Verify diagnoses.

EUSES Spreadsheet Corpus

- Publicly available
- Created by Marc Fisher and Gregg Rothermel
- Consists of 4498 spreadsheets found through web search
  - Financial reports
  - Grading sheets
  - Private calculations
- Many not suitable for spreadsheet debugging
  - Not supported spreadsheet functions
  - Forms

Mutated EUSES Spreadsheet Corpus

- Single-fault corpus*
  - 267 spreadsheets
- Multi-fault corpus
  - 217 spreadsheets
    - 122 double fault spreadsheets
    - 95 triple fault spreadsheets
- Integer and real numbers
- 6 to 604 formula cells
  - Average 105 formula cells

* Source: Hofer et al.

Evaluation

- Comparison of the runtime behavior
  - Single-fault corpus
- Comparison of the diagnoses quality
  - Single-fault corpus
- Comparison of the faulty cells' distribution
  - Multi-fault corpus

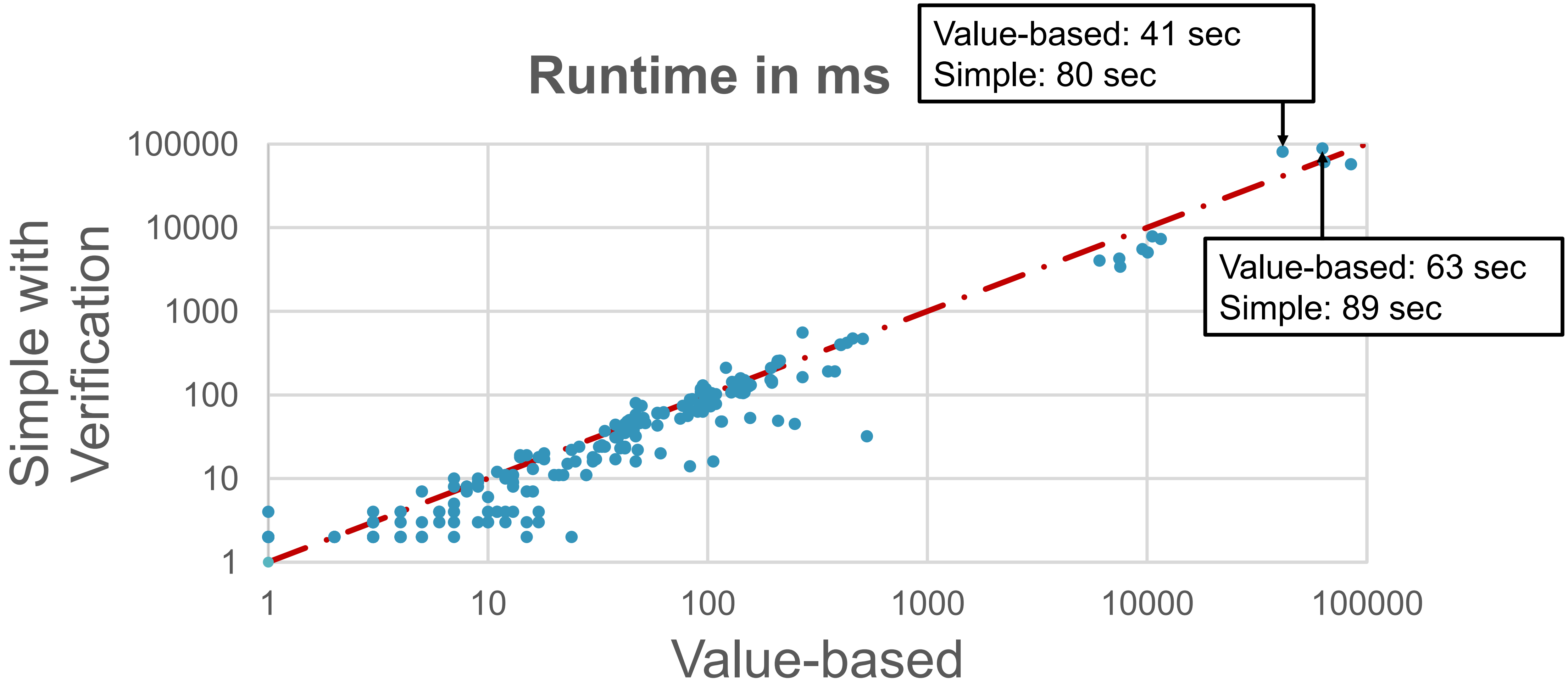
Runtime Comparison

	Value-based	Simple with Verification	Sophisticated with Verification	Simple without Verification	Sophisticated without Verification
Accum. Avg. Runtime	330 sec	337 sec	235 sec	94 sec	44 sec
Accum. Avg. Runtime in %	743 %	761 %	530 %	213 %	100 %

Runtime Comparison

	Value-based	Simple with Verification	Sophisticated with Verification
Value-based	-	72 %	62 %
Simple with Verification	28 %	-	41 %
Sophisticated with Verification	38 %	59 %	-

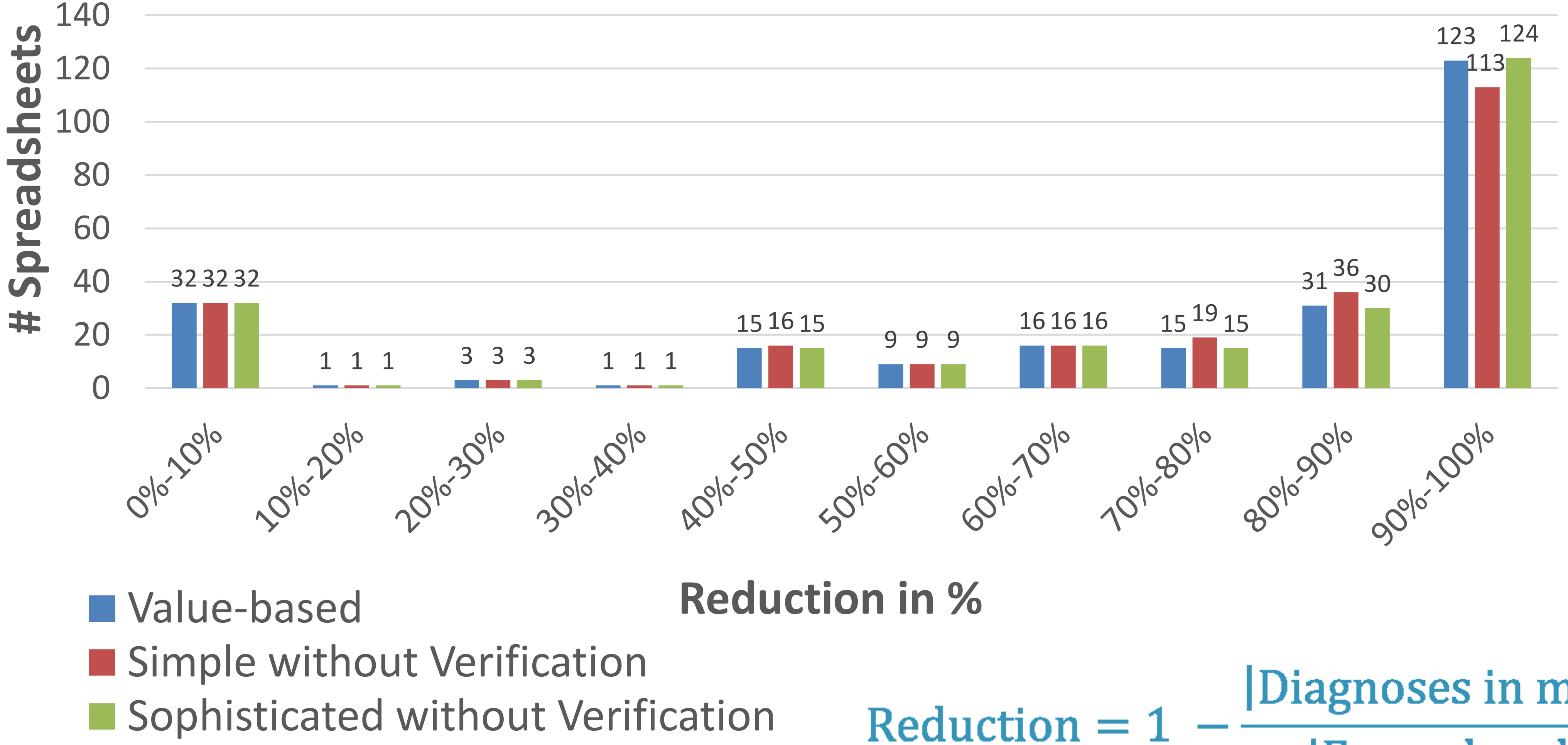
Runtime Comparison



Diagnosis Comparison

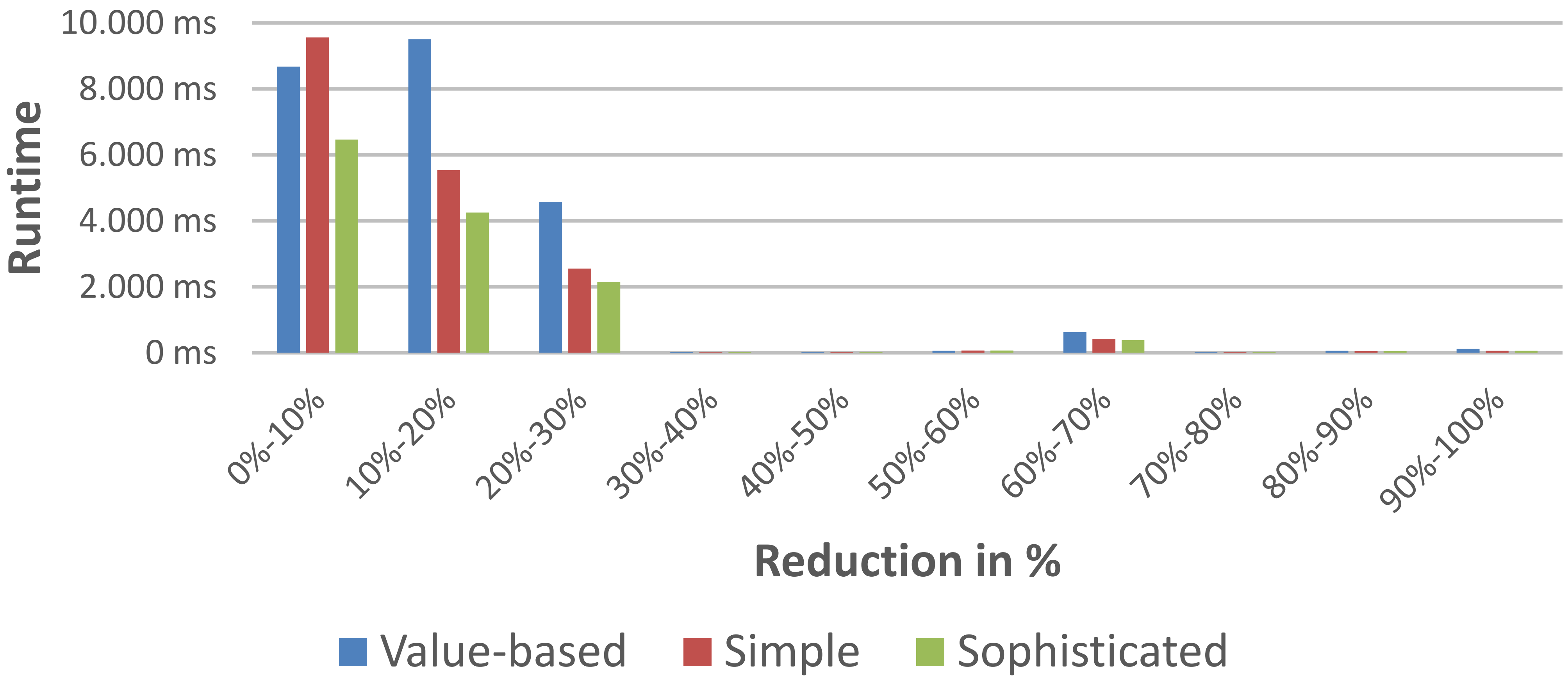
		Value-based	Simple with Verification	Sophisticated with Verification	Simple without Verification	Sophisticated without Verification
	Diagnoses	6400	6400 HPDs 155 LPDs	6399 HPDs 14 LPDs	6555	6413
Compared to Value-based	Absolute	-	0	-1	+155	+13
	In percentage	-	0 %	-0.02 %	+2.4 %	+0.2 %
	Per spreadsheet	-	0	0	+0.6	+0.1

Diagnoses Quality



$$\text{Reduction} = 1 - \frac{|\text{Diagnoses in model}|}{|\text{Formula cells}|}$$

Diagnoses Quality and Runtime



Conclusion

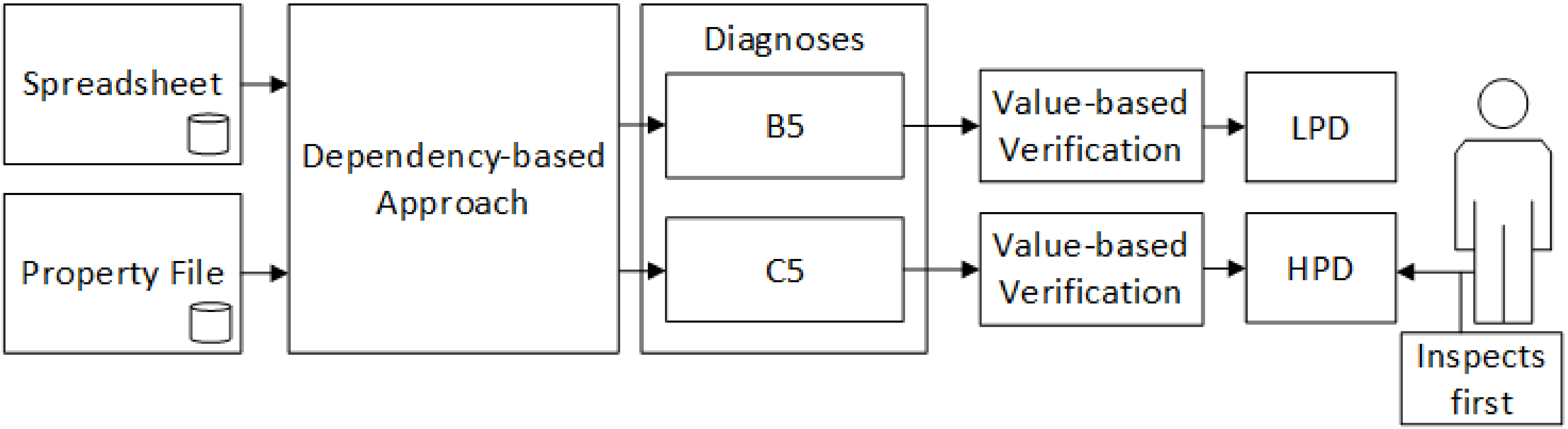
- Dependency-based approaches on average faster
  - Even with verification method
- Diagnosis quality higher for the value-based approach
  - Verification method improves quality to equal that of the value-based approach
- All approaches can aid the user in spreadsheet debugging
- Future Work
  - Integrate additional SMT Solvers
  - Compare their performance to Z3

Conclusion

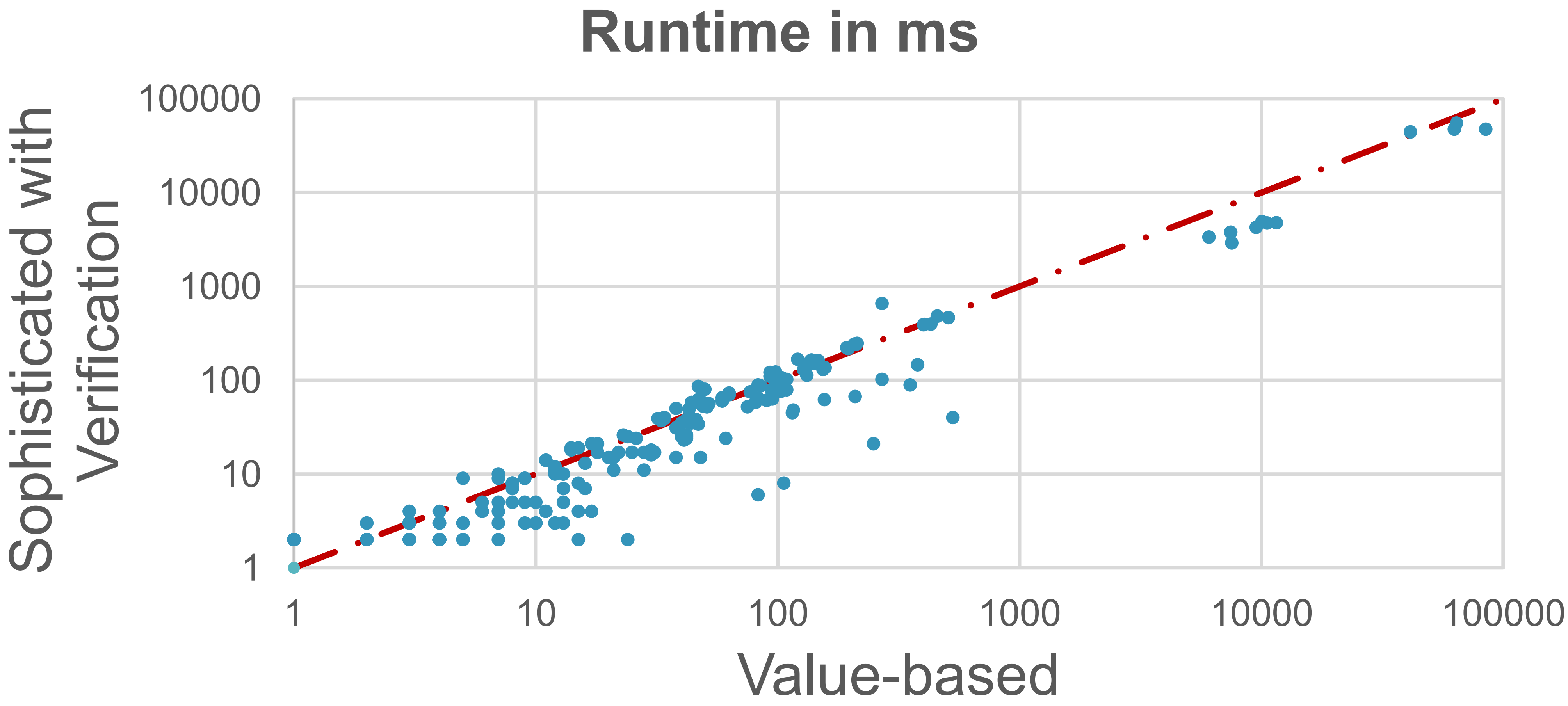
- Overview of basic functionality of constraint-, SAT-, and SMT solvers
- SMT solver comparison
- 2 dependency-based models for Z3 (simple, sophisticated)
- Value-based verification method
- Integration of additional spreadsheet functions
- Cases of coincidental correctness
- Comparison of value-based and dependency-based models
  - Runtime
  - Diagnoses quality
  - Faulty cell's distribution

Thank you for your attention!

Value-based Verification



Runtime Comparison



Runtime Comparison