

Fuzz Testen van Constraint Programming

Ruben Kindt | R0656495

Constraint Programming

- Wat is CP
- Wat maakt CP anders dan andere programmeer talen

Constraint Programming

- Wat is CP
- Wat maakt CP anders dan andere programmeer talen
- CPMpy

SEND
+ MORE

MONEY

Constraint Programming



- Wat is CP
- Wat maakt CP anders
- CPMpy

*SEND
+ MORE

MONEY*

```
1  from cpmPy import *
2  import numpy as np
3
4  s,e,n,d,m,o,r,y = intvar(lb=0,ub=9, shape=8)
5
6  model = Model()
7  model += sum([s,e,n,d] * np.array([1000, 100, 10, 1])) \
8            + sum([m,o,r,e] * np.array([1000, 100, 10, 1])) \
9            == sum([m,o,n,e,y] * np.array([10000, 1000, 100, 10, 1]))
10 model += s > 0
11 model += m > 0
12 model += AllDifferent([s,e,n,d,m,o,r,y])
13
14 model.solve(solver="minizinc:chuffed")
15 print("  S,E,N,D =   ", [x.value() for x in [s,e,n,d]])
16 print("  M,O,R,E =   ", [x.value() for x in [m,o,r,e]])
17 print("M,O,N,E,Y = ", [x.value() for x in [m,o,n,e,y]])
```

Waarom Bugs zoeken?

- Bugs zitten overal
- Definitie bug
 - Crash 
 - Vast hangen 
 - Wrongly unsatisfiable
 - Wrongly satisfiable
 - Verkeerd aantal oplossingen

Wat is Fuzz Testen?

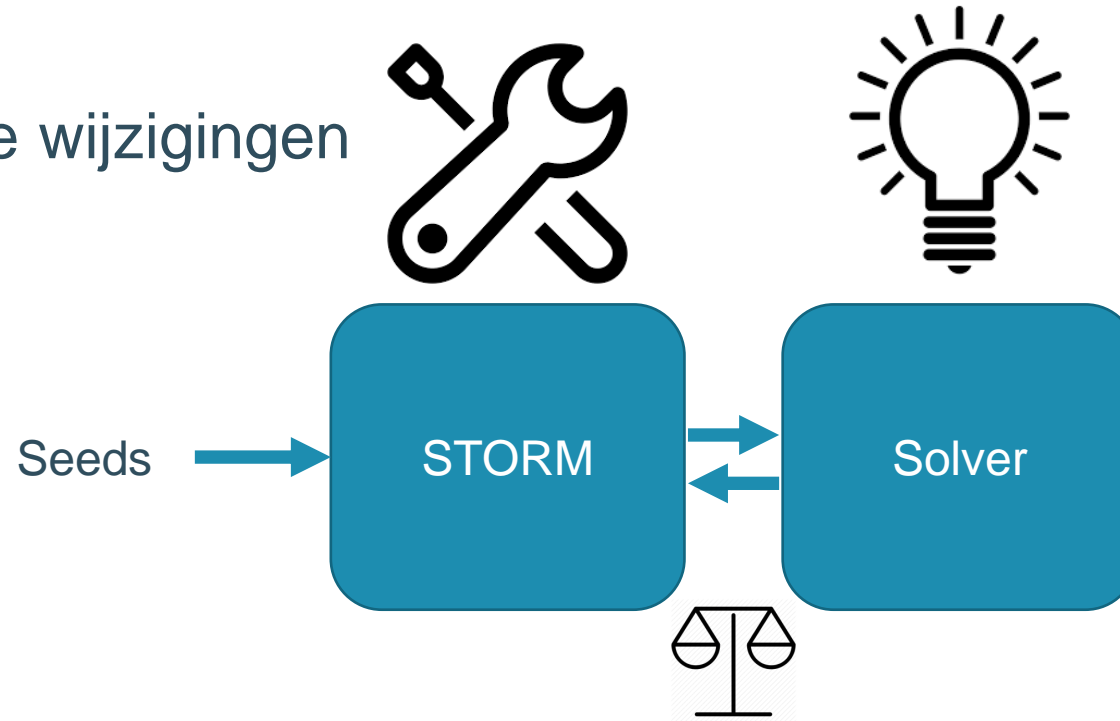
- Wat is fuzzing
- Waarom fuzzing
- Ground truth

Onderzoeksvragen

1. Welke technieken vinden de meeste bugs?
2. Welke technieken vinden de meeste kritieke bugs?
3. Welke type bugs worden er gevonden met welke techniek?
4. Hoeveel en hoe erg zijn de gevonden (kritieke) bugs?
5. Waar liggen de oorzaken van de bugs?

SMT fuzz tester: STORM^[1]

1. Test origineel
2. Satisfiable equivalente wijzigingen
3. Test opnieuw

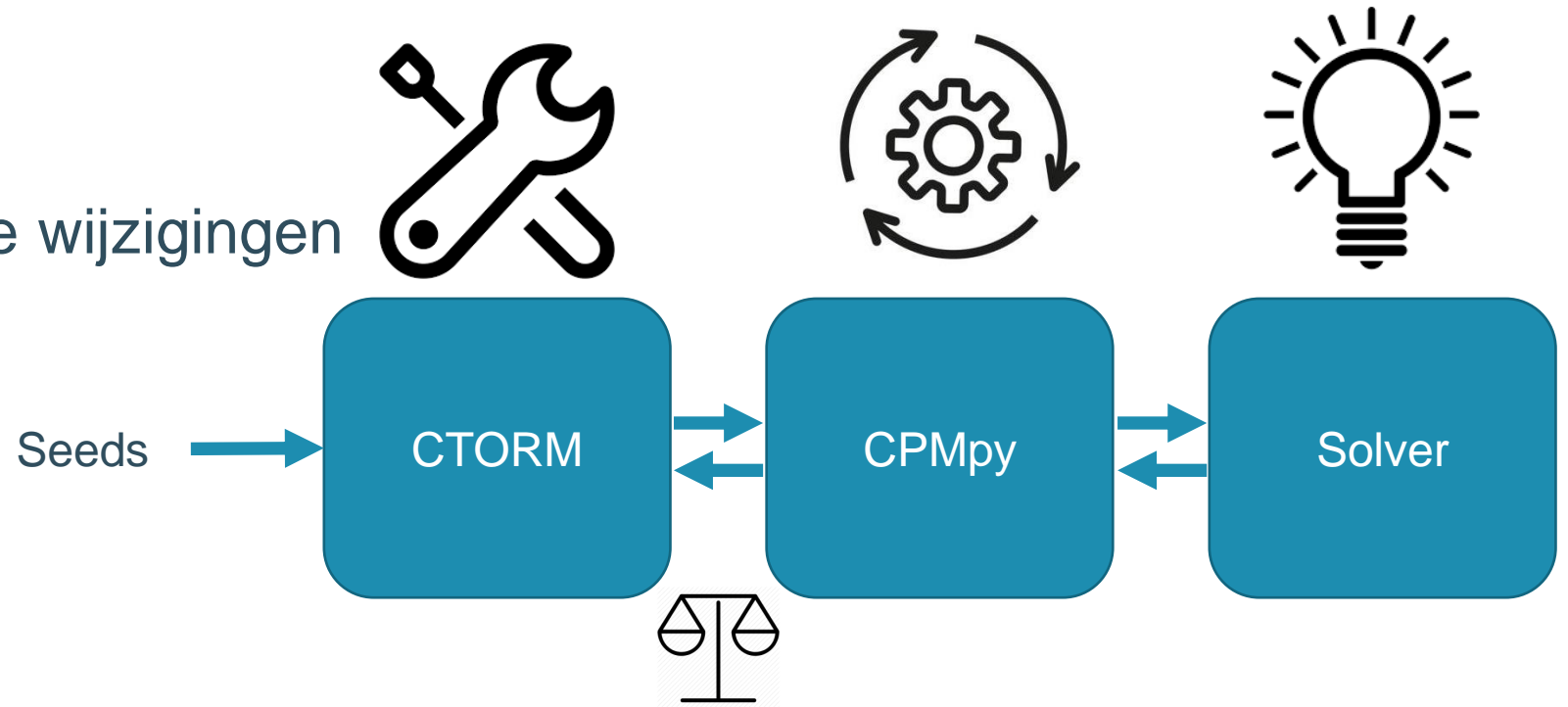


[1] Muhammad Numair Mansur et al. “Detecting critical bugs in SMT solvers using blackbox mutational fuzzing”. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020, pp. 701–712.

STORM naar CTORM

CPMpy-STORM

1. Test origineel
2. Satisfiable equivalente wijzigingen
3. Test opnieuw

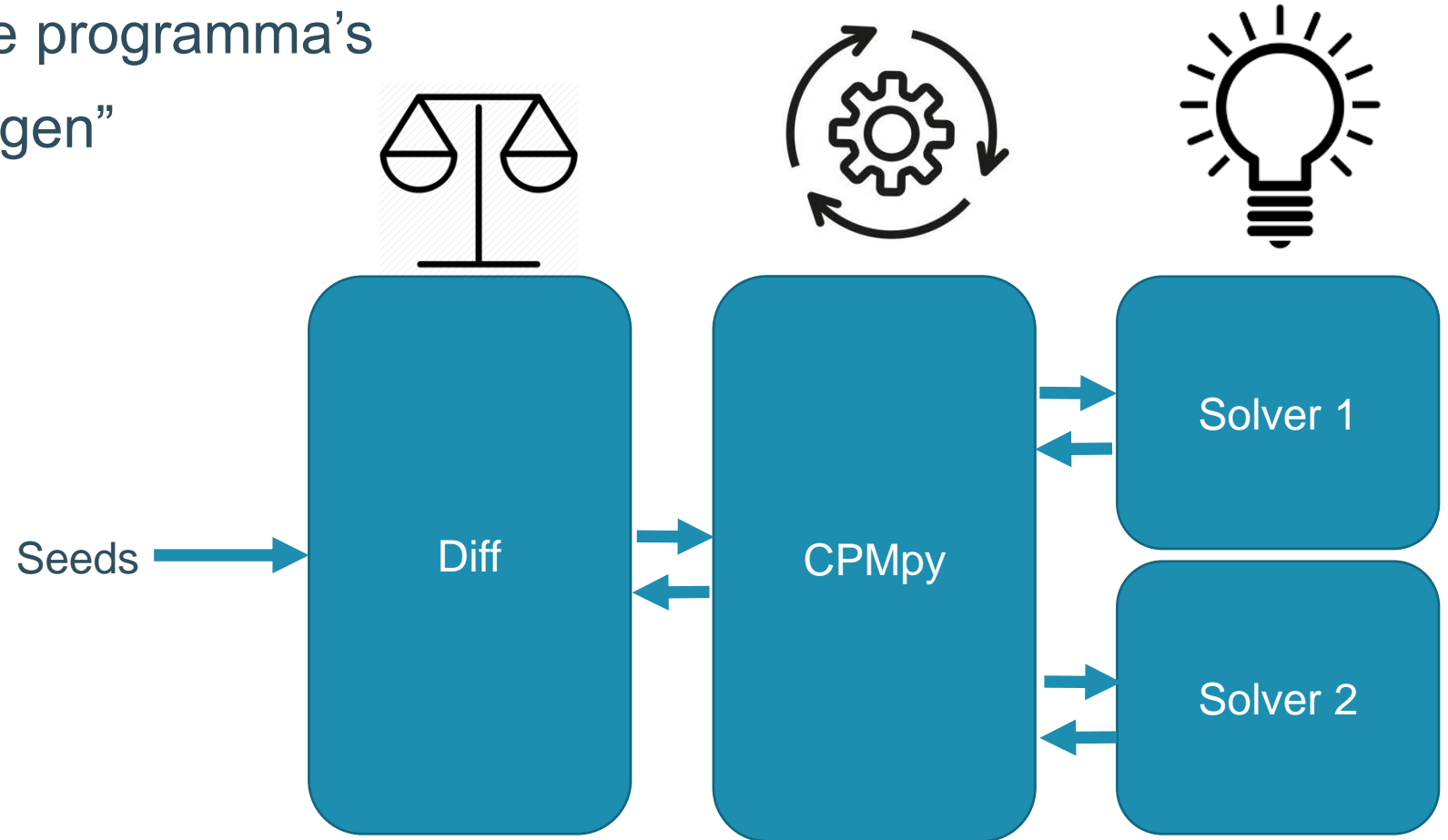


Metamorphic Testen

- Andere (maar equivalente) manier oplossen
- Enkele Metamorphic transformaties
 - Alldifferent([var1, var2, var3]) wordt [var1 != var2, var2!=var3, var3!=var1]
 - (Boolean) constraint wordt constraint and True
 - $A \leq B$ wordt $A < (B+1)$
 - ...
- Flexibiliteit

Differentiël testen

- Vergelijken met analoge programma's
- Ook “zoek alle oplossingen”

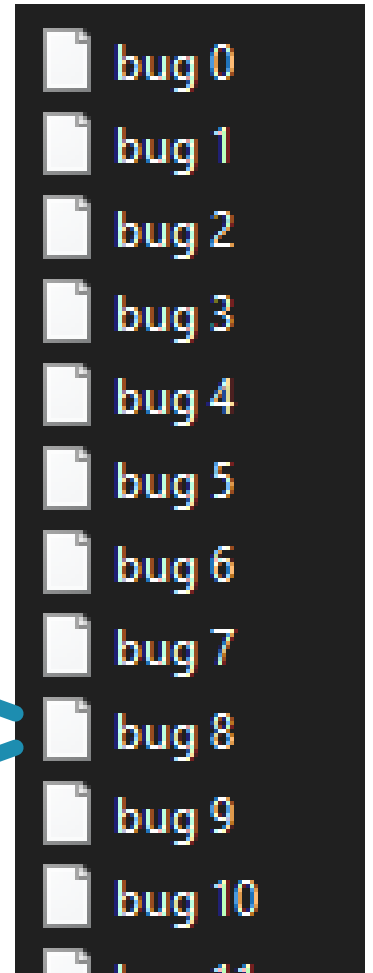


Resultaten Verwerken

- Frequent gelijkaardige bugs
 - Origineel plan: deobfuscatie (MUS) + deduplicatie
 - Gewijzigd plan: filteren dan deobfuscatie (MUS)

Variable declaration 1
Variable declaration 2

Constraint 1
Constraint 2
Constraint 3
Constraint 4
Constraint 5
Constraint 6



Resultaten: *Double negatie-bug*

```
1 from cpmPy import *
2 from cpmPy.transformations.flatten_model import flatten_model
3
4 X = intvar(lb=0, ub=9, name='X')
5 m = Model()
6 m += X == 3
7 m += ~(~(X == 3))
8
9 m.solve(solver="gurobi")
10 print(m.status().exitstatus.name) # UNSATISFIABLE
11 m.solve(solver="ortools")
12 print(m.status().exitstatus.name) # UNSATISFIABLE
13
14 m.solve(solver="minizinc:chuffed")
15 print(m.status().exitstatus.name) # FEASIBLE
16
17 print(m)
18 mf = flatten_model(m)
19 print(mf)
```

Constraints:

$X == 3$

$X == 3 == 0 == 0$

Constraints:

$X == 3$

$X != 3$

Resultaten: *Benaming van variabelen*

```
1 from cpmPy import *
2
3 i = intvar(lb=0, ub=9, name="+i")
4 m = Model()
5 m += i > 0
6
7 m.solve(solver="minizinc:chuffed") # Crash
```

Resultaten: *Benaming van variabelen*

```
1  from cpmPy import *
2
3  i = intvar(lb=0, ub=9, name="+i")
4  m = Model()
5  m += i > 0
6
7  # m.solve(solver="minizinc:chuffed") # Crash
8
9  s = SolverLookup.get("minizinc:chuffed", m)
10 print("".join(map(str, s.mzn_model._code_fragments)))
```

% Generated by CPMpy
include "globals.mzn";

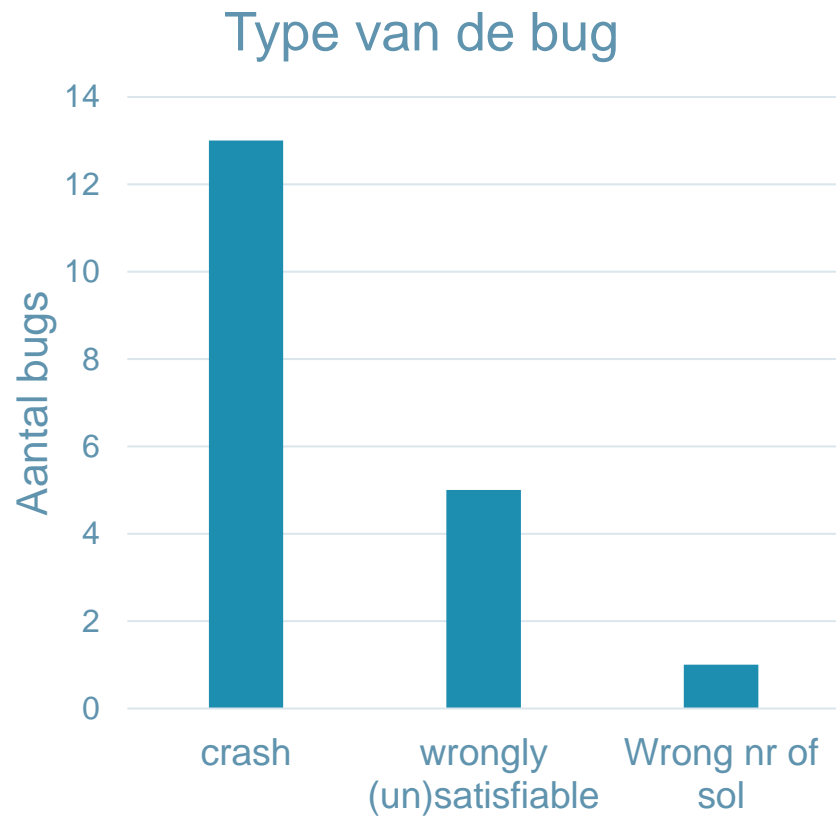
var 0..9: ~~+~~i;
constraint (+i) > 0;

Resultaten

Model, Transformation, Solver	Crash, wrongly (un)sat or wrong nr of Solutions	Bug nr	solver independent Bug	OR-Tools	Gurobi	MiniZinc subsolvers	PySAT subsolvers
Model	crash	145	Diff				
Model	UNSATISFIABLE	158	Meta				
Model	UNSATISFIABLE	161	CTORM, Meta				
Transformation	UNSATISFIABLE	142		CTORM	CTORM		
Transformation	crash	143		CTORM, Meta	CTORM, Meta		
Transformation	crash	157	Meta				
Transformation	crash	164				Meta	
Transformation	crash	165				Meta, Diff	
Transformation	UNSATISFIABLE	168			CTORM, Meta, Diff		
Transformation	(UN)SATISFIABLE	170		CTORM, Meta	CTORM, Meta		
Solver Interface	crash	149			CTORM, Diff		
Solver Interface	crash	150					Diff
Solver Interface	crash	152	Meta, Diff				
Solver Interface	Wrong Nr of sol	153			Diff		
Solver Interface	crash	154				CTORM, Meta, Diff	
Solver Interface	crash	155			CTORM, Meta, Diff		
Solver Interface	crash	159			CTORM, Diff		
Solver Interface	crash	162				Meta	
Solver	crash	156				CTORM, Meta, Diff	

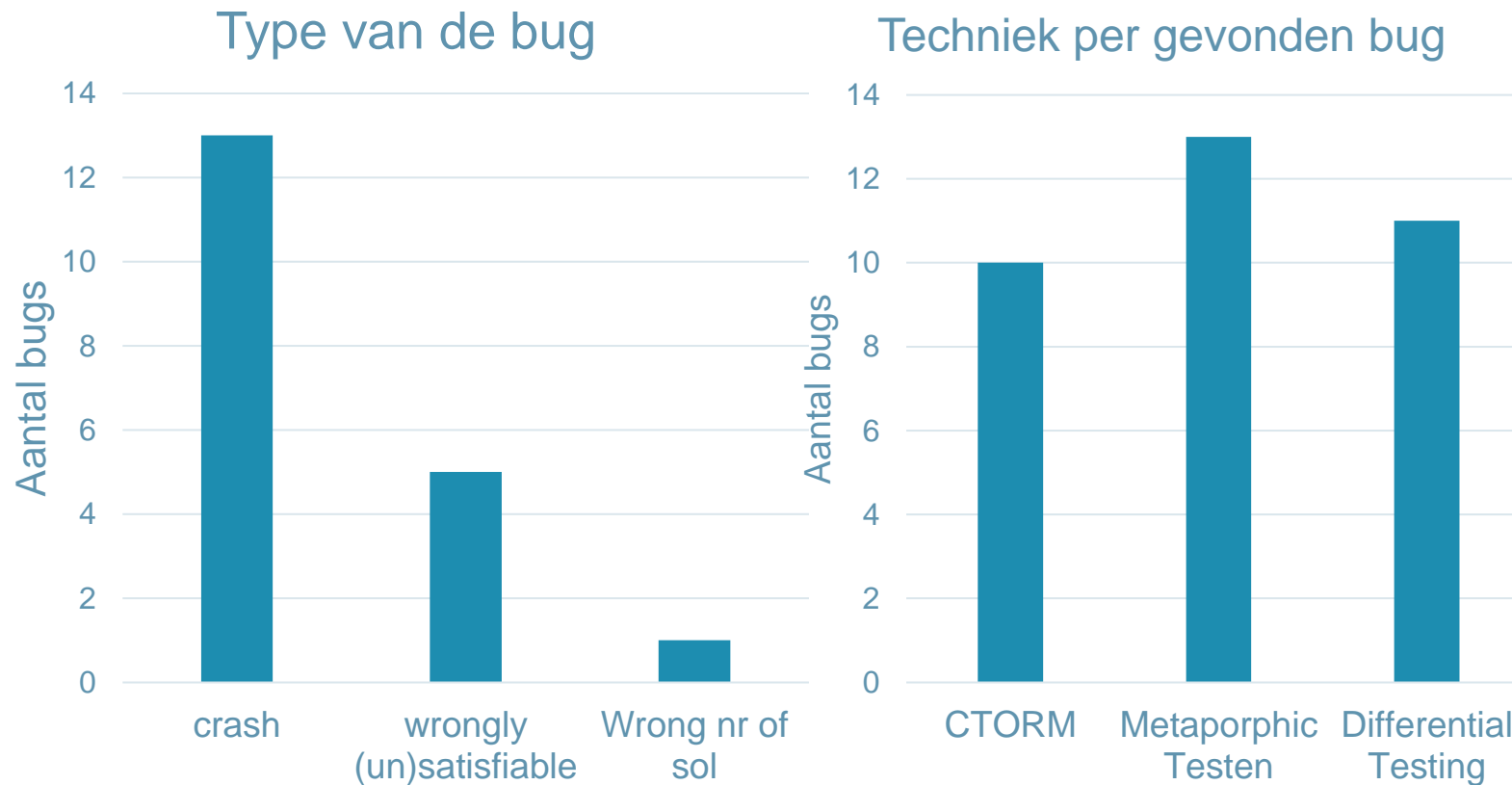
Resultaten

- 23 GitHub issues -> 19 bugs



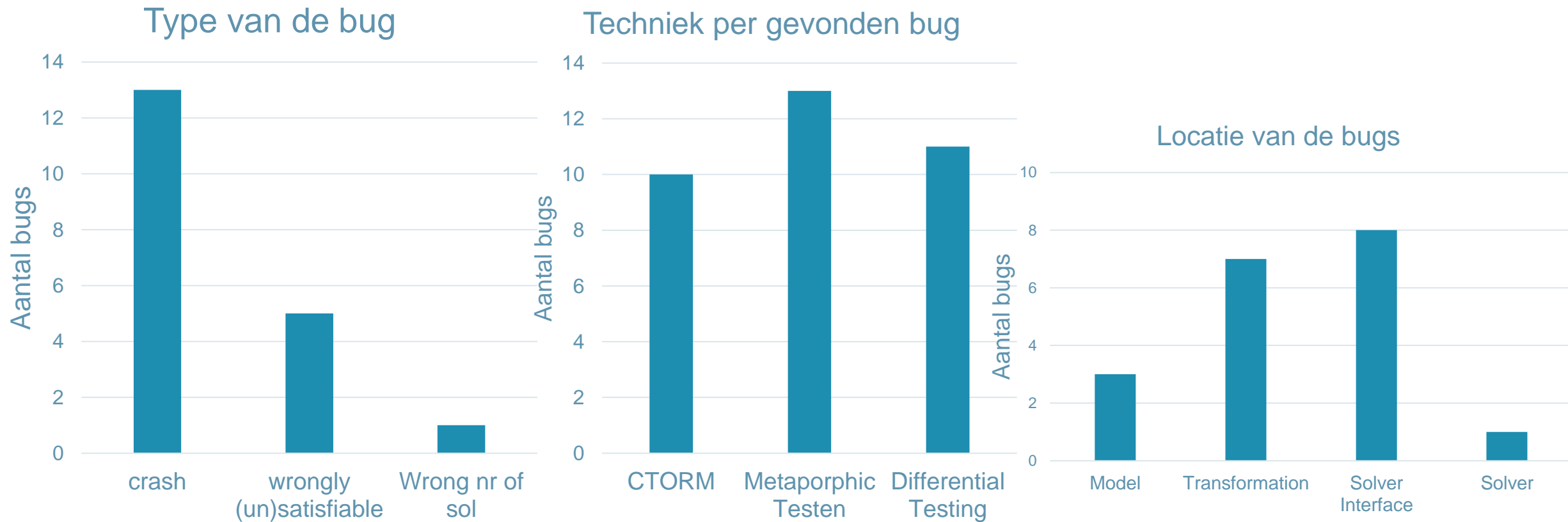
Resultaten

- 23 GitHub issues -> 19 bugs



Resultaten

- 23 GitHub issues -> 19 bugs



Besluiten

- Net niet automatisch
- Metamorphic testen meest flexibele
- Technieken zijn best combineerbaar

Verwezenlijkingen

- Vinden van (kritieke) bugs in constraint programming
- Ontwikkeling van STORM naar CTORM
- 19 bugs gevonden in CPMpy (waaronder kritieke bugs)

Toekomstige werk

- De frequente bugs
- Nieuwe code is nieuwe bugs
- Ook de hyperparameters van solvers testen[2]
 - Bv. bugs bij combinatie van constraints en zoek methodes

[2] Peisen Yao et al. “Fuzzing smt solvers via two-dimensional input space exploration”. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2021, pp. 322–335.

Vragen?