SPONSORED BY THE
Federal Ministry of Education and Research

Hochparallele Software-Verifikation

nebenläufiger Anwendungen in der Automobilindustrie

8. HPC-Status-Konferenz Erlangen
9. Oktober 2018

Marc Hartung (Zuse Institute Berlin)

ZIB

MES
MODEL ENGINEERING SOLUTIONS

SYMTA VISION

C A U
Christian-Albrechts-Universität zu Kiel

Motivation

Stromausfall in USA und Kanada 2003

- Betraf 50 Millionen Menschen bis zu 2 Tage
- Auslöser: Data-Race in einer Management-Software

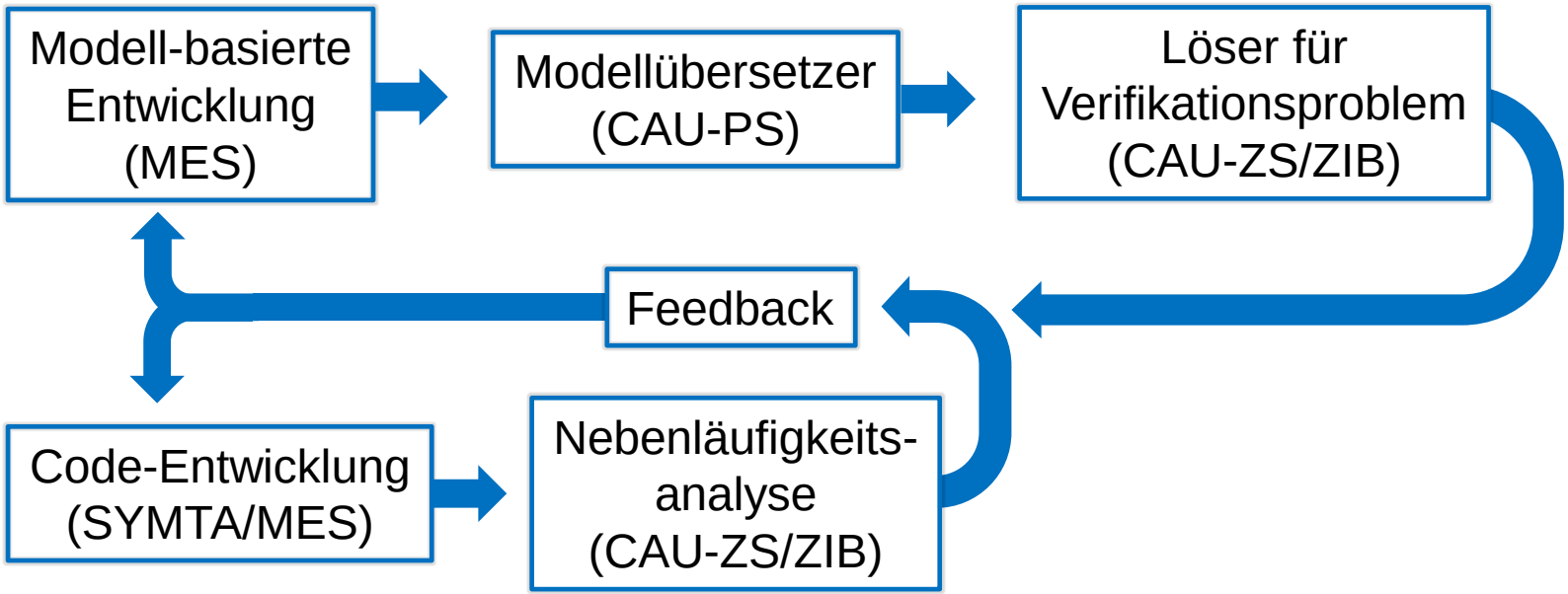
"We had in excess of three million online operational hours in which nothing had ever exercised that bug. I'm not sure that more testing would have revealed it."

General Electric Energy's Mike Unum

Ähnliche Herausforderung in der Automobilindustrie

HPSV-Project

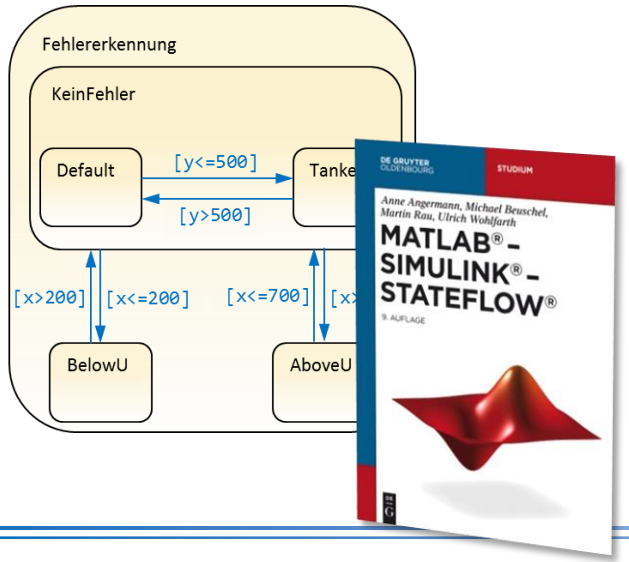
Toolchain für verbesserte Sotwarequalität



Modell-basierte Entwicklung (MES)

- Zustands- und Flussdiagramme als Programmrepräsentation
- Korrektheit definiert über Anforderungen

Modell



Anforderungen

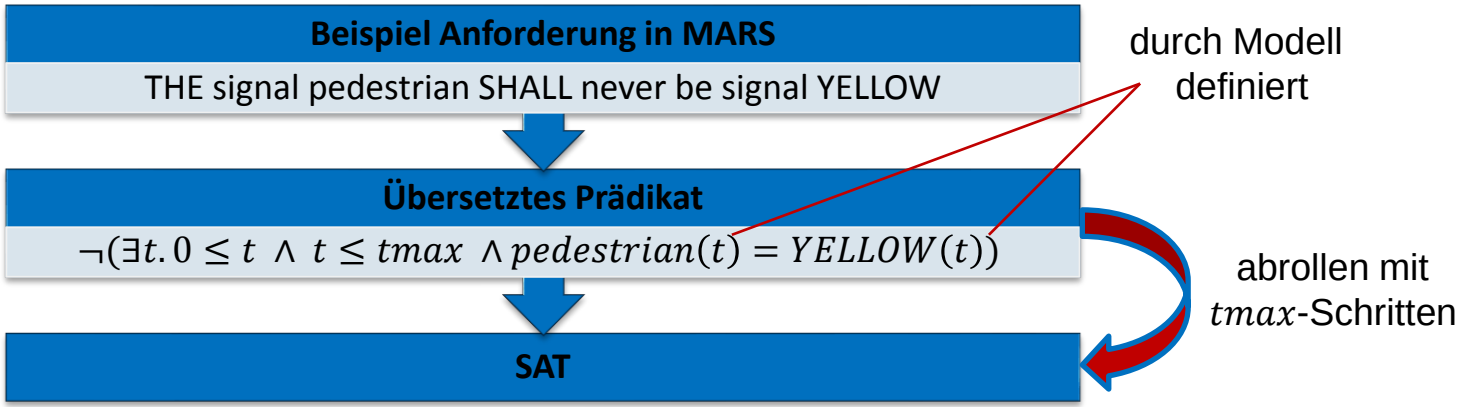
ID	Description	Testability	Type	Status
req_001			heading	
req_002	WHEN signal y becomes less than 200 THE signal Sensorfehler SHALL change to true immediately	Yes	functional	reviewed
req_003	WHEN signal y becomes greater than 700 THE signal Sensorfehler SHALL change to true immediately	Yes	functional	reviewed
req_004	WHILE signal Schwimmerschalter has been in between 200 and 700 for at least 1 time step THE signal Sensorfehler SHALL be false	Yes	functional	reviewed
req_005	THE signal Fuellstand	Yes		

Formale Beschreibung MARS v1.0

Modellübersetzung (CAU-PS)

Übersetzer für Simulink StateCharts und MARS

- Ausrollen des Modells und der Anforderungen (Bounded Model Checking)
- Übersetzung über Prädikatenlogik in Erfüllbarkeitsproblem (SAT)



Löser für Verifikationsproblem (CAU-ZS)

SAT-Solver TopoSat2

- Basiert auf Conflict-Driven-Clause-Learning
- Parallelisierung durch Portfolio-Ansatz und Klauselaustausch
- Im Rahmen von HPSV weiterentwickelt
- Verbesserung der
  - Suchheuristiken
  - Austauschstrategie
  - Klauselverwaltung
- Download: *https://github.com/the-kiel/TopoSAT2*

Effekt der Verbesserungen

Vergleich auf Basis der SAT Competition

- HLRN-Großprojekt „Hochparalleles SAT-Solving“
- 350 Probleme aus verschiedensten Bereichen
- Vergleichskriterium: gelöste Probleme (solved)

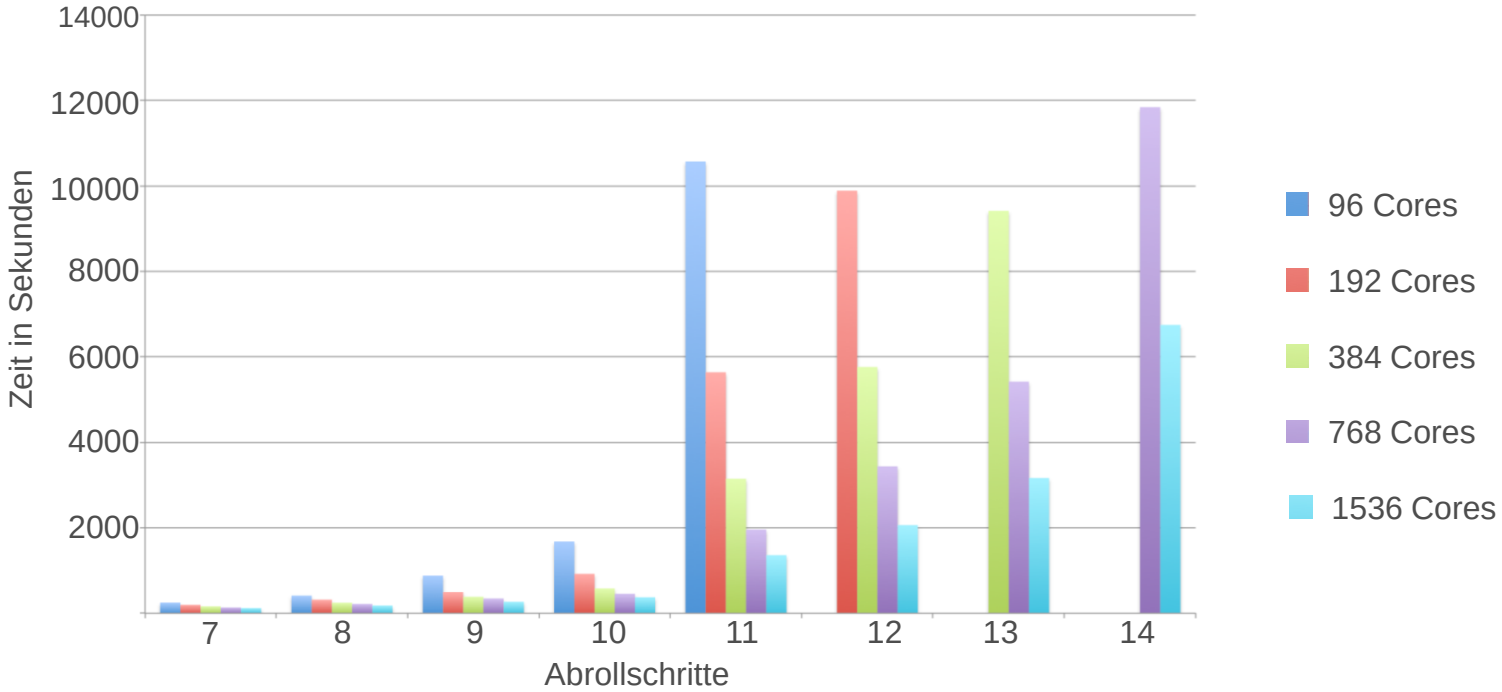
Cores	Baseline[1]		Improved[1]	
	speedup[2]	solved	speedup[2]	solved
96	49.6	217	80.6	264
192	48.3	225	102.8	273
384	93.7	226	142.6	278
768	160.3	230	-	-
1536	216.6	239	-	-

[1] *auf bis zu 64 Knoten (Konrad), Knoten: 2x Intel E5-2680v3 12 Cores*

[2] *Median Speedup*

Bounded Model Checking-Probleme

HPC erlaubt verifizieren komplexerer Modelle



Nebenläufigkeitsanalyse mit „Actul“ (ZIB)

Testet parallele C/C++-Anwendungen mit Hilfe von LLVM/Clang

```
char str[] = “hello world”;

int main() {
   printf(“%s\n”,str);
   return 0;
}
```

LLVM-IR code

```
@.str = internal constant [14 x i8] c"hello world"

define i32 @main() {
entry:
  %tmp1 = getelementptr [11 x i8]* @.str, i32 0, i32 0
  %tmp2 = call i32 (i8*, ...)* @printf( i8* %tmp1 )
  ret i32 0
}
```

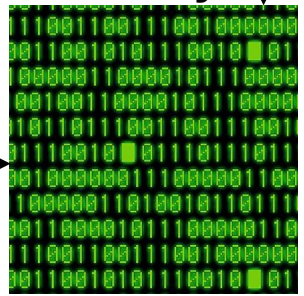
Instrumented LLVM-IR code

```
@.str = internal constant [11 x i8] c"hello world"

define i32 @main() {
entry:
  call funcEntry(%entry)
  call read(%tmp1)
  %tmp1 = getelementptr [14 x i8]* @.str, i32 0, i32 0
  call  write(%tmp2))
  %tmp2 = call i32 (i8*, ...)* @printf( i8* %tmp1 )
  call funcExit()
  ret i32 0
}
```

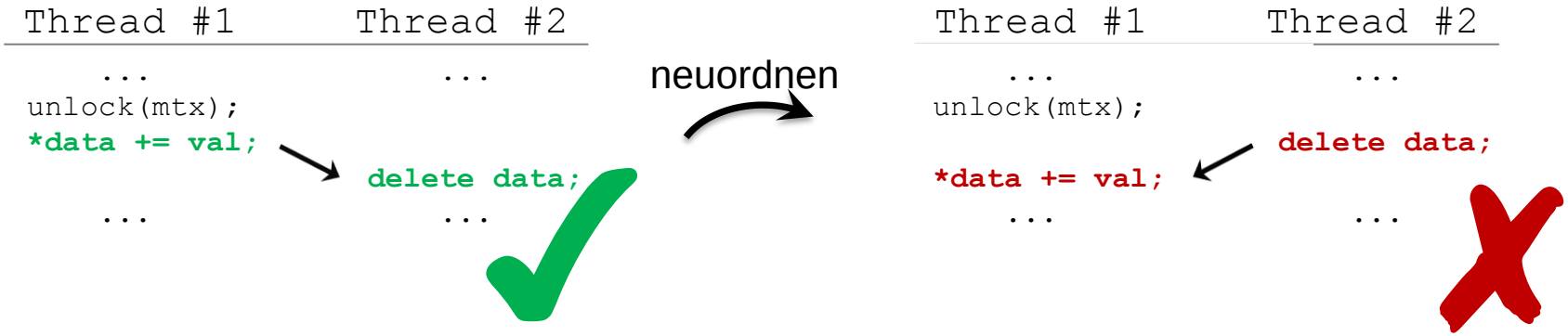
Actul RTL

Binary



Download: *https://github.com/marchartung/Actul*

Actul – Data-Race-Test-Ansatz



- Ausführen mehrerer Programmläufe (serialisiert)
- Neuordnen von Datas-Race-Zugriffen
- Erkennen von Programmabstürzen
- Identifizieren verantwortlicher (harmful) Data-Races
- Ausschließen falsch detektierter und unwichtiger (benign) Data-Races

Actul - Testauswahl

Mittels skalierbaren Algorithmus

- Initiale Ausführungen zur Detektion von Data-Races
- Vereinfachte Darstellung:

```
foreach data race subset D
          foreach complete order O of D
                    execute program with enforced O of D
```

- Beschränkung von **|D| <= N**
- Polynomieller Algorithmus $O(n^N)$ bei n Data-Races und $N \ll n$
- Skalierbar in Genauigkeit und paralleler Berechnung

Actul - Mehrwert

Verbesserte Genauigkeit

- Erkennt Abhängigkeiten zwischen Data Races (ungünstige Zugriffsketten)
- Skalierbare Testabdeckung
- Exploration ungewöhnlicher Thread-Verzweigungen

Hoch-parallelisierbar

- Nach Data-Race-Detektierung trivial parallel ausführbar
- HPC erlaubt ... genaueres testen (höhere **N**)

  ... testen längerer Programmausführungen

Experimentelle Resultate

Ausführliche Studie auf Basis SCTBench

- Offizielle Benchmark für Concurrency Testing
- Vergleich auf Basis verschiedener Kriterien
- (Übersteigt den gegebenen Zeitrahmen)

Tool	TSan_v2 Default			Actul Systematic N=2				Random				RaceFuzzer			
Test Case	#Races	Found	Time	H/B/F	Correct	Time	#Test Runs	H/B/F	correct	Time	#Test Runs	H/B/F	Correct	Time	#Test Runs
cb.aget-bug2	3	x	0.29	1/3/2	✓	1.77	216	0/0/2	x	0.22	216	2/2/0	x	0.79	7
cb.pbzip2-0.9.4	8	✓	1.05	1/0/33	✓	aborted	5,769	1/0/33	✓	aborted	5,769	1/0/33	✓	aborted	4,714
cs.din_phil6_sat	1	-	0.18	0/3/0	✓	12.95	10,838	0/3/0	✓	12.50	10,838	1/2/0	x	0.87	76
cs.micro_ok_2	1	x	1.03	0/320/0	✓	679.65	202,899	0/320/0	✓	885.65	202,899	0/320/0	✓	1. 66	321
cs.micro_bad_2	1	x	0.16	2/318/0	✓	708.05	202,909	0/320/0	x	845.71	202,909	41/279/0	x	1.92	331
cs.reorder_bad_4	3	x	0.16	0/5/0	x	1.087	381	2/3/0	x	1.059	381	0/5/0	x	0.75	16
cs.thread-pool	1	✓	2.16	0/1/0	✓	41.88	7,474	0/1/0	✓	8.22	7,474	0/1/0	✓	4.53	848
cs.token_ring_bad	6	✓	0.178	3/3/4	✓	8.83	55	3/3/4	✓	0.82	55	1/5/0	x	0.69	11
cs.wronglock_bad	1	x	0.163	2/3/0	✓	17.26	12,347	0/5/0	x	18.76	12,347	5/0/0	x	0.79	81
git.matrix	1	✓	0.163	1/0/0	✓	aborted	79,474	1/0/0	✓	aborted	79,474	1/0/0	✓	178. 65	3,096
git.object-pool	0	✓	0.03	0/5/17	✓	1.52	524	0/5/17	✓	2.39	524	0/5/17	✓	1.26	376
git.sudoku	1	x	0.17	1/0/13	✓	0.81	15	1/0/13	✓	0.77	15	1/0/13	✓	0.78	15
misc.stringbuffer	0	-	0.31	0/0/0	✓	0.01	1	0/0/0	✓	0.01	1	0/0/0	✓	0.01	1

Experimentelle Resultate

Kernerkenntnisse der Studie

	Korrekte Testfälle (max. 16)	Falsche Klassifikation von Data Races	Abbrüche
Actul $\mathrm{N} = 2$	12	1	4
Random Zufällige Thread-Ausführung	9	12	4
RaceFuzzer (entspricht Actul $\mathrm{N} = 1$)	7	47	2

Zusammenfassung

Verifikation

- ✓ Übersetzen gängiger Modelle (Simulink StateCharts)
- ✓ Vollständige Toolchain zur Verifikation
- ✓ HPC-SAT-Löser TopoSat2

Testen

- ✓ Freiverfügbares Tool Actul für C/C++-Programme
- ✓ Höhere Genauigkeit durch bessere Testabdeckung
- ✓ Test- und Analysephase hochskalierbar

Christian-Albrechts-Universität zu Kiel

ZIB

MES
MODEL ENGINEERING SOLUTIONS

SYMTA VISION

SPONSORED BY THE
Federal Ministry of Education and Research

Marc Hartung - hartung@zib.de

ZIB

MES
MODEL ENGINEERING SOLUTIONS

SYMTA VISION

C A U
Christian-Albrechts-Universität zu Kiel