Extending the Framework JavaSMT with the SMT Solver Yices2

LMU München
Lehr- und Forschungseinheit
Software and Systems Engineering

Michael Obermeier

13.05.2020

Motivation & Ziele

Motivation

Erweitern der in JavaSMT verfügbaren Solver

Ziele

- Integration des Solvers
- Testen der Funktionalität
- Dokumentieren des Vorgehens
- Evaluation

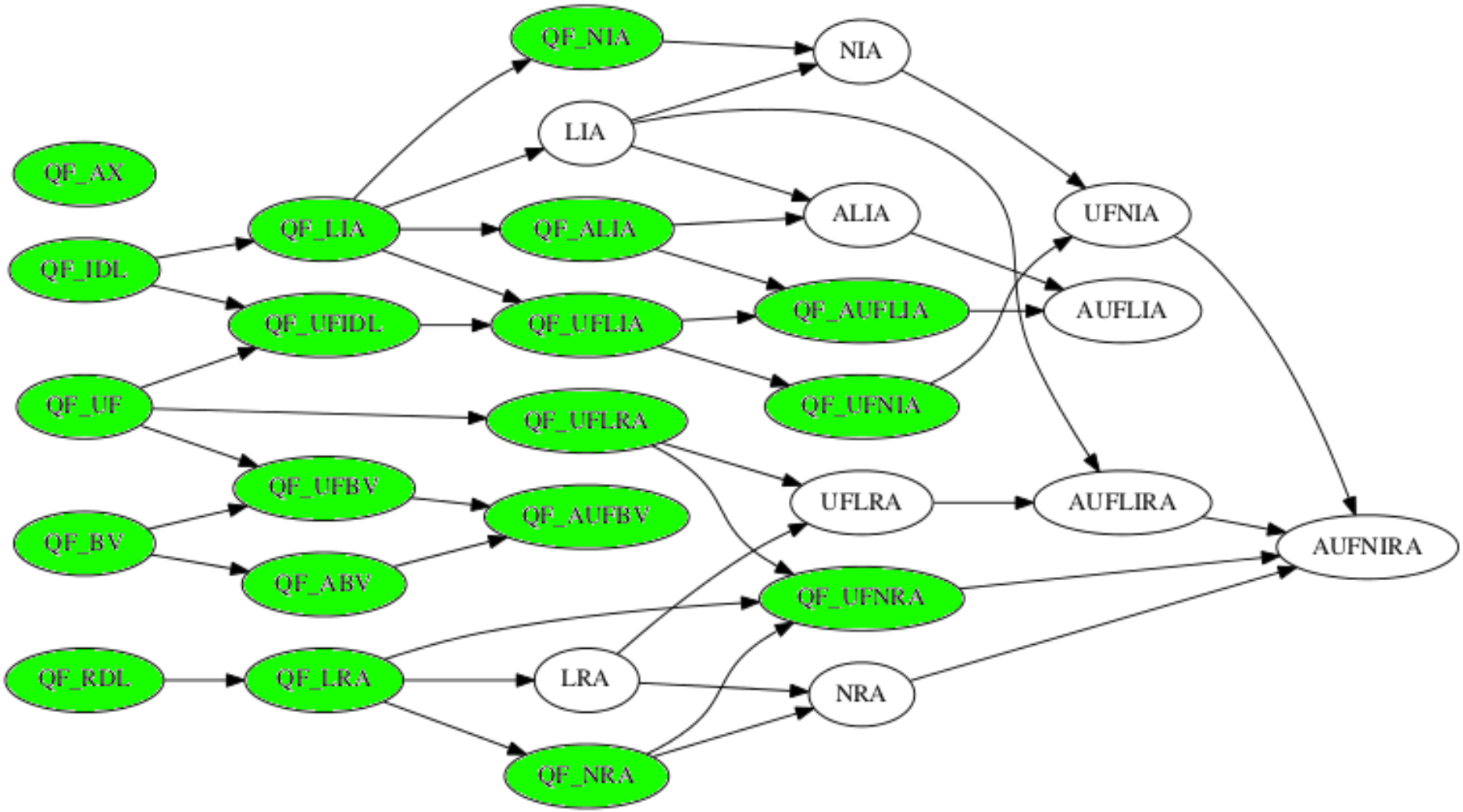
Einführung: SMT

Satisfiability Modulo Theories (SMT)

- Erweiterung des Erfüllbarkeitsproblems der Aussagenlogik (SAT)
- Erweitert SAT um Bitvector, Integer, Real... Theorien
- Sowohl SAT als auch SMT sind NP-vollständig

Einführung: SMT-LIB2

- Gemeinsamer Standard für Ein-/Ausgabe von SMT-Solvern
- Definition von Theorien und Logiken
- Übersicht definierter Logiken (Grün → in Yices2 verfügbar)



Einführung: JavaSMT

Gemeinsame API für SMT-Solver

Ziele

- Minimaler Overhead
- Typsicherheit
- Nutzbarkeit zusätzlicher Fähigkeiten

	Boolector	CVC4	MathSAT5	Princess	SMTInterpol	Z3	**Yices2**
AllSAT	✗	✗	✓	✓	✓	✓	✗
AssumptionSolving	✓	✗	✓	✗	✗	✓	✓
Interpolation	✗	✗	✓	✓	✓	✓	✗
Optimization	✗	✗	✓	✗	✗	✓	✗
UnsatCore	✗	✓	✓	✓	✓	✓	✓
UnsatCore with Assumptions	✗	✗	✓	✗	✗	✓	✓

Einführung: JavaSMT

Gemeinsame API für SMT-Solver

Ziele

- Minimaler Overhead
- Typsicherheit
- Nutzbarkeit zusätzlicher Fähigkeiten
- Verfügbare Solver in verschiedenen Frameworks:

	Boolector	Z3	MathSAT5	CVC4	Princess	SMTInterpol	**Yices2**	PicoSAT	SWORD
JavaSMT	✓	✓	✓	✓	✓	✓	✓	✗	✗
ScalaSMT	✗	✓	✓	✓	✗	✓	✓	✗	✗
MetaSMT	✓	✓	✗	✓	✗	✗	✗	✓	✓
PySMT	✓	✓	✓	✓	✗	✗	✓	✓	✗

Weshalb mehr Solver?

- Unterschiedliche Features
- Unterschiede in Theorie-Unterstützung (in JavaSMT)

	Boolector	CVC4	MathSAT5	Princess	SMTInterpol	Z3	**Yices2**
Bool	✓	✓	✓	✓	✓	✓	✓
BV	✓	✓	✓	✓	✗	✓	✓
Int	✗	✓	✓	✓	✓	✓	✓
Real	✗	✓	✓	✗	✓	✓	✓
Float	✗	✓	✓	✗	✗	✓	✗
UF	✓	✓	✓	✓	✓	✓	✓
Array	✓	✓	✓	✓	✓	✓	✗
Quantifier	✓	✗	✗	✓	✗	✓	✗

Weshalb mehr Solver?

- Unterschiedliche Features
- Unterschiede in Theorie-Unterstützung (in JavaSMT)
- Unterschiedliche Leistungsfähigkeit

Yices2

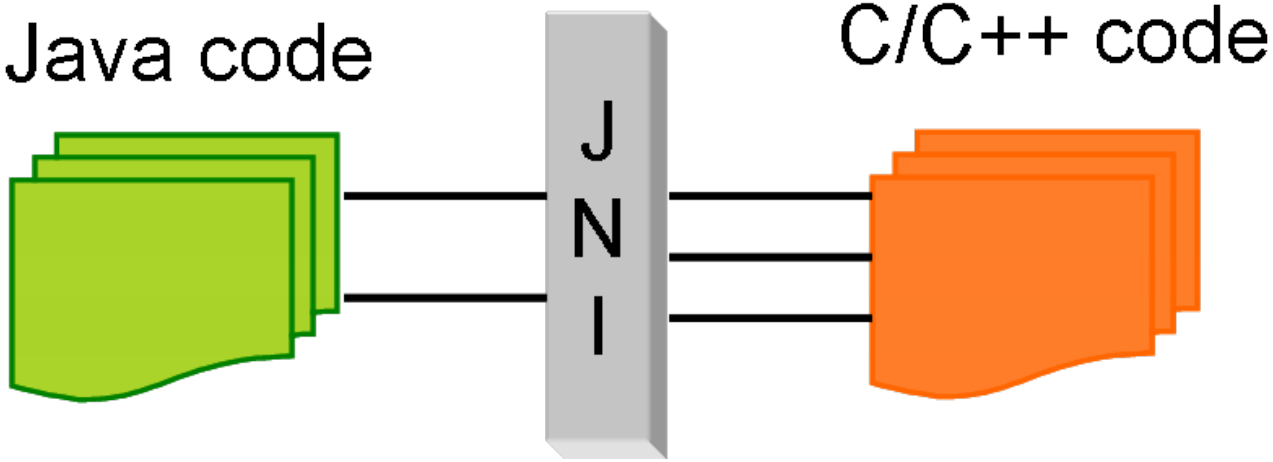
- Geschrieben in C
- Python, Go, OCaml Bindings verfügbar
- Weiterentwicklung von Yices 1 mit Fokus auf:
  - Bessere API
  - Bessere Leistung v.a. für Bitvector
- Zwei Solver Ansätze:
  - DPLL(T)
  - MCSAT
- Unterstützt Model-Generierung
- GPLv3 Lizenz

Yices2

- Geschrieben in C
- Python, Go, OCaml Bindings verfügbar
- Weiterentwicklung von Yices 1 mit Fokus auf:
  - Bessere API
  - Bessere Leistung v.a. für Bitvector
- Zwei Solver Ansätze:
  - DPLL(T)
  - MCSAT
- Unterstützt Model-Generierung
- GPLv3 Lizenz $\not\downarrow$ Apache2.0 Lizenz von JavaSMT

JNI-Wrapper

- Handgeschriebener JNI-Wrapper
- Verwendet C Makros für bessere Lesbarkeit
- Überträgt Daten Struct $\leftrightarrow$ Java Objekt wo nötig



Yices 2 API Übersicht

JavaSMT muss nur 3 Pointer kennen:

- Konfiguration
- Kontext
- Model

Yices 2 API Übersicht

Restliche Interaktion mit Yices2 API größtenteils über:

- type_t → positiver Integer
- term_t → positiver Integer
- term_constructor_t → positiver Integer
- yval_t → Struct aus 2 positiven Integer Werten

Gelöste Probleme

- Kein UnsatCore bei inkrementellem Lösen
- Stack kann ohne Lösen UNSAT werden
- Kollision zwischen term_constructor und UF beim Besuchen

Gelöste Probleme

- Kein UnsatCore bei inkrementellem Lösen → interner Stack
- Stack kann ohne Lösen UNSAT werden
- Kollision zwischen term_constructor und UF beim Besuchen

Gelöste Probleme

- Kein UnsatCore bei inkrementellem Lösen $\rightarrow$ interner Stack
- Stack kann ohne Lösen UNSAT werden $\rightarrow$ interner Stack
- Kollision zwischen term_constructor und UF beim Besuchen

Gelöste Probleme

- Kein UnsatCore bei inkrementellem Lösen → interner Stack
- Stack kann ohne Lösen UNSAT werden → interner Stack
- Kollision zwischen term_constructor und UF beim Besuchen
  → term_constructor in negativen Integer-Bereich verschieben

Ungelöste Probleme

- Bv2int/int2bv werden nicht unterstützt
- Quantifizierte Formeln brauchen spezielles Symbol
- Umsetzung von Array
- Parsen von SMT-LIB2

Evaluation mit CPAchecker und BenchExec

- Voraussetzungen:
  - Bounded Model Checking (BMC) mit k=1 und k=10
  - 5796 Tasks aus SV-Benchmarks
  - Ausführung auf Apollon Cluster der VerifierCloud
  - Limits für jeden Task: 2 Cores, 900s, 15GB RAM
- Evaluierte Solver:
  - Boolector
  - CVC4
  - MathSAT5
  - Yices2
  - Z3

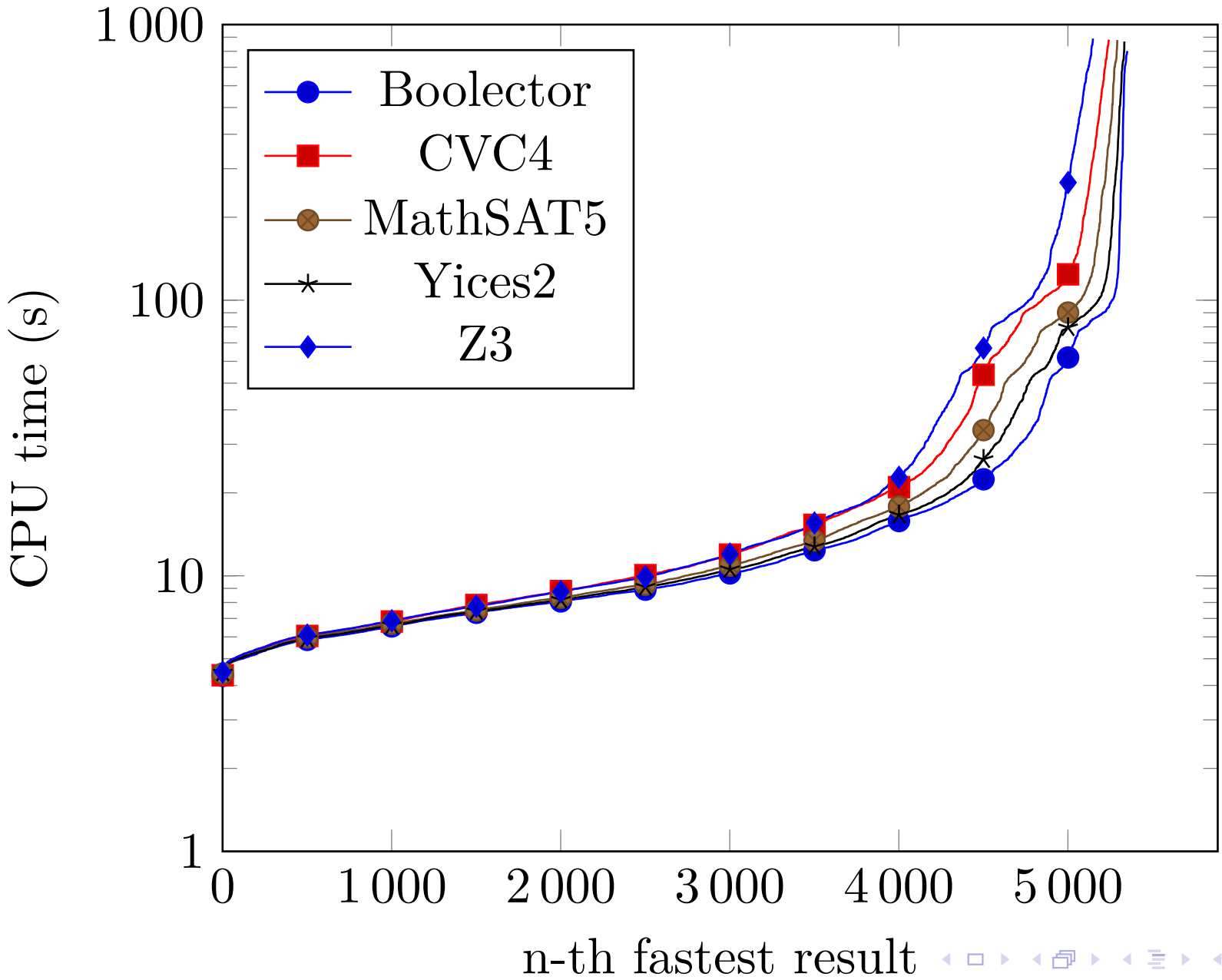
Evaluation mit CPAchecker und BenchExec

- Verwendete Optionen:
  - useArraysForHeap=false für alle Solver
  - encodeFloatAs=RATIONAL für Yices2
  - Benötigte Optionen für Boolector
- Auswertung:
  - Aufgrund der Besonderheiten von BMC wurden alle Ergebnisse außer Zeit/Speicherüberschreitung und sonstige Fehler berücksichtigt.

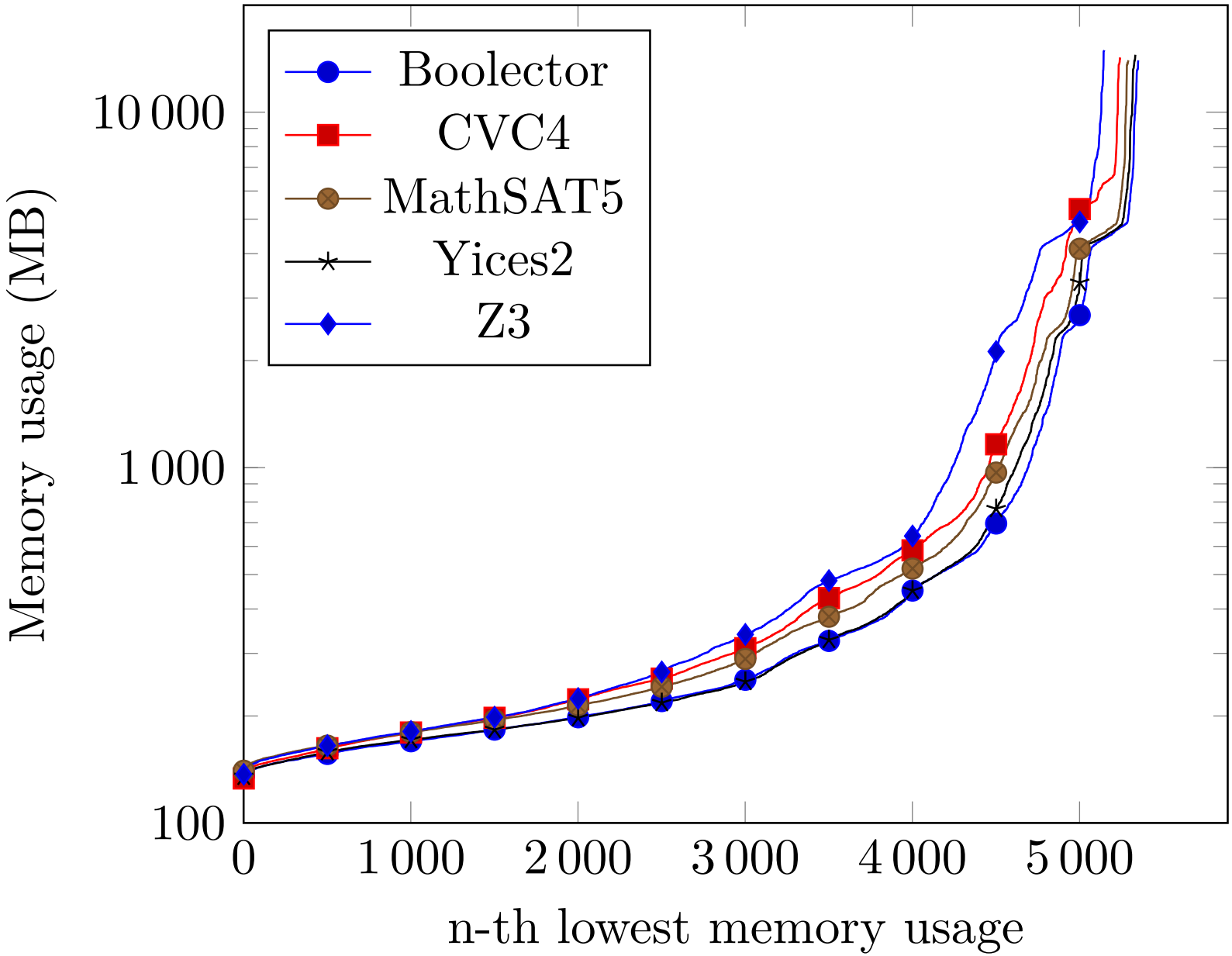
Übersicht Resultate

k	Solver	Korrekt			Inkorrekt			Unbekannt	Fehler
		Gesamt	Wahr	Falsch	Gesamt	Wahr	Falsch		
1	Boolector	737	416	321	74	0	74	4540	445
	CVC4	752	475	277	1	0	1	4490	553
	MathSAT5	773	447	296	1	0	1	4518	504
	Yices2	775	477	298	1	0	1	4558	462
	Z3	761	475	286	1	0	1	4386	648
10	Boolector	1590	716	874	248	120	128	1746	2212
	CVC4	1317	627	690	3	0	3	1406	3070
	MathSAT5	1542	650	892	3	0	3	1824	2427
	Yices2	1599	646	953	4	0	4	1886	2307
	Z3	1339	589	750	3	0	3	1374	3080

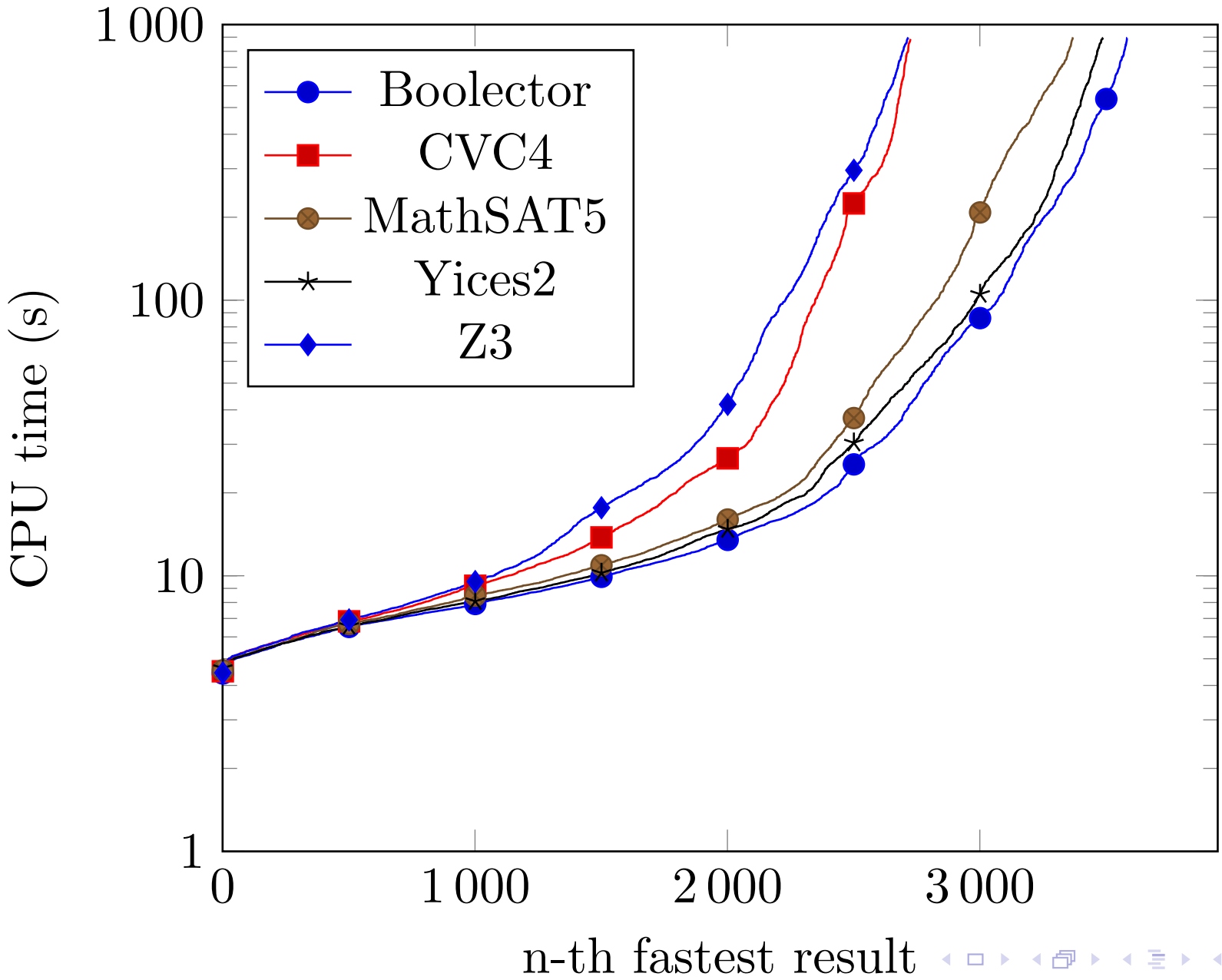
CPU-Zeit BMC mit k=1



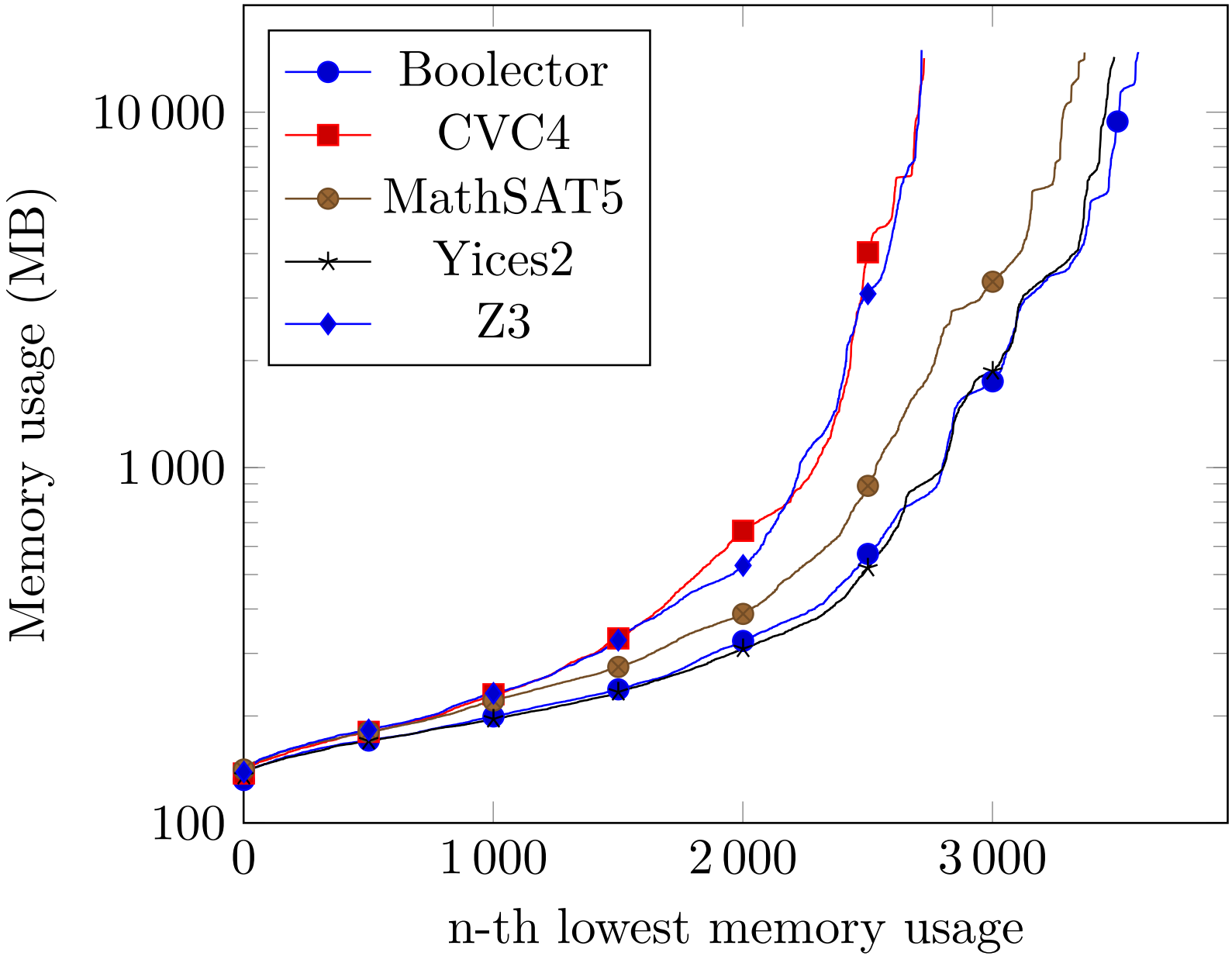
Arbeitsspeicherverbrauch BMC mit k=1



CPU-Zeit BMC mit k=10



Arbeitsspeicherverbrauch BMC mit k=10



Zusammenfassung

Gut:

- Performanz vergleichbar zu Boolector trotz mehr Features
- Durchdachte und gut dokumentierte API vereinfacht Implementierung
- Alle für JavaSMT notwendigen Funktionen umsetzbar, aber...

Problematisch:

- Array/ Quantified Theorien konnten nicht umgesetzt werden
- Yices2 kann SMT-LIB2 parsen, API nur eigene Eingabesprache
- Bv2int/int2bv fehlt derzeit, trotz Unterstützung beider Theorien

Zukünftige Arbeiten

JavaSMT:

- GPL-konforme Unterstützung von Yices2
- Unterstützung spezieller Symbole

Yices2:

- Implementieren fehlender Features (Bv2int/int2bv, SMT-LIB2 parsen)
- Future Work: Neue Features wie Interpolation oder Optimization

Quellen

- Yices2: `https://yices.csl.sri.com/index.html`
- JavaSMT: `https://github.com/sosy-lab/java-smt`
- CPAchecker: `https://github.com/sosy-lab/cpachecker`
- BenchExec: `https://github.com/sosy-lab/benchexec`