Report on the 2022 edition

https://chc-comp.github.io/

Emanuele De Angelis, Inst. for Systems Analysis and Computer Science - National Research Council, Italy

Hari Govind V K, University of Waterloo, Canada

Outline

- Tracks
- Benchmarks
- Teams & Solvers
- Results
- Discussion

Tracks

- Linear Integer Arithmetic, **Lin**ear clauses (**LIA-Lin**)
- **LIA**, **Nonlin**ear clauses (**LIA-Nonlin**)
- **LIA** and **Arrays**, **Lin**ear clauses (**LIA-Lin-Array**)
- **LIA** and **Arrays**, **Nonlin**ear clauses (**LIA-Nonlin-Array**)
- Linear Real Arithmetic, Transition Systems (**LRA-TS**)
- **LRA-TS-par**allel, Transition Systems (**LRA-TS-par**)
- Algebraic Data-Types, **Nonlin**ear clauses (**ADT-Nonlin**)
- **LIA**, **Arrays** and **non-rec**ursive ADT, **Nonlin**ear clauses (**LIA-Nonlin-Arrays-nonrecADT**)

New in 2022

Benchmarks
Inventory & Selection process

New Benchmarks

- **Solidity CHC Benchmarks** (Thanks to *Leonardo Alt*)
  Nonlinear clauses; LIA + Arrays + non-recursive ADTs
  Source: Solidity SMTChecker, Eth2 Deposit Contract, and OpenZeppelin's ERC777 implementation
  https://github.com/leonardoalt/chc_benchmarks_solidity
- **TriCera** (Thanks to *Zafer Esen* & *Philipp Rümmer*)
  Nonlinear clauses; LIA + Arrays + non-recursive ADTs
  Source: TriCera model checker (for C), https://github.com/zafer-esen/tricera-adt-arr
- **RInGen** (Thanks to *Yurii Kostyukov*)
  Nonlinear clauses; ADTs only
  Source: TIP/Isaplanner (CHC-COMP 2021 updated benchmarks)
  https://github.com/chc-comp/ringen-adt-benchmarks
- **Ultimate** (Thanks to *Matthias Heizmann* & *Frank Schüssele*)
  Nonlinear clauses; LIA + Arrays
  Source: Ultimate framework, https://github.com/schuessf/chc-comp-benchmarks
- **Golem** (Thanks to *Martin Blicha*)
  Linear clauses; LIA
  Source: unsafe version of AE-VAL benchmarks, https://github.com/blishko/chc-benchmarks

Benchmark processing

Repositories: https://github.com/chc-comp, https://www.starexec.org/starexec/secure/explore/spaces.jsp?id=73700

- LIA-{lin,nonlin}, LIA-{lin,nonlin}-Arrays, LRA-TS (same as in CHC-COMP 2021)
  a. Formatted (according to the CHC-COMP rules) using format.py (https://github.com/chc-comp/chc-tools)
  b. Categorized using check[-TRACK] (https://github.com/chc-comp/chc-tools)
  c. Removed duplicated benchmarks
- ADT-nonlin (same as in CHC-COMP 2021)
  a. Purified by using the RInGEN transformation: elimination of SMT theories and recursively-defined functions (e.g. Int -> Nat + SMT rec. functions)
- LIA-nonlin-Arrays-nonrecADT
  a. Removed duplicated benchmarks

Benchmarks inventory

(total/unique #benchmarks)

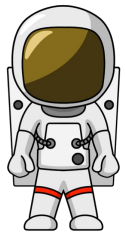
Repository	LIA-nonlin	LIA-lin	LIA-nonlin-Arrays	LIA-lin-Arrays	LRA-TS	ADT-nonlin	LIA-nonlin-Arrays-nonrecADT
adt-purified						67/67	
aeval		54/54					
aeval-unsafe		54/54					
chc-comp19				290/290	228/226		
eldarica-misc	69/66	149/136					
extra-small-lia		55/55					
hcai	133/131	101/87	25/25	39/39			
hopv	68/67	49/48					
jayhorn	7325/7224	75/73					
kind2	851/737						
ldv-ant-med			79/79	10/10			
ldv-arrays			822/546	3/2			
llreve	43/42	44/44		31/31			
quic3				43/43			
ringen						454/440	
sally					177/174		
seahorn	68/66	3379/2812					
solidity							3571/3548
sv-comp	1643/1169	3150/2930	856/780	79/73			
synth/nay-horn	119/114						
synth/semgus			5371/4839				
tricera	4/4	405/405					
tricera/adt-arrays							156/156
ultimate	8/8		21/21				
vmt		906/803			99/98		
total	10223/**9628**	8421/**7501**	7174/**6290**	495/**488**	504/**498**	521/**507**	3727/**3704**

Benchmark selection

The “hardness” of the benchmarks is determined by using the results of the **two top solvers** from **2021**:

- **A** - rated benchmarks: *both* solvers can *solve it*
- **B** - rated benchmarks: *one* solver can *solve it*
- **C** - rated benchmarks: *both* solvers *time out*

(approach similar to that of CHC-COMP 2021)



(5s timeout)

ELDARICA

(10s timeout)

Selection process applied to tracks where we have:

- too many benchmarks, or
- a few repos with a significant difference in the number of benchmarks

(LIA-lin, LIA-nonlin, LIA-nonlin-Arrays, ADT-nonlin, LIA-nonlin-Arrays-nonrecADT)

Benchmark selection

- Run the two top solvers to get the ratings **A**, **B**, and **C**. This yields four classes:
  - **A**-rated benchmarks
  - **B**-rated benchmarks (solved by **Spacer** only)
  - **B**-rated benchmarks (solved by **Eldarica** only)
  - **C**-rated benchmarks
- For each *benchmark repository*, we
  - choose a number **Nr** of benchmarks to be selected
  - **randomly** select:
    - ✓ up to **0.2**x**Nr** **A**-rated benchmarks
    - ✓ up to **0.4**x**Nr** **B**-rated benchmarks **equally distributed between Spacer and Eldarica**
    - ✓ up to **0.4**x**Nr** **C**-rated benchmarks

    (If any repo contains fewer benchmarks than required, take the rest from the next higher rating class)

Hardness statistics

Repository	LIA-nonlin	LIA-lin	LIA-nonlin-Arrays	ADT-nonlin	LIA-nonlin-Arrays-nonrecADT
adt-purified				5/32/1/29	
aeval		11/9/2/32			
aeval-unsafe		11/5/0/38			
eldarica-misc	9/26/1/30	84/39/2/11			
extra-small-lia		13/22/3/17			
hcai	71/41/0/19	77/5/0/5	12/6/1/6		
hopv	46/14/7/0	47/1/0/0			
jayhorn	1870/3441/1/1912	73/0/0/0			
kind2	54/660/0/23				
ldv-ant-med			0/15/0/64		
ldv-arrays			0/112/0/434		
llreve	10/20/1/11	34/4/2/4			
ringen				11/17/3/409	
seahorn	28/25/0/13	678/1306/1/827			
solidity					1033/1849/68/598
sv-comp	309/766/5/89	2361/475/1/93	245/254/1/280		
synth/nay-horn	25/45/0/44				
synth/semgus			136/2386/0/2317		
tricera/svcomp20	4/0/0/0	15/27/1/362			
tricera/adt-arrays					2/29/0/125
ultimate	0/0/8/8		0/0/0/21		
vmt		26/680/0/97			
total	2426/5038/15/2164	3430/2573/12/1486	393/2773/2/3122	16/49/4/438	1035/1878/68/791

Competition benchmarks

Repository	LIA-nonlin	LIA-lin	LIA-nonlin-Arrays	ADT-nonlin	LIA-nonlin-Arrays-nonrecADT
adt-purified				67/52	
aeval		30/30			
aeval-unsafe		30/30			
eldarica-misc	30/30	45/31			
extra-small-lia		30/30			
hcai	60/43	45/19	15/13		
hopv	30/18	30/7			
jayhorn	90/90	30/6			
kind2	90/59				
ldv-ant-med			60/60		
ldv-arrays			90/90		
llreve	45/30	30/16			
ringen				134/131	
seahorn	45/31	90/90			
solidity					468/310
sv-comp	90/90	90/90	135/135		
synth/nay-horn	60/60				
synth/semgus			135/135		
tricera/svcomp20	3/0	60/60			
tricera/adt-arrays					156/155
ultimate	6/5		15/15		
vmt		90/90			
total	549/**456**	600/**499**	450/**448**	201/**183**	624/**465**

Solvers (5 competing + 1 hors concours) & Tracks (7)

	LIA-nonlin	LIA-lin	LIA-nonlin-Arrays	LIA-lin-Arrays	LRA-TS LRA-TS-parallel	ADT-nonlin	LIA-nonlin-Arrays-nonrecATD
Eldarica	Yes	Yes	Yes	Yes	No	Yes	Yes
Golem	Yes	Yes	No	No	Yes	No	No
RInGEN	No	No	No	No	No	Yes	No
Ultimate TreeAutomizer	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ultimate Unihorn	Yes	Yes	Yes	Yes	Yes	No	No
Spacer (Hors Concours)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Competition Runs (same as CHC-COMP 2021)

- LIA-{lin,nonlin}, LIA-{lin,nonlin}-Arrays, LRA-TS, ADT-nonlin, LIA-nonlin-Arrays-nonrecADT
  - Timeout 1800s, CPU time
  - Memory limit: 64GB
  - Two jobs per StarExec node, two cores for each job
- LRA-TS-parallel
  - Timeout 1800s, wall-clock time
  - Memory limit: 64GB
  - One job per StarExec node, four cores for each job

Competing Solvers

Eldarica Overview

- Horn solver developed since 2011
- Open-source, implemented in Scala, running in JVM

- **Input formats:**
  SMT-LIB, Prolog, C, timed automata
- **Theories:**
  LIA, NIA, arrays, heap, algebraic data-types, bit-vectors

- **New in 2022 (v2.0.8):**
  Fixes & improvements in array and heap solver; different portfolio

- Scala/Java API
- Support for linear + non-linear clauses

- `https://github.com/uuverifiers/eldarica`

Golem CHC solver

- Active development at USI Formal Verification and Security Lab
- Builds on top of the interpolating SMT solver OpenSMT
- Supports linear real and integer arithmetic
- Three engines
  - Impact [McMillan '06] (Lazy Abstraction with Interpolants)
  - Spacer [Komuravelli et al. '16]
  - TPA [Blicha et al. '22] (Transition Power Abstraction)
- https://github.com/usi-verification-and-security/golem
- Competition tracks and configurations
  - LRA-TS - portfolio
  - LIA-Lin - portfolio
  - LIA-Nonlin - only Spacer engine

RInGen: *R*egular *In*variant *Gen*erator

- RInGen solves (nonlinear) CHCs over ADTs
- RInGen supports many backend solvers:
  - Z3, Eldarica, cvc5, VeriMAP, Vampire
- CHC-COMP'22 submission uses Vampire as a backend
- Source code: `https://github.com/Columpio/RInGen`
- Developed by Yurii Kostyukov (JetBrains Research)

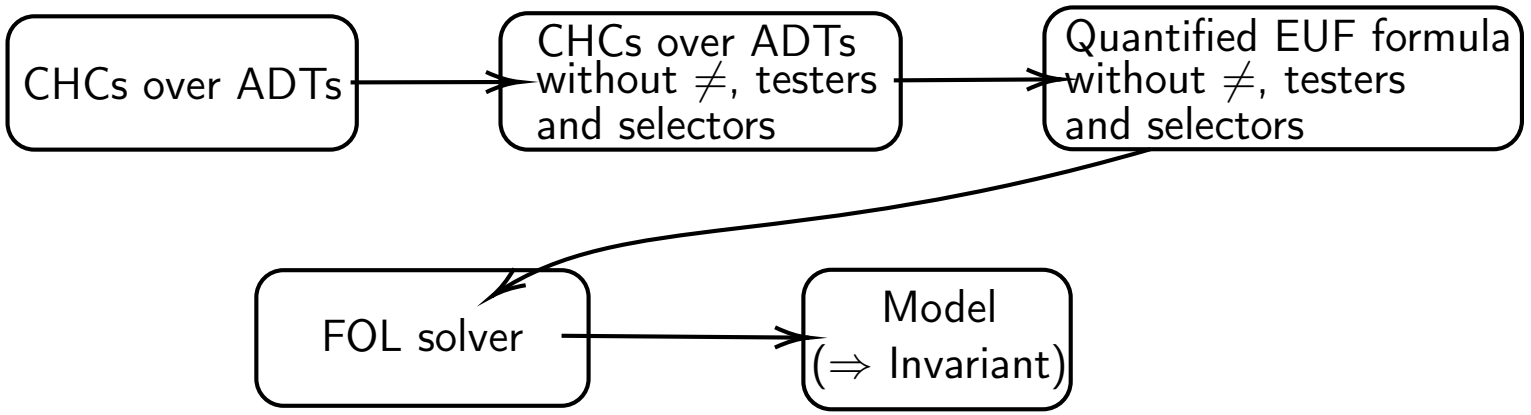


Figure: RInGen workflow

Competition Results

LIA-lin

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Spacer	499	338	235	103	161	155	3	299420	149835	1950	36
Golem	499	309	215	94	190	190	0	374736	142604	549	25
Eldarica	499	307	219	88	192	192	0	372231	134933	7722	38
Ultimate Unihorn	499	169	107	62	330	290	0	551859	466284	18581	0
Ultimate TreeAutomizer	499	139	81	58	360	343	0	633917	605367	18178	0

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

LIA-nonlin

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Spacer	456	421	286	135	35	27	0	75414	39303	425	40
Golem	456	365	240	125	91	91	0	196890	196913	148	2
Eldarica	456	358	229	129	98	98	0	215589	76099	7051	7
Ultimate Unihorn	456	204	123	81	252	243	0	485808	391416	17094	1
Ultimate TreeAutomizer	456	50	13	37	406	379	0	691499	648778	16621	0

```
cnt   number of benchmarks          mo    number of memory outs
ok    number of solved benchmarks   time  sum of total cpu time in seconds
sat   number of SAT solved          real  sum of total wall-clock time in seconds
uns   number of UNSAT solved        space sum of memory used in MB
fld   number of failed unsolved     uniq  number of unique benchmarks solved
to    number of timeouts
```

LIA-lin-Arrays

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Spacer	488	288	213	75	200	196	0	355686	178328	546	85
Eldarica	488	220	149	71	268	264	0	481857	163636	7564	10
Ultimate Unihorn	488	204	137	67	284	257	0	479998	393423	18193	2
Ultimate TreeAutomizer	488	170	113	57	318	266	0	491989	470080	17777	0

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

LIA-nonlin-Arrays

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Spacer	448	342	197	145	106	86	0	180474	95392	674	115
Eldarica	448	215	129	86	233	233	0	449249	177427	6929	10
Ultimate Unihorn	448	168	88	80	280	175	0	368162	297747	16584	2
Ultimate TreeAutomizer	448	89	19	70	359	337	0	618917	488550	16481	5

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

LRA-TS

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Spacer	498	317	234	83	181	169	0	355136	181996	653	37
Golem	498	311	235	76	187	187	0	364678	121607	926	19
Ultimate TreeAutomizer	498	155	114	41	343	343	0	646968	619460	18134	4
Ultimate Unihorn	498	103	65	38	395	395	0	725651	613022	18446	1

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

LRA-TS-parallel

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Spacer	498	341	256	85	157	145	0	627685	319147	887	35
Golem	498	333	256	77	165	165	0	988621	329628	1694	22
Ultimate TreeAutomizer	498	155	114	41	343	343	0	671801	641980	18134	3
Ultimate Unihorn	498	103	65	38	395	384	0	1072345	719996	18535	1

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

ADT-nonlin

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
RInGen	183	92	50	42	91	91	0	165240	163756	1010	27
Eldarica	183	60	29	31	123	122	0	223335	73011	2856	4
Spacer	183	55	25	30	128	125	0	226205	226257	246	5
Ultimate TreeAutomizer	183	0	0	0	183	0	0	1881	745	6673	0

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

LIA-nonlin-Arrays-nonrecADT

solver	cnt	ok	sat	uns	fld	to	mo	time	real	space	uniq
Eldarica	465	395	242	153	70	58	0	125994	46440	7108	103
Spacer	465	298	179	119	167	66	0	131079	131124	185	6
Ultimate TreeAutomizer	465	0	0	0	465	0	0	4825	1894	16958	0

```
cnt    number of benchmarks
ok     number of solved benchmarks
sat    number of SAT solved
uns    number of UNSAT solved
fld    number of failed unsolved
to     number of timeouts
```

```
mo     number of memory outs
time   sum of total cpu time in seconds
real   sum of total wall-clock time in seconds
space  sum of memory used in MB
uniq   number of unique benchmarks solved
```

Results

	LIA-lin	LIA-nonlin	LIA-Nonlin-Arrays	LIA-Lin-Arrays	LRA-TS	LRA-TS-parallel	ADT-Nonlin	LIA-Nonlin-Arrays-nonrecADT
1st	Golem	Golem	Eldarica	Eldarica	Golem	Golem	RInGen	Eldarica
2nd	Eldarica	Eldarica	Ultimate Unihorn	Ultimate Unihorn	Ultimate TreeAutomizer	Ultimate TreeAutomizer	Eldarica	
3rd	Ultimate Unihorn	Ultimate Unihorn	Ultimate TreeAutomizer	Ultimate TreeAutomizer	Ultimate TreeAutomizer	Ultimate Unihorn		

Spacer for many unofficial 1st places

Big Thanks to

Discussion

- **Results validation**:
  - models/counterexamples ?
  - set-info :status ?
- **Tracks**:
  - Discontinue LRA-TS? Move to LRA-lin track or a more general LRA track ?
  - A more general LIA-nonlin-Arrays-ADT track?
    - Should we continue the new track?
- **CHC-COMP 2023 Organisers**?