CPA'16 / 6th LDV Workshop, Passau, Germany

LinuxTesting.org

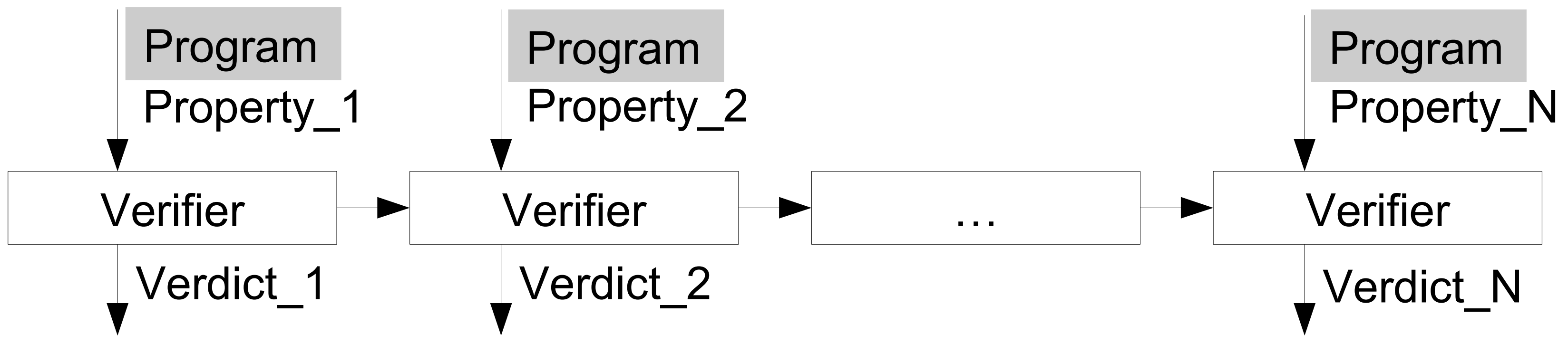
Further Development of Methods for Verifying Several Properties at once

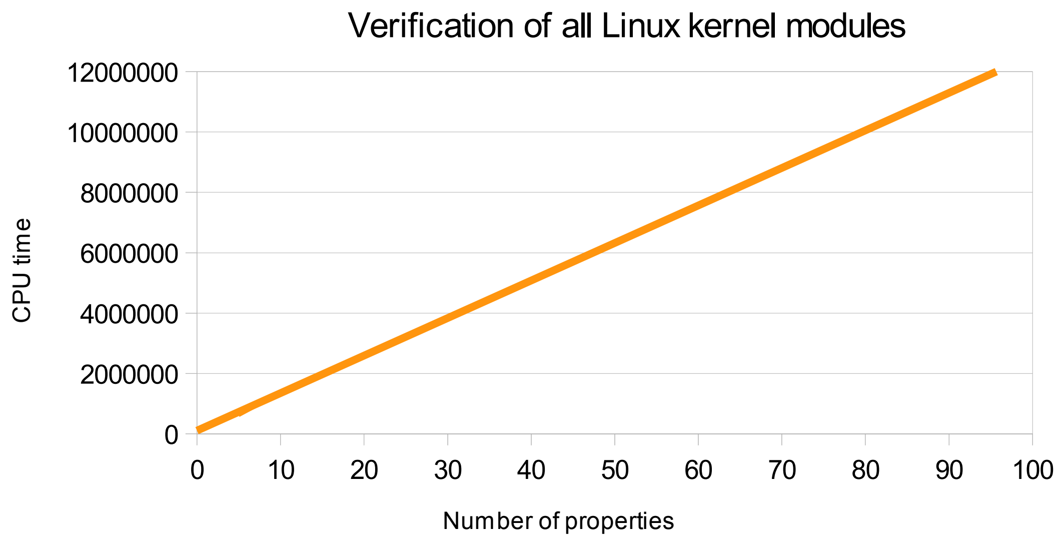
Vitaly Mordan
mordan@ispras.ru

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

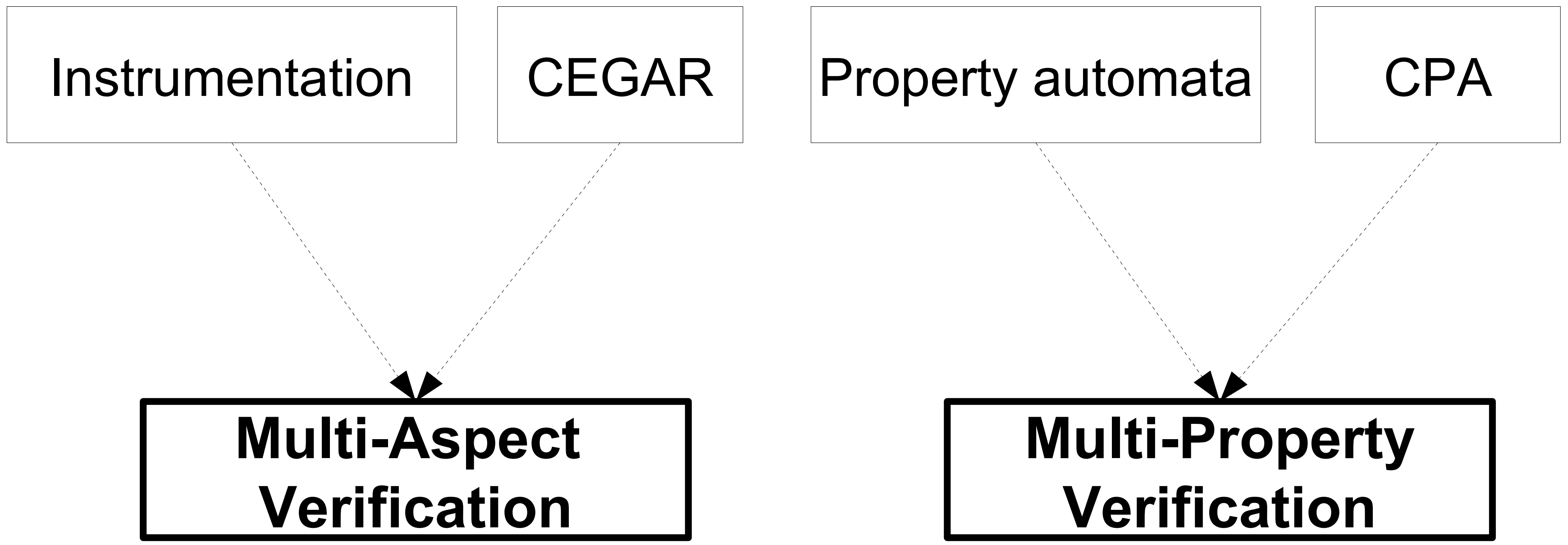
Separated Verification of Properties





- **Lose** of intermediate results
- **Waste** of resources

Verifying Several Properties at once



Multi-Aspect Verification Results*

- 17 properties, 6021 tasks
- **3** times overall speedup
- **5** times average speedup
- **5** times speedup with tasks preparation
- **2%** losses

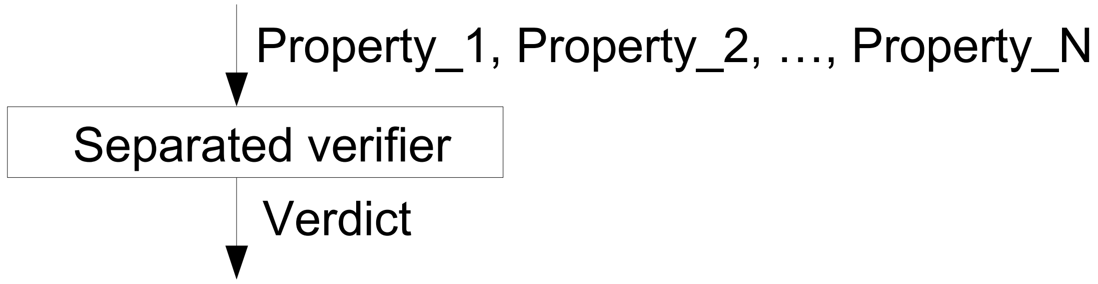
* V. Mordan, V. Mutilin. *Checking Several Requirements at once by CEGAR*. LNCS 9609.

Multi-Property Verification Results*

- 14 properties, 4336 tasks
- **3** times overall speedup
- **8** times average speedup
- **0.4%** losses

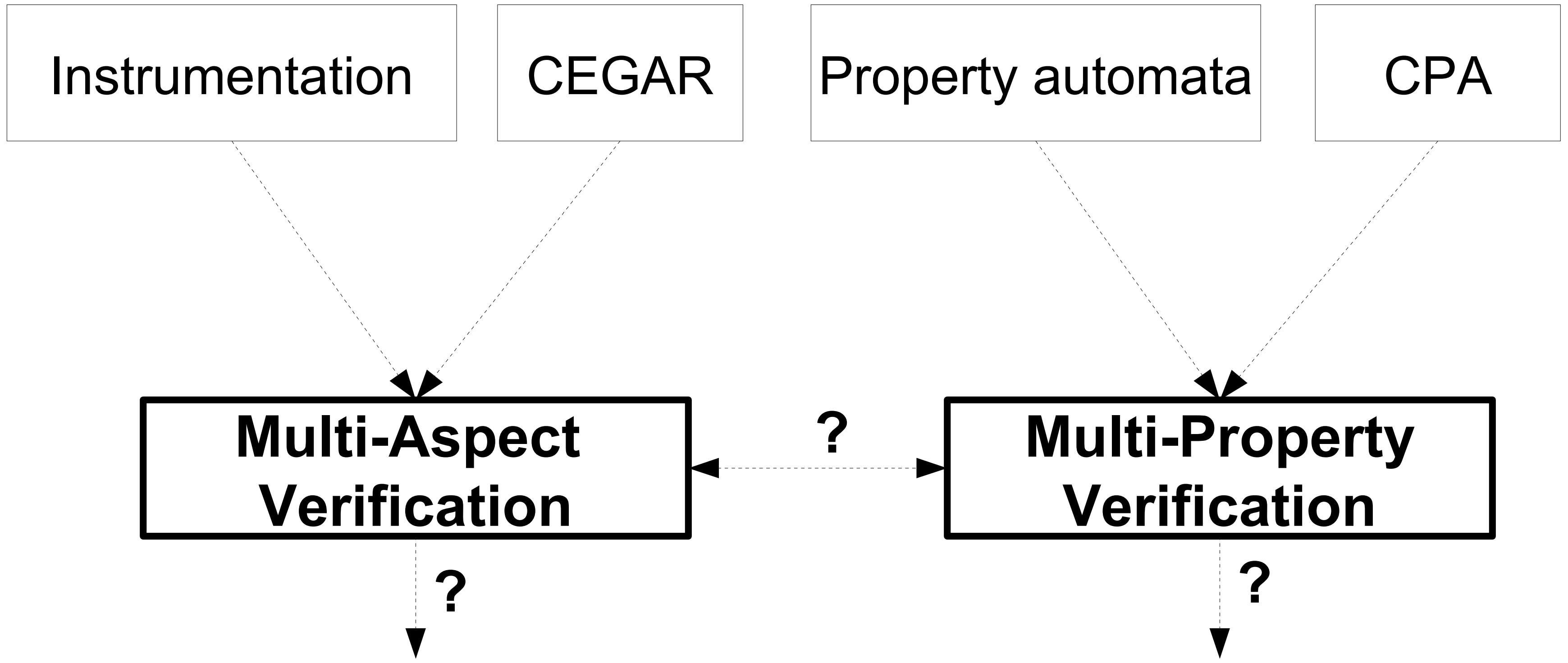
* S. Apel, D. Beyer, V. Mordan, V. Mutilin, A. Stahlbauer. *On-The-Fly Decomposition of Specifications in Software Model Checking*. FSE 2016.

Batch Methods Results



- Almost **no speedup**
- **>15%** losses

Verifying Several Properties at once



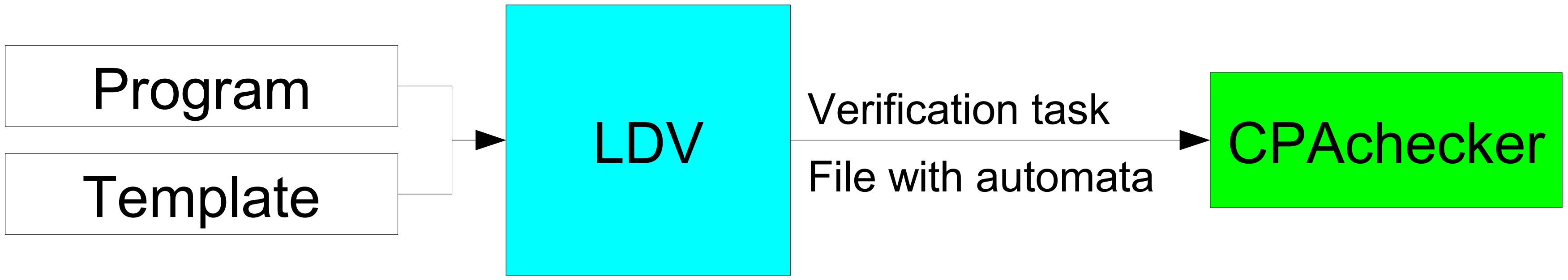
Instrumentation vs Automata

Instrumentation	Property automata
More general **(?)**	**Extension***
Complicate tasks	Do not affect tasks
	More efficient

* S. Apel, D. Beyer, V. Mordan, V. Mutilin, A. Stahlbauer. *On-The-Fly Decomposition of Specifications in Software Model Checking*. FSE 2016.

Complex Property Automata

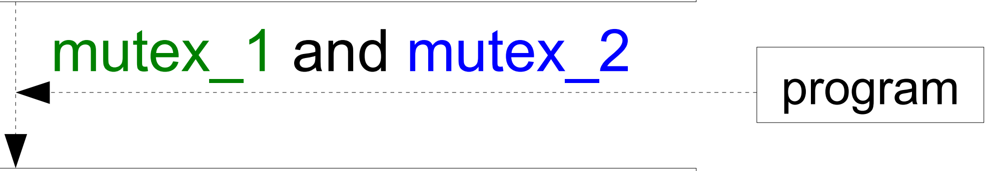
- Generate several automata for property
- Use templates (LDV infrastructure)



File with Automata Example

Template

```
OBSERVER AUTOMATON linux_mutex_{{ arg_sign.id }}
…
MATCH CALL {ldv_mutex_lock{{ arg_sign.id }}($?)} -> …
```



File with automata

```
OBSERVER AUTOMATON linux_mutex_mutex_1
…
MATCH CALL {ldv_mutex_lock_mutex_1($?)} -> …
OBSERVER AUTOMATON linux_mutex_mutex_2
…
MATCH CALL {ldv_mutex_lock_mutex_2($?)} -> …
```

Experiments Setup

- CMAV
  - CPAchecker branch *cmav*, revision 20410
- MPV
  - CPAchecker branch *muauto*, revision 20125
- Machines
  - Intel Xeon E312xx (Sandy Bridge) 2.6GHz (8 cores)
  - 64Gb RAM
  - Ubuntu 14.04 (64-bit) with Linux 3.13
  - Java version 1.7.0_95

Experiments Setup

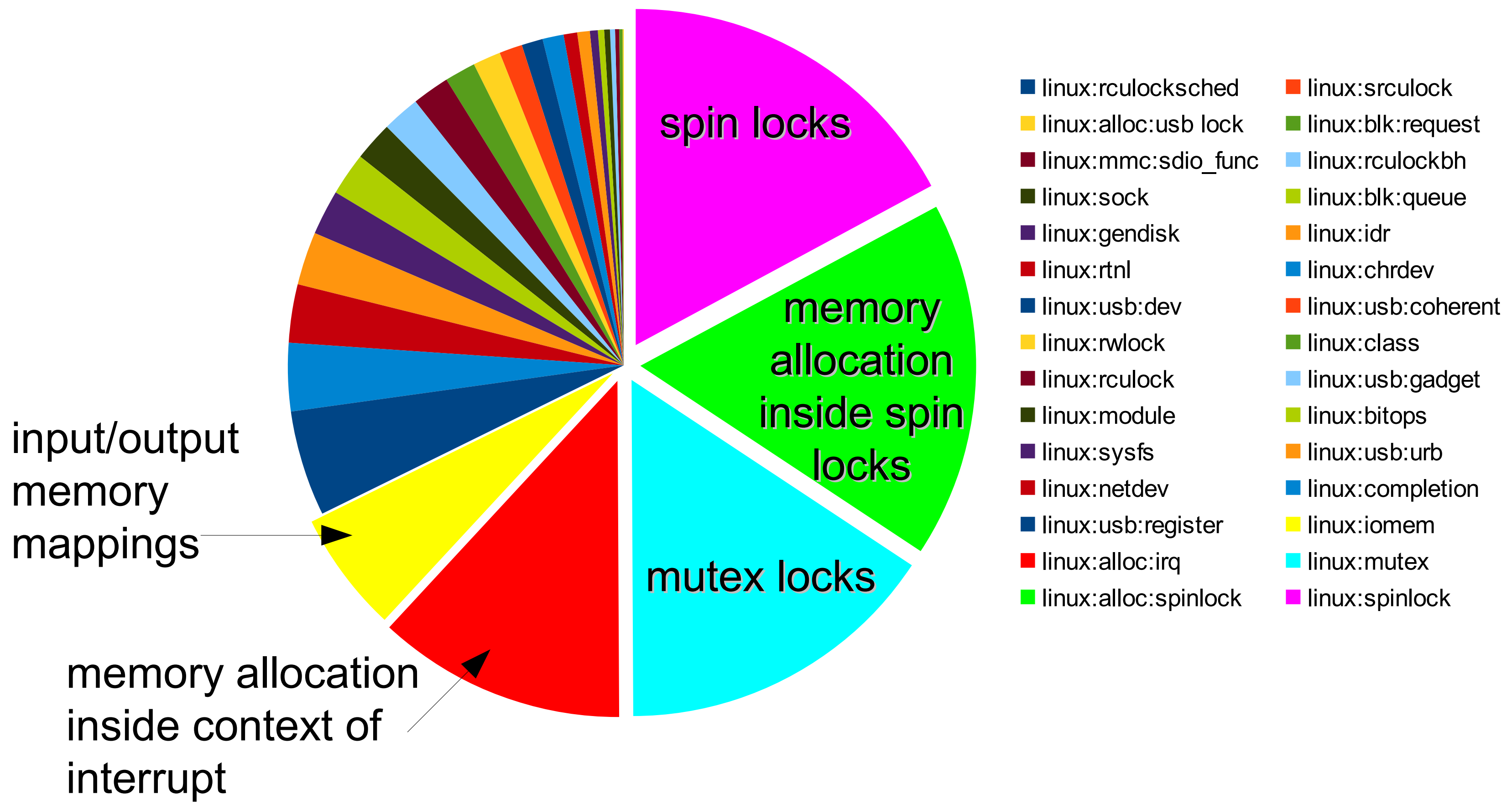
- 30 LDV properties (including 6 complex)
  - Instrumentation and automata
- 4044 verification tasks
  - Based on linux-4.0-rc1
- Limitations
  - 900* CPU seconds (per 1 task and per 1 property)
  - 15GB* of RAM and 13GB Java heap (per 1 verifier run)

27 000 CPU seconds per 1 task and per 30 properties

* According to Competition on Software Verification

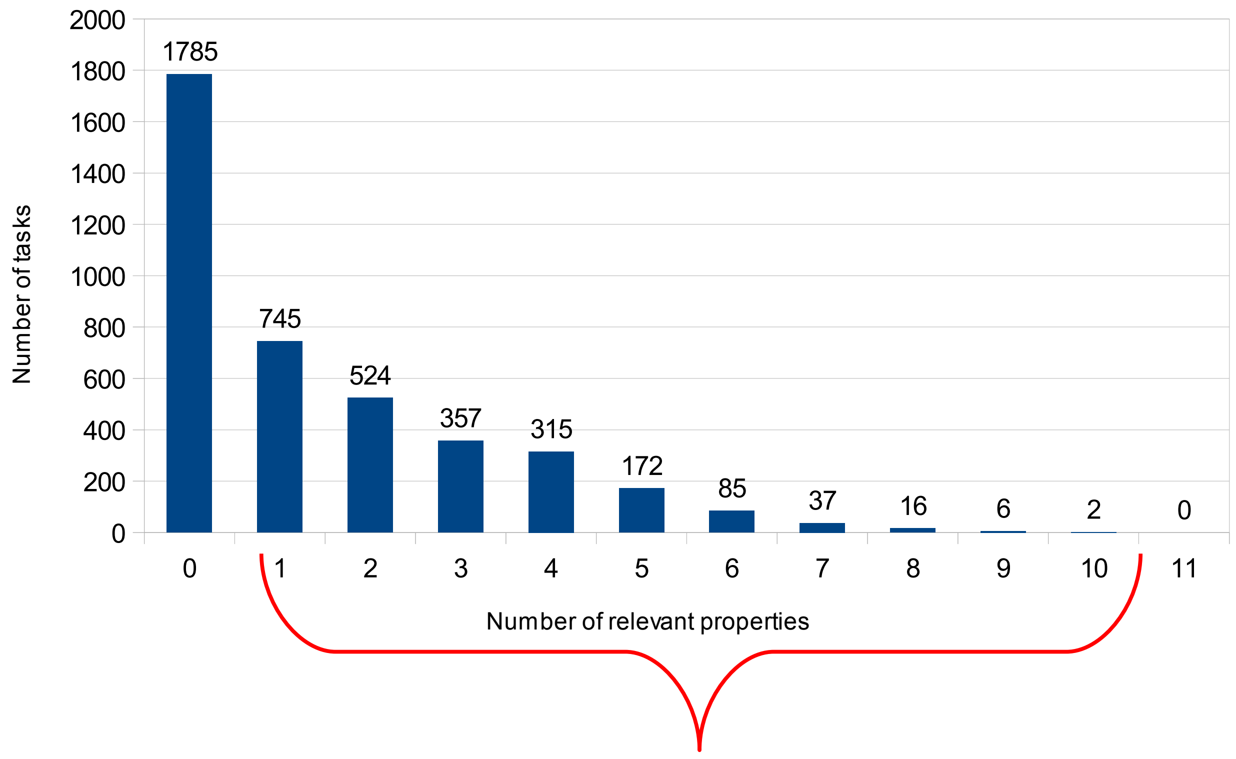
Evaluation of Properties

Distribution of relevant tasks per property



Evaluation of Tasks

- 1.46 properties are relevant per one task



may cause complications

Instrumentation vs Automata Evaluation

Method	Safe	Unsafe	CPU Time
Instrumentation	118 704	667	3 867 000
Property automata	118 946	679	3 485 000

Overall:

- Property automata are faster in **1.11** times
- Property automata solve **0.21%** more tasks

Detailed Comparison for Unsafes

- **0** transitions
  - Safe->Unsafe (additional false alarms)
  - Unsafe->Safe (missed bugs)
- **36** new Unsafes
  - time limits in instrumentation
- **24** lost Unsafes
  - 2 exceptions in automata
  - 22 time limits in automata

+17 Unsafes (input/output memory mappings)

-5 Unsafes (mutex locks)

Features of Automata

- Pure functions with no parameters

```
extern int rtnl_trylock(void);
```

- Potential error trace

```
if (rtnl_trylock()) {
  rtnl_unlock();
} else {
  // . . .
  if (rtnl_trylock()) {
    // no rtnl_unlock
  }
}
```

returns 0

Change of rtnl

returns 1

ERROR

Features of Automata

- Instrumentation

```
int ldv_rtnl_trylock(void) {
  if (/*...*/) {
    /*...*/
    return 1;
  } else
    return 0;
}
```

- Automata
  - Model it by pure function with parameter
  - Add function in CPAchecker configuration

Limitations of Automata

- Do not support pointers

```
MATCH ENTRY -> ENCODE {void *pointer;} ...
```

Exception!

- Currently properties do not need it

Checking Each Assert

Checking only **double locks**

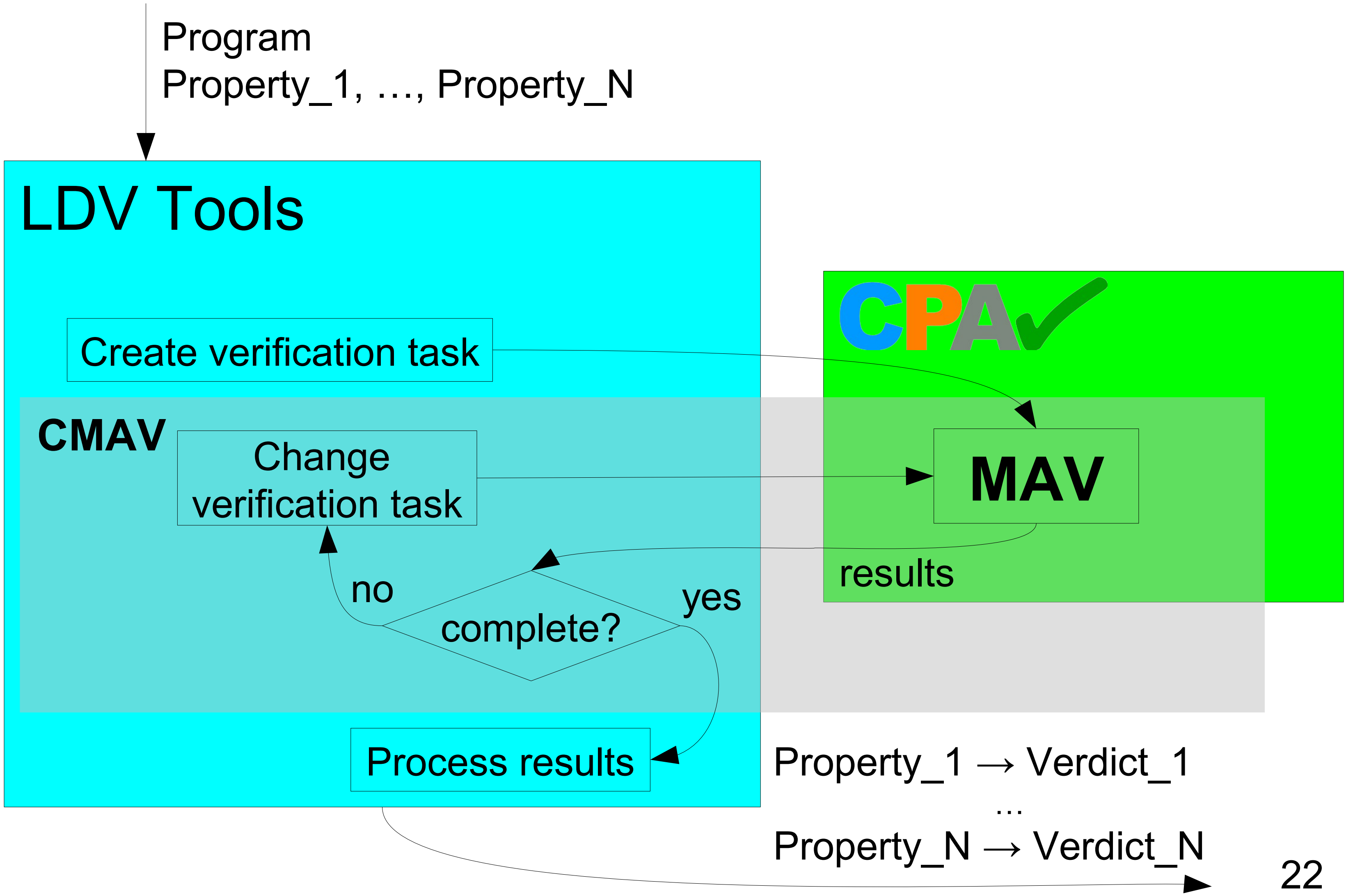
```
STATE USEFIST Unlocked :
  MATCH CALL {ldv_mutex_lock($?)} -> GOTO Locked;
  MATCH {ldv_mutex_unlock($?)} -> ERROR("double unlock");
STATE USEFIST Locked :
  MATCH CALL {ldv_mutex_lock($?)} -> ERROR("double lock");
  MATCH CALL {ldv_mutex_unlock($?)} ->  GOTO Unlocked;
  MATCH CALL {ldv_check_final_state($?)} -> ERROR("locked at exit");
```

```
STATE USEFIST Unlocked :
  MATCH CALL {ldv_mutex_lock($?)} -> GOTO Locked;
  MATCH {ldv_mutex_unlock($?)} -> GOTO Unlocked;
STATE USEFIST Locked :
  MATCH CALL {ldv_mutex_lock($?)} -> ERROR("double lock");
  MATCH CALL {ldv_mutex_unlock($?)} ->  GOTO Unlocked;
  MATCH CALL {ldv_check_final_state($?)} -> GOTO Locked;
```

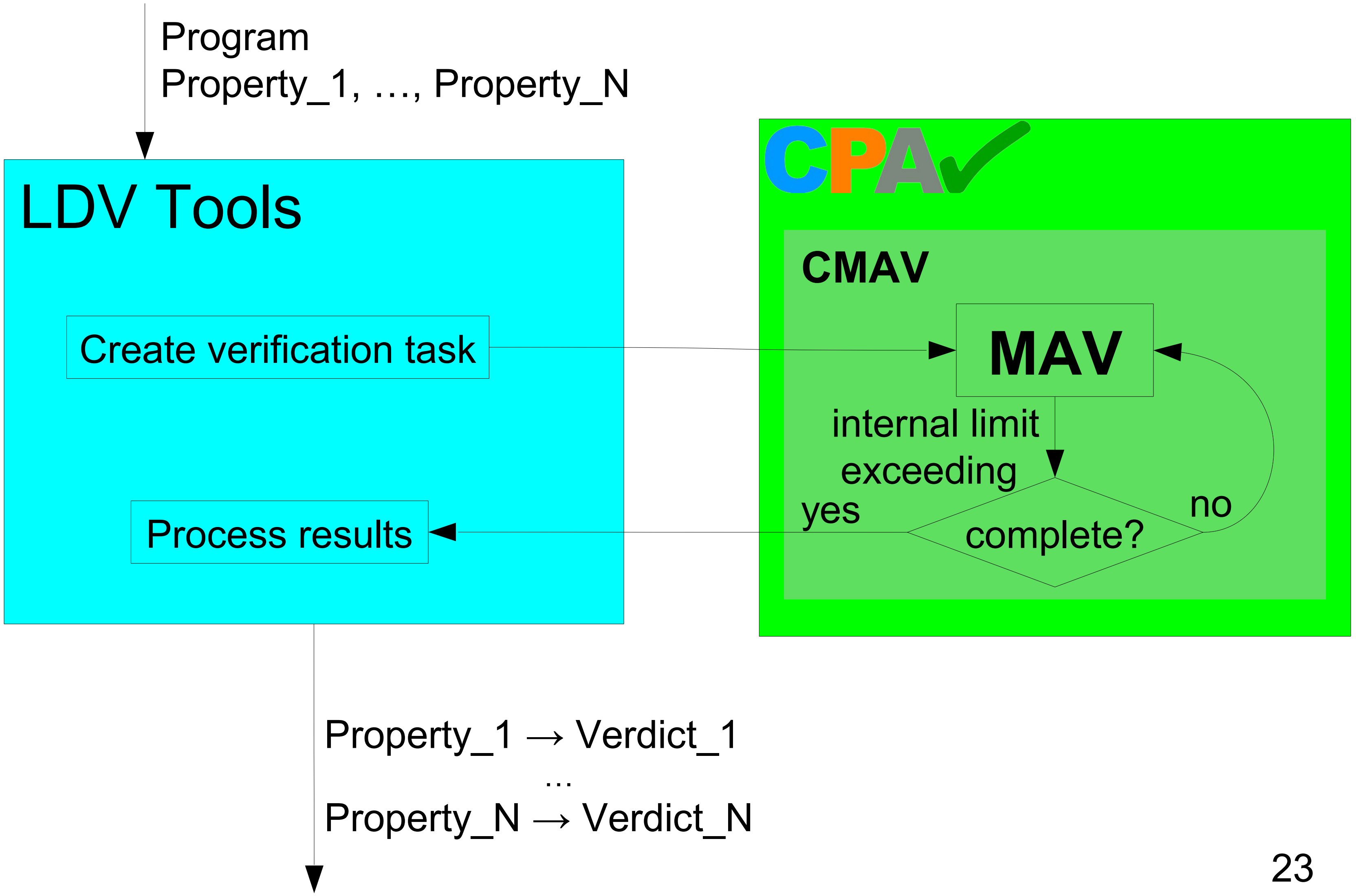
Further Development of MAV

- Internal relaunch
  - Balance between reuse
- Different configurations of internal limitations
  - Balance between speedup and quality

External CMAV



Internal CMAV



Internal vs External

External relaunch	Internal relaunch
Change of tasks	No change of tasks
Detect exceptions	Cannot detect exceptions
Loss of CFA, caches	Reuse of CFA, caches
More additional actions	Less additional actions
Low limitations for iteration	Require more RAM

Internal vs External CMAV Evaluation

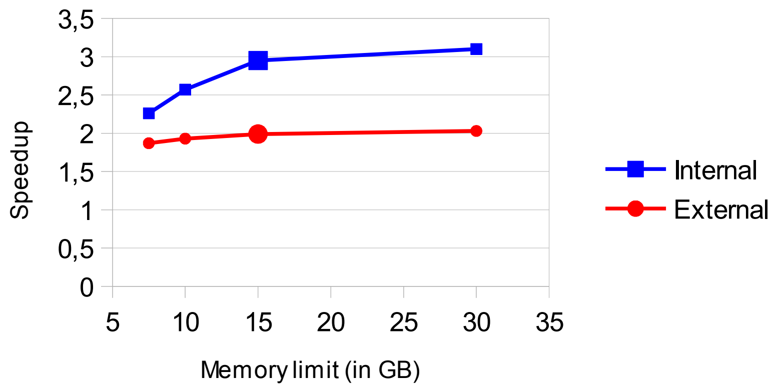
- Limitations per 1 verifier run
  - Internal: 27 000 CPU seconds
  - External: 1200 CPU seconds
    - All iterations no longer than 27 000 seconds

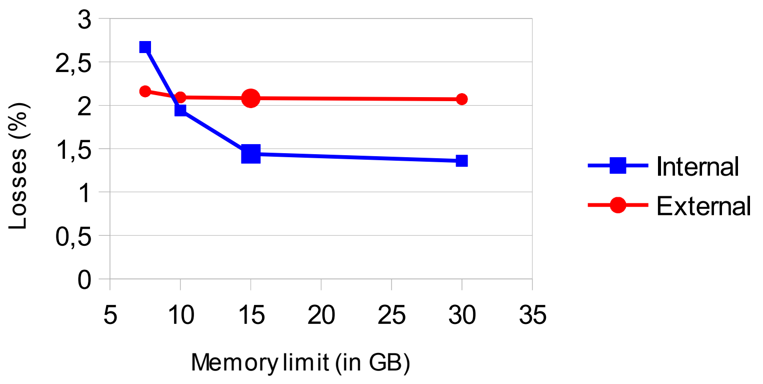
To clean ARG

Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
External CMAV	116 266	621	2.08	1 940 000 1.99	248 000 10.62	2 188 000 2.97
Internal CMAV	117 006	643	1.44	1 310 000 2.95	220 000 12	1 530 000 4.25

Internal vs External CMAV (Different Memory Limitations)

- Memory limit
    - 7.5GB, 10GB, **15GB**, 30GB
- Java heap limit
    - 13/15 of memory limit

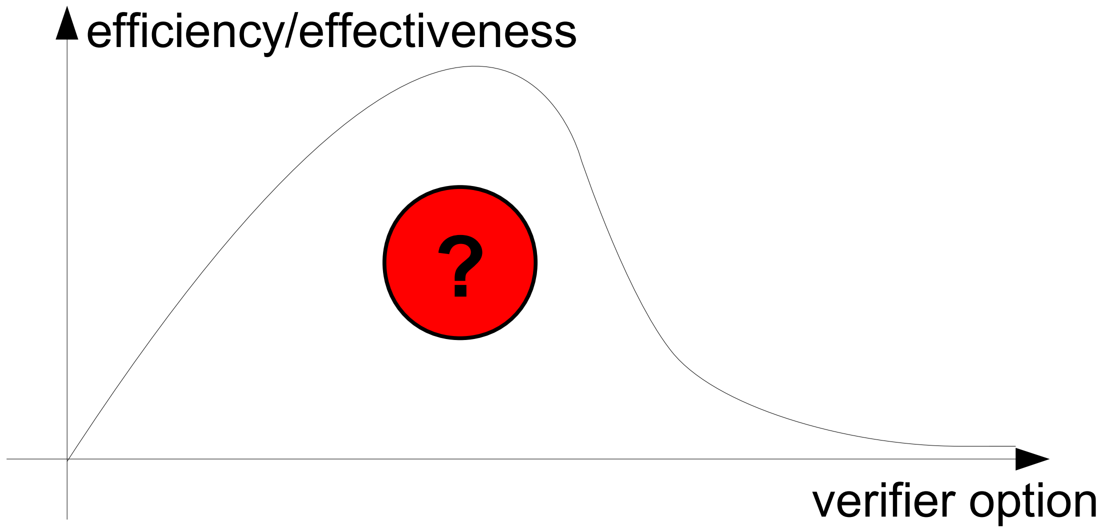




Internal Limitations Idea

- Specify used heuristics
- Implicitly change quality of verification
- **DO NOT** change resources

  per 1 task and per 1 property



Internal Limitations

same as in separated verification

for CMAV iterations

L4. No heuristics

- ATL: 900s, IITL: 20s

L3. Long intervals in MAV leads to Unknown

- ATL: 900s, IITL: 20s, BITL: 450s

L2. More strict version of L3

- ATL: 900s, IITL: 20s, BITL: 100s

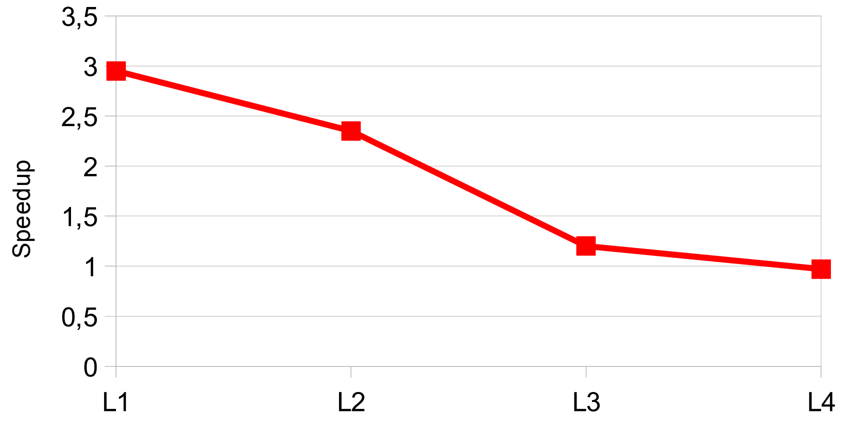
L1* L2 + long first interval leads to Unknown for all properties

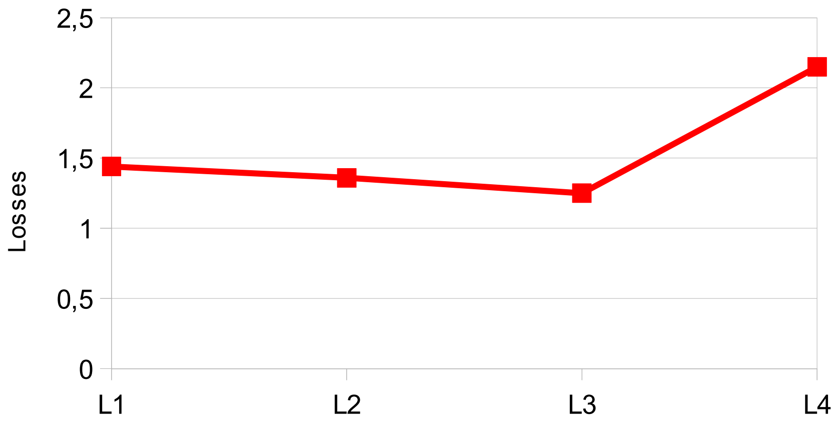
- ATL: 900s, IITL: 20s, BITL: 100s, FITL: 100s

* V. Mordan, V. Mutilin. *Checking Several Requirements at once by CEGAR*. LNCS 9609.

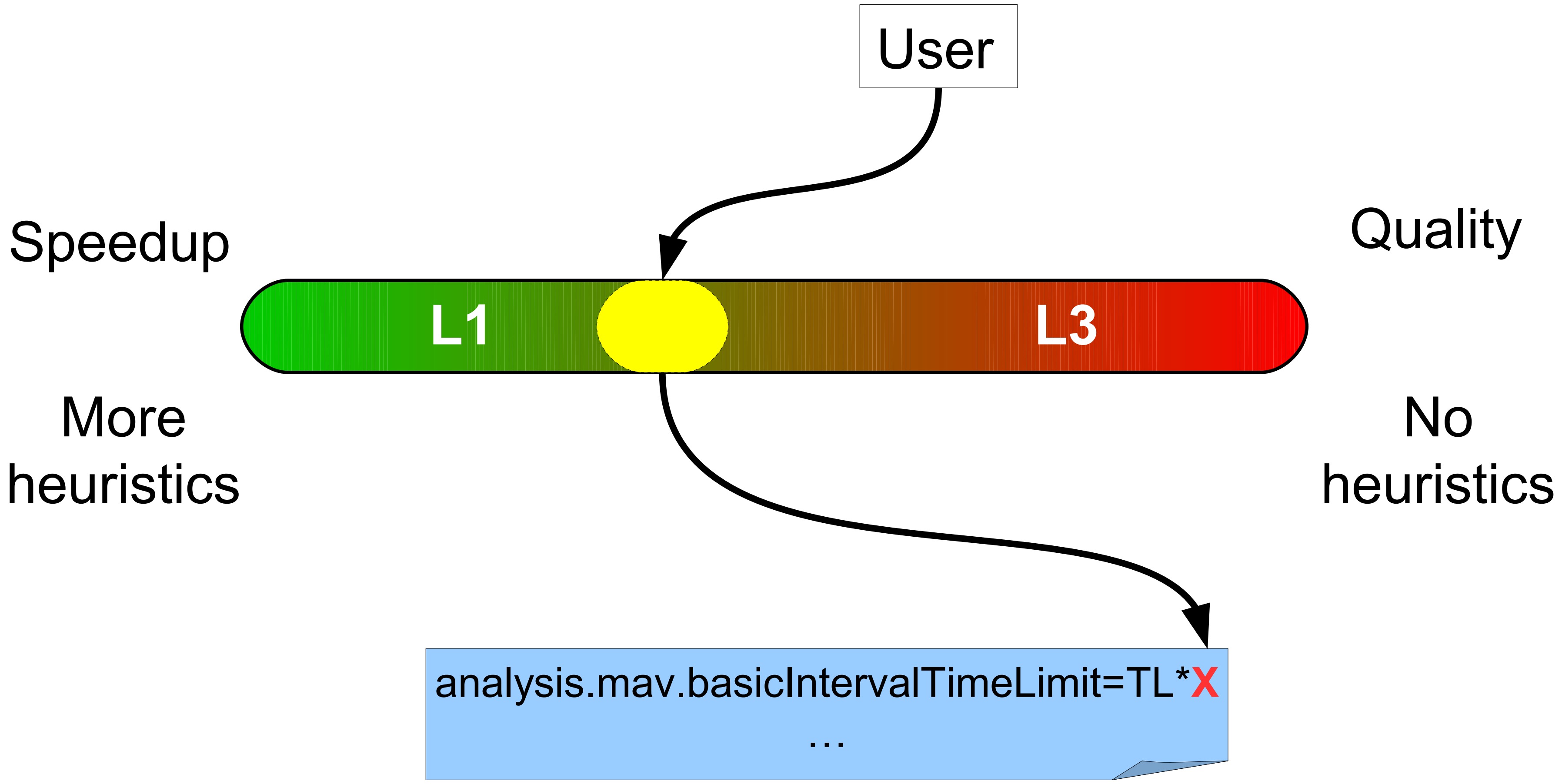
Different Internal Limitations Evaluation

Method	Safe	Unsafe	CPU Time	Speedup	Losses
L1	117 006	643	1 310 000	2.95	1.44%
L2	117 100	643	1 642 000	2.36	1.36%
L3	117 220	656	3 218 000	1.2	1.25%
L4	**115 838**	**653**	**3 971 000**	**0.97**	**2.15%**





User Perspective



analysis.mav.basicIntervalTimeLimit=TL*X
…

MAV vs MPV Comparison Idea

- Irrelevant tasks
- “Easy” tasks
- “Hard” tasks
- All tasks (4044)

Setup

  - CMAV: L1 configuration, internal relaunch
  - MPV: strategy *Relevance*
    (step 1: 200s; step 2: 1200s; step 3: 900s)

MAV vs MPV (Irrelevant Tasks)

- 1785 tasks (irrelevant for all properties)
- 53 550 Safe verdicts
- MAV
  - **55 000** CPU seconds
- MPV
  - **39 000** CPU seconds (**1.4 faster**)
- Instrumentation **complicates** tasks
- **40%** of all tasks → **3-4%** of overall CPU time

MAV vs MPV (“Easy” Tasks)

- 90 tasks (based on auxiliary modules)
- 2690 Safes, 5 Unsafes, 5 Unknowns
- MAV
  - **7 000** CPU seconds (**~1.57 faster**)
- MPV
  - **11 000** CPU seconds
- MAV solves tasks **faster**
  - **Heuristics**

MAV vs MPV (“Hard” Tasks)

- 96 tasks (based on *fs* modules)
- MAV
  - **64 000** CPU seconds
  - **2 517** Safes, 10 Unsafes
- MPV
  - **57 000** CPU seconds (**1.12 faster**)
  - **2 601** Safes, 10 Unsafes (**3%** more solved)
- MPV solves **more** tasks **faster**
  - **Heuristics**

MAV vs MPV (All Tasks)

Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
Separated (instrumentation)	118 704	667	0	3 867 000 -	2 639 000 -	6 506 000 -
Separated (automata)	118 946	679	-0.21	3 485 000 1.11	2 639 000 -	6 124 000 1.06
CMAV	117 006	643	1.44	**1 310 000** **2.95**	220 000 12	1 530 000 **4.25**
MPV	**118 312**	**669**	**0.54**	1 357 000 2.57	**169 000** **15.62**	**1 526 000** 4.01

MAV and MPV vs Batch

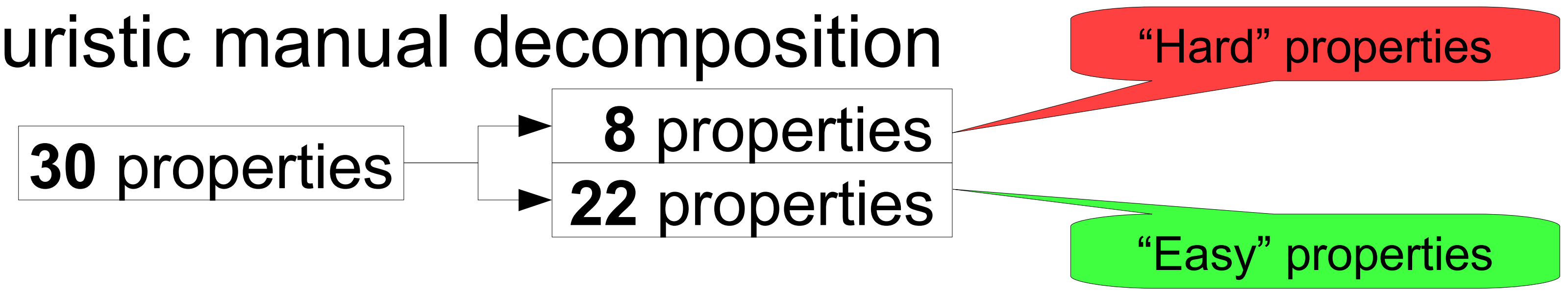
Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
CMAV (no limitations)	**111 156**	**678**	**6.31**	**4 558 000 0.85**	220 000 12	**4 778 000 1.36**
CMAV (*L1*)	117 006	643	1.44	1 310 000 2.95	220 000 12	1 530 000 4.25
MPV (*All*)	**110 681**	**532**	**7.03**	**6 275 000 0.56**	169 000 15.62	**6 444 000 0.95**
MPV (*Relevance*)	118 312	669	0.54	1 357 000 2.57	169 000 15.62	1 526 000 4.01

CMAV is more **optimized**

- Cleaning precision
- Single property automaton

"Unfair" CMAV

- Observation
    - Some properties are better to verify separately
- Heuristic manual decomposition



Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
CMAV	117 006	643	1.44	1 310 000 2.95	**220 000** **12**	1 530 000 4.25
"Unfair" CMAV	**117 404**	**650**	**1.1**	**1 066 000** **3.63**	**324 000** **8.15**	**1 390 000** **4.68**

x2 tasks

Overhead Costs of MPV

- MPV (*Sep*) vs Separated runs
  - Reuse CFA and caches
  - Overhead costs **(?)**

Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
Separated (automata)	118 946	679	0	3 485 000 -	2 639 000 -	6 124 000 -
MPV (*Relevance*)	**118 312**	**669**	**0.54**	**1 357 000** **2.57**	**169 000** **15.62**	**1 526 000** **4.01**
MPV (*Sep*)	117 688	652	1.07	**3 496 000** **0.99**	169 000 15.62	3 665 000 1.67

MAV and MPV

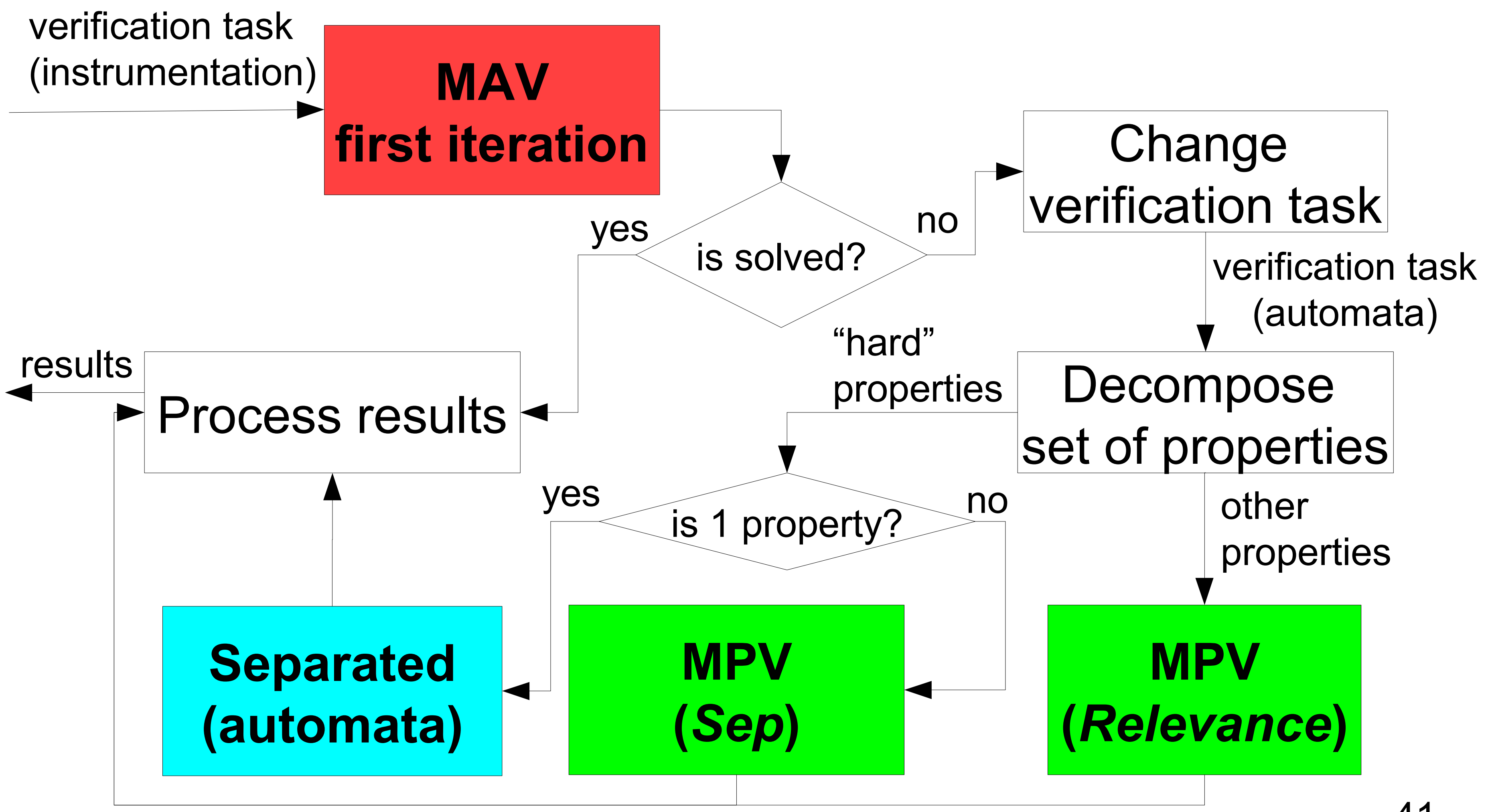
MAV	MPV
Better for: easy tasks	Better for: hard tasks
More optimized	More resource intensive
More speedup	Less speedup
More heuristics	Less heuristics
More losses	Less losses

Methods have different application area

Sequential Combination Idea

- Unite strengths of MAV and MPV
    - Instrumentation vs Automata
    - Internal vs External relaunches
    - Heuristics and optimizations vs Quality
- Minimize the influence of their weaknesses
- Default method for the users

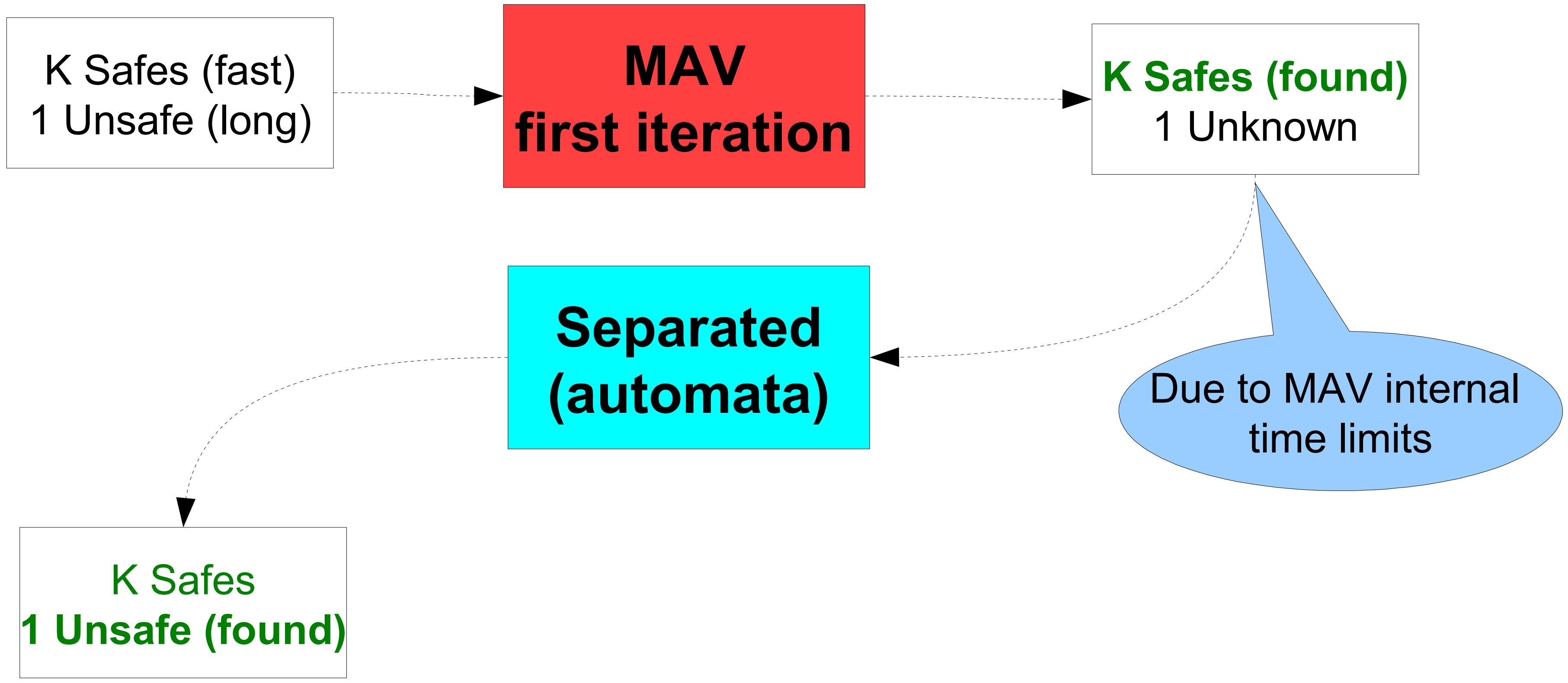
Sequential Combination Schema



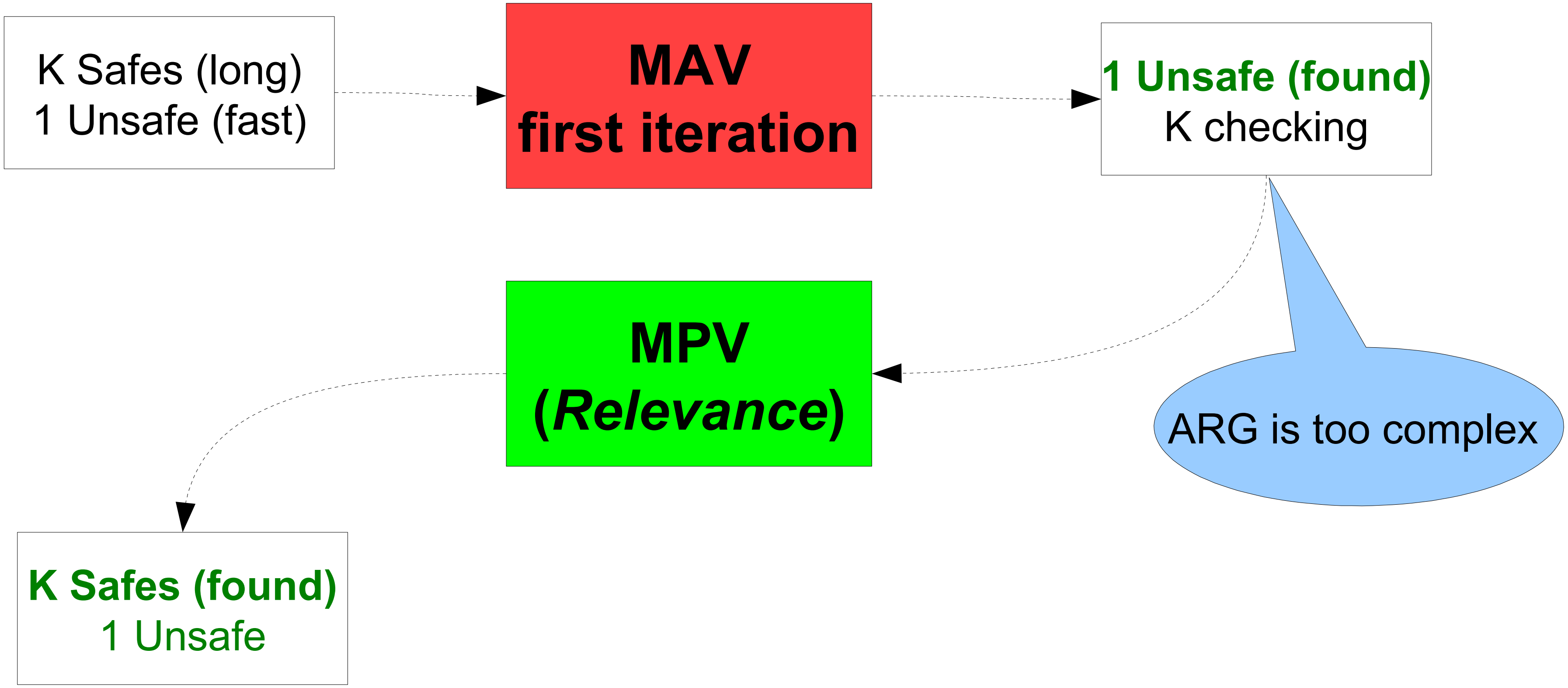
Sequential Combination Advantages

- Relaunches
  - External between steps – **change tasks**
  - Internal inside each step – **efficiency**
- Preparation of verification tasks
  - **Strengths** of instrumentation and automata
- MAV solves “easy” tasks – **optimizations**
- MPV solves “hard” tasks – **quality**

Motivating Example 1



Motivating Example 2



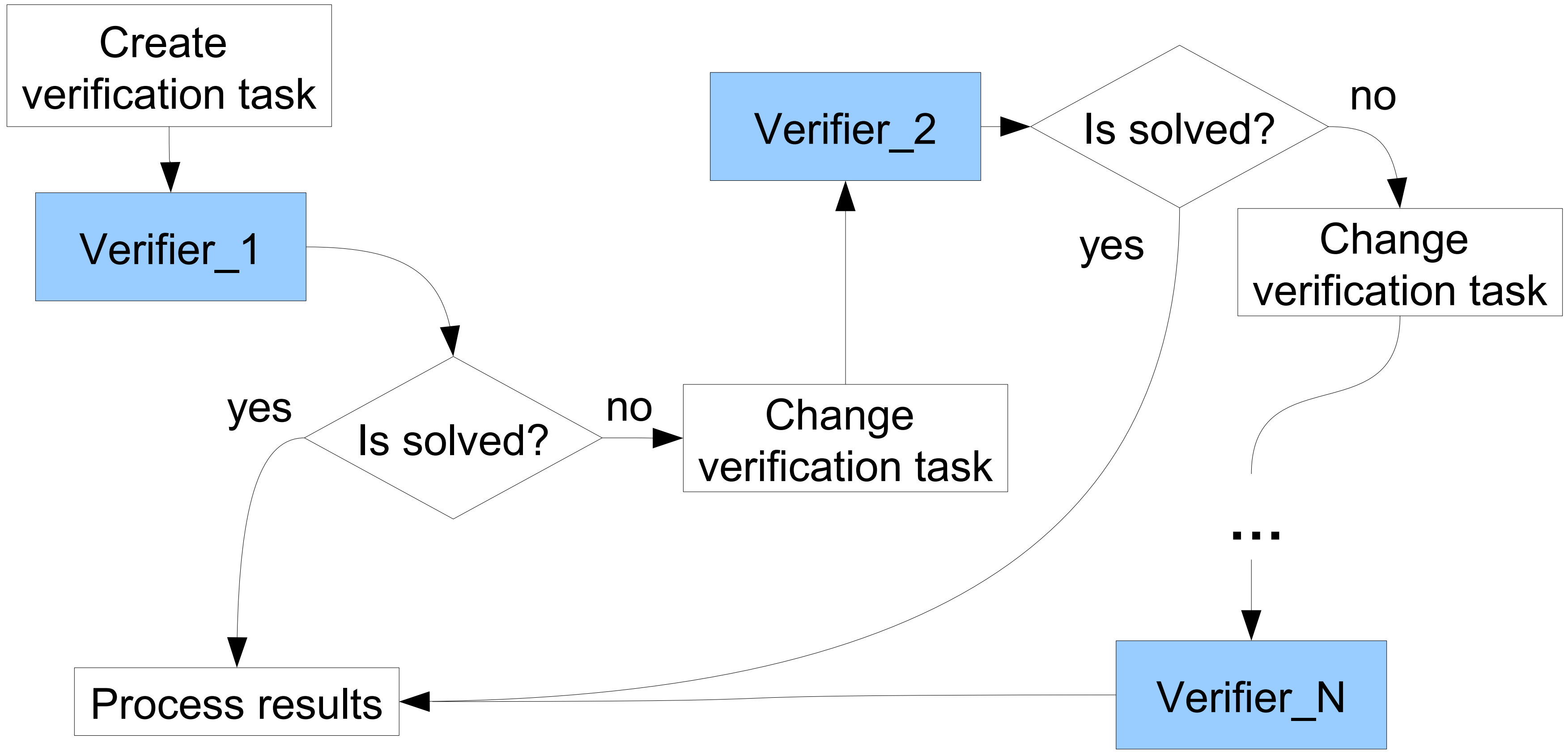
Sequential Combination Implementation

- Sequential Combination 1 (precise)
  - Recheck all “heuristic” Unknowns of MAV
    - “Hard” properties
  - Limit MAV to half of basic time limit
- Sequential Combination 2 (fast)
  - Recheck only “heuristic” L1 Unknowns
  - Limit MAV as external (1200 CPU seconds)
- …

Evaluation

Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
Separated (instrumentation)	118 704	667	0	3 867 000 -	2 639 000 -	6 506 000 -
Separated (automata)	118 946	679	-0.21	3 485 000 1.11	2 639 000 -	6 124 000 1.06
CMAV	117 006	643	1.44	**1 310 000 2.95**	**220 000 12**	**1 530 000 4.25**
MPV	**118 312**	**669**	**0.54**	1 357 000 2.57	**169 000 15.62**	1 526 000 4.01
Sequential combination 1	**118 734**	**691**	**-0.05**	**1 342 000 2.88**	227 000 11.63	**1 569 000 4.14**
Sequential combination 2	**118 492**	**650**	**0.19**	**1 231 000 3.14**	**223 000 11.82**	**1 454 000 4.47**

General Case Schema

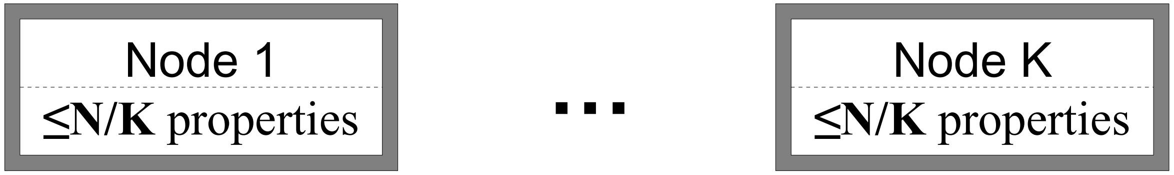


Any Number of Properties

- In practice
  - Hundreds of properties
- Methods may lose efficiency/effectiveness
  - Instrumentation in MAV
  - Overhead costs in MPV
- Manual decomposition?
  - Solve more tasks, which are more “easier”

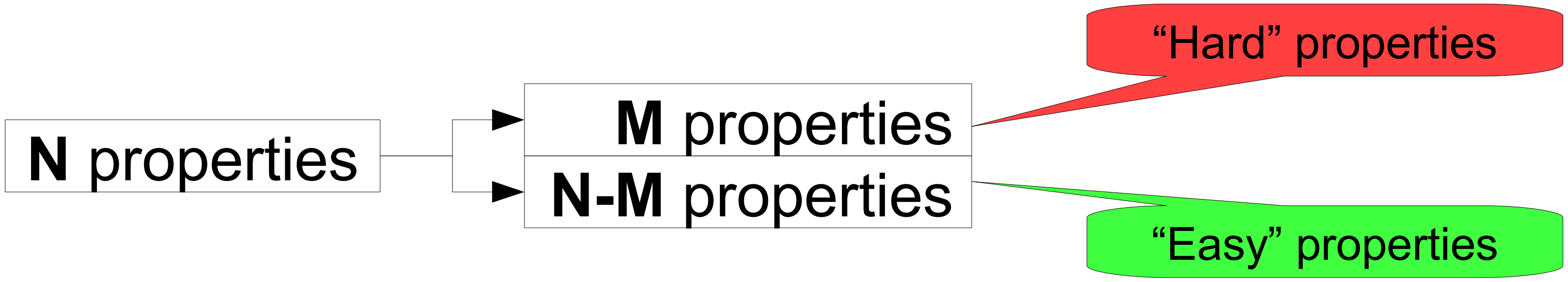
Manual Decomposition of Properties

- Random decomposition



  - **Significant loss of efficiency**

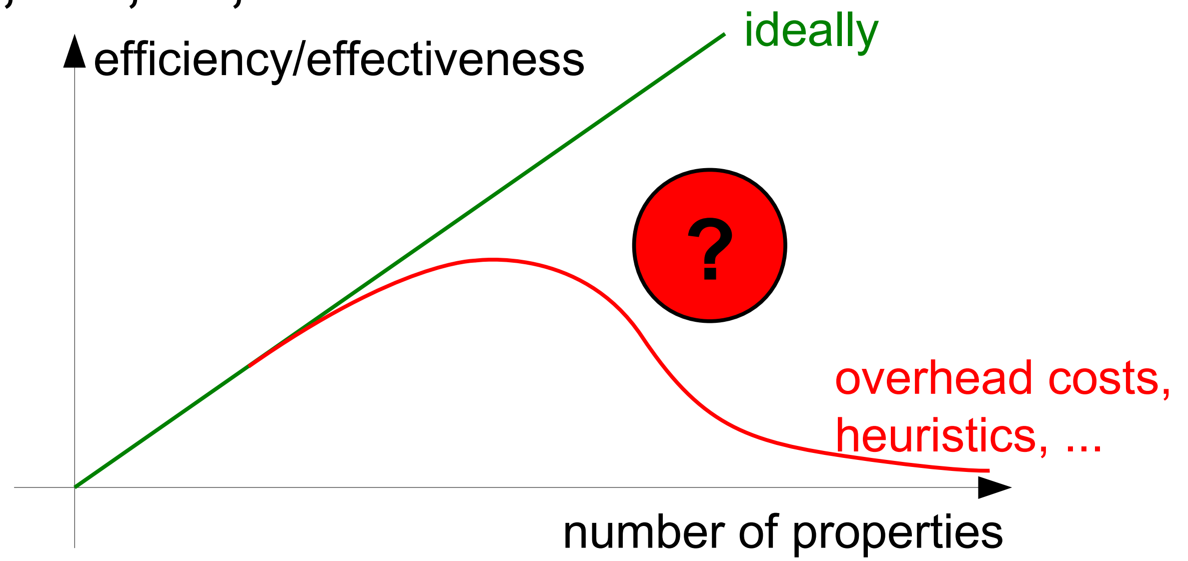
- Based on difficulty function (“unfair” CMAV)



  - **<u>ONLY</u> for fixed set of tasks**

Different Number of Properties Experiment Idea

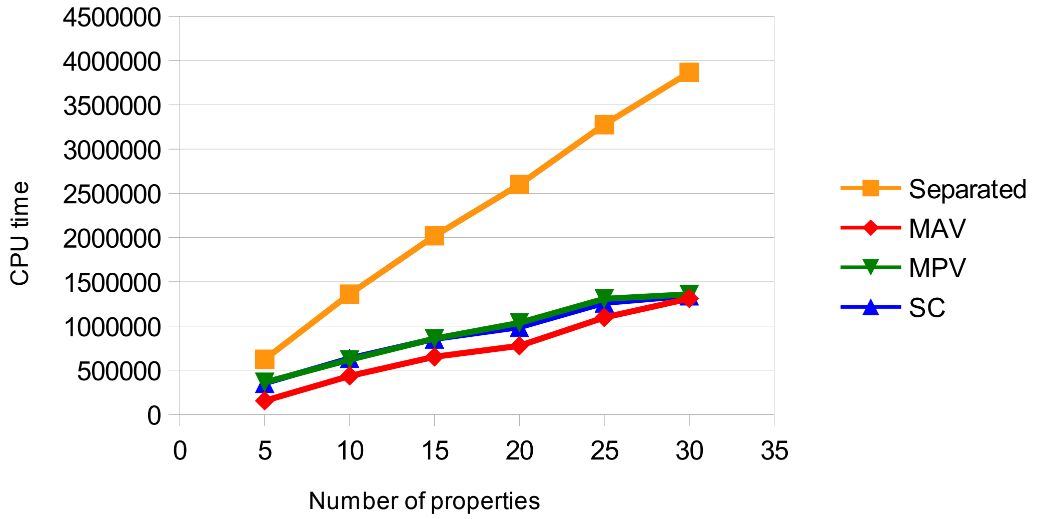
- Sort 30 properties randomly
    - *linux:module*, *linux:alloc:irq*, …, *linux:rtnl*
- Take N properties to verify
    - 5, 10, ..., 30



LinuxTesting.org

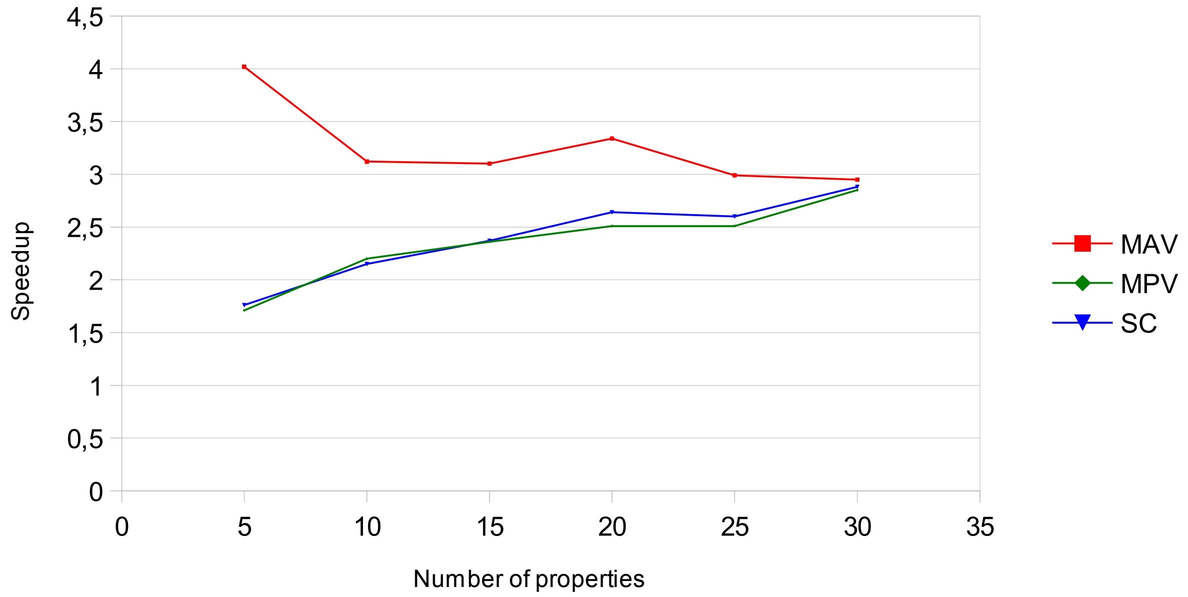
Comparison of CPU Time

Properties	Separated	CMAV	MPV	SC
5	623 000	155 000	364 000	353 000
10	1 361 000	436 000	619 000	634 000
15	2 020 000	651 000	857 000	853 000
20	2 599 000	777 000	1 036 000	986 000
25	3 276 000	1 096 000	1 307 000	1 260 000
30	3 866 000	1 310 000	1 357 000	1 342 000



Comparison of Speedup

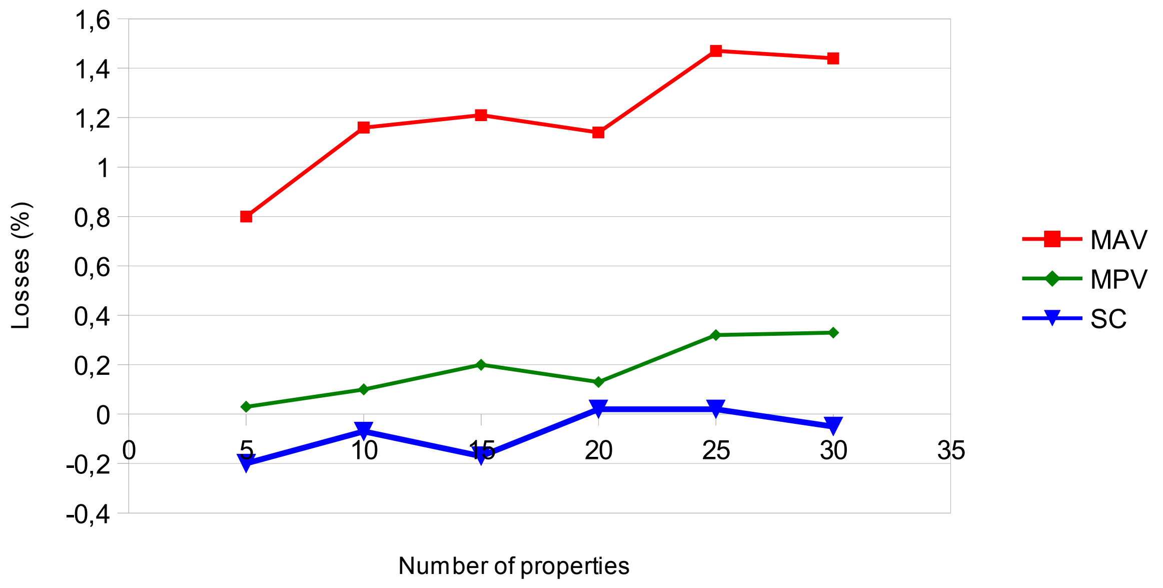
Properties	CMAV	MPV	SC
5	4.02	1.71	1.76
10	3.12	2.2	2.15
15	3.1	2.36	2.37
20	3.34	2.51	2.64
25	2.99	2.51	2.6
30	2.95	2.85	2.88



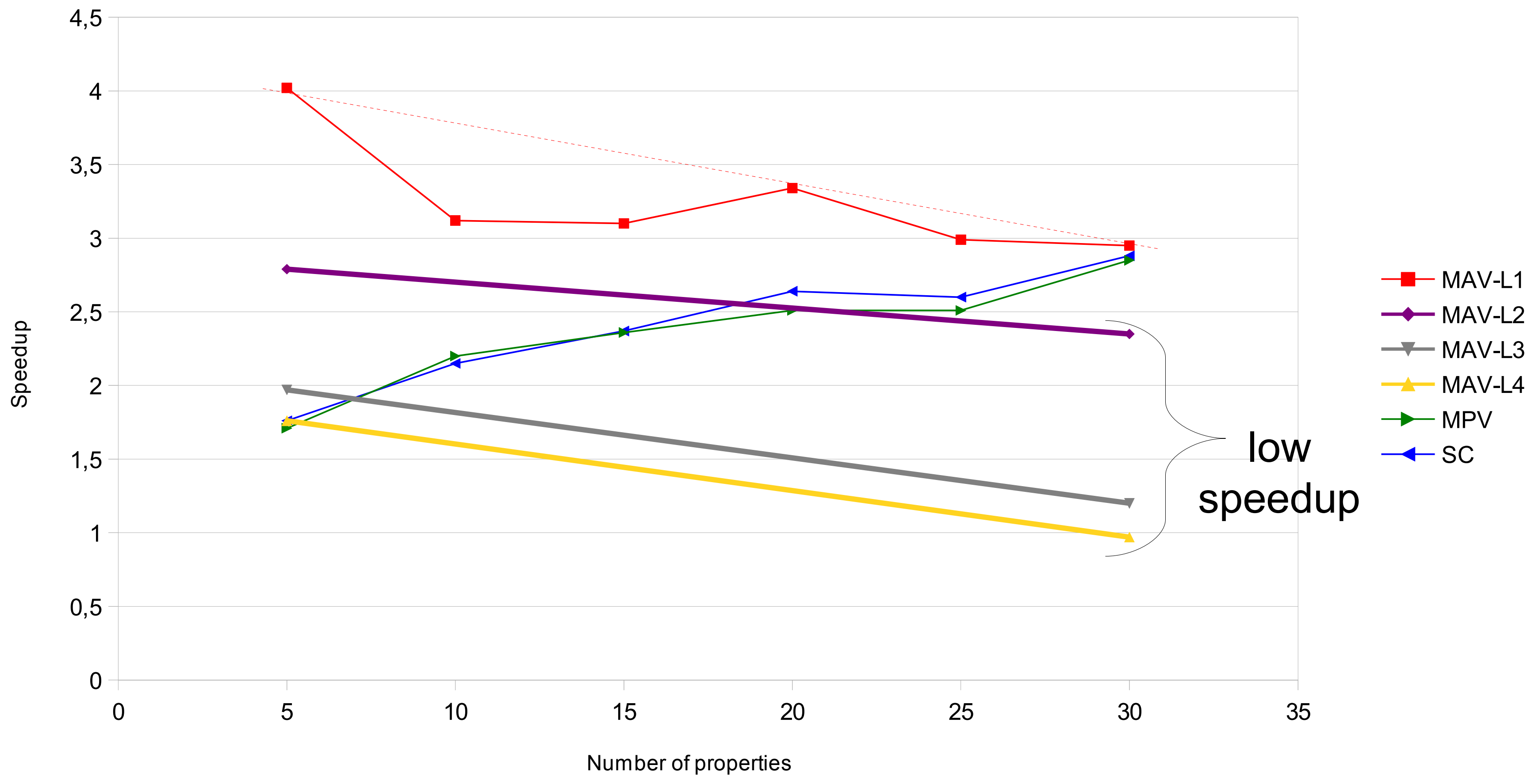
LinuxTesting.org

Comparison of Losses

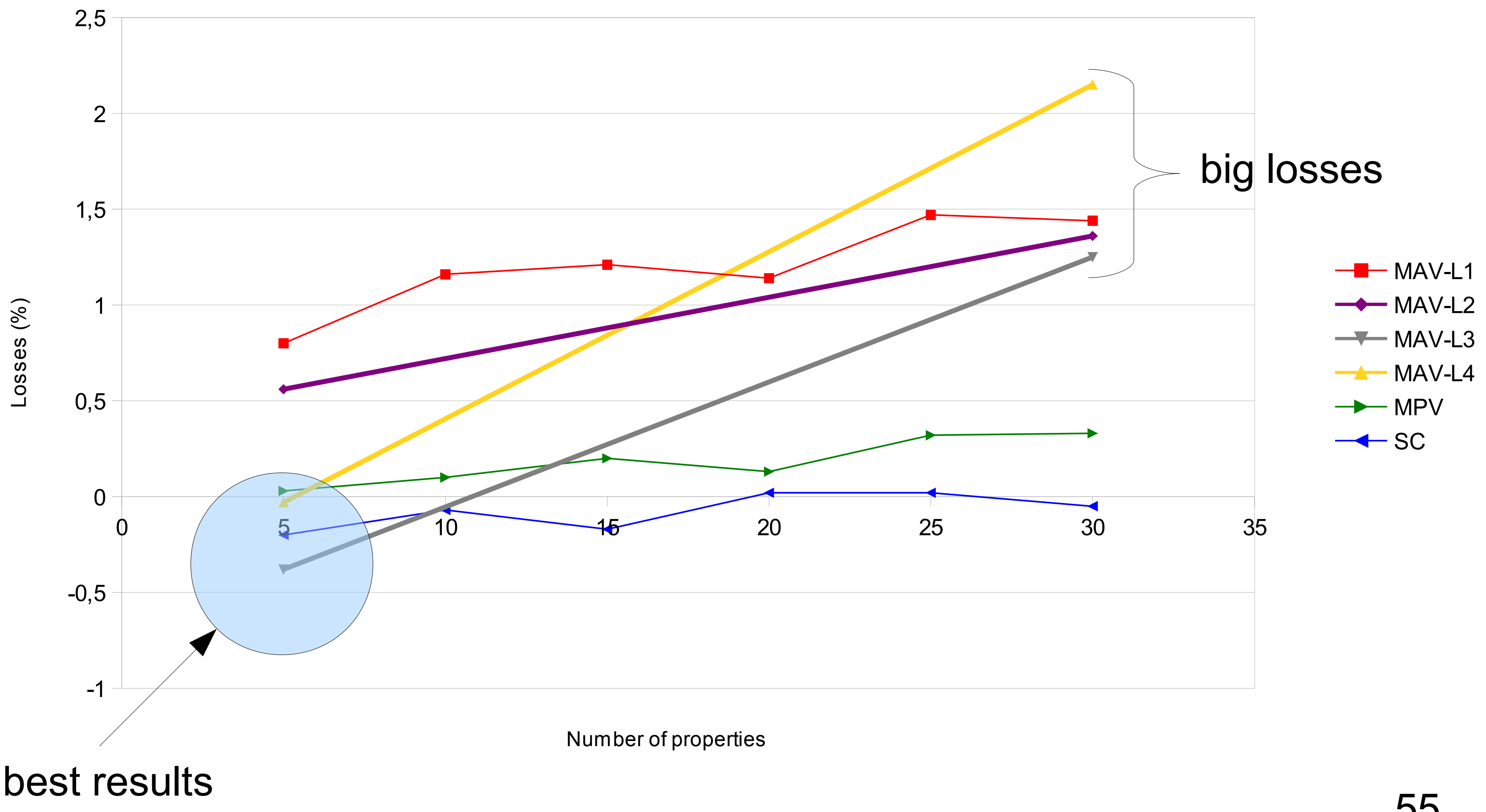
Properties	CMAV	MPV	SC
5	0.8	0.03	-0.2
10	1.16	0.1	-0.07
15	1.21	0.2	-0.17
20	1.14	0.13	0.02
25	1.47	0.32	0.02
30	1.44	0.33	-0.05



Comparison of Speedup (CMAV)

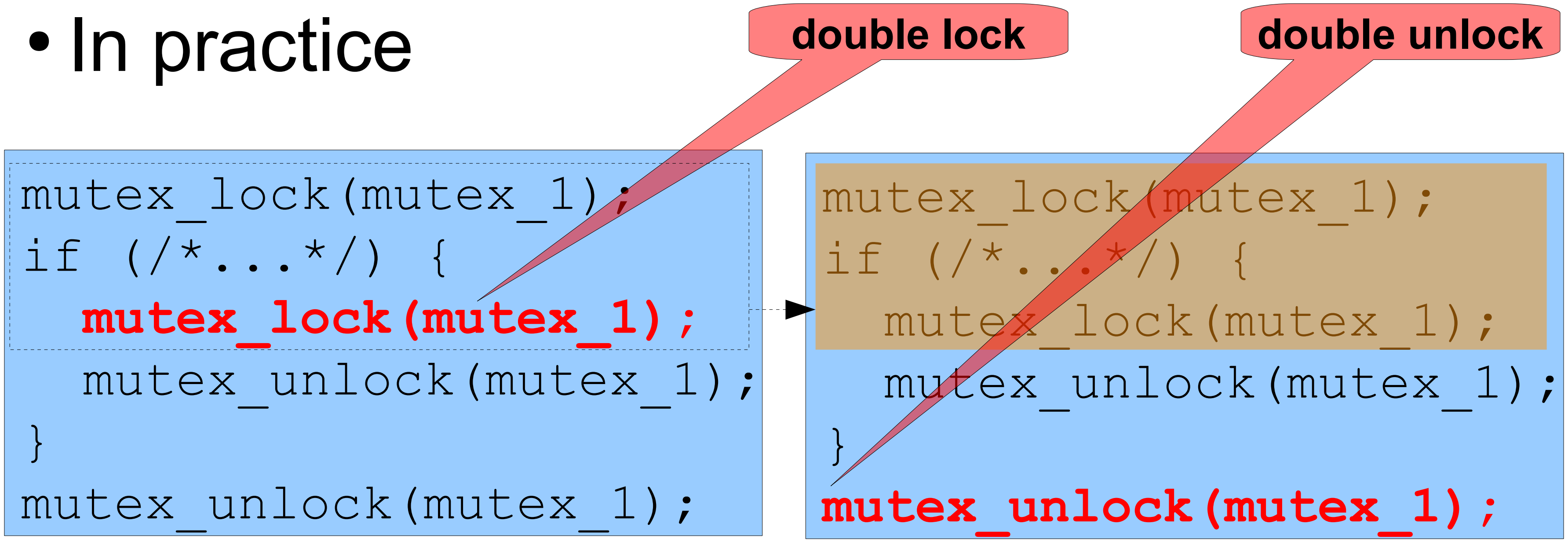


Comparison of Losses (CMAV)



88 Properties

- Each property → several asserts
- 30 properties → 88 asserts
- In practice



```
mutex_lock(mutex_1);
if (/*...*/) {
  mutex_lock(mutex_1);
  mutex_unlock(mutex_1);
}
mutex_unlock(mutex_1);
```

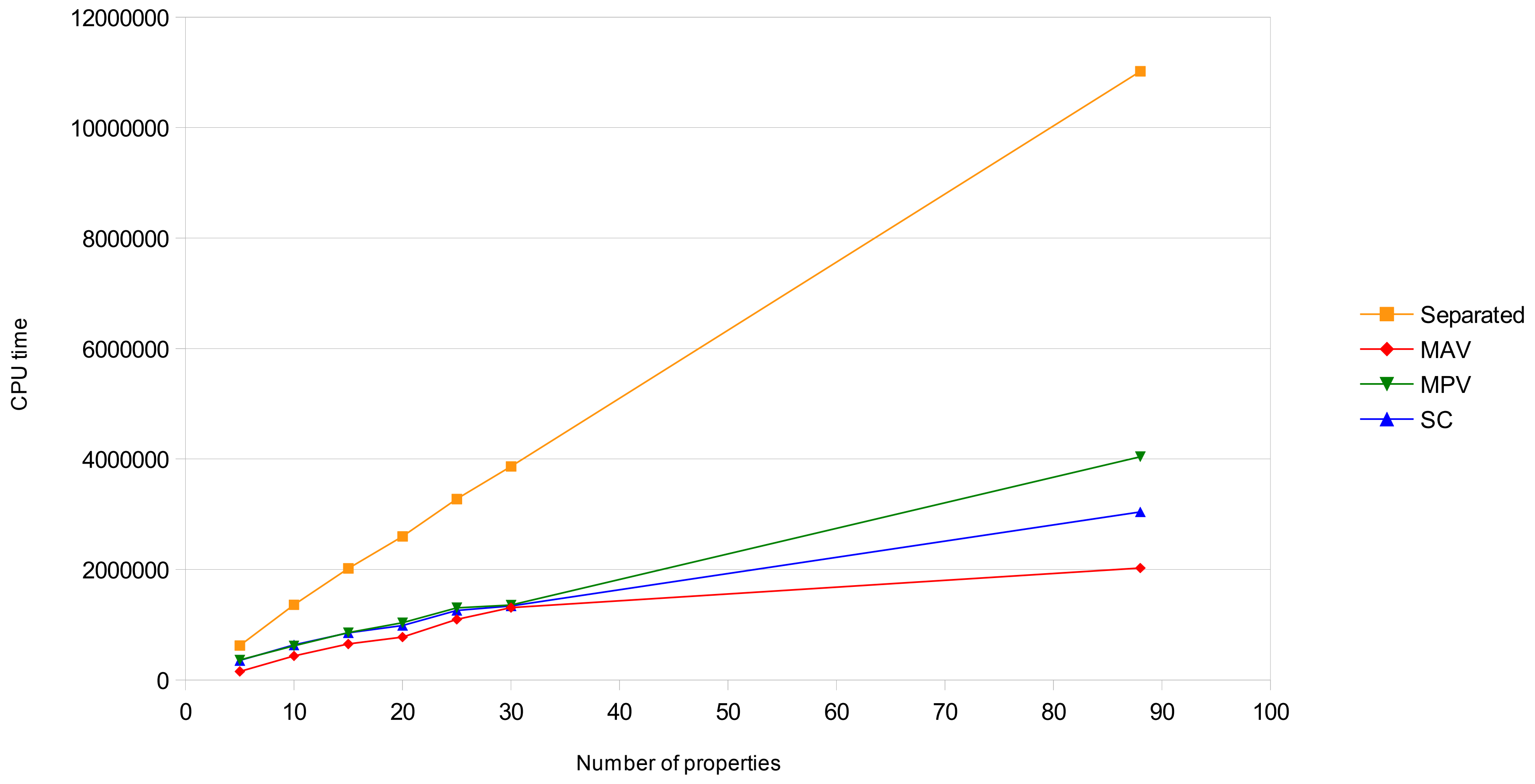
```
mutex_lock(mutex_1);
if (/*...*/) {
  mutex_lock(mutex_1);
  mutex_unlock(mutex_1);
}
mutex_unlock(mutex_1);
```

88 Properties (Evaluation)

- Limitation per 1 task and 88 properties
    - 79 200 seconds (22 hours)

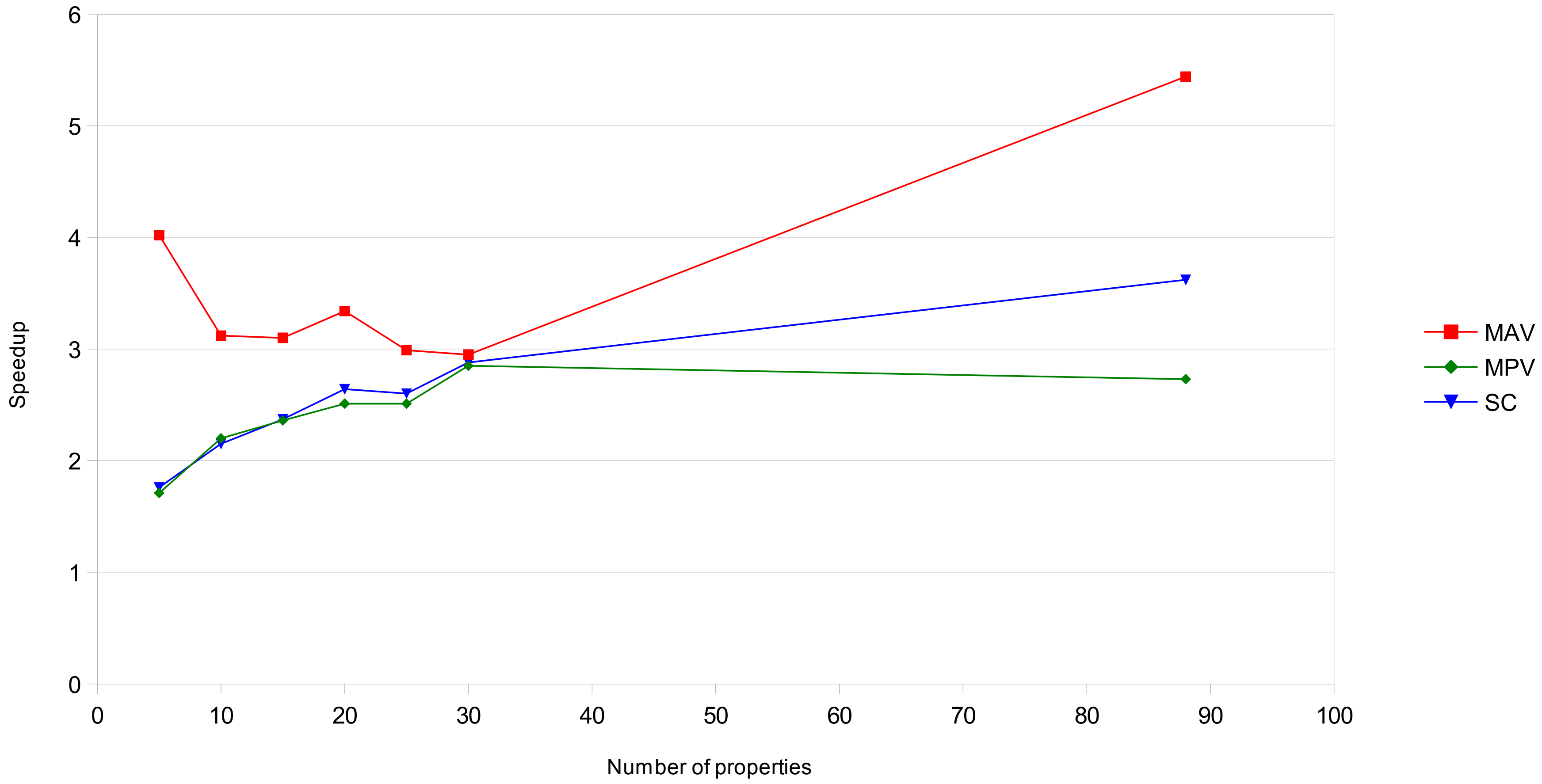
Method	Safe	Unsafe	Losses	CPU time	Speedup
Separated	349 884	1 059	0%	11 020 000	1
CMAV	344 903	952	1.45%	**2 027 000**	**5.44**
MPV	344 109	907	1.69%	4 043 000	2.73
Sequential combination	**347 594**	**1 045**	**0.66%**	3 042 000	3.62

88 Properties (CPU Time)

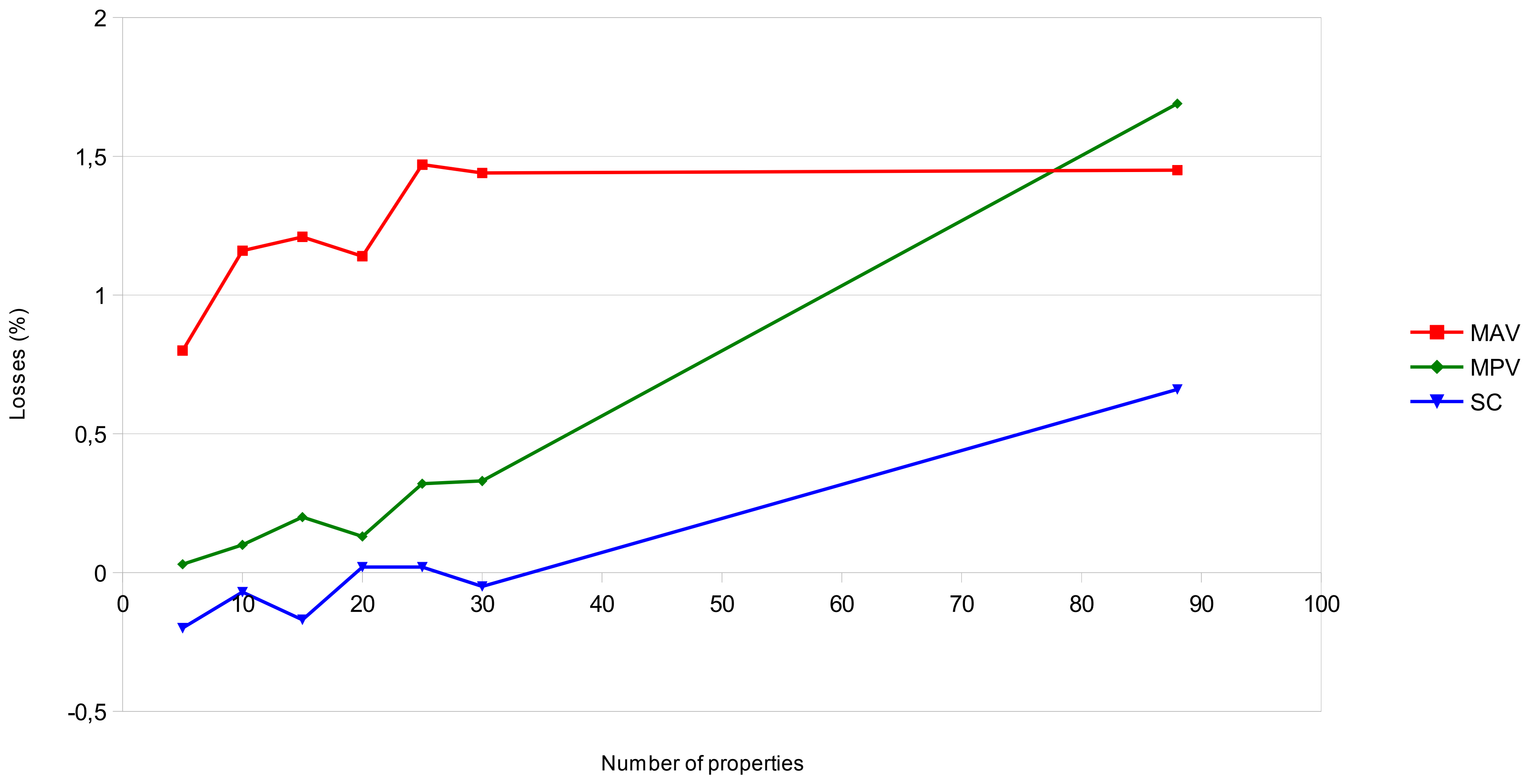


LinuxTesting.org

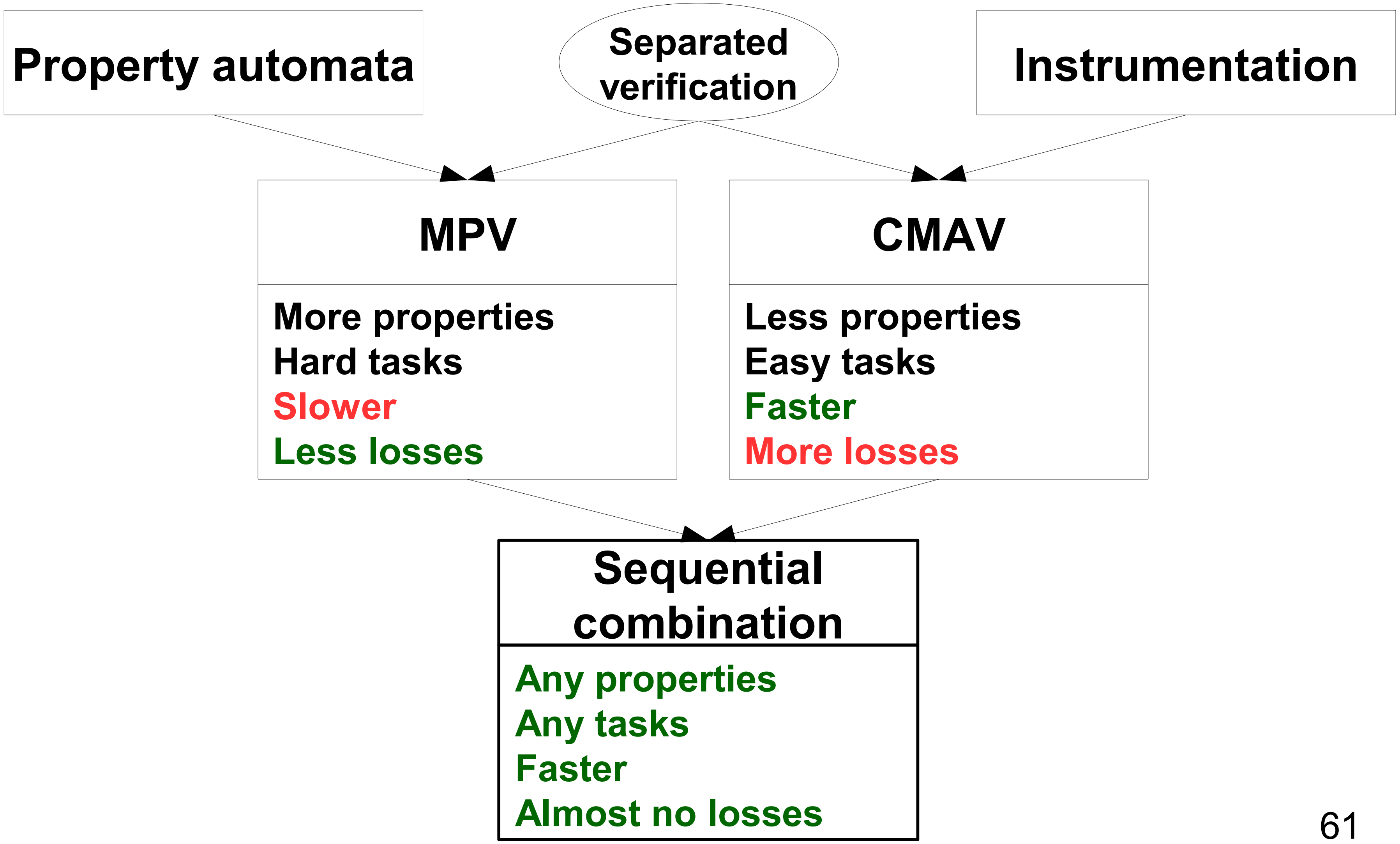
88 Properties (Speedup)



88 Properties (Losses)



Conclusion


Property automata
Separated verification
Instrumentation
MPV
More properties
Hard tasks
Slower
Less losses
CMAV
Less properties
Easy tasks
Faster
More losses
Sequential combination
Any properties
Any tasks
Faster
Almost no losses

LinuxTesting.org

Thank you

Vitaly Mordan
mordan@ispras.ru

ISP RAS
Institute for System Programming of the Russian Academy of Sciences

Comparison of Verdicts

Properties	Separated		CMAV		MPV		Sequential combination	
	Safe	Unsafe	Safe	Unsafe	Safe	Unsafe	Safe	Unsafe
5	19843	85	19684	85	19833	89	19879	89
10	39491	292	39036	285	39449	294	39515	296
15	59359	307	58646	298	59236	309	59457	313
20	79230	369	78338	353	79125	370	79202	382
25	98842	610	97406	582	98516	614	98791	639
30	118702	669	117006	643	118312	669	118734	691

LinuxTesting.org

Instrumentation vs Automata (Detailed Unsafes)

Property	Instrumentation	Automata	Unknown ->Unsafe	Unsafe ->Unknown
linux:module	76	79	3	0
linux:gendisk	18	18	1	1
linux:blk:queue	13	15	2	0
linux:mutex	56	52	1	5
linux:spinlock	22	22	1	1
linux:alloc:spin lock	5	6	1	0
linux:usb:urb	92	92	5	5
linux:usb:coherent	39	39	1	1
linux:class	3	4	1	0
linux:chrdev	9	9	1	1
linux:usb:dev	35	34	0	1
linux:completion	63	61	0	2
linux:sysfs	47	47	2	2
linux:iomem	131	143	17	5

There are no changes for other properties

Internal vs External CMAV (Different Memory Limitations)

Method	Safe	Unsafe	Losses (%)	CPU time (seconds) / speedup		
				CPAchecker	LDV	Overall
External CMAV 7.5 GB	116 172	620	2.16	2 063 000 1.87	274 000 9.63	2 337 000 2.78
Internal CMAV 7.5 GB	115 559	629	2.67	1 710 000 2.26	220 000 12	1 930 000 3.37
External CMAV 10 GB	116 249	621	2.09	2 005 000 1.93	270 000 9.77	2 275 000 2.86
Internal CMAV 10 GB	116 434	629	1.94	1 505 000 2.57	220 000 12	1 725 000 3.77
External CMAV 30 GB	116 278	621	2.07	1 901 000 2.03	245 000 10.78	2 146 000 3.03
Internal CMAV 30 GB	117 107	643	1.36	1 249 000 3.1	220 000 12	1 469 000 4.43

Different Internal Limitations (5 Properties)

Method	Safe	Unsafe	CPU Time	Speedup	Losses
L1	19 684	85	155 000	4.02 ↑	0.8% ↓
L2	19 733	85	223 000	2.79	0.55%
L3	19 928	85	317 000	1.97	-0.43%
L4	**19 849**	**85**	**353 000**	**1.76**	**-0.03%**