2015, LDV Workshop, Moscow

LinuxTesting.org

Checking Several Aspects at once

Vitaly Mordan
mordan@ispras.ru

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Aspects

- Safety property
  - *an offset should not be greater than a size of an array*
- Aspect

Program source code

```
...
assert(offset <= size);
...
```

or

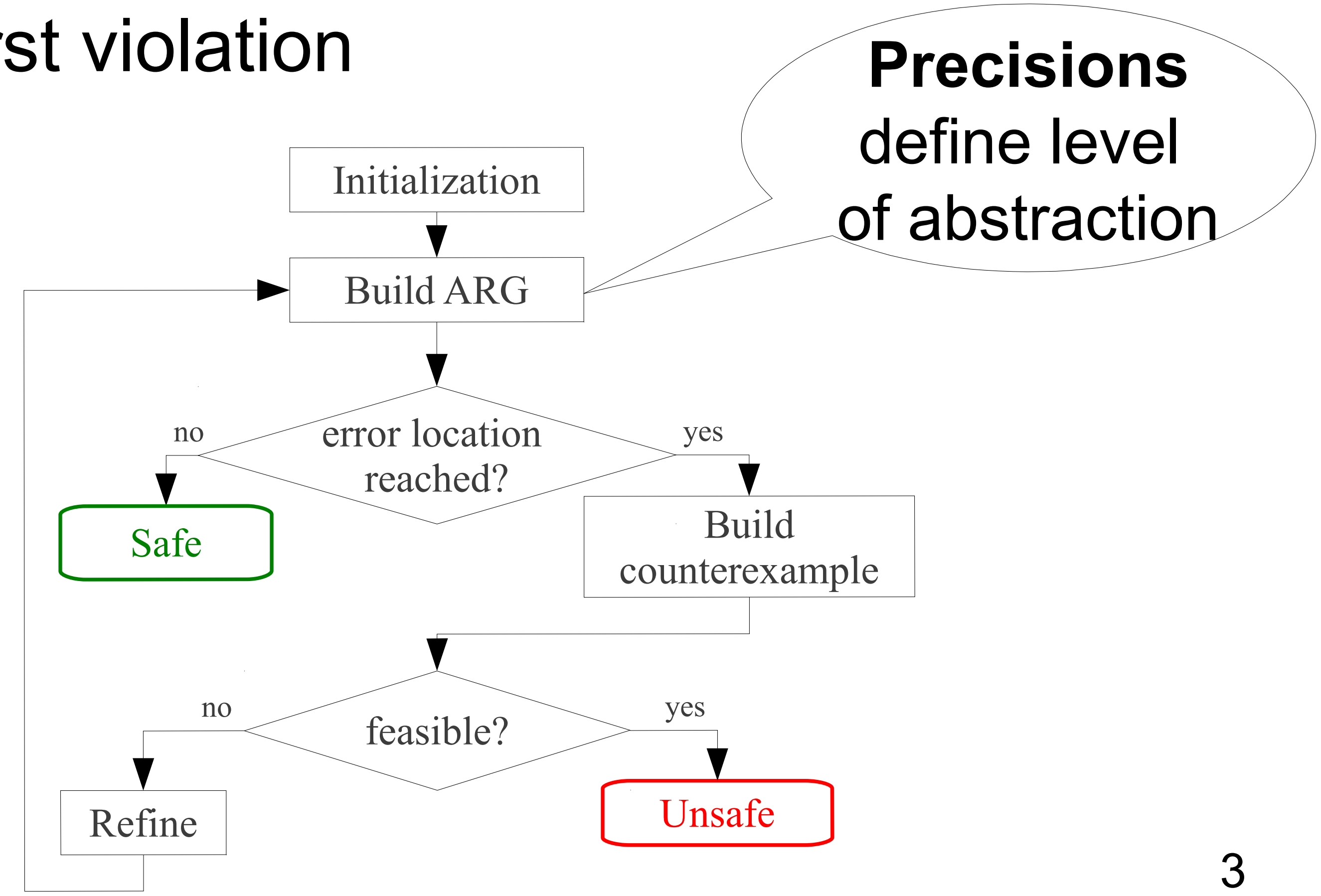
Program source code

```
...
if(offset > size) {
  ERROR: error();
  // error location
} ...
```

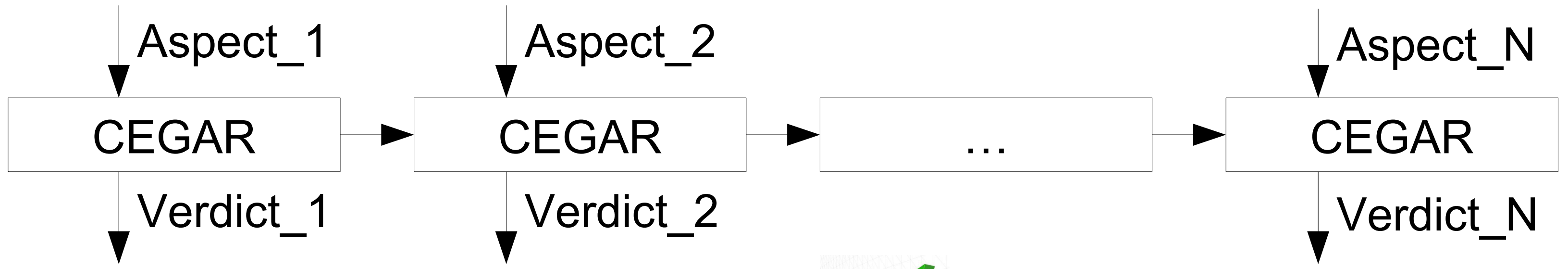
Counterexample Guided Abstraction Refinement (CEGAR)

- Checks 1 aspect
- Finds the first violation

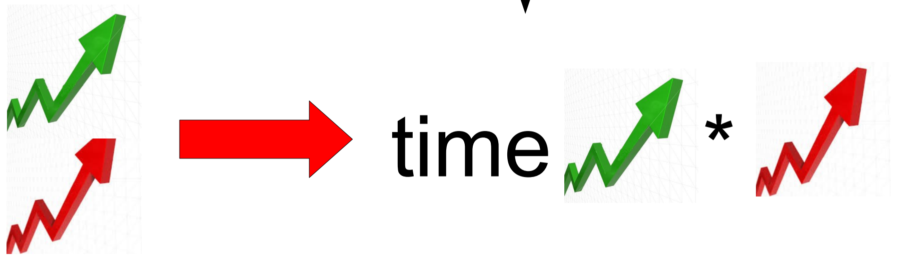
Verdict
- Safe
- Unsafe
- Unknown



Sequential CEGAR



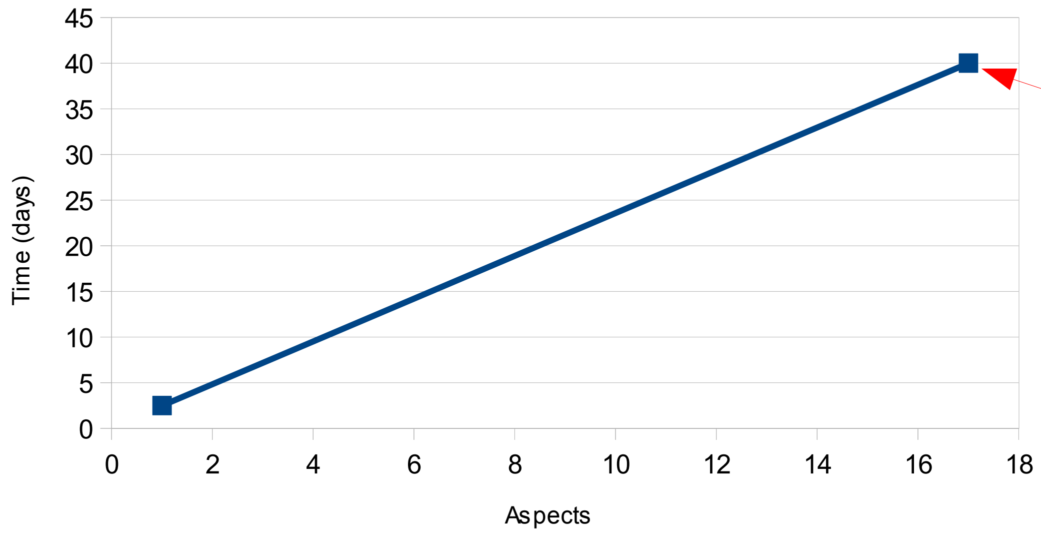
- Number of programs
- Number of aspects



- Verification facts after each step are lost
  - Abstract states, precision, etc.
- **Large amount** of resources are wasted

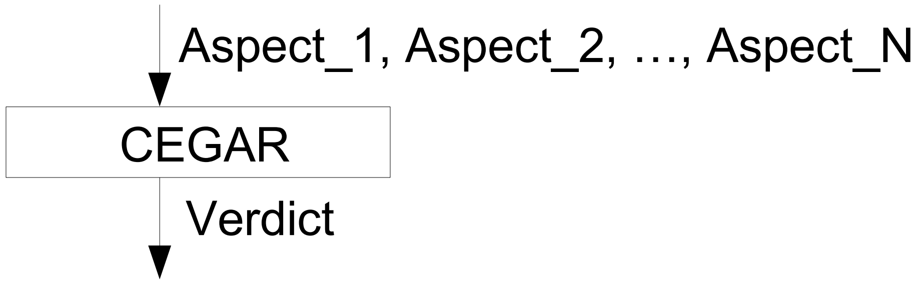
Linux Driver Verification Tools

- More than 50 aspects to check
- Much more are expected



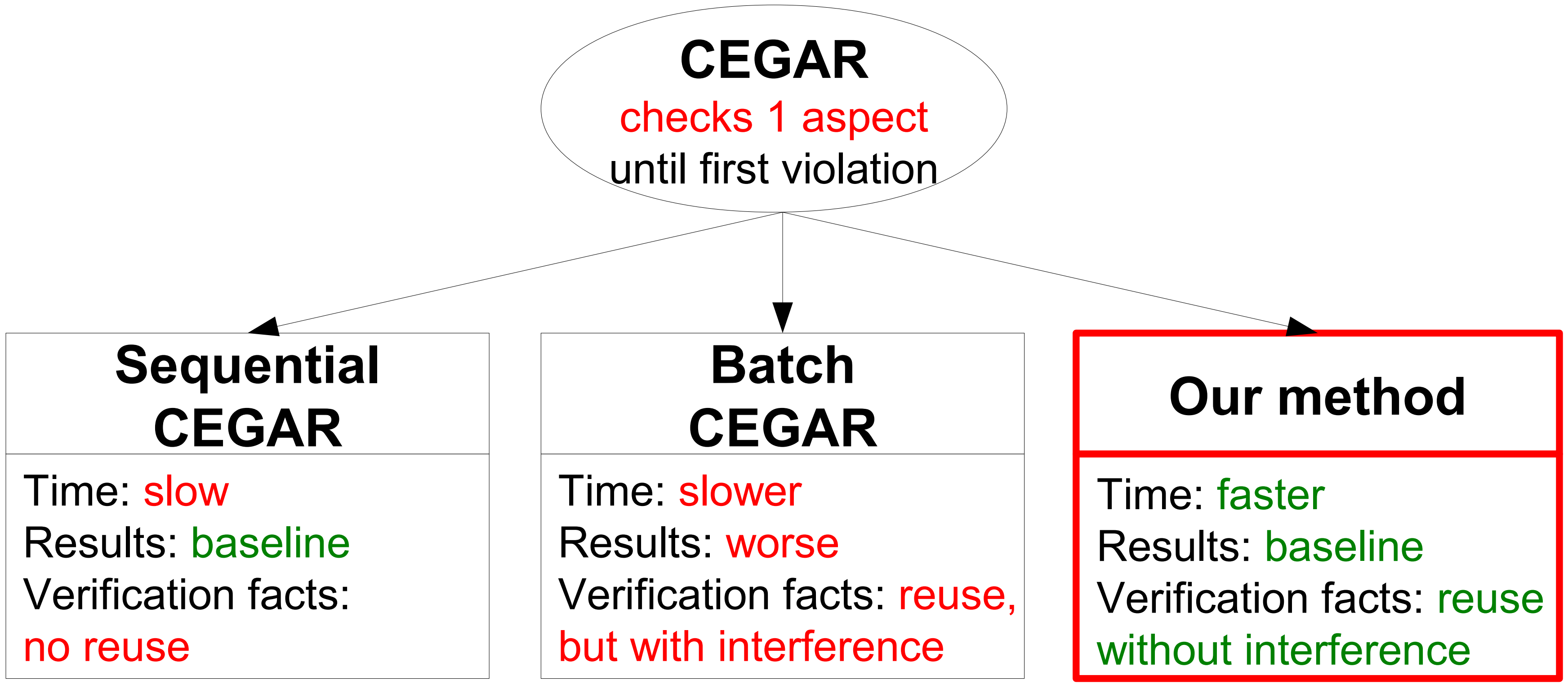
It takes 40 days to check all Linux kernel modules against only 17 aspects

Batch CEGAR



- Verification facts for different aspects may interfere with each other
- CEGAR stops after finding a bug
- Some aspect may exhaust all resources
- Verification tasks are more complex
  - More than 30% of all bugs are lost

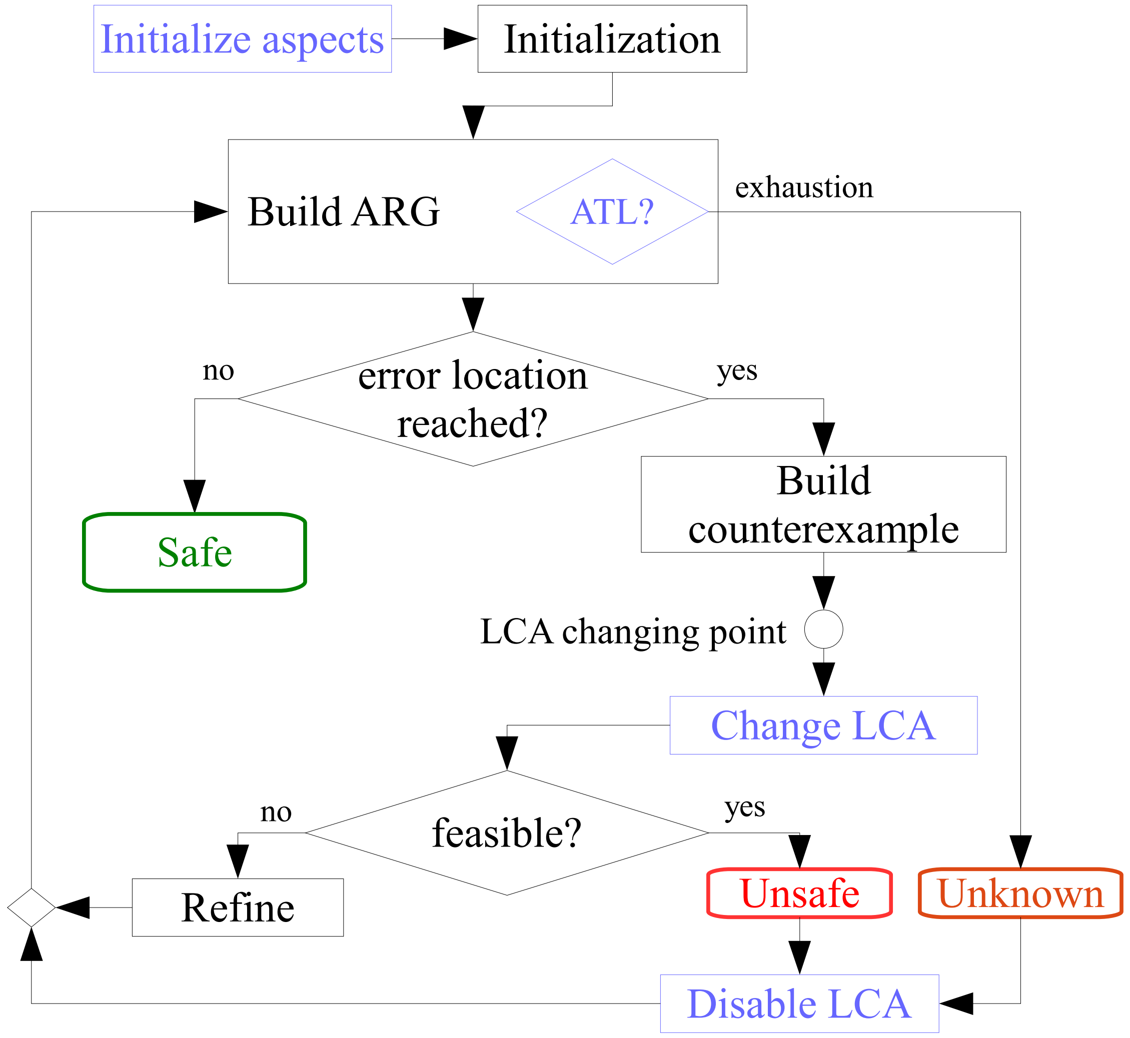
Checking Several Aspects



Multi-Aspect Verification (MAV)

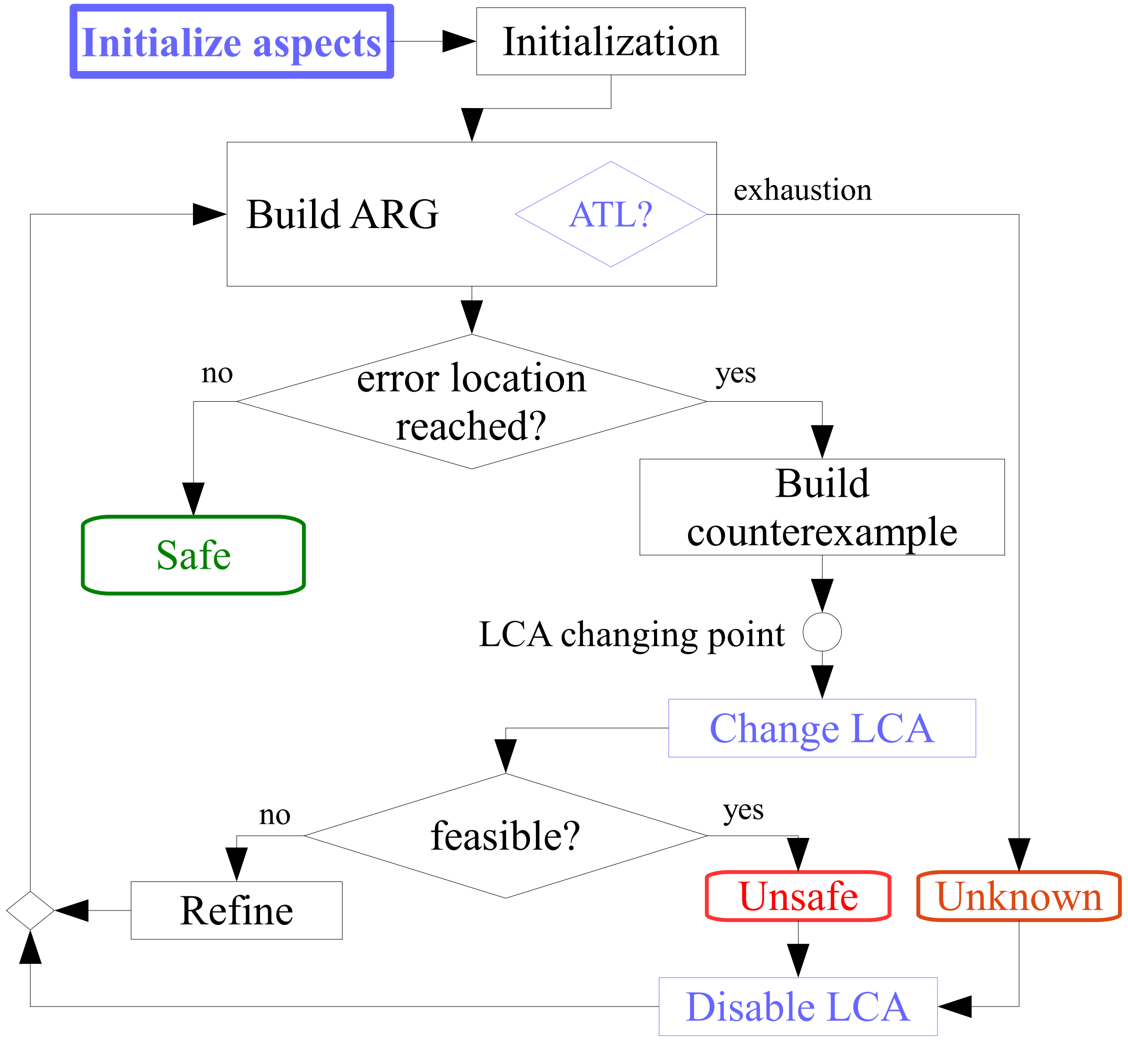
- We propose
  - A new CEGAR-based method
  - Checking several aspects at once
- Main requirements
  - Get the same results as Sequential CEGAR
  - Reduce verification time

The MAV Algorithm



Initialize Aspects Step

- Aspects → error locations

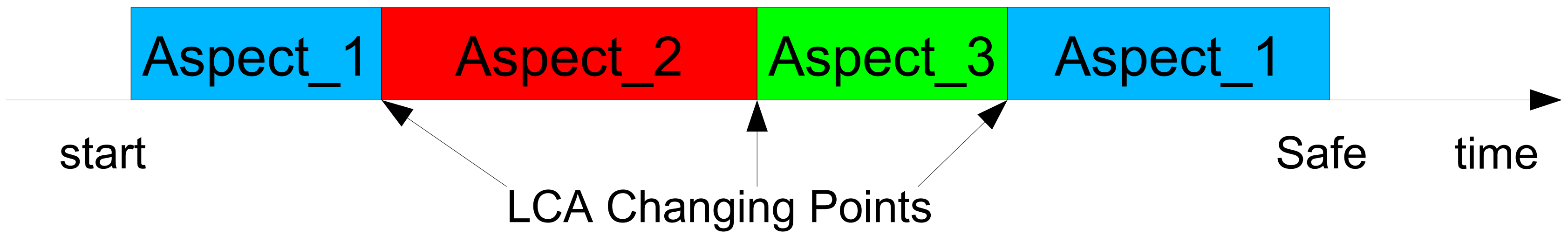


Aspects Representation in MAV

- Error location (e.g., label *ERROR_ID*)
- Verdict (Safe, Unsafe, Unknown, Checking)
- Consumed time
    - to set time limit
- Corresponding verification facts
    - Can be reused by other aspects

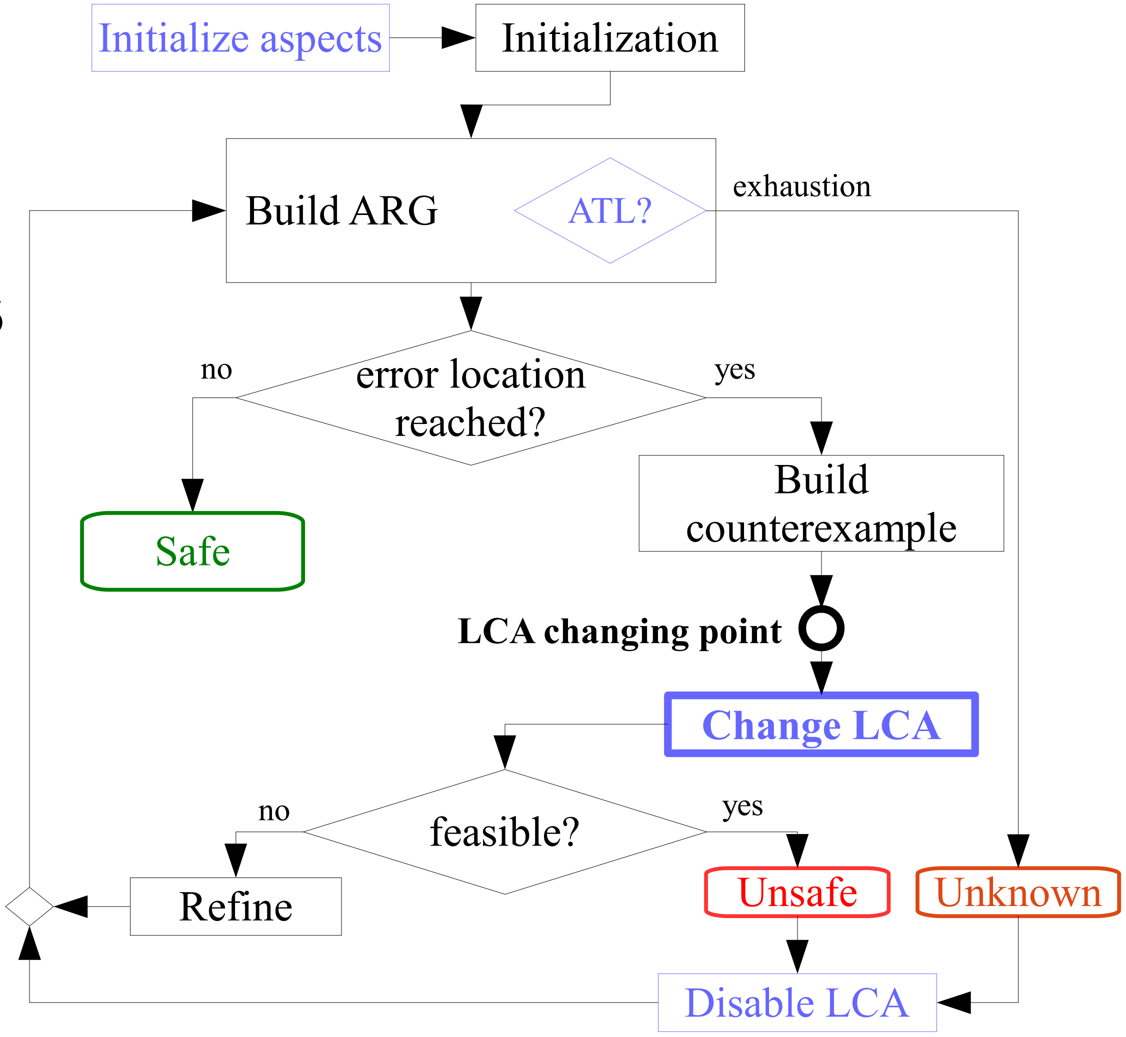
Latest Checked Aspect (LCA)

- **Approximation**: only one aspect is checked at a time
- Time line is divided by LCA Changing Points



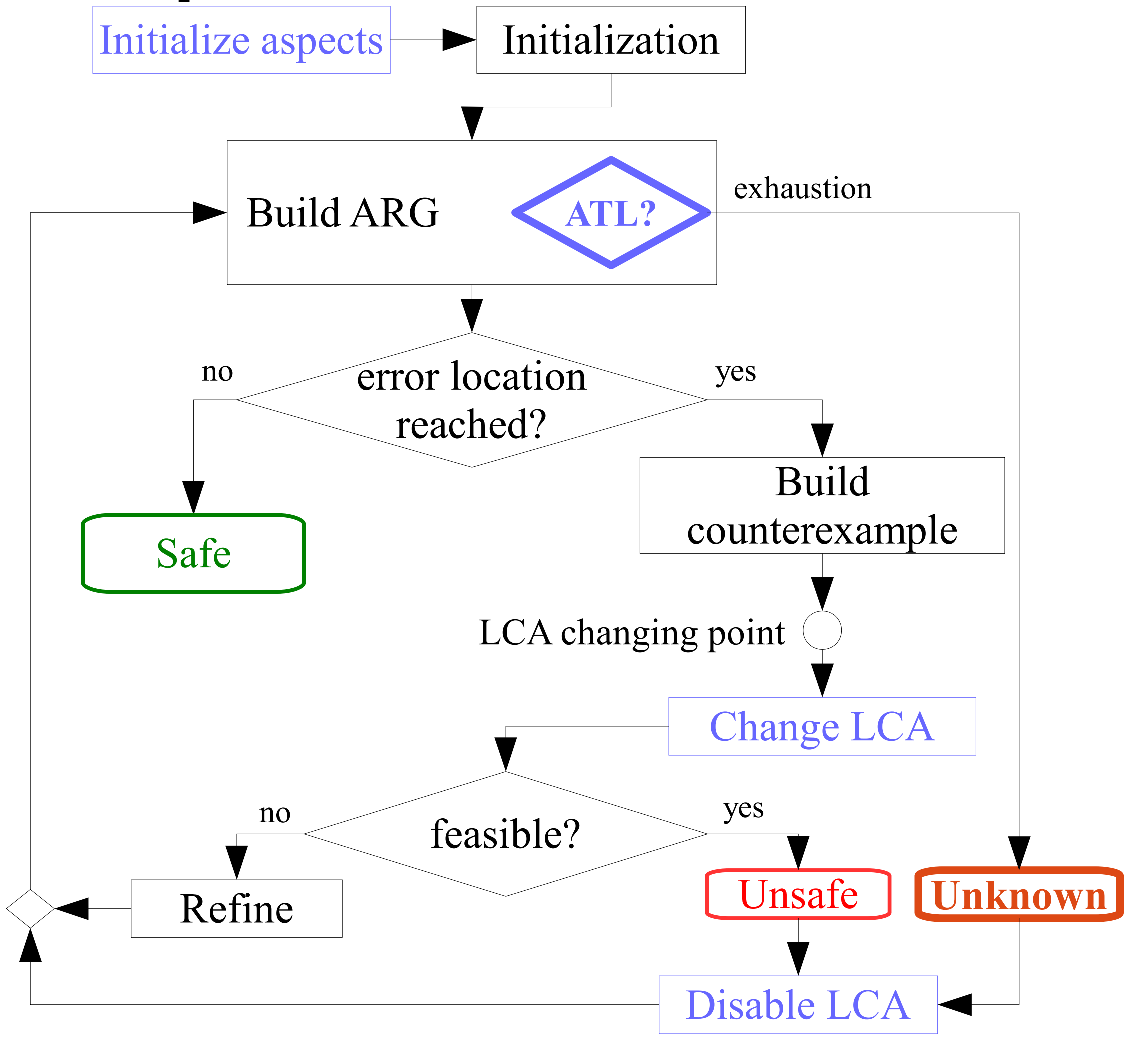
Change LCA Step

- Add time and verification facts to aspect



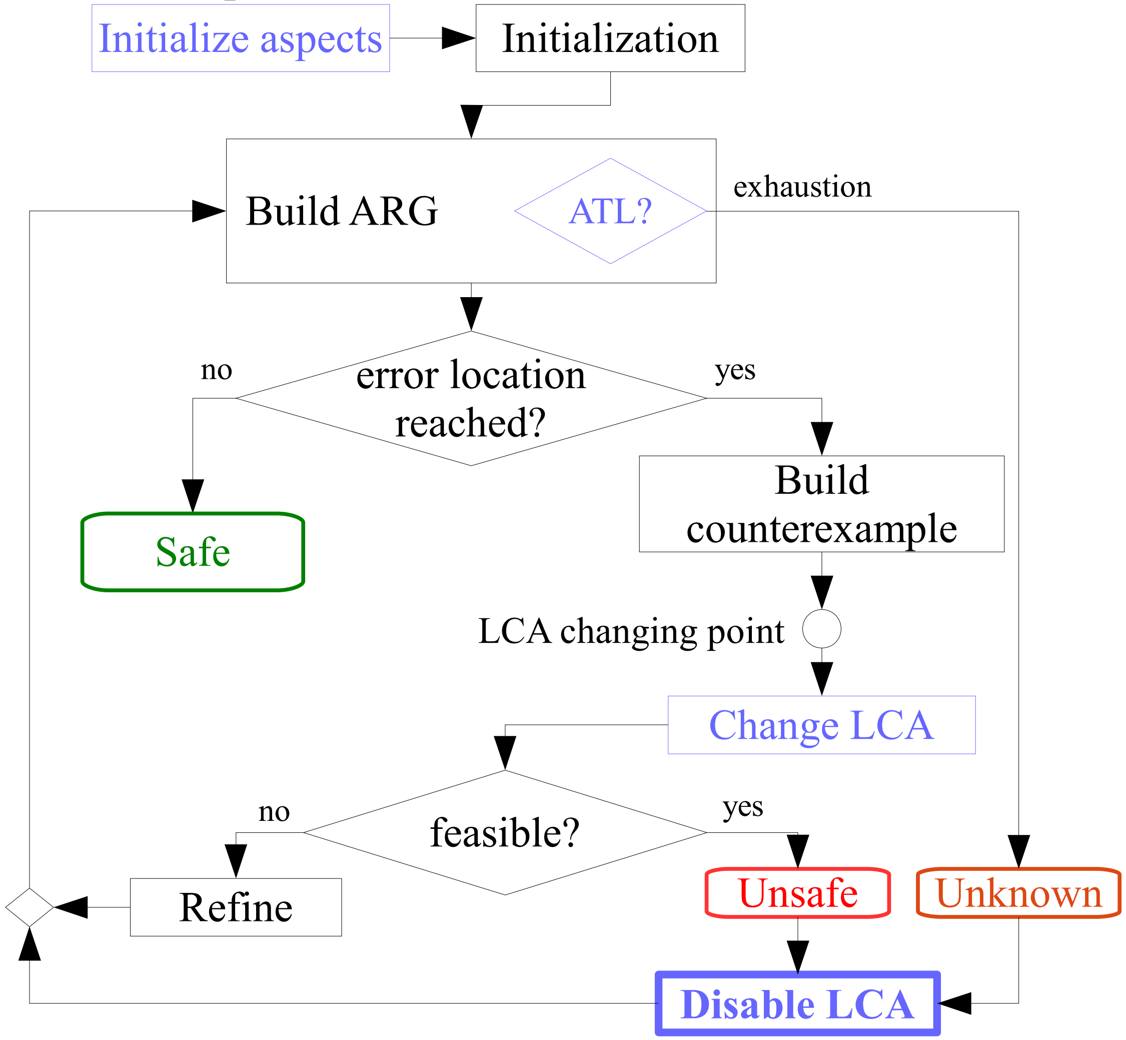
Check ATL Step

- **A**spect
  Time
  Limit



Disable LCA Step

- Continue after Unsafe and Unknown
- Without LCA



Disable Aspect

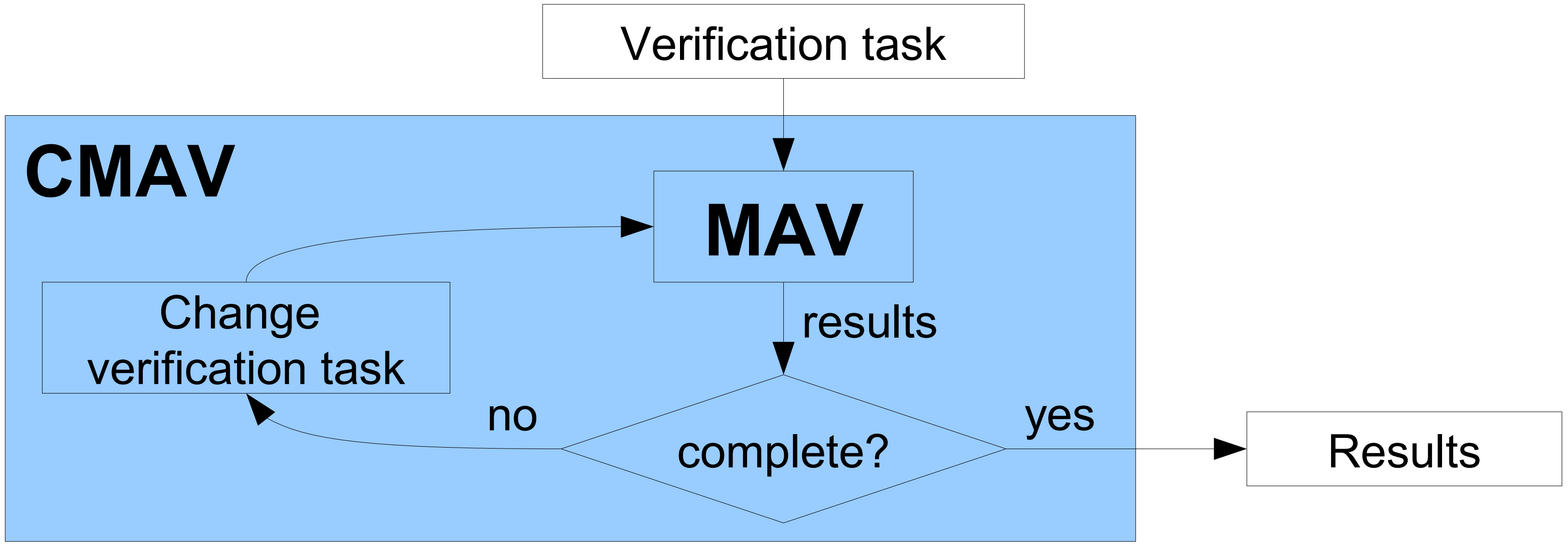
- Stop checking error location
- Remove all relevant verification facts
  - to prevent interference with other aspects

Abnormal Termination

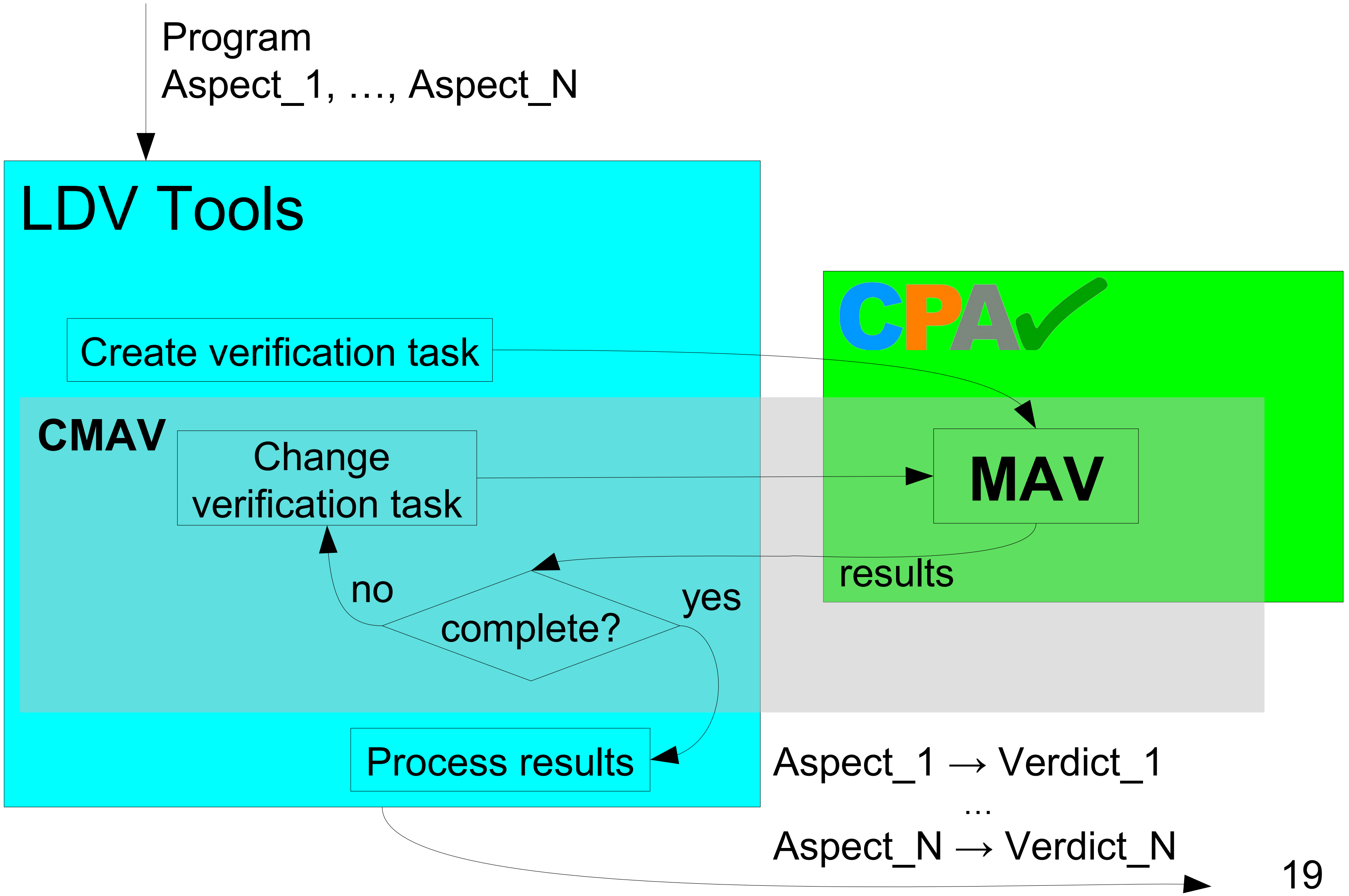
- Memory exhaustion, bugs in verifiers
- All aspects get *checking* verdict
- ➢ MAV cannot meet the requirements

Conditional **MAV** (CMAV)

- Based on Conditional Model Checking
- Launch MAV several times
- All aspects get verdicts

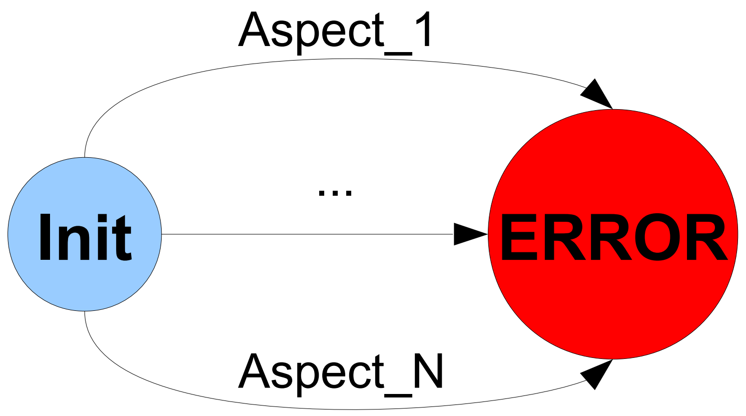


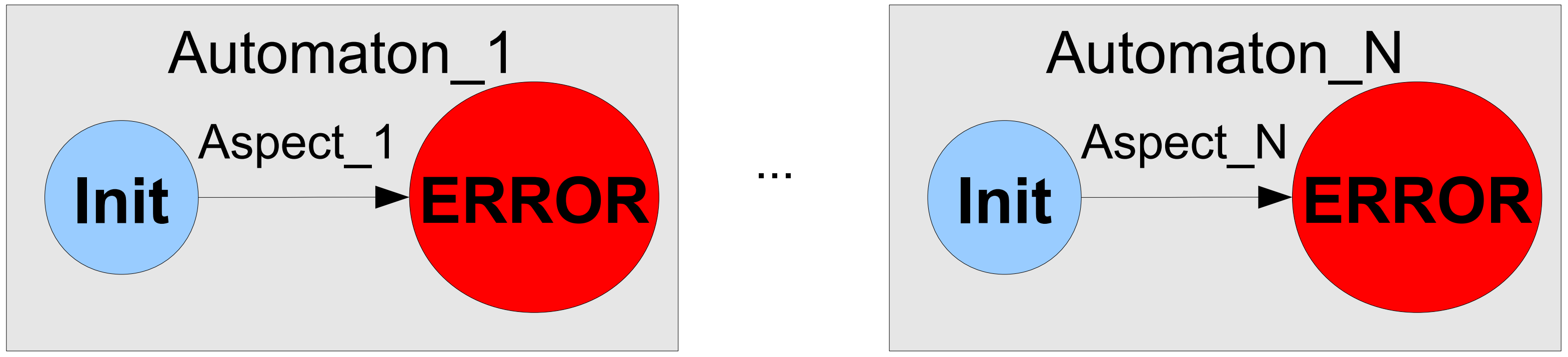
CMAV Implementation



Error Location Representation

- One automaton
  - Efficient
  - Simple automata
- Automaton composition
  - Any automata





Internal Time Limits

- Aspect Time Limit (ATL)
    - ~CEGAR time limit for aspects
- Idle Interval Time Limit (IITL)
    - Limits intervals after the Disable LCA step
    - Starts next iteration of CMAV
    - Without disabled aspect

Optional Internal Time Limits

- Observations
  - Long LCA Intervals → Unknown
  - Long LCA Intervals → aspects interference
- Basic Interval Time Limit (BITL)
  - Limits LCA intervals
- First Interval Time Limit (FITL)
  - Limits the first LCA interval

Verification Facts

- We consider
  - Abstraction precisions
  - Abstract states
- Change LCA
  - Track precision for LCA
- Disable LCA
  - Clean LCA precision

Verification Facts Reuse

- Abstract states
  - Full reuse inside CMAV iteration
  - Can be cleaned by starting the next iteration
    - Idle Interval Time Limit
- Abstraction precision
  - Full reuse until the Disable LCA step
  - Can be cleaned at the Disable LCA step
    - Different cleaning strategies

Precision Reuse Effects

- Positive precision reuse effect
    - Reduces time (regression verification)
    - It is better to keep precision
- Negative precision reuse effect
    - Increases time (interference)
    - It is better to remove precision
- How to determine balance?

Suggested Cleaning Strategies

- None
    - Do not clean at all
- Waitlist/Subtraction
    - Subtract LCA precision in waitlist
- Waitlist/Clear
    - Clear waitlist precision
- ARG/Subtraction
    - Subtract LCA precision in ARG
- ALL
    - Clear ARG precision

Cleaning Strategies Comparison

1000 verification tasks, Linux kernel 3.16-rc1, 17 aspects

	Strategy	Time (hours)	Unsafes	Launches per task
CMAV	None	140	534	2.084
CMAV	**Waitlist/Subtraction**	**118**	**610**	**1.984**
CMAV	Waitlist/Clear	127	599	1.580
CMAV	ARG/Subtraction	122	590	2.005
CMAV	All	120	605	1.451
	Sequential CEGAR	389	604	17

Different strategies represent different balance between positive and negative precision reuse effects

Sequential CEGAR vs CMAV

- All Linux kernel modules of version 4.0
- 6021 verification tasks, 17 aspects, 102 357 verdicts

Sequential CEGAR	CMAV
• Time limit: 900s	• ATL: 900s
• Memory limit: 15Gb	• Memory limit: 15Gb
	• Iteration time limit: 1200s
	• IITL: 20s
	• FITL: 100s
	• BITL: 100s
	• Waitlist/Subtraction

Sequential CEGAR vs CMAV

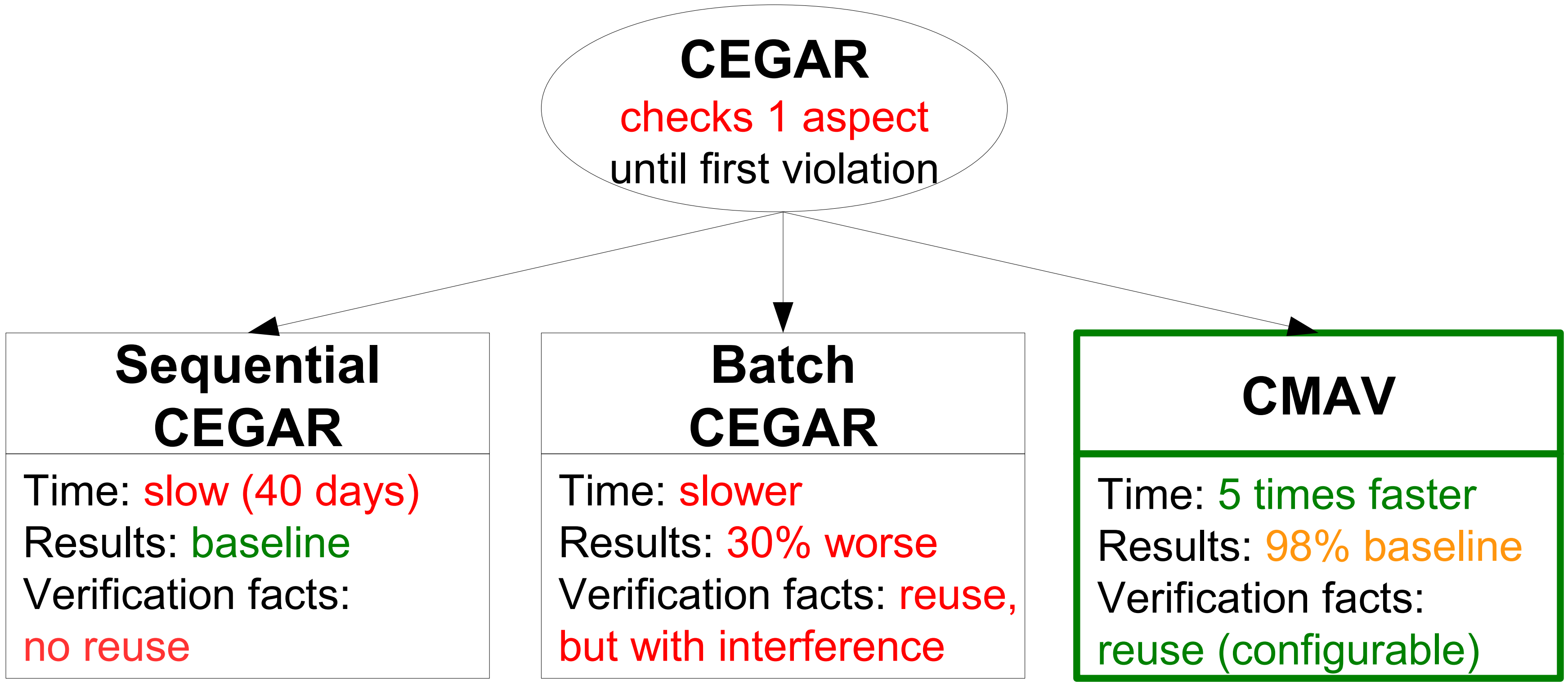
Algorithm	Safe		Unsafe		Unknown	Time
Sequential CEGAR	98 651		623		3 083	860 hours (40 days)
CMAV	96 654	-2195 (2.15%) +198 (0.20%)	624	-22 (0.02%) +23 (0.02%)	5 079	200 hours (8 days)

- CMAV **5** times faster (CPU time)
- Same results: ~97.61%
- Negative transitions: ~2.17%
  - Safe/Unsafe in CEGAR → Unknown in CMAV
- Positive transitions: ~0.22%
  - Unknown in CEGAR → Safe/Unsafe in CMAV

Sequential CEGAR vs CMAV

- CEGAR: always 17 iterations (17 aspects)
- CMAV:
  - Total: 8395 iterations / 6021 tasks
  - Average: ~1.39 iterations
  - First iteration: 5511 tasks (~92%)

Conclusion

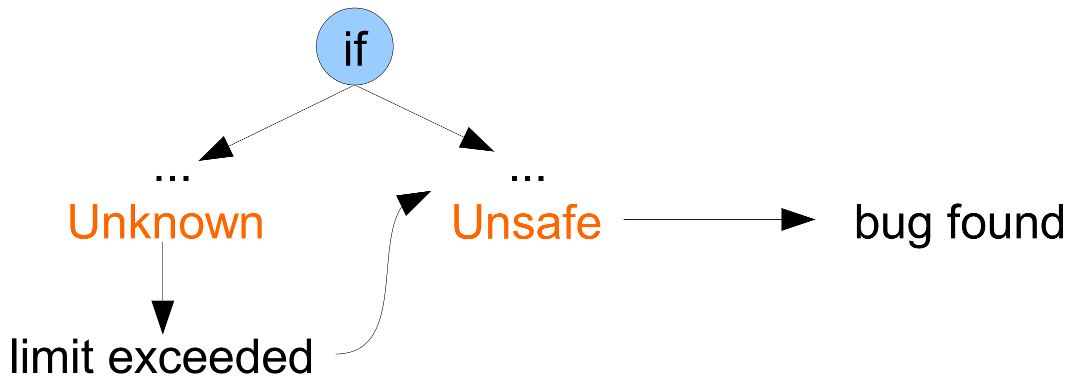

CEGAR
checks 1 aspect
until first violation
Sequential CEGAR
Time: slow (40 days)
Results: baseline
Verification facts: no reuse
Batch CEGAR
Time: slower
Results: 30% worse
Verification facts: reuse, but with interference
CMAV
Time: 5 times faster
Results: 98% baseline
Verification facts: reuse (configurable)

Future Plans

- Integration with Multiple Error Analysis
    - Find all violations of all aspects
- Complex automata for aspects
- MAV and bug finding
- Checking of hundreds of aspects
- …

MAV and Bug Finding (Idea)

- Skip checking of some complex part of ARG
    - Additional Internal Time Limits
        - Limit time for CFA nodes?
        - Limit time for Build ARG step?
        - Limit time for “if” branches?
        - …



Hundreds of Aspects

- Determine dependence between time and number of aspects
- Divide aspects into groups?
    - For minimal interferences between aspects
- Check groups in parallel?

LinuxTesting.org

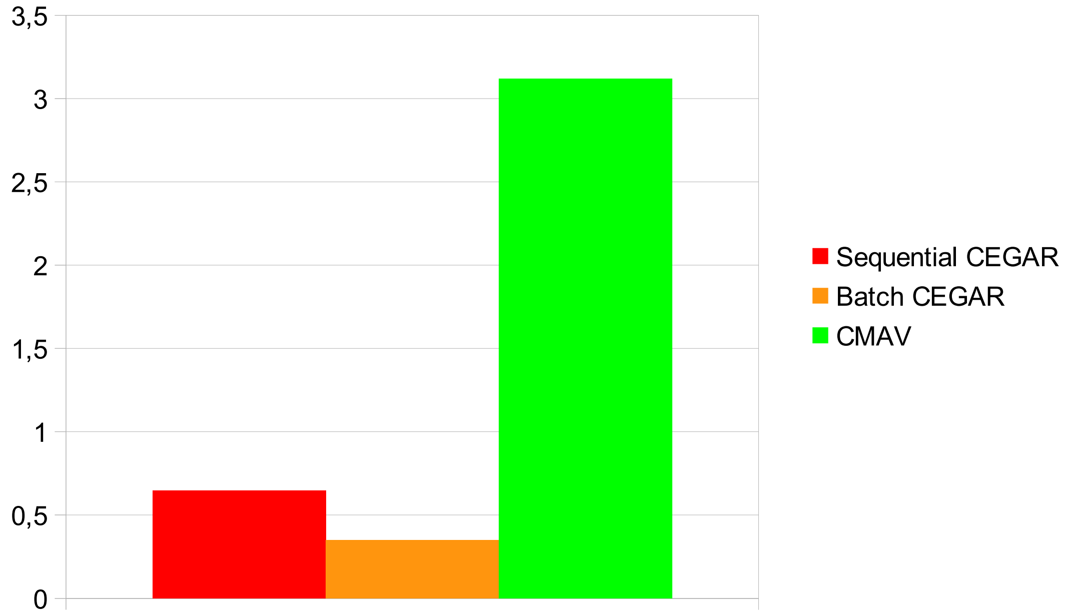
Thank you

Vitaly Mordan
mordan@ispras.ru

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

CEGAR vs CMAV (Efficiency)

- Potential bugs to time (hours) ratio
- Same results in parallel


Sequential CEGAR
Batch CEGAR
CMAV