The joint 4rd International Workshop on CPAchecker (CPA'19) and 9th Linux Driver Verification (LDV) Workshop
October 2, 2019, Frauenchiemsee, Germany

Static Verification Results Visualization in the Context of SV-COMP

Vitaly Mordan
mordan@ispras.ru

ISP RAS LinuxTesting.org

Ivannikov Institute for System Programming of the Russian Academy of Sciences

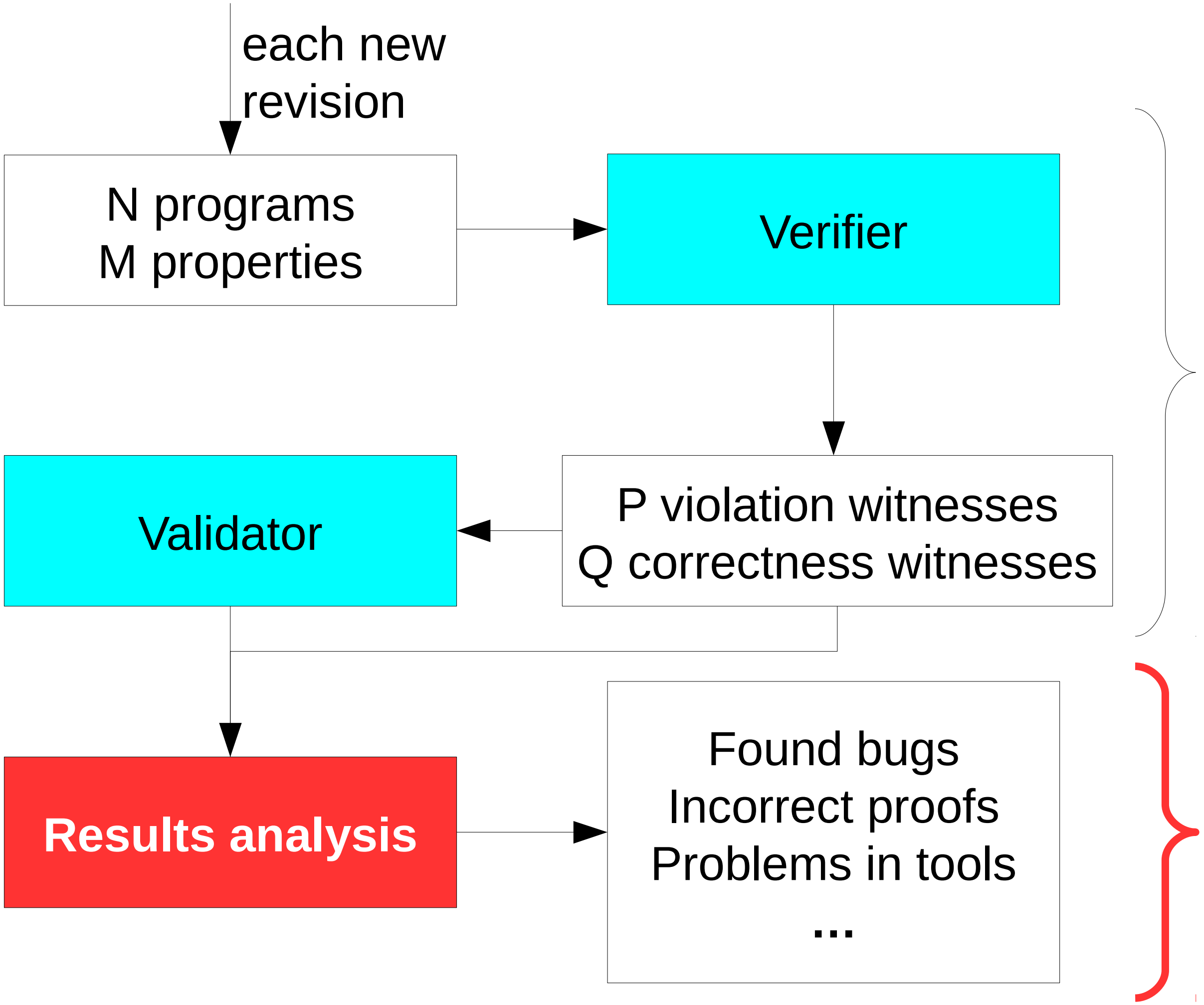
Static Verification State of the Art

- More than 31 tools*
- Improvements in effectiveness and efficiency*
- Validation of verification results*
    - Both property violations and correctness proofs*
- Different properties*
    - Potential for extensions (e.g., property automata)
- Verification of C and Java programs*

➡ What about results analysis?

* D. Beyer. *Automatic Verification of C and Java Programs: SV-COMP 2019*.

Verification of Industry System



Automatic step:
required resources **can** be minimized with modern approaches and cloud technologies

Manual step:
required resources are **harder** to reduce (user experience), but cost much more

Related Work

- BenchExec* table-generator
  - Score (based on tasks definition)
  - Plots with consumed resources
  - Comparison tables
  - Witness validation results
  - Witnesses are not visualized
- LDV Tools (Klever)**
  - Preset environment models for Linux/BusyBox/etc.
  - Violation witnesses visualization
    - Specific format

Presented in machine-readable format

Not supported by SV-COMP tools

➡ **Cannot** visualize generic witness from SV-COMP tools

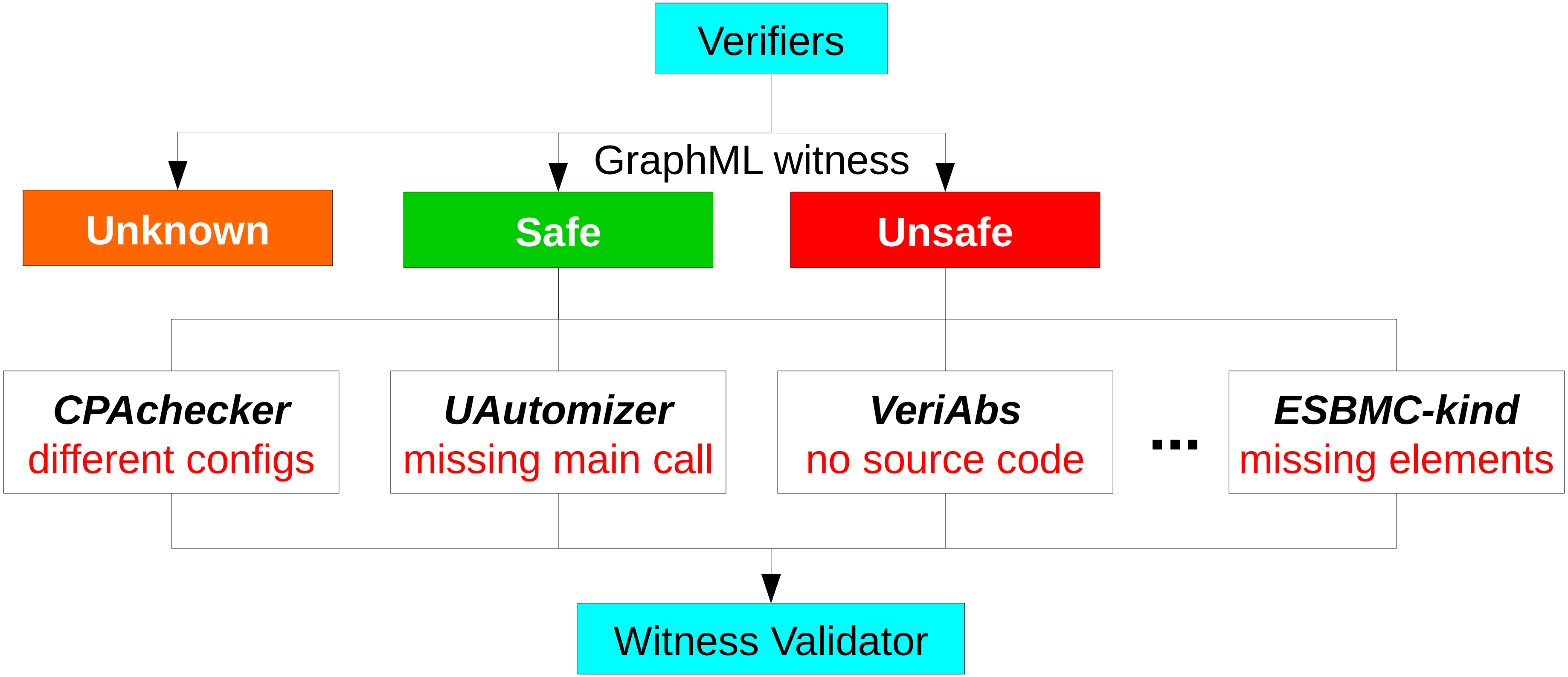
[*] https://github.com/sosy-lab/benchexec

[**] https://forge.ispras.ru/projects/klever

Suggested Solutions

- Witness Visualizer (user-friendly witnesses)
    - Helps to locate bugs for the users
    - Helps to reveal problems in tools
- Correctness witnesses visualization (idea)
    - Shows main proof hints (for developers)
    - Presents source code coverage (for users)
- Benchmark Visualizer (continuous verification)
    - Visualizes BenchExec results
    - Groups witnesses for each benchmark

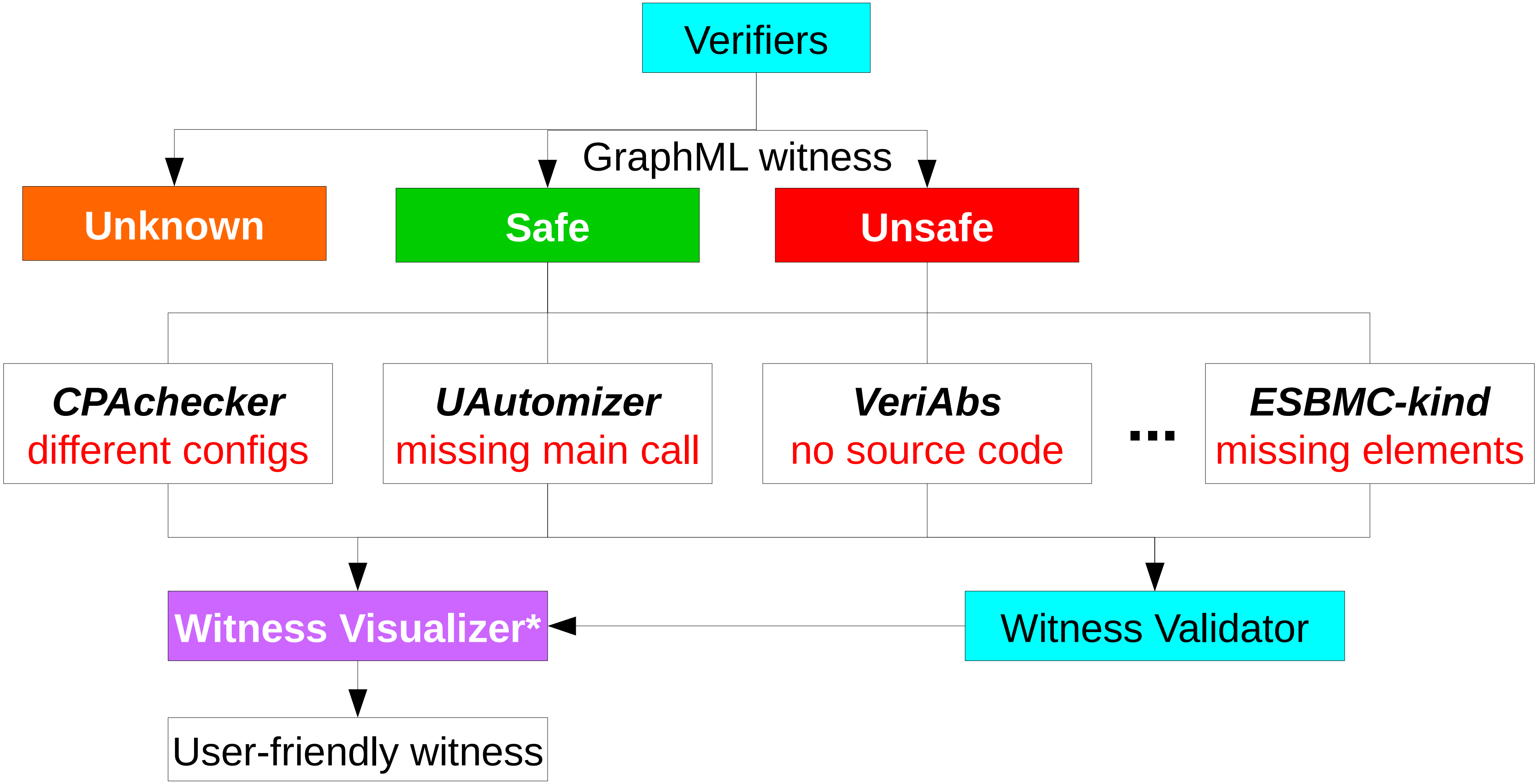
Common Witness Format*



- Machine-readable format
- There are still differences among tools

* D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, A. Stahlbauer. *Witness validation and stepwise testification across software verifiers*. ACM, 2015.

Witness Visualizer



Verifiers

GraphML witness

Unknown

Safe

Unsafe

CPAchecker
different configs

UAutomizer
missing main call

VeriAbs
no source code

...

ESBMC-kind
missing elements

Witness Visualizer*

Witness Validator

User-friendly witness

* https://github.com/ispras/cv

Requirements to the Witness Visualizer

- Fault tolerance (to the missing elements)
    - Support common witness format (GraphML)
- Quality control (for developers)
    - Provide feedback on the missing elements
- Support violation hints
    - Helps with large witnesses
- Provide operations with witnesses
    - Comparison
- Support both violation and correctness types

Fault Tolerance

- Cannot be tolerated
    - Parsing failures (wrong format)
    - Empty witnesses
- Restorable missing elements
    - Source code (program file + line/offset)
    - Entry point (based on property description)
    - Property violation (last edge)
- Elements, which cannot be restored
    - Call stack
    - Assumptions/controls

Quality Control

- Provide useful feedback to the developers
  - Source files do not exist
  - Call stack is missing
  - Conditions are missing
  - Entry point is missing
  - Produced warnings during visualization

1) No call stack (enterFunction tag)
2) No conditions (control tag)

Warning: some elements are missing

Violation Hints

- Core elements, which describe the given violation
- Reason – visualize large witnesses
  - Highlight violation hints
    - With call stack, source code link, thread id, etc.
  - Hide other elements
- Violation hints extraction
  - From witnesses (“note”, “warning”)

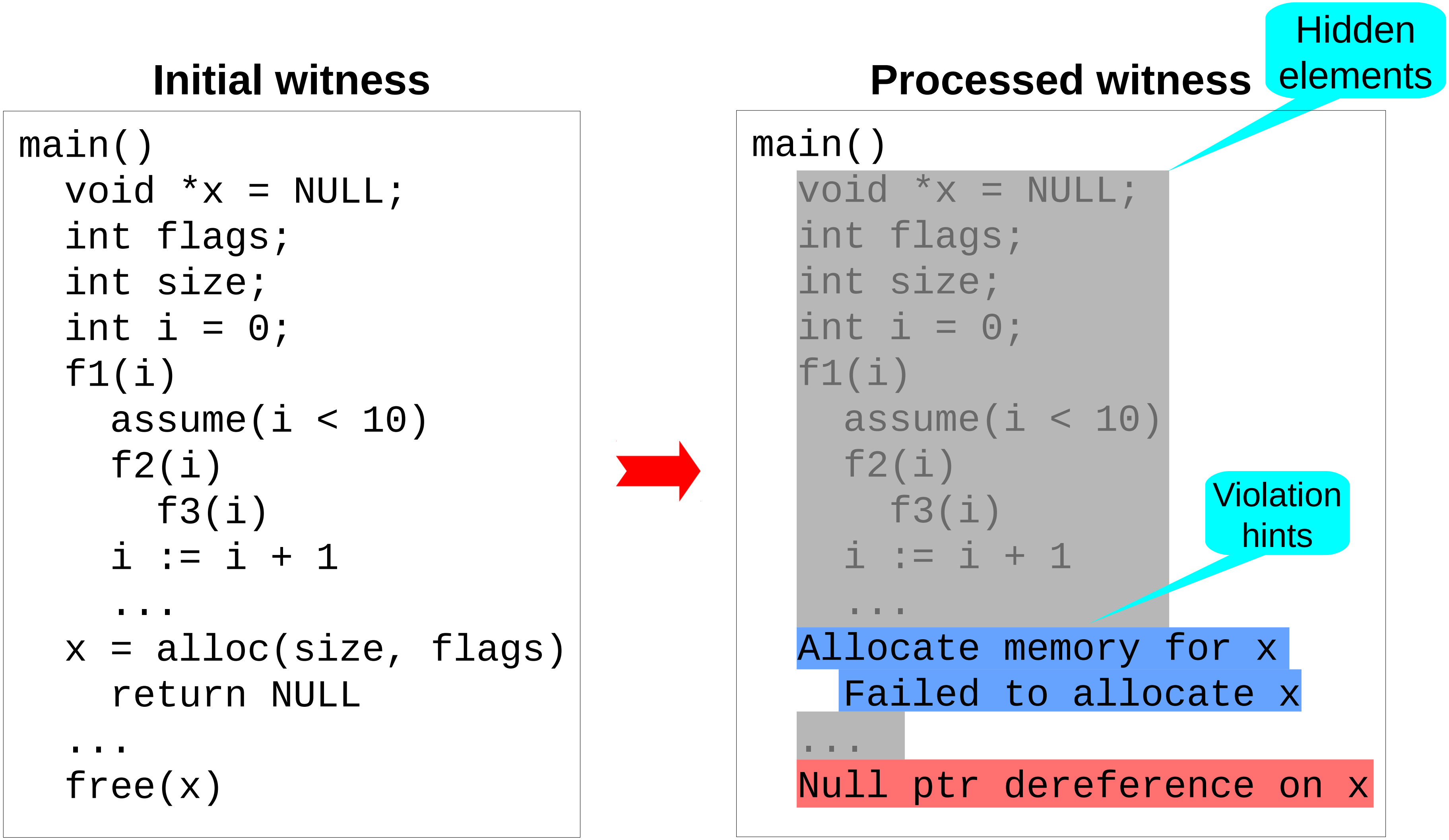
    <data key="note">Acquire mutex_lock</data>*
  - From property
  - From source code**

```
OBSERVER AUTOMATON A
...
  MATCH {func($?)} -> ...
...
```

* Example is based on witnesses from *CPA-Lockator* tool.

** Based on model comments (applied in LDV Tools).

Violation Hints Usage Example



Initial witness

```
main()
  void *x = NULL;
  int flags;
  int size;
  int i = 0;
  f1(i)
    assume(i < 10)
    f2(i)
      f3(i)
    i := i + 1
    ...
  x = alloc(size, flags)
    return NULL
  ...
  free(x)
```

Processed witness

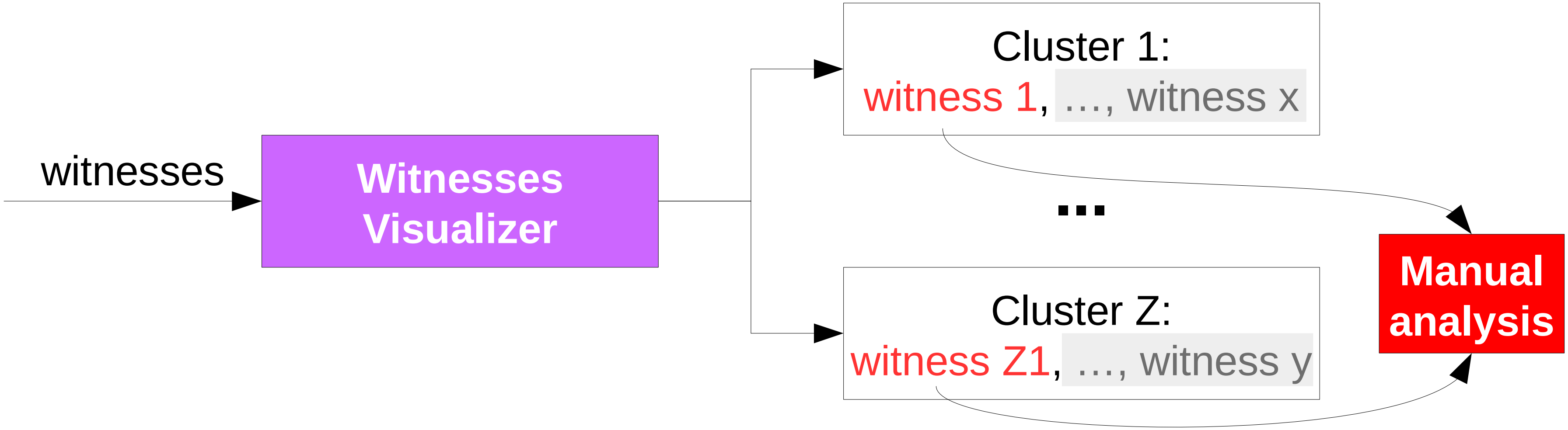
```
main()
  void *x = NULL;
  int flags;
  int size;
  int i = 0;
  f1(i)
    assume(i < 10)
    f2(i)
      f3(i)
    i := i + 1
    ...
  Allocate memory for x
    Failed to allocate x
  ...
  Null ptr dereference on x
```

Hidden elements

Violation hints

Operations with Witnesses

- Witnesses comparison
  - Distinguish different witnesses (error paths)
  - Filter several witnesses*
  - Can be done for validated witnesses only



* For example, SV-COMP tool *CPA-Lockator* can produce several witnesses for concurrency properties.

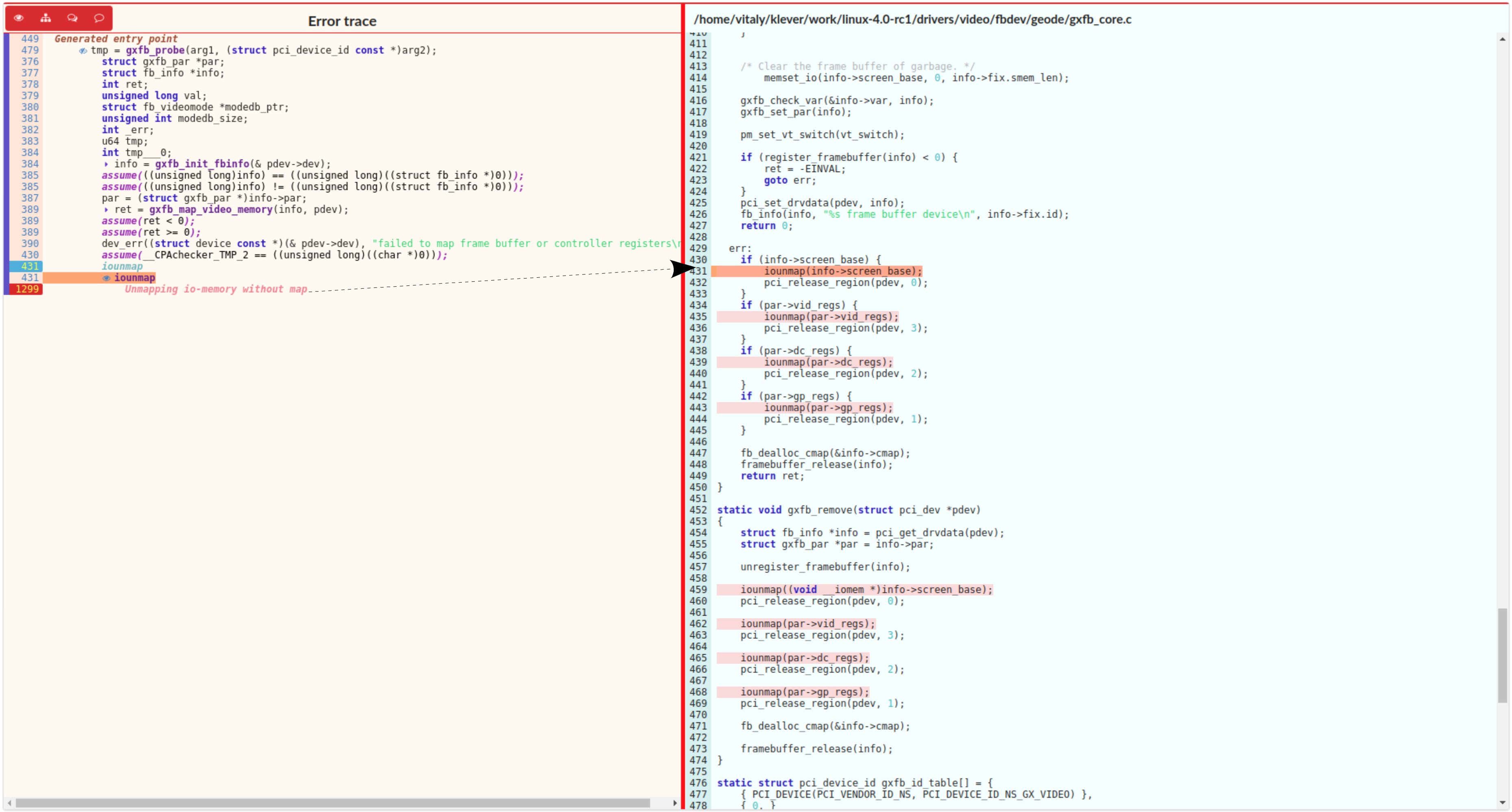
SV-COMP Tools Violation Witnesses

SV-COMP Tool	Witness elements			Source code				Violation Hints
	Call stack	Entry point	Assumptions/controls	String	Offset	Line number	File name	
2LS	-	-	+	-	-	+	+	-
AProVE	-	+	-	+	-	+	+	-
CBMC	-	-	+	-	-	+	+	-
CBMC-Path	-	-	+	-	-	+	+	-
CPA-BAM-BnB	+	+	+	+/-	+	+	+	+/-
CPA-Lockator	+	+	+	+/-	+	+	+	+/-
CPA-Seq	+	+	+	+/-	+	+	+	+/-
DepthK	-	+	+	-	-	+	+	-
DIVINE-explicit	+	+	-	-	-	+	+	-
DIVINE-SMT	+	+	-	-	-	+	+	-
ESBMC-kind	-	+	+	-	-	+	+	-
Lazy-CSeq	+	+	+	-	-	+	+	-
Map2Check	-	-	+	-	-	+	+	-
PeSCo	+	+	+	+/-	+	+	+	+/-
Pinaka	-	-	+	-	-	+	+	-
PredatorHP	-	-	-	-	-	+	+	-
Skink	-	-	+	-	-	+	+	-
SMACK	-	-	+	-	-	+	+	-
Symbiotic	-	-	+	-	-	+	+	-
UAutomizer	+	-	+	+	-	+	+	-
UKojak	+	-	+	+	-	+	+	-
UTaipan	+	-	+	+	-	+	+	-
VeriAbs	+	+	+	-	+	+	+	-
VeriFuzz	-	-	+	-	-	+	+	-
VIAP	+	+	-	+	-	+	+	-
Yogar-CBMC	+	+	-	-	-	+	+	-
Yogar-CBMC-Parallel	+	+	-	-	-	+	+	-

* 4 verifiers for Java programs were excluded from this comparison, because they do not produce witnesses.

Example of a Witness with Violation Hints

- Input/output memory map operations: `ioremap`, `pci_ioremap_bar`, …
- Input/output memory unmap operation: `iounmap`



Error trace

```
Generated entry point
    tmp = gxfb_probe(arg1, (struct pci_device_id const *)arg2);
        struct gxfb_par *par;
        struct fb_info *info;
        int ret;
        unsigned long val;
        struct fb_videomode *modedb_ptr;
        unsigned int modedb_size;
        int _err;
        u64 tmp;
        int tmp___0;
        info = gxfb_init_fbinfo(& pdev->dev);
        assume(((unsigned long)info) == ((unsigned long)((struct fb_info *)0)));
        assume(((unsigned long)info) != ((unsigned long)((struct fb_info *)0)));
        par = (struct gxfb_par *)info->par;
        ret = gxfb_map_video_memory(info, pdev);
        assume(ret < 0);
        assume(ret >= 0);
        dev_err((struct device const *)(& pdev->dev), "failed to map frame buffer or controller registers\r
        assume(__CPAchecker_TMP_2 == ((unsigned long)((char *)0)));
        iounmap
        iounmap
            Unmapping io-memory without map
```

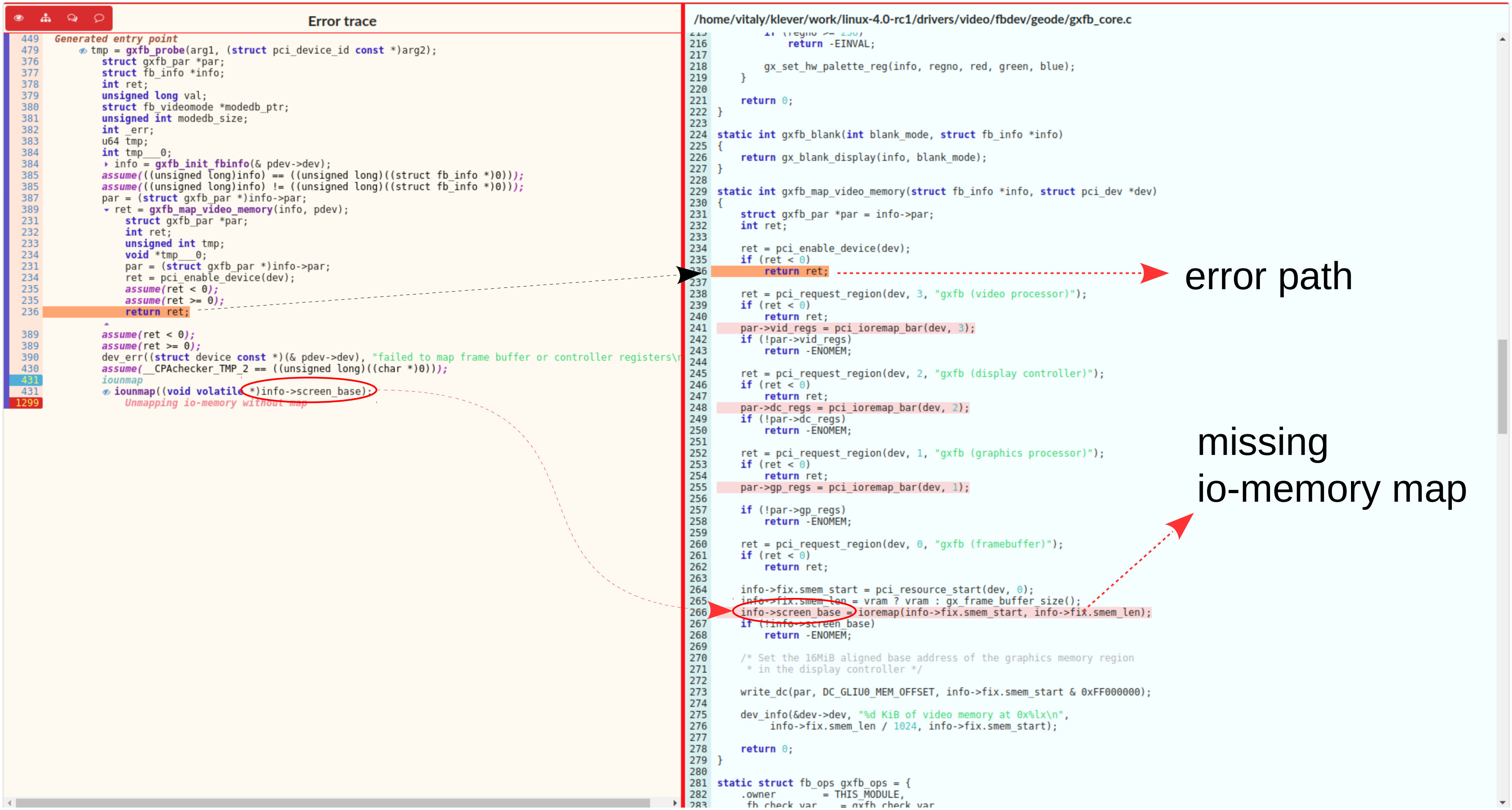
/home/vitaly/klever/work/linux-4.0-rc1/drivers/video/fbdev/geode/gxfb_core.c

```
    }


    /* Clear the frame buffer of garbage. */
        memset_io(info->screen_base, 0, info->fix.smem_len);

    gxfb_check_var(&info->var, info);
    gxfb_set_par(info);

    pm_set_vt_switch(vt_switch);

    if (register_framebuffer(info) < 0) {
        ret = -EINVAL;
        goto err;
    }
    pci_set_drvdata(pdev, info);
    fb_info(info, "%s frame buffer device\n", info->fix.id);
    return 0;

  err:
    if (info->screen_base) {
        iounmap(info->screen_base);
        pci_release_region(pdev, 0);
    }
    if (par->vid_regs) {
        iounmap(par->vid_regs);
        pci_release_region(pdev, 3);
    }
    if (par->dc_regs) {
        iounmap(par->dc_regs);
        pci_release_region(pdev, 2);
    }
    if (par->gp_regs) {
        iounmap(par->gp_regs);
        pci_release_region(pdev, 1);
    }

    fb_dealloc_cmap(&info->cmap);
    framebuffer_release(info);
    return ret;
}

static void gxfb_remove(struct pci_dev *pdev)
{
    struct fb_info *info = pci_get_drvdata(pdev);
    struct gxfb_par *par = info->par;

    unregister_framebuffer(info);

    iounmap((void __iomem *)info->screen_base);
    pci_release_region(pdev, 0);

    iounmap(par->vid_regs);
    pci_release_region(pdev, 3);

    iounmap(par->dc_regs);
    pci_release_region(pdev, 2);

    iounmap(par->gp_regs);
    pci_release_region(pdev, 1);

    fb_dealloc_cmap(&info->cmap);

    framebuffer_release(info);
}

static struct pci_device_id gxfb_id_table[] = {
    { PCI_DEVICE(PCI_VENDOR_ID_NS, PCI_DEVICE_ID_NS_GX_VIDEO) },
    { 0, }
```

* Violation witness visualization is based on LDV Tools (Klever): https://forge.ispras.ru/projects/klever

Example of a Witness with Violation Hints

- Input/output memory map operations: `ioremap`, `pci_ioremap_bar`, ...
- Input/output memory unmap operation: `iounmap`



Error trace

```
449  Generated entry point
479      tmp = gxfb_probe(arg1, (struct pci_device_id const *)arg2);
376          struct gxfb_par *par;
377          struct fb_info *info;
378          int ret;
379          unsigned long val;
380          struct fb_videomode *modedb_ptr;
381          unsigned int modedb_size;
382          int _err;
383          u64 tmp;
384          int tmp___0;
384          info = gxfb_init_fbinfo(& pdev->dev);
385          assume(((unsigned long)info) == ((unsigned long)((struct fb_info *)0)));
385          assume(((unsigned long)info) != ((unsigned long)((struct fb_info *)0)));
387          par = (struct gxfb_par *)info->par;
389          ret = gxfb_map_video_memory(info, pdev);
231              struct gxfb_par *par;
232              int ret;
233              unsigned int tmp;
234              void *tmp___0;
231              par = (struct gxfb_par *)info->par;
234              ret = pci_enable_device(dev);
235              assume(ret < 0);
235              assume(ret >= 0);
236              return ret;

389          assume(ret < 0);
389          assume(ret >= 0);
390          dev_err((struct device const *)(& pdev->dev), "failed to map frame buffer or controller registers\r
430          assume(__CPAchecker_TMP_2 == ((unsigned long)((char *)0)));
431          iounmap
431          iounmap((void volatile *)info->screen_base);
1299             Unmapping io-memory without map
```

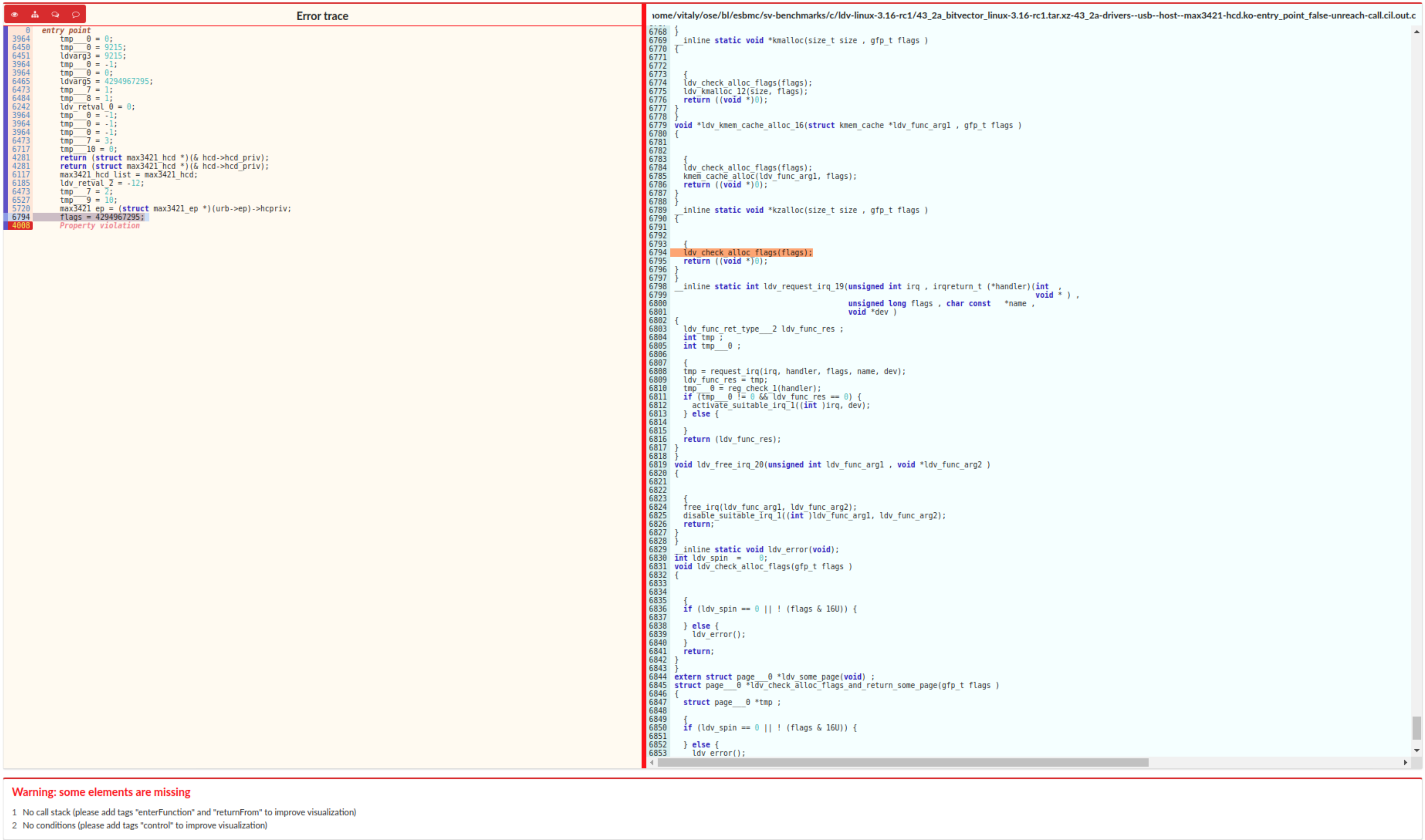
/home/vitaly/klever/work/linux-4.0-rc1/drivers/video/fbdev/geode/gxfb_core.c

```
		if (regno >= 256)
			return -EINVAL;

		gx_set_hw_palette_reg(info, regno, red, green, blue);
	}

	return 0;
}

static int gxfb_blank(int blank_mode, struct fb_info *info)
{
	return gx_blank_display(info, blank_mode);
}

static int gxfb_map_video_memory(struct fb_info *info, struct pci_dev *dev)
{
	struct gxfb_par *par = info->par;
	int ret;

	ret = pci_enable_device(dev);
	if (ret < 0)
		return ret;

	ret = pci_request_region(dev, 3, "gxfb (video processor)");
	if (ret < 0)
		return ret;
	par->vid_regs = pci_ioremap_bar(dev, 3);
	if (!par->vid_regs)
		return -ENOMEM;

	ret = pci_request_region(dev, 2, "gxfb (display controller)");
	if (ret < 0)
		return ret;
	par->dc_regs = pci_ioremap_bar(dev, 2);
	if (!par->dc_regs)
		return -ENOMEM;

	ret = pci_request_region(dev, 1, "gxfb (graphics processor)");
	if (ret < 0)
		return ret;
	par->gp_regs = pci_ioremap_bar(dev, 1);

	if (!par->gp_regs)
		return -ENOMEM;

	ret = pci_request_region(dev, 0, "gxfb (framebuffer)");
	if (ret < 0)
		return ret;

	info->fix.smem_start = pci_resource_start(dev, 0);
	info->fix.smem_len = vram ? vram : gx_frame_buffer_size();
	info->screen_base = ioremap(info->fix.smem_start, info->fix.smem_len);
	if (!info->screen_base)
		return -ENOMEM;

	/* Set the 16MiB aligned base address of the graphics memory region
	 * in the display controller */

	write_dc(par, DC_GLIU0_MEM_OFFSET, info->fix.smem_start & 0xFF000000);

	dev_info(&dev->dev, "%d KiB of video memory at 0x%lx\n",
		 info->fix.smem_len / 1024, info->fix.smem_start);

	return 0;
}

static struct fb_ops gxfb_ops = {
	.owner		= THIS_MODULE,
	.fb_check_var	= gxfb_check_var,
```

error path

missing io-memory map

Example of a Witness with Missing Elements



Error trace

```
   0   entry point
3964       tmp___0 = 0;
6450       tmp___0 = 9215;
6451       ldvarg3 = 9215;
3964       tmp___0 = -1;
3964       tmp___0 = 0;
6465       ldvarg5 = 4294967295;
6473       tmp___7 = 1;
6484       tmp___8 = 1;
6242       ldv_retval_0 = 0;
3964       tmp___0 = -1;
3964       tmp___0 = -1;
3964       tmp___0 = -1;
6473       tmp___7 = 3;
6717       tmp___10 = 0;
4281       return (struct max3421_hcd *)(& hcd->hcd_priv);
4281       return (struct max3421_hcd *)(& hcd->hcd_priv);
6117       max3421_hcd_list = max3421_hcd;
6185       ldv_retval_2 = -12;
6473       tmp___7 = 2;
6527       tmp___9 = 10;
5720       max3421_ep = (struct max3421_ep *)(urb->ep)->hcpriv;
6794       flags = 4294967295;
4008       Property violation
```

ıome/vitaly/ose/bl/esbmc/sv-benchmarks/c/ldv-linux-3.16-rc1/43_2a_bitvector_linux-3.16-rc1.tar.xz-43_2a-drivers--usb--host--max3421-hcd.ko-entry_point_false-unreach-call.cil.out.c

```
}
__inline static void *kmalloc(size_t size , gfp_t flags )
{


  {
  ldv_check_alloc_flags(flags);
  ldv_kmalloc_12(size, flags);
  return ((void *)0);
}
}
void *ldv_kmem_cache_alloc_16(struct kmem_cache *ldv_func_arg1 , gfp_t flags )
{


  {
  ldv_check_alloc_flags(flags);
  kmem_cache_alloc(ldv_func_arg1, flags);
  return ((void *)0);
}
}
__inline static void *kzalloc(size_t size , gfp_t flags )
{


  {
  ldv_check_alloc_flags(flags);
  return ((void *)0);
}
}
__inline static int ldv_request_irq_19(unsigned int irq , irqreturn_t (*handler)(int  ,
                                                                                 void * ) ,
                                       unsigned long flags , char const   *name ,
                                       void *dev )
{
  ldv_func_ret_type___2 ldv_func_res ;
  int tmp ;
  int tmp___0 ;

  {
  tmp = request_irq(irq, handler, flags, name, dev);
  ldv_func_res = tmp;
  tmp___0 = reg_check_1(handler);
  if (tmp___0 != 0 && ldv_func_res == 0) {
    activate_suitable_irq_1((int )irq, dev);
  } else {

  }
  return (ldv_func_res);
}
}
void ldv_free_irq_20(unsigned int ldv_func_arg1 , void *ldv_func_arg2 )
{


  {
  free_irq(ldv_func_arg1, ldv_func_arg2);
  disable_suitable_irq_1((int )ldv_func_arg1, ldv_func_arg2);
  return;
}
}
__inline static void ldv_error(void);
int ldv_spin  =    0;
void ldv_check_alloc_flags(gfp_t flags )
{


  {
  if (ldv_spin == 0 || ! (flags & 16U)) {

  } else {
    ldv_error();
  }
  return;
}
}
extern struct page___0 *ldv_some_page(void) ;
struct page___0 *ldv_check_alloc_flags_and_return_some_page(gfp_t flags )
{
  struct page___0 *tmp ;

  {
  if (ldv_spin == 0 || ! (flags & 16U)) {

  } else {
    ldv_error();
```

Warning: some elements are missing

1 No call stack (please add tags "enterFunction" and "returnFrom" to improve visualization)
2 No conditions (please add tags "control" to improve visualization)

Witness was produced by ESBMC-kind tool.

Witness Visualizer Application Area

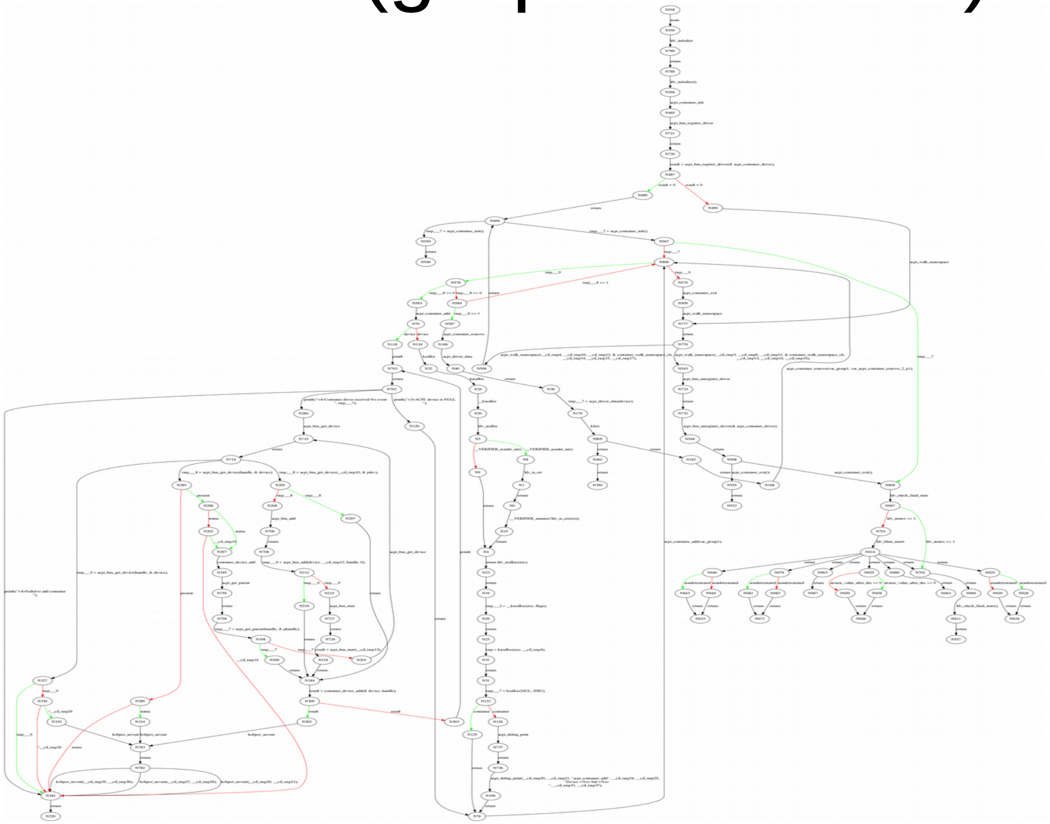
- Demonstration of a generic witness
  - Supports any SV-COMP tool
- Feedback to the developers
  - Missing elements, warnings, etc.
- Large witnesses visualization
  - Based on extracted violation hints
- Comparison of witnesses
  - Required for several witnesses

Suggested Solutions

- Witness Visualizer (user-friendly witnesses)
  - Helps to locate bugs for the users
  - Helps to reveal problems in tools
- Correctness witnesses visualization (idea)
  - Shows main proof hints (for developers)
  - Presents source code coverage (for users)
- Benchmark Visualizer (continuous verification)
  - Visualizes BenchExec results
  - Groups witnesses for each benchmark

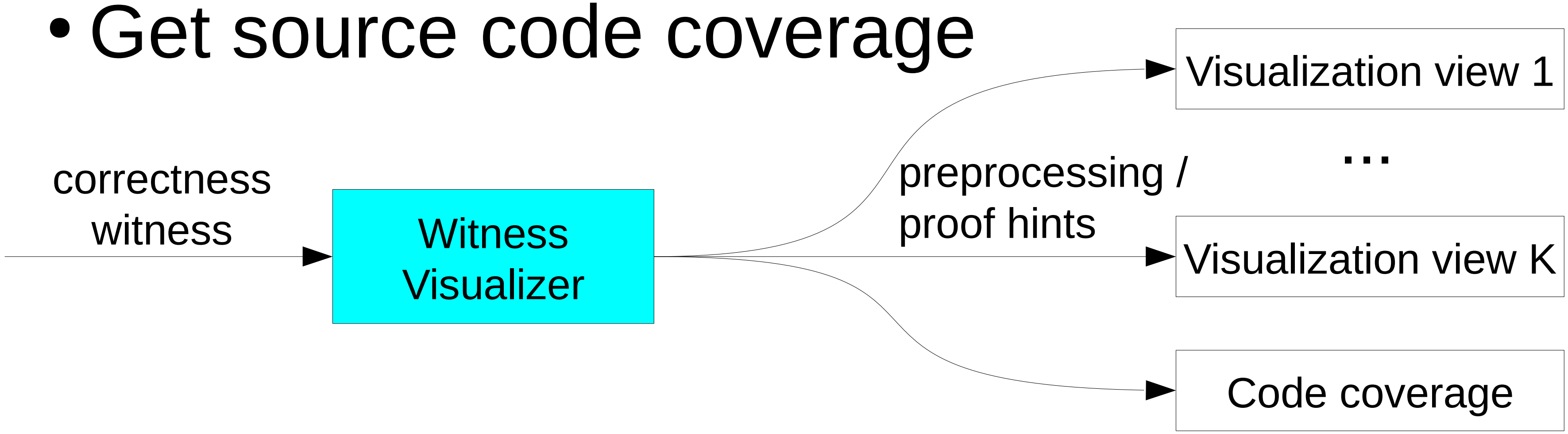
Correctness Witnesses

- Present main verification result (proof)
  - Ensure the absence of missed bugs
- Hard to visualize (graph structure)



General Ideas of the Visualization

- Support of common format* (GraphML)
- Witness preprocessing
    - Convert to the plain structure
- Extract main proof hints
    - Conditions, invariants, etc.
- Get source code coverage

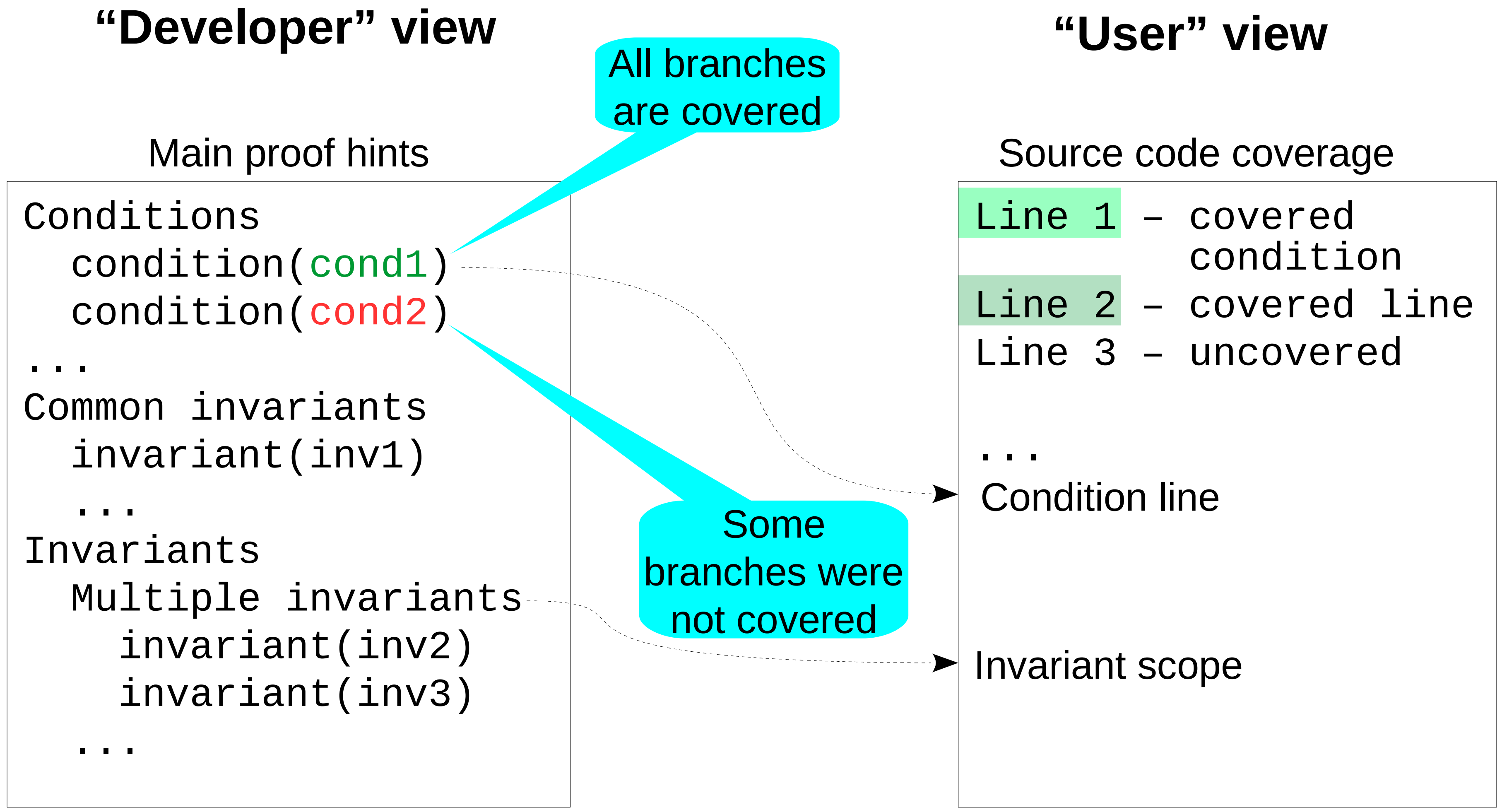


* D. Beyer, M. Dangl, D. Dietsch, M. Heizmann. *Correctness witnesses: exchanging verification results between verifiers*. ACM, 2016.

Implementation of the Suggested Ideas

- Proof hints
  - Conditions
  - Invariants (common and local)
- Witness preprocessing
  - Sort all elements by line/thread/source file
  - Combine all assumptions for conditions
  - Extract common invariants
- Witness comparison
  - Is not supported (only 1 (?) witness is expected)

Correctness Witness Model Example



“Developer” view

Main proof hints

```
Conditions
  condition(cond1)
  condition(cond2)
...
Common invariants
  invariant(inv1)
  ...
Invariants
  Multiple invariants
    invariant(inv2)
    invariant(inv3)
  ...
```

All branches are covered

Some branches were not covered

“User” view

Source code coverage

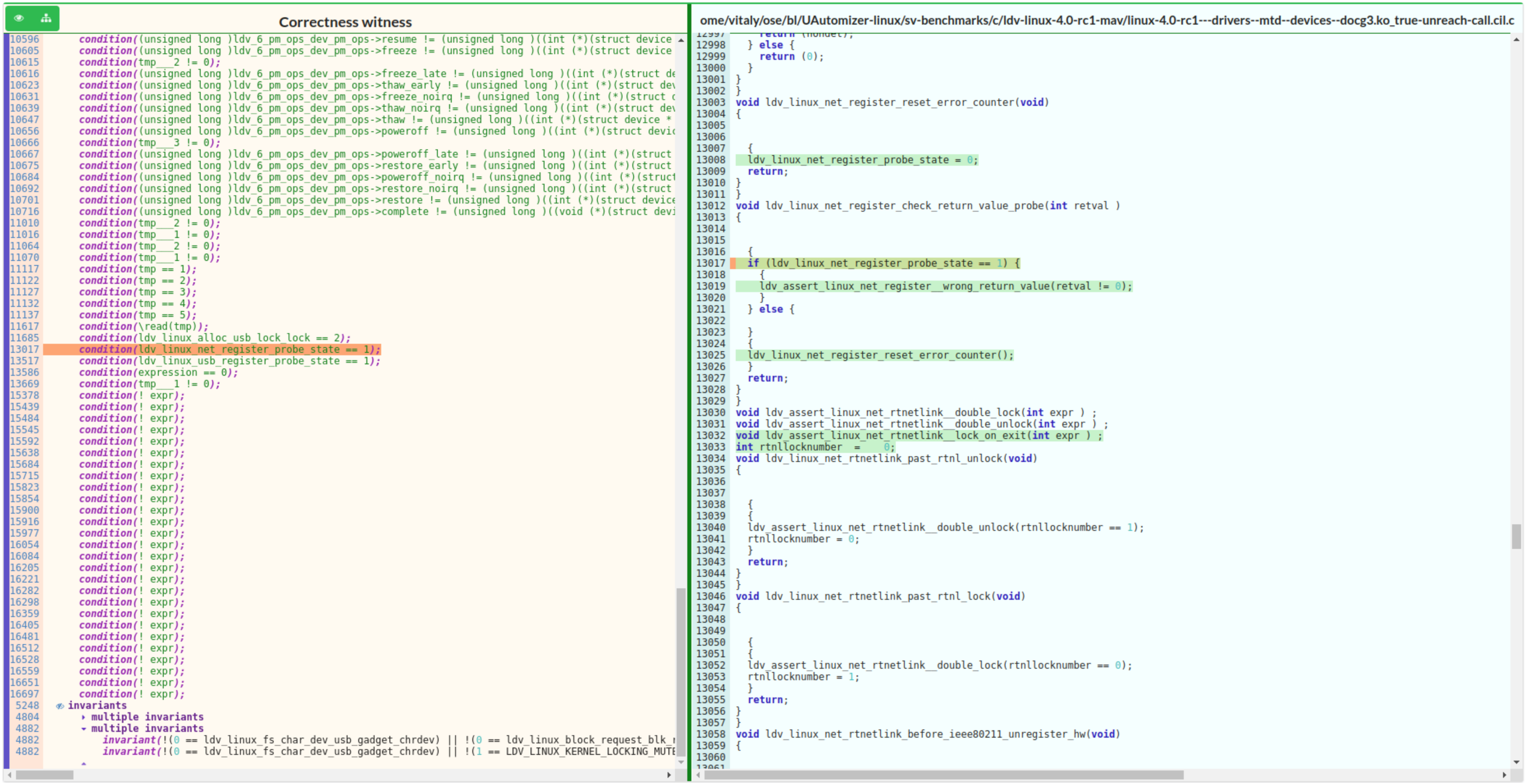
```
Line 1 – covered
         condition
Line 2 – covered line
Line 3 – uncovered

...
```

Condition line

Invariant scope

Correctness Witness Example

UAutomizer correctness witness visualization*



Correctness witness

```
10596       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->resume != (unsigned long )((int (*)(struct device
10605       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->freeze != (unsigned long )((int (*)(struct device
10615       condition(tmp___2 != 0);
10616       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->freeze_late != (unsigned long )((int (*)(struct de
10623       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->thaw_early != (unsigned long )((int (*)(struct dev
10631       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->freeze_noirq != (unsigned long )((int (*)(struct c
10639       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->thaw_noirq != (unsigned long )((int (*)(struct dev
10647       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->thaw != (unsigned long )((int (*)(struct device *
10656       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->poweroff != (unsigned long )((int (*)(struct devic
10666       condition(tmp___3 != 0);
10667       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->poweroff_late != (unsigned long )((int (*)(struct
10675       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->restore_early != (unsigned long )((int (*)(struct
10684       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->poweroff_noirq != (unsigned long )((int (*)(struct
10692       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->restore_noirq != (unsigned long )((int (*)(struct
10701       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->restore != (unsigned long )((int (*)(struct device
10716       condition((unsigned long )ldv_6_pm_ops_dev_pm_ops->complete != (unsigned long )((void (*)(struct devi
11010       condition(tmp___2 != 0);
11016       condition(tmp___1 != 0);
11064       condition(tmp___2 != 0);
11070       condition(tmp___1 != 0);
11117       condition(tmp == 1);
11122       condition(tmp == 2);
11127       condition(tmp == 3);
11132       condition(tmp == 4);
11137       condition(tmp == 5);
11617       condition(\read(tmp));
11685       condition(ldv_linux_alloc_usb_lock_lock == 2);
13017       condition(ldv_linux_net_register_probe_state == 1);
13517       condition(ldv_linux_usb_register_probe_state == 1);
13586       condition(expression == 0);
13669       condition(tmp___1 != 0);
15378       condition(! expr);
15439       condition(! expr);
15484       condition(! expr);
15545       condition(! expr);
15592       condition(! expr);
15638       condition(! expr);
15684       condition(! expr);
15715       condition(! expr);
15823       condition(! expr);
15854       condition(! expr);
15900       condition(! expr);
15916       condition(! expr);
15977       condition(! expr);
16054       condition(! expr);
16084       condition(! expr);
16205       condition(! expr);
16221       condition(! expr);
16282       condition(! expr);
16298       condition(! expr);
16359       condition(! expr);
16405       condition(! expr);
16481       condition(! expr);
16512       condition(! expr);
16528       condition(! expr);
16559       condition(! expr);
16651       condition(! expr);
16697       condition(! expr);
 5248   invariants
 4804     ▸ multiple invariants
 4882     ▾ multiple invariants
 4882         invariant(!(0 == ldv_linux_fs_char_dev_usb_gadget_chrdev) || !(0 == ldv_linux_block_request_blk_r
 4882         invariant(!(0 == ldv_linux_fs_char_dev_usb_gadget_chrdev) || !(1 == LDV_LINUX_KERNEL_LOCKING_MUTE
```

ome/vitaly/ose/bl/UAutomizer-linux/sv-benchmarks/c/ldv-linux-4.0-rc1-mav/linux-4.0-rc1---drivers--mtd--devices--docg3.ko_true-unreach-call.cil.c

```
12998   } else {
12999     return (0);
13000   }
13001 }
13002 }
13003 void ldv_linux_net_register_reset_error_counter(void)
13004 {
13005
13006
13007   {
13008   ldv_linux_net_register_probe_state = 0;
13009   return;
13010 }
13011 }
13012 void ldv_linux_net_register_check_return_value_probe(int retval )
13013 {
13014
13015
13016   {
13017   if (ldv_linux_net_register_probe_state == 1) {
13018     {
13019     ldv_assert_linux_net_register__wrong_return_value(retval != 0);
13020     }
13021   } else {
13022
13023   }
13024   {
13025   ldv_linux_net_register_reset_error_counter();
13026   }
13027   return;
13028 }
13029 }
13030 void ldv_assert_linux_net_rtnetlink__double_lock(int expr ) ;
13031 void ldv_assert_linux_net_rtnetlink__double_unlock(int expr ) ;
13032 void ldv_assert_linux_net_rtnetlink__lock_on_exit(int expr ) ;
13033 int rtnllocknumber  =    0;
13034 void ldv_linux_net_rtnetlink_past_rtnl_unlock(void)
13035 {
13036
13037
13038   {
13039   {
13040   ldv_assert_linux_net_rtnetlink__double_unlock(rtnllocknumber == 1);
13041   rtnllocknumber = 0;
13042   }
13043   return;
13044 }
13045 }
13046 void ldv_linux_net_rtnetlink_past_rtnl_lock(void)
13047 {
13048
13049
13050   {
13051   {
13052   ldv_assert_linux_net_rtnetlink__double_lock(rtnllocknumber == 0);
13053   rtnllocknumber = 1;
13054   }
13055   return;
13056 }
13057 }
13058 void ldv_linux_net_rtnetlink_before_ieee80211_unregister_hw(void)
13059 {
13060
```

* Sometimes SV-COMP tools may produce empty correctness witnesses.

Correctness Witnesses Visualization

- Suggested general ideas of the visualization
  - Based on plain structure and proof hints
- Suggested implementation of the ideas
  - Based on conditions and invariants
  - Other implementations can be suggested
- Idea to extract source code coverage
- Can be useful for both developers and users

Suggested Solutions

- Witness Visualizer (user-friendly witnesses)
  - Helps to locate bugs for the users
  - Helps to reveal problems in tools
- Correctness witnesses visualization (idea)
  - Shows main proof hints (for developers)
  - Presents source code coverage (for users)
- Benchmark Visualizer (continuous verification)
  - Visualizes BenchExec results
  - Groups witnesses for each benchmark

Benchmark Visualizer*

- Web-interface** for verification results visualization
  - Easy to setup and use
- Database
  - Benchmark verification results
  - User marks: bugs, incorrect proofs, etc.

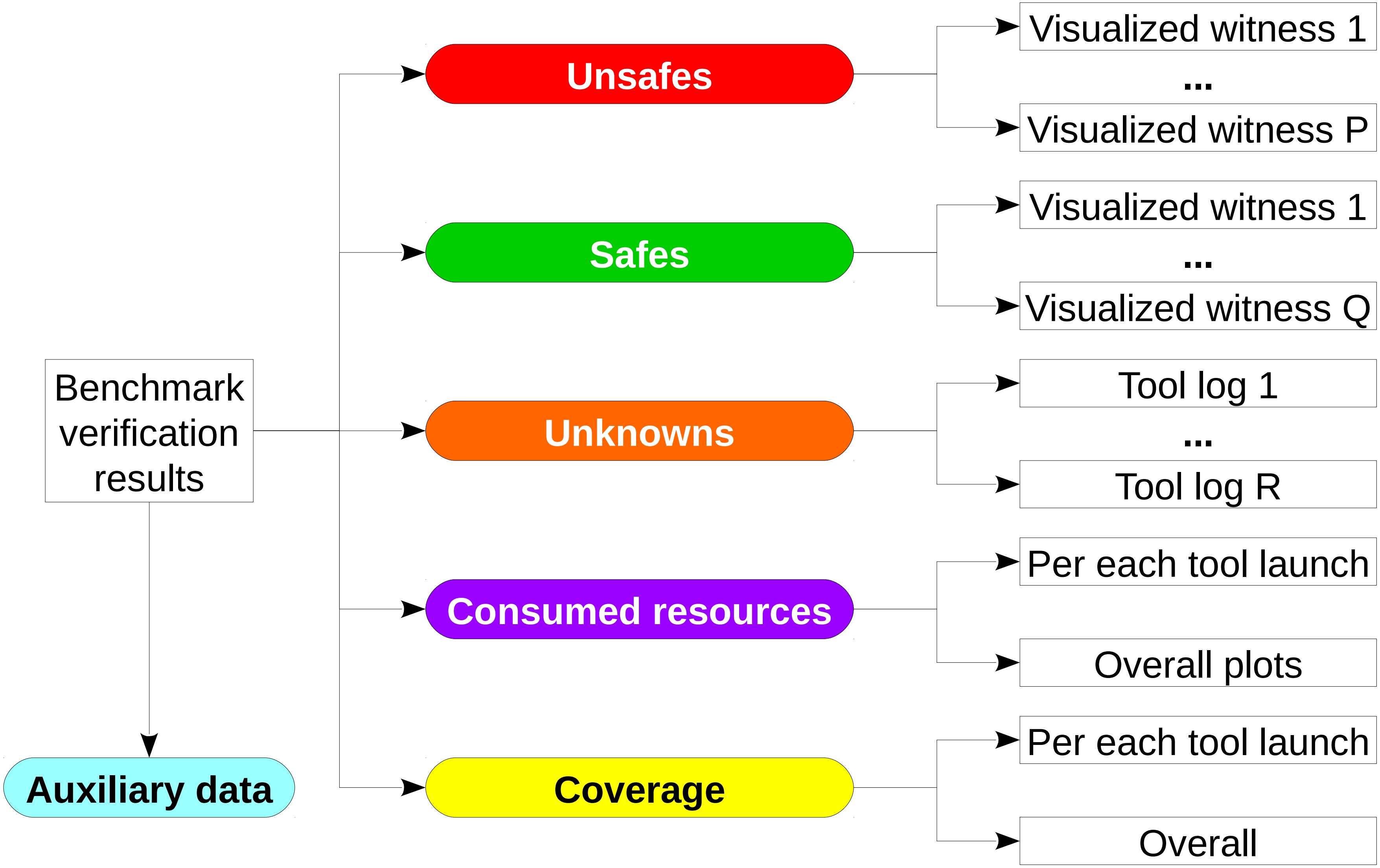
➡ Differences with BenchExec table-generator

  - Witnesses in user-friendly format
  - Means for manual results analysis
  - Coverage (if presented)

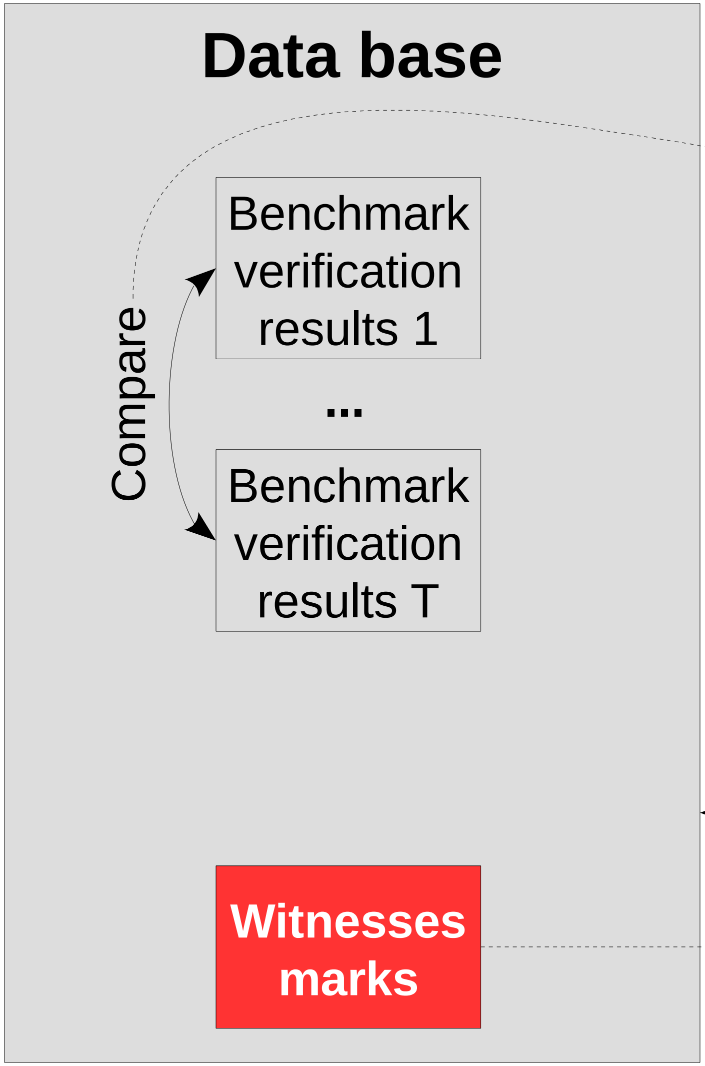
* https://github.com/ispras/cv

** https://github.com/mutilin/klever (branch cv-v2.0)

Benchmark Verification Results Visualization

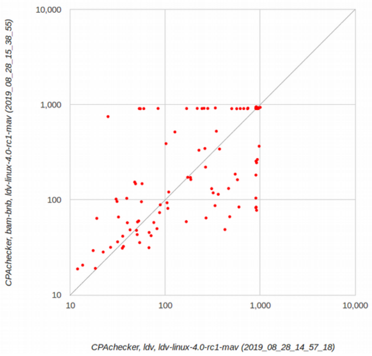


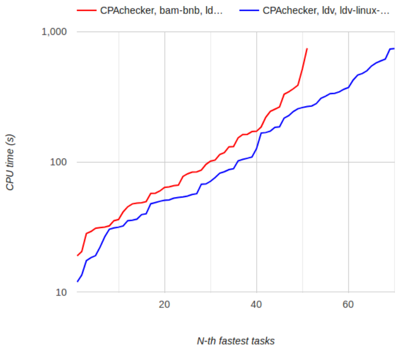
Benchmark Visualizer Database


Data base
Compare
Benchmark verification results 1
...
Benchmark verification results T
Witnesses marks

Results transitions	Witness clusters
Safe → Unknown: 2 Unsafe → Unknown: 25 Unknown → Safe: 6 Unknown → Unsafe: 2	Common: 23 Lost: 27 New: 24

Consumed resources





Benchmark verification results T+1

Add new results

Compare new witnesses with already marked ones

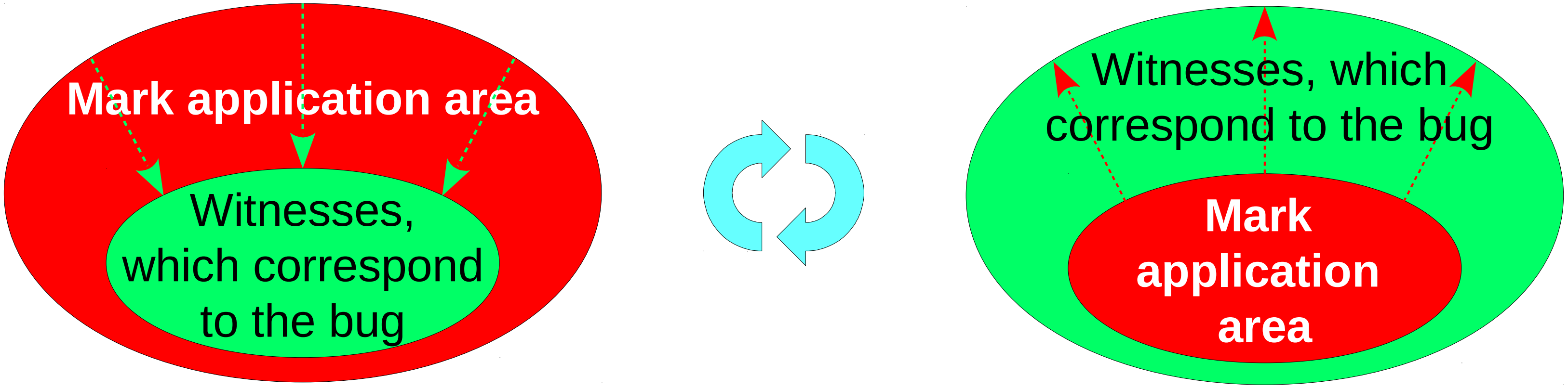
Speedups manual analysis

Manual Results Analysis*

- User analyses a witness to determine a bug
- User creates a mark for a bug
    - Witness + comparison criteria + description
- User adjusts the created mark

The mark is applied to a witness, which corresponds to other bugs

There is a witness, which corresponds to the same bug, but the mark was not applied to it



Mark application area

Witnesses, which correspond to the bug

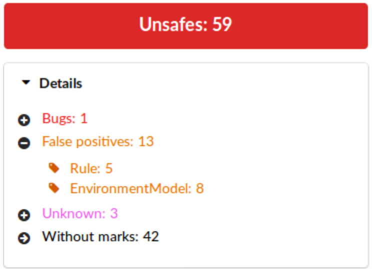
Witnesses, which correspond to the bug

Mark application area

* Based on violation witnesses.

Benchmark Visualizer Example

Violation witnesses

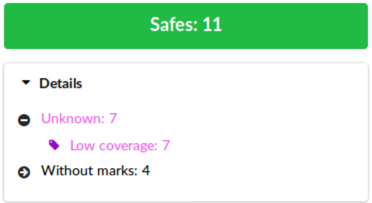


Unsafes: 59

▾ **Details**

- ⊕ Bugs: 1
- ⊖ False positives: 13
  - Rule: 5
  - EnvironmentModel: 8
- ⊕ Unknown: 3
- ➔ Without marks: 42

Correctness witnesses

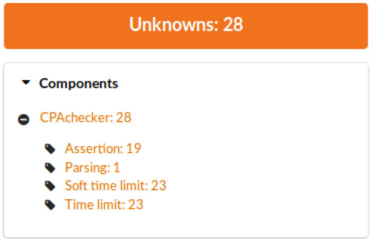


Safes: 11

▾ **Details**

- ⊖ Unknown: 7
  - Low coverage: 7
- ➔ Without marks: 4

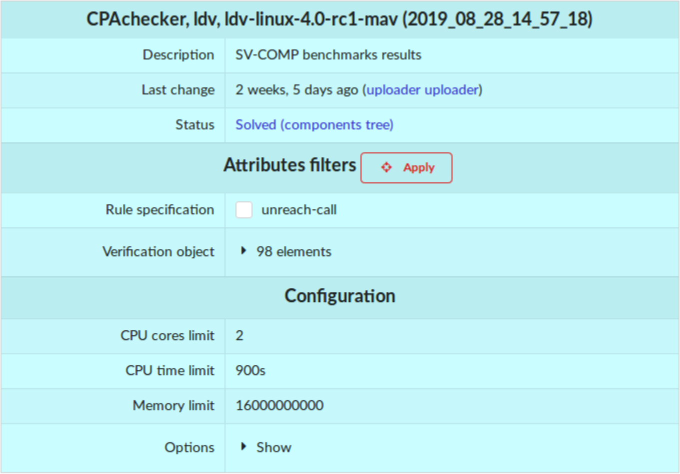
Verifier logs



Unknowns: 28

▾ **Components**

- ⊖ CPAchecker: 28
  - Assertion: 19
  - Parsing: 1
  - Soft time limit: 23
  - Time limit: 23



CPAchecker, ldv, ldv-linux-4.0-rc1-mav (2019_08_28_14_57_18)

Description	SV-COMP benchmarks results
Last change	2 weeks, 5 days ago (uploader uploader)
Status	Solved (components tree)

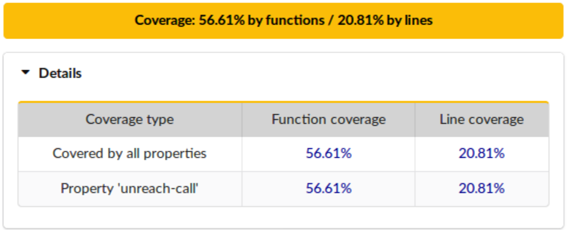
Attributes filters ✧ Apply

Rule specification	□ unreach-call
Verification object	▸ 98 elements

Configuration

CPU cores limit	2
CPU time limit	900s
Memory limit	16000000000
Options	▸ Show

Auxiliary data / filters

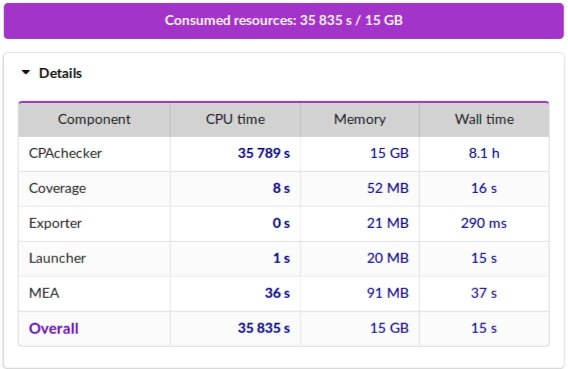


Coverage: 56.61% by functions / 20.81% by lines

▾ **Details**

Coverage type	Function coverage	Line coverage
Covered by all properties	56.61%	20.81%
Property 'unreach-call'	56.61%	20.81%

Coverage per property



Consumed resources: 35 835 s / 15 GB

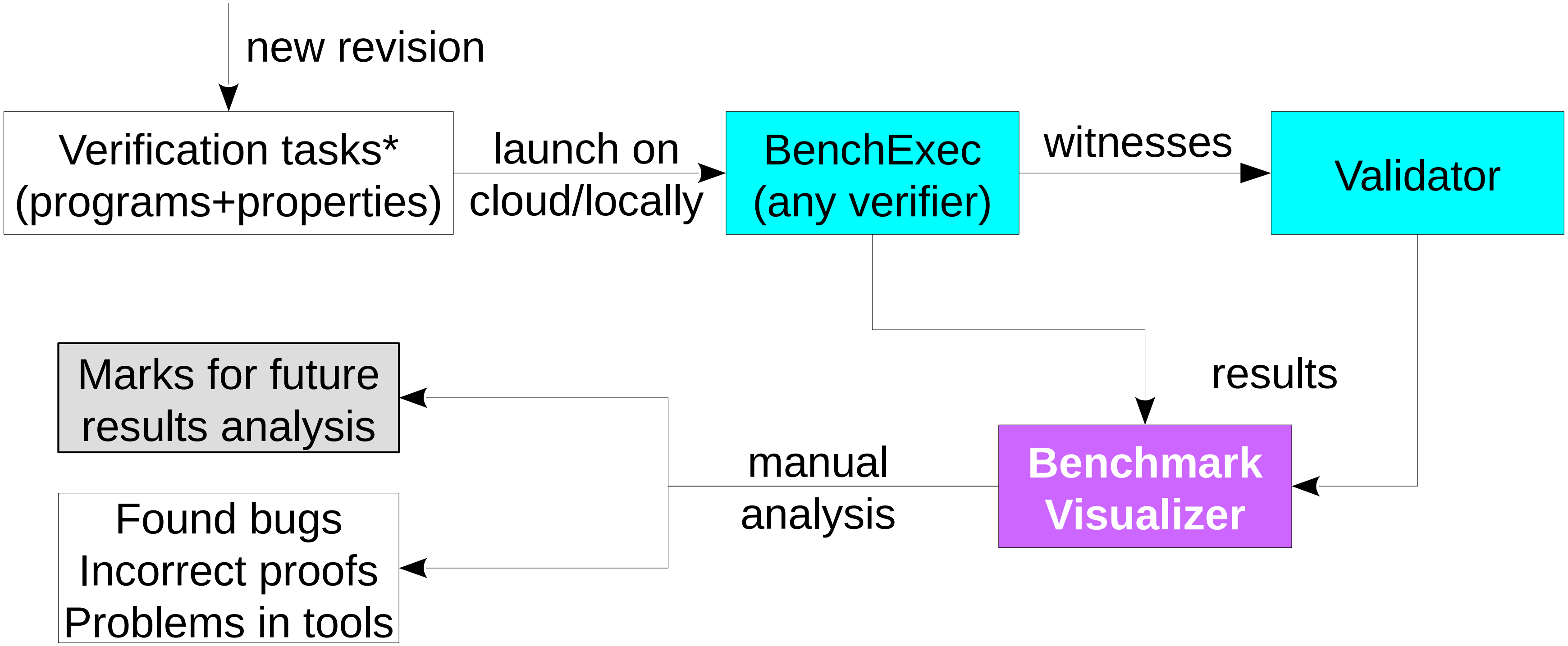
▾ **Details**

Component	CPU time	Memory	Wall time
CPAchecker	35 789 s	15 GB	8.1 h
Coverage	8 s	52 MB	16 s
Exporter	0 s	21 MB	290 ms
Launcher	1 s	20 MB	15 s
MEA	36 s	91 MB	37 s
Overall	35 835 s	15 GB	15 s

Consumed resources plots

Benchmark Visualizer Application Area

- Continuous verification of industry systems



- SV-COMP tasks
  - Similar work-flow with the given verification tasks

* May require additional preparation by the user or other tool.

Conclusion

- Witness Visualizer
    - Converts witnesses to user-friendly format
- Correctness witnesses visualization
    - New ideas of visualization
    - A simple implementation of the ideas
- Benchmark Visualizer
    - Successfully applied to continuous verification
    - Can be applied for SV-COMP tasks

Future Plans

- Witness Visualizer improvements
    - Restore missing elements where possible
    - Improve feedback to the developers
- Correctness witnesses visualization
    - Suggest other views based on proof hints
    - Implement automatic coverage extraction
- Benchmark Visualizer improvements
    - Regression verification
    - Verification tasks preparation

The joint 4rd International Workshop on CPAchecker (CPA'19) and 9th Linux Driver Verification (LDV) Workshop
October 2, 2019, Frauenchiemsee, Germany

Thank you

Vitaly Mordan
mordan@ispras.ru

ISP RAS LinuxTesting.org

Ivannikov Institute for System Programming of the Russian Academy of Sciences