Просеминар ВМК
29 марта 2013 г.

LinuxTesting.org

Технологии программирования и верификации: от ядра Linux до систем авионики

часть 1

Хорошилов Алексей
khoroshilov@ispras.ru

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Linux Kernel Statistics (1)

- More than 1000 active developers

Kernel Version	# of Developers	# of Known Companies
2.6.11	483	71
2.6.12	701	90
2.6.13	637	91
2.6.14	625	89
2.6.15	679	96
2.6.16	775	100
2.6.17	784	106
2.6.18	897	121
2.6.19	878	126
2.6.20	728	130
2.6.21	834	132
2.6.22	957	176
2.6.23	991	178
2.6.24	1,057	186
All	3,678	271

[*] Kroah-Hartman G, Corbet J, McPherson A (2008) Linux Kernel Development
http://www.linux-foundation.org/publications/linuxkerneldevelopment.php

Linux Kernel St

- More than 1000 a

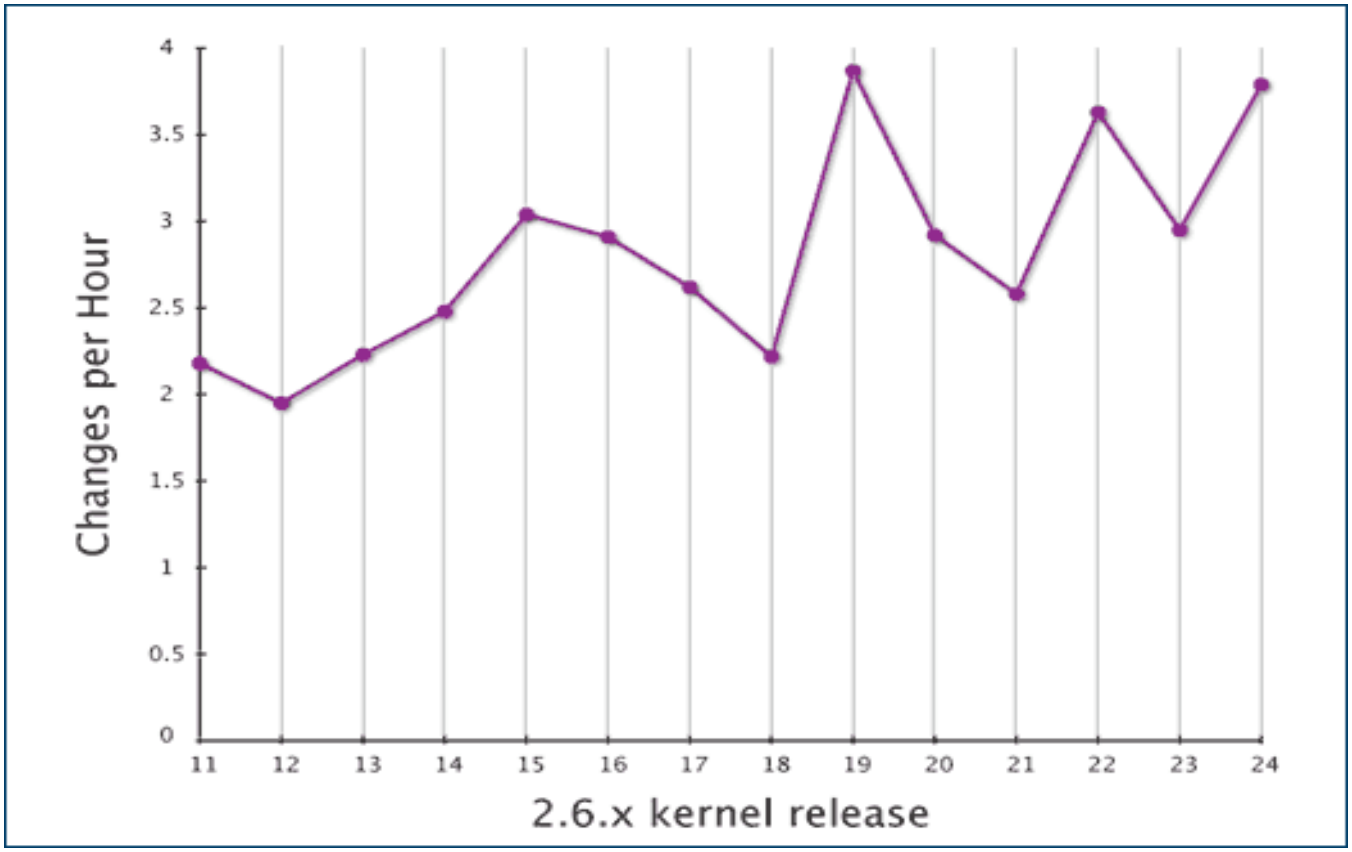
Kernel Version	#
2.6.11	
2.6.12	
2.6.13	
2.6.14	
2.6.15	
2.6.16	
2.6.17	
2.6.18	
2.6.19	
2.6.20	
2.6.21	
2.6.22	
2.6.23	
2.6.24	
All	

Kernel Version	Number of Developers	Number of Known Companies
2.6.11	389	68
2.6.12	566	90
2.6.13	545	94
2.6.14	553	90
2.6.15	612	108
2.6.16	709	111
2.6.17	726	120
2.6.18	815	133
2.6.19	801	128
2.6.20	673	138
2.6.21	767	143
2.6.22	870	180
2.6.23	912	181
2.6.24	1,057	193
2.6.25	1,123	232
2.6.26	1,027	203
2.6.27	1,021	187
2.6.28	1,075	212
2.6.29	1,180	233
2.6.30	1,150	249
2.6.31	1,166	227
2.6.32	1,248	261
2.6.33	1,196	238
2.6.34	1,150	243
2.6.35	1,187	209
2.6.36	1,176	207
2.6.37	1,276	221
2.6.38	1,198	220
2.6.39	1,258	239
3.0	1,131	331
3.1	1,168	212
3.2	1,316	226
All	7,944	855

[*] Kroah-Hartman G, Corbet J, McPh
http://www.linux-foundation.org/publica

Linux Kernel Statistics (2)

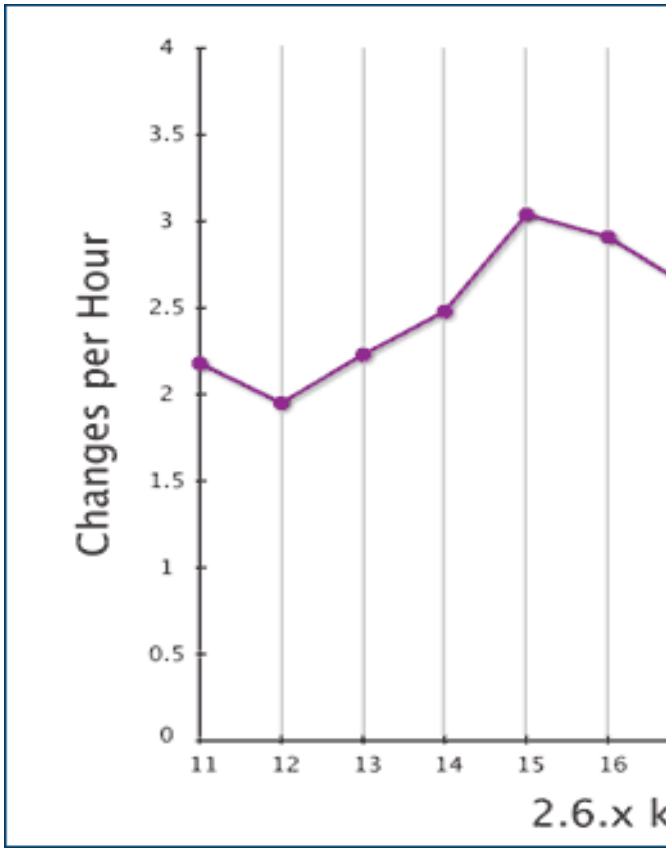
- 2.83 patches per hour applied (avg for 2.5 years)



[*] Kroah-Hartman G, Corbet J, McPherson A (2008) Linux Kernel Development
http://www.linux-foundation.org/publications/linuxkerneldevelopment.php

Linux Kernel Stat

- 2.83 patches per ho .5 years)

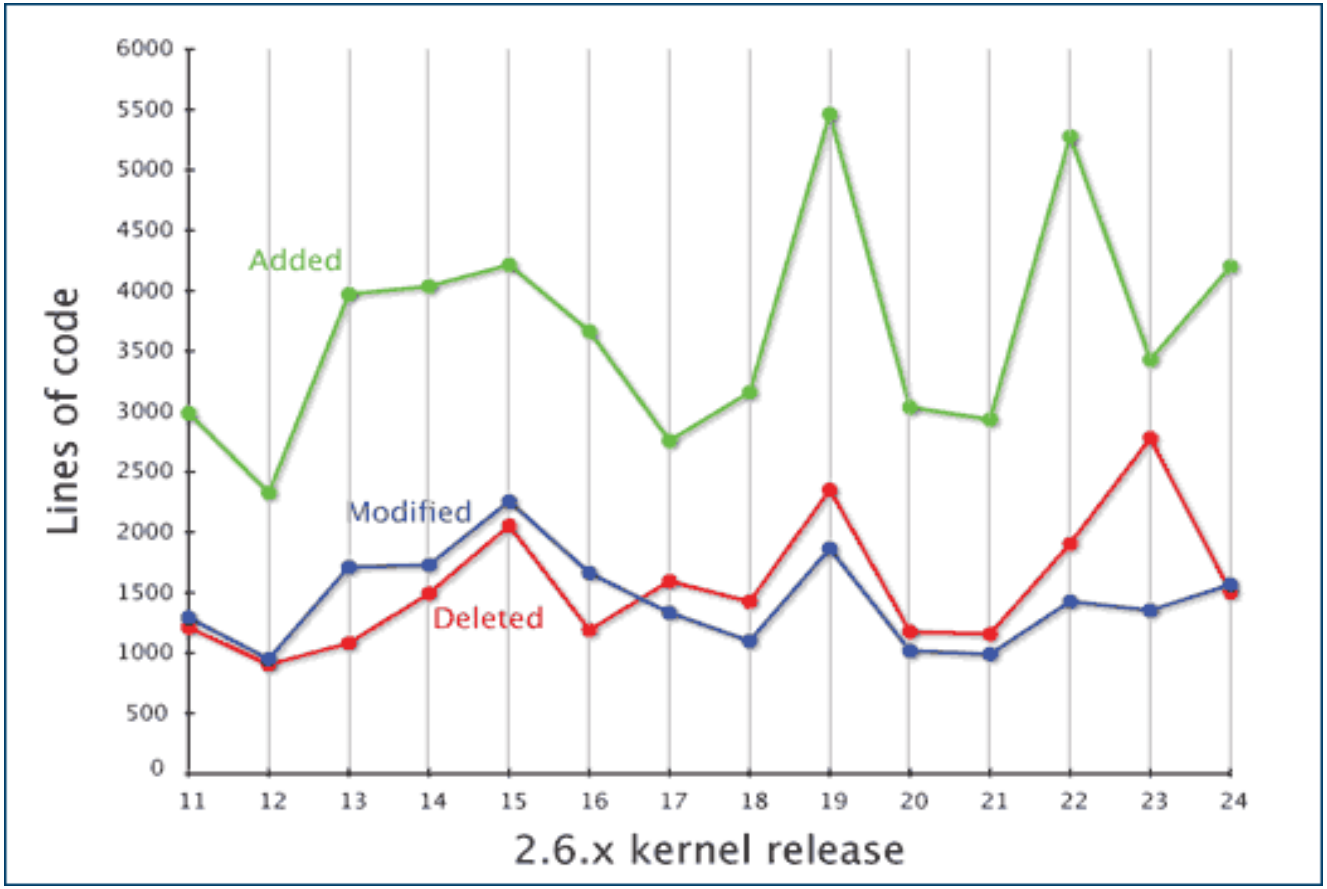


2.6.15	3.04
2.6.16	2.91
2.6.17	2.62
2.6.18	2.22
2.6.19	3.87
2.6.20	2.92
2.6.21	2.58
2.6.22	3.63
2.6.23	2.95
2.6.24	3.79
2.6.25	6.15
2.6.26	4.71
2.6.27	5.03
2.6.28	4.96
2.6.29	5.47
2.6.30	6.40
2.6.31	4.93
2.6.32	5.46
2.6.33	5.39
2.6.34	4.86
2.6.35	5.30
2.6.36	4.95
2.6.37	6.28
2.6.38	5.78
2.6.39	6.58
3.0	5.96
3.1	3.81
3.2	6.88

[*] Kroah-Hartman G, Corbet J, McPherso t
http://www.linux-foundation.org/publicatior

Linux Kernel Statistics (3)

- 3621 added/1550 removed/1425 modified every day



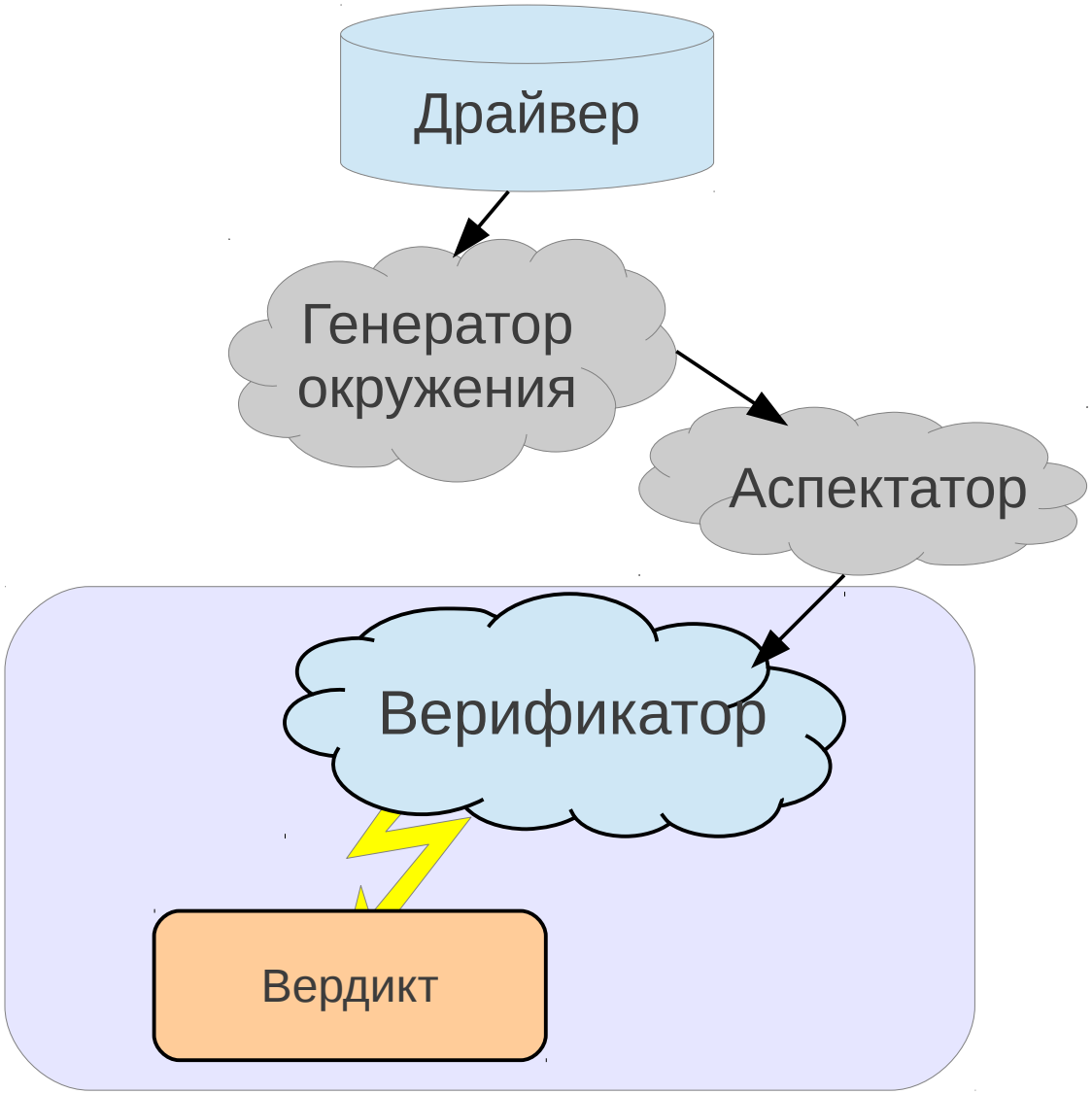
[*] Kroah-Hartman G, Corbet J, McPherson A (2008) Linux Kernel Development
http://www.linux-foundation.org/publications/linuxkerneldevelopment.php

Теорема Крейга (1957)

если для двух логических формул A и B общезначима (тождественно истинна на любой модели) импликация A => B, то существует логическая формула C, называемая интерполянтом Крейга, которая удовлетворяет трём условиям:

1. A => C;
2. C => B;
3. каждый неинтерпретируемый символ в формуле является общим для формул A и B.

Схема работы LDV



Код на входе инструмента

```
int probe()
{
    //...
    int x, y;
    int z;
    if (x > y)
        z = x - y;
    else
        z = y - x;
    ldv_assert(z >= 0);
    //...
}

int main()
{
    //...
    probe();
    //...
}
```

```
void ldv_assert(int condition)
{
    if (!condition)
    ERROR: goto ERROR;
}
```

ошибочная метка

точка входа

Задача инструмента

Проверка достижимости ошибочной метки

Достижима ли ошибочная метка при ***каком-либо*** из возможных вариантов исполнения программы?

Комбинаторный взрыв

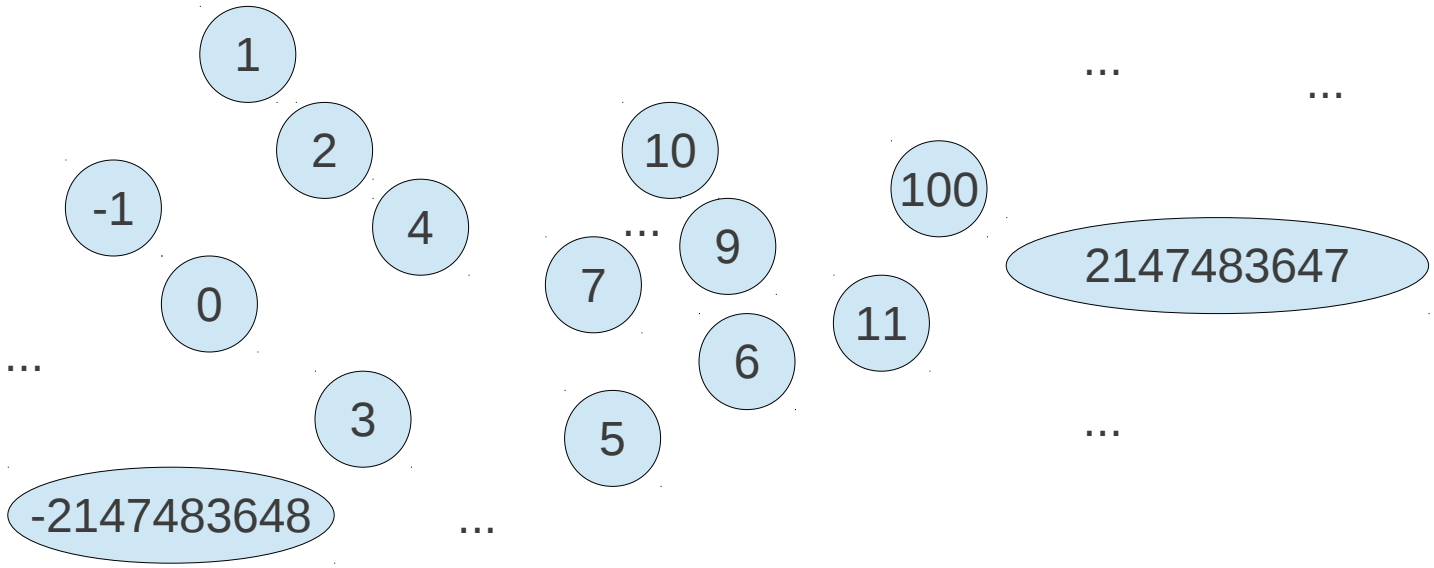
```
int x, y;
int z;
if (x > y)
    z = x - y;
else
    z = y - x;
if (!(z >= 0))
ERROR: goto ERROR;
```

Число начальных состояний:

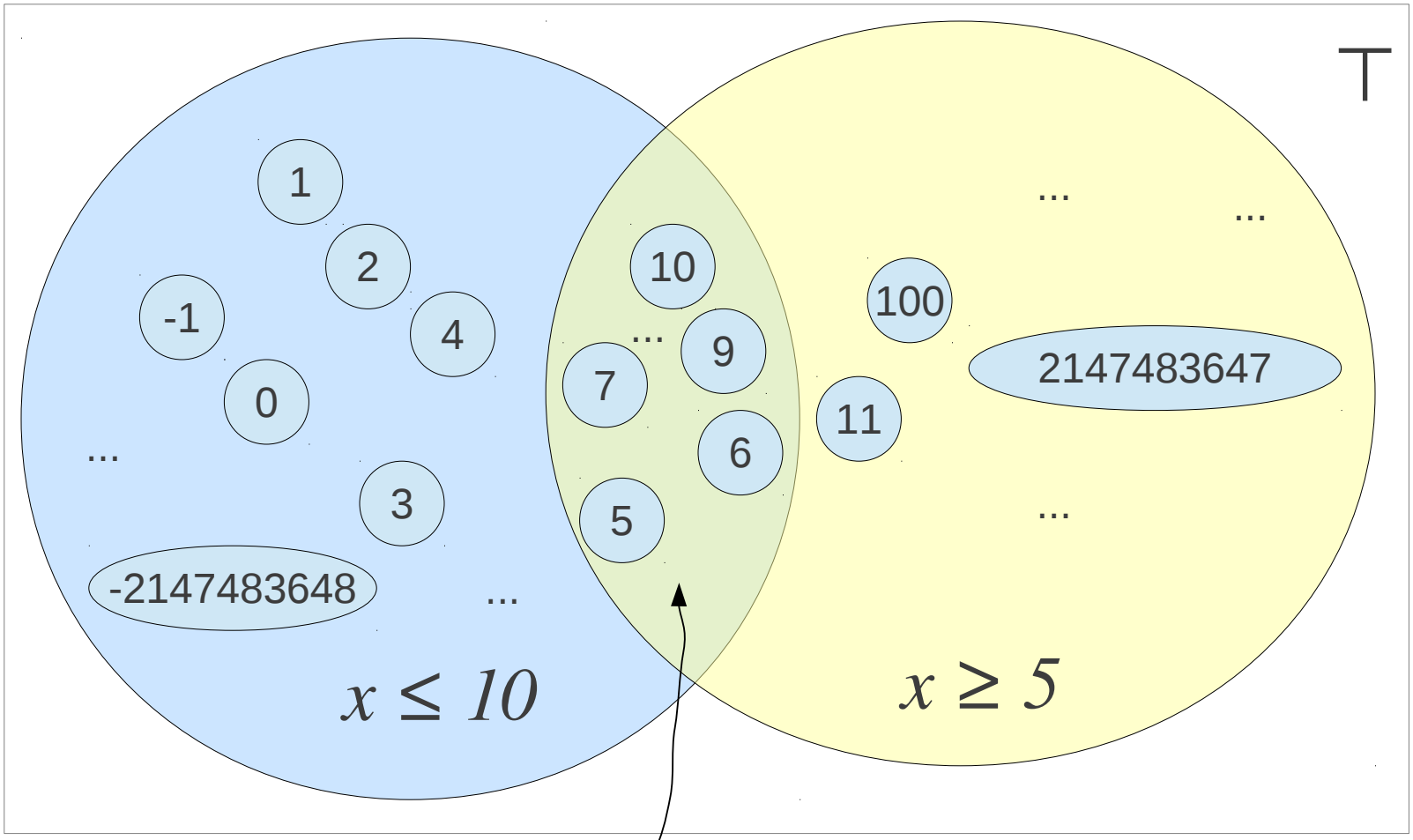
$(2^{32})^3 > 7{,}9 \cdot 10^{28}$ –
несколько миллиардов лет на перебор

Предикатная абстракция (1)

Значения переменной x



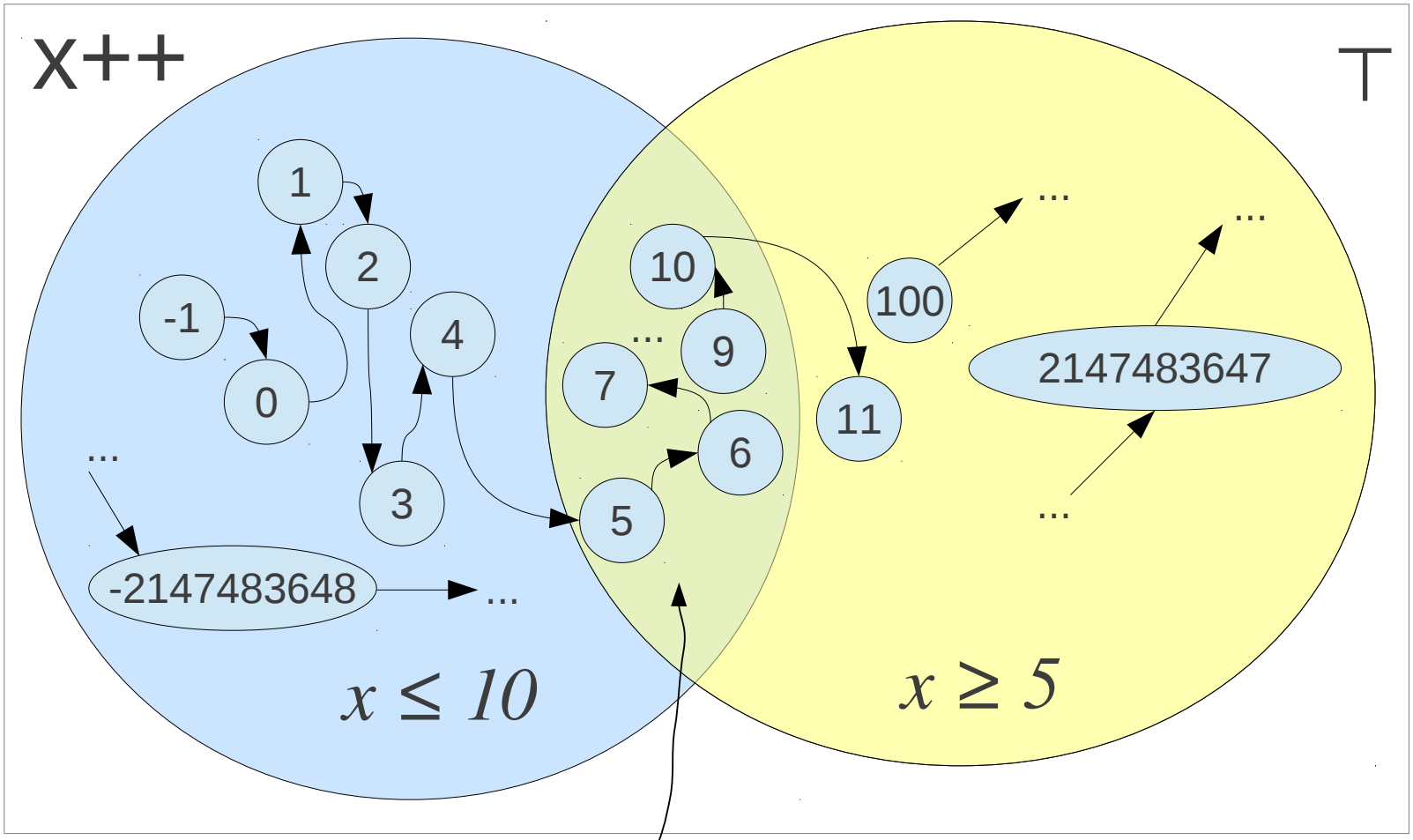
Предикатная абстракция (2)



$x \geq 5 \wedge x \leq 10$

Абстрактные соятояния

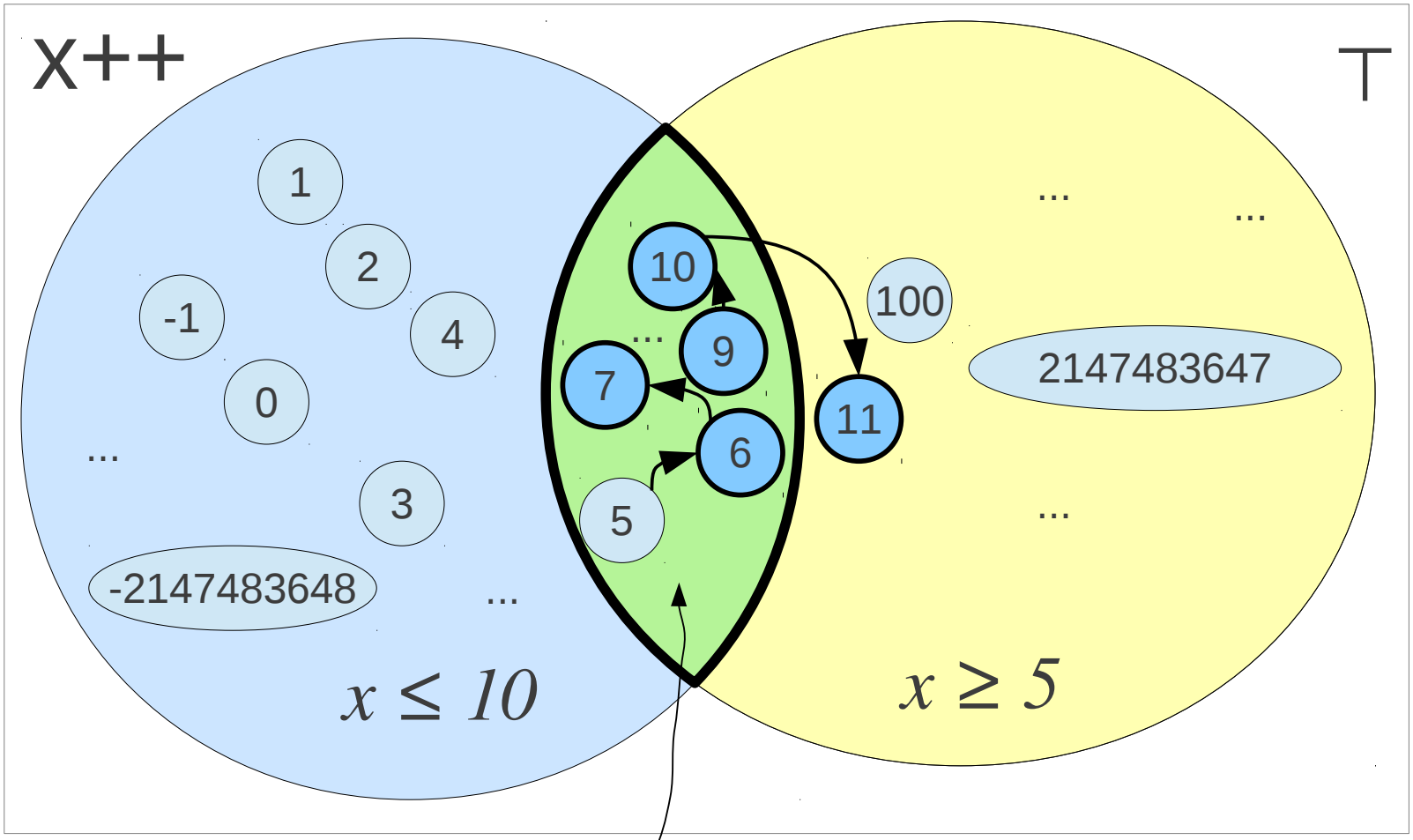
Предикатная абстракция (3)



$x \geq 5 \wedge x \leq 10$

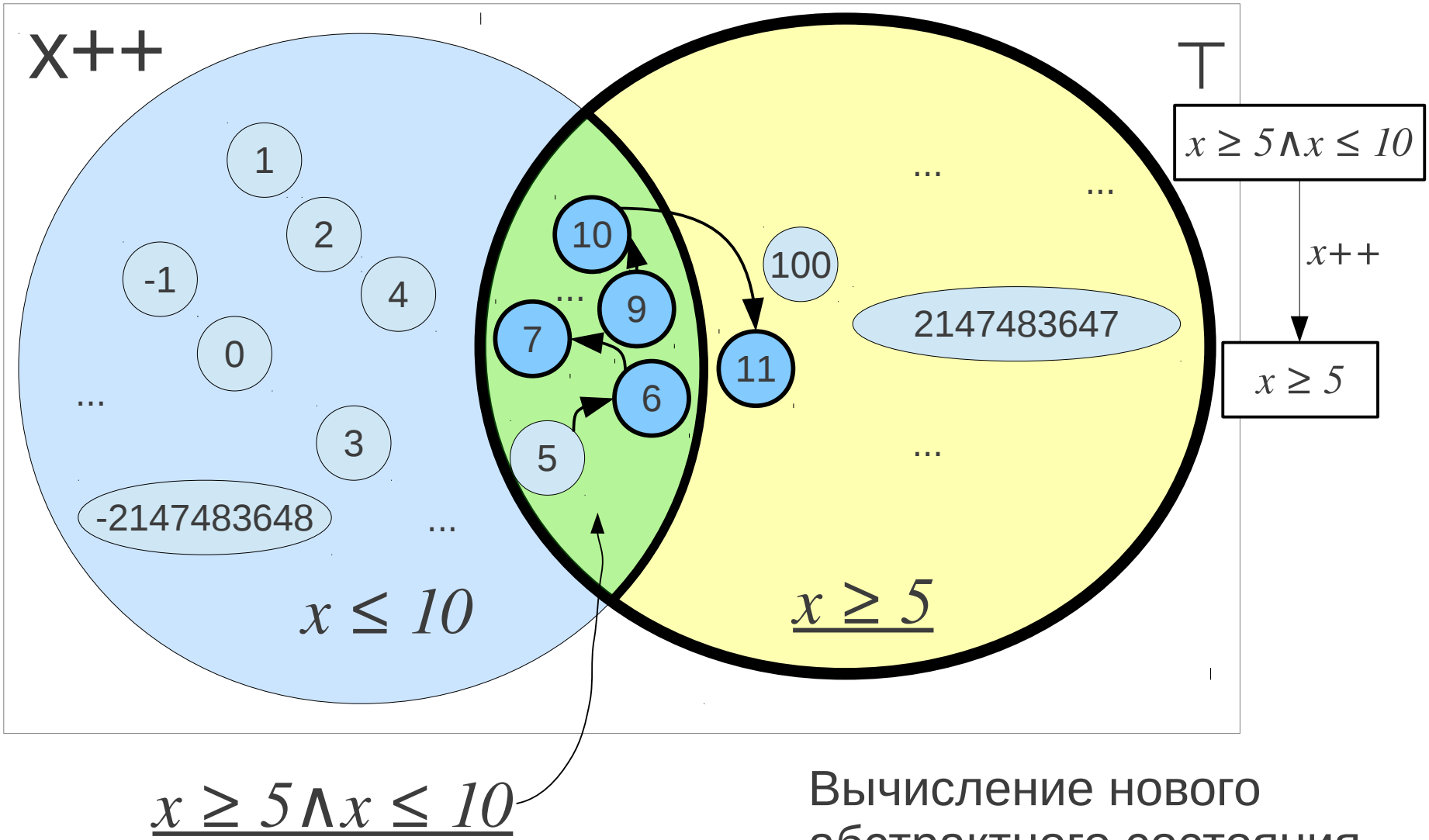
Переходы между состояниями

Предикатная абстракция (4)



$x \geq 5 \wedge x \leq 10$ Переход из абстрактного состояния

Предикатная абстракция (5)

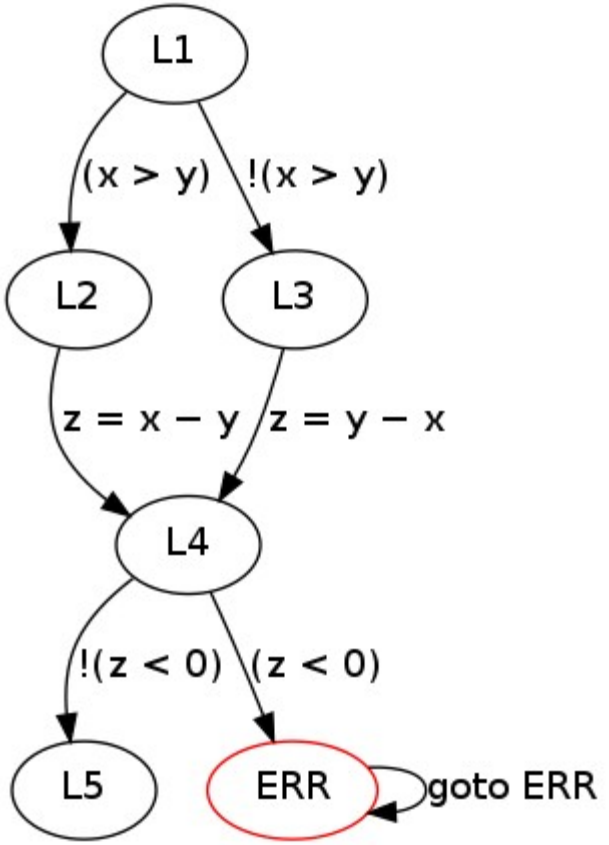


Вычисление нового абстрактного состояния

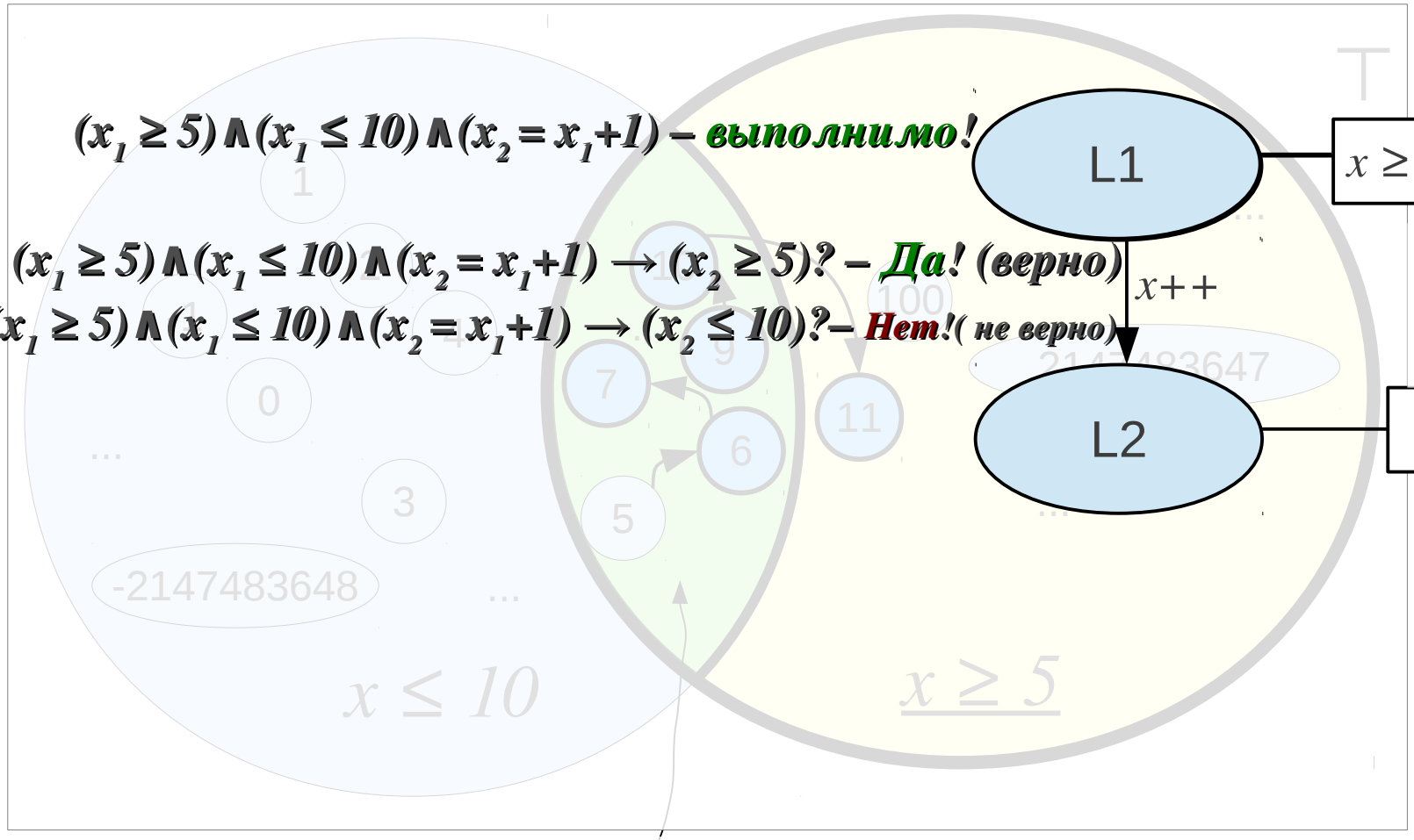
Построение ГПУ

```
L1: if (x > y)
L2:     z = x - y;
    else
L3:     z = y - x;

L4: if (!(z >= 0))
ERROR: goto ERROR;
L5:
```

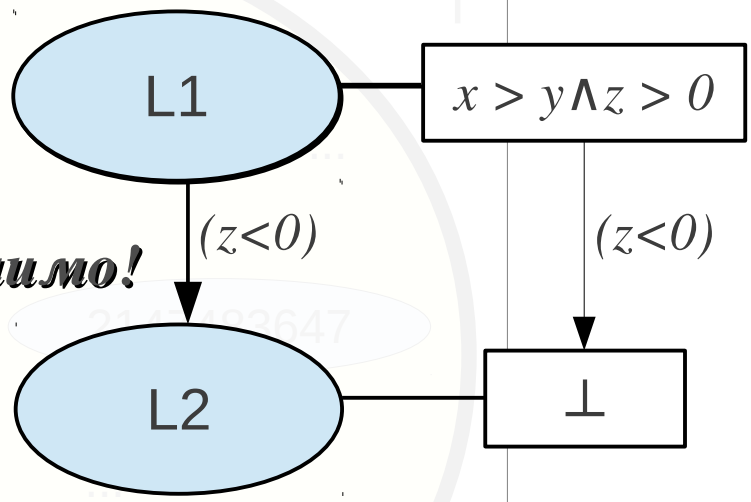



Предикатная абстракция (6)



Предикатная абстракция (7)

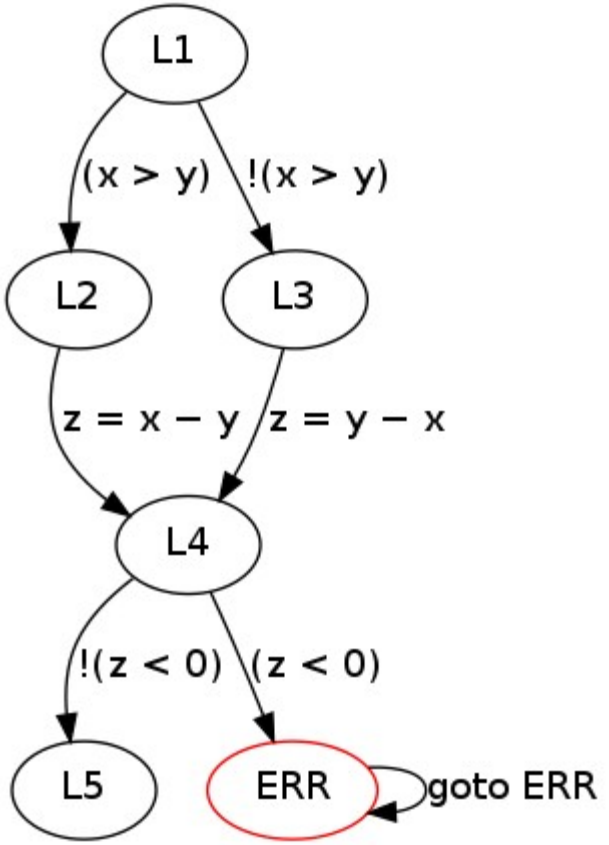
$(x_1 > y_1) \wedge (z_1 > 0) \wedge (z_1 < 0)$ – **не** выполнимо!



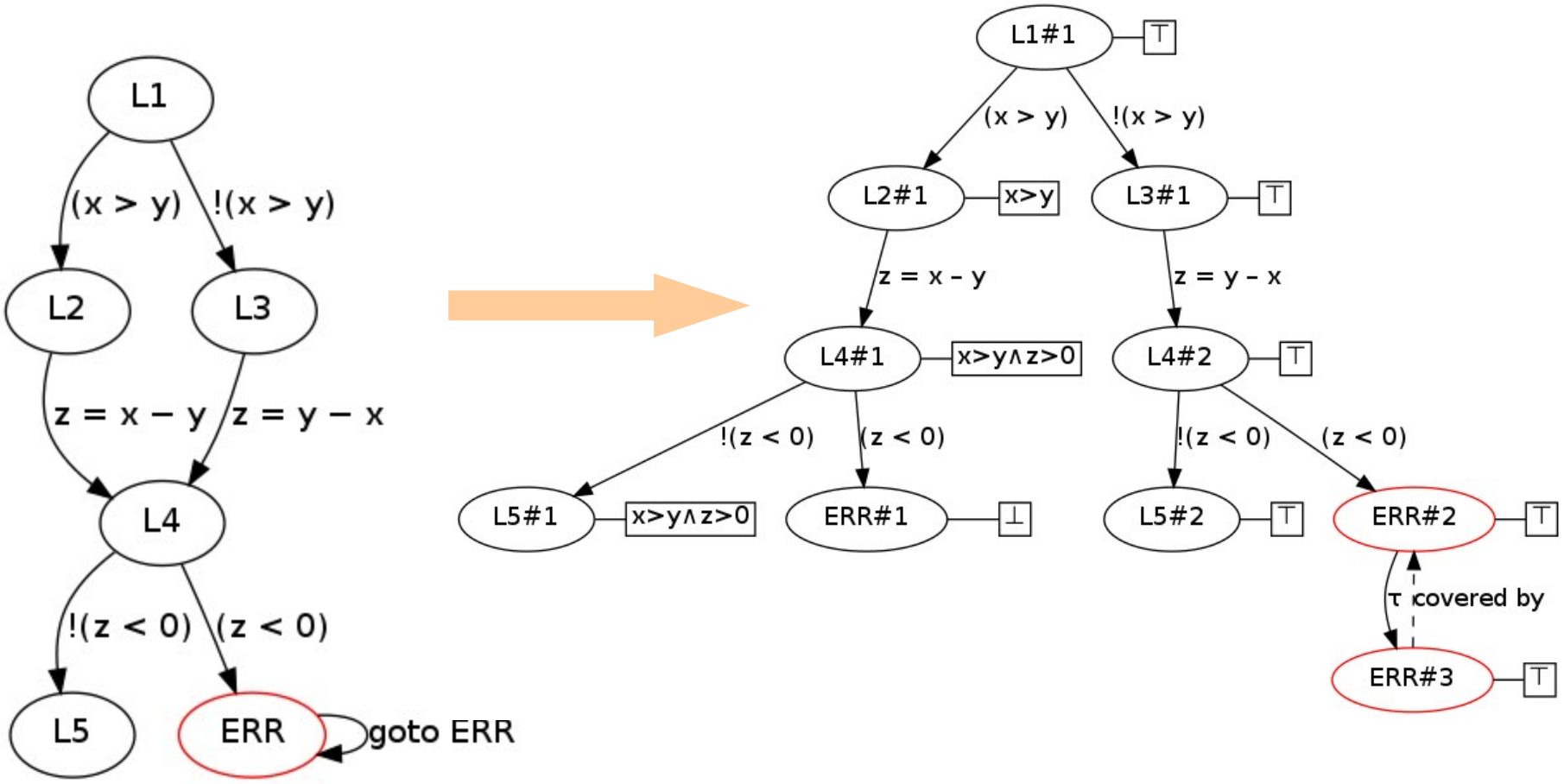
Построение ГПУ

```
L1: if (x > y)
L2:     z = x - y;
    else
L3:     z = y - x;

L4: if (!(z >= 0))
ERROR: goto ERROR;
L5:
```

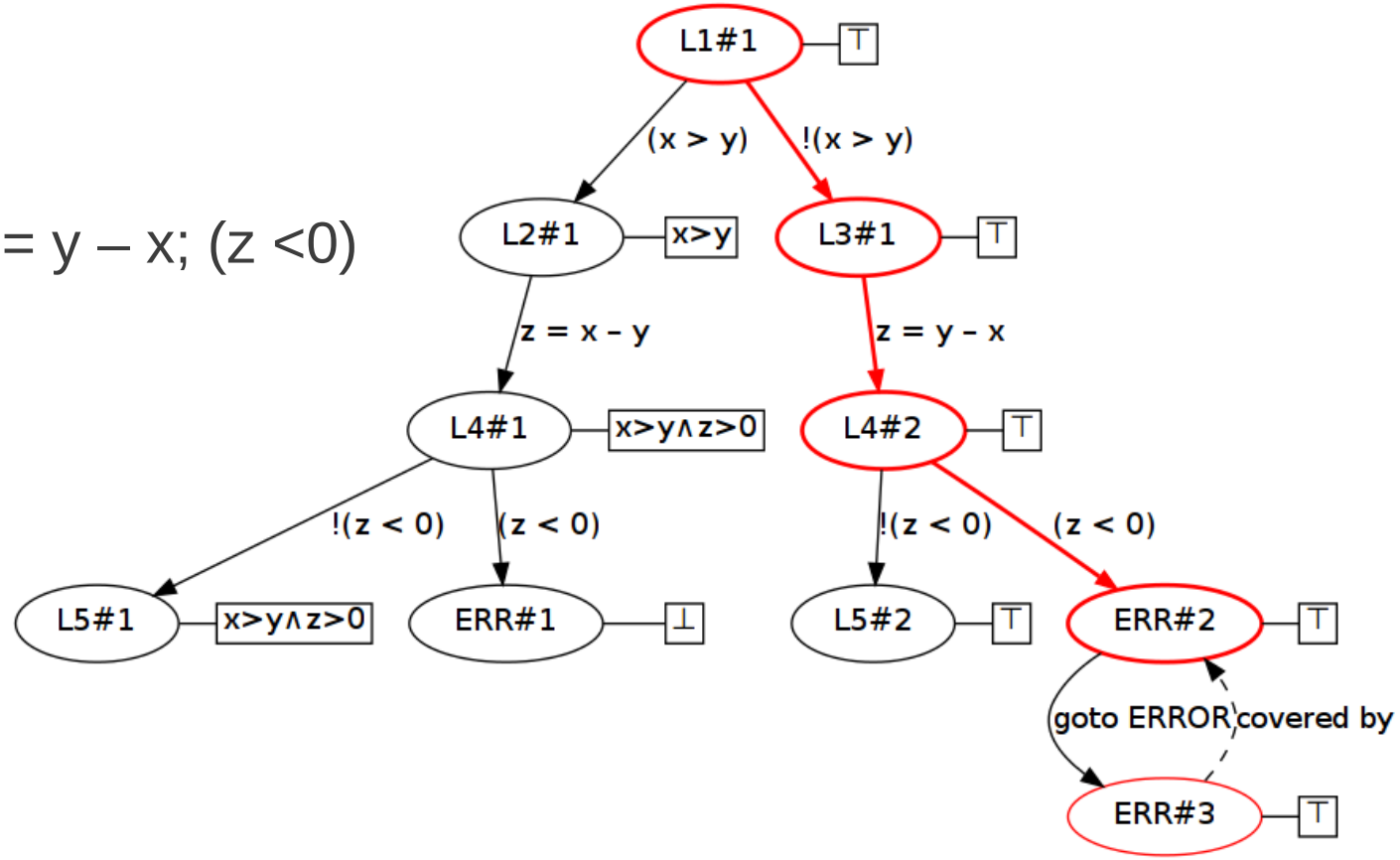



Построение АДД



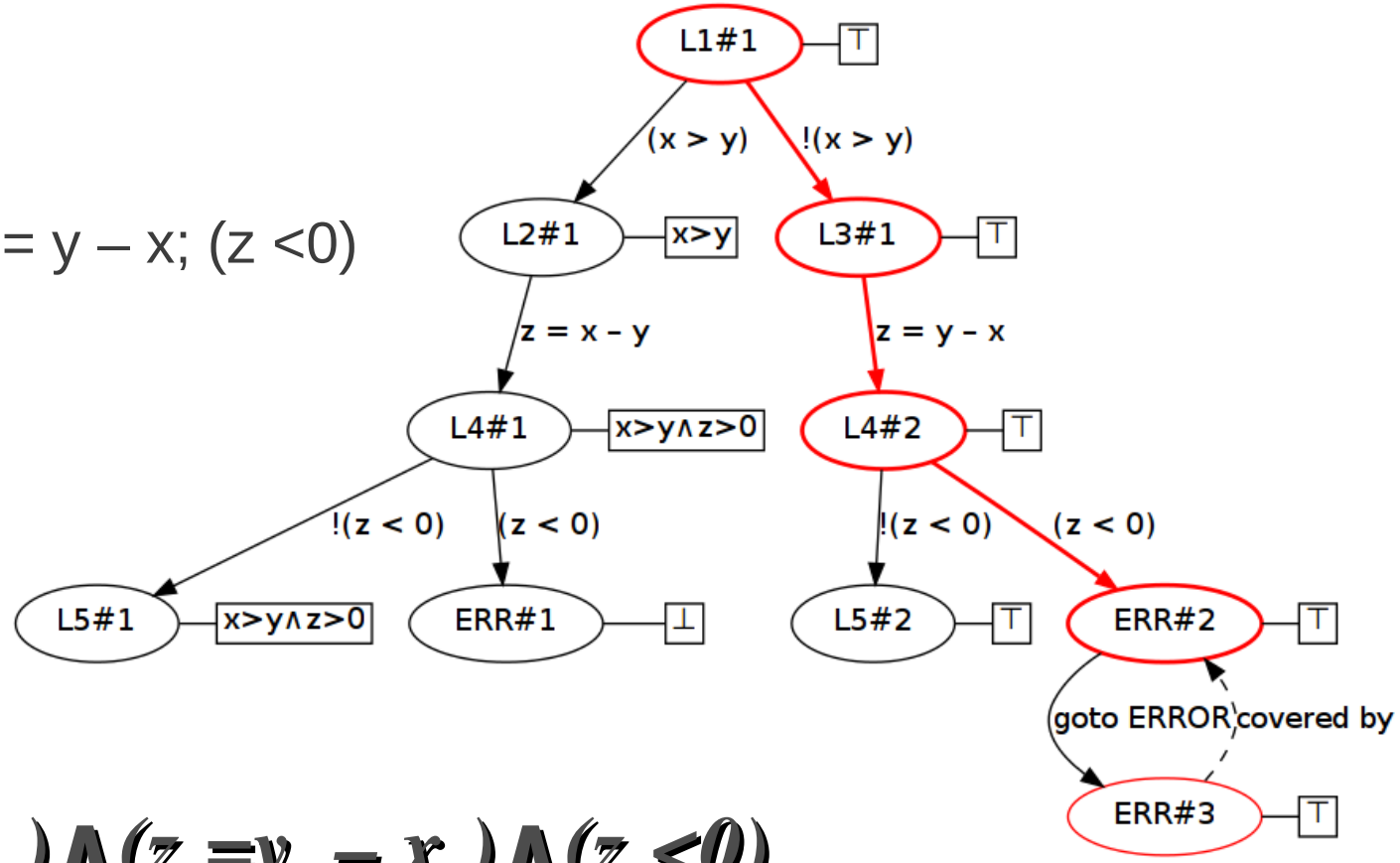
Контрпример

!(x > y); z = y – x; (z <0)



Формула пути

!(x > y); z = y – x; (z <0)

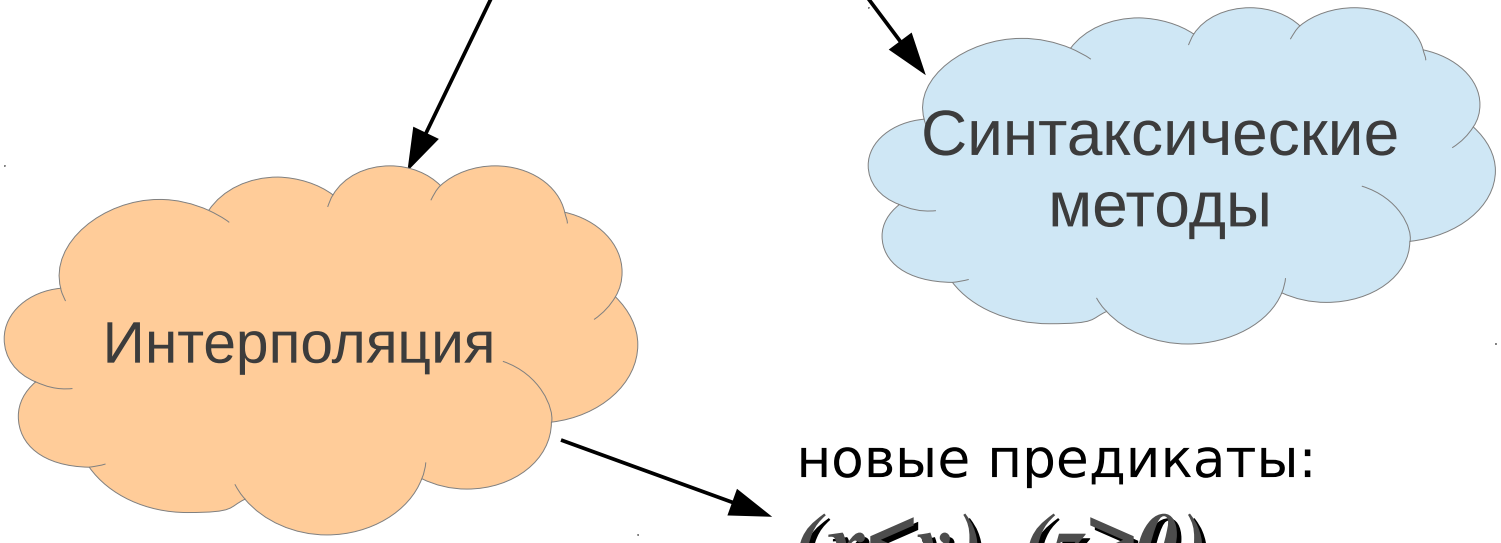


$$\neg(x_1 > y_1) \wedge (z_2 = y_1 - x_1) \wedge (z_2 < 0)$$

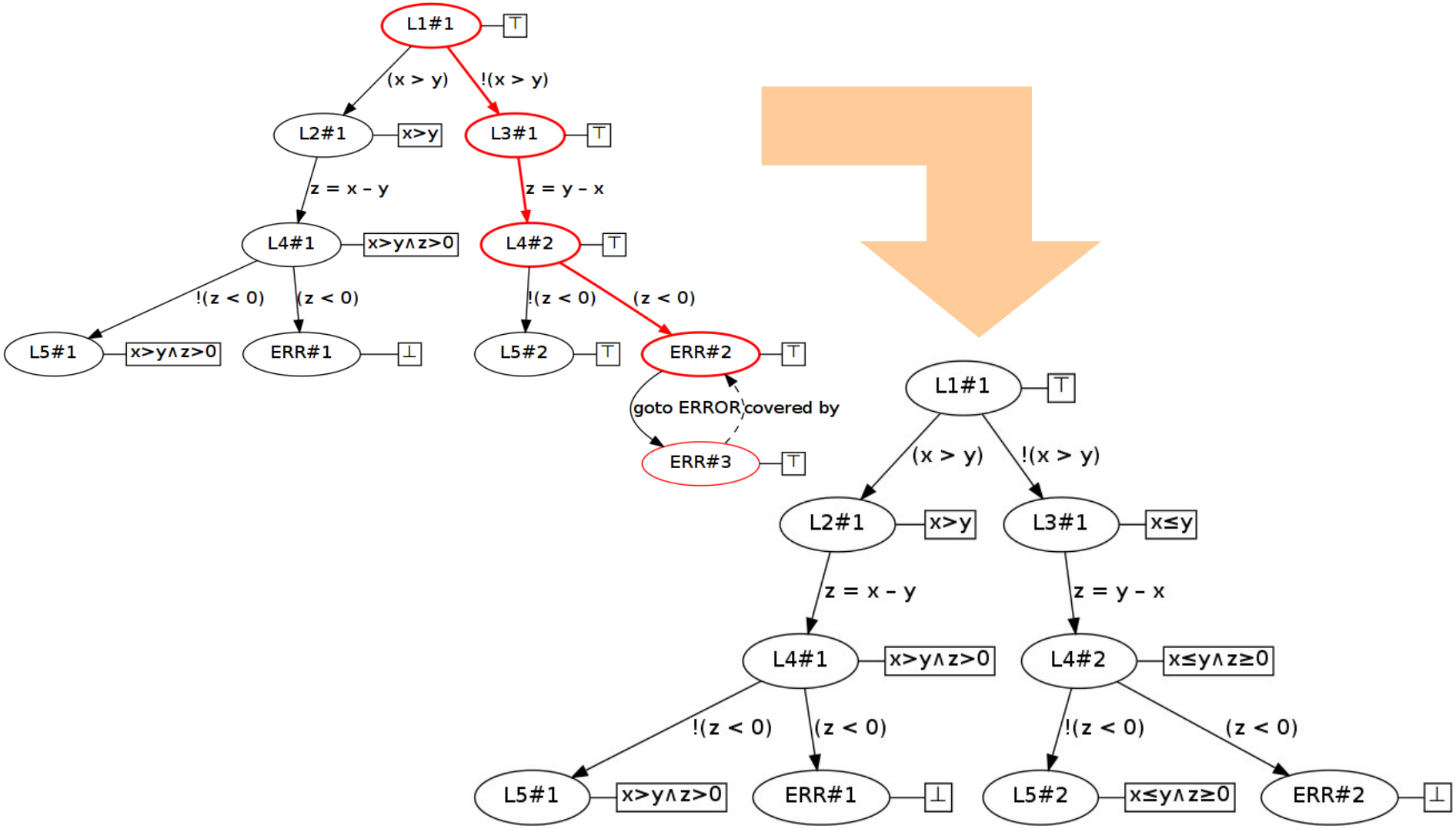
Получение предикатов

!(x > y); z = y – x; (z <0)

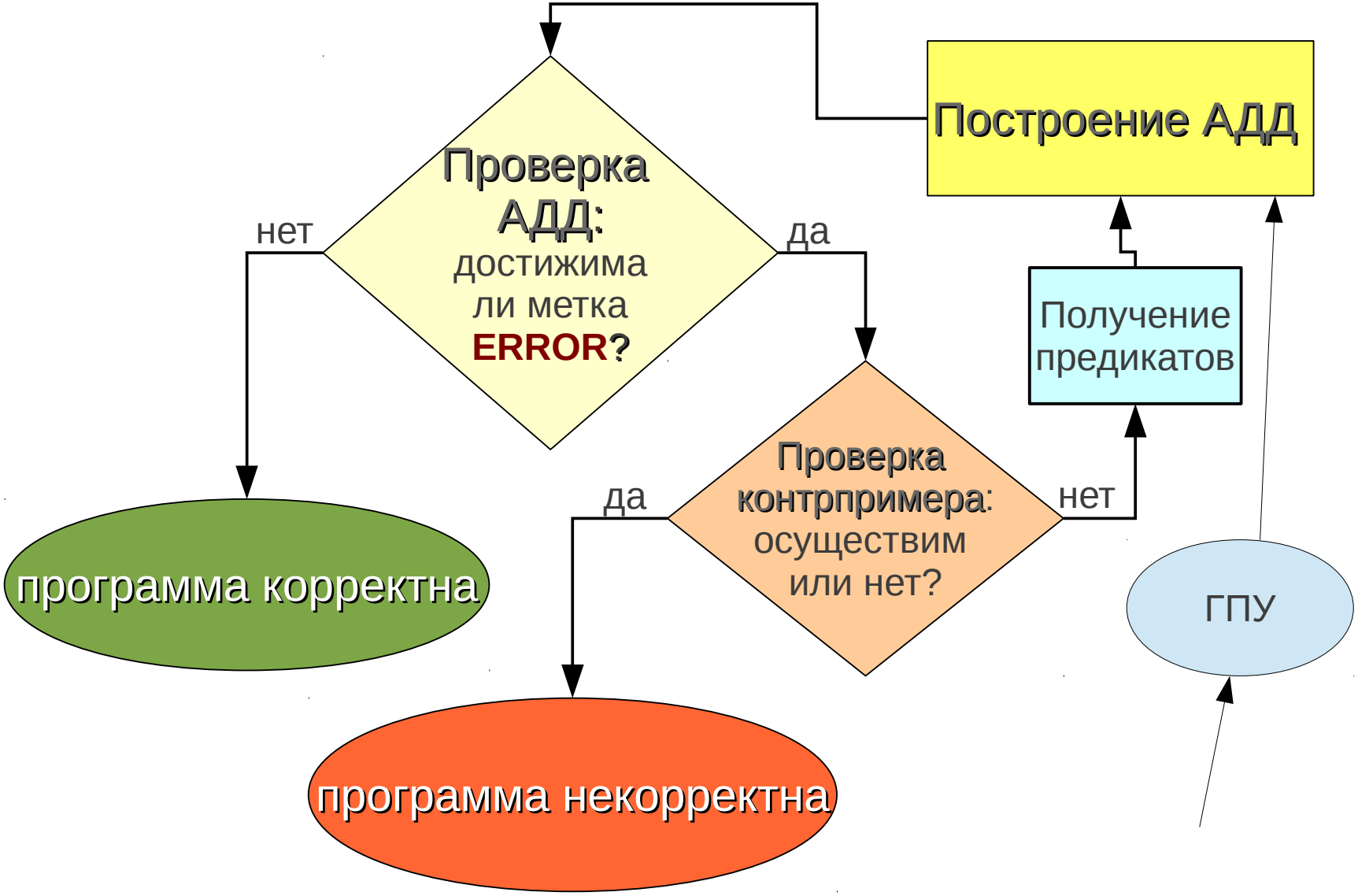
$\neg(x_1 > y_1) \wedge (z_2 = y_1 - x_1) \wedge (z_2 < 0)$ – **не** выполнима



Уточнение абстракции



Подход CEGAR



Подход CEGAR

Couner-Example Guilded Abstraction Refinement – уточнение абстракции по контрпримеру

- Построение графа потока управления программы
- Разворачивание ГПУ в дерево достижимости
- В вершинах дерева хранятся логические формулы над некоторым набором предикатов
- Набор предикатов выводится во время верификации – по контрпримерам
- Чаще всего используются SMT-решатели

Результаты соревнований SV-COMP--2013

	Competition candidate	BLAST 2.7.1	CPAchecker-Explicit 1.1.10	CPAchecker-SeqCom 1.1.10	UFO 2012-10-22	ESBMC 1.20	LLBMC 2012-10-23	Predator 2012-10-20
	Representing Jury Member	Vadim Mutilin	Stefan Löwe	Philipp Wendler	Arie Gurfinkel	Lucas Cordeiro	Carsten Sinz	Tomas Vojnar
	Affiliation	Moscow, Russia	Passau, Germany	Passau, Germany	Pittsburgh, USA	Manaus, Brazil	Karlsruhe, Germany	Brno, Czechia
Битовые операции	BitVectors 32 files, max score: 60	--	16 86 s	**17** **190 s**	--	**24** **480 s**	**60** **36 s**	-75 95 s
	ControlFlowInteger 94 files, max score: 146	93 7 100 s	**143** **1 200 s**	**141** **3 400 s**	**146** **450 s**	90 * 17 000 s	--	-27 650 s
Драйверы	DeviceDrivers64 1237 files, max score: 2419	**2 338** **2 400 s**	**2 340** **9 700 s**	2 186 30 000 s	**2 408** **2 500 s**	2 233 46 000 s	--	0 0 s
Указатели	HeapManipulation 28 files, max score: 48	--	22 30 s	**22** **29 s**	--	--	**32** **310 s**	**40** **2.3 s**
	MemorySafety 36 files, max score: 54	--	0 0 s	0 0 s	--	**3** **1 300 s**	**24** **38 s**	**52** **61 s**

CEGAR

BMC

Shape-анализ

Device Driver World

```
static struct pci_driver DAC960_pci_driver = {
        .name           = "DAC960",
        .id_table       = DAC960_id_table,
        .probe          = DAC960_Probe,
        .remove         = DAC960_Remove,
};

static int DAC960_init_module(void)
{
        int ret;
        ret = pci_register_driver(&DAC960_pci_driver)
#ifdef DAC960_GAM_MINOR
        if (!ret)
                DAC960_gam_init();
#endif
        return ret;
}
...
module_init(DAC960_init_module);
module_exit(DAC960_cleanup_module);
```

Callback interface procedures registration

No explicit calls to linking-level init procedures

Pseudo-main generation

```
int main(int argc,char* argv[])
{
  init_module()
  for(;;) {
    switch(*) {
     case 0: driver_probe(*,*,*);break;
     case 1: driver_open(*,*);break;
     ...
    }
  }
  exit_module();
}
```

Pseudo-main generation (2)

- Order limitation
  - open() after probe(), but before remove()
- Implicit limitations
  - read() only if open() succeed
- and it is specific for each class of drivers

Rule Instrumentor

```
mutex x;
int f(int y)
{
  lock(x);
  ...
  unlock(x);
  return y;
}
```

```
int x_locked = 0;
int f(int y)
{
  assert(x_locked == 0);
  x_locked = 1;
  ...
  assert(x_locked == 1);
  x_locked = 0;
  return y;
}
```

Aspect-Oriented Approach

```
mutex x;
int f(int y)
{
   lock(x);
   ...
   unlock(x);
   return y;
}
```

Aspect:

```
around:
call(int lock(mutex x)
{
 assert(x_locked == 0);
 x_locked = 1;
}
```

Rule Instrumentor

```
mutex x;
int f(int y)
{
  lock(x);
  ...
  unlock(x);
  return y;
}
```

```
int x_locked = 0;
int f(int y)
{
  assert(x_locked == 0);
  x_locked = 1;
  ...
  assert(x_locked == 1);
  x_locked = 0;
  return y;
}
```

Rule Instrumentor: Implementation

- **CIF** – C Instrumentation Framework
  - gcc-based aspect-oriented programming tool for C language
  - available at forge.ispras.ru under GPLv3

LinuxTesting.org

Спасибо!

Хорошилов Алексей
khoroshilov@ispras.ru

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