Математические методы верификации схем и программ

Инструкция по использованию средства SPIN

для верификации LTL-требований

Захаров В.А.
Подымов В.В.

Все вопросы писать на почту

valdus@yandex.ru

Windows vs. Linux

Spin – мультиплатформенное средство

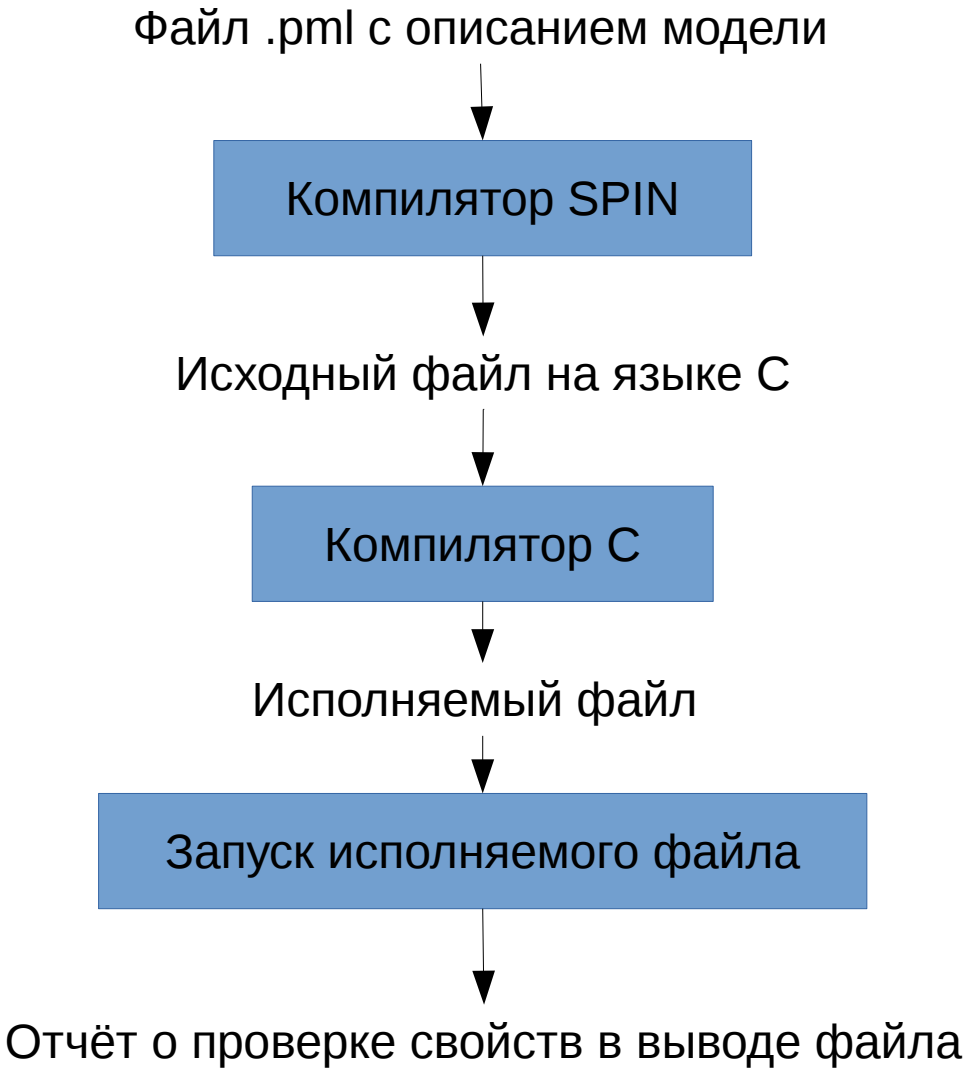
И в Windows, и в Linux доступны как консольная утилита, так и GUI Spin

Данная инструкция ориентирована на пользователей **Linux**

Вроде бы в Windows всё делается аналогично,
однако детали запуска средства, само собой, могут отличаться

Общая схема работы SPIN

Работа средства SPIN для проверки LTL-свойств в общем виде выглядит так:

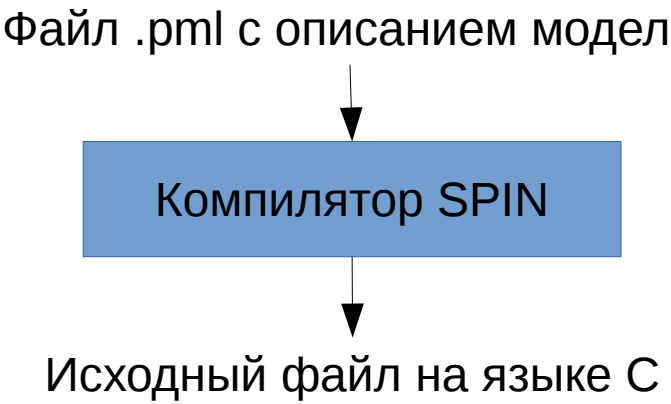


Консоль + Linux

Файл .pml с описанием модели

```
terminal> ls
ispin.tcl  spin  spin643_linux64  test.pml
terminal> cat test.pml
int a;
active proctype P() {
do
:: a < 3 -> a = a + 1;
:: a = 3 -> a = 0;
od
}
ltl f {[] (0 <= a & a <= 2)}
terminal>
```

Консоль + Linux



“-a” - создать файл “pan.c” с исходным кодом верификатора данной конкретной модели

По умолчанию создаётся верификатор в режиме проверки свойств живости

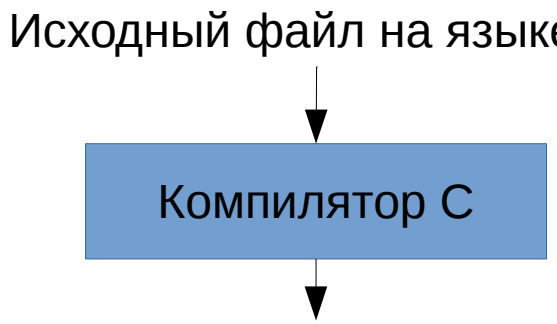
```
terminal> ./spin -a test.pml
ltl f: [] (((0<=a)&(a<=2)))
terminal> ls pan.c
pan.c
terminal>
```

Если уверены, что режима проверки свойств безопасности будет достаточно, то флаг “DSAFETY” переключает верификатор на этот режим

(*свойства безопасности проверяются быстрее с DSAFETY*)

```
terminal> ./spin -a -DSAFETY test.pml
ltl f: [] (((0<=a)&(a<=2)))
terminal> ls pan.c
pan.c
terminal>
```

Консоль + Linux




```
terminal> ls pan.c
pan.c
terminal> gcc -o pan pan.c
terminal> ls pan
pan
terminal>
```

Консоль + Linux

Исполняемый файл → Запуск исполняемого файла → Отчёт о проверке свойств в выводе файла

“-N <имя>” - проверить свойство, описанное как “ltl <имя>”

Свойство не выполнено

Проверяем свойство f

Про дополнительные опции читайте в документации

Статистика

```
terminal> ./pan -N f
warning: only one claim defined, -N ignored
warning: never claim + accept labels requires -a flag to fully verify
hint: this search is more efficient if pan.c is compiled -DSAFETY
pan:1: assertion violated  !( !((0<=a)&(a<=2)))) (at depth 12)
pan: wrote test.pml.trail

(Spin Version 6.4.3 -- 16 December 2014)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             + (f)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     - (not selected)
        invalid end states      - (disabled by never claim)

State-vector 28 byte, depth reached 12, errors: 1
        7 states, stored
        0 states, matched
        7 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.000       equivalent memory usage for states (stored*(State-vector + overhead))
    0.291       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage



pan: elapsed time 0 seconds
terminal>
```

GUI + Linux

Есть *как минимум* две популярных графических оболочки для SPIN

ispin

Это оболочка от разработчиков spin, она идёт в комплекте со средством

Оболочка основана на **tcl**

Она выглядит старомодно, но в ней есть всё, что нужно

О ней будут следующие слайды

jspin http://spinroot.com/spin/Man/README.html#S3

Это оболочка **не** от разработчиков spin

Оболочка основана на java

Она выглядит более современно, и *вроде бы* в ней тоже есть всё, что нужно

В ней всё делается примерно так же, как в ispin, так что о ней слайдов не будет

ispin + Linux

Чтобы оболочка ispin работала, необходимо сделать так, чтобы терминал понимал команду “spin”:

- поставить spin в стандартные системные папки, или
- прописать в нужную переменную окружения путь к spin

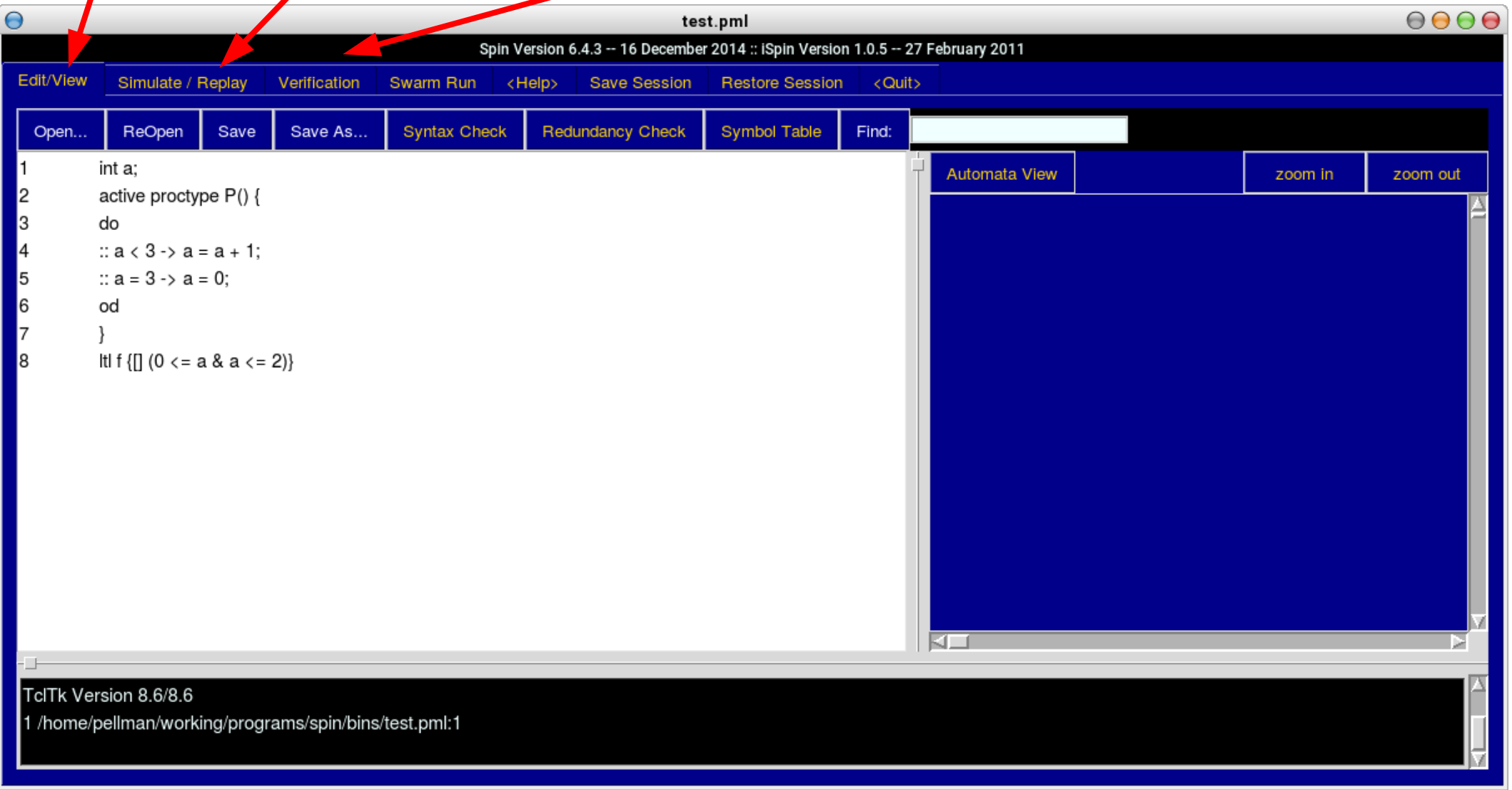
Оболочка запускается исполняемым файлом **ispin.tcl**

```
terminal> ls
ispin.tcl  spin  spin643_linux64  test.pml  test.pml.trail
terminal> spin
If 'spin' is not a typo you can use command-not-found to lookup the package that contains it, like this:
    cnf spin
terminal> export PATH=$PATH:$PWD
terminal> ispin.tcl
```

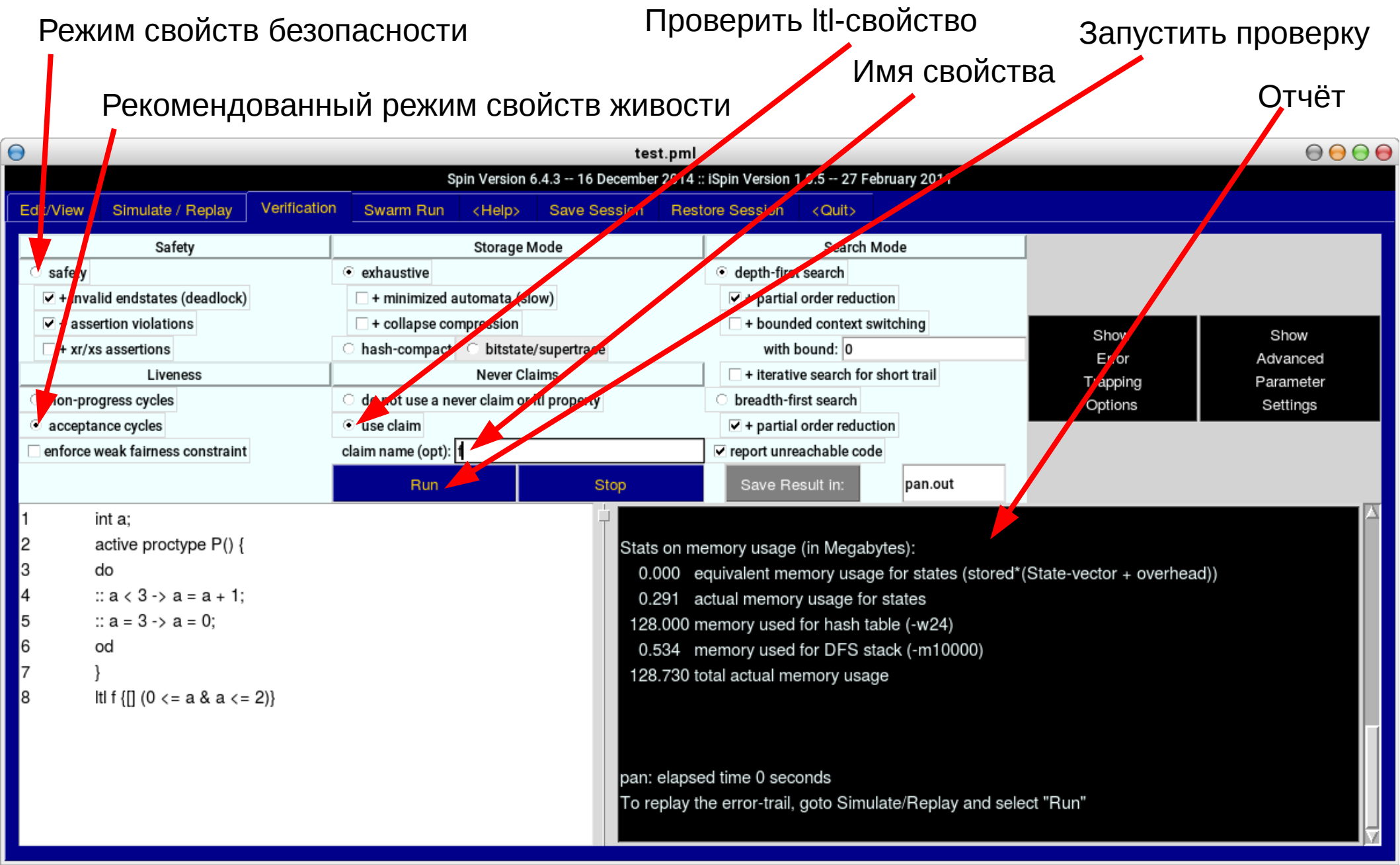
ispin

Всё довольно user-friendly

Редактор Симуляция Верификация



ispin


Режим свойств безопасности
Рекомендованный режим свойств живости
Проверить ltl-свойство
Имя свойства
Запустить проверку
Отчёт
test.pml
Edit/View
Simulate / Replay
Verification
Swarm Run
<Help>
Save Session
Restore Session
<Quit>
Safety
safety
+ invalid endstates (deadlock)
+ assertion violations
+ xr/xs assertions
Liveness
non-progress cycles
acceptance cycles
enforce weak fairness constraint
Storage Mode
exhaustive
+ minimized automata (slow)
+ collapse compression
hash-compact
bitstate/supertrace
Never Claims
do not use a never claim or ltl property
use claim
claim name (opt):
Search Mode
depth-first search
+ partial order reduction
+ bounded context switching
with bound: 0
+ iterative search for short trail
breadth-first search
+ partial order reduction
report unreachable code
Show Error Trapping Options
Show Advanced Parameter Settings
Run
Stop
Save Result in:
pan.out
1 int a;
2 active proctype P() {
3 do
4 :: a < 3 -> a = a + 1;
5 :: a = 3 -> a = 0;
6 od
7 }
8 ltl f {[] (0 <= a & a <= 2)}
Stats on memory usage (in Megabytes):
0.000 equivalent memory usage for states (stored*(State-vector + overhead))
0.291 actual memory usage for states
128.000 memory used for hash table (-w24)
0.534 memory used for DFS stack (-m10000)
128.730 total actual memory usage
pan: elapsed time 0 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"