МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное

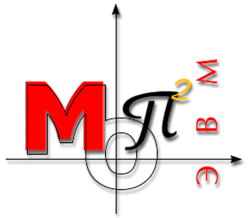
учреждение высшего профессионального образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Инженерно-технологическая академия**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра Математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

по лабораторной работе № 1

по курсу «Методология программной инженерии и верификация программного обеспечения»

Вариант \_

Выполнил:

студент группы КТмо1-3

Калиниченко Г.А.

Проверил:

преподаватель каф. МОП ЭВМ

Пирская Л.В.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

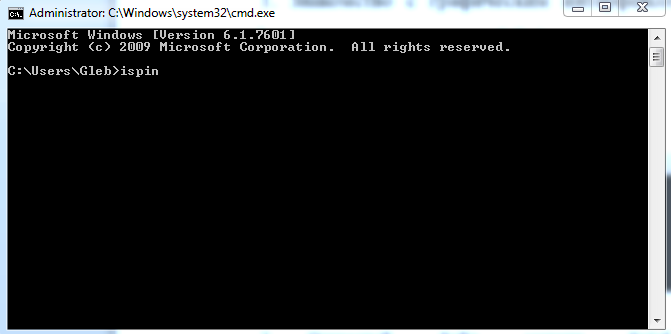
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2016 г.

Таганрог 2016

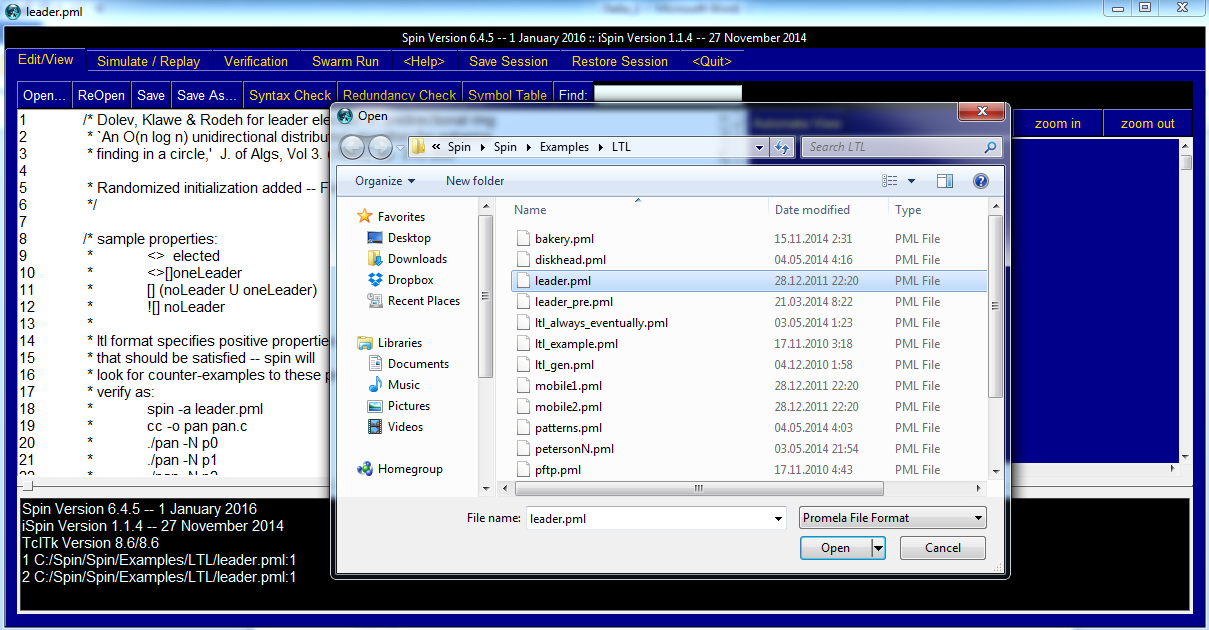
Цель работы: познакомится с системой верификации на основе моделей Spin, ее графическим интерфейсом и базовыми элементами

1. Знакомство с графическим интерфейсом к инструменту верификации на основе моделей Spin

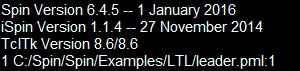
1. Запуск iSpin через команду в терминале



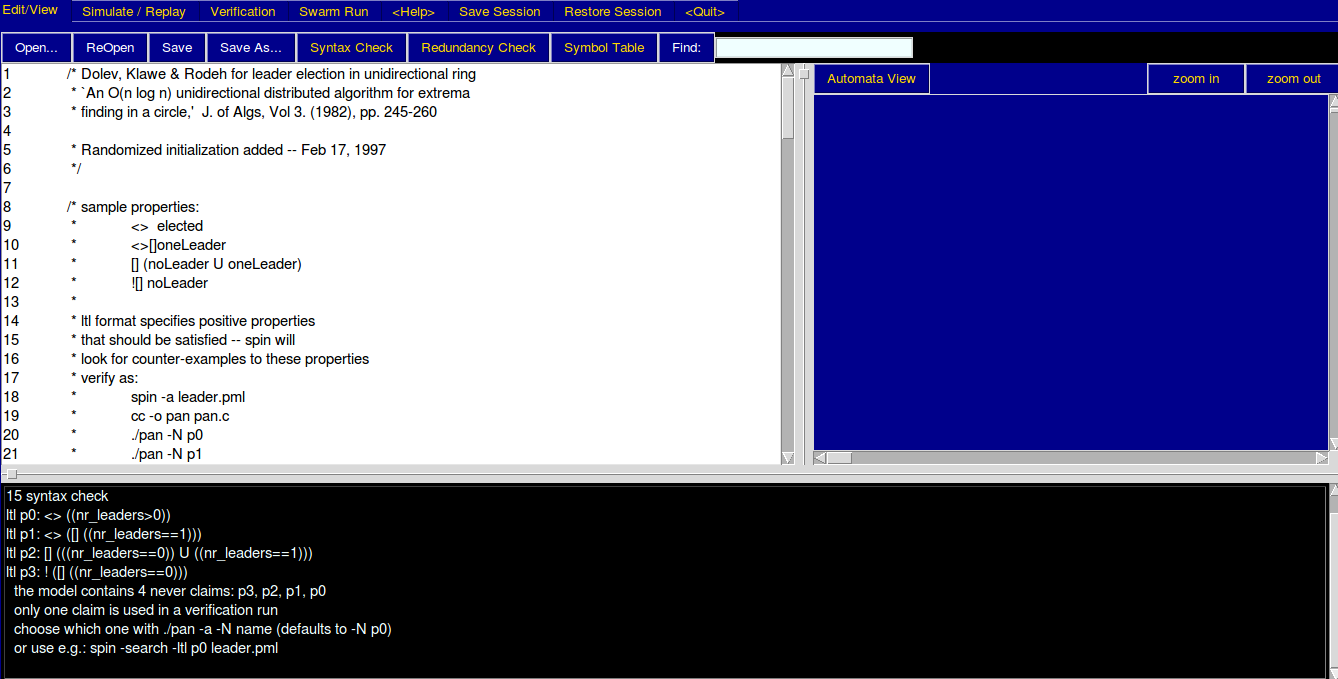
1. Открытый файл модели Spin/Spin/Examples/LTL/leader.pml (Edit/View→Open)



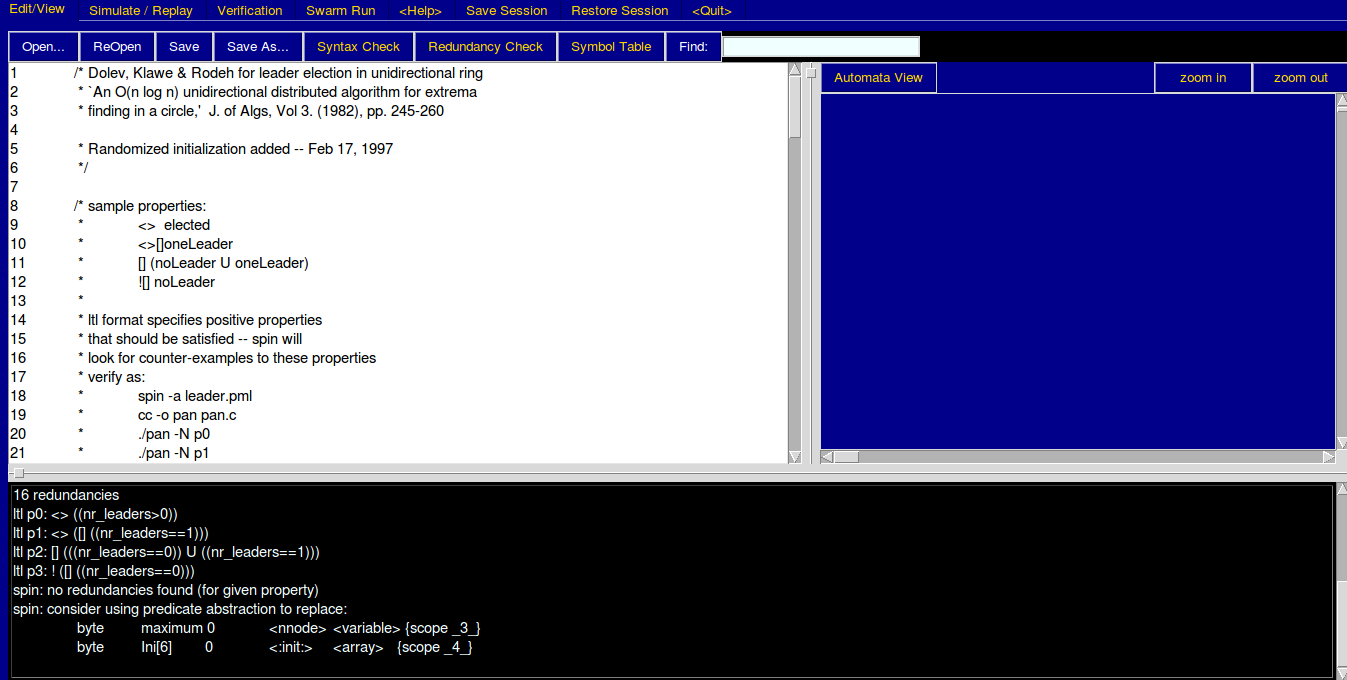
1. В логе команд iSpin (нижняя часть окна на вкладке Edit/View)



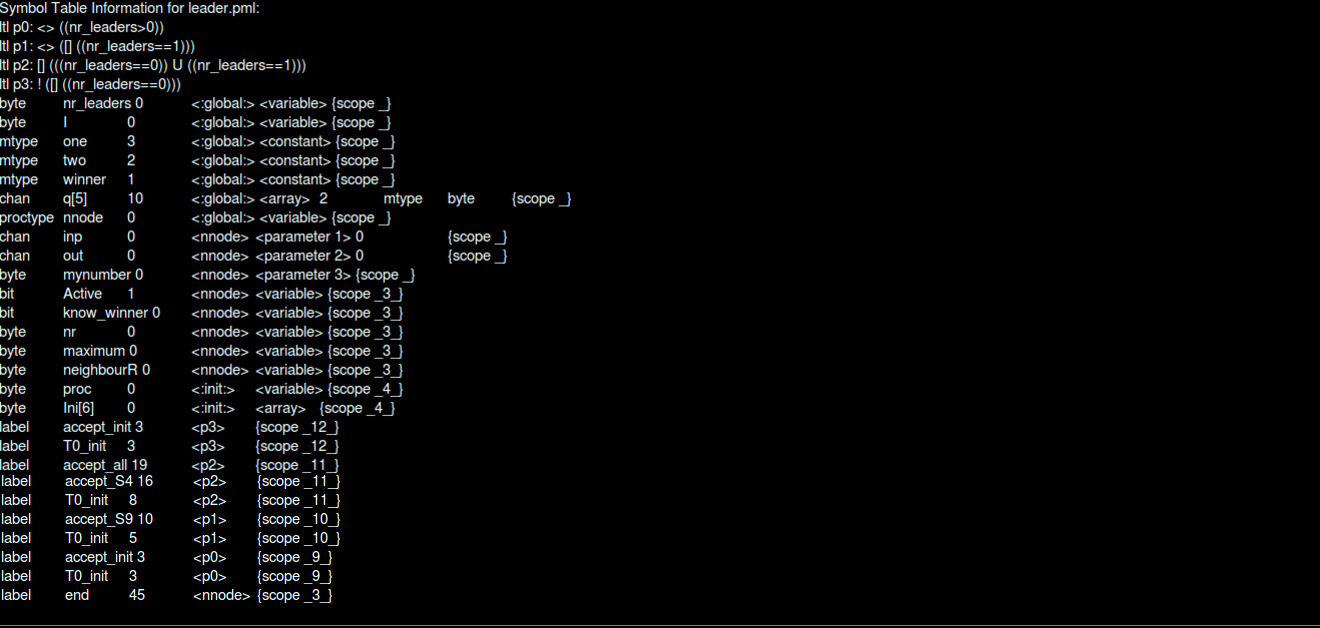
* 1. Для проверки синтаксиса модели нажмите Edit/View→Syntax Check



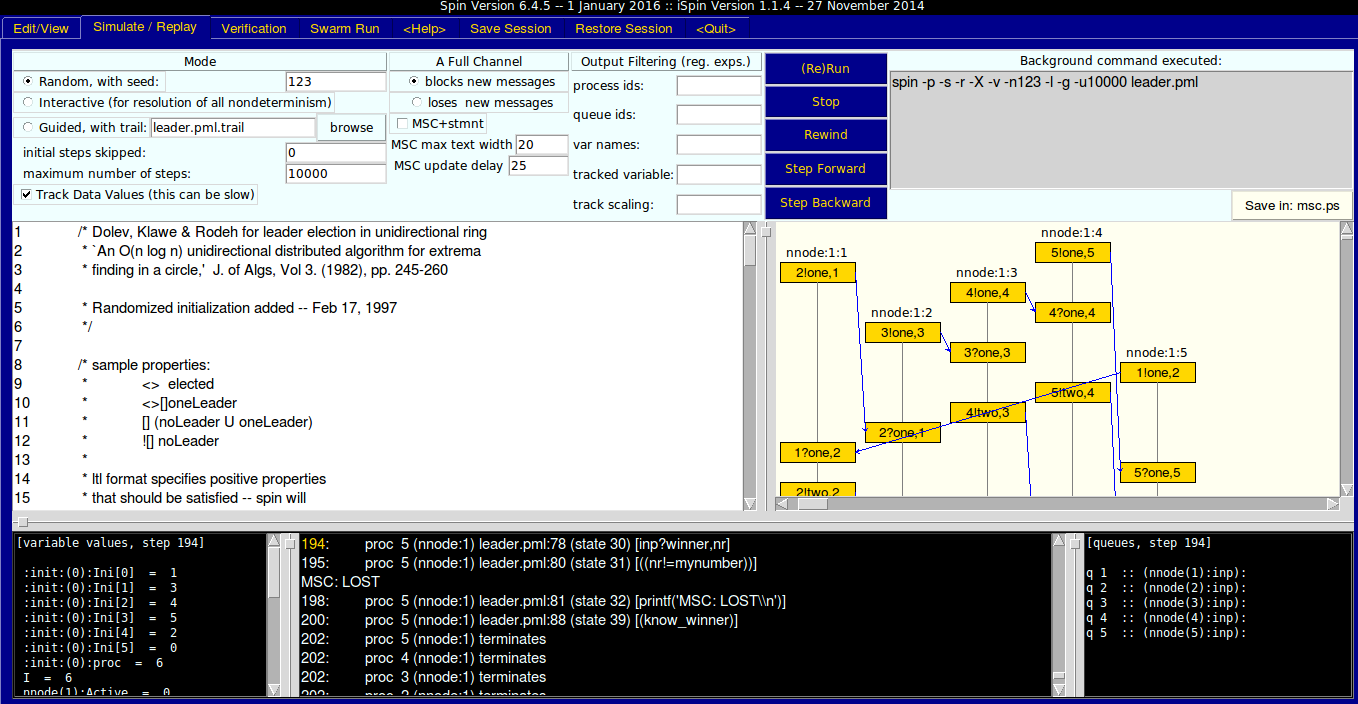
* 1. Проверка избыточности кода модели нажмите Edit/View→Redundancy Check



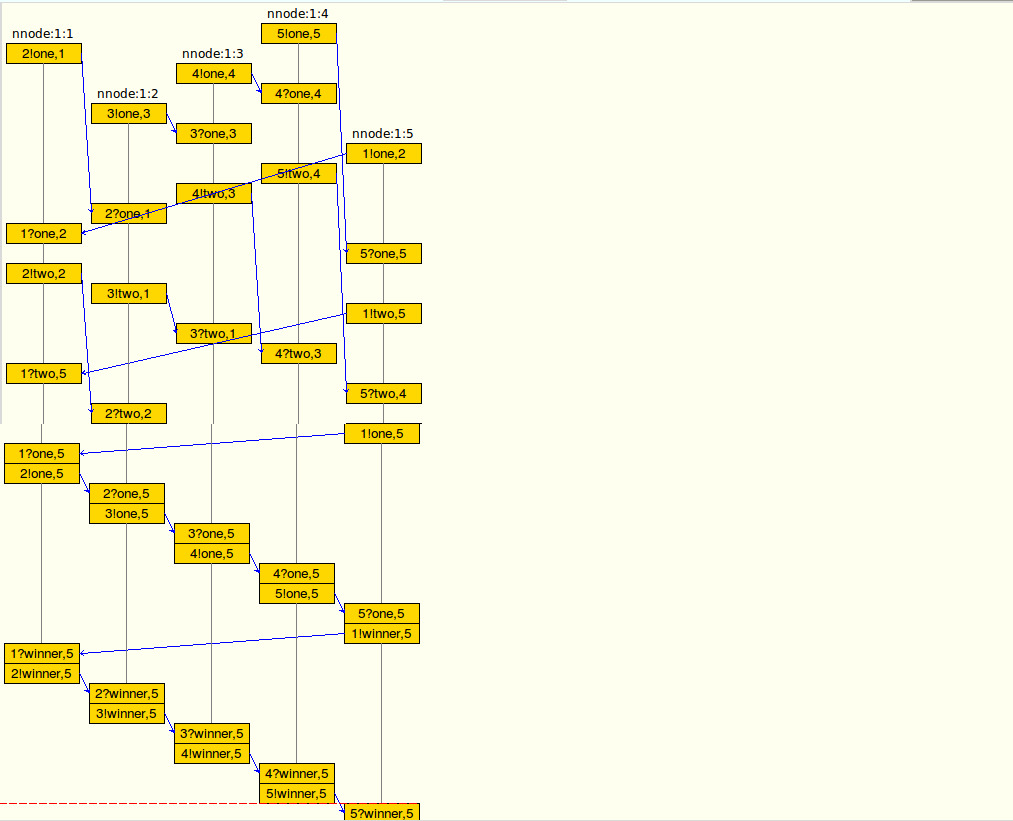
* 1. Log просмотра таблицы символов для спецификации модели, Edit/View→Symbol Table



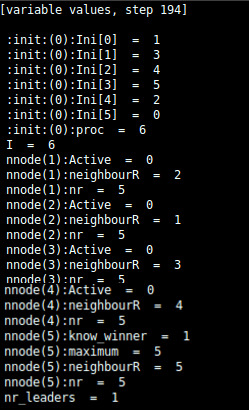
1. Вкладка Simulate/Replay



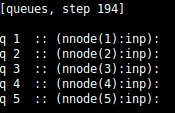
1. Представления Simulate/Replay
   1. Графическое представление последовательности сообщений



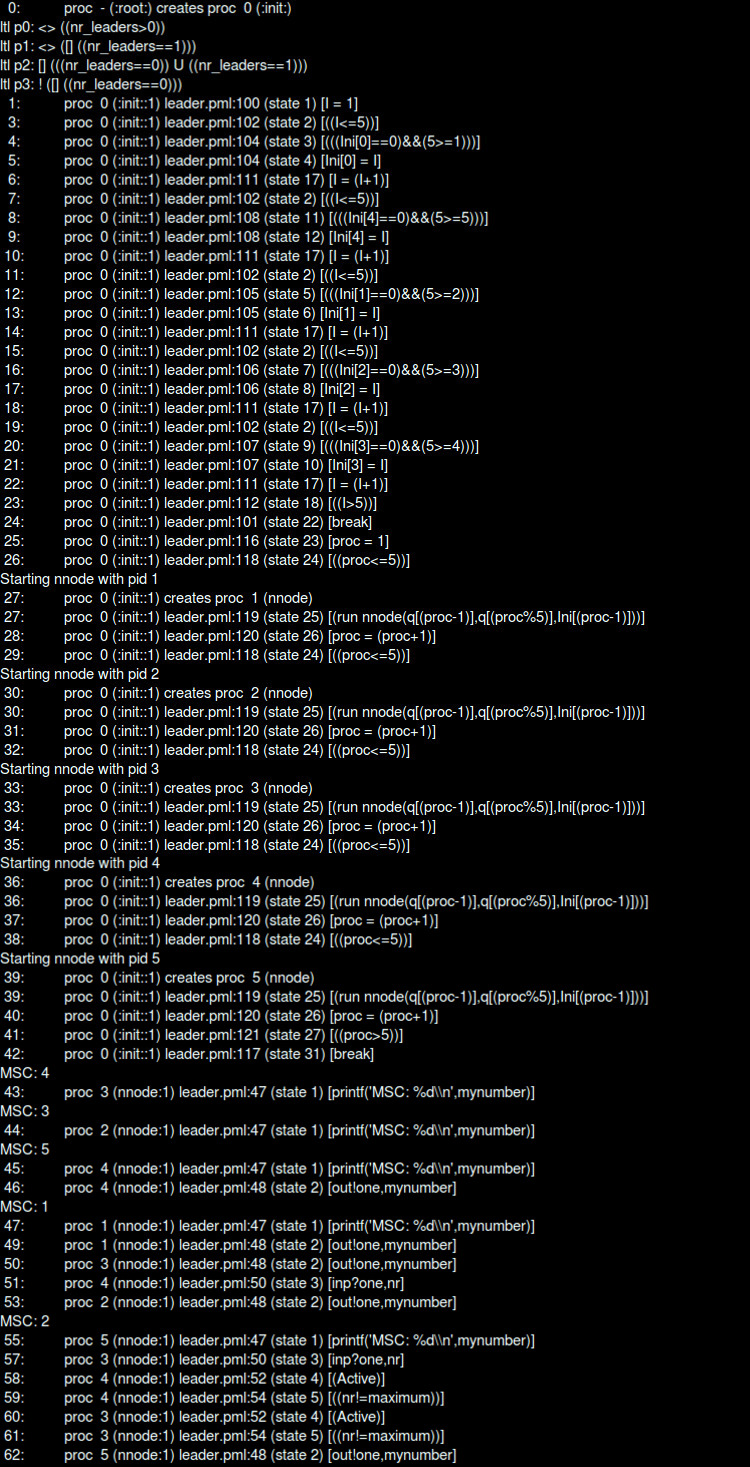
* 1. Значения переменных модели



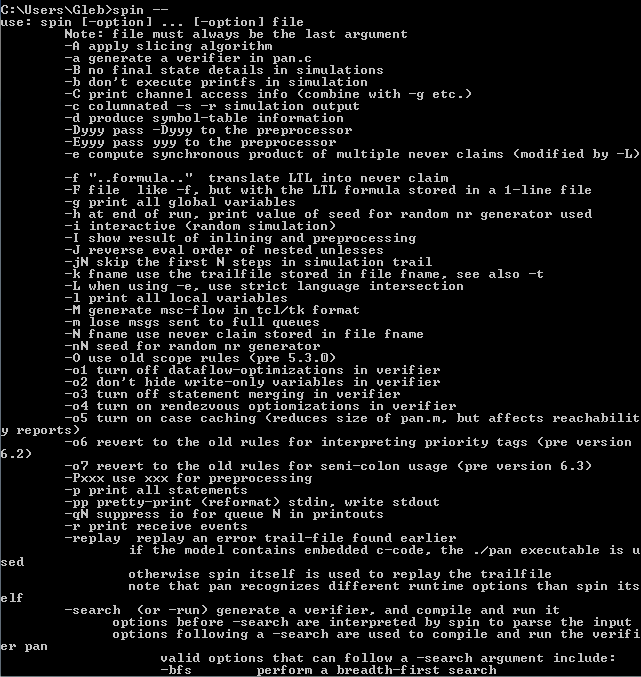
* 1. Содержимое очереди каналов



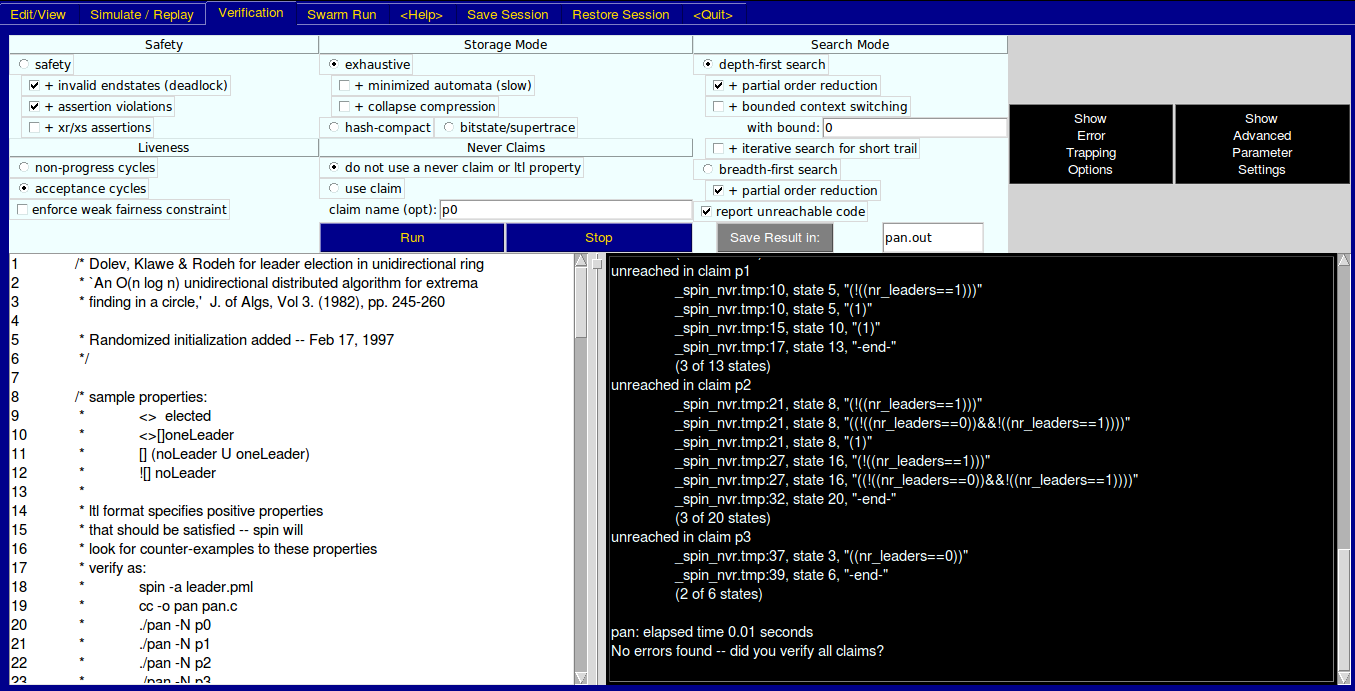
* 1. Результат симуляции



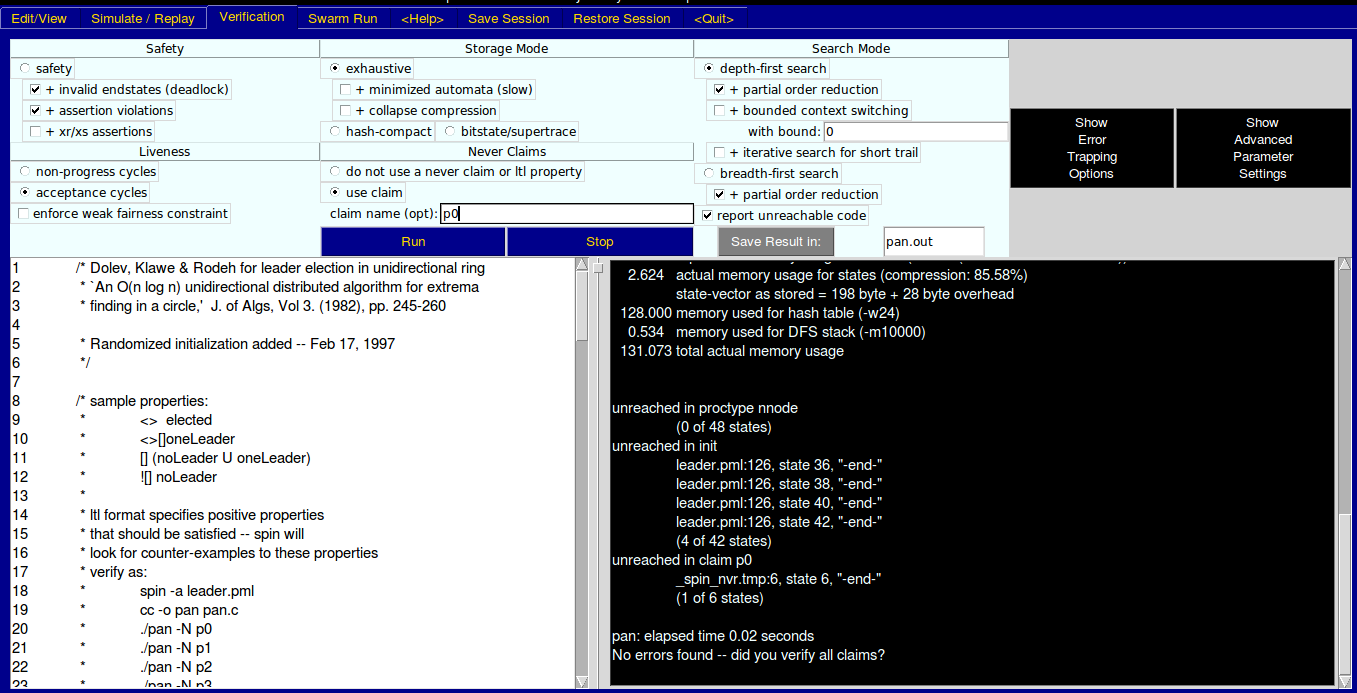
1. Опции верификатора spin



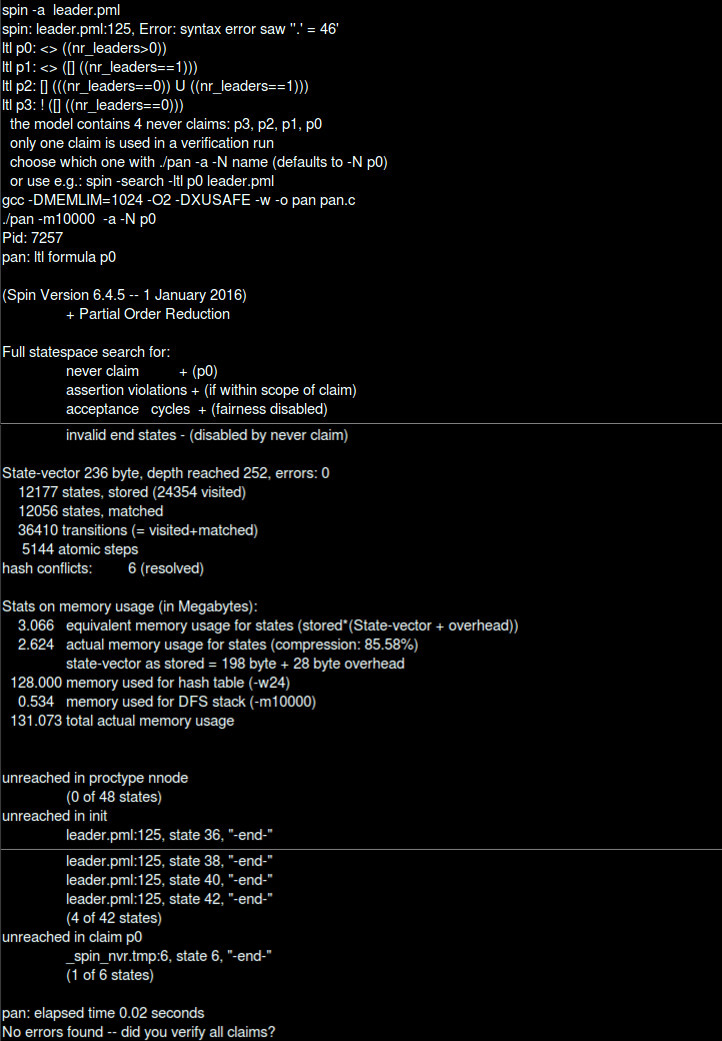
1. Результат верификации по умолчанию (свойства безопасности)



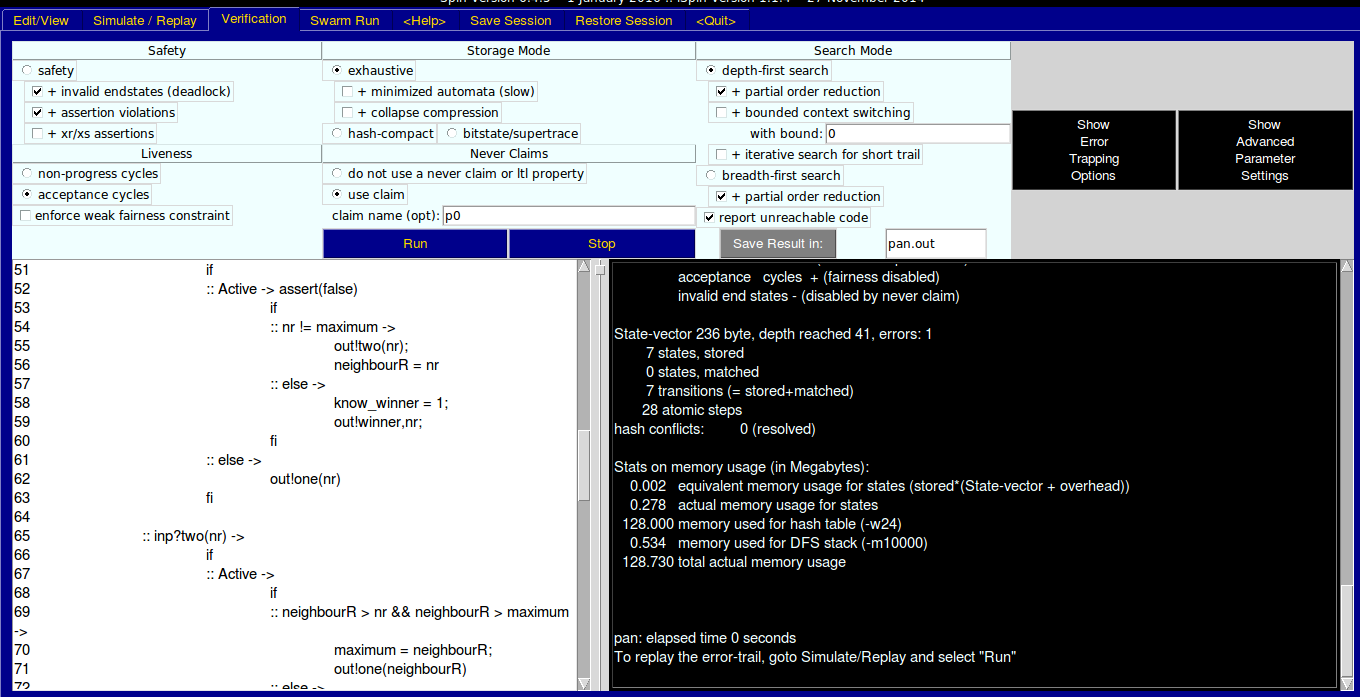
Результат верификации с использованием свойств живучести, для темпорального оператора p0



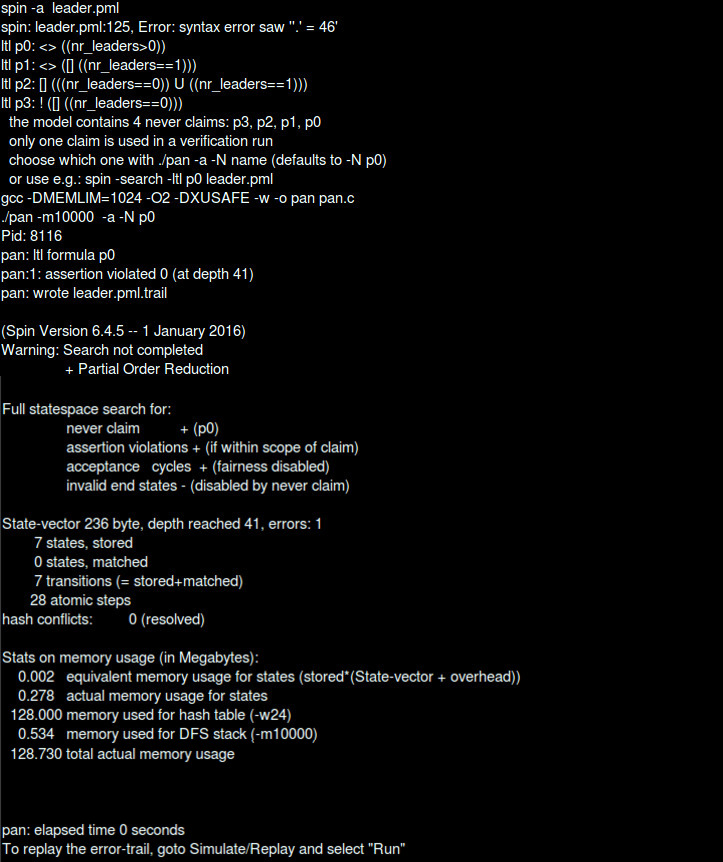
Весь лог для верификации с использованием свойств живучести, для темпорального оператора p0. При выполнении данной верификации не происходит верификация других операторов (p1, p2, p3).



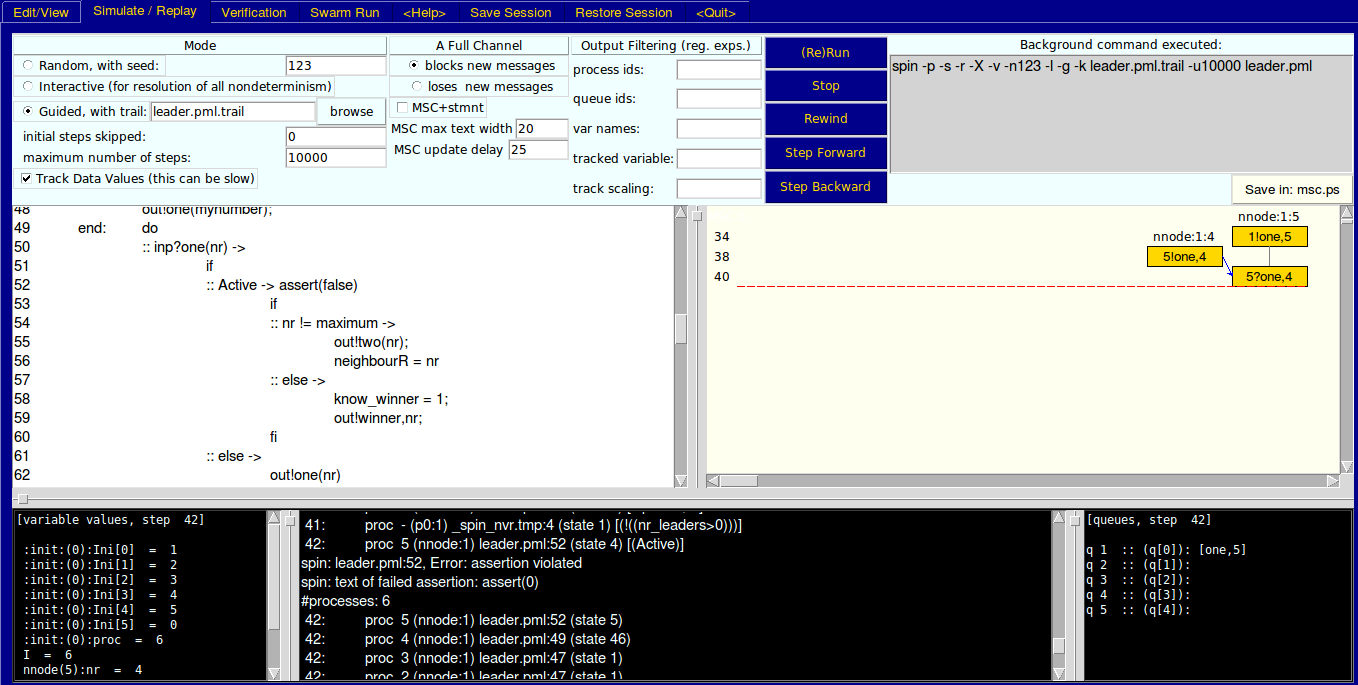
1. Добавленная ошибка ::Active -> assert(false)



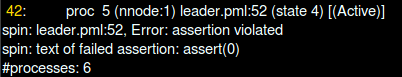
Полный лог верификации



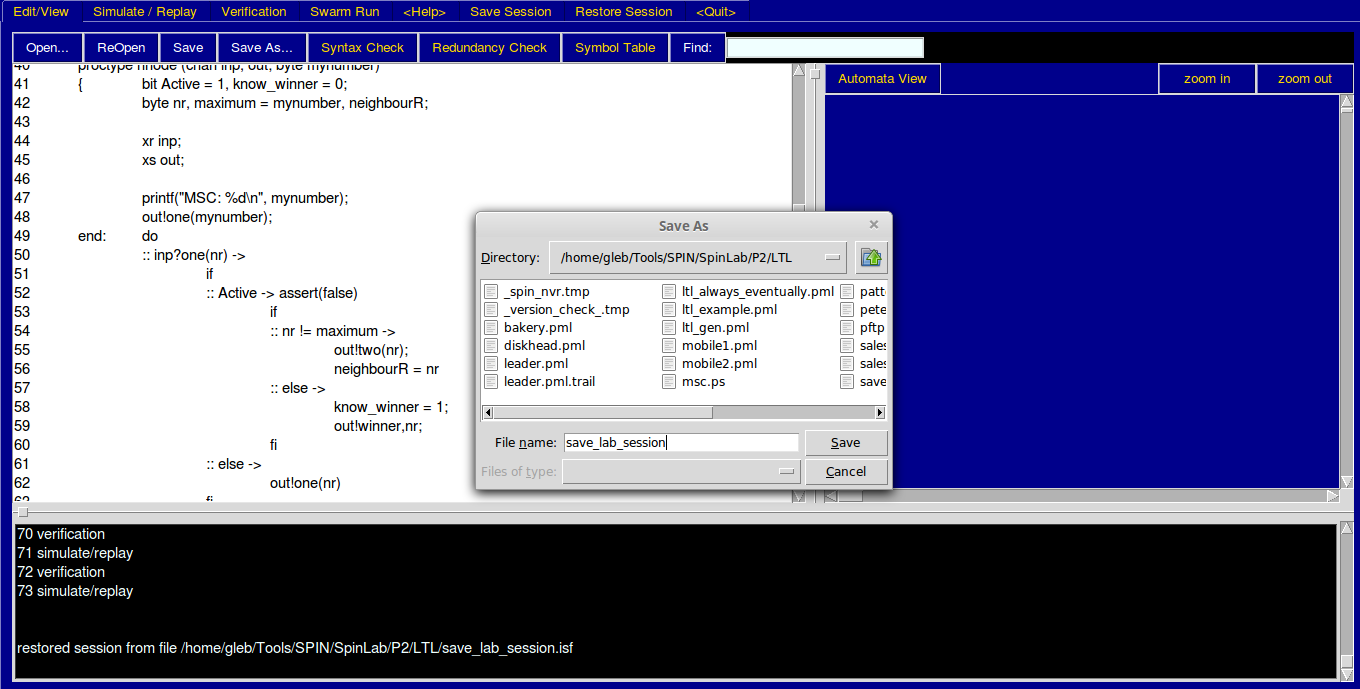
Повторная симуляция в ручном режиме, для обнаружения ошибки( Simulate/Replay -> Guided, with trail)



Сообщение в окне симуляции

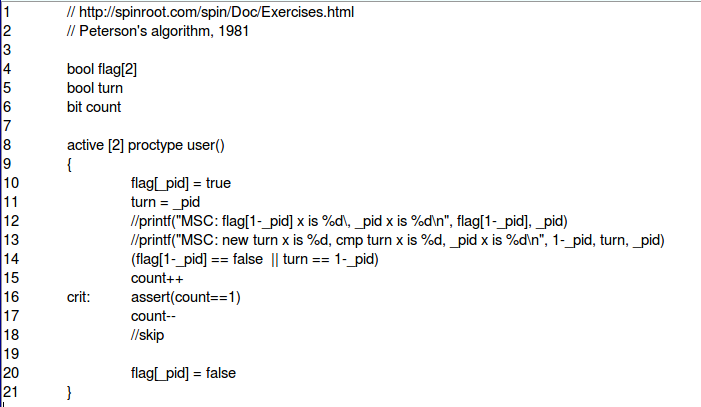


1. Сохранение настроек сессии верификации

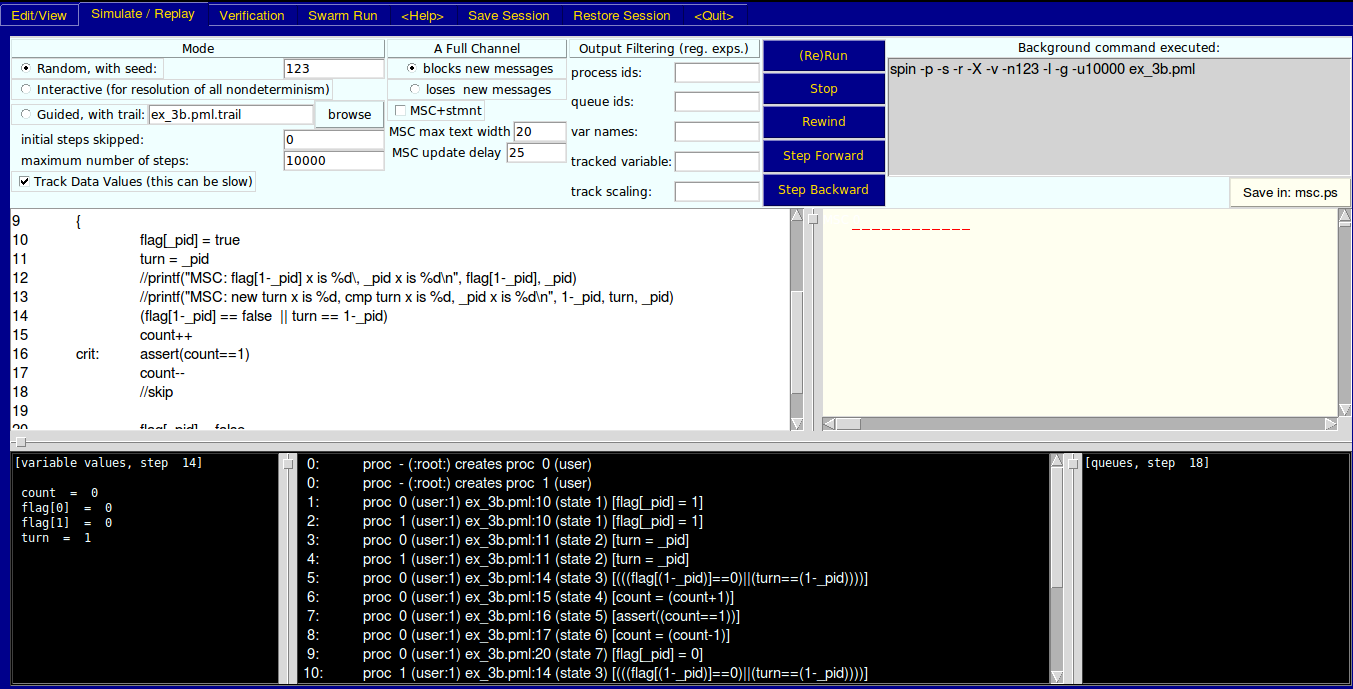


2. Знакомство с базовыми элементами процесса верификации с использованием верификатора Spin

Исходный код измененной модели алгоритма Петерсона



Измененная модель алгоритма Петерсона с логами



bool flag[2] //массив индикаторов заинтересованности процессов

bool turn //флаг очереди исполенения

bit count //количество процессов в критической секции

active [2] proctype user() //запуск сразу 2 процессов

{

flag[\_pid] = true //указываем, что заинтересованы в выполнении текущего процесса

turn = \_pid //исполняем текущий поток

(flag[1-\_pid] == false || turn == 1-\_pid) // цикл ожидания, если какой-то процесс уже занял критическую секцию (мы не заинтересованы в выполнении противоположного процесс ИЛИ если текущий поток и противоположный процесс являются исполняемым, то первых в критическую область войдет текущий процесс)

count++ //увеличиваем количество процессов

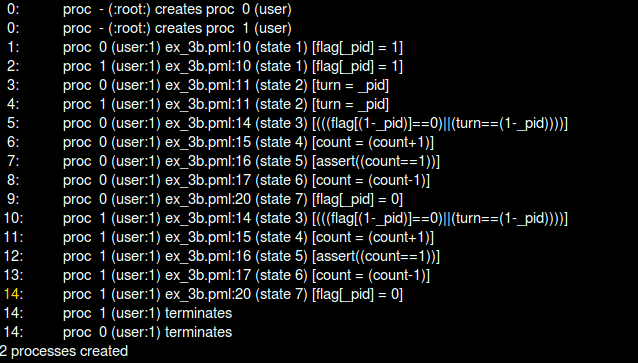
crit: assert(count==1) //сравниваем количество процессов в критической секции

count-- //при выходе из критической секции уменьшаем количество процессов

flag[\_pid] = false //сбрасываем флаг заинтересованности

}

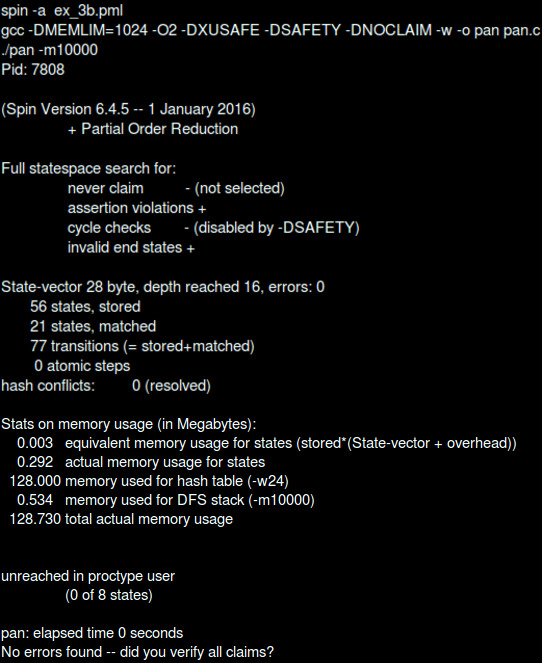
Лог сообщений симуляции



Первый создает 0 процесс, а затем 1 процесс. Устанавливаем флаги заинтересованности для этих процессов. Затем указываем что оба процесса будут исполняемыми. Т.к. оба процесса исполняемыми, то первый из процессов занимает критическую секцию, а второй процесс ожидает выполнения первого. После прохождения критической секции, сбрасываем флаг заинтересованности с первого процесса.

Лог верификации показывает, что ошибок при выполнении программы не найдено и происходит взаимное исключение процессов.

Дополнительные вопросы. Переменная cnt может иметь тип bit, т.к. для взаимного исключения процессов необходимо, чтобы в критической секции находился 1 процесс, а bit принимает значение 0..1. В случае, если значение bit превысит 1 (количество процессов в критической секции будет больше 1), система сгенерирует ошибку. Также переменная cnt может иметь тип int, но это приведет к неоправданному использованию памяти. Переменная cnt должна быть глобальной, чтобы можно было отследить количество процессов, которые будут находиться в критической секции.



ВЫВОДЫ

Познакомился с системой верификации на основе моделей Spin. Получил опыт работы с графическим интерфейсом системы Spin. Был изучен базовый синтаксис моделирующего языка Promela. В ходе выполнения верификации было осуществлено: проверка синтаксиса модели, проверка избыточности кода модели, симулация работы модели с различными характеристиками. Во время моделирования было изучено поведение системы в случае наличия в модели ошибок.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. И.В. Шошмина, Ю.Г. Карпов, Введение в язык Promela и систему

комплексной верификации Spin, СПбГПУ, 2009, 66 стр.

1. Algorithms for Mutual Exclusion
2. <http://spinroot.com/spin/Man/README.html>
3. <https://ru.wikipedia.org/wiki/Алгоритм_Петерсона>