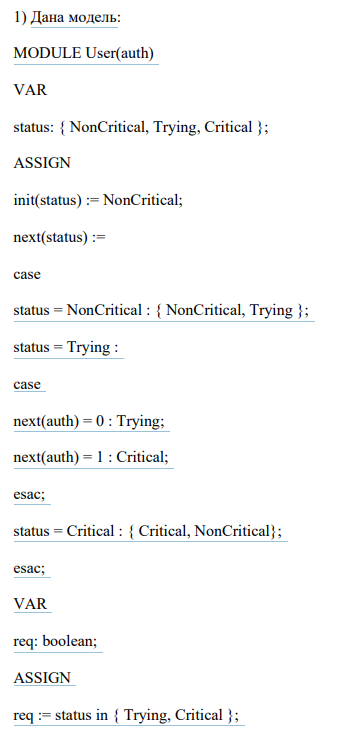
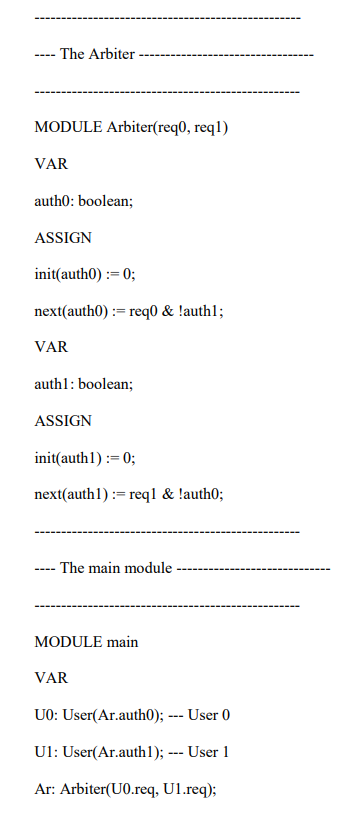
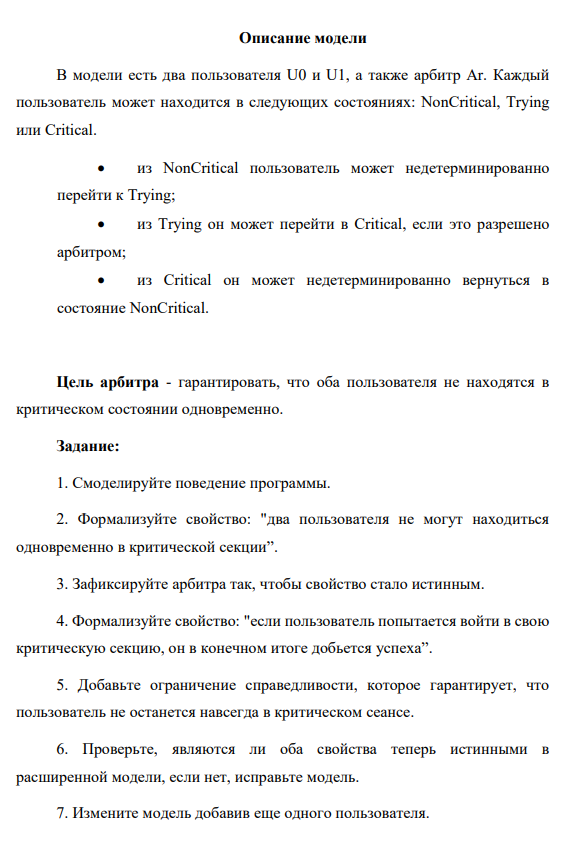
**Постановка задачи**



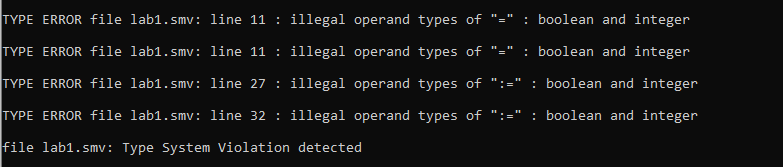




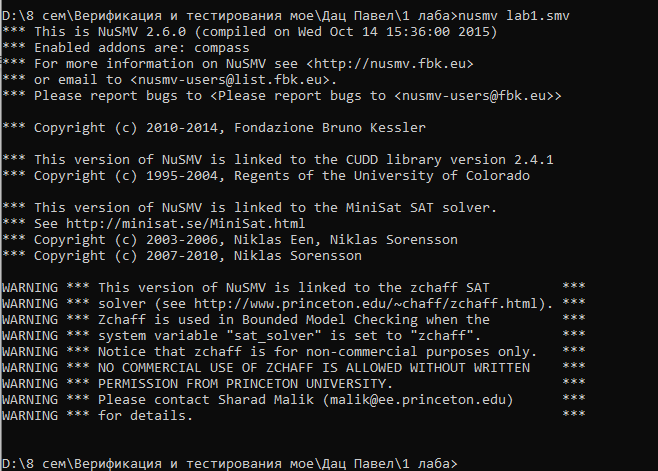
**Ход работы**

1. Модель была сохранена в файл “lab1.smv”.

Запускаем модель при помощи команды *“nusmv lab1.smv”*. Но обнажурились ошибки:



В строчках 11, 27 и 32 были обнаружены несоответствие типов. Заменим числа 0 и 1 на переменные False и True. Теперь протестируем, запускаем модель:

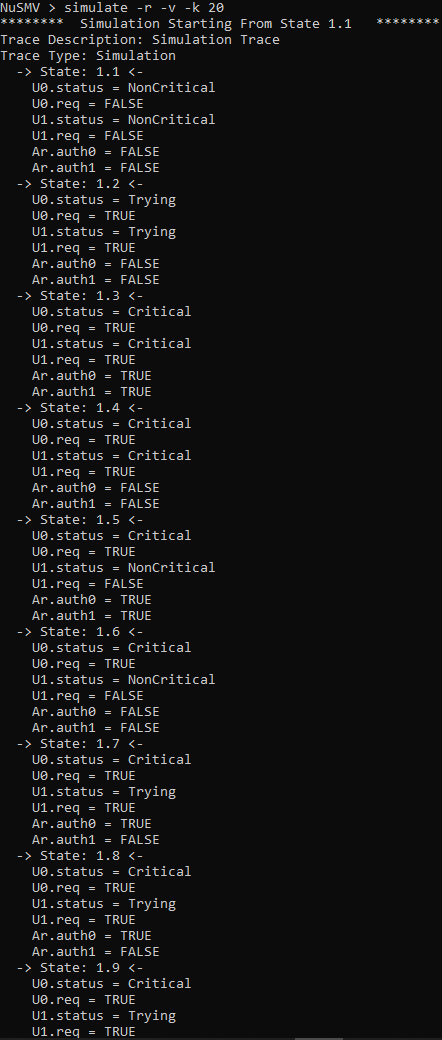


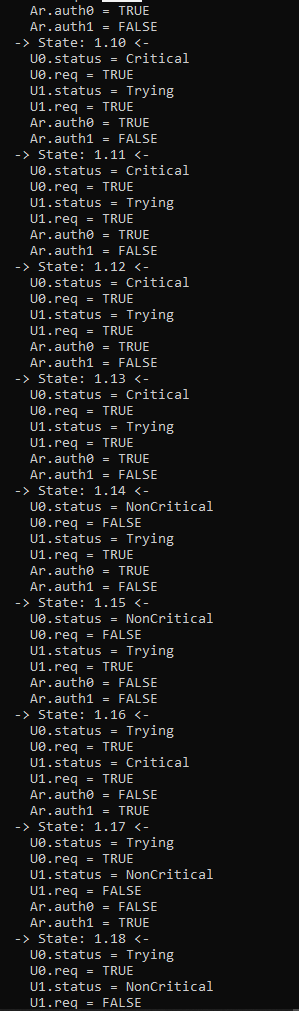
Как видим модель выполнилась без ошибок.

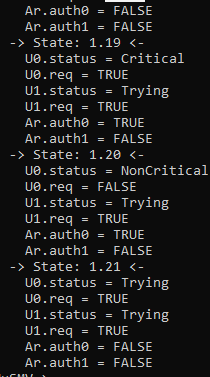
Для дальнейшего удобства введем команду: *nusmv -int RET.*

Далее при помощи команды **pick\_state -r** было выбрано случайное начальное состояние.

Смоделируем поведение программы на 15 шагов, при помощи команды **simulate -r -v -k 20**:



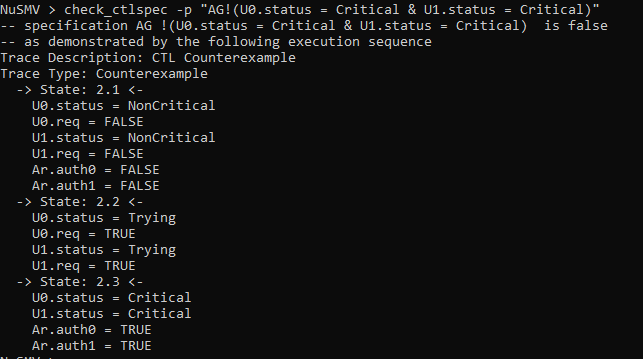




1. Было формализовано свойство взаимно исключающего доступа «два пользователя не могут находиться одновременно в критической секции»

**AG!(U0.status=Critical & U1.status=Critical)**

Проверим данное свойство:



Заметили, что данное свойство не выполнятся на исходной модели. Приведен контрпример, при котором оба пользователя находятся в критической секции одновременно.

1. Теперь зафиксируем арбитра так, чтобы свойство стало истинным.

Для этого исправим 28 строку модели, а именно:



Проверим снова предыдущее свойство:

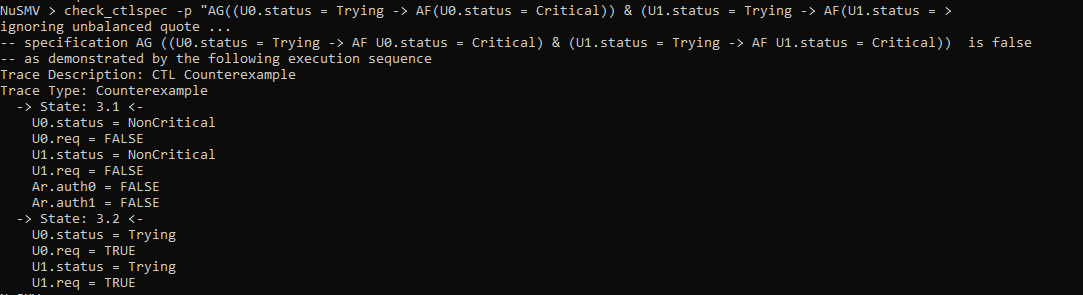


Видим, что теперь наша модель удовлетворяет данному свойству.

1. Формализуем свойство: «если пользователь попытается войти в свою критическую секцию, он в конечном итоге добьется успеха». Запишем данное свойство:

**AG((U0.status = Trying -> AF(U0.status = Critical)) & (U1.status = Trying -> AF(U1.status = Critical))).**

Проверим теперь данное свойство на нашей модели:



Заметим, что данное свойство не выполняется.

1. Добавим ограничение справедливости, которое гарантирует, что пользователь не останется навсегда в критическом сеансе:

**FAIRNESS**

**Status != Critical**

****

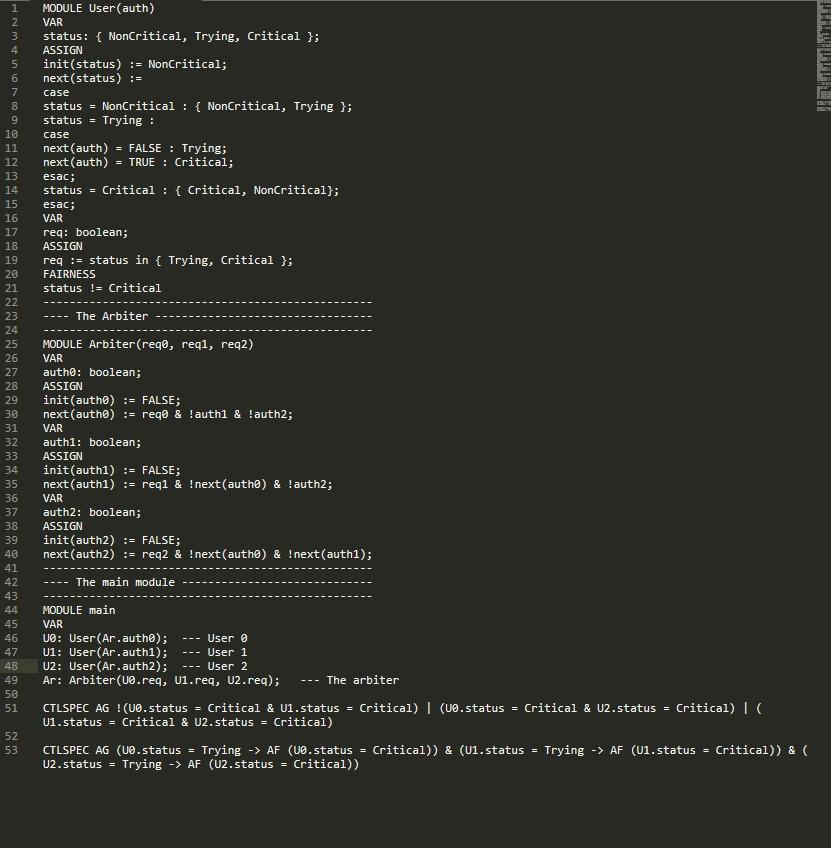
1. Проверим, являются ли оба свойства теперь истинными:



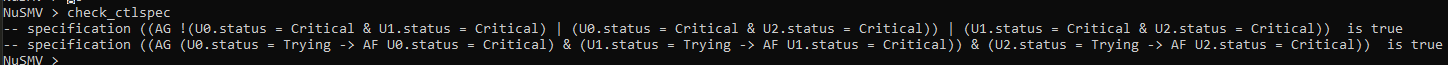


Видим, что модель удовлетворяет обоим свойствам.

1. Изменим модель, добавив еще одного пользователя



Проверим теперь наши свойства на новой модели



Модель удовлетворяет им.

Код программы lab1:

MODULE User(auth)

VAR

status: { NonCritical, Trying, Critical };

ASSIGN

init(status) := NonCritical;

next(status) :=

case

status = NonCritical : { NonCritical, Trying };

status = Trying :

case

next(auth) = FALSE : Trying;

next(auth) = TRUE : Critical;

esac;

status = Critical : { Critical, NonCritical};

esac;

VAR

req: boolean;

ASSIGN

req := status in { Trying, Critical };

FAIRNESS

status != Critical

--------------------------------------------------

---- The Arbiter ---------------------------------

--------------------------------------------------

MODULE Arbiter(req0, req1)

VAR

auth0: boolean;

ASSIGN

init(auth0) := FALSE;

next(auth0) := req0 & !auth1;

VAR

auth1: boolean;

ASSIGN

init(auth1) := FALSE;

next(auth1) := req1 & !next(auth0);

--------------------------------------------------

---- The main module -----------------------------

--------------------------------------------------

MODULE main

VAR

U0: User(Ar.auth0); --- User 0

U1: User(Ar.auth1); --- User 1

Ar: Arbiter(U0.req, U1.req);

CTLSPEC AG !(U0.status=Critical & U1.status=Critical)

CTLSPEC AG (U0.status = Trying -> AF(U0.status = Critical)) & (U1.status = Trying -> AF(U1.status = Critical))

Код программы lab1\_7:

MODULE User(auth)

VAR

status: { NonCritical, Trying, Critical };

ASSIGN

init(status) := NonCritical;

next(status) :=

case

status = NonCritical : { NonCritical, Trying };

status = Trying :

case

next(auth) = FALSE : Trying;

next(auth) = TRUE : Critical;

esac;

status = Critical : { Critical, NonCritical};

esac;

VAR

req: boolean;

ASSIGN

req := status in { Trying, Critical };

FAIRNESS

status != Critical

--------------------------------------------------

---- The Arbiter ---------------------------------

--------------------------------------------------

MODULE Arbiter(req0, req1, req2)

VAR

auth0: boolean;

ASSIGN

init(auth0) := FALSE;

next(auth0) := req0 & !auth1 & !auth2;

VAR

auth1: boolean;

ASSIGN

init(auth1) := FALSE;

next(auth1) := req1 & !next(auth0) & !auth2;

VAR

auth2: boolean;

ASSIGN

init(auth2) := FALSE;

next(auth2) := req2 & !next(auth0) & !next(auth1);

--------------------------------------------------

---- The main module -----------------------------

--------------------------------------------------

MODULE main

VAR

U0: User(Ar.auth0); --- User 0

U1: User(Ar.auth1); --- User 1

U2: User(Ar.auth2); --- User 2

Ar: Arbiter(U0.req, U1.req, U2.req); --- The arbiter

CTLSPEC AG !(U0.status = Critical & U1.status = Critical) | (U0.status = Critical & U2.status = Critical) | (U1.status = Critical & U2.status = Critical)

CTLSPEC AG (U0.status = Trying -> AF (U0.status = Critical)) & (U1.status = Trying -> AF (U1.status = Critical)) & (U2.status = Trying -> AF (U2.status = Critical))