**Приложение Б**

**Пример использования системы**

Запуск системы можно осуществить следующим образом:

java -cp proto-0.0.1.jar;proto-0.0.1-jar-with-dependencies.jar proto.Launcher

Welcome to Proto Compiler 0.0.1! [http://github.com/hisohito/proto-code]

USAGE:

java -cp proto-0.0.1.jar proto.Launcher [--target=<TARGET>] [--output=<OUTPUT\_FILE>] <SOURCE>

<SOURCE> is an proto source file

<TARGET> is one of supported targets (promela by default)

<OUTPUT\_FILE> is an output file name (out.pml by default)

Возьмём в качестве примера описанный ранее пример про имитацию работы автоматической коробки передач:

interface Driver { number press(number boxState);}

class DefaultDriver() < Driver {

number press(number boxState){

return random boxState;

}

}

class AutoGear() << Prototype {

Driver driver = new DefaultDriver();

number cmd;

state finalize() { return "End of work"; }

state sleep() { return neutral(); }

state neutral(){

number cmd = driver.press(3);

if (cmd == 1){ return finalize();

} else if (cmd == 2){ return reverse();

} else { return gear1();}

}

state reverse(){ return neutral(); }

state gear1(){

cmd = driver.press(2);

if (cmd == 1){ return neutral();

} else { return gear2(); }

}

state gear2(){

cmd = driver.press(2);

if (cmd == 1){ return gear1();

} else { return gear3(); }

}

state gear3(){

cmd = driver.press(2);

if (cmd == 1){ return gear2();

} else { return gear4(); }

}

state gear4(){

cmd = driver.press(2);

if (cmd == 1){ return gear3();

} else { return gear5(); }

}

state gear5(){ return gear4(); }

state main(){ return sleep(); }

}

prototype AutoGear() { finally {AutoGear::finalize}; }

Запустим систему, с указанием входного файла autogear.proto:

java -cp proto-0.0.1.jar;proto-0.0.1-jar-with-dependencies.jar proto.Launcher "../examples/autogear.proto"

В результате запуска был создан файл protoOutput0.pml:

int stateA1;

#define AUTOGEAR\_\_GEAR5 0

#define AUTOGEAR\_\_NEUTRAL 1

#define AUTOGEAR\_\_REVERSE 2

#define AUTOGEAR\_\_MAIN 3

#define AUTOGEAR\_\_FINALIZE 4

#define AUTOGEAR\_\_GEAR1 5

#define AUTOGEAR\_\_GEAR2 6

#define AUTOGEAR\_\_SLEEP 7

#define AUTOGEAR\_\_GEAR3 8

#define AUTOGEAR\_\_GEAR4 9

inline A1() {

stateA1 = AUTOGEAR\_\_MAIN;

do

:: ( stateA1 == AUTOGEAR\_\_GEAR5 ) ->

printf("AutoGear::gear5");

if

::stateA1 = AUTOGEAR\_\_GEAR4;

fi;

:: ( stateA1 == AUTOGEAR\_\_NEUTRAL ) ->

printf("AutoGear::neutral");

if

::stateA1 = AUTOGEAR\_\_REVERSE;

::stateA1 = AUTOGEAR\_\_FINALIZE;

::stateA1 = AUTOGEAR\_\_GEAR1;

fi;

:: ( stateA1 == AUTOGEAR\_\_REVERSE ) ->

printf("AutoGear::reverse");

if

::stateA1 = AUTOGEAR\_\_NEUTRAL;

fi;

:: ( stateA1 == AUTOGEAR\_\_MAIN ) ->

printf("AutoGear::main");

if

::stateA1 = AUTOGEAR\_\_SLEEP;

fi;

:: ( stateA1 == AUTOGEAR\_\_FINALIZE ) ->

printf("AutoGear::finalize");

break;

:: ( stateA1 == AUTOGEAR\_\_GEAR1 ) ->

printf("AutoGear::gear1");

if

::stateA1 = AUTOGEAR\_\_NEUTRAL;

::stateA1 = AUTOGEAR\_\_GEAR2;

fi;

:: ( stateA1 == AUTOGEAR\_\_GEAR2 ) ->

printf("AutoGear::gear2");

if

::stateA1 = AUTOGEAR\_\_GEAR1;

::stateA1 = AUTOGEAR\_\_GEAR3;

fi;

:: ( stateA1 == AUTOGEAR\_\_SLEEP ) ->

printf("AutoGear::sleep");

if

::stateA1 = AUTOGEAR\_\_NEUTRAL;

fi;

:: ( stateA1 == AUTOGEAR\_\_GEAR3 ) ->

printf("AutoGear::gear3");

if

::stateA1 = AUTOGEAR\_\_GEAR2;

::stateA1 = AUTOGEAR\_\_GEAR4;

fi;

:: ( stateA1 == AUTOGEAR\_\_GEAR4 ) ->

printf("AutoGear::gear4");

if

::stateA1 = AUTOGEAR\_\_GEAR5;

::stateA1 = AUTOGEAR\_\_GEAR3;

fi;

od;

}

proctype Model() { A1(); }

init { run Model(); }

never { /\* !(<>{stateA1 == AUTOGEAR\_\_FINALIZE}) \*/

accept\_init:

T0\_init: if

:: (! ((stateA1 == AUTOGEAR\_\_FINALIZE))) -> goto T0\_init fi;

}

Используя полученную модель, можем запустить SPIN на верификацию:

spin -a protoOutput0.pml

SPIN генерирует файл pan.c, представляющий собой верификатор для данной модели на языке C.

Скомпилируем полученный файл с помощью компилятора gcc:

gcc –o”autogear.proto” pan.c