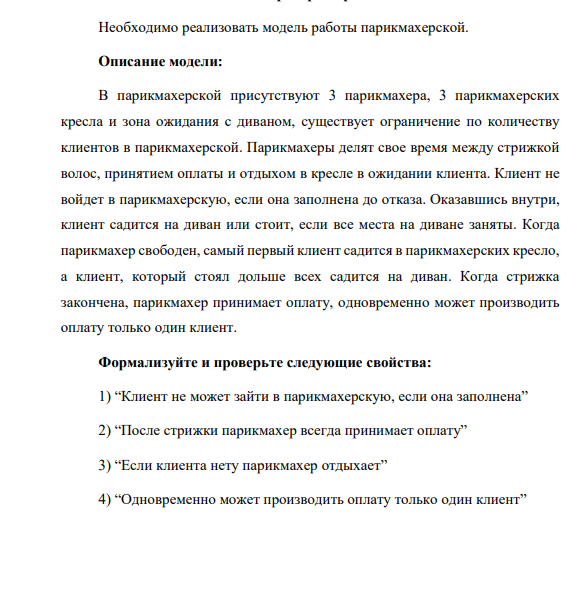
**Постановка задачи**



**Решение**

1. «Клиент не может зайти в парикмахерскую, если она заполнена»

***AG (BarberShop.spotsCounter = 6 -> BarberShop.customerInQueue = FALSE)***

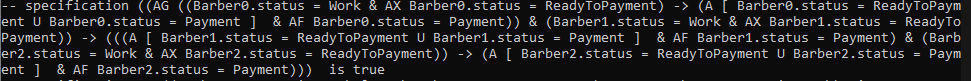
Свойство выполняется

1. «После стрижки парикмахер всегда принимает оплату»

***CTLSPEC AG ((Barber0.status = Work & AX (Barber0.status = ReadyToPayment)) -> (A[Barber0.status = ReadyToPayment U Barber0.status = Payment] & AF(Barber0.status = Payment))) &***

***(Barber1.status = Work & AX (Barber1.status = ReadyToPayment)) -> (A[Barber1.status = ReadyToPayment U Barber1.status = Payment] & AF(Barber1.status = Payment)) &***

***(Barber2.status = Work & AX (Barber2.status = ReadyToPayment)) -> (A[Barber2.status = ReadyToPayment U Barber2.status = Payment] & AF(Barber2.status = Payment))***

****

Свойство выполняется

1. «Если клиента нету парикмахер отдыхает»

***CTLSPEC AG ((Barber0.status != Work & BarberShop.spotsCounter = 0) -> AX (Barber0.status != Work)) &***

***(Barber1.status != Work & BarberShop.spotsCounter = 0) -> AX (Barber1.status != Work) &***

***(Barber2.status != Work & BarberShop.spotsCounter = 0) -> AX (Barber2.status != Work)***



Свойство выполняется

1. «Одновременно может производить оплату только один клиент»

***CTLSPEC AG (Barber0.status = Payment -> Barber1.status != Payment & Barber2.status != Payment) & (Barber1.status = Payment -> Barber0.status != Payment & Barber2.status != Payment)***

***& (Barber2.status = Payment -> Barber0.status != Payment & Barber1.status != Payment)***

******

**Код программы**

MODULE Barber(workPermission, paymentPermission)

VAR

status: { Sleep, Work, Payment, ReadyToPayment};

ASSIGN

init(status) := Sleep;

next(status) :=

case

next(workPermission) = TRUE : Work;

status = Work : {Work, ReadyToPayment};

status = ReadyToPayment & next(paymentPermission) = TRUE : Payment;

status = ReadyToPayment & next(paymentPermission) = FALSE : ReadyToPayment;

TRUE : Sleep;

esac;

JUSTICE !(status = Work)

MODULE BarberShop(Barber0,Barber1,Barber2)

VAR

sofa : array 0..2 of boolean;

standing: array 0..2 of boolean;

workPermission : array 0..2 of boolean;

paymentPermission : array 0..2 of boolean;

customerArrived : boolean;

customerInQueue : boolean;

ASSIGN

init(sofa[0]) := FALSE;

init(sofa[1]) := FALSE;

init(sofa[2]) := FALSE;

init(standing[0]) := FALSE;

init(standing[1]) := FALSE;

init(standing[2]) := FALSE;

init(workPermission[0]) :=FALSE;

init(workPermission[1]) :=FALSE;

init(workPermission[2]) :=FALSE;

init(paymentPermission[0]) := FALSE;

init(paymentPermission[1]) := FALSE;

init(paymentPermission[2]) := FALSE;

init(customerArrived) := FALSE;

init(customerInQueue) := FALSE;

next(customerArrived) :=

case

customerArrived = FALSE: {TRUE, FALSE};

customerArrived = TRUE : {TRUE, FALSE};

esac;

next(customerInQueue) :=

case

next(customerArrived) & next(spotsCounter) < 6 : TRUE;

TRUE: FALSE;

esac;

next(workPermission[0]) :=

case

spotsCounter > 0 & (Barber0.status = Sleep | Barber0.status = Payment) : TRUE;

TRUE: FALSE;

esac;

next(workPermission[1]) :=

case

spotsCounter > 0 & next(workPermission[0]) = FALSE & (Barber1.status = Sleep | Barber1.status = Payment): TRUE;

TRUE: FALSE;

esac;

next(workPermission[2]) :=

case

spotsCounter > 0 & next(workPermission[0]) = FALSE & next(workPermission[1]) = FALSE & (Barber2.status = Sleep | Barber2.status = Payment) : TRUE;

TRUE: FALSE;

esac;

next(paymentPermission[0]) :=

case

Barber0.status = ReadyToPayment: TRUE;

TRUE: FALSE;

esac;

next(paymentPermission[1]) :=

case

Barber1.status = ReadyToPayment & next(Barber0.status) != Payment: TRUE;

TRUE: FALSE;

esac;

next(paymentPermission[2]) :=

case

Barber2.status = ReadyToPayment & next(Barber1.status) != Payment & next(Barber0.status) != Payment: TRUE;

TRUE: FALSE;

esac;

next(sofa[0]) :=

case

spotsCounter = 0 & customerInQueue = TRUE : TRUE;

spotsCounter = 1 & customerInQueue = FALSE & shift = TRUE: FALSE;

TRUE: sofa[0];

esac;

next(sofa[1]) :=

case

spotsCounter = 1 & customerInQueue = TRUE & shift = FALSE : TRUE;

spotsCounter = 2 & customerInQueue = FALSE & shift = TRUE : FALSE;

TRUE: sofa[1];

esac;

next(sofa[2]) :=

case

spotsCounter = 2 & customerInQueue = TRUE & shift = FALSE : TRUE;

spotsCounter = 3 & customerInQueue = FALSE & shift = TRUE : FALSE;

TRUE: sofa[2];

esac;

next(standing[0]) :=

case

spotsCounter = 3 & customerInQueue = TRUE & shift = FALSE : TRUE;

spotsCounter = 4 & customerInQueue = FALSE & shift = TRUE : FALSE;

TRUE: standing[0];

esac;

next(standing[1]) :=

case

spotsCounter = 4 & customerInQueue = TRUE & shift = FALSE : TRUE;

spotsCounter = 5 & customerInQueue = FALSE & shift = TRUE : FALSE;

TRUE: standing[1];

esac;

next(standing[2]) :=

case

spotsCounter = 5 & customerInQueue = TRUE & shift = FALSE : TRUE;

spotsCounter = 6 & customerInQueue = FALSE & shift = TRUE : FALSE;

TRUE: standing[2];

esac;

DEFINE

spotsCounter := count(sofa[0], sofa[1], sofa[2], standing[0], standing[1], standing[2]);

shift := next(workPermission[0]) | next(workPermission[1]) | next(workPermission[2]);

JUSTICE (customerArrived != FALSE)

MODULE main

VAR

Barber0:Barber(BarberShop.workPermission[0], BarberShop.paymentPermission[0]);

Barber1:Barber(BarberShop.workPermission[1], BarberShop.paymentPermission[1]);

Barber2:Barber(BarberShop.workPermission[2], BarberShop.paymentPermission[2]);

BarberShop: BarberShop(Barber0,Barber1,Barber2);

-- Клиент не может зайти в парикмахерскую, если она заполнена

CTLSPEC AG (BarberShop.spotsCounter = 6 -> BarberShop.customerInQueue = FALSE)

-- После стрижки парикмахер всегда принимает оплату

CTLSPEC AG ((Barber0.status = Work & AX (Barber0.status = ReadyToPayment)) -> (A[Barber0.status = ReadyToPayment U Barber0.status = Payment] & AF(Barber0.status = Payment))) &

(Barber1.status = Work & AX (Barber1.status = ReadyToPayment)) -> (A[Barber1.status = ReadyToPayment U Barber1.status = Payment] & AF(Barber1.status = Payment)) &

(Barber2.status = Work & AX (Barber2.status = ReadyToPayment)) -> (A[Barber2.status = ReadyToPayment U Barber2.status = Payment] & AF(Barber2.status = Payment))

-- Если клиента нет парикмахер спит

CTLSPEC AG ((Barber0.status != Work & BarberShop.spotsCounter = 0) -> AX (Barber0.status != Work)) &

(Barber1.status != Work & BarberShop.spotsCounter = 0) -> AX (Barber1.status != Work) &

(Barber2.status != Work & BarberShop.spotsCounter = 0) -> AX (Barber2.status != Work)

-- Одновременно может производить оплату только один клиент

CTLSPEC AG (Barber0.status = Payment -> Barber1.status != Payment & Barber2.status != Payment) & (Barber1.status = Payment -> Barber0.status != Payment & Barber2.status != Payment)

& (Barber2.status = Payment -> Barber0.status != Payment & Barber1.status != Payment)