《通信协议软件设计》课程作业二

**姓名:Jayce 学号:201866666**

1. **安装模型检测软件spin，分析该软件自带的练习例子ex\_6.pml执行过程**



并说明：

1. source通道的作用；
2. 将source通道中的存储单元数定义为0，起到什么作用？
3. 在此协议的Promela描述中，是怎样使用通道数组q[2]的两个通道q[0]、q[1]的？
4. 这个Promela描述中，是怎样描述响应消息超时的情况的？
5. 在这个Promela模型中，为什么定义了变量received\_r、received\_b？这两个变量将起什么作用？
6. 对ex\_6.pml的执行过程做总体性的描述。

**解答：**

1. source通道类似于网络层向数据链路层传输数据包的通道，在本程序中Source()进程为网络层，p5()进程为数据链路层，Source()进程将mtype类型数据放到source通道中，p5()进程从source通道接收数据,并放到ball变量中。
2. 存储单元数定义为0说明source通道不能存储报文，每当source()进程发送数据时,数据会被立即放的通道中保证通道不空,而不是缓存起来，当p5()进程从source通道接收数据时，有数据，那么会从中立即接收，提高接收操作的可执行性。
3. 程序中新建一个输入通道和输出通道来使用通道数组q[2]，对应代码：

chan inp out; inp=q[\_pid]; out[1-\_pid]，

此程序在运行时会产生两个p5()进程，pid分别为0和1，pid0进程使用q[0]作为输入通道inp，使用q[1]作为输出通道out，pid1则相反，这样一来，两个进程就可以通过这两个通道收发数据，来实现通信了。pid0向q[0]发送数据，pid1从q[1]接收数据，pid1从q[1]发送数据，pid0从q[1]接收数据。

1. 当出现p5()进程既不能发送不了数据，也接受不了数据后，则判定为帧超时，进入timeout状态，进程p5()输出超时信息，对应代码：

printf("MSC: timeout\n")

之后设变量i=1用于计数，进入循环根据超时的帧号重新发送数据帧，每重新发送一个数据帧，i++，当不满足i <= nbuf时，跳出循环，重新发送数据帧结束。

对应代码：

:: timeout -> // 帧号超时

NextFrame = AckExp

printf("MSC: timeout\n")

i = 1

do // 回退N 重新发送帧号

:: i <= nbuf ->

if

:: \_pid%2 -> ball = frames[NextFrame]

:: else

fi

out!ball, NextFrame, (FrameExp + MaxSeq) % (MaxSeq + 1)

inc(NextFrame)

i++

:: else ->

break

od

1. 变量received\_r、received\_b 是数据标志位，用于标志red和blue是否接受到了，与sent\_r,和sent\_b相对应，当p5()进程从数据帧中接受收到red和blue时，就将这两个标志位设为true。
2. 总体性描述：该程序类似于回退N协议，首先启动进程source()和p5()。

* source()向source通道发送数据颜色数据source!white、source!blue、source!red，当red和blue发送出去之后，跳出两个循环，进程source()结束。
* p5()从source通道接收数据，并将其存入缓冲区，数据被封装成帧，打印nbuf的值，将数据帧发送到out通道中。
* p5()从inp通道接收数据帧，如果数据帧的帧号等于期待接收的帧号，则将其帧中的数据打印输出，之后判断数据是否为red或blue， 如果满足条件，则将对应的标志位received\_r\ received\_b设为true；否则抛弃数据帧。最后循环判断接收的确认帧号是否满足条件，如果条件满足，则nbuf--，如果不满足条件，则输出对应的信息，跳出循环。
* p5()进程如果发送超时，则循环重新发送之前未接收到确认的帧。

1. **分析并说明下面的Promela程序的执行过程。**

#define true 1

#define false 0

#define Aturn false

#define Bturn true

bool x, y, t;

proctype A()

{ x = true;

t = Bturn;

(y == false || t == Aturn);

/\* 临界区\*/

x = false

}

proctype B()

{ y = true;

t = Aturn;

(x == false || t == Bturn);

/\* 临界区\*/

y = false

}

init

{ run A(); run B()

}

该程序是一个控制多进程访问临界区的程序PV操作，bool x y t类似于三把互斥锁，来保证在同一时刻只有一个进程能访问临界区。在init进程执行后，它会创建两个子进程A和B。

* 假设A进程先执行B后执行，在进程A中x=true , t=Bturn, 由于y的默认值为false 0, 那么进程A进入临界区修改数据，如果进程B开始执行想要进入临界区时，由于目前x=true t=Aturn 所以进程B不能进入临界区修改数据，然后开始进入忙等待状态。当进程A访问临界区结束后，执行x=false，进程B可以进入临界区访问数据，两个进程有序访问临界区。相反进程B先执行A后执行也是如此。
* 假设A、B进程同时执行，x=true，y=true，两个进程中y==false，x==false都不满足，所以只能看由于t的取值，由于t在同一时刻只能等于Aturn或者Bturn，所以进程A和B只有一个能进入临界区。假设当前t=Aturn，进程A开始访问临界区，进程B忙等待，当进程A访问临界区结束后，x=false，进程B停止忙等待，开始访问临界区。

1. **Promlea语言中的语句的可执行性（executability）是什么含义？**

在PROMELA中，条件和语旬是一样的，即使是一个独立的布尔表达式也可以作为一条语旬。每个语句是否能执行是有条件的，称为语句的 “**可执行性**“。

所有PROMELA语旬要么是可执行的， 要么处千阻塞状态，这取决于变量的当前值或者报文通道中的当前内容。 所以在PROMELA中，语句的可执行性蕴含了同步的含义： 进程等待一个事件的发生可以通过等待一条陈述语句变为可执行来实现。例如：对于下面的遇忙等待循环：

**while (a!= b) skip**

当a 与b相等时，条件被执行，进程继续往下执行。否则该条件被阻寒， 直到a等于b成立。

此外在在PROMELA中，所有的变量声明和赋值语句都是可执行的。

1. **Promela语言中的End标签、Progress标签、Accept标签有什么作用？**

**【End标签】**

在Promela程序中，一个可终止执行序列的期望的结束状态至少满足以下两个条件：①所有启动的进程实例均终止（到达进程体的最后）；②所有报文通道均为空。

需要注意的是，并不是所有进程一定要终止，其中有的进程可能会停留在空闲状态， 也可能会在某些状态间循环， 等待某个消息的到来后再进行其他操作。 例如， 当客户进程终止时，服务器进程仍然在等待新的客户进程的连接请求。 因此，在建模过程中必须能够标志哪些是期望的结束状态。 在PROMELA中， 用"end"标记来实现。 例如：

protype dijkstra()

{

end:do

:: sema ! p -> sema ? v

od

}

该例中，end标签告诉验证器：所有 dijkstra进程实例处Tend标签所示的状态是一个期望的结束状态。

**【Progress标签】**

系统不能无限循环经过一个没有标记为进展状态(progress state)的状态，状态标记用"progress"标签来实现。也就是说，当一个循环被标记为progress后，验证器就会认为该循环是一个“好”循环，可以被无限经过。例如：

protype dijkstra()

{

end: do

:: sema ! p ->

progress: sema ? v

od

}

在该例中，将测试信号量语句标记为进展状态，表明信号量守护语旬不能无限地被阻塞，例如，被一个从不经过进展状态的无限循环所阻塞。

**【Accept标签】**

系统不能无限经过一个标记为不能无限发生的状态，状态标记用"accept"标签来

实现。也就说，当一个循环被标记为accept后，该循环不能无限的执行，如果出现无限执行的情况，那么验证器就会将其识别为一个“坏”循环。例如：

protype dijkstra()

{

end: do

:: sema ! p ->

accept: sema ? v

od

}

该例中的 accept 告诉验证器不能无限地进行PV 操作。当然，这一要求是错误的，可以通过手工证明它是错误的，也可以通过一个自动验证器找到反例。

1. **请说明模型检测技术的基本原理？并说明模型检测技术有怎样的优势和不足？**

**【基本原理】**

模型检测技术通常采用Dolev-Yao模型、模态逻辑、有限状态机和进程代数等理论作为协议分析的理论基础，其基本原理是用状态迁移系统S表示系统的行为，用模态/时序逻辑公式F描述系统的性质。这样 ”系统是否具有所期望的性质” 就转化为数学问题 “状态迁移系统S是否是公式F的一个模型? ", 用公式表示为 Sj=F。对有限状态系统，这个问题是可判定的，即可以用计算机程序在有限时间内自动确定。

**【优势】**

模型检测技术是二十世纪最为成功的形式化验证技术其优点有如下四条：

1. 该技术可以精确地描述系统的要求；
2. 可以对系统进行形式化验证；
3. 通过该技术可以获得一个无二义性的协议规范；
4. 可以通过计算机辅助工具来帮助实施自动或半自动验证。

**【不足】**

模型检测技术存在的主要问题是：

1. 协议运行的无限状态空间不搜索性。
2. 协议运行过程中的无穷消息空间的不可搜索性。

上述问题导致了模型检测技术中搜索状态空间爆炸问题，即当协议并发进程分量较多时，系统的全局状态空间会随着并发量的增加，呈指数增长。对于这一缺点，可以通过缩减状态空间的方法来减少其计算量，如符号模型检测、偏序规约、程序切片、对称模型检测等技术。

1. **请说明LTL中的下列操作符（operator）：◇（eventually）、□（always）、U（Until）的含义。**

**◇（eventually）：**在以后某个时刻会真

例如：◇p表示p在以后某个时刻会真



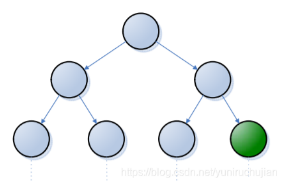
**□（always）：**现在和将来总是为真

例如：□p表示从当前时刻起，p总是为真



**U（Until）：**路径上存在某个状态满足性质q，而且它之前的每个状态都满足性质p

例如：pUq表示 公式p一直为真，直到公式q为真



**思考题（选做题）：**

1. **Promela语言这种建模语言中的通道为什么能够用于对系统中的进程间的通信建模？**

因为通道可以实现跨进程的数据传输；

例如：当进程A想要发送数据给进程B时，可以定义一个通道chan q1=[16]of{short}，进程A执行 q1！data1 将数据放入通道q1中，之后进程B执行q1？data2 从通道q1中取出数据，这样就实现了进程A与进程B之间的数据通信。所以进程间可以使用通道进行通信建模。

1. **Promela语言中的进程按照交替（interleaving）方式执行，对系统建模来说，其合理性在哪里？**

保证多进程运行的有序性，进程不会出现饿死症状，方便观察各个进程的运行状态。