Ptolemy 库	名称	作用	BMC 编码
Sources Library		生成一个稳定递增或递减的数值序列.	设左侧 init 和 step 输入端口所连接 relation 分别为 rIIn 和 rSIn，右侧输出端口所连接 relation 为 rOut，状态为 ST_Ramp，内部参数分别为 init，step，limit 则 情形一：当 limit<=0 时 ①没有输入连接时 $(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)$ $\vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)$ ②init 连接，step 未连接时 $(rIIn[i]!=nil \wedge rOut[i]=rIIn[i])$ $\vee\left(rIIn[i]=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)$ ③init 未连接，step 连接时 $\left(rSIn[i]!=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+rSIn[i-1] \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)$ $\vee\left(rSIn[i]=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)$ ④init 连接，step 连接时 $(rIIn[i]!=nil \wedge rOut[i]=rIIn[i])$ $\vee\left(\begin{array}{l}rIIn[i]=nil \wedge \\ \left(\begin{array}{l}\left(rSIn[i]!=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+rSIn[i-1] \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\\ \vee\left(rSIn[i]=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp_{i-1}>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\end{array}\right)\end{array}\right)$ 情形二：当 limit>0 时 ①没有输入连接时 $(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)$ $\vee(limit>ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)$ $\vee(ST\_Ramp[i-1]>=limit \wedge rOut[i]=nil \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)$ ②init 连接，step 未连接时 $\left(\begin{array}{l}(ST\_Ramp[i-1]<limit) \wedge \\ \left(\begin{array}{l}(rIIn[i]!=nil \wedge rOut[i]=rIIn[i] \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\\ \vee\left(rIIn[i]=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\end{array}\right)\end{array}\right)$ $\vee(ST\_Ramp[i-1]>=limit \wedge rOut[i]=nil \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)$ ③init 未连接，step 连接时 $\left(\begin{array}{l}(ST\_Ramp[i-1]<limit) \wedge \\ \left(\begin{array}{l}\left(rSIn[i]!=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+rSIn[i-1] \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\\ \vee\left(rSIn[i]=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\end{array}\right)\end{array}\right)$ $\vee(ST\_Ramp[i-1]>=limit \wedge rOut[i]=nil \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)$ ④init 连接，step 连接时. $\left(\begin{array}{l}(ST\_Ramp[i-1]<limit) \wedge \\ \left(\begin{array}{l}(rIIn[i]!=nil \wedge rOut[i]=rIIn[i] \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\\ \vee\left(\begin{array}{l}rIIn[i]=nil \wedge \\ \left(\begin{array}{l}\left(rSIn[i]!=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+rSIn[i-1] \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\\ \vee\left(rSIn[i]=nil \wedge \left(\begin{array}{l}(ST\_Ramp[i-1]=0 \wedge rOut[i]=init \wedge ST\_Ramp[i]=1)\\ \vee(ST\_Ramp[i-1]>0 \wedge rOut[i]=rOut[i-1]+step \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)\end{array}\right)\right)\end{array}\right)\end{array}\right)\end{array}\right)$ $\vee(ST\_Ramp[i-1]>=limit \wedge rOut[i]=nil \wedge ST\_Ramp[i]=ST\_Ramp[i-1]+1)$ 注：参数 init 和 step 默认情况下分别是 0 和 1，如果更改，则可从 Ramp

			actor 的 property 中读取得到.
		生成一个由参数所设定的值或者表达式（这里是 1）.	设右侧输出端口所连接 relation 为 rOut，内部参数分别为 parameter，limit 则 情形一：limit<=0 时 $$rOut[i] = parameter$$ 情形二：limit>0 时 $$c[i] = c[i-1]+1 \wedge \begin{pmatrix} (c[i] <= limit \wedge rOut[i] = parameter) \\ \vee (c[i] > limit \wedge rOut[i] = nil) \end{pmatrix}$$ $$c[0] = 0$$ 注：参数的值默认情况下是 1, 如果更改，可从模型的 MoML xml 文件中 Const actor 的 property 中读取得到.
		生成一个由指定值构成的序列（这里是生成 1 构成的序列）.	情形一：设左侧端口所连接 relation 为 rIn, 右侧输出端口所连接 relation 为 rOut，数组 s 存储 Sequence 中的数据则 ①repeat 为 true（repeat 优先级最高）时 $$((rIn[i] = false \vee rIn[i] = nil) \wedge rOut[i] = nil)$$ $$\vee \left(rIn[i] = true \wedge \begin{pmatrix} (index[i] >= 0 \wedge index[i] < SeqNum) \wedge ((rOut[i] = s_{index} \wedge index[i+1] = index[i]+1)) \\ \vee ((index[i] >= SeqNum) \wedge (rOut[i] = s_1) \wedge (index[i+1] = 2)) \end{pmatrix} \right)$$ ②holdLastPut 为 false，repeat 为 false 时 $$((rIn[i] = false \vee rIn[i] = nil) \wedge rOut[i] = nil)$$ $$\vee \left(rIn[i] = true \wedge \begin{pmatrix} (index[i] >= 0 \wedge index[i] < SeqNum) \wedge ((rOut[i] = s_{index} \wedge index[i+1] = index[i]+1)) \\ \vee ((index[i] >= SeqNum) \wedge (rOut[i] = nil) \wedge (index[i+1] = index[i])) \end{pmatrix} \right)$$ ③holdLastPut 为 true，repeat 为 false 时 $$((rIn[i] = false \vee rIn[i] = nil) \wedge rOut[i] = nil)$$ $$\vee \left(rIn[i] = true \wedge \begin{pmatrix} (index[i] >= 0 \wedge index[i] < SeqNum) \wedge ((rOut[i] = s_{index} \wedge index[i+1] = index[i]+1)) \\ \vee ((index[i] >= SeqNum) \wedge (rOut[i] = s_{SeqNum}) \wedge (index[i+1] = index[i])) \end{pmatrix} \right)$$ 注：参数 index 表示序列的索引，参数 vindex 对应于 Sequence 所给出的序列中索引为 index 的元素 s1, s2,sv3…，且这些值可从模型的 MoML xml 文件中 Sequence actor 的 property 中读取得到 情形二：设左侧端口没有连接 relation, 右侧输出端口所连接 relation 为 rOut，则 ①repeat 为 true（repeat 优先级最高）时 $$((index[i] >= 0 \wedge index[i] < SeqNum) \wedge rOut[i] = s_{index} \wedge index[i+1] = index[i]+1)$$ $$\vee ((index[i] >= SeqNum) \wedge (rOut[i] = s_1) \wedge (index[i+1] = 2))$$ ②holdLastPut 为 false，repeat 为 false 时 $$((index[i] >= 0 \wedge index[i] < SeqNum) \wedge rOut[i] = s_{index} \wedge index[i+1] = index[i]+1)$$ $$\vee ((index[i] >= SeqNum) \wedge (rOut[i] = nil) \wedge (index[i+1] = index[i]))$$ ③holdLastPut 为 true，repeat 为 false 时 $$((index[i] >= 0 \wedge index[i] < SeqNum) \wedge rOut[i] = s_{index} \wedge index[i+1] = index[i]+1)$$ $$\vee ((index[i] >= SeqNum) \wedge (rOut[i] = s_{SeqNum}) \wedge (index[i+1] = index[i]))$$ 注：参数 index 表示序列的索引，参数 vindex 对应于 Sequence 所给出的序列中索引为 index 的元素, 且该值可从模型的 MoML xml 文件中 Sequence actor 的 property 中读取得到.
		生成一个常字符串序列.	该 actor 属于 Const 的子类 actor，因此编码方式和 Const 相同

Math Library		加上+输入端口处的输入数据，并减去-输入端口处的输入数据	情形一：假设+端口没有 relation 连接，而-端口所连接的 relation 为 rMIn，而右侧 out 端口的 relation 为 rOut，创建数组 sM 和 nilNumberM 来辅助编码则： $\bigwedge_{j=1}^{inMinusPort.width}\left(\left(rMIn_j[i]!=nil\wedge sM_j[i]=rMIn_j\wedge nilNumberM_j[i]=0\right)\vee\left(rMIn_j[i]=nil\wedge sM_j[i]=0\wedge nilNumberM_j[i]=1\right)\right)$ $\wedge\left\{\begin{array}{l}\left(rOut[i]=-\sum_{j=1}^{inMinusPort.width}sM_j[i]\wedge\sum_{j=1}^{inMinusPort.width}nilNumberM_j[i]!=inMinusPort.width\right)\\ \vee\left(rOut[i]=nil\wedge\sum_{j=1}^{inMinusPort.width}nilNumberM_j[i]!=inMinusPort.width\right)\end{array}\right.$ 情形二：假设-端口没有 relation 连接，而+端口所连接的 relation 为 rAIn，而右侧 out 端口的 relation 为 rOut，创建数组 sA 和 nilNumberA 来辅助编码则： $\bigwedge_{j=1}^{inAddPort.width}\left(\left(rAIn_j[i]!=nil\wedge sA_j[i]=rAIn_j\wedge nilNumberA_j[i]=0\right)\vee\left(rAIn_j[i]=nil\wedge sA_j[i]=0\wedge nilNumberA_j[i]=1\right)\right)$ $\wedge\left\{\begin{array}{l}\left(rOut[i]=\sum_{j=1}^{inAddPort.width}sA_j[i]\wedge\sum_{j=1}^{inAddPort.width}nilNumberA_j[i]!=inAddPort.width\right)\\ \vee\left(rOut[i]=nil\wedge\sum_{j=1}^{inAddPort.width}nilNumberA_j[i]=inAddPort.width\right)\end{array}\right.$ 情形三：假设+端口所连接的 relation 为 rAIn，而-端口所连接的 relation 为 rMIn，而右侧 out 端口的 relation 为 rOut，创建数组 sA，sM 和 nilNumberA，nilNumberM 来辅助编码则： $\bigwedge_{j=1}^{inAddPort.width}\left(\left(rAIn_j[i]!=nil\wedge sA_j[i]=rAIn_j\wedge nilNumberA_j[i]=0\right)\vee\left(rAIn_j[i]=nil\wedge sA_j[i]=0\wedge nilNumberA_j[i]=1\right)\right)$ $\wedge\bigwedge_{j=1}^{inMinusPort.width}\left(\left(rMIn_j[i]!=nil\wedge sM_j[i]=rMIn_j\wedge nilNumberM_j[i]=0\right)\vee\left(rMIn_j[i]=nil\wedge sM_j[i]=0\wedge nilNumberM_j[i]=1\right)\right)$ $\wedge\left\{\begin{array}{l}\left(rOut[i]=\sum_{j=1}^{inAddPort.width}sA_j[i]-\sum_{j=1}^{inMinusPort.width}sM_j[i]\wedge\left(\begin{array}{l}\sum_{j=1}^{inAddPort.width}nilNumberA_j[i]!=inAddPort.width\\ \vee\sum_{j=1}^{inMinusPort.width}nilNumberM_j[i]!=inMinusPort.width\end{array}\right)\right)\\ \vee\left(rOut[i]=nil\wedge\sum_{j=1}^{inAddPort.width}nilNumberA_j[i]=inAddPort.width\wedge\sum_{j=1}^{inMinusPort.width}nilNumberM_j[i]=inMinusPort.width\right)\end{array}\right.$ 注：inPort.width 可通过看 BooleanSwitch actor 的 input 连接几条 relation 获知.
		给输入乘以一个由参数所设定的常数 c（这里乘以-1）	设左、右两侧端口所对应的 relation 分别是 rIn 和 rOut, Scale 中的常数为 c，则 $\left(rOut[i]=rIn[i]*c\wedge rIn[i]!=nil\right)$ $\vee\left(rIn[i]=nil\wedge rOut[i]=nil\right)$ 注：参数 c 默认情况下是 1，如果更改可从模型的 MoML xml 文件中 Scale actor 的 Property 中读取得到
		执行三角函数和反三角函数，包括 sin, cos, tan, arcsin, arccos, arctan.	设左、右侧端口所对应的 relation 分别是 rIn 和 rOut, 则 $rOut[i]=\sin\left(rIn[i]\right);\ rOut[i]=\cos\left(rIn[i]\right);\ rOut[i]=\tan\left(rIn[i]\right);$ $rOut[i]=\arcsin\left(rIn[i]\right);rOut[i]=\arccos\left(rIn[i]\right);rOut[i]=\arctan\left(rIn[i]\right).$
		执行多种只有一个参数的数学函数，包括 exp, log, sign, square, sqrt	设左、右两侧端口所对应的 relation 分别是 rIn 和 rOut, 则 $rOut[i]=\exp\left(rIn[i]\right);rOut[i]=\log\left(rIn[i]\right);rOut[i]=sign\left(rIn[i]\right);$ $rOut[i]=square\left(rIn[i]\right);rOut[i]=sqrt\left(rIn[i]\right).$
		计算输入值的绝对值.	设左、右两侧端口所对应的 relation 分别是 rIn 和 rOut, 则 $\left(rOut[i]=abs\left(rIn[i]\right)\wedge rIn[i]!=nil\right)\vee\left(rIn[i]=nil\wedge rOut[i]=nil\right)$
		乘以×输入端口处的输	情形一：假设*端口没有 relation 连接，而÷端口所连接的 relation 为 rDIn，而右侧 out 端口的 relation 为 rOut，创建数组 sD 和 nilNumberD 来辅助编

		入数据，并除以÷输入端口处的输入数据.	码，则： $\bigwedge_{j=1}^{inDividePort.width}\left(\left(rDIn_j[i]!=nil\wedge sD_j[i]=rDIn_j\wedge nilNumberD_j[i]=0\right)\vee\left(rDIn_j[i]=nil\wedge sD_j[i]=0\wedge nilNumberD_j[i]=1\right)\right)$ $\wedge\left(\begin{array}{l}\left(rOut[i]=\prod_{j=1}^{inDividePort.width}\left(1/sD_j[i]\right)\wedge\sum_{j=1}^{inDividePort.width}nilNumberD_j[i]!=inDividePort.width\right)\\ \vee\left(rOut[i]=nil\wedge\sum_{j=1}^{inDividePort.width}nilNumberD_j[i]=inDividePort.width\right)\end{array}\right)$ 情形二：假设÷端口没有 relation 连接，而*端口所连接的 relation 为 rMIn，而右侧 out 端口的 relation 为 rOut，创建数组 sM 和 nilNumberM 来辅助编码，则： $\bigwedge_{j=1}^{inMultPort.width}\left(\left(rMIn_j[i]!=nil\wedge sM_j[i]=rMIn_j\wedge nilNumberM_j[i]=0\right)\vee\left(rMIn_j[i]=nil\wedge sM_j[i]=0\wedge nilNumberM_j[i]=1\right)\right)$ $\wedge\left(\begin{array}{l}\left(rOut[i]=\prod_{j=1}^{inMultPort.width}sM_j[i]\wedge\sum_{j=1}^{inMultPort.width}nilNumberM_j[i]!=inMultPort.width\right)\\ \vee\left(rOut[i]=nil\wedge\sum_{j=1}^{inMultPort.width}nilNumberM_j[i]=inMultPort.width\right)\end{array}\right)$ 情形三：假设*端口所连接的 relation 为 rMIn，而÷端口所连接的 relation 为 rDIn，而右侧 out 端口的 relation 为 rOut，创建数组 sM，sD 和 nilNumberM，nilNumberD 来辅助编码，则： $\bigwedge_{j=1}^{inMultPort.width}\left(\left(rMIn_j[i]!=nil\wedge sM_j[i]=rMIn_j\wedge nilNumberM_j[i]=0\right)\vee\left(rMIn_j[i]=nil\wedge sM_j[i]=0\wedge nilNumberM_j[i]=1\right)\right)$ $\wedge\bigwedge_{j=1}^{inDividePort.width}\left(\left(rDIn_j[i]!=nil\wedge sD_j[i]=rDIn_j\wedge nilNumberD_j[i]=0\right)\vee\left(rDIn_j[i]=nil\wedge sD_j[i]=0\wedge nilNumberD_j[i]=1\right)\right)$ $\wedge\left(\begin{array}{l}\left(rOut[i]=\prod_{j=1}^{inMultPort.width}sM_j[i]*\prod_{j=1}^{inDividePort.width}\left(1/sD_j[i]\right)\wedge\left(\begin{array}{l}\sum_{j=1}^{inMultPort.width}nilNumberM_j[i]!=inMultPort.width\\ \vee\sum_{j=1}^{inDividePort.width}nilNumberD_j[i]!=inDividePort.width\end{array}\right)\right)\\ \vee\left(rOut[i]=nil\wedge\sum_{j=1}^{inDividePort.width}nilNumberD_j[i]=inDividePort.width\wedge\sum_{j=1}^{inMultPort.width}nilNumberM_j[i]=inMultPort.width\right)\end{array}\right)$ 注：inPort.width 可通过看 MultiplyDivide actor 的 input 连接几条 relation 获知.
		输出从上次 reset 端口处接收到 true 值时到当前 tick，左侧端口所接收到的所有输入的和.	情形一：假设 reset 端口处没有连接，而左侧端口处所连接的 relation 为 rIn，右侧输出端口处所连接的 relation 为 rOut，创建数组 s，n，变量 sum 来辅助编码，则 $\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]!=nil\wedge s_j[i]=rIn_j[i]\wedge n_j[i]=1\right)\vee\left(rIn_j[i]=nil\wedge s_j[i]=0\wedge n_j[i]=0\right)\right)$ $\wedge\left(\begin{array}{l}\left(\sum_{j=1}^{inPort.width}n_j[i]!=0\wedge rOut=sum[i]\wedge sum[i]=sum[i-1]+\sum_{j=1}^{inPort.width}s_j[i]\right)\\ \vee\left(\sum_{j=1}^{inPort.width}n_j[i]=0\wedge rOut=sum[i]\wedge sum[i]=sum[i-1]\right)\end{array}\right)$ 注：$sum[0]=init$原模型有初始值 情形二：假设 reset 端口处有连接 rRIn, 而左侧端口处所连接的 relation 为 rIn, 右侧输出端口处所连接的 relation 为 rOut, 创建数组 s，n，变量 sum 来辅助编码，则 $\left(\begin{array}{l}rRIn[i]=true\wedge\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]!=nil\wedge s_j[i]=rIn_j[i]\wedge n_j[i]=1\right)\vee\left(rIn_j[i]=nil\wedge s_j[i]=0\wedge n_j[i]=0\right)\right)\\ \wedge\left(\begin{array}{l}\left(\sum_{j=1}^{inPort.width}n_j[i]!=0\wedge rOut=sum[i]\wedge sum[i]=init+\sum_{j=1}^{inPort.width}s_j[i]\right)\\ \vee\left(\sum_{j=1}^{inPort.width}n_j[i]=0\wedge rOut=sum[i]\wedge sum[i]=init\right)\end{array}\right)\end{array}\right)$ $\vee\left(\begin{array}{l}\left(rRIn[i]=false\vee rRIn[i]=nil\right)\wedge\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]!=nil\wedge s_j[i]=rIn_j[i]\wedge n_j[i]=1\right)\vee\left(rIn_j[i]=nil\wedge s_j[i]=0\wedge n_j[i]=0\right)\right)\\ \wedge\left(\begin{array}{l}\left(\sum_{j=1}^{inPort.width}n_j[i]!=0\wedge rOut=sum[i]\wedge sum[i]=sum[i-1]+\sum_{j=1}^{inPort.width}s_j[i]\right)\\ \vee\left(\sum_{j=1}^{inPort.width}n_j[i]=0\wedge rOut=sum[i]\wedge sum[i]=sum[i-1]\right)\end{array}\right)\end{array}\right)$ 注：$sum[0]=init$ 注：inPort.width 可通过看 Accumulator actor 的 input 连接几条 relation 获知.

		输出从上次reset 端口处接收到 true 值时到当前 tick，左端端口所接收到的所有输入的平均值.	情形一：假设 reset 端口处没有连接，而左侧端口处所连接的 relation 为 rIn, 右侧端口处所连接的 relation 为 rOut, 创建变量 sum，count 来辅助编码，则 ①输入为Int型时： $\left(\begin{array}{l} rIn[i]!=nil \wedge \left(\begin{array}{l}(rOut[i]=sum[i]/count[i] \wedge sum[i]>=0) \\ \vee(rOut[i]=-to\_\text{int}((-sum[i])/count[i]) \wedge sum[i]<0)\end{array}\right) \\ \wedge sum[i]=sum[i-1]+rIn[i] \wedge count[i+1]=count[i]+1\end{array}\right)$ $\vee(rIn[i]=nil \wedge rOut[i]=nil \wedge sum[i]=sum_{i-1} \wedge count_{i+1}=count[i])$ ②输入非Int型： $(rIn[i]!=nil \wedge rOut[i]=sum[i]/count[i] \wedge sum[i]=sum[i-1]+rIn[i] \wedge count[i+1]=count[i]+1)$ $\vee(rIn[i]=nil \wedge rOut[i]=nil \wedge sum[i]=sum[i-1] \wedge count[i+1]=count[i])$ 注：$sum[0]=0, count[1]=1$ 情形二：假设 reset 端口处有连接 rRIn,左侧端口处所连接的 relation 为 rIn, 右侧端口处所连接的 relation 为 rOut, 创建变量 sum，count 来辅助编码，则 ①输入为Int型时： $\left(\begin{array}{l}(rIn[i]!=nil) \wedge \left(\begin{array}{l}(rRIn[i]=true \wedge rOut[i]=rIn[i] \wedge sum[i]=rIn[i] \wedge count[i]=1) \\ \vee\left(\begin{array}{l}(rRIn[i]=false \vee rRIn[i]=\text{nil}) \\ \wedge\left(\begin{array}{l}(rOut[i]=sum[i]/count[i] \wedge sum[i]>=0) \\ \vee(rOut[i]=-to\_\text{int}((-sum[i])/count[i]) \wedge sum[i]<0)\end{array}\right) \\ \wedge sum[i]=sum[i-1]+rIn[i] \wedge count[i+1]=count[i]+1)\end{array}\right)\end{array}\right)\end{array}\right)$ $\vee(rIn[i]=nil \wedge rOut[i]=nil \wedge sum[i]=sum[i-1] \wedge count[i+1]=count[i])$ ②输入非Int型： $\left(\begin{array}{l}(rIn[i]!=nil) \wedge \left(\begin{array}{l}(rRIn[i]=true \wedge rOut[i]=rIn[i] \wedge sum[i]=rIn[i] \wedge count[i]=1) \\ \vee\left(\begin{array}{l}(rRIn[i]=false \vee rRIn[i]=\text{nil}) \wedge rOut[i]=sum[i]/count[i] \\ \wedge sum[i]=sum[i-1]+rIn[i] \wedge count[i+1]=count[i]+1\end{array}\right)\end{array}\right)\end{array}\right)$ $\vee(rIn[i]=nil \wedge rOut[i]=nil \wedge sum[i]=sum[i-1] \wedge count[i+1]=count[i])$ 注：$sum[0]=0, count[1]=1$
		将输入值限制在由 top 和bottom 所指定的范围内. 如果输入值介于 top 和 bottom 之间，则输出该输入值；如果输入值比 top 大，则输出 top；如果输入值比 bottom 小，则输出 bottom.	设左、右两侧连接的 relation 分别为 rIn 和 rOut，Limiter 中内部参数 top，bottom，则 $\left(\begin{array}{l}rIn[i]!=nil \wedge \left(\begin{array}{l}(rIn[i]>=bottom \wedge rIn[i]<=top \wedge rOut[i]=rIn[i]) \\ \vee(rIn[i]>top \wedge rOut[i]=top) \vee (rIn[i]<bottom \wedge rOut[i]=bottom)\end{array}\right)\end{array}\right)$ $\vee(rIn[i]=nil \wedge rOut[i]=nil)$
		输出所给出的表达式所计算得到的值(这里给出的表达式是 x-2*y).	设左、右两侧连接的 relation 分别为 rxIn, ryIn 和 rOut，则 $((rOut[i]=rxIn[i]-2*ryIn[i]) \wedge (rxIn[i]!=nil \wedge ryIn[i]!=nil))$ $\vee((rOut[i]=nil) \wedge (rxIn[i]=nil \vee ryIn[i]=nil))$ （注：表达式可以从模型的 xml 文件中读取得到，具体是：Expression actor---StringAttribute 类型---名为“expression”的 property 对象的 value.）
		输出当前 tick 的输入	

		token 中的最大值.	情形一：设左侧连接的 relation 为 rIn, 而右侧连接的 relation 只有 rMax, 且在 channelNumber 端口处没有 relation 连接，创建数组 s 辅助编码，则 $\left(\bigwedge_{j=1}^{inPort.width} rIn_j[i]=nil \wedge rMax[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width} rIn_j[i]!=nil\right)\\ \wedge\bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}\left(\begin{array}{l}rIn_j[i]!=nil\wedge\left(\begin{array}{l}\left(not(s_{j-1}[i]=s_0[i])\wedge\left(\begin{array}{l}(s_{j-1}[i]>=rIn_j[i]\wedge s_j[i]=s_{j-1}[i])\\ \vee(s_{j-1}[i]<rIn_j[i]\wedge s_j[i]=rIn_j[i])\end{array}\right)\right)\\ \vee(s_{j-1}[i]=s_0[i]\wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(rIn_j[i]=nil\wedge s_j[i]=s_{j-1}[i])\end{array}\right)\\ \wedge rMax_i=s_{inPort.width}[i]\end{array}\right)$ 情形二：设左侧连接的 relation 为 rIn, 而右侧连接的 relation 只有 rMaxCN, 且在 maximumValue 端口处没有 relation 连接, 创建数组 s，c 辅助编码，则 $\left(\bigwedge_{j=1}^{inPort.width} rIn_j[i]=nil \wedge rMaxCN[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width} rIn_j[i]!=nil\right)\\ \wedge\bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}\left(\begin{array}{l}rIn_j[i]!=nil\wedge\left(\begin{array}{l}\left(not(s_{j-1}[i]=s_0[i])\wedge\left(\begin{array}{l}(s_{j-1}[i]>=rIn_j[i]\wedge s_j[i]=s_{j-1}[i]\wedge c_j[i]=c_{j-1}[i])\\ \vee(s_{j-1}[i]<rIn_j[i]\wedge s_j[i]=rIn_j[i]\wedge c_j[i]=j)\end{array}\right)\right)\\ \vee(s_{j-1}[i]=s_0[i]\wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(rIn_j[i]=nil\wedge s_j[i]=s_{j-1}[i]\wedge c_j[i]=c_{j-1}[i])\end{array}\right)\\ \wedge rMaxCN[i]=c_{inPort.width}[i]\end{array}\right)$ 情形三：设左侧连接的 relation 为 rIn, 而右侧连接的 relation 为 rMax, rMaxCN, 创建数组 s，c 辅助编码，则 $\left(\bigwedge_{j=1}^{inPort.width} rIn_j[i]=nil \wedge rMaxCN[i]=nil \wedge rMax[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width} rIn_j[i]!=nil\right)\\ \wedge\bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}\left(\begin{array}{l}rIn_j[i]!=nil\wedge\left(\begin{array}{l}\left(not(s_{j-1}[i]=s_0[i])\wedge\left(\begin{array}{l}(s_{j-1}[i]>=rIn_j[i]\wedge s_j[i]=s_{j-1}[i]\wedge c_j[i]=c_{j-1}[i])\\ \vee(s_{j-1}[i]<rIn_j[i]\wedge s_j[i]=rIn_j[i]\wedge c_j[i]=j)\end{array}\right)\right)\\ \vee(s_{j-1}[i]=s_0[i]\wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(rIn_j[i]=nil\wedge s_j[i]=s_{j-1}[i]\wedge c_j[i]=c_{j-1}[i])\end{array}\right)\\ \wedge rMaxCN[i]=c_{inPort.width}[i]\wedge rMax[i]=s_{inPort.width}[i]\end{array}\right)$ 注：inPort.width 可通过看 Maximum actor 的 input 连接几条 relation 获知.
		输出当前 tick 的输入 token 中的最小值.	情形一：设左侧连接的 relation 为 rIn 而右侧连接的 relation 只有 rMin, 且在 channelNumber 端口处没有 relation 连接, 创建数组 s 辅助编码， 则

			$\left(\bigwedge_{j=1}^{inPort.width} rIn_j[i]=nil \wedge rMin[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width} rIn_j[i]!=nil\right)\\ \wedge \bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}rIn_j[i]!=nil \wedge \left(\begin{array}{l}not(s_{j-1}[i]=s_0[i]) \wedge \left(\begin{array}{l}(s_{j-1}[i]<=rIn_j[i] \wedge s_j[i]=s_{j-1}[i])\\ \vee(s_{j-1}[i]>rIn_j[i] \wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(s_{j-1}[i]=s_0[i] \wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(rIn_j[i]=nil \wedge s_j[i]=s_{j-1}[i])\end{array}\right)\\ \wedge rMin[i]=s_{inPort.width}[i]\end{array}\right)$ 情形二：设左侧连接的 relation 为 rIn, 而右侧连接的 relation 只有 rMinCN, 且在 minimumValue 端口处没有 relation 连接, 创建数组 s，c 辅助编码，则 $\left(\bigwedge_{j=1}^{inPort.width} rIn_j[i]=nil \wedge rMinCN[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width} rIn_j[i]!=nil\right)\\ \wedge \bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}rIn_j[i]!=nil \wedge \left(\begin{array}{l}not(s_{j-1}[i]=s_0[i]) \wedge \left(\begin{array}{l}(s_{j-1}[i]<=rIn_j[i] \wedge s_j[i]=s_{j-1}[i] \wedge c_j[i]=c_{j-1}[i])\\ \vee(s_{j-1}[i]>rIn_j[i] \wedge s_j[i]=rIn_j[i] \wedge c_j[i]=j)\end{array}\right)\\ \vee(s_{j-1}[i]=s_0[i] \wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(rIn_j[i]=nil \wedge s_j[i]=s_{j-1}[i] \wedge c_j[i]=c_{j-1}[i])\end{array}\right)\\ \wedge rMinCN[i]=c_{inPort.width}[i]\end{array}\right)$ 情形三：设左侧连接的 relation 为 rIn, 而右侧连接的 relation 为 rMin, rMinCN, 创建数组 s，c 辅助编码，则 $\left(\bigwedge_{j=1}^{inPort.width} rIn_j[i]=nil \wedge rMinCN[i]=nil \wedge rMin[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width} rIn_j[i]!=nil\right)\\ \wedge \bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}rIn_j[i]!=nil \wedge \left(\begin{array}{l}not(s_{j-1}[i]=s_0[i]) \wedge \left(\begin{array}{l}(s_{j-1}[i]<=rIn_j[i] \wedge s_j[i]=s_{j-1}[i] \wedge c_j[i]=c_{j-1}[i])\\ \vee(s_{j-1}[i]>rIn_j[i] \wedge s_j[i]=rIn_j[i] \wedge c_j[i]=j)\end{array}\right)\\ \vee(s_{j-1}[i]=s_0[i] \wedge s_j[i]=rIn_j[i])\end{array}\right)\\ \vee(rIn_j[i]=nil \wedge s_j[i]=s_{j-1}[i] \wedge c_j[i]=c_{j-1}[i])\end{array}\right)\\ \wedge rMinCN[i]=c_{inPort.width}[i] \wedge rMin[i]=s_{inPort.width}[i]\end{array}\right)$ 注：inPort.width 可通过看 Minimum actor 的 input 连接几条 relation 获知.
		输出从模型开始运行到当前 tick 在输入端口处可看到的最大值.	设左、右两侧所连接的 relation 分别为 rIn 和 rOut，创建变量 index，max 辅助编码，则： $index[i+1]=index[i]+1 \wedge \left(\begin{array}{l}\left(\begin{array}{l}rIn[i]!=nil \wedge \left(\begin{array}{l}(index[i]=1 \wedge rOut[i]=\max[i+1] \wedge \max[i+1]=rIn[i])\\ \vee\left((index[i]>1) \wedge \left(\begin{array}{l}(rIn[i]>\max[i] \wedge rOut[i]=\max[i+1] \wedge \max[i+1]=rIn[i])\\ \vee(rIn[i]\le\max[i] \wedge rOut[i]=\max[i+1] \wedge \max[i+1]=\max[i])\end{array}\right)\right)\end{array}\right)\\ \vee(rIn[i]=nil \wedge rOut[i]=nil \wedge \max[i+1]=\max[i])\end{array}\right)$
		输出从模型开始运行到当前 tick 在输入端口处可看到的最小值.	设左、右两侧所连接的 relation 分别为 rIn 和 rOut，创建变量 index，max 辅助编码，则： $index_{i+1}=index_i+1 \wedge \left(\begin{array}{l}\left(\begin{array}{l}rIn[i]!=nil \wedge \left(\begin{array}{l}(index[i]=1 \wedge rOut[i]=\min[i+1] \wedge \min[i+1]=rIn_i)\\ \vee\left((index[i]>1) \wedge \left(\begin{array}{l}(rIn[i]<\min[i] \wedge rOut[i]=\min[i+1] \wedge \min[i+1]=rIn[i])\\ \vee(rIn[i]\ge\min[i] \wedge rOut[i]=\min[i+1] \wedge \min[i+1]=\min[i])\end{array}\right)\right)\end{array}\right)\\ \vee(rIn[i]=nil \wedge rOut[i]=nil \wedge \min[i+1]=\min[i])\end{array}\right)$
Flow Control		消费来自于左侧输入端口的 token 及底端控制端口的整数	设左侧端口所连接 relation 为 rIn, 底端端口所连接 relation 为 rCtrl, 右侧端口所连接 relation 为 rOut，创建变量 s 辅助编码，则：

		值 token，并将输入 token 的路线确定为控制 token 所指定的输出管道，而其它管道在此次激励中不生成 token.	$$\bigvee_{j=1}^{inPort.width}\left(\begin{array}{l}\left(rCtrl[i]=j\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\\ \wedge s[i]=j\wedge rOut_j[i]=rIn[i]\wedge\bigwedge_{h=1}^{j-1}rOut_h[i]=nil\wedge\bigwedge_{h=j+1}^{inPort.width}rOut_h[i]=nil\end{array}\right)$$ $$\vee\left(\begin{array}{l}\left(\left(rCtrl[i]=value\wedge s[i]=rCtrl[i]\right)\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\\ \wedge\left(s[i]<0\vee s[i]>=width\right)\wedge\bigwedge_{h=1}^{inPort.width}rOut_j[i]=nil\end{array}\right)$$ $$s[0]=0$$ 注：inPort.width 可通过看 Switch actor 的 input 连接几条 relation 获知.
		是 Switch 角色的一个变体，其中输出仅限为两个，且控制 token 是布尔值，因此作用为：消费来自于左侧输入端口的 token 及底端控制端口的布尔值 token，并将输入 token 的路线确定为控制 token 所指定的输出管道，而另一个输出管道在此次激励中不生成 token.	设左侧端口所连接 relation 为 rIn，底端端口所连接 relation 为 rCtrl，右侧 trueOutput 端口所连接 relation 为 rtOut，且右侧 falseOutput 端口所连接 relation 为 rfOut，创建变量 s 辅助编码，则：$$\left(\begin{array}{l}\left(rCtrl[i]=true\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\wedge s[i]=true\\ \wedge\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]!=nil\wedge rtOut_j[i]=rIn_j[i]\right)\vee\left(rIn_j[i]=nil\wedge rtOut_j[i]=nil\right)\right)\\ \wedge\bigwedge_{j=1}^{falsePort.width}rfOut_i[i]=nil\end{array}\right)$$ $$\vee\left(\begin{array}{l}\left(rCtrl[i]=false\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\wedge s[i]=false\\ \wedge\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]!=nil\wedge rfOut_j[i]=rIn_j[i]\right)\vee\left(rIn_j[i]=nil\wedge rfOut_j[i]=nil\right)\right)\\ \wedge\bigwedge_{j=1}^{truePort.width}rtOut_j[i]=nil\end{array}\right)$$ $$s[0]=false$$ *rtOut*和*rfOut*可以不连接 注：inPort.width 可通过看 BooleanSwitch actor 的 input 连接几条 relation 获知.
		消费由底端控制端口的整数值 token 所指定的管道中的单个 token，并将之发送给输出，其它输入端口处不消费 token.	设左侧端口所连接 relation 依次为 rIn1, rIn2, rIn3，底端端口所连接 relation 为 rCtrl，右侧端口所连接 relation 为 rOut，创建变量 s 辅助编码，则：$$\bigvee_{j=1}^{inPort.width}\left(\begin{array}{l}\left(rCtrl[i]=j\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\wedge s[i]=j\\ \wedge\left(\left(rIn_j[i]!=nil\wedge rOut[i]=rIn_j[i]\right)\vee\left(rIn_j[i]=nil\wedge rOut[i]=nil\right)\right)\end{array}\right)$$ $$\vee\left(\begin{array}{l}\left(\left(rCtrl[i]=value\wedge s[i]=rCtrl[i]\right)\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\\ \wedge\left(s[i]<1\vee s[i]>=inPort.width\right)\wedge rOut[i]=nil\end{array}\right)$$ $$s[0]=0$$ 注：inPort.width 可通过看 Select actor 的 input 连接几条 relation 获知.
		将底端 select 端口指定的输入管道中的 token，复制到输出端口处，而其它输入管道中的 token 全部舍弃.	设左侧端口所连接 relation 为 rIn，底端端口所连接 relation 为 rSec，右侧端口所连接 relation 为 rOut，创建变量 s 辅助编码，则：$$\bigvee_{j=1}^{inPort.width}\left(\begin{array}{l}\left(rSec[i]=j\vee\left(rSec[i]=nil\wedge s[i]=s[i-1]\right)\right)\wedge s[i]=0\\ \wedge\left(\left(rIn_j[i]!=nil\wedge rOut[i]=rIn_j[i]\right)\vee\left(rIn_j[i]=nil\wedge rOut[i]=nil\right)\right)\end{array}\right)$$ $$\vee\left(\begin{array}{l}\left(\left(rCtrl[i]=value\wedge s[i]=rCtrl[i]\right)\vee\left(rCtrl[i]=nil\wedge s[i]=s[i-1]\right)\right)\\ \wedge\left(s[i]<1\vee s[i]>inPort.width\right)\wedge rOut[i]=nil\end{array}\right)$$ $$s[0]=-1$$ 注：inPort.width 可通过看 Multiplexor actor 的 input 连接几条 relation 获知.
		输出底端控制端口指定	设左侧 trueInput, falseInput 端口所连接 relation 分别为 rtIn, rfIn，底端端口所连接 relation 为 rSec，右侧端口所连接 relation 为 rOut，则：

		的输入管道中的 token，而另一个输入管道中的 token 全部舍弃.	$\left(\begin{array}{l}(rSec[i]=true\vee(rSec[i]=nil\wedge s[i]=s[i-1]))\wedge s[i]="true"\\ \wedge((rtIn[i]!=nil\wedge rOut[i]=rtIn[i])\vee(rtIn[i]=nil\wedge rOut[i]=nil))\end{array}\right)$ $\vee\left(\begin{array}{l}(rSec[i]=\ false\vee(rSec[i]=nil\wedge s[i]=s[i-1]))\wedge s[i]="\ false"\\ \wedge((rfIn[i]!=nil\wedge rOut[i]=rfIn[i])\vee(rfIn[i]=nil\wedge rOut[i]=nil))\end{array}\right)$ $\vee(rSec[i]=nil\wedge s[i]=s[i-1]\wedge s[i]="nil"\wedge rOut[i]=nil)$ 这里的s是String类型，$s[0]="nil"$
		输出一个 record，且对于每个输入端口，该 record 会包含有一个与该输入端口同名的域.	令左侧输入端口名称依次为 fieldName1, fieldName2, fieldName3, …fieldNamen，且令这些端口所连接 relation 依次为: rIn_fieldName1, rIn_fieldName2, rIn_fieldName3, … rIn_fieldNamen，另令右侧端口所连接 relation 为 rOut，则 $\left(\bigvee_{j=1}^{n}(rIn\_\ fieldName_j[i]=nil)\wedge(rOut[i]=nil)\right)$ $\vee\left(\begin{array}{l}\neg\left(\bigvee_{j=1}^{n}(rIn\_\ fieldName_j[i]=nil)\right)\\ \wedge(rOut[i]=(valueRecord(rIn\_\ fieldName_1[i],rIn\_\ fieldName_2[i],rIn\_\ fieldName_3[i],\cdots rIn\_\ fieldName_n[i])))\end{array}\right)$
		从输入的 record 中提取域，且输出端口的名字必须与域名相匹配.	令左侧输入端口所连接 relation 为 rIn，令右侧输出端口名称依次为 fieldName1, fieldName2, fieldName3, …fieldNamen，且令这些端口所连接 relation 依次为: rOut_fieldName1, rOut_fieldName2, rOut_fieldName3, … rOut_fieldNamen，则： $\bigwedge_{j=1}^{n} rOut\_\ fieldName_j[i]=\ fieldName_j(rIn[i])$
		增加或修改所输入 record 中的 field，并输出更新后的 record.	令左侧内置输入端口所连接 relation 为 rIn，其它输入端口名称依次为 fieldName1, fieldName2, fieldName3, …fieldNamen，令这些端口所连接 relation 依次为: rIn_fieldName1, rIn_fieldName2, rIn_fieldName3, … rIn_fieldNamen，另令右侧输出端口所连接 relation 为 rOut，则： $\left(\begin{array}{l}\neg\left(\bigvee_{j=1}^{n}(rIn\_\ fieldName_j[i]=nil)\right)\\ \wedge\left((is-valueRecord(rOut[i]))\wedge\left(\bigwedge_{j=1}^{n}(fieldName_j(rOut[i])=rIn\_\ fieldName_j[i])\right)\wedge\left(\bigwedge_{\{i,j,k,\cdots\}\subseteq\{1,2,3,\cdots,n\}}(fieldName_j(rOut[i])=\ fieldName_j(rIn[i]))\right)\right)\\ \vee\left(\bigvee_{j=1}^{n}(rIn\_\ fieldName_j[i]=nil)\wedge(rOut[i]=nil)\right)\end{array}\right)$
Conver-sions		将一个 double 类型的数值转换为 int 型.	设左、右两侧端口所连接的 relation 分别为 rIn, rOut，Round 中参数 functionName，则 $\left(\begin{array}{l}rIn[i]!=nil\wedge rIn[i]!=to\_int(rIn[i])\wedge\\ \left(\begin{array}{l}(functionName=="ceil"\wedge rOut_i=to\_int(rIn[i])+1)\\ \vee(functionName=="floor"\wedge rOut_i=to\_int(rIn[i]))\\ \vee\left(functionName=="round"\wedge\left(\begin{array}{l}(rIn[i]-to\_int(rIn[i])>=0.5\wedge rOut[i]=to\_int(rIn[i])+1)\\ \vee(rIn[i]-to\_int(rIn[i])<0.5\wedge rOut[i]=to\_int(rIn[i]))\end{array}\right)\right)\\ \vee(functionName=="truncate"\wedge((rIn[i]>=0\wedge rOut[i]=to\_int(rIn[i]))\vee(rIn[i]<0\wedge rOut[i]=to\_int(rIn[i])+1)))\end{array}\right)\\ \vee(rIn[i]=nil\wedge rOut[i]=nil)\\ \vee(rIn[i]!=nil\wedge rIn[i]=to\_int(rIn[i])\wedge rOut[i]=rIn[i])\end{array}\right)$
		将布尔值转化为两个任意的值.	设左、右两侧端口所连接的 relation 分别为 rIn, rOut，则 $\left(\begin{array}{l}rIn[i]!=nil\wedge\left(\begin{array}{l}(rIn[i]=true\wedge rOut[i]=trueValue)\\ \vee(rIn[i]=\ false\wedge rOut[i]=\ falseValue)\end{array}\right)\\ \vee(rIn[i]=nil\wedge rOut[i]=nil)\end{array}\right)$ 注：默认情况下，trueValue 为 1，falseValue 为 0，如果更改的话，可从模型的 xml 文件中 BooleanToAnything actor 的 property 获得.
Random Number Generators		随机地输出布尔值 T 和 F.	设右侧端口所连接 relation 为 rOut，则 $rOut[i]=true\vee rOut[i]=\ false$
Array		判定数组中是否包含一	情形一：设左侧 element 端口未连接 relation, array 端口所连接 relation 为

Library		个指定元素.	rIn, 右侧端口所连接 relation 为 rOut，ArrayContains 中参数 elementParameter，则 $(rIn[i]=nil \wedge rOut[i]=nil)$ $\vee\left(\begin{array}{l} rIn[i]=value \\ \wedge\left(\begin{array}{l}(rIn[i].contains(elemParameter)=true \wedge rOut[i]=true) \\ \vee(rIn[i].contains(elemParameter)=\ false \wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)$ 情形二：设左侧 element 端口连接 relation 为 reIn, array 端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut，ArrayContains 中参数 elementParameter，创建变量 ST_ArrayContains，element 辅助编码，则 $\left(\begin{array}{l} ST\_ArrayContains[i-1]=0 \wedge \left(\begin{array}{l}(rIn[i]=nil \wedge rOut[i]=nil) \\ \vee\left(\begin{array}{l} rIn[i]=value \wedge\left(\begin{array}{l}(reIn[i]=nil \wedge element[i]=elementParameter) \\ \vee(reIn[i]=value \wedge element[i]=reIn[i])\end{array}\right) \\ \wedge\left(\begin{array}{l}(rIn[i].contains(element[i])=true \wedge rOut_i=true) \\ \vee(rIn[i].contains(element[i])=\ false \wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)\end{array}\right) \\ \wedge ST\_ArrayContains[i]=1\end{array}\right)$ $\vee\left(\begin{array}{l} ST\_ArrayContains[i-1]=1 \wedge \left(\begin{array}{l}(rIn[i]=nil \wedge rOut[i]=nil) \\ \vee\left(\begin{array}{l} rIn[i]=value \wedge\left(\begin{array}{l}(reIn[i]=nil \wedge element[i]=element[i-1]) \\ \vee(reIn[i]=value \wedge element[i]=reIn[i])\end{array}\right) \\ \wedge\left(\begin{array}{l}(rIn[i].contains(element[i])=true \wedge rOut[i]=true) \\ \vee(rIn[i].contains(element[i])=\ false \wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)\end{array}\right) \\ \wedge ST\_ArrayContains[i]=1\end{array}\right)$
		提取出数组中的一个元素.	情形一：设左侧 element 端口未连接 relation, array 端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut，ArrayElement 中参数 indexParameter，则 $(rIn[i]=value \wedge rOut[i]=select(rIn[i],indexParameter)) \vee (rIn[i]=nil \wedge rOut[i]=nil)$ 情形二：设左侧 element 端口连接 relation 为 rIndex, array 端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut，创建变量 ST_ArrayElement，index 辅助编码，则 $\left(\begin{array}{l} ST\_ArrayElement[i-1]=0 \wedge \left(\begin{array}{l}(rIn[i]=nil \wedge rOut[i]=nil) \\ \vee\left(\begin{array}{l} rIn[i]=value \wedge rOut[i]=select(rIn[i],index[i]) \\ \wedge\left(\begin{array}{l}(rIndex[i]=nil \wedge index[i]=indexParameter) \\ \vee(rIndex[i]=value \wedge index[i]=rIndex[i])\end{array}\right)\end{array}\right)\end{array}\right) \\ \wedge ST\_ArrayElement[i]=1\end{array}\right)$ $\vee\left(\begin{array}{l} ST\_ArrayElement[i-1]=1 \wedge \left(\begin{array}{l}(rIn[i]=nil \wedge rOut[i]=nil) \\ \vee\left(\begin{array}{l} rIn[i]=value \wedge rOut[i]=select(rIn[i],index[i]) \\ \wedge\left(\begin{array}{l}(rIndex[i]=nil \wedge index[i]=index[i-1]) \\ \vee(rIndex[i]=value \wedge index[i]=rIndex[i])\end{array}\right)\end{array}\right)\end{array}\right) \\ \wedge ST\_ArrayElement[i]=1\end{array}\right)$
		输出所输入数组的长度.	设左侧端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut,则: $(rIn[i]=nil \wedge rOut[i]=nil) \vee (rIn[i]=value \wedge rOut[i]=rIn[i].length)$
		查找数组中的最大元素.	设左侧端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut，rIndex, 则 $(rIn[i]=nil \wedge rOut[i]=nil \wedge rIndex[i]=nil)$ $\vee\left(\begin{array}{l} rIn[i]=value \\ \wedge rOut[i]=rIn[i].max.data \wedge rIndex[i]=rIn[i].max.index\end{array}\right)$
		查找数组中的最小元素.	设左侧端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut, rIndex 则 $(rIn[i]=nil \wedge rOut[i]=nil \wedge rIndex[i]=nil)$ $\vee\left(\begin{array}{l} rIn[i]=value \\ \wedge rOut[i]=rIn[i].min.data \wedge rIndex[i]=rIn[i].min.index\end{array}\right)$

		将数组中的元素输出到输出端口的每个 channel 上.	设左侧端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut, 则 $\left(rIn[i]=nil\wedge\bigwedge_{j=1}^{outPort.width}\left(rOut_j[i]=nil\right)\right)\vee\left(rIn[i]=value\wedge\left(\bigwedge_{j=1}^{outPort.width}\left(rOut_j[i]=select\left(rIn[i],j-1\right)\right)\right)\right)$
		改变数组中的一个元素，并将改变后的数组输出.	设左侧端口所连接 relation 为 rIn, 底端端口所连接 relation 为 rIndex 和 rValue，右侧端口所连接 relation 为 rOut，ArrayUpdate 中参数 valueParameter，indexParameter，则 $\left(\begin{array}{l}rIn[i]=vlaue\\\wedge\left(\left(index[i]=index[i-1]\wedge rIndex[i]=nil\right)\vee\left(rIndex[i]=value\wedge index[i]=rIndex[i]\right)\right)\\\wedge\left(\left(rValue[i]=nil\wedge value[i]=value[i-1]\right)\vee\left(rValue[i]=value\wedge value[i]=rValue[i]\right)\right)\\\wedge\left(\begin{array}{l}\left(0<=index[i]<rIn[i].length\wedge rOut[i]=store\left(rIn[i],Index[i],value[i]\right)\right)\\\vee\left(\left(index[i]<0\vee index[i]>=rIn[i].length\right)\wedge rOut[i]=rIn[i]\right)\end{array}\right)\end{array}\right)$ $\vee\left(rIn[i]=nil\wedge rOut[i]=nil\right)$ $value[0]=valueParameter, index[0]=indexParameter$
		根据输入端口各个 channel 上的元素构建一个数组.	设左侧端口所连接 relation 依次为 rIn1, rIn2, rIn3, 右侧端口所连接 relation 为 rOut，则 $\left(\bigvee_{j=1}^{n}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}\bigwedge_{j=1}^{n}\left(rIn_j[i]=value\right)\\\wedge rOut[i]=\underbrace{store\left(store\left(\cdots store\left(store\left(store\left(rOut[i],0,rIn_1[i]\right),1,rIn_2[i]\right),2,rIn_3[i]\right)\cdots\right),(n-1),rIn_n[i]\right)}_{input.Width\text{个}store\text{函数嵌套}}\end{array}\right)$
Logic Library		输出与输入相反的值.	设左、右两侧端口所连接 relation 分别为 rIn, rOut，则 $\left(rIn[i]=value\wedge rOut[i]=\neg rIn_i\right)\vee\left(rIn[i]=nil\wedge rOut[i]=nil\right)$
		输出左侧输入值的合取值(and)、析取值(or)、合取值的否定(nand)、析取值的否定(nor)、异或运算值(xor)、同或运算值(xnor).	①若 logic 为"and": 设左侧端口所连接 relation 为 rIn，右侧端口所连接 relation 为 rOut，则 $\left(\begin{array}{l}\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\wedge\left(\begin{array}{l}\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=true\vee rIn_j[i]=nil\right)\wedge rOut[i]=true\right)\\\vee\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=false\right)\wedge rOut[i]=false\right)\end{array}\right)\\\vee\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)\end{array}\right)$ ②若 logic 为"or"：设左侧端口所连接 relation 为 rIn，右侧端口所连接 relation 为 rOut，则 $\left(\begin{array}{l}\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\wedge\left(\begin{array}{l}\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=false\vee rIn_j[i]=nil\right)\wedge rOut[i]=false\right)\\\vee\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=true\right)\wedge rOut[i]=true\right)\end{array}\right)\\\vee\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)\end{array}\right)$ ③若 logic 为"nand"：设左侧端口所连接 relation 为 rIn，右侧端口所连接 relation 为 rOut，则 $\left(\begin{array}{l}\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\wedge\left(\begin{array}{l}\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=true\vee rIn_j[i]=nil\right)\wedge rOut[i]=false\right)\\\vee\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=false\right)\wedge rOut[i]=true\right)\end{array}\right)\\\vee\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)\end{array}\right)$

			④若 logic 为“nor”: 设左侧端口所连接 relation 为 rIn，右侧端口所连接 relation 为 rOut，则 $\left(\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\right)\wedge\left(\begin{array}{l}\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=false\vee rIn_j[i]=nil\right)\wedge rOut[i]=true\right)\\ \vee\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=true\right)\wedge rOut[i]=false\right)\end{array}\right)\right)$ $\vee\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)$ ⑤若 logic 为“xor”：设左侧端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut，创建数组 trueNumber，falseNumber，numberOfTrue，numberOfFalse 辅助编码，则： $\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\right)\\ \wedge\left(\bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}\left(rIn_j[i]=true\wedge trueNumber_j[i]=1\wedge falseNumber_j[i]=0\right)\\ \vee\left(\left(rIn_j[i]=false\vee rIn_j[i]=nil\right)\wedge trueNumber_j[i]=0\wedge falseNumber_j[i]=1\right)\end{array}\right)\right)\\ \wedge\left(numberOfTrue[i]=\sum_{j=1}^{inPort.width}trueNumber_j[i]\right)\wedge\left(numberOfFalse[i]=\sum_{j=1}^{inPort.width}falseNumber_j[i]\right)\\ \wedge\left(\begin{array}{l}\left(numberOfTrue[i]=0\wedge numberOfFalse[i]!=0\wedge rOut[i]=false\right)\\ \vee\left(numberOfTrue[i]!=0\wedge\left(\begin{array}{l}\left(numberOfTrue[i]\%2=0\wedge rOut[i]=false\right)\\ \vee\left(numberOfTrue[i]\%2=1\wedge rOut[i]=true\right)\end{array}\right)\right)\end{array}\right)\end{array}\right)$ $\vee\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)$ ⑥若 logic 为“xnor”：设设左侧端口所连接 relation 为 rIn, 右侧端口所连接 relation 为 rOut，创建数组 trueNumber，falseNumber，numberOfTrue，numberOfFalse 辅助编码，则： $\left(\begin{array}{l}\left(\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\right)\\ \wedge\left(\bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}\left(\left(rIn_j[i]=true\vee rIn_j[i]=nil\right)\wedge trueNumber_j[i]=1\wedge falseNumber_j[i]=0\right)\\ \vee\left(rIn_j[i]=false\wedge trueNumber_j[i]=0\wedge falseNumber_j[i]=1\right)\end{array}\right)\right)\\ \wedge\left(numberOfTrue[i]=\sum_{j=1}^{inPort.width}trueNumber_j[i]\right)\wedge\left(numberOfFalse[i]=\sum_{j=1}^{inPort.width}falseNumber_j[i]\right)\\ \wedge\left(\begin{array}{l}\left(numberOfTrue[i]=0\wedge numberOfFalse[i]!=0\wedge rOut[i]=true\right)\\ \vee\left(numberOfTrue[i]!=0\wedge\left(\begin{array}{l}\left(numberOfTrue[i]\%2=0\wedge rOut[i]=true\right)\\ \vee\left(numberOfTrue[i]\%2=1\wedge rOut[i]=false\right)\end{array}\right)\right)\end{array}\right)\end{array}\right)$ $\vee\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge rOut[i]=nil\right)$ 注：inPort.width 可通过看 LogicGate actor 的 input 连接几条 relation 获知.
		比较所输入的两个 double 类型的数值的大小，并输出布尔值，可能的比较符号 有：>,>=,<,<=及==.	设左侧端口所连接的 relation 分别为 rIn1, rIn2, 右侧端口所连接 relation 为 rOut，Comparator 内部参数 tolerance，则 ①若比较符号是 “==”，则 $\left(\left(rIn1[i]=nil\vee rIn2[i]=nil\right)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}\left(rIn1[i]=value\wedge rIn2[i]=value\right)\\ \wedge\left(\begin{array}{l}\left(\left(\left(abs\left(rIn1[i]-rIn2[i]\right)\right)<=tolerance\right)\wedge rOut[i]=true\right)\\ \vee\left(\left(\left(abs\left(rIn1[i]-rIn2[i]\right)\right)>tolerance\right)\wedge rOut[i]=false\right)\end{array}\right)\end{array}\right)$ ②若比较符号是 “>=”，则

			$\left((rIn1[i]=nil \vee rIn2[i]=nil)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}(rIn1[i]=value\wedge rIn2[i]=value)\\ \wedge\left(\begin{array}{l}((rIn1[i]-rIn2[i]+tolerance>=0)\wedge rOut[i]=true)\\ \vee((rIn1[i]-rIn2[i]+tolerance<0)\wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)$ ③若比较符号是“<=”，则 $\left((rIn1[i]=nil \vee rIn2[i]=nil)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}(rIn1[i]=value\wedge rIn2[i]=value)\\ \wedge\left(\begin{array}{l}((rIn2[i]-rIn1[i]+tolerance>=0)\wedge rOut[i]=true)\\ \vee((rIn2[i]-rIn1[i]+tolerance<0)\wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)$ ④若比较符号是“>”，则 $\left((rIn1[i]=nil \vee rIn2[i]=nil)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}(rIn1[i]=value\wedge rIn2[i]=value)\\ \wedge\left(\begin{array}{l}((rIn1[i]-rIn2[i]+tolerance>0)\wedge rOut[i]=true)\\ \vee((rIn1[i]-rIn2[i]+tolerance<=0)\wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)$ ⑤若比较符号是“<”，则 $\left((rIn1[i]=nil \vee rIn2[i]=nil)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}(rIn1[i]=value\wedge rIn2[i]=value)\\ \wedge\left(\begin{array}{l}((rIn2[i]-rIn1[i]+tolerance>0)\wedge rOut[i]=true)\\ \vee((rIn2[i]-rIn1[i]+tolerance<=0)\wedge rOut[i]=\ false)\end{array}\right)\end{array}\right)$
	Equals ==	比较所输入的任意数量的任意类型的 token 是否相等，如果相等，输出 true；否则，输出 false.	设左侧端口所连接的 relation 为 rIn,右侧端口所连接 relation 为 rOut，创建数组 s 辅助编码，则 $\left(\bigwedge_{j=1}^{inPort.width}\left(rIn_j[i]=nil\right)\wedge\neg\left(s_1[i]=s_2[i]=...=s_{inPort.width}[i]\right)\wedge rOut[i]=nil\right)$ $\vee\left(\begin{array}{l}\bigvee_{j=1}^{inPort.width}\left(rIn_j[i]=value\right)\\ \wedge\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]=value\wedge s_j[i]=rIn_j[i]\right)\vee\left(rIn_j[i]=nil\wedge s_j[i]=s_{j-1}[i]\right)\right)\\ \wedge\left(\begin{array}{l}\left(s_1[i]=s_2[i]=...=s_{inPort.width}[i]\wedge rOut[i]=true\right)\\ \vee\left(\neg\left(s_1[i]=s_2[i]=...=s_{inPort.width}[i]\right)\wedge rOut[i]=\ false\right)\end{array}\right)\end{array}\right)$ 注：inPort.width 可通过看 Equals actor 的 input 连接几条 relation 获知.
	IsPresent ?	若输入 present，则输出 true；否则输出 false.	设左侧多端口所连接的 relation 为 rIn，右侧多端口所连接的 relation 为 rOut，则 $\bigwedge_{j=1}^{inPort.width}\left(\left(rIn_j[i]=value\wedge rOut_j[i]=true\right)\vee\left(rIn_j[i]=nil\wedge rOut_j[i]=\ false\right)\right)$ 注：inPort.width 可通过看 IsPresent actor 的 input 连接几条 relation 获知.
	TrueGate true?	若输入为布尔值 true，则输出 true；否则不输出 token.	设左侧多端口所连接的 relation 依次为 rIn1, rIn2, rIn3，右侧多端口所连接的 relation 依次为 rOut1, rOut2, rOut3,则 $\bigwedge_{j=1}^{inPort.width}\left(\begin{array}{l}\left(rIn_j[i]=true\wedge rOut_j[i]=true\right)\\ \vee\left(\left(rIn_j[i]=\ false\vee rIn_j[i]=nil\right)\wedge rOut_j[i]=nil\right)\end{array}\right)$ 注：inPort.width 可通过看 TrueGate actor 的 input 连接几条 relation 获知.
DomainSpecific -- SR	NonStrictDelay	提供一个 tick 的延迟：每次激励时，输出上个 tick 时输入端口处所	设左、右两侧端口所连接的 relation 分别是 rIn 和 rOut，创建变量 lastInput 辅助编码，则

		读入的任何值；第一个tick时，输出值由参数initialValue给出；若未给定任何值，则输出absent.	$\left(\begin{array}{l}\left(rOut[i]=lastInput[i-1]\right)\\ \wedge\left(\begin{array}{l}\left(rIn[i]=value\right)\wedge\left(lastinput[i]=rIn[i]\right)\\ \vee\left(rIn[i]=nil\right)\wedge\left(lastinput[i]=nil\right)\end{array}\right)\end{array}\right)$
		基于底部信号来过滤当前输入信号：若底部控制端口输入为present且值为true，则输出为左侧的输入数据；而如果底部控制端口输入为absent, 或者为false，亦或者底部控制端口输入为present且值为true但左侧输入端口为absent，则输出为absent.	设左侧、底部和右侧端口所对应的relation分别是rIn, rCtrl 和 rOut,则： $\begin{array}{l}\left(rCtrl[i]=true\wedge rOut[i]=rIn[i]\right)\\ \vee\left(\left(rCtrl[i]=\ false\vee rCtrl[i]=nil\right)\wedge rOut[i]=nil\right)\end{array}$
		将两个信号有优先级地进行合并：若优先级较高的左侧端口输入为present，则输出为该输入数据；如果左侧输入为absent，则输出为底部的输入(无论其是否为absent).	设左侧、底部和右侧端口所对应的relation分别是rIn, rBot 和 rOut,则： $\begin{array}{l}\left(rIn[i]=value\wedge rOut[i]=rIn[i]\right)\\ \vee\left(\begin{array}{l}rIn[i]=nil\wedge\left(\begin{array}{l}\left(rBot[i]=nil\wedge rOut[i]=nil\right)\\ \vee\left(rBot[i]=value\wedge rOut[i]=rBot[i]\right)\end{array}\right)\end{array}\right)\end{array}$
		输出上次接收到的非absent输入.若当前输入是absent，则输出也是absent；第一次输入是present时，输出由参数initialValue给出(该参数默认为	设左侧和右侧端口所对应的relation分别是rIn和rOut, 创建变量s，index辅助编码，则 $\left(rIn[i]=nil\wedge rOut[i]=nil\wedge ST[i-1]=ST[i]\wedge index[i]=index[i-1]\right)$ $\vee\left(\begin{array}{l}rIn[i]=value\wedge ST[i]=ST[i-1]+1\wedge s_{index[i]}=rIn[i]\\ \wedge\left(\begin{array}{l}\left(ST[i-1]=0\wedge rOut[i]=initialValue[i]\wedge index[i]=index[i-1]\right)\\ \vee\left(ST[i]\geq 1\wedge rOut[i]=s_{index[i-1]}\wedge index[i]=index[i-1]+1\right)\end{array}\right)\end{array}\right)$

		absent).	
		输出最新接收到的非absent输入，若没有接收到输入，则输出为absent.	设左侧和右侧端口所对应的 relation 分别是 rIn 和 rOut, 创建变量 s 辅助编码，则 $(rIn[i]=value \wedge rOut[i]=rIn[i] \wedge s[i]=rIn[i])$ $\vee\begin{pmatrix} rIn[i]=nil \wedge s[i]=s[i-1] \\ \wedge((rOut[i]=s[i] \wedge s[i]=value) \vee (s[i]=nil \wedge rOut[i]=nil)) \end{pmatrix}$ $s[0]=nil$
		输出总是absent.	设右侧端口所对应的 relation 是 rOut, 则 $rOut[i]=nil$
		若输入为present，则输出true;否则输出false.	设左、右侧端口所对应 relation 分别是 rIn1, rIn2 和 rOut1, rOut2，则: $\bigwedge_{j=1}^{inPort.width}((rIn_j[i]=value \wedge rOut_j[i]=true) \vee (rIn_j[i]=nil \wedge rOut_j[i]=false))$ 注：inPort.width 可通过看 IsPresent actor 的 input 连接几条 relation 获知.
		若输入为present且值为true,则输出true;否则输出 absent.	设左、右侧端口所对应的 relation 分别是 rIn1, rIn2 和 rOut1, rOut2，则: $\bigwedge_{j=1}^{inPort.width}((rIn_j[i]=true \wedge rOut_j[i]=true) \vee ((rIn_j[i]=nil \vee rIn_j[i]=false) \wedge rOut_j[i]=nil))$ 注：inPort.width 可通过看 TrueGate actor 的 input 连接几条 relation 获知.
Sink		设定由variableName 参数所指定的变量的取值，且有两种情况：情况一(delayed=false): 在对该 actor 进行 fire 时，设定变量的取值为输入值，然后输出与输入端口相同的 token; 情况二(delayed=true)：输出所访问变量的当前值，且所访问变量在当前迭代运行结束时才进行设定。	情况一：令输入端口所连接 relation 为 rIn, 令 variableName 为 x, delayed=false，另令输出端口所连接 relation 为 rOut，则： $(is-vaue(rIn[i]) \wedge x[i]=rIn[i] \wedge rOut[i]=rIn[i])$ $\vee(rIn[i]=absent \wedge x[i]=x[i-1] \wedge rOut[i]=absent)$; 情况二：令输入端口所连接 relation 为 rIn, 令 variableName 为 x, delayed=true，另令输出端口所连接 relation 为 rOut，则： $(is-vaue(rIn[i]) \wedge x[i]=rIn[i] \wedge rOut[i]=x[i-1])$ $\vee(rIn[i]=absent \wedge x[i]=x[i-1] \wedge rOut[i]=x[i-1])$