دانشگاه صنعتی اصفهان

ارائه دهنده : حسن موسوی

استاد راهنما : دکتر الهام محمود زاده

۲۹ خرداد ۱۳۹۸

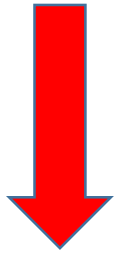
هدف از موازی سازی (چند نخی) چیست؟

هدف از موازی سازی (چند نخی) چیست؟

افزایش سرعت پردازش

هدف از موازی سازی (چند نخی) چیست؟

افزایش سرعت پردازش



به نحوه ای که خروجی قابل انتظار تغییری نکند

چند نخی *Multi Threading*

Thread Q	Thread P
Void Q( ) **{** **n = 1;** **cout << “Thread Q,n=“ <<n << endl;** **}**	**Void P( )** **{** **n = 2;** **cout << “Thread P,n=“ <<n <<endl;** **}**

چند نخی *Multi Threading*

Thread Q	Thread P
Void Q( ) **{** **n = 1;** **cout << “Thread Q,n=“ <<n << endl;** **}**	**Void P( )** **{** **n = 2;** **cout << “Thread P,n=“ <<n <<endl;** **}**

خروجی بدست آمده	خروجی مورد انتظار
	Thread Q , n = ۱ **Thread P , n = 2**

چند نخی *Multi Threading*

Thread Q	Thread P

Void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	Void P( )
	{
	n = 2;
	cout << “Thread P,n=“ <<n <<endl;
	}

خروجی بدست آمده	خروجی مورد انتظار
	Thread Q , n = ۱ Thread P , n = 2

چند نخی *Multi Threading*

Thread Q	Thread P

Void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	Void P( )
	{
	n = 2;
	cout << “Thread P,n=“ <<n <<endl;
	}

خروجی بدست آمده	خروجی مورد انتظار
Thread Q , n = ۱	Thread Q , n = ۱ Thread P , n = 2

چند نخی *Multi Threading*

Thread Q	Thread P

Void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	Void P( )
	{
	n = 2;
	cout << “Thread P,n=“ <<n <<endl;
	}

خروجی بدست آمده	خروجی مورد انتظار
Thread Q , n = ۱	Thread Q , n = ۱ Thread P , n = 2

چند نخی *Multi Threading*

Thread Q	Thread P

Void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	Void P( )
	{
	n = 2;
	cout << “Thread P,n=“ <<n <<endl;
	}

خروجی بدست آمده	خروجی مورد انتظار
Thread Q , n = ۱ Thread P , n = 2	Thread Q , n = ۱ Thread P , n = 2

چند نخی *Multi Threading*

Thread Q	Thread P

Void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	Void P( )
	{
	n = 2;
	cout << “Thread P,n=“ <<n <<endl;
	}

خروجی بدست آمده	خروجی مورد انتظار
Thread Q , n = ۱ Thread P , n = 2	Thread Q , n = ۱ Thread P , n = 2

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی
void Q( ) **{** **n = 1;** **cout << “Thread Q,n=“ <<n << endl;** **}**	**void P()** **{** **n = 2;** **cout << “Thread P,n=“ <<n << endl;** **}**		

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
 
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}		

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
 
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
	
		1
		 	

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
		1	

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
 
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
	 
		2
		 	

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread Q,n=“ <<n << endl;
	}
		2	

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
		2	Thread P,n = 2

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
		2	Thread P,n = 2

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
		2	Thread P,n = 2
			Thread Q,n =2

چند نخی *Multi Threading*

Thread Q	Thread P	مقدار n	خروجی نهایی

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}
		2	Thread P,n = 2
			Thread Q,n =2

چند نخی *Multi Threading* – ناحیه بحرانی

Thread Q	Thread P

void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}

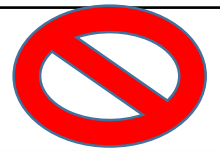
چند نخی *Multi Threading* – ناحیه بحرانی

Thread Q	Thread P

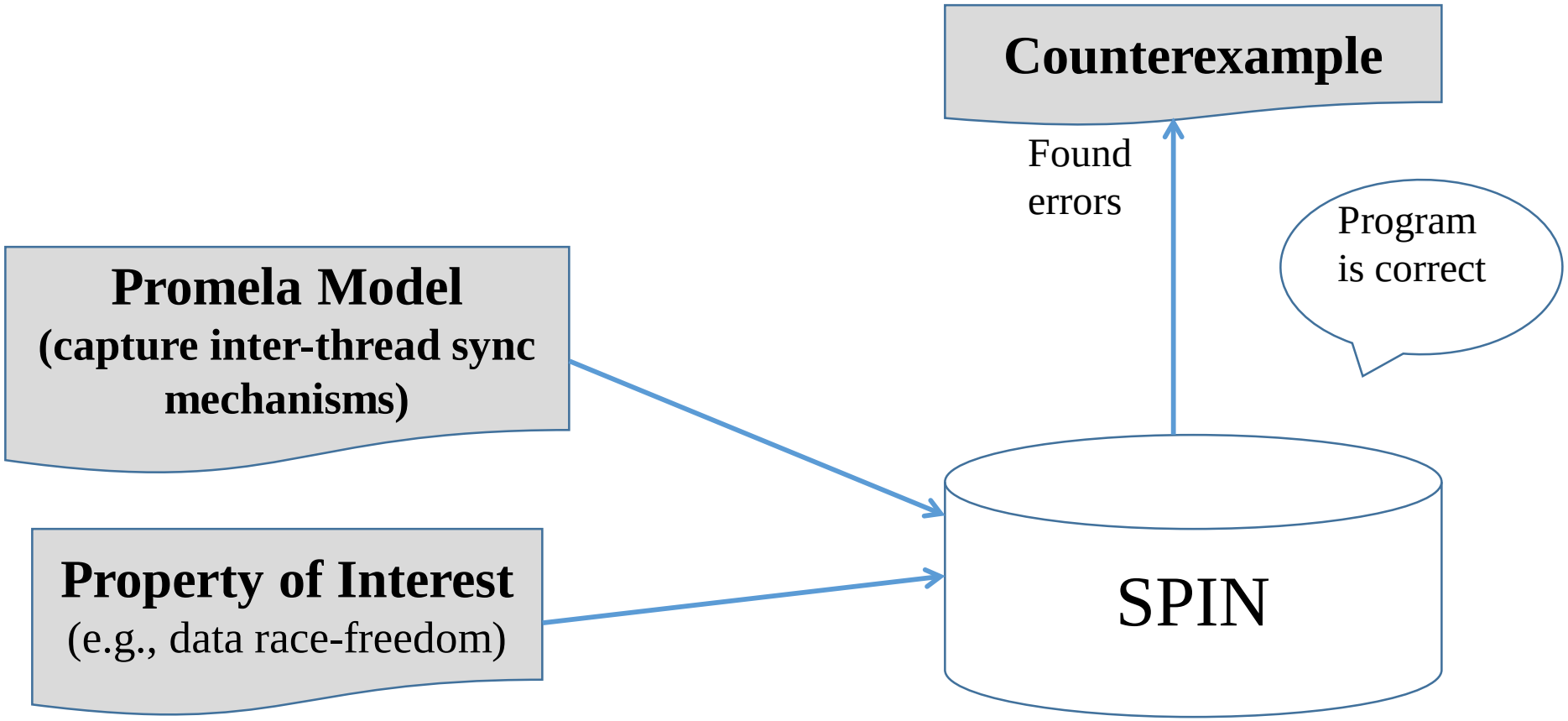
void Q( )
{
n = 1;
 
cout << “Thread Q,n=“ <<n << endl;
}
 	
	void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}

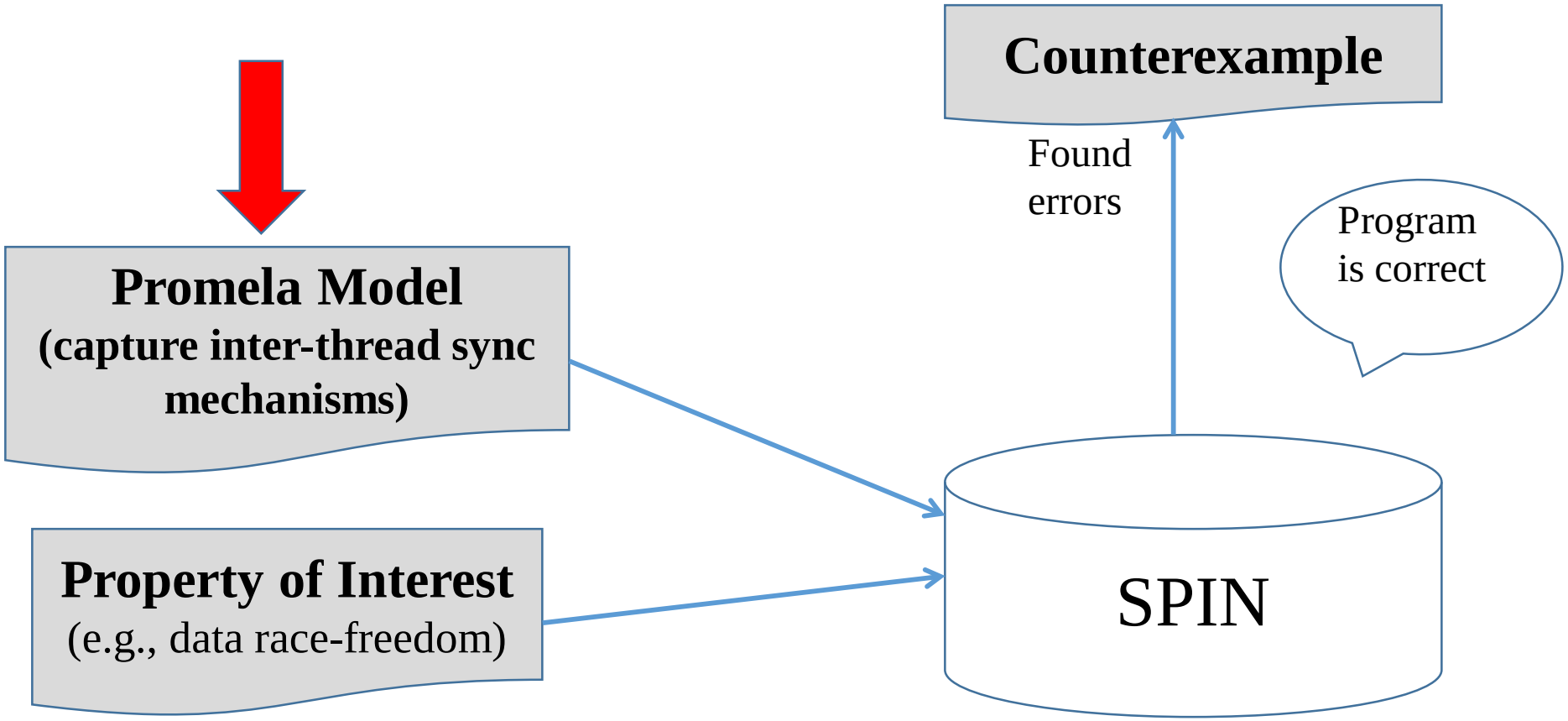
چند نخی Multi Threading – شرایط رقابت Race Condition

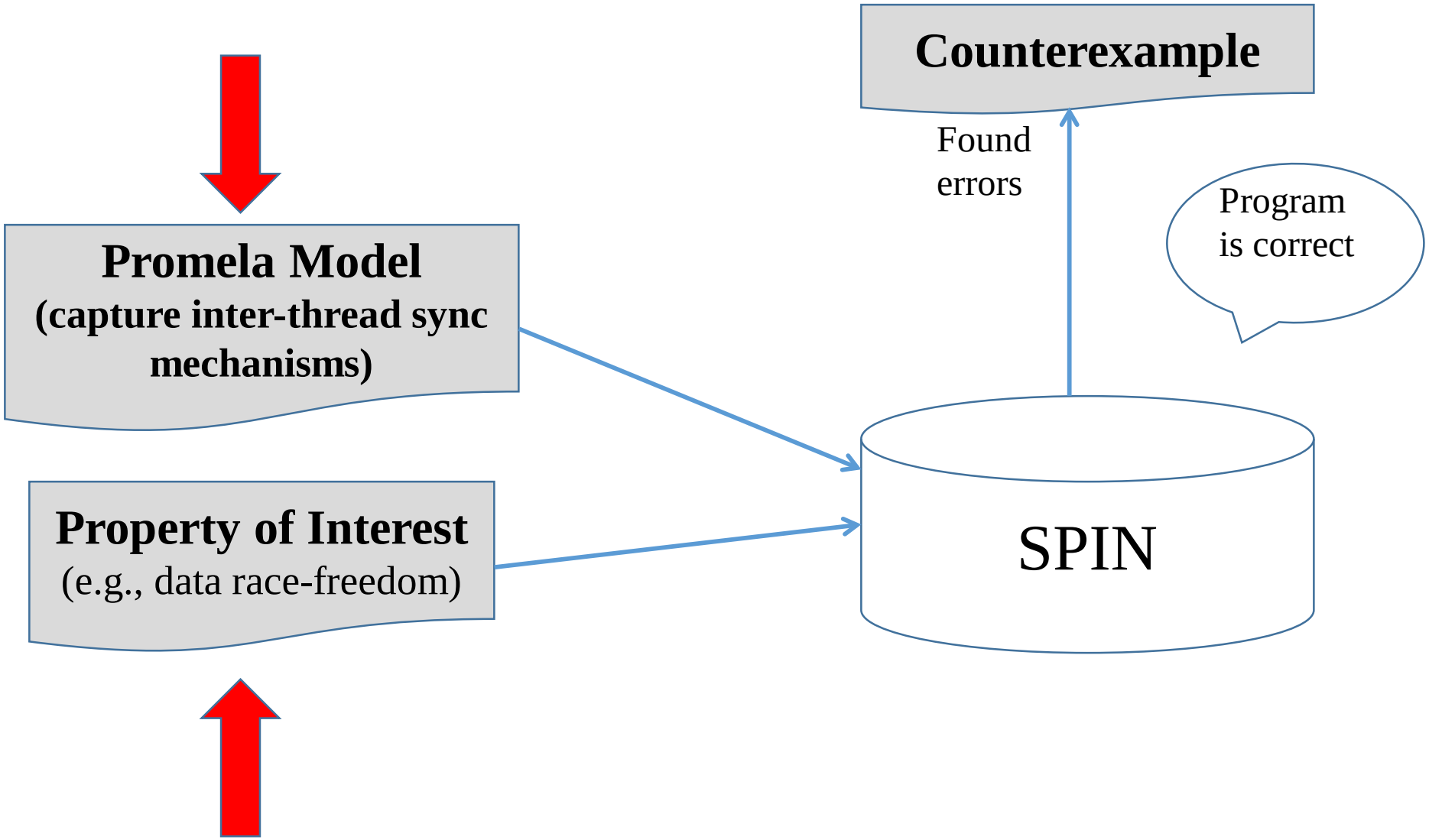
Thread Q	Thread P

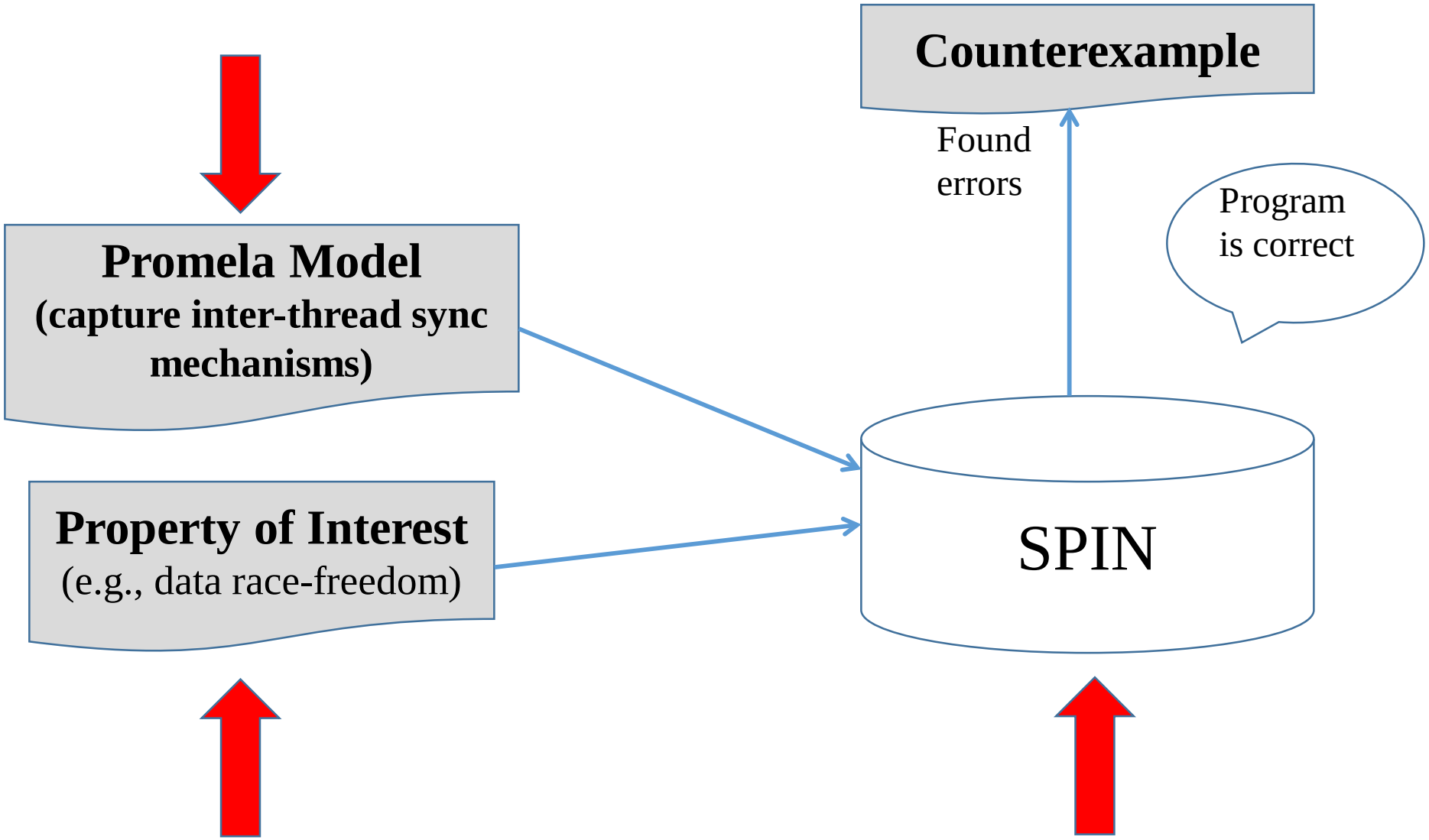
void Q( )
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	void P()
	 
	{
	n = 2;
	cout << “Thread P,n=“ <<n << endl;
	}

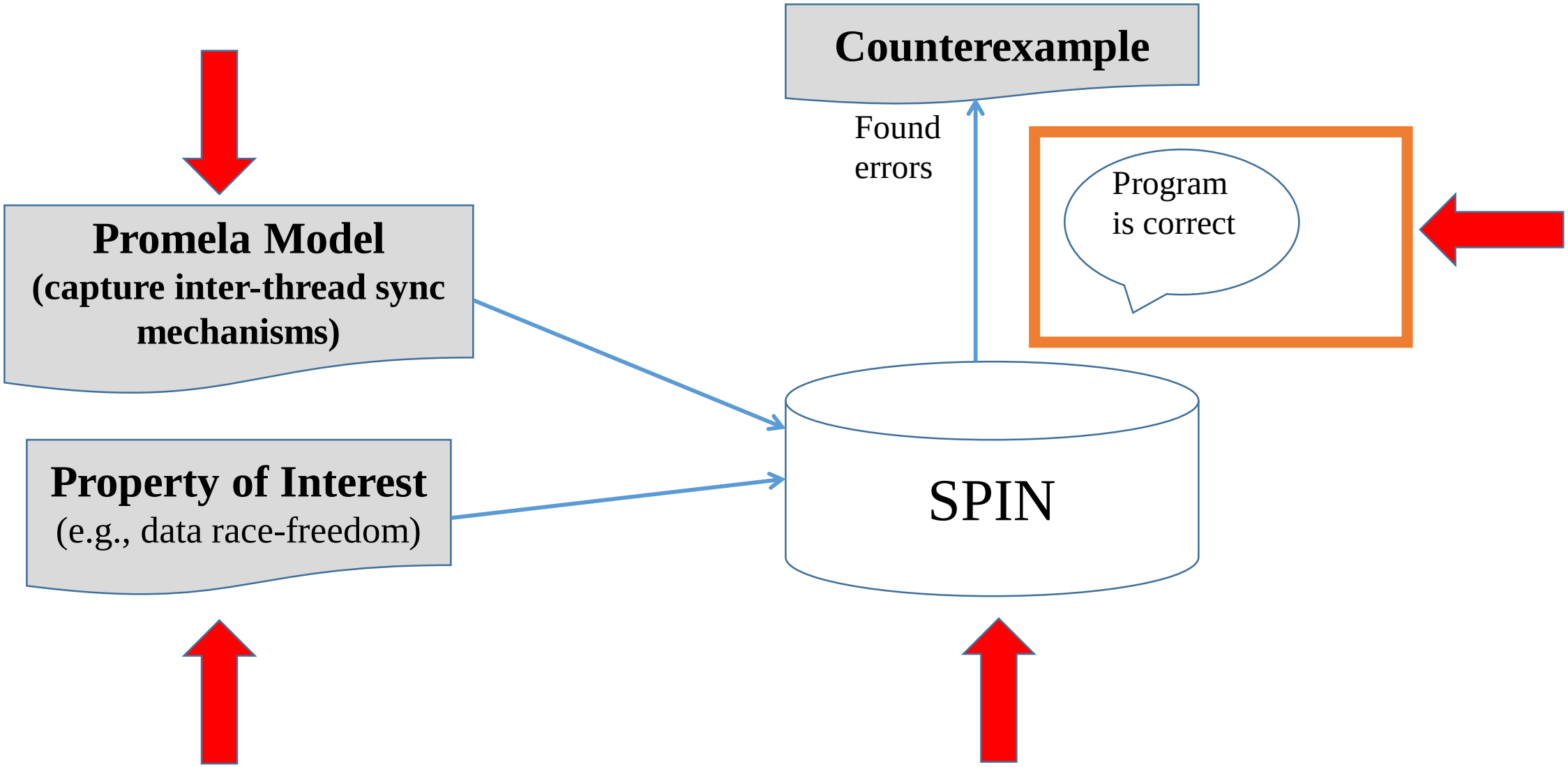
```
shared double t[regLen*THREADS];     double tmp[2], e, etmp;
        . . . // Perform some local computations
upc_barrier;
base = MYTHREAD*regLen;
 for (j =0; j < regLen+1; j++) {  // start of the main loop
     if (MYTHREAD == 0) { tmp[0] = t[0]; }  else {
                                                tmp[0] = (t[base-1] + t[base] + t[base+1])/3.0;
                                                e = fabs(t[base]-tmp[0]);       }  //else
    for (i=base+1; i<base+regLen-1; ++i) {
                tmp[1] = (t[i-1] + t[i] + t[i+1]) / 3.0;
                etmp = fabs(t[i]-tmp[1]);
                t[i-1] = tmp[0];                        }  // for
     if (MYTHREAD < THREADS-1) {
            tmp[1] = (t[base+regLen-2] +  t[base+regLen-1] + t[base+regLen]) / 3.0;
            etmp = fabs(t[base+regLen-1]-tmp[1]);
             t[base+regLen-1] = tmp[1];        }
       upc_barrier;
        t[base+regLen-2] =  some local expression;
 }  // end of the main for-loop
```

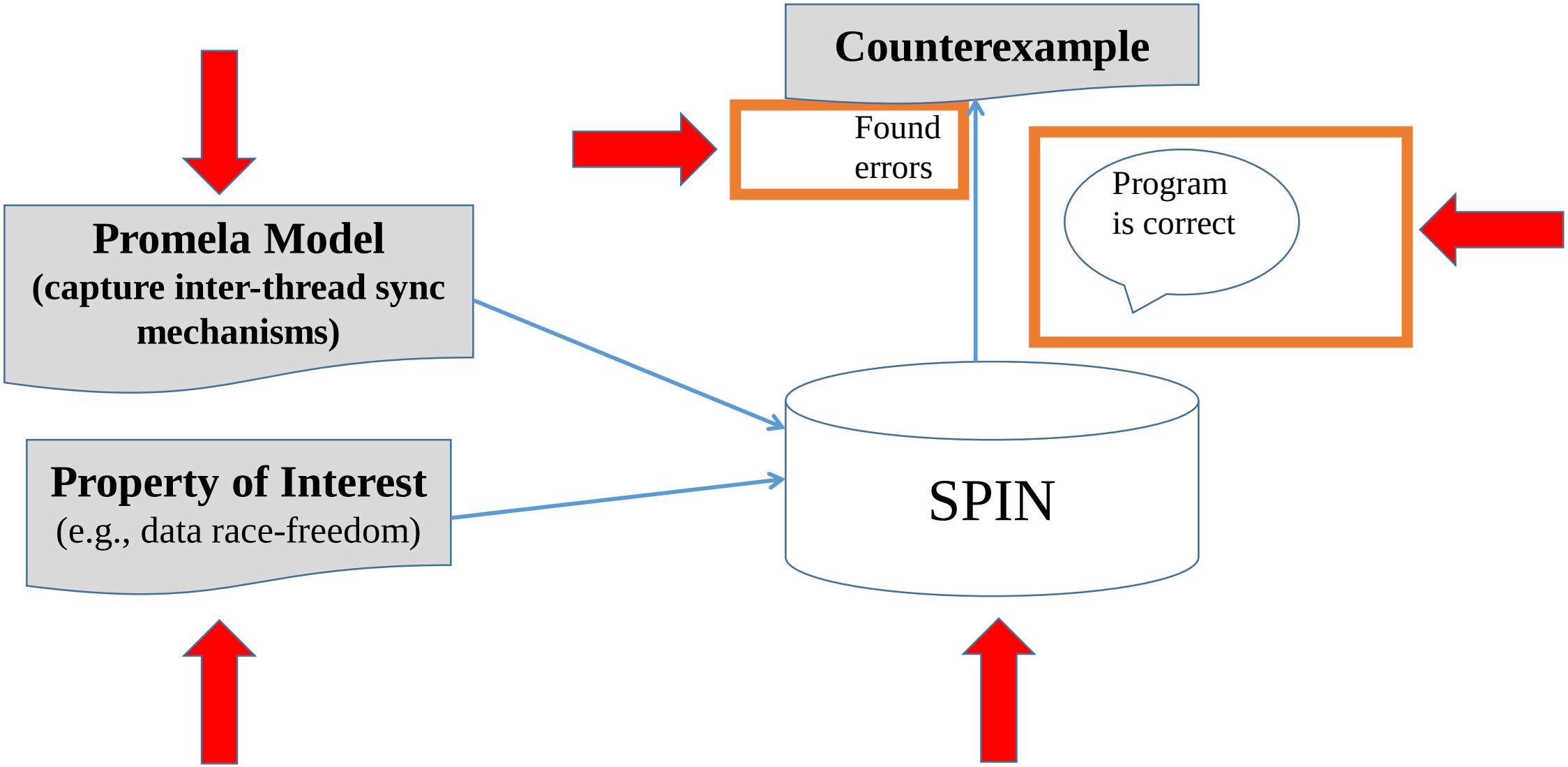

Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN

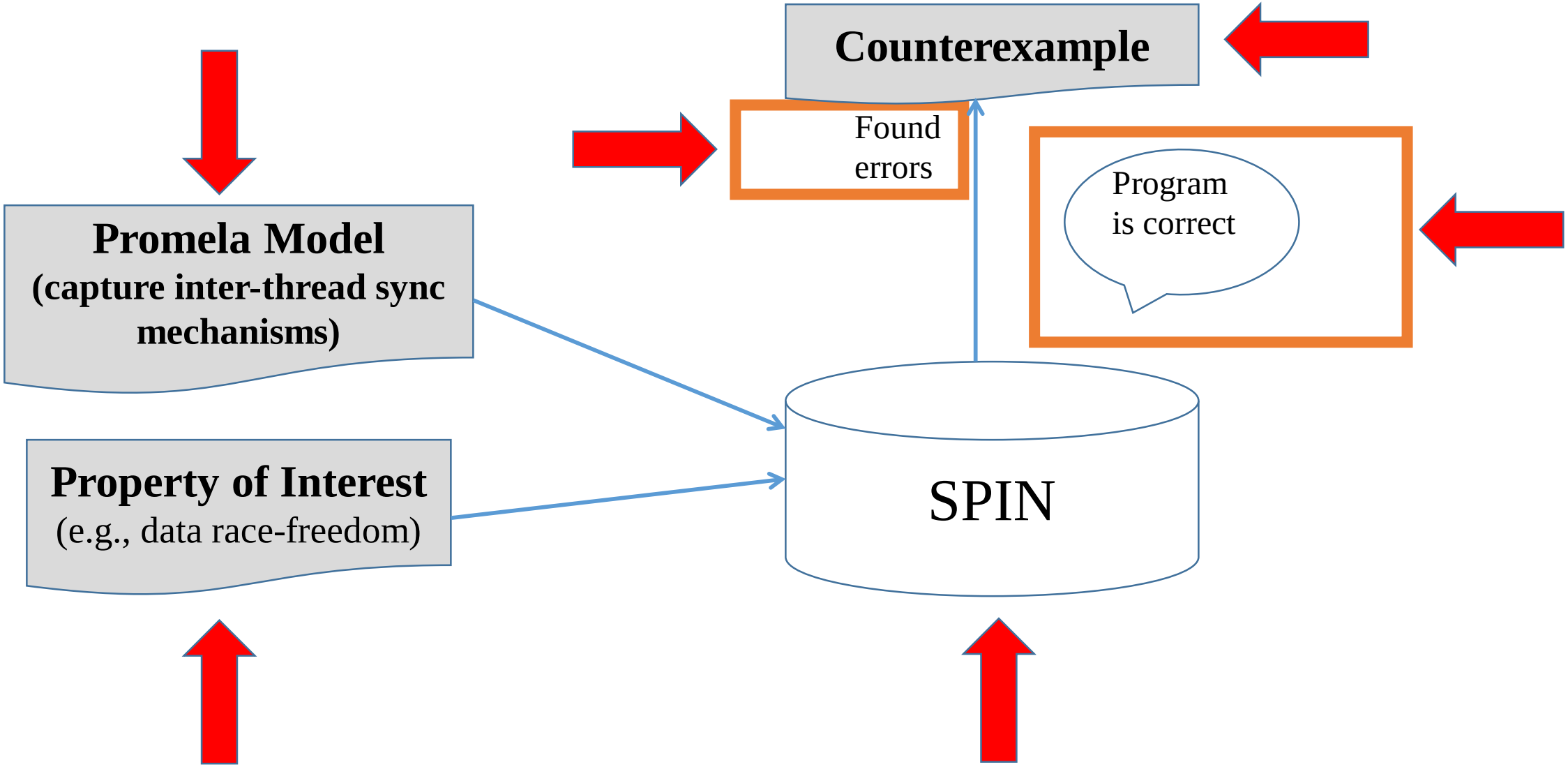

Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN


Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN


Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN


Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN


Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN


Counterexample
Found errors
Program is correct
Promela Model
(capture inter-thread sync mechanisms)
Property of Interest
(e.g., data race-freedom)
SPIN

زبان *Promela*

زبان *Promela*

- یک زبان برنامه نویسی نیست و یک زبان مدلسازی است.

زبان ***Promela***

- یک زبان برنامه نویسی نیست و یک زبان مدلسازی است.
- **ساختار و دستوراتی شبیه به زبان *C* دارد.**

زبان ***Promela***

- یک زبان برنامه نویسی نیست و یک زبان مدلسازی است.
- ساختار و دستوراتی شبیه به زبان C دارد.

چند نخی *Multi Threading* - مدلسازی

Thread Q	Thread P

Void Q()
{
n = 1;
 	
cout << “Thread Q,n=“ <<n << endl;
}
	Void P()
	{
	n = 2;
	
	cout << “Thread P,n=“ <<n<<endl;
	}

active proctype Q()
{
n = 1;
 	
printf(“Thread Q, n= %d\n”)
}
	active proctype P()
	{
	n = 2;
	
	printf(“Thread P, n= %d\n”)
	}

چند نخی *Multi Threading*- مدلسازی

Thread Q	Thread P

Void Q()
{
n = 1;
 	
cout << “Thread Q,n=“ <<n << endl;
}
	Void P()
	{
	n = 2;
	
	cout << “Thread P,n=“ <<n<<endl;
	}

active proctype Q()
{
n = 1;
 	
printf(“Thread Q, n= %d\n”)
}
	active proctype P()
	{
	n = 2;
	
	printf(“Thread P, n= %d\n”)
	}

چند نخی *Multi Threading* -مدلسازی

Thread Q	Thread P

Void Q()
{
n = 1;
 	
cout << “Thread Q,n=“ <<n << endl;
}
	Void P()
	{
	n = 2;
	
	cout << “Thread P,n=“ <<n<<endl;
	}

active proctype Q()
{
n = 1;
 	
printf(“Thread Q, n= %d\n”)
}
	active proctype P()
	{
	n = 2;
	
	printf(“Thread P, n= %d\n”)
	}

چند نخی *Multi Threading* - مدلسازی

Thread Q	Thread P

Void Q()
{
n = 1;
cout << “Thread Q,n=“ <<n << endl;
}
	Void P()
	{
	n = 2;
	cout << “Thread P,n=“ <<n<<endl;
	}

active proctype Q()
{
n = 1;
printf(“Thread Q, n= %d\n”)
}
	active proctype P()
	{
	n = 2;
	printf(“Thread P, n= %d\n”)
	}

چند نخی *Multi Threading* – مثال شماره ۱

Thread Q	Thread P

Void Q()
{
n = 1;
 	
cout << “Thread Q,n=“ <<n << endl;
}
	Void P()
	{
	n = 2;
	
	cout << “Thread P,n=“ <<n<<endl;
	}

active proctype Q()
{
n = 1;
 	
printf(“Thread Q, n= %d\n”)
}
	active proctype P()
	{
	n = 2;
	
	printf(“Thread P, n= %d\n”)
	}

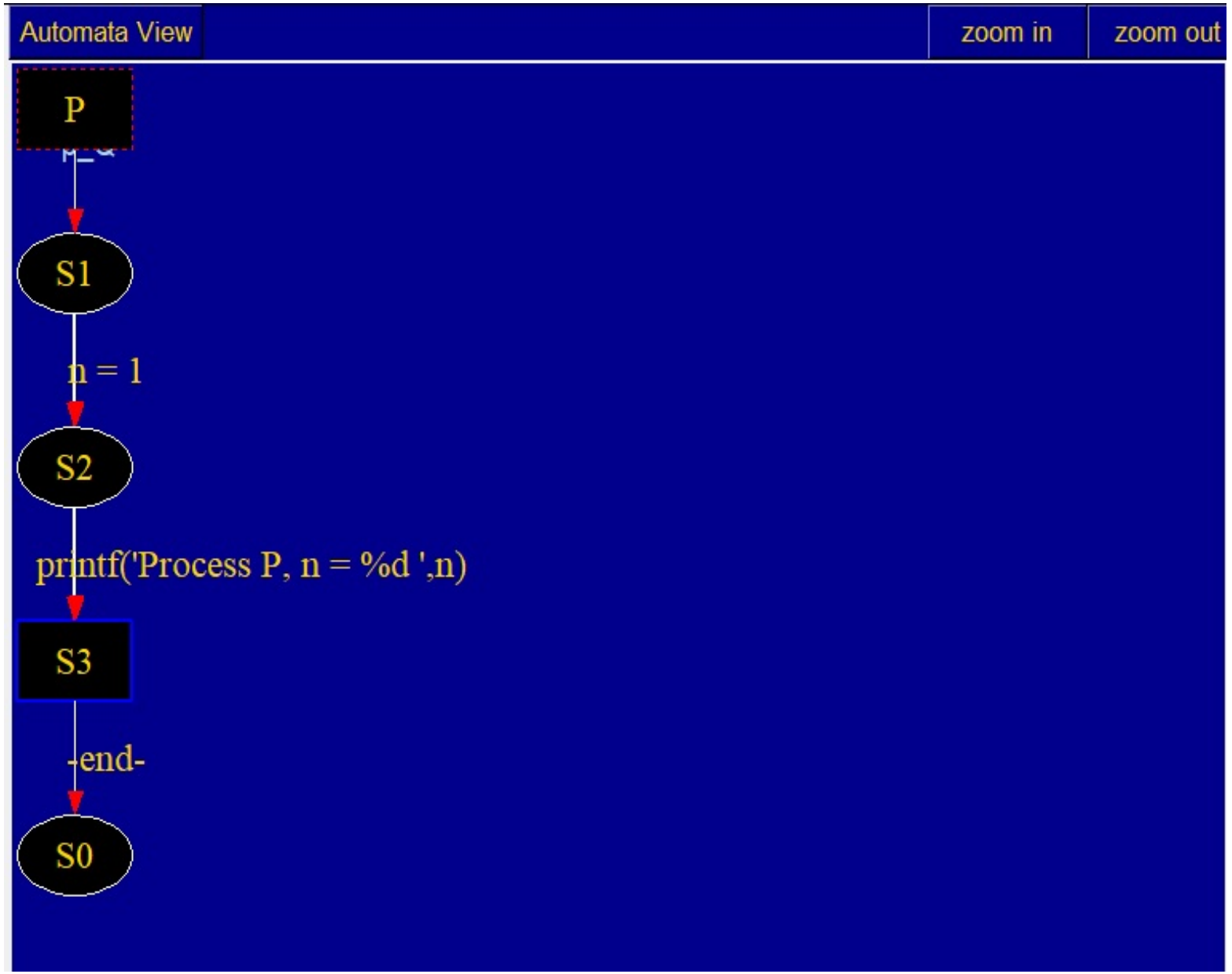
شبیه سازی تصادفی *Random Simulation*

شبیه سازی تصادفی *Random Simulation*

مقادیر متغیرها *Variable values*	مسیر اجرایی *Execution Path*
[variable values, step 3] Process P, n = 2 Process Q, n = 2 n = 2	0: proc - (:root:) creates proc 0 (P) 0: proc - (:root:) creates proc 1 (Q) spin: concurent.pml:0, warning, global, 'byte n' variable is never used (other than in print stmnts) 1: proc 0 (P:1) concurent.pml:4 (state 1) [n = 1] 2: proc 1 (Q:1) concurent.pml:9 (state 1) [n = 2] 3: proc 0 (P:1) concurent.pml:5 (state 2) [printf('Process P, n = %d\\n',n)] 4: proc 1 (Q:1) concurent.pml:10 (state 2)[printf('Process Q, n = %d\\n',n)] 4: proc 1 (Q:1) terminates 4: proc 0 (P:1) terminates 2 processes created

ماشین حالت *Automata*

ماشین حالت *Automata*



شبیه سازی *Interactive*

شبیه سازی *Guided, with trail*

انواع داده ای پایه (*Basic Data Types*) در زبان *Promela*

نام	معادل C	محدوده
bit or bool	*bit-field*	*0 … 1*
byte	*uchar*	*0 … 255*
short	*short*	*-2 ^ 15 -1 … 2 ^ 15 -1*
int	*int*	*-2 ^ 31 -1 … 2 ^ 31 - 1*

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	bit	break
chan	do	d_step	else	fi
goto	if	init	int	od
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	break
chan	do	d_step	else	fi
goto	if	init	**int**	od
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	break
chan	do	d_step	else	fi
goto	if	init	**int**	od
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	break
chan	do	d_step	else	**fi**
goto	**if**	init	**int**	od
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	break
chan	do	d_step	else	**fi**
goto	**if**	init	**int**	od
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	break
chan	**do**	d_step	else	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	break
chan	**do**	d_step	else	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	else	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	else	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	typedef

کلمات کلیدی *keywords* در زبان *Promela*

active	assert	atomic	**bit**	**break**
chan	**do**	d_step	**else**	**fi**
goto	**if**	init	**int**	**od**
printf	proctype	run	skip	**typedef**

اجرای دستورات در *Promela*

اجرای دستورات در *Promela*

- قابلیت اجرا شدن دارند

اجرای دستورات در *Promela*

- قابلیت اجرا شدن دارند
- **بلوکه شدن تا زمانی که قابلیت اجرا شدن پیدا شود**

اجرای دستورات در *Promela*

- قابلیت اجرا شدن دارند
- بلوکه شدن تا زمانی که قابلیت اجرا شدن پیدا شود

int a,b; ……..; ……..; **While (a == b)** **;** ……..; ……..;	int a; int b; ……..; ……..; **(a == b);** ……..; ……..;
C ++	***Promela***

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

برخی از دستوراتی که اجرا می شوند	برخی از دستوراتی که بر اساس شرایطی اجرا می شوند
int a;	if :: (a ==1) -> skip; fi
a = 1;	(a == b);
skip;	do :: (a < 10) -> a ++; od
break;	

int a,b; ……..; ……..; if (a > b) cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: (a > b) -> printf(“a is greater than b”); fi; ……..; ……..;
C ++	***Promela***

int a,b; ……..; ……..; **if (a > b)** cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: **(a > b)** -> printf(“a is greater than b”); fi; ……..; ……..;
C ++	***Promela***

C ++	Promela
int a,b; ……..; ……..; **if (a > b)** cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: **(a > b)** -> printf(“a is greater than b”); fi; ……..; ……..;

int a,b; ……..; ……..; if (a > b) cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: (a > b) -> printf(“a is greater than b”); fi; ……..; ……..;
C ++	***Promela***

```
int a,b;
……..;
……..;

if ( a > b )
    cout << " a is greater than b";

……..;
……..;
```

```
int a;
int b;
……..;
……..;
if
:: ( a > b ) -> printf( "a is greater than b");
fi;
……..;
……..;
```

C ++	***Promela***

int a,b; ……..; ……..; if (a > b) cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: (a > b) -> printf(“a is greater than b”); fi; ……..; ……..;
C ++	***Promela***

C ++	Promela
int a,b; ……..; ……..; **if (a > b)** cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: **(a > b)** -> printf(“a is greater than b”); **:: else -> skip** fi; ……..; ……..;

C ++	Promela
int a,b; ……..; ……..; **if (a > b)** cout << “ a is greater than b”; ……..; ……..;	int a; int b; ……..; ……..; if :: **(a > b)** -> printf(“a is greater than b”); **:: else -> skip** fi; ……..; ……..;

چند مثال دیگر

```
if
:: ( a == 1) -> state++;
:: ( b == 1) -> state--;
:: ( c == 1) -> skip;
fi
```

```
do
:: ( a == b) -> break;
:: ( a == c) -> skip;
od
```

```
( a == b ) -> q = 1;
```

```
do
:: ( a == b ) -> break;
:: ( a == c ) -> skip;
:: else -> a++;
od
```

چند مثال دیگر

if **:: (a == 1) -> state++;** **:: (b == 1) -> state--;** **:: (c == 1) -> skip;** **fi**	**do** **:: (a == b) -> break;** **:: (a == c) -> skip;** **od**
(a == b) -> q = 1;	do :: (a == b) -> break; :: (a == c) -> skip; :: else -> a++; od

چند مثال دیگر

```
if
:: ( a == 1) -> state++;
:: ( b == 1) -> state--;
:: ( c == 1) -> skip;
fi
```

```
do
:: ( a == b) -> break;
:: ( a == c) -> skip;
od
```

```
( a == b ) -> q = 1;
```

```
do
:: ( a == b ) -> break;
:: ( a == c ) -> skip;
:: else -> a++;
od
```

چند مثال دیگر

```
if
:: ( a == 1) -> state++;
:: ( b == 1) -> state--;
:: ( c == 1) -> skip;
fi
```

```
do
:: ( a == b) -> break;
:: ( a == c) -> skip;
od
```

```
( a == b ) -> q = 1;
```

```
do
:: ( a == b ) -> break;
:: ( a == c ) -> skip;
:: else -> a++;
od
```

چند مثال دیگر

```
if
:: ( a == 1) -> state++;
:: ( b == 1) -> state--;
:: ( c == 1) -> skip;
fi
```

```
do
:: ( a == b) -> break;
:: ( a == c) -> skip;
od
```

```
( a == b ) -> q = 1;
```

```
do
:: ( a == b ) -> break;
:: ( a == c ) -> skip;
:: else -> a++;
od
```

استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
 	
active proctype Q( )
{
n = 1;
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	printf(“Thread P, n= %d\n”);
	
	}		

استفاده از *Channel* همگام ساز

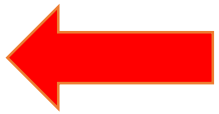
Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	
	printf(“Thread P, n= %d\n”);
	
	}		

استفاده از *Channel* همگام ساز

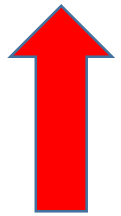
Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
printf(“Thread Q, n= %d\n”);
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	n = 2;
	printf(“Thread P, n= %d\n”);
	}		



استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	printf(“Thread P, n= %d\n”);
	
	}
		1 	

استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
printf(“Thread Q, n= %d\n”);
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	n = 2;
	printf(“Thread P, n= %d\n”);
	}
		1	

استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
printf(“Thread Q, n= %d\n”);
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	n = 2;
	printf(“Thread P, n= %d\n”);
	}
		1	

استفاده از *Channel* همگام ساز

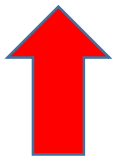
Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
printf(“Thread Q, n= %d\n”);
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	n = 2;
	printf(“Thread P, n= %d\n”);
	}
		1	Thread Q, n = 1

استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	
	printf(“Thread P, n= %d\n”);
	
	}
		1	Thread Q, n = 1



استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	
	printf(“Thread P, n= %d\n”);
	
	}
		1	Thread Q, n = 1

استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	
	printf(“Thread P, n= %d\n”);
	}
		2	Thread Q, n = 1

استفاده از *Channel* همگام ساز

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	
	printf(“Thread P, n= %d\n”);
	
	}
		2	Thread Q, n = 1
			Thread P, n = 2

استفاده از *Channel* همگام ساز – انحصار متقابل

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
printf(“Thread Q, n= %d\n”);
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	n = 2;
	printf(“Thread P, n= %d\n”);
	}
		2	Thread Q, n = 1
			Thread P, n = 2

استفاده از *Channel* همگام ساز – مثال ۲

Thread Q	Thread P	مقدار n	خروجی نهایی

chan sync_ch = [0] of {int};
active proctype Q( )
{
n = 1;
 	
printf(“Thread Q, n= %d\n”);
 	
sync_ch ! 0;
}
	active proctype P()
	{
	sync_ch ? 0;
	
	n = 2;
	
	printf(“Thread P, n= %d\n”);
	
	}		

بن بست هنگام استفاده از ***Channel*** همگام ساز

Thread P	Thread Q	Thread R

Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};
 	
active proctype P()
{
sync_P_R ? 0;
 	
skip;
 	
sync_P_Q ! 0;
 	
printf("End of P");
 	
}
	active proctype Q()
	{
	sync_P_Q ? 0;
	
	skip;
	
	sync_R_Q ! 0;
	
	printf("End of Q");
	}
		active proctype R()
		{
		sync_R_Q ? 0;
		
		skip;
		
		sync_P_R ! 0;
		
		printf("End of R");
		}

بن بست هنگام استفاده از ***Channel*** همگام ساز

Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};
 	
active proctype P()
{
sync_P_R ? 0;
 	
skip;
 	
sync_P_Q ! 0;
 	
printf("End of P");
 	
}
	active proctype Q()
	{
	sync_P_Q ? 0;
	
	skip;
	
	sync_R_Q ! 0;
	
	printf("End of Q");
	}
		active proctype R()
		{
		sync_R_Q ? 0;
		
		skip;
		
		sync_P_R ! 0;
		
		printf("End of R");
		}

بن بست هنگام استفاده از ***Channel*** همگام ساز

Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};
 	
active proctype P()
{
sync_P_R ? 0;
 
skip;
 	
sync_P_Q ! 0;
 	
printf("End of P");
 	
}
	active proctype Q()
	{
	sync_P_Q ? 0;
	
	skip;
	
	sync_R_Q ! 0;
	
	printf("End of Q");
	}
		active proctype R()
		{
		sync_R_Q ? 0;
		
		skip;
		
		sync_P_R ! 0;
		
		printf("End of R");
		}

بن بست هنگام استفاده از ***Channel*** همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از *Channel* همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از *Channel* همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از *Channel* همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از ***Channel*** همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از *Channel* همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از *Channel* همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

بن بست هنگام استفاده از *Channel* همگام ساز

```
Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};

active proctype P()
{
    sync_P_R ? 0;

    skip;

    sync_P_Q ! 0;

    printf("End of P");

}
```

```
active proctype Q()
{
    sync_P_Q ? 0;

    skip;

    sync_R_Q ! 0;

    printf("End of Q");
}
```

```
active proctype R()
{
    sync_R_Q ? 0;

    skip;

    sync_P_R ! 0;

    printf("End of R");
}
```

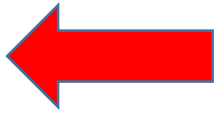
بن بست هنگام استفاده از *Channel* همگام ساز- مثال ۳

Chan sync_p_Q = [0] of {int};
Chan sync_p_R = [0] of {int};
Chan sync_R_Q = [0] of {int};
 	
active proctype P()
{
sync_P_R ? 0;
 	
skip;
 	
sync_P_Q ! 0;
 	
printf("End of P");
 	
}
	active proctype Q()
	{
	sync_P_Q ? 0;
	
	skip;
	
	sync_R_Q ! 0;
	
	printf("End of Q");
	}
		active proctype R()
		{
		sync_R_Q ? 0;
		
		skip;
		
		sync_P_R ! 0;
		
		printf("End of R");
		}

Verification

Verification

State-vector 48 byte, depth reached 0, errors: 1



1 states, stored

0 states, matched

1 transitions (= stored+matched)

0 atomic steps

hash conflicts: 0 (resolved)

Stats on memory usage (in Megabytes):

0.000 equivalent memory usage for states (stored*(State-vector + overhead))

0.292 actual memory usage for states

64.000 memory used for hash table (-w24)

0.343 memory used for DFS stack (-m10000)

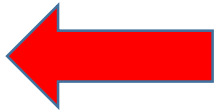
64.539 total actual memory usage

pan: elapsed time 0 seconds

To replay the error-trail, goto Simulate/Replay and select "Run"

Verification

State-vector 48 byte, depth reached 0, errors: 1



1 states, stored

0 states, matched

1 transitions (= stored+matched)

0 atomic steps

hash conflicts: 0 (resolved)

Stats on memory usage (in Megabytes):

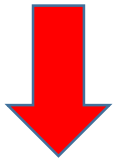
0.000 equivalent memory usage for states (stored*(State-vector + overhead))

0.292 actual memory usage for states

64.000 memory used for hash table (-w24)

0.343 memory used for DFS stack (-m10000)

64.539 total actual memory usage



pan: elapsed time 0 seconds

To replay the error-trail, goto Simulate/Replay and select "Run"

Error-trail

using statement merging

spin: trail ends after -4 steps

#processes: 3

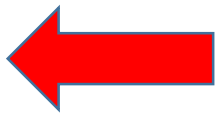
-4: proc 2 (R:1) sample3.pml:20 (state 1)

-4: proc 1 (Q:1) sample3.pml:13 (state 1)

-4: proc 0 (P:1) sample3.pml:7 (state 1)

3 processes created

Exit-Status 0



گرسنگی *Starvation*

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

Starvation گرسنگی

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

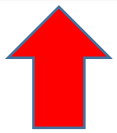
گرسنگی *Starvation*

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

گرسنگی *Starvation*

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 
active proctype
Producer()
{
do
:: nfull(feed) -> feed !
1;
od;
}
	active proctype
	Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1;
	feed?1; got[0] = 0;
	
	od;
	}
		active proctype
		Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] =
		1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

Starvation گرسنگی

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype
Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype
	Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1;
	feed?1; got[0] = 0;
	
	od;
	}
		active proctype
		Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] =
		1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

Starvation گرسنگی

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype
Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype
	Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype
		Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

Starvation گرسنگی

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype
Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype
	Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype
		Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

Starvation گرسنگی

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype
Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype
	Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype
		Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}
			 

گرسنگی *Starvation* – مثال ۴

Producer	Consumer 1	Consumer 2	liveness

chan feed = [1] of { byte };
bool got[2]=0;
 	
active proctype
Producer()
{
do
:: nfull(feed) -> feed ! 1;
od;
}
	active proctype
	Consumer()
	{
	byte x;
	do
	:: feed ? [1] -> got[0] = 1; feed?1; got[0] = 0;
	
	od;
	}
		active proctype
		Consumer()
		{
		byte x;
		do
		:: feed ? [1] -> got[1] = 1; feed?1; got[1] = 0;
		
		od;
		}
			ltl liveness {
			[] <>
			(got[1])
			}

مثال ۵

Server	Client 0	Client 1

chan request = [0] of { byte };
chan reply = [0] of { bool };
 	
active proctype Server() {
byte client;
end:do
:: request ? client -> printf("Client %d\n", client);
reply ! true
od
}
	active proctype Client0() {
	request ! 0;
	reply ? _
	}
		active proctype Client1()
		{
		request ! 1;
		reply ? _
		}

مثال ۵

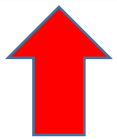
Server	Client 0	Client 1

chan request = [0] of { byte };
chan reply = [0] of { bool };
 	
active proctype Server() {
byte client;
end:do
:: request ? client -> printf("Client %d\n", client);
reply ! true
od
}
	active proctype Client0() {
	request ! 0;
	reply ? _
	}
		active proctype Client1()
		{
		request ! 1;
		reply ? _
		}

مثال ۵

Server	Client 0	Client 1

chan request = [0] of { byte };
chan reply = [0] of { bool };
 	
active proctype Server() {
byte client;
end:do
:: request ? client -> printf("Client %d\n", client);
reply ! true
od
}
	active proctype Client0() {
	request ! 0;
	reply ? _
	}
		active proctype Client1()
		{
		request ! 1;
		reply ? _
		}



مثال ۵

Server	Client 0	Client 1

chan request = [0] of { byte };
chan reply = [0] of { bool };
 	
active proctype Server() {
byte client;
end:do
:: request ? client -> printf("Client %d\n", client);
reply ! true
od
}
	active proctype Client0() {
	request ! 0;
	reply ? _
	}
		active proctype Client1()
		{
		request ! 1;
		reply ? _
		}

مثال ۵

Server	Client 0	Client 1

chan request = [0] of { byte };
chan reply = [0] of { bool };
 	
active proctype Server() {
byte client;
end:do
:: request ? client -> printf("Client %d\n", client);
reply ! true
od
}
	active proctype Client0() {
	request ! 0;
	reply ? _
	}
		active proctype Client1()
		{
		request ! 1;
		reply ? _
		}

مثال ۵

Server	Client 0	Client 1

chan request = [0] of { byte };
chan reply = [0] of { bool };
 	
active proctype Server() {
byte client;
end:do
:: request ? client -> printf("Client %d\n", client);
reply ! true
od
}
	active proctype Client0() {
	request ! 0;
	reply ? _
	}
		active proctype Client1()
		{
		request ! 1;
		reply ? _
		}

مثال ۶

Sender	Receiver	Ch		

mtype { red, yellow, green };
 	
chan ch = [0] of { mtype, byte, bool };
 	
active proctype Sender()
{
 	
ch ! red, 20, false;
 	
printf("Sent message\n")
 	
}
	active proctype Receiver() {
	
	mtype color;
	byte time;
	bool flag;
	
	ch ? color, time, flag;
	
	printf("Received message %e, %d, %d\n“, color, time,flag)
	
	}			

مثال ۶

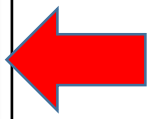
Sender	Receiver	Ch		

mtype { red, yellow, green };
 	
chan ch = [0] of { mtype, byte, bool };
 	
active proctype Sender()
{
 	
ch ! red, 20, false;
 	
printf("Sent message\n")
 	
}
	active proctype Receiver() {
	
	mtype color;
	byte time;
	bool flag;
	
	ch ? color, time, flag;
	
	printf("Received message %e, %d, %d\n“, color, time,flag)
	
	}			

مثال ۶

Sender	Receiver	Ch		

mtype { red, yellow, green };
 	
chan ch = [0] of { mtype, byte, bool };
 	
active proctype Sender()
{
 	
ch ! red, 20, false;
 	
printf("Sent message\n")
 	
}
	active proctype Receiver() {
	
	mtype color;
	byte time;
	bool flag;
	
	ch ? color, time, flag;
	
	printf("Received message %e, %d, %d\n“, color, time,flag)
	
	}			



مثال ۶

Sender	Receiver	Ch		

mtype { red, yellow, green };
 	
chan ch = [0] of { mtype, byte, bool };
 	
active proctype Sender()
{
ch ! red, 20, false;
 	
printf("Sent message\n")
 	
}
	active proctype Receiver() {
	
	mtype color;
	byte time;
	bool flag;
	
	ch ? color, time, flag;
	
	printf("Received message %e, %d, %d\n“, color, time,flag)
	
	}			

مثال ۶

Sender	Receiver	Ch		

mtype { red, yellow, green };
 	
chan ch = [0] of { mtype, byte, bool };
 	
active proctype Sender()
{
 	
ch ! red, 20, false;
 	
printf("Sent message\n")
 	
}
	active proctype Receiver() {
	
	mtype color;
	byte time;
	bool flag;
	
	ch ? color, time, flag;
	
	printf("Received message %e, %d, %d\n“, color, time,flag)
	
	}			

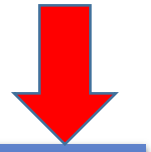
مثال ۷

Producer	Consumer	Monitor

mtype = {P , C};
mtype turn = P;
 	
int count = 0;
 	
active proctype producer(){
do
:: (turn == P) ->
count ++;
printf("Produce\n");
count --;
turn = C;
od
}
	active proctype Consumer(){
	do
	:: (turn == C) ->
	count ++;
	printf(“Consume\n");
	count --;
	turn = P;
	od
	}
		active proctype monitor(){
		
		assert(count == 0 || count == 1);
		
		}

مثال ۷

Producer	Consumer	Monitor

mtype = {P , C};
mtype turn = P;
 
int count = 0;
 	
active proctype producer(){
do
:: (turn == P) ->
count ++;
printf("Produce\n");
count --;
turn = C;
od
}
	active proctype Consumer(){
	do
	:: (turn == C) ->
	count ++;
	printf(“Consume\n");
	count --;
	turn = P;
	od
	}
		active proctype monitor(){
		
		assert(count == 0 || count == 1);
		
		}

مثال ۷

Producer	Consumer	Monitor

mtype = {P , C};
mtype turn = P;
 	
int count = 0;
 	
active proctype producer(){
do
:: (turn == P) ->
count ++;
printf("Produce\n");
count --;
turn = C;
od
}
	active proctype Consumer(){
	do
	:: (turn == C) ->
	count ++;
	printf(“Consume\n");
	count --;
	turn = P;
	od
	}
		active proctype monitor(){
		
		assert(count == 0 || count == 1);
		
		}

مثال ۷

Producer	Consumer	Monitor

mtype = {P , C};
mtype turn = P;
 	
int count = 0;
 	
active proctype producer(){
do
:: (turn == P) ->
count ++;
printf("Produce\n");
count --;
turn = C;
od
}
	active proctype Consumer(){
	do
	:: (turn == C) ->
	count ++;
	printf(“Consume\n");
	count --;
	turn = P;
	od
	}
		active proctype monitor(){
		
		assert(count == 0 || count == 1);
		
		}

مثال ۷

Producer

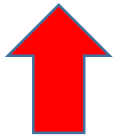
```
mtype = {P , C};
mtype turn = P;

int count  = 0;

active proctype producer(){
        do
         ::  ( turn == P ) ->
                count ++;
                printf( "Produce\n");
                count --;
                turn = C;
        od
}
```

Consumer

```
active proctype Consumer(){
        do
         ::  ( turn == C ) ->
                count ++;
                printf( “Consume\n");
                count --;
                turn = P;
        od
}
```

Monitor

```
active proctype monitor(){
        assert(count == 0 || count == 1);
}
```



مثال ٧

Producer	Consumer	Monitor

mtype = {P , C};
mtype turn = P;
 	
int count = 0;
 	
active proctype producer(){
do
:: (turn == P) ->
count ++;
printf("Produce\n");
count --;
turn = C;
od
}
	active proctype Consumer(){
	do
	:: (turn == C) ->
	count ++;
	printf(“Consume\n");
	count --;
	turn = P;
	od
	}
		active proctype monitor(){
		
		assert(count == 0 || count == 1);
		
		}