8 лекция

Крипке құрылымы әсер ететін жүйелердің моделі ретінде

Орындаған: Жарымбетова Т.
Имаш А.
Өзбек Қ.

Параллельді процестерді талдау

Параллельді процестер түсіну үшін өте қиын

x:=1; x++ || x :=3
екі детерминирленген процесс; олардың параллель орындалуы детерминирленген емес (қанша нәтиже? қандай?) (2, 3, 4)

Әрекет ететін жүйелердің жұмыс істеуінің жалпы сипаттамасы:

`loop`

алынған мәндерге негізделген кіріс сенсорларынан мәндерді оқу аффективті іске қосады

`forever`

- Реакция беретін жүйелер-бұл бір-бірімен және сыртқы ортамен өзара әрекеттесетін үдерістердің параллель жұмыс істейтін, әдетте аяқталмаған жиынтықтары

Олардың мінез-құлқын талдау қалай орындалады (аталар, нөсер, аштық және т. б.)?

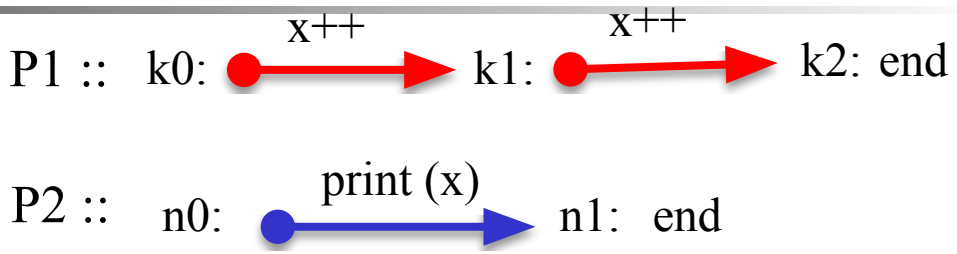
Модель НЕДЕТЕРМИНИЗМДІ сипаттауға мүмкіндік беруі тиіс! (Функционалдық бағдарламалардан айырмашылығы, недетерминизм – параллель процестердің маңызды құрамдас бөлігі!!)

Интерливинг әсері

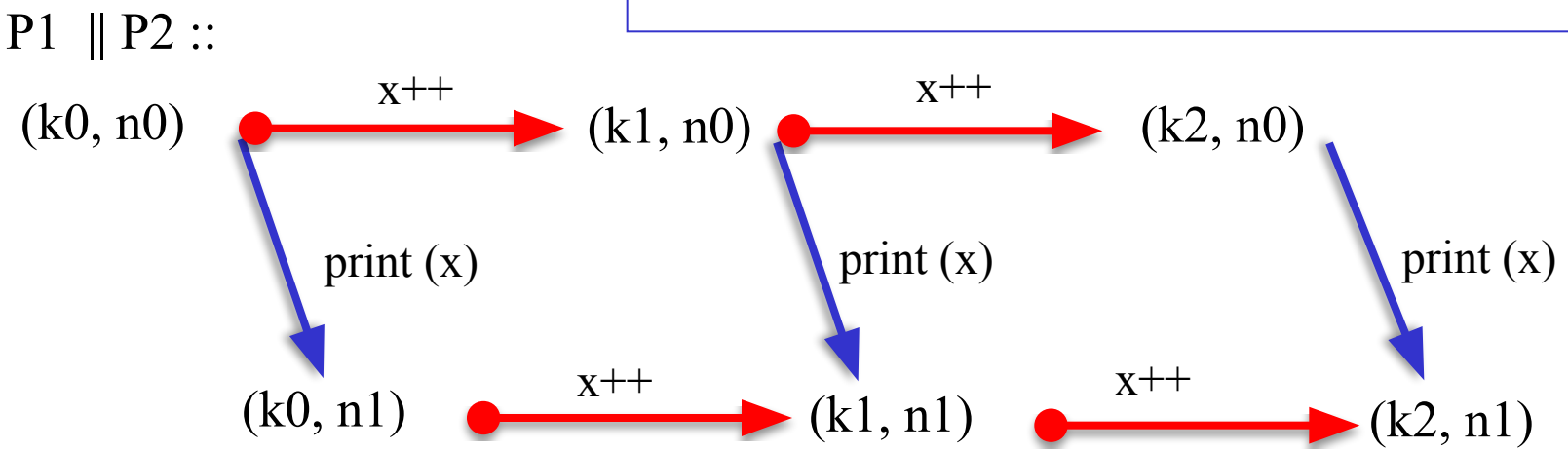
x=0

P1::	P2::
k0: x++;	n0: print(x);
k1: x++;	n1: end
k2: end	

Не басылады?



Параллель бағдарламадағы Интерливинг-бұл әр үдерістің операторлары орындайтын тәртіптің өзгермейтін таңдауы



Нәтижесінде біз 0, 1 немесе 2 санын шығара аламыз

Параллельді процестер: өткелдердің түйіршіктілігі

(x=1; y=2) – бастапқы жағдайы

Аяқтағаннан кейін x және y мәндері?

Process A::	Process B::
... k: x:=x+y ...	... m: y:=y+x ...

1 нұсқасы. Егер қосу процессордың бір атомдық операциясы ретінде іске асырылса, онда интерливинг екі ықтимал нәтиже береді: (x=3; y=5), (x=4; y = 3)

k_0: LD R_A, x k_1: ADD R_A, y k_2: ST R_A, x	m_0: LD R_B, y m_1: ADD R_B, x m_2: ST R_B, y

2 нұсқасы. Егер қосу процессордың үш атомдық операциялары ретінде іске асырылса, онда интерливинг үш мүмкін нәтиже береді: (x=3; y=5), (x = 4; y=3), (x=3; y=3)

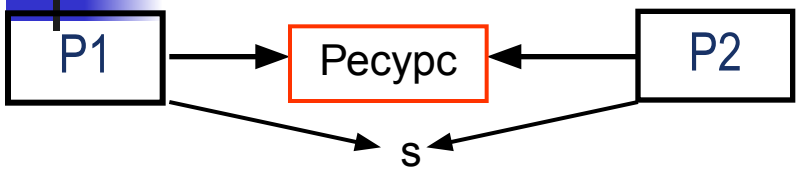
Тексерілген сипат болсын R: EF(x=y)

Егер іске асыруда-1 нұсқа, ал модельде 2 нұсқа пайдаланылса, онда біз жүйеде R қасиеті орындалатынын қате аламыз

Егер 2 нұсқаны іске асыруда, ал модельде 1 нұсқа пайдаланылса, онда біз жүйеде R қасиеті орындалмайды деп қате аламыз

Модельге түрлендіру іске асыруды ескеруі тиіс

Өзара ерекшелік мәселесі



s=1 – ресурс бос

P1::

Сыни емес интервал

Кіру кезінде тексеру

Сыни интервал

Шығу туралы хабарлама

Сыни емес интервал

...

Pn::

Сыни емес интервал

Кіру кезінде тексеру

Сыни интервал

Шығу туралы хабарлама

Сыни емес интервал

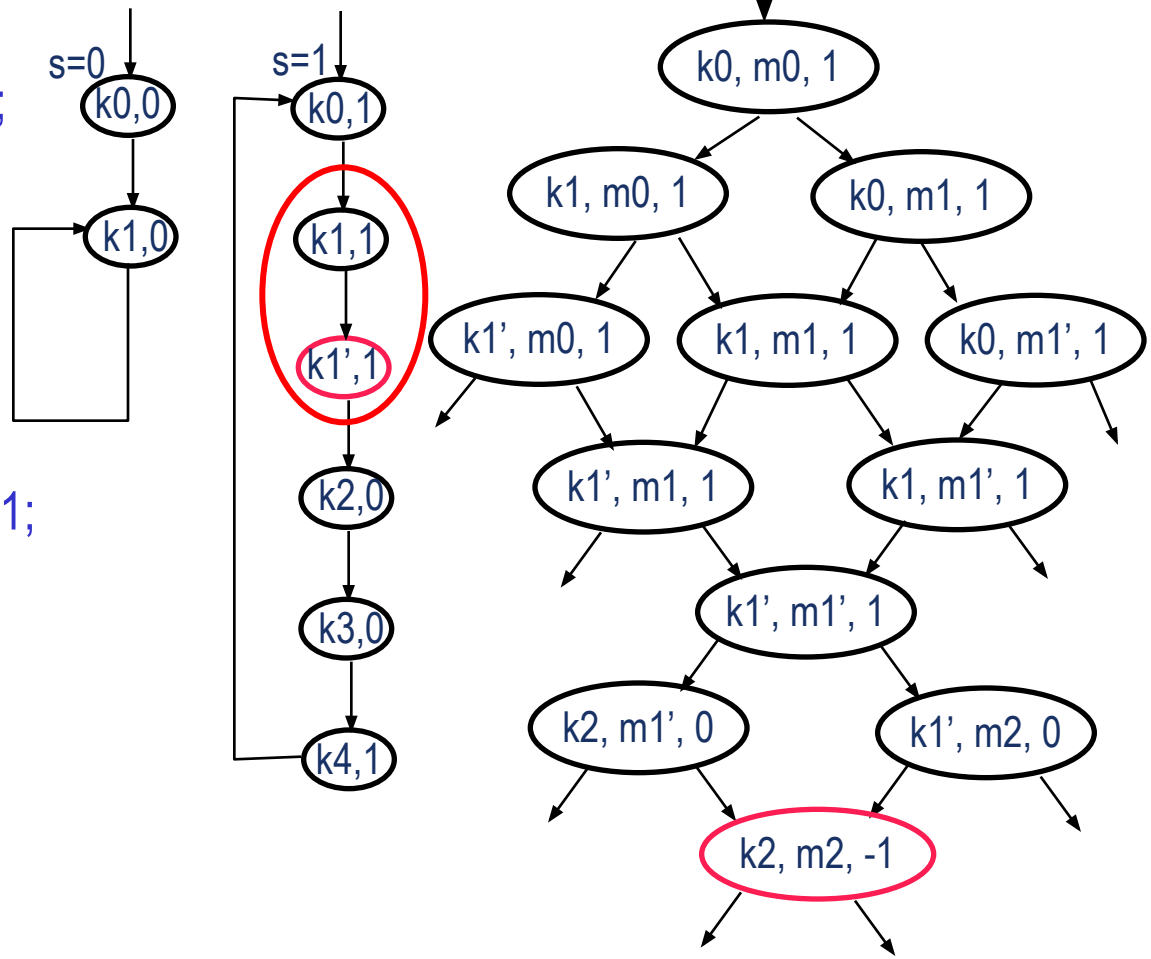
```
P1::
    k0: noncritical1;
    k1: if s>0 then s:=s-1else goto k1;
    k2: critical1;
    k3: s:=s+1;
    k4: goto k0;

P2::
    m0: noncritical2;
    m1: if s>0 then s:=s-1else goto m1;
    m2: critical2;
    m3: s:=s+1
    m4: goto m0;
```

Өзара алып тастау проблемасын шешудің бірі

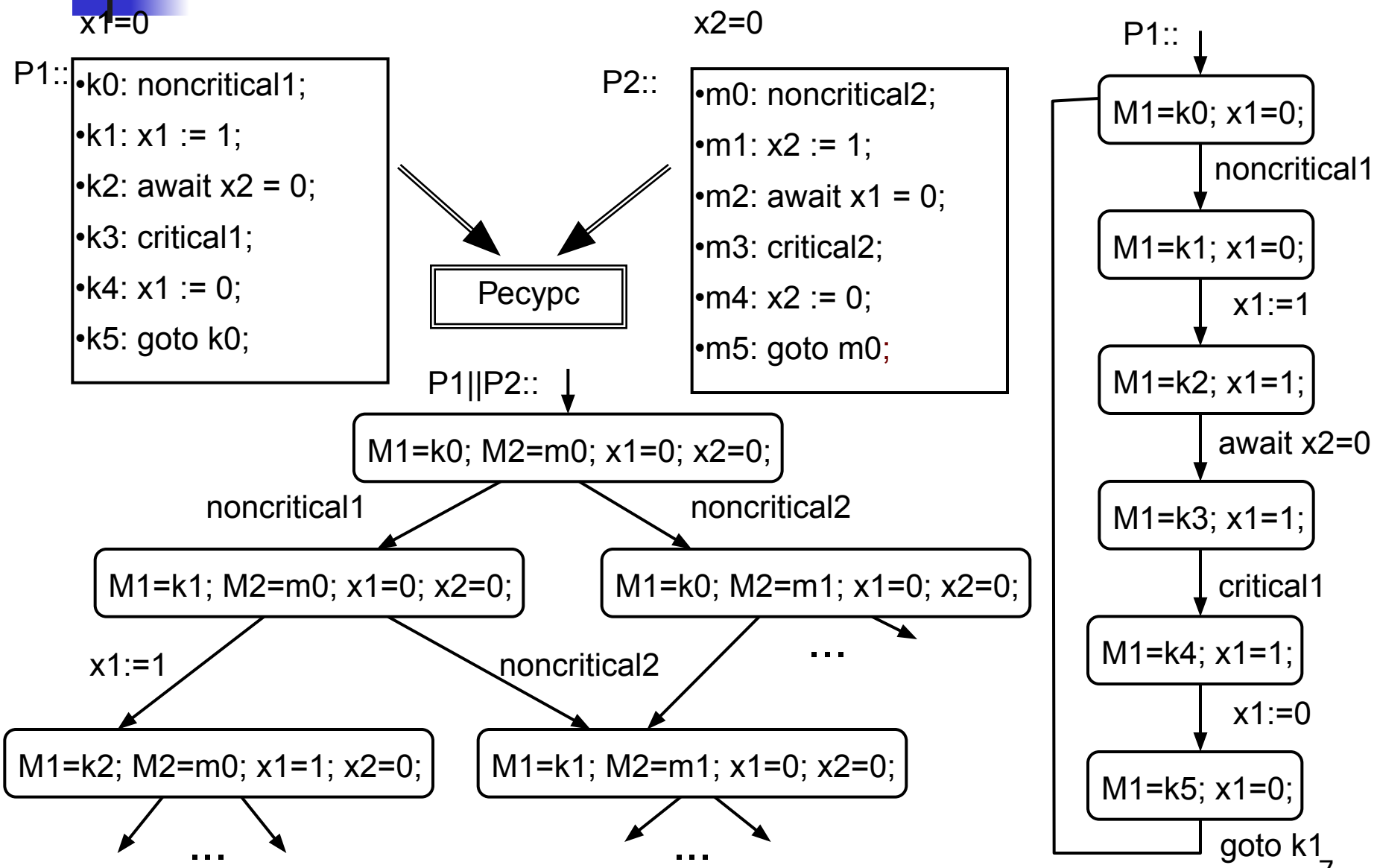
P1:: k0: noncritical1;
k1: if s>0 then s:=s-1else goto k1;
k2: critical1;
k3: s:=s+1;
k4: goto k0;

P2:: m0: noncritical2;
m1: if s>0 then s:=s-1else goto m1;
m2: critical2;
m3: s:=s+1
m4: goto m0;



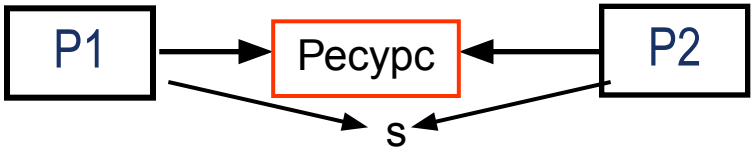
Шешім дұрыс емес: оператор *m: if s>0 then s:=s-1else goto m* екі операторлан тұрады.

Өзара ерекшелік проблемасының басқа шешімі



Семафор-Дейкстраның данышпан идеясы

Егер оператор *m: if s>0 then s:=s-1* атомдық қылса, онда шешім дұрыс болады. Бірақ бұл да жалпы ресурсқа қол жеткізу! Яғни тұйық шеңбер!!



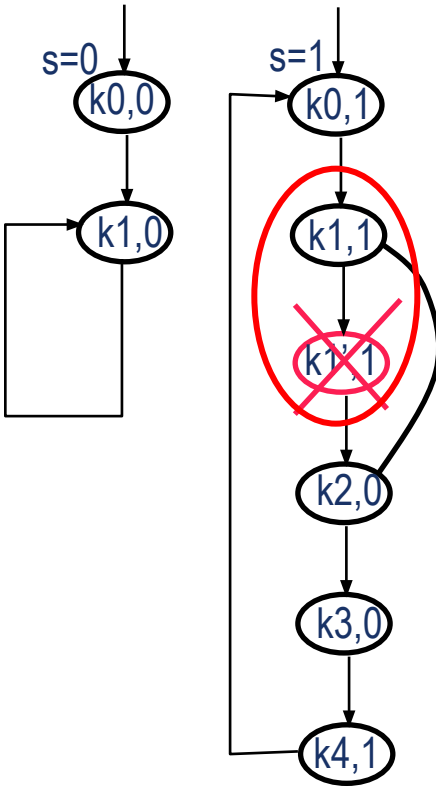
Бізге дереу қажет емес, атомдық операцияны орындау қажет

Жалпы ресурсқа қол жеткізудің стандартты операциясын – жалпы айнымалыны атомдық, бөлінбейтін (бірақ жедел емес) орындауды қамтамасыз етеді.

P(s) ≡ *m: if s>0 then s:=s-1* атомдық орындалады

V(s) ≡ *s:=s+1* атомдық орындалады

```
P1::
  k0: noncritical1;
  k1: if s>0 then s:=s-1;
  k2: critical1;
  k3: s:=s+1;
  k4: goto k0;
```



Семафор көмегімен өзара ерекшелік

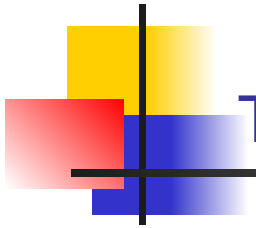
```
P1::
    k0: noncritical1;
    k1: P(s);
    k2: critical1;
    k3: V(s) ;
    k4: goto k0;
```

Параллель процестерді орындау нәтижелерінің болжамсыздығы

Параллельді бағдарламалардағы қателіктердің көпшілігі-параллельді үдерістер операцияларының күтпеген жабылуына байланысты. Мысалы, DeepSpace1 табылған қателер

```
byte state = 1;
proctype A( ) {
byte tmp; (state==1) -> tmp = state; tmp = tmp+1;
state = tmp
}
proctype B( ) {
byte tmp; (state==1) -> tmp = state; tmp = tmp-1;
state = tmp
}
init {
run A(); run B()
}
```

Егер қандай да бір процесс басқа процесс state==1 тексеруді орындағанға дейін аяқталса, онда "кешіккен" процесс біржола оқшауланады. Егер шартты тексеру басқа процесс аяқталғанша процестермен орындалса, онда екі процесс аяқталады, бірақ айнымалы state мәні күтпеген болады – ол кез келген мәнді қабылдай алады: 0, 1 немесе 2



Тапсырма үлгісі

- Циклде үш параллельді процесс 10 рет жалпы бөлінетін айнымалы x, алдымен 0 тең көбейтеді.
- Сұрақтар :
  Каково будет значение x после окончания процесса init?
  Каковы возможные максимальное и минимальное значения x?

```
proctype Pi ( ) {
  for k=1, ..., 10 do
      x:=x+1
  od }
init {
   int x := 0;
   run P1();
   run P2();
   run P3()
  }
```

x:=x+1 –
три атомарные операции:

LOAD (x, Ri);
INC(Ri);
STORE (Ri, x)

Как в системах верификации?

- В реальных системах может быть любая степень грануляции – от микрокоманд до групп операторов, защищенных семафорами или другими средствами синхронизации
- Разработчик модели сам ответственен за то, будет ли модель адекватно отражать поведение системы (верификация мощна настолько, насколько адекватной является построенная для анализа модель)
- Неделимой единицей является обычно оператор входного языка системы верификации. Он целиком либо выполняется, либо нет.
- Существует возможность объединять группы операторов в атомарные последовательности. Атомарные последовательности операторов должны быть явно объявлены, например, специальные скобки ⟨ ... ⟩
- В Promela: атомарными являются
  - любой отдельный оператор x = f(x, y), например m = (a>b → a : b)
  - *atomic* { ... } – группа операторов, заключенная в скобки с *atomic*

Пример. Выполнение параллельных процессов с атомарной цепочкой операторов

```
byte state = 1;

proctype A( ) {
atomic { (state==1) -> state = state+1 }
}

proctype B( ) {
atomic { (state==1) -> state = state-1 }
}

init {
 run A( ); run B( )
}
```

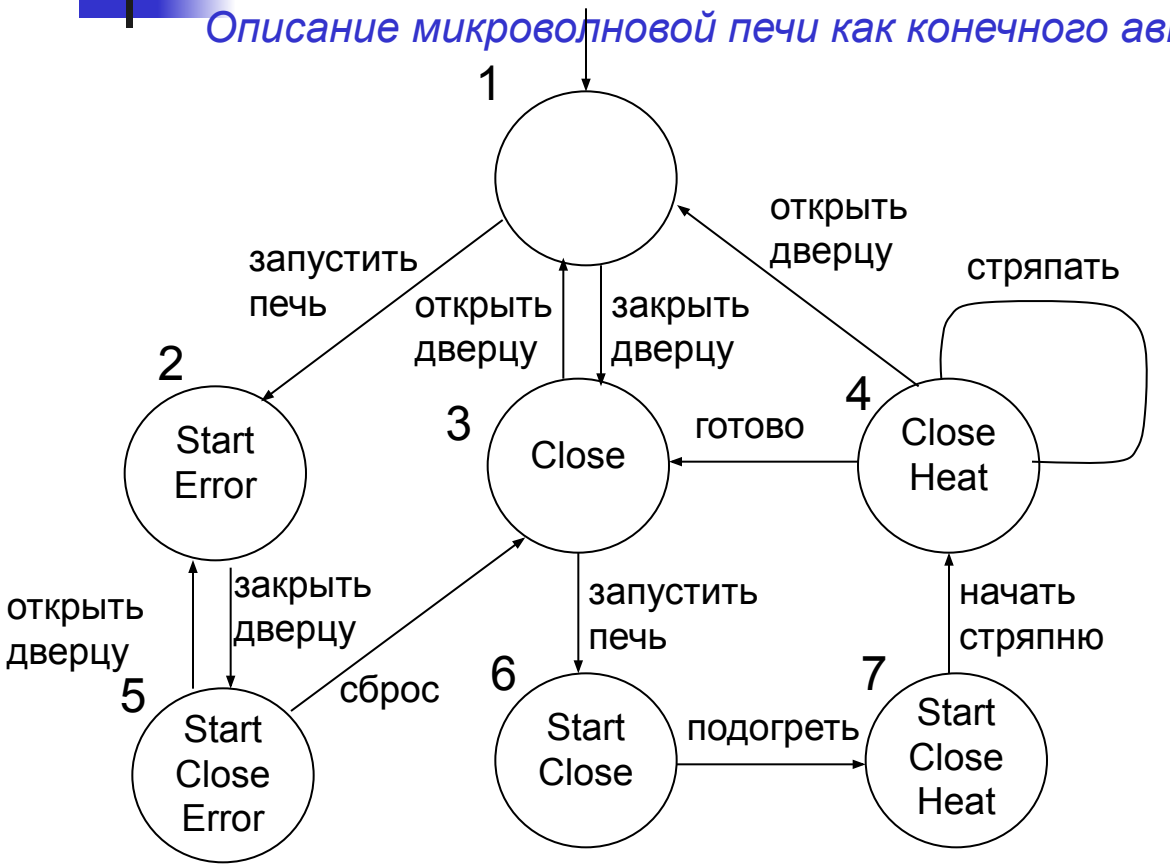
Как будет себя вести параллельная программа???

Значение переменной state станет равным 2 или 0, в зависимости от того, какой из процессов, A или B, первым выполнит свою атомарную операцию. Другой процесс будет при этом заблокирован навсегда

- Как преобразовать программу в структуру Крипке?

Системы, специфицированные в виде КА

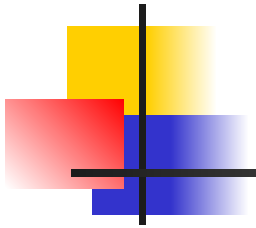
Описание микроволновой печи как конечного автомата



Структура Крипке получается из КА отбрасыванием действий на переходах – неважно, какие последовательности входов привели к ошибке

Можно проверить систему относительно любой спецификации, основанной на атомарных предикатах, например: **AG**(¬*Close*⇒¬*Heat*)

“*В любом состоянии всегда при открытой дверце нагревание не происходит*”



Спасибо за внимание