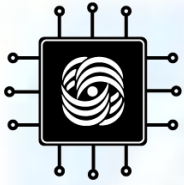


НАДЁЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лекция 3:

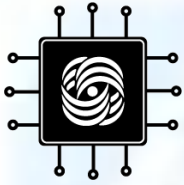
Системы переходов (LTS). Корректность и адекватность LTS модели

ВМК МГУ им. М.В. Ломоносова,
Кафедра АСВК, Лаборатория Вычислительных Комплексов
к.ф.-м.н., доцент Волканов Д.Ю.



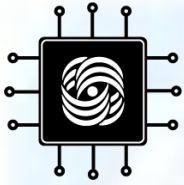
План лекции

- Система понятий, используемых в курсе
- Размеченные системы переходов (LTS)
- Недетерминизм систем переходов
- Пути, вычисления, трассы
- Свойства линейного времени
- Корректность модели
- Адекватность модели



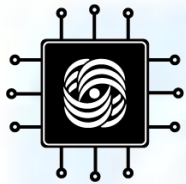
Итоги прошлой лекции

- Свойства проверяются на состояниях и их последовательностях
- Чтобы перебирать меньше состояний, исследуется не исходная система, а её модель
- Модель должна быть простой, корректной и адекватной
- В этом случае из правильности модели следует правильность программы

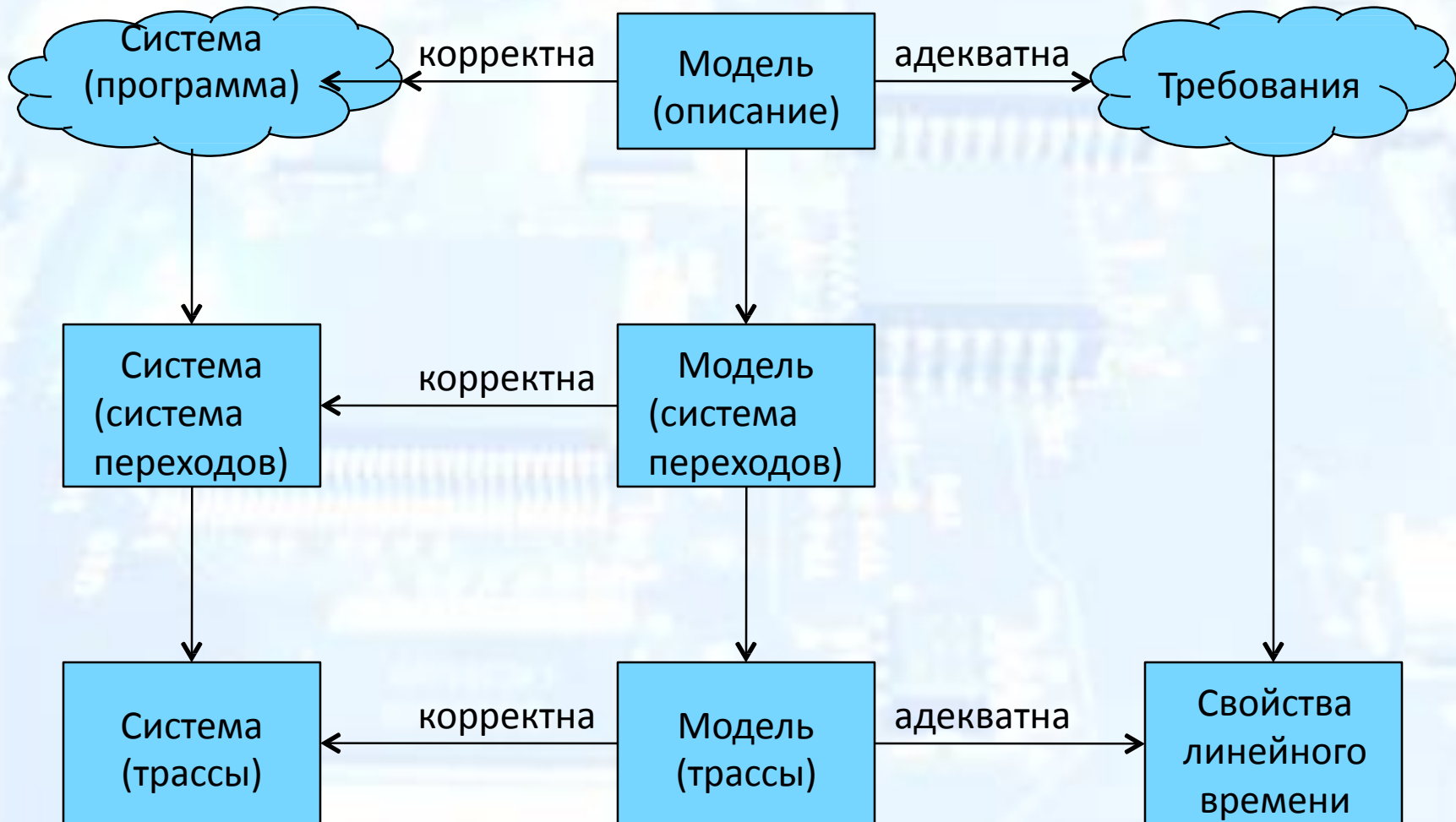


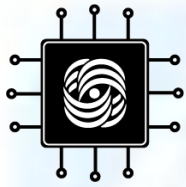
Строго говоря...

Если мы хотим доказать, что после проверки правильности на модели в программе нет ошибок, то все эти понятия стоит сформулировать более строго

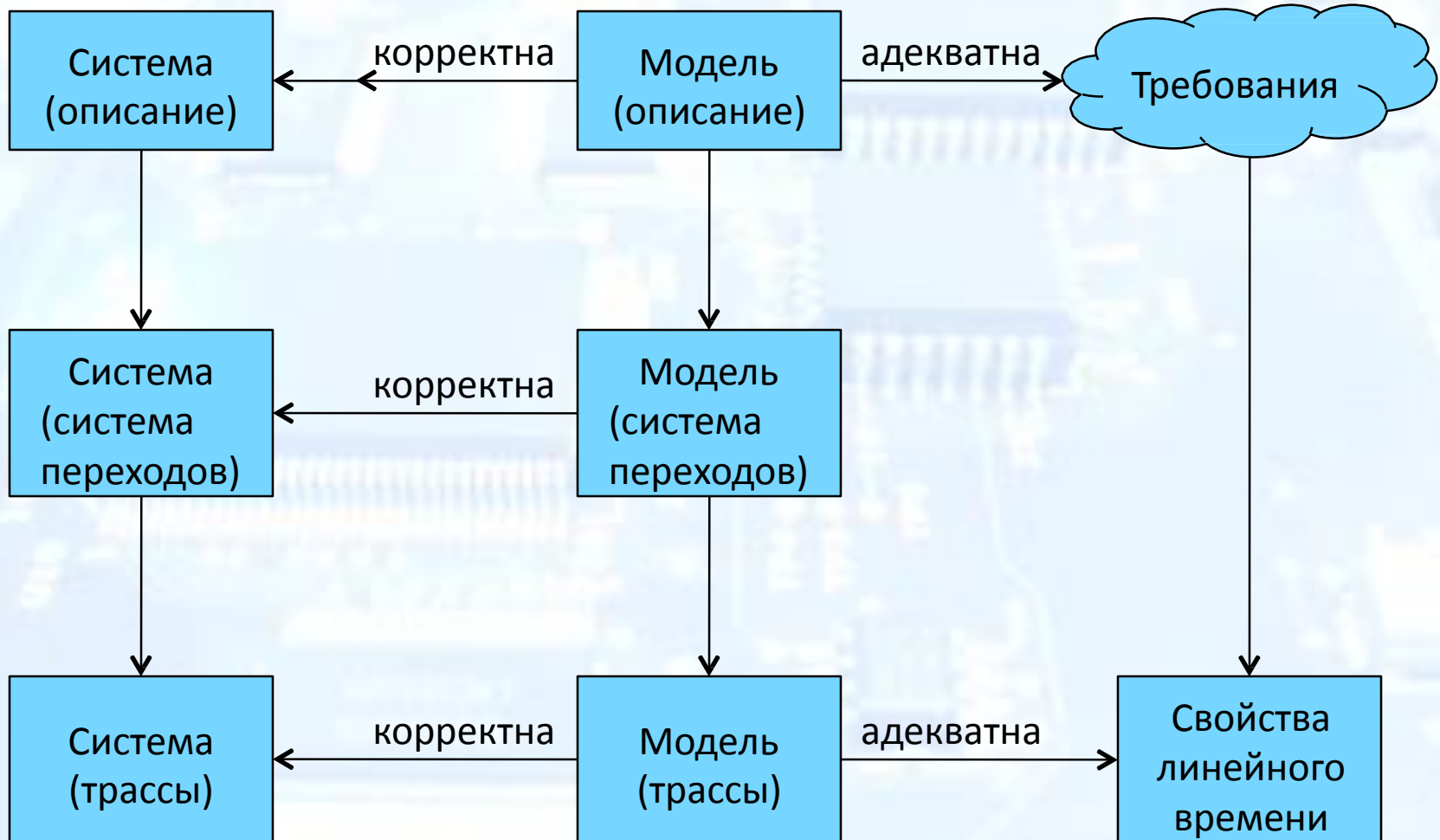


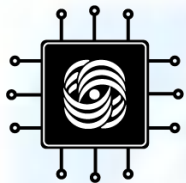
Строго говоря, всё должно быть так:





Строго говоря, всё должно быть так:

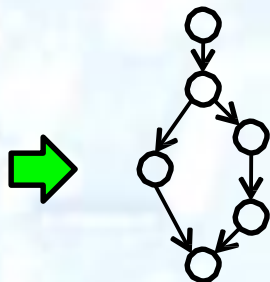




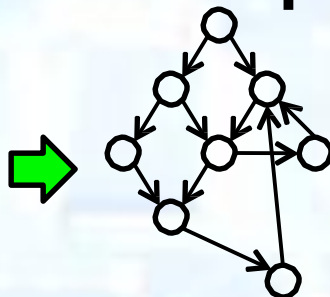
Различные представления программы

```
int main  
( ) {  
  printf(  
  }  
}
```

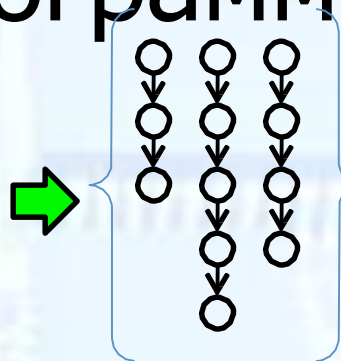
Исходный код программы



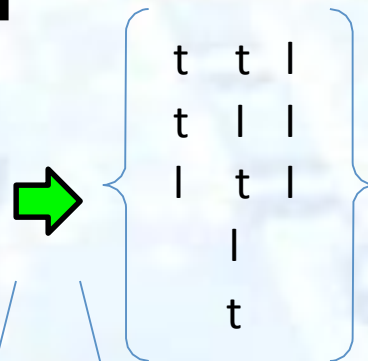
Граф программы (ACFG)



Размеченная система переходов



Множество вычислений



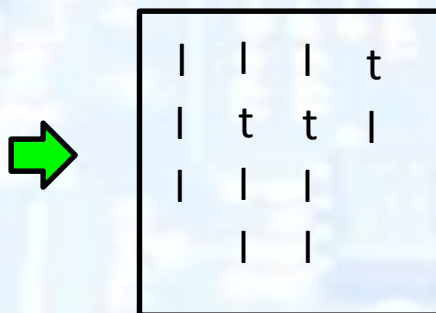
Множество трасс

Взлетает,
не падает,
приземляется

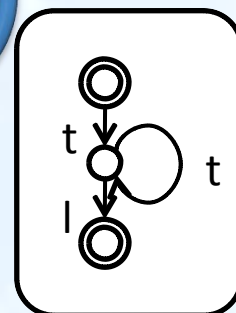
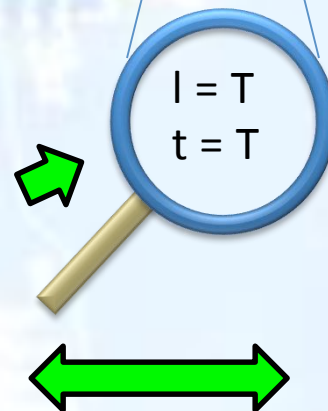
Требования

$\square ($
TAKEOFF $\rightarrow$
(!FALL)
 $\cup$
(LANDED) $)$

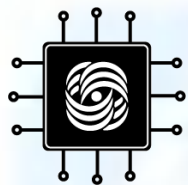
Спецификация
(линейного времени)



Допустимые
последовательности
атомарных высказываний



Язык
допустимых трасс

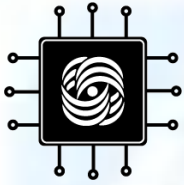


Размеченные системы переходов (LTS)

$$TS = \langle S, Act, \xrightarrow{a}, I, AP, L \rangle$$

- S – множество состояний,
- Act – множество действий, τ – невидимое действие,
- $\xrightarrow{a} \subseteq S \times Act \times S$ – **тотальное** отношение переходов,
- $I \subseteq S$ – **множество** начальных состояний,
- AP – множество атомарных высказываний,
- $L : S \rightarrow 2^{AP}$ – функция разметки.

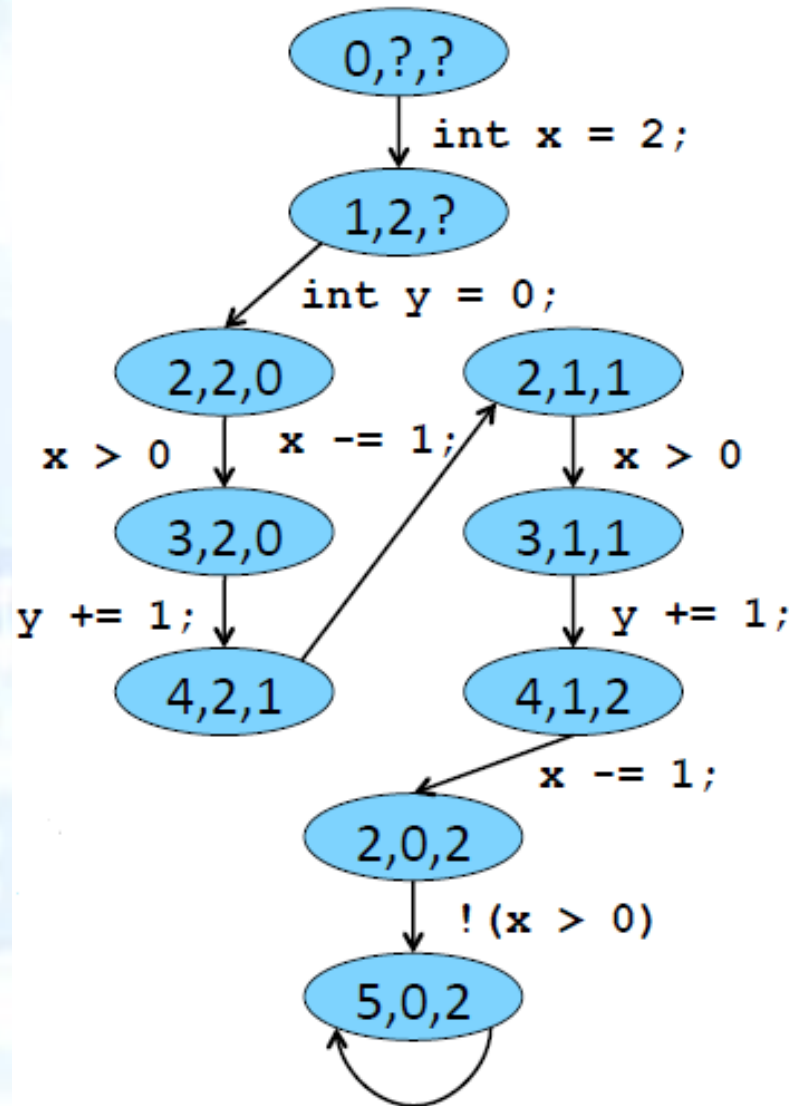
$$\left(\begin{array}{l} S \text{ – конечное или счётное множество, } Act \text{ – конечное} \\ \text{Нотация: } \langle s, a_0, s' \rangle \in \xrightarrow{a} \equiv s \xrightarrow{a_0} s' \end{array} \right)$$

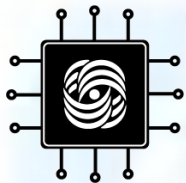


Пример LTS

→ 0: `int x = 2;`
1: `int y = 0;`
2: `while (x > 0)`
 {
3: `y += 1;`
4: `x -= 1;`
 }
5:

[Состояние: (счётчик, x, y)]

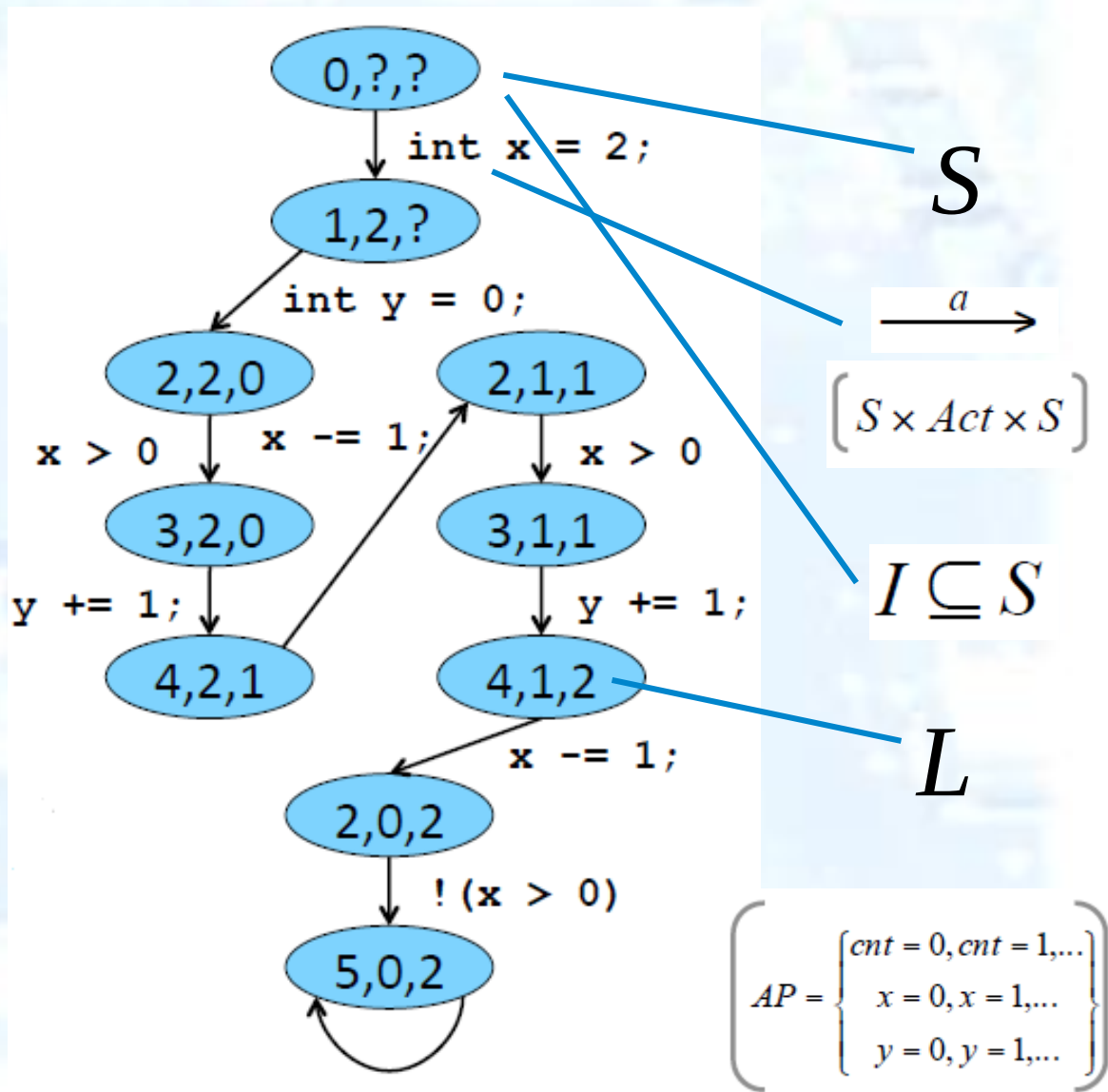


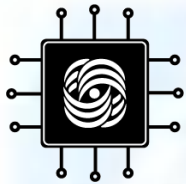


Пример LTS

0: `int x = 2;`
 1: `int y = 0;`
 2: `while (x > 0)`
 {
 3: `y += 1;`
 4: `x -= 1;`
 }
 5:

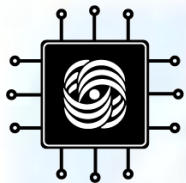
[Состояние: (счётчик, x, y)]





Что включать в состояние

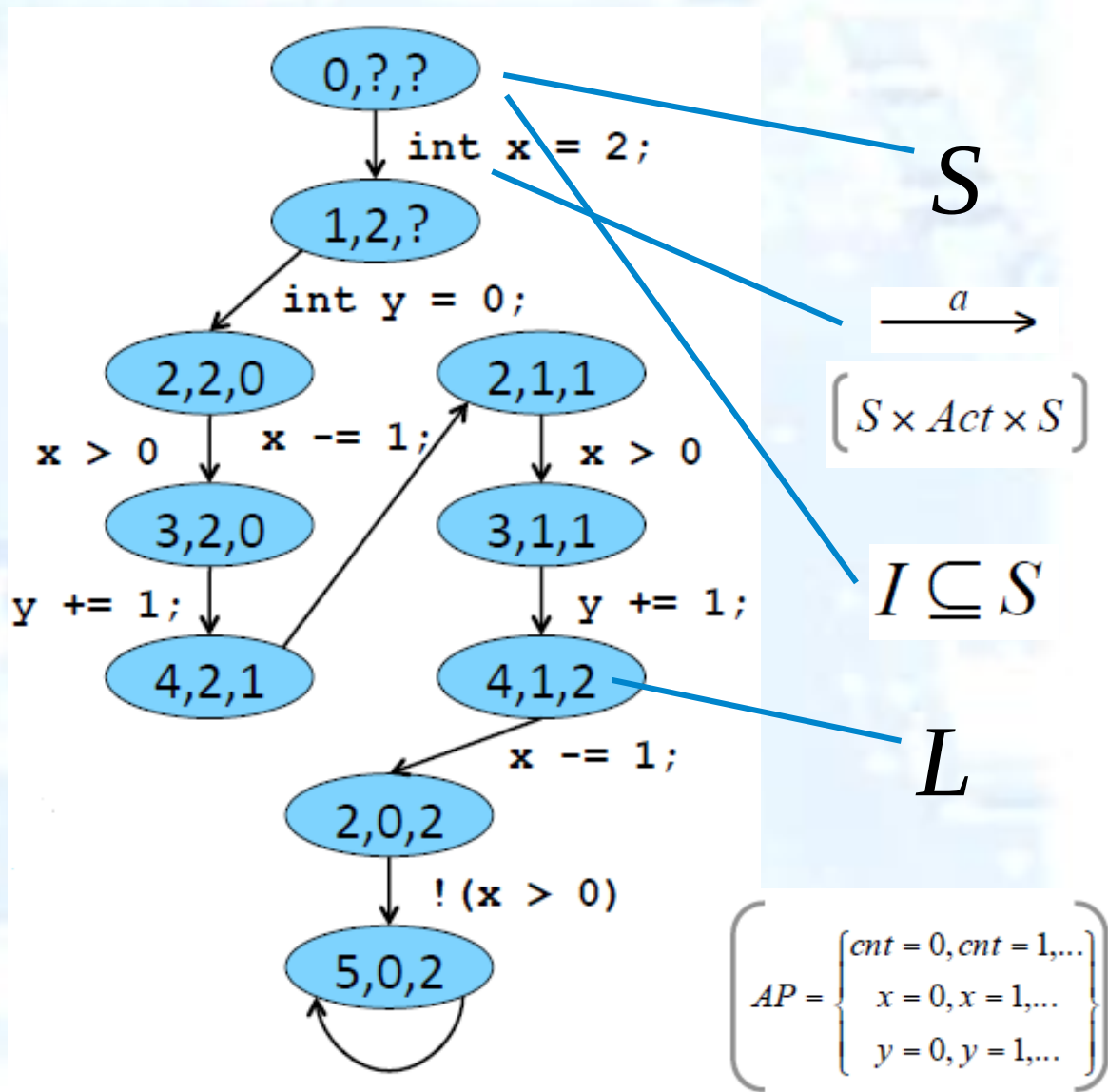
- Набор атомарных высказываний **AP** определяется свойствами, которые нужно проверить
- Изменение состояния связано с изменением выполнимости хотя бы одного атомарного высказывания
- Исходя из этого мы определяем, что включать в состояние программы
- Главное – не «потерять» ни одного изменения атомарных состояний

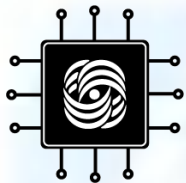


Пример LTS

0: `int x = 2;`
 1: `int y = 0;`
 2: `while (x > 0)`
 {
 3: `y += 1;`
 4: `x -= 1;`
 }
 5:

[Состояние: (счётчик, x, y)]

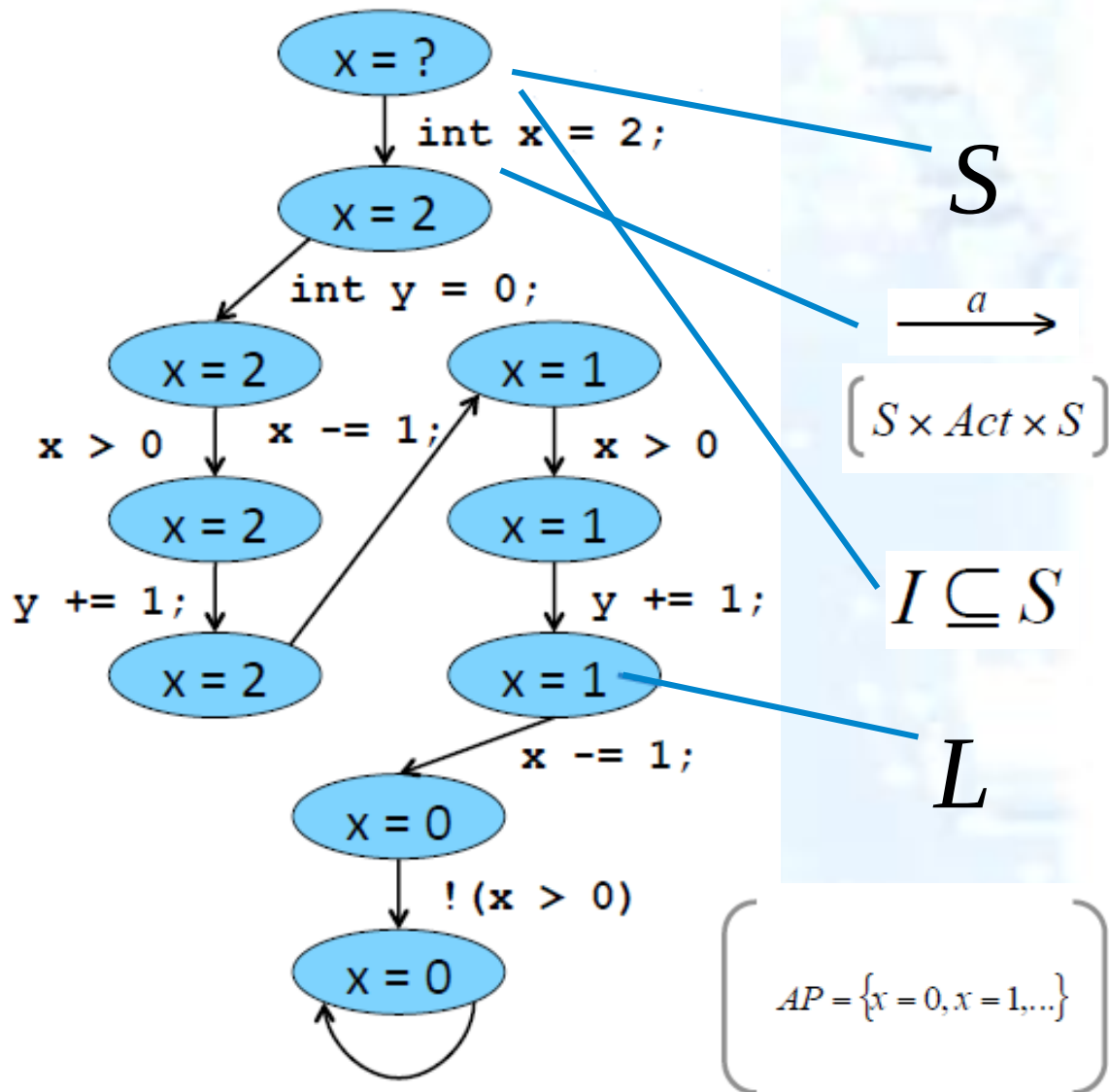


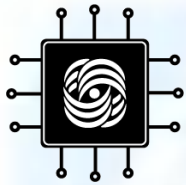


Пример LTS

→ 0: `int x = 2;`
1: `int y = 0;`
2: `while (x > 0)`
 {
3: `y += 1;`
4: `x -= 1;`
 }
5:

[Состояние: (счётчик, x, y)]

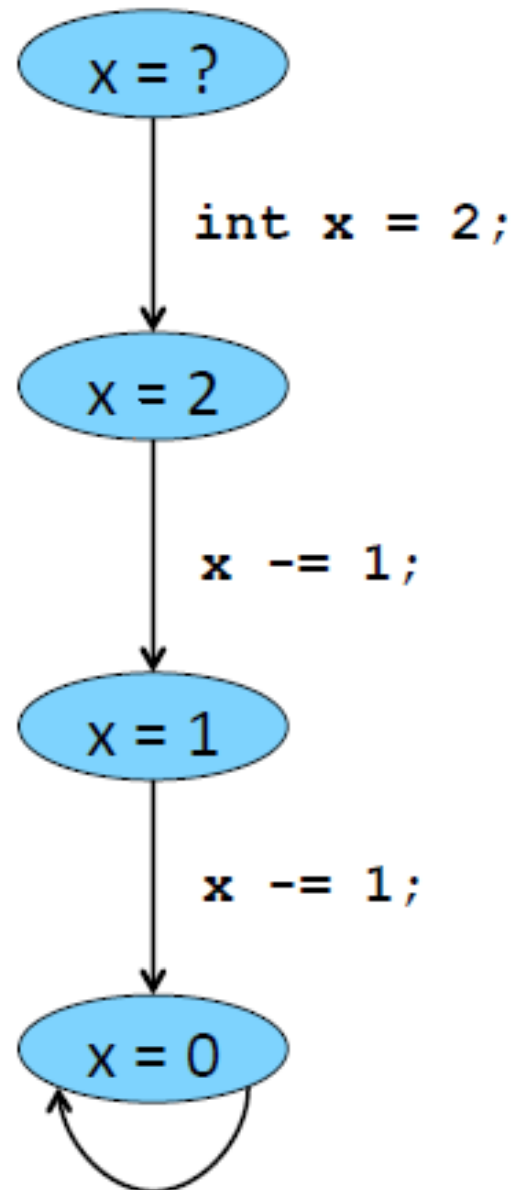




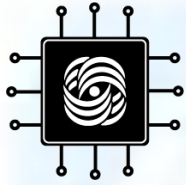
Пример LTS

→ 0: `int x = 2;`
1: `int y = 0;`
2: `while (x > 0)`
 {
3: `y += 1;`
4: `x -= 1;`
 }
5:

[Состояние: (счётчик, x, y)]



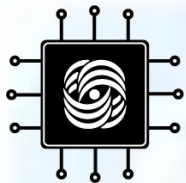
[$AP = \{x = 0, x = 1, \dots\}$]



Что включать в состояние?

В дальнейшем мы будем рассматривать «универсальное» определение состояния, достаточное для проверки свойств линейного времени и локализации их нарушения в описании программы

Совокупность значения счётчиков управления последовательных процессов и переменных программы



Пример LTS

```
int p;
```

```
Process Prod() {
```

```
    while(1)
```

```
1.1:    if(p < 2)
```

```
1.2:    p += 1;
```

```
}
```

```
process Cons() {
```

```
    while(1)
```

```
2.1:    if(p > 0)
```

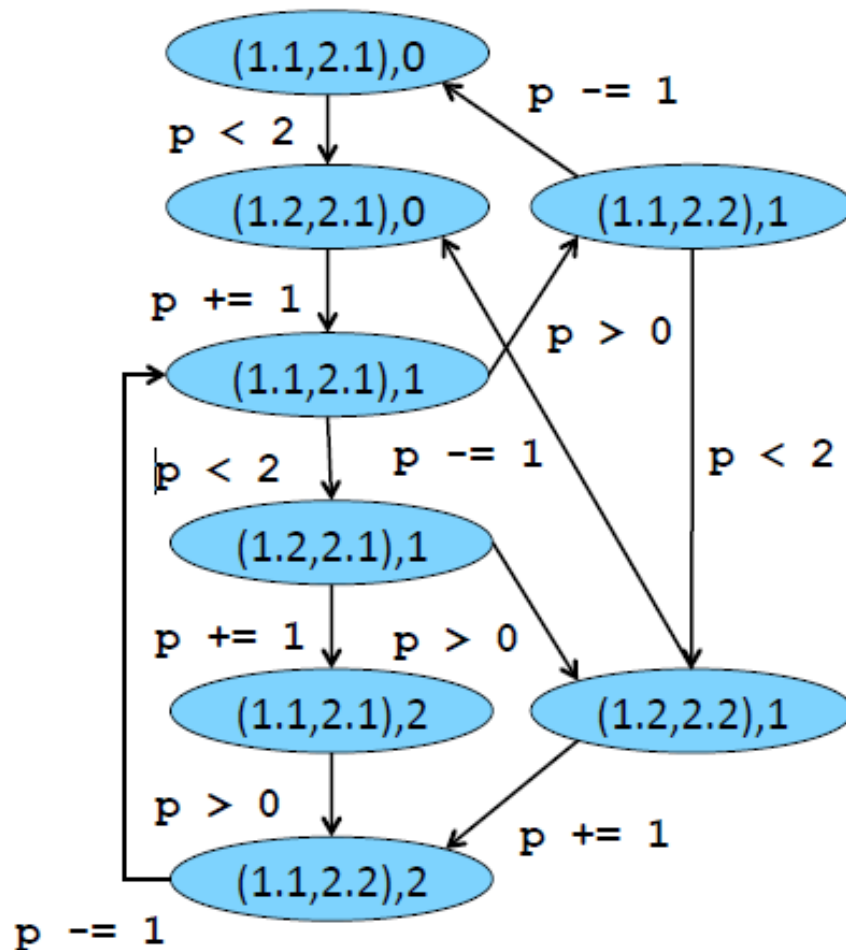
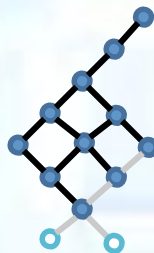
```
2.2:    p -= 1;
```

```
}
```

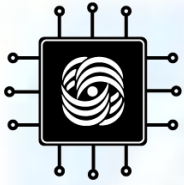
Состояние: (счётчик Prod,
счётчик Cons, p)

Часть состояний не показана
(оператор беск. цикла,
стартовые состояния)

Prod ↔ Cons

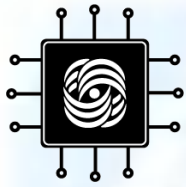


Бесконечное количество вычислений, однако
размеченная система переходов конечна¹⁶



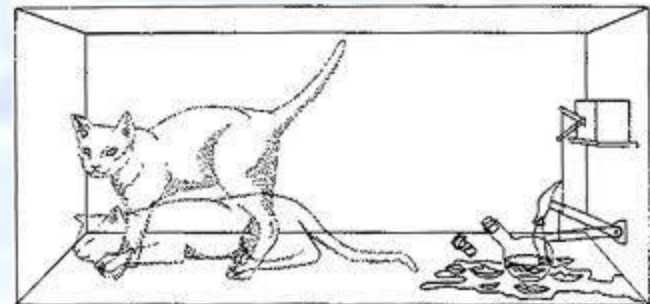
Недетерминизм

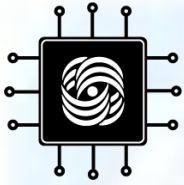
- В ряде ситуаций шаг может сделать любой из двух процессов, порядок действий не определён
- **Недетерминизм = неопределённость**
- При построении LTS рассматриваются все возможные варианты последовательности действий



Недетерминизм

- Вообще-то в природе довольно мало недетерминированных процессов
- Да и те считаются недетерминированными, поскольку физические законы, по которым выбирается действие, нам не известны
- Недетерминизм появляется, как только мы не знаем причин выбора действия или считаем их несущественными

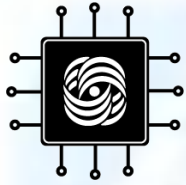




Недетерминизм

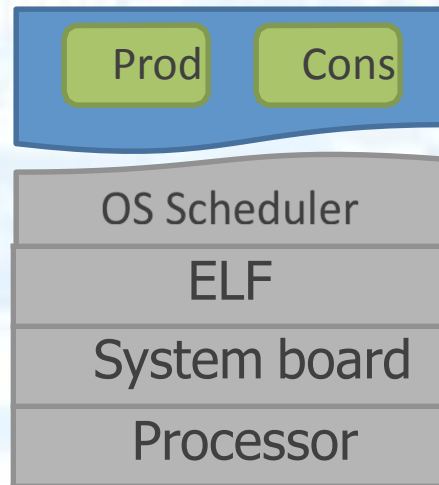
- Порядок выполнения Prod и Cond на конкретном компьютере детерминирован и определяется алгоритмом диспетчера операционной системы и состоянием ресурсов
- Для проверки правильности программы мы решили абстрагироваться от всего, кроме описания двух процессов
- Итог – недетерминизм очерёдности выполнения процессов

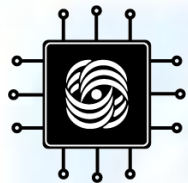
（ Подробнее про разные виды недетерминизма –
ниже ）



Недетерминизм – это фича!

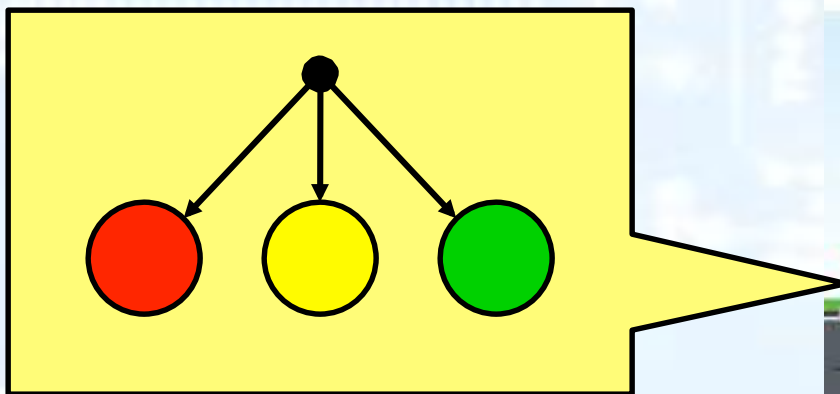
- Используется для:
 - моделирования параллельного выполнения процессов в режиме чередования (интерливинга)
 - позволяет абстрагироваться от алгоритма диспетчеризации и скорости выполнения процессов

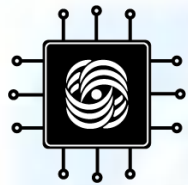




Недетерминизм – это фишка!

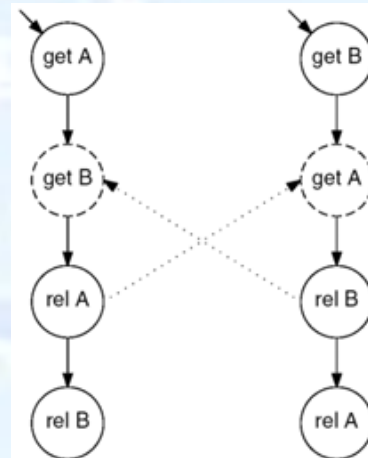
- Используется для:
 - моделирования прототипа системы
 - не ограничивает будущую реализацию заданным порядком выполнения операторов или конкретными входными данными

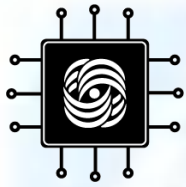




Недетерминизм – это фишка!

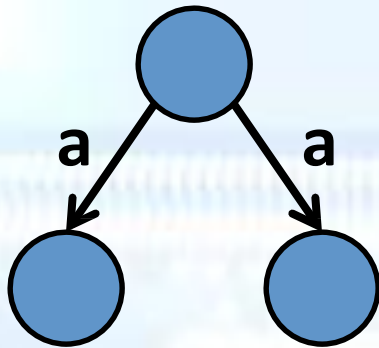
- Используется для:
 - построения абстракции реальной системы
 - для абстрагирования от деталей, несущественных для проверки свойств и для построения модели по неполной информации



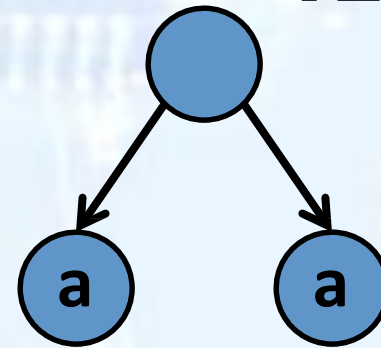


Недетерминизм в LTS

- В LTS недетерминизм проявляется в виде состояний, из которых можно перейти более чем в одно состояние



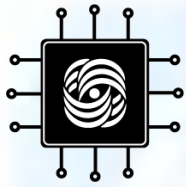
недетерминизм
действий



$AP = \{a\}$

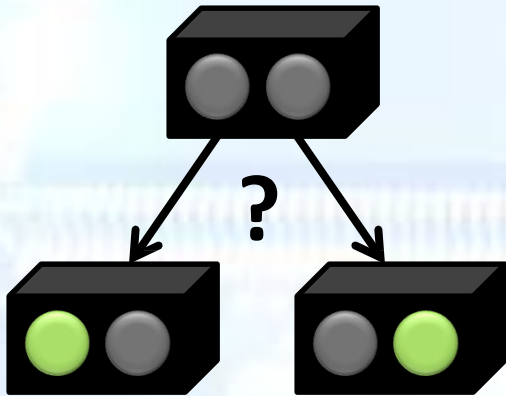
недетерминизм
атомарных высказываний

{ строгие определения – далее }

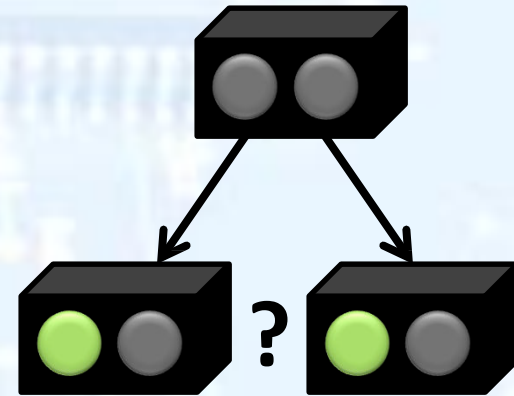


Недетерминизм в LTS

- В LTS недетерминизм проявляется в виде состояний, из которых можно перейти более чем в одно состояние

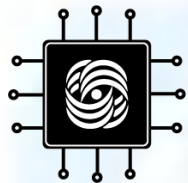


может произойти одно
из нескольких действий



наблюдатель не может
отличить два состояния

{ строгие определения – далее }



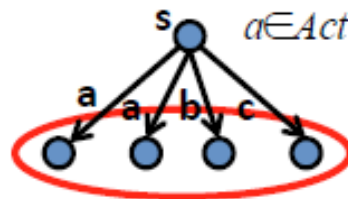
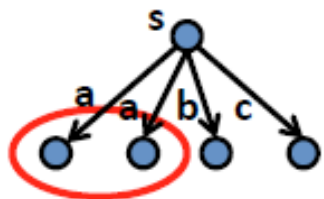
Вспомогательные определения

Прямые потомки вершины s

по действию a

по всем действиям

$$Post(s, a) = \{s' \in S \mid s \xrightarrow{a} s'\}, Post(s) = \bigcup_{a \in Act} Post(s, a)$$

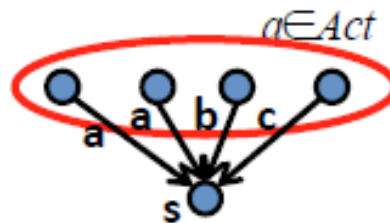
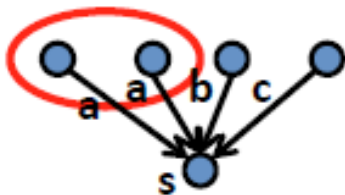


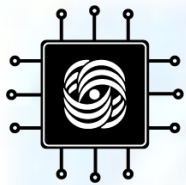
Прямые предки вершины s

по действию a

по всем действиям

$$Pre(s, a) = \{s' \in S \mid s' \xrightarrow{a} s\}, Pre(s) = \bigcup_{a \in Act} Pre(s, a)$$





Детерминизм

- Система $TS = \langle S, Act, \xrightarrow{a}, I, AP, L \rangle$

детерминирована

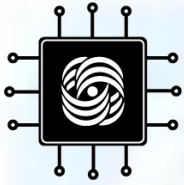
– по действиям, тогда и только тогда:

$$|I| \leq 1 \quad \text{и} \quad |Post(s, a)| \leq 1, \forall s \in S, a \in Act$$

– по атомарным высказываниям, тогда и только тогда:

$$|I| \leq 1 \quad \text{и} \quad \underbrace{|Post(s) \cap \{s' \in S \mid L(s') = A\}|}_{\text{одинаково размеченные потомки } s} \leq 1, \forall s \in S, A \in 2^{AP}$$

одинаково размеченные потомки s



Путь

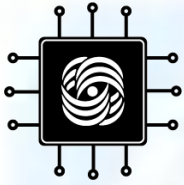
- *Конечным путём* σ системы переходов TS называется такая последовательность чередующихся состояний и действий, заканчивающаяся состоянием:

$$\sigma = s_0 a_1 s_1 a_2 s_2 \dots a_n s_n, \text{ что } s_i \xrightarrow{a_{i+1}} s_{i+1} \quad \forall 0 \leq i < n.$$

- *Бесконечным путём* σ системы переходов TS называется такая бесконечная последовательность чередующихся состояний и действий:

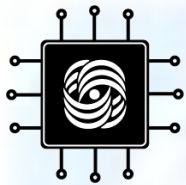
$$\sigma = s_0 a_1 s_1 a_2 s_2 a_3 \dots, \text{ что } s_i \xrightarrow{a_{i+1}} s_{i+1} \quad \forall 0 \leq i.$$

- Путь называется *начальным*, если $s_0 \in I$

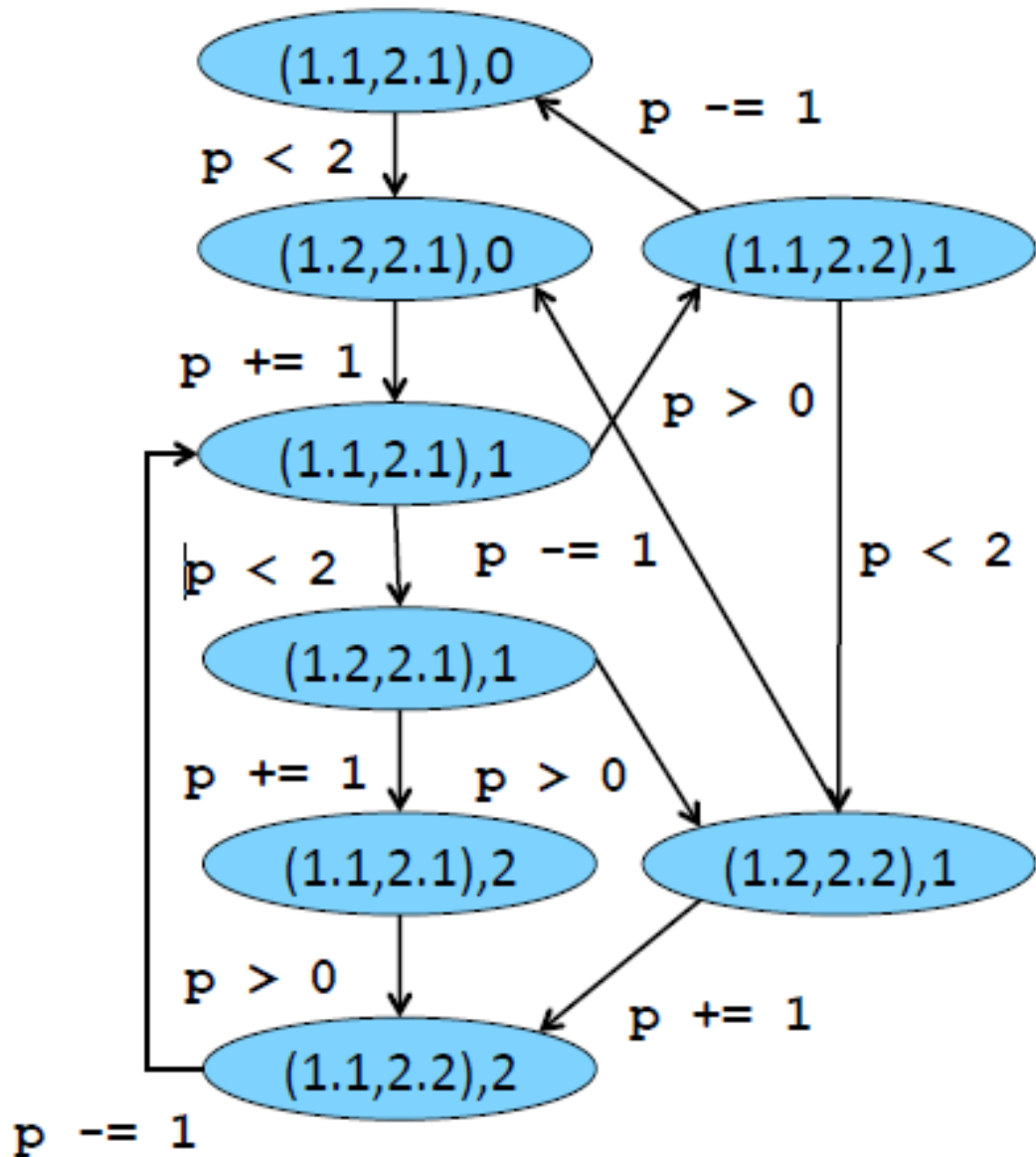
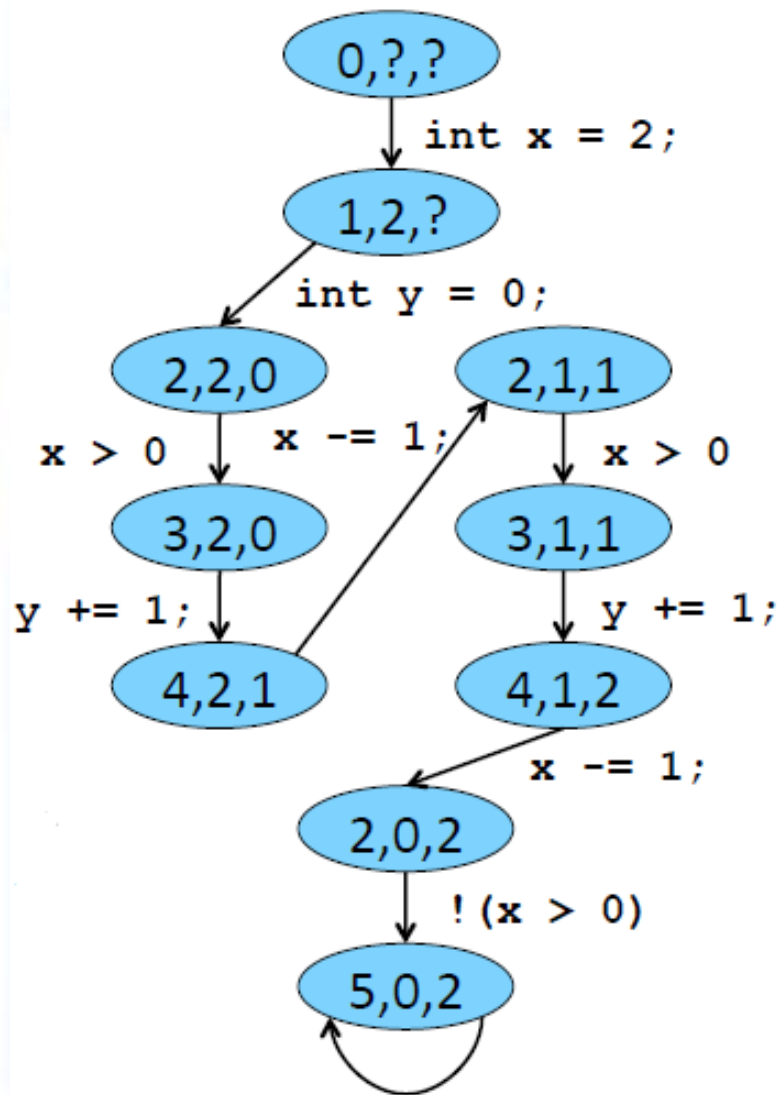


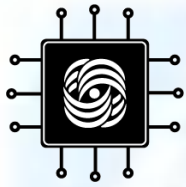
Вычисление

- Путь называется *максимальным*, если он бесконечен.
- *Вычислением* системы переходов TS называется начальный максимальный путь.

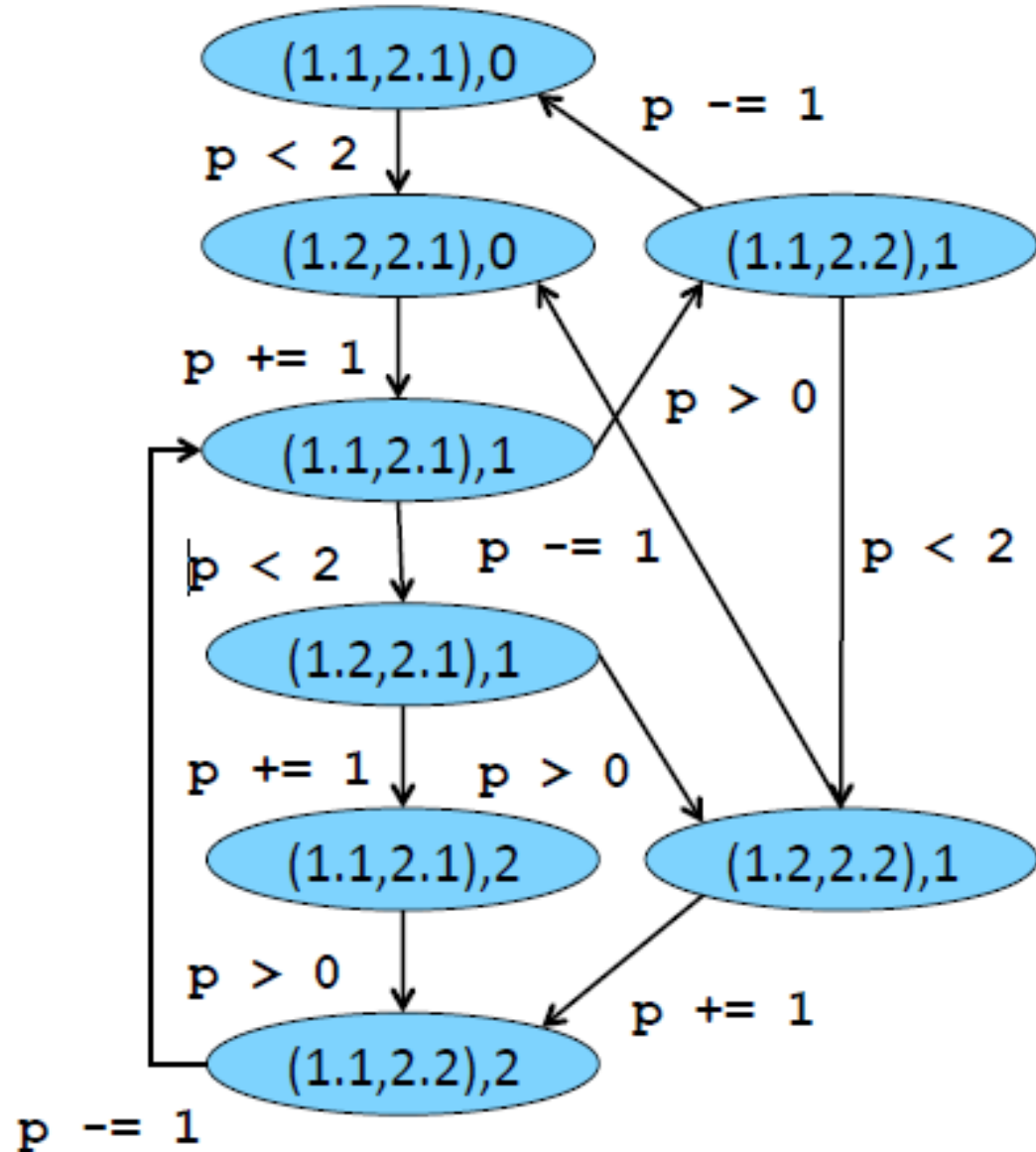
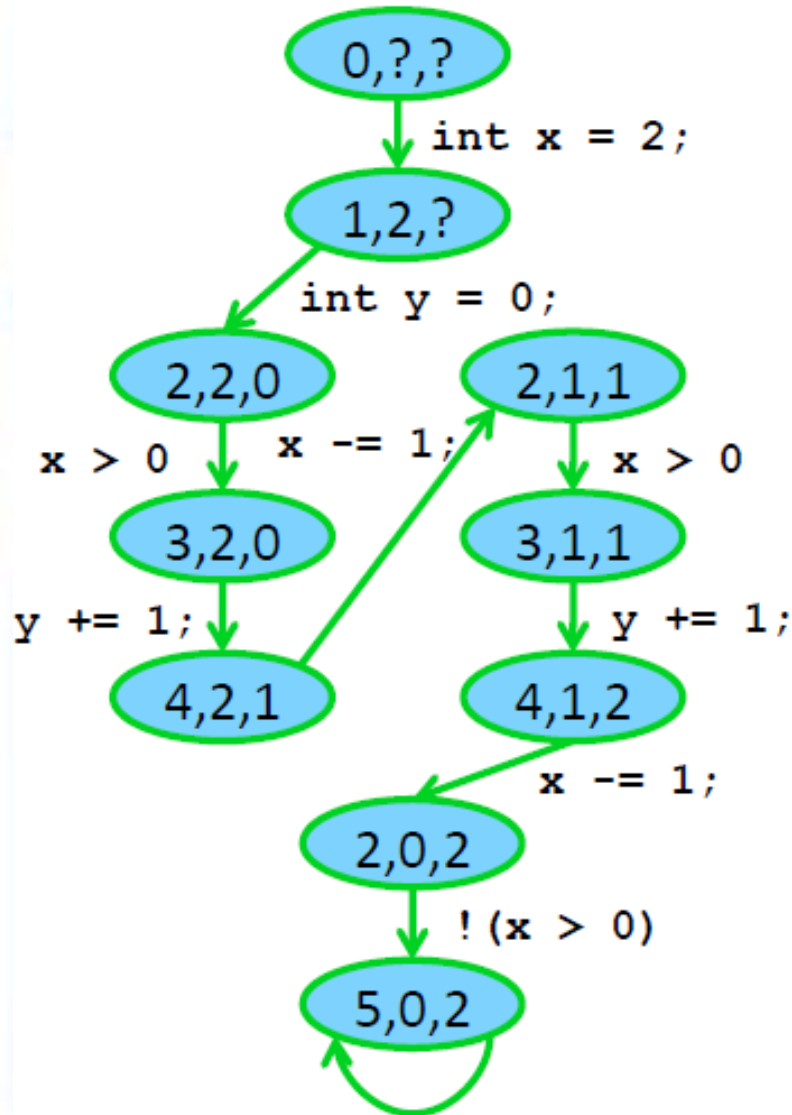


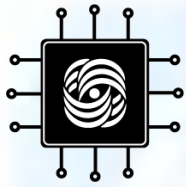
Примеры вычислений



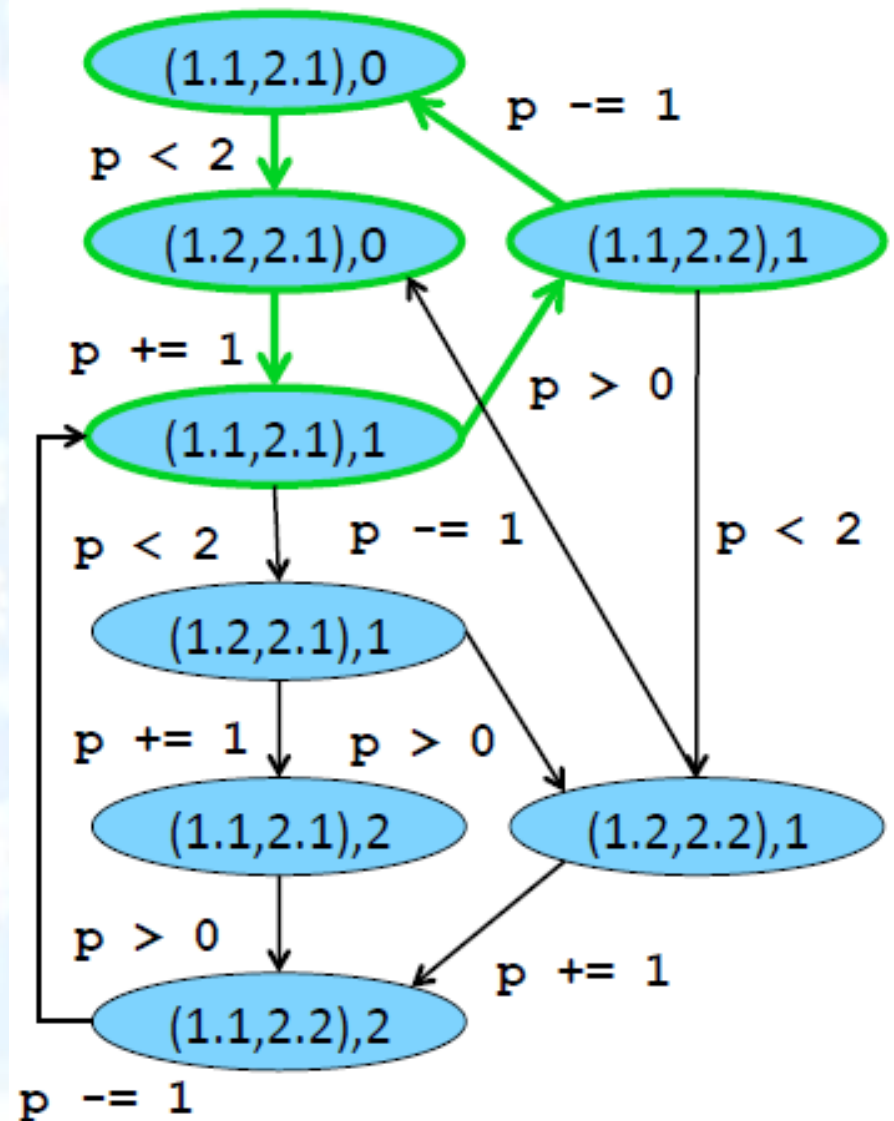
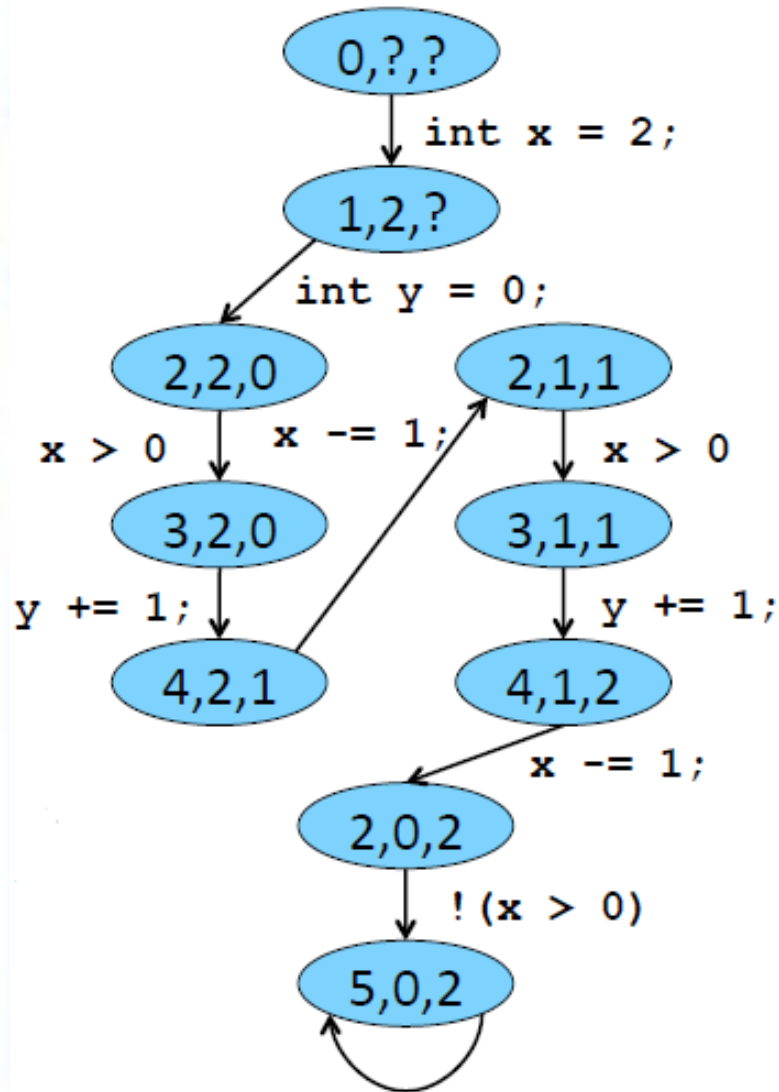


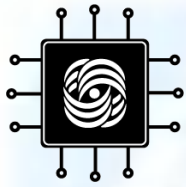
Примеры вычислений



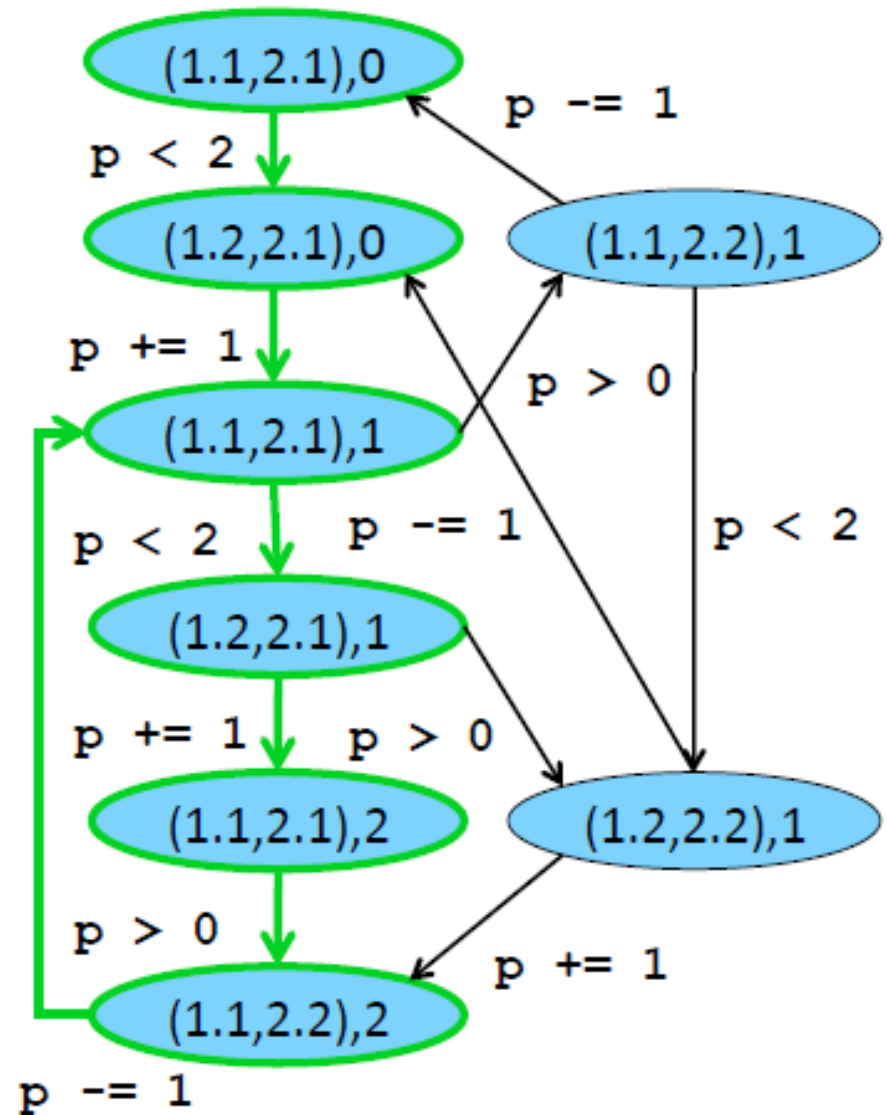
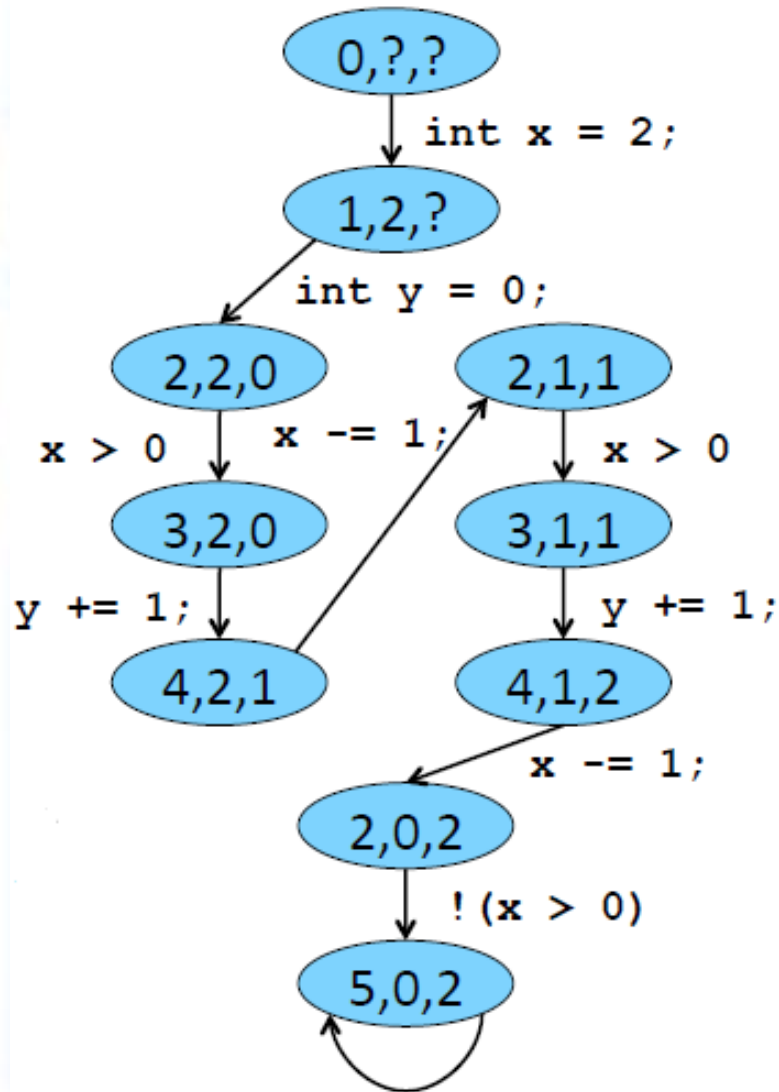


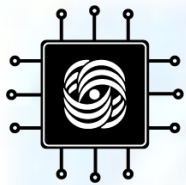
Примеры вычислений



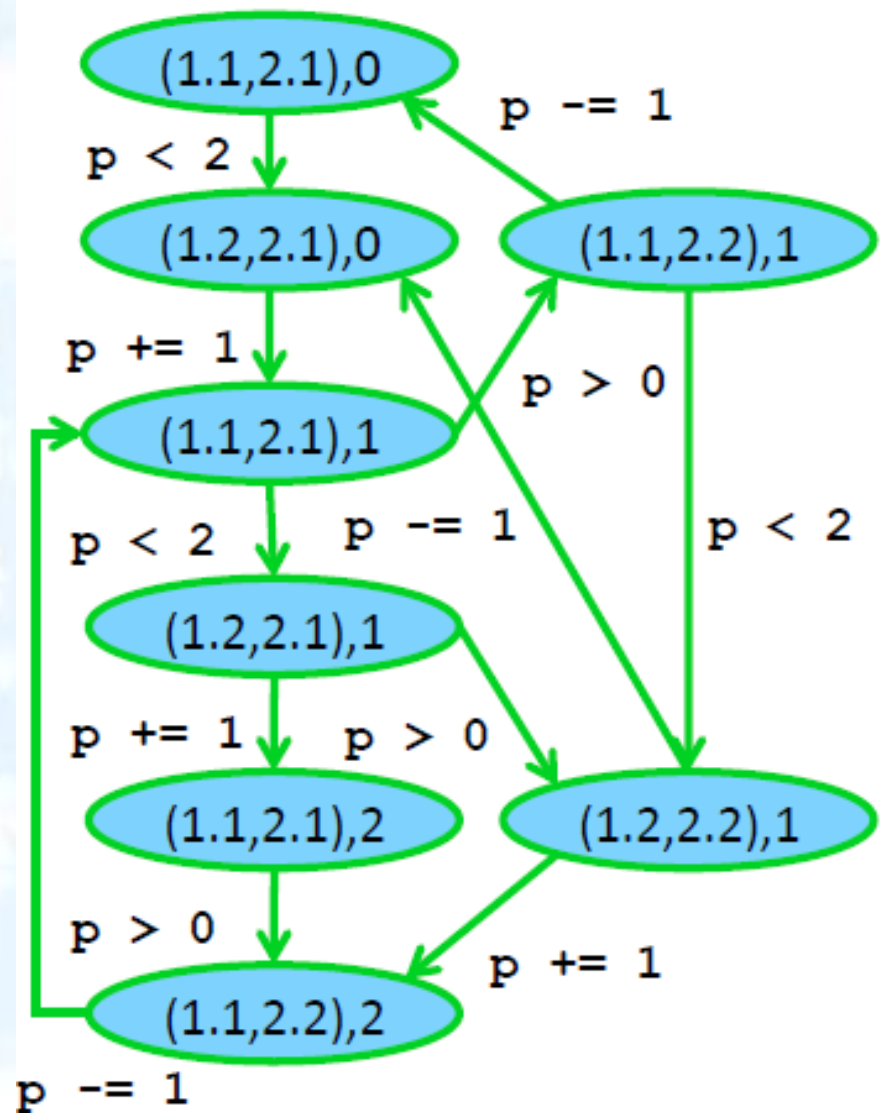
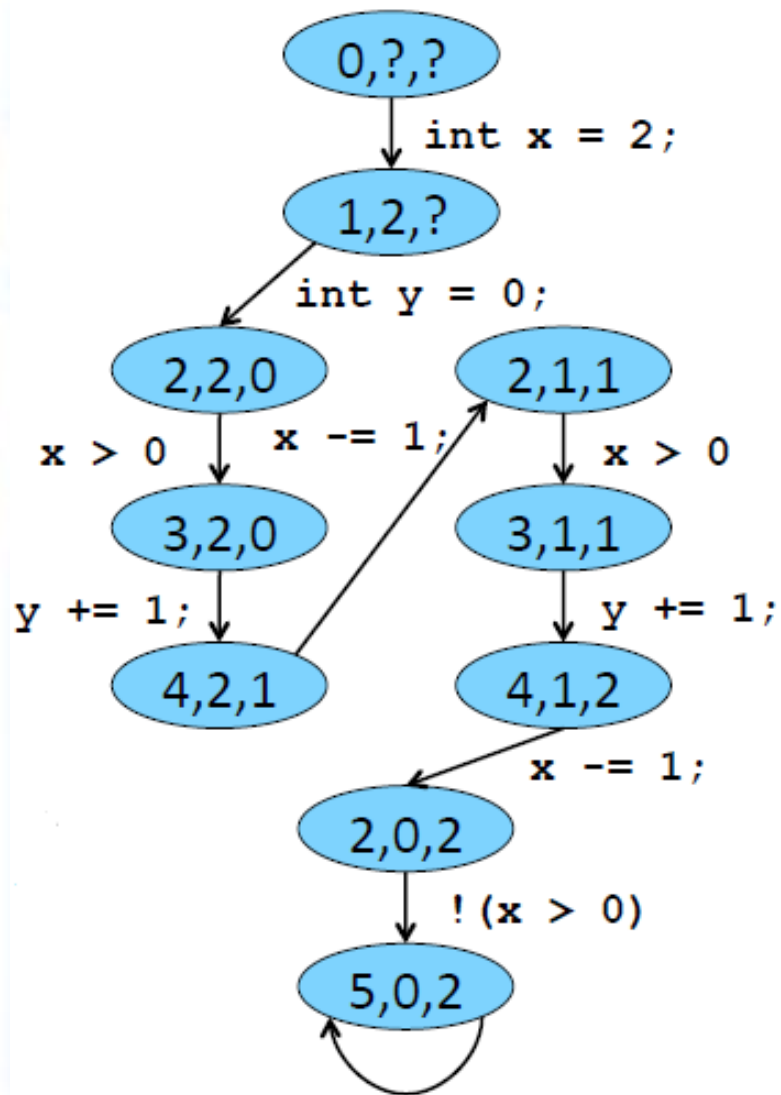


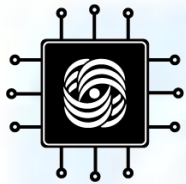
Примеры вычислений





Примеры вычислений



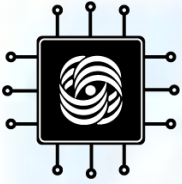


Достижимость состояний

- Состояние $s \in S$ называется достижимым (из начального) в системе переходов TS, если существует начальный, конечный путь

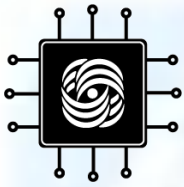
$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n = s.$$

- $Reach(TS)$ обозначает множество всех состояний, достижимых в TS.



Трассы

- Вычисление описывает последовательность состояний и действий; что происходит в системе. Требуется для описания семантики программы (позже).
- Свойства корректности формулируются в терминах последовательностей значений атомарных высказываний в состояниях модели.



Трассы

- Система переходов:

$$TS = \langle S, Act, \xrightarrow{a}, I, AP, L \rangle$$

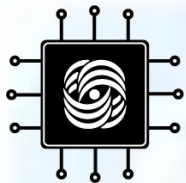
- Путь (фрагмент вычисления):

$$\sigma = s_0 a_1 s_1 a_2 s_2 a_3 \dots$$

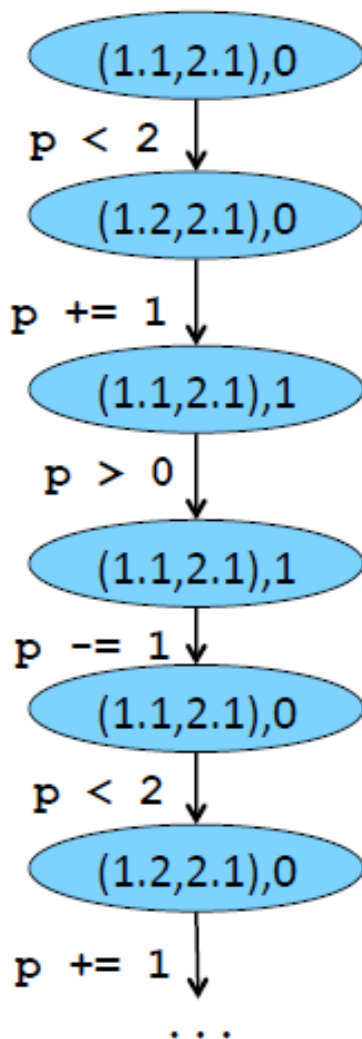
- Трасса:

$$tr = L(s_0)L(s_1)L(s_2)\dots \in (2^{AP})^\omega$$

[фокусируемся на «наблюдаемом» поведении]



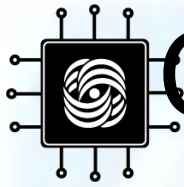
вычисление



Примеры трасс

трасса $\{a \equiv p = 0\}$ трасса 2 $\left\{ \begin{array}{l} a \equiv p \neq 0 \\ b \equiv p = 1 \end{array} \right\}$

a	!a !b
a	!a !b
!a	a
!a	a b
a	!a !b
a	!a !b



Свойства линейного времени

- Атомарные высказывания:

$\left\{ \begin{array}{l} \mathbf{a} - \text{«находимся в точке отправки запроса»} \\ \mathbf{b} - \text{«находимся в точке приёма ответа»} \end{array} \right\}$

- СВОЙСТВО:

«после отправки запроса рано или поздно получим ответ»

- Пример допустимых трасс:

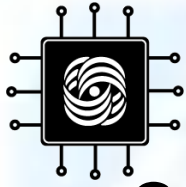
$a \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ b \ \emptyset \ \dots$

$\emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \dots$

- Пример недопустимых трасс:

$a \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ \dots$

$a \ \emptyset \ \emptyset \ \emptyset \ \emptyset \ b \ a \ \emptyset \ \dots$



Свойства линейного времени

- Свойство φ определяет набор допустимых трасс

$$\varphi \subseteq (2^{AP})^\omega$$

- Свойство φ выполнимо на трассе σ :

$$\sigma \models \varphi \Leftrightarrow \sigma \in \varphi$$

- Система переходов TS удовлетворяет свойству линейного времени φ :

$$TS \models \varphi \Leftrightarrow Traces(TS) \subseteq \varphi$$

$\models$

$TS(P)$

$\varphi \equiv P$

φ

$\models$

$\models$

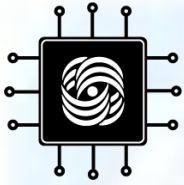
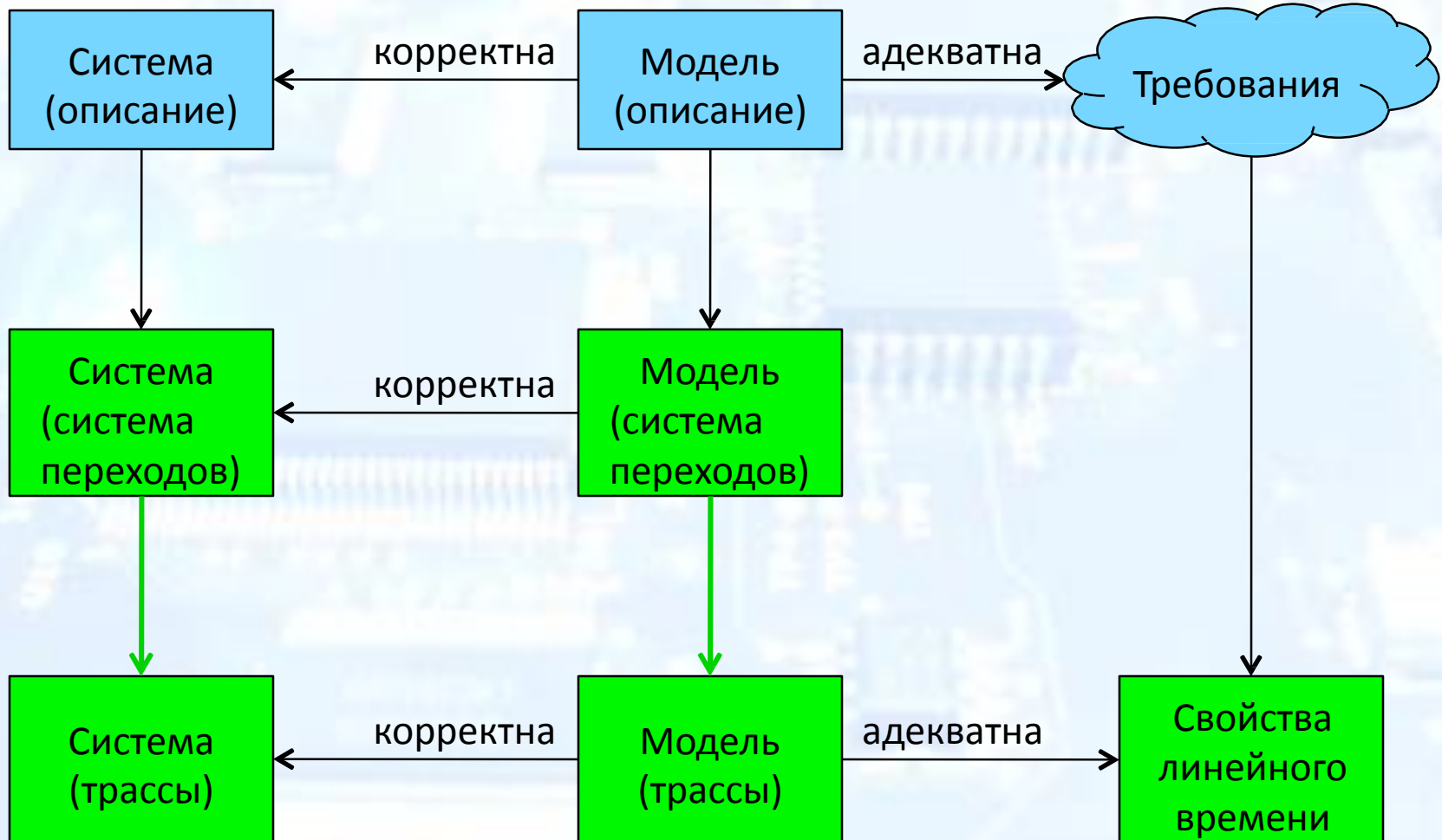
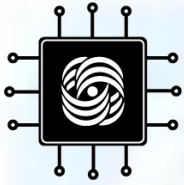


Схема понятий





Абстракция трасс

- Представим трассу в форме интерпретации I :

$$I(tr) = \langle N, \leq, \xi \rangle$$

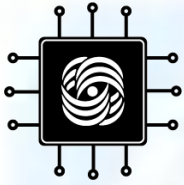
где N – множество натуральных чисел,

$\leq$ – отношение порядка на N , а

$\xi : N \times AP \rightarrow \{T, \perp\}$, и при этом

$$\forall n > 0, p \in AP$$

$$\xi(n, p) = T \Leftrightarrow p \in L(s)$$



Абстракция трасс

- Рассмотрим трассы tr и tr' такие, что:

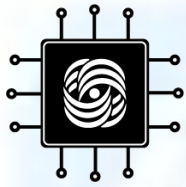
$$I(tr) = \langle N, \leq, \xi \rangle \quad I(tr') = \langle N, \leq, \xi' \rangle$$
$$\xi : N \times AP \rightarrow \{T, \perp\} \quad \xi' : N \times AP' \rightarrow \{T, \perp\}$$

- Будем говорить, что tr' является *(корректной) абстракцией* tr ($tr \prec tr'$), если

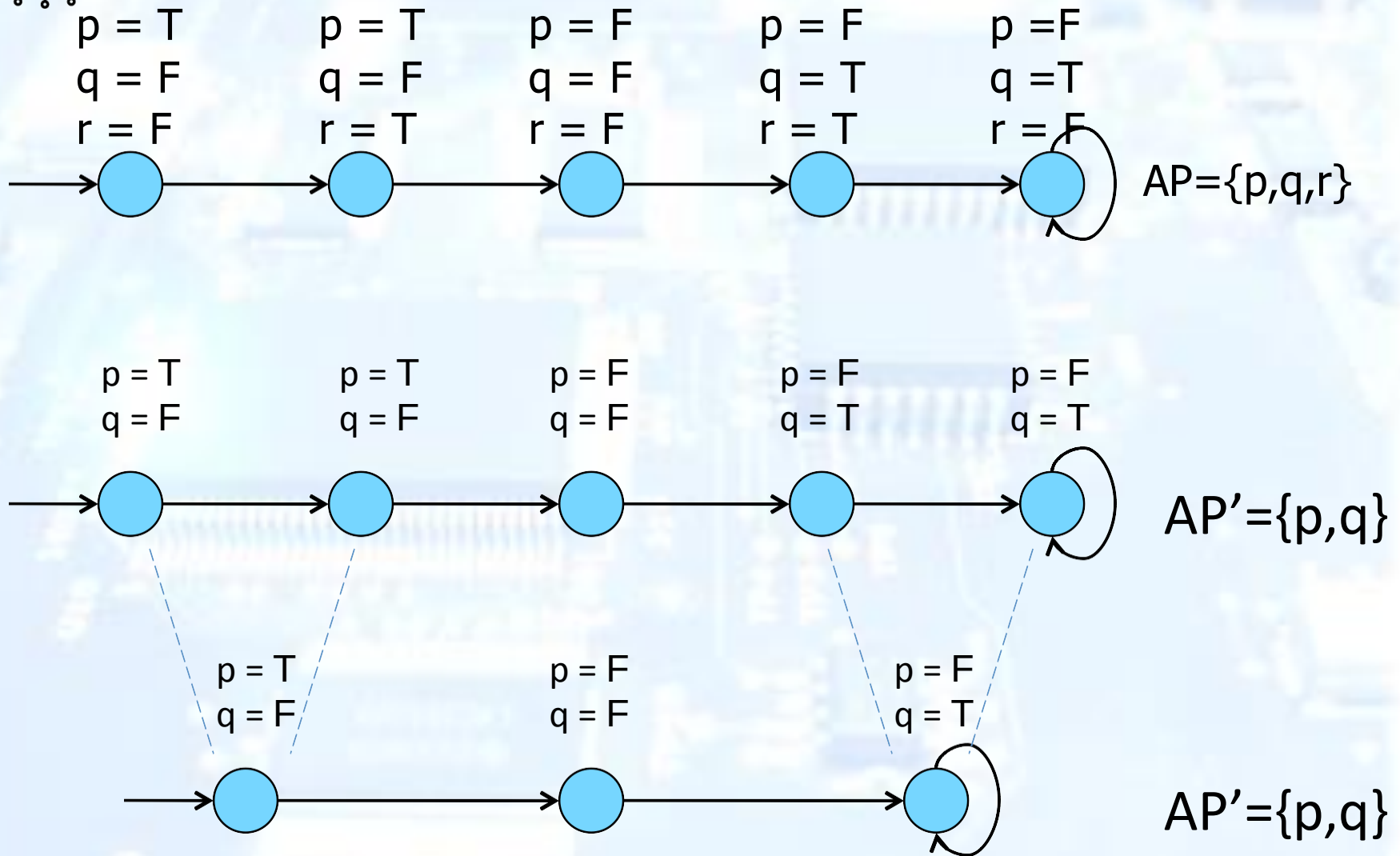
$$AP' \subseteq AP$$

$$\exists \alpha : N \rightarrow N : \forall n, k \in N \left(n \leq k \Rightarrow \alpha(n) \leq \alpha(k) \right)$$

$$\forall n \in N, p \in AP' \left(\xi(n, p) = \xi'(\alpha(n), p) \right)$$



Пример абстракции трасс



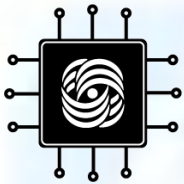
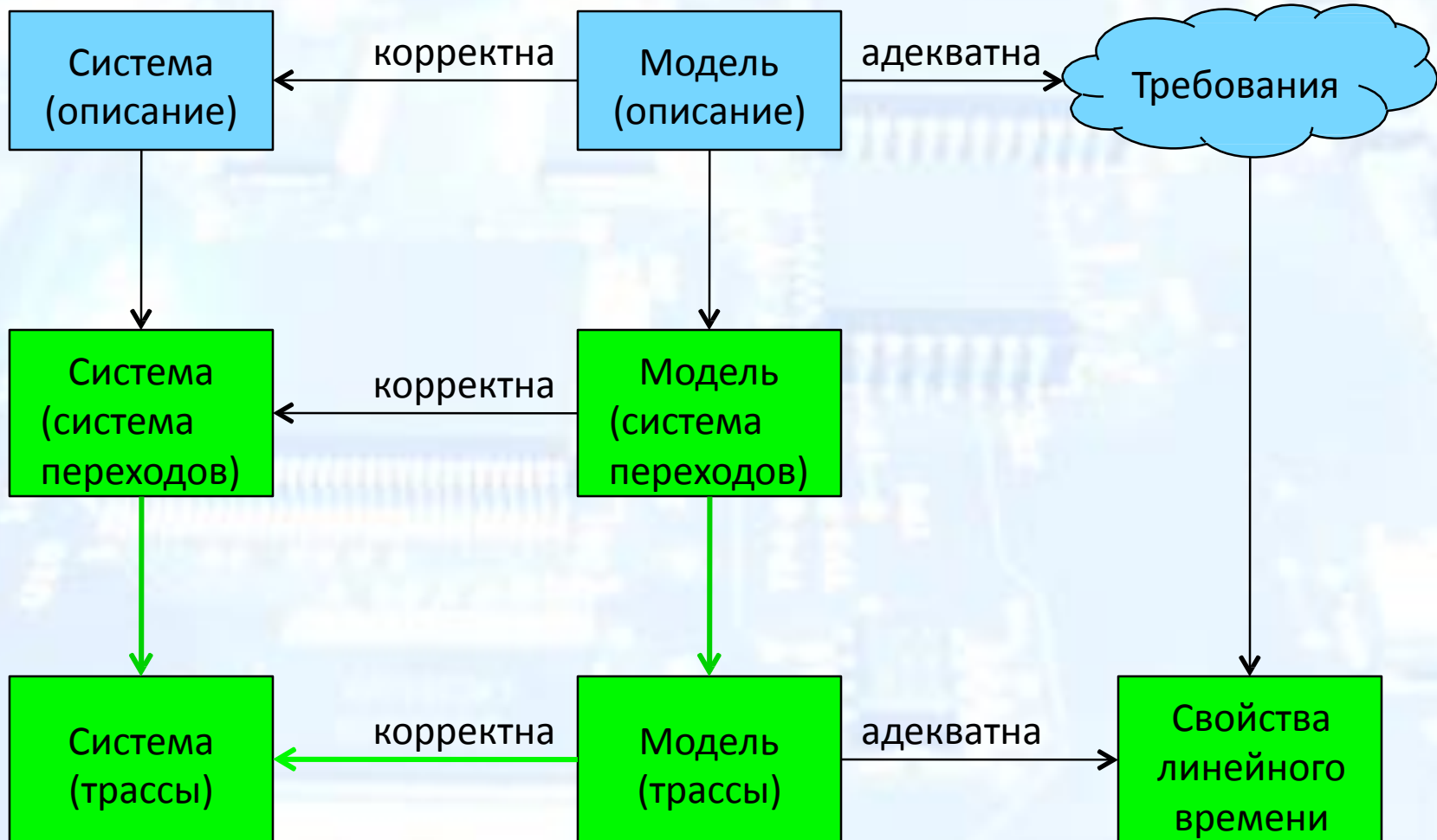
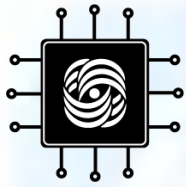


Схема понятий





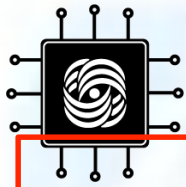
Условие корректности модели

- Пусть P – система, φ – произвольное свойство линейного времени. (Корректной) моделью P называется такое M , что:

если свойство выполняется на модели, то оно выполняется и на системе

- Это выполняется тогда и только тогда, когда:

для любой трассы исходной системы в модели найдётся её корректная абстракция



Условие корректности модели

- Пусть P – система, φ – произвольное свойство линейного времени. (Корректной) моделью P называется такое M , что:

$$M \models \varphi \Rightarrow P \models \varphi$$

⎡ позволяет проверять свойства
программы на её модели ⎤

определение

- Это выполняется тогда и только тогда, когда:

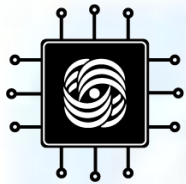
$$\forall tr \in Traces(TS(P)) \exists tr' \in Traces(TS(M)) :$$

$$tr \prec tr'$$

⎡ для проверки такого условия нужно рассмотреть
все трассы исходной системы ⎤

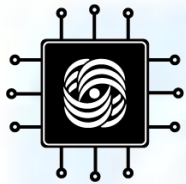
⎡ допускает, что в модели
больше состояний ⎤

необходимое и достаточное условие



Достаточное условие корректности

- Какими же свойствами должна обладать TS модели, чтобы быть корректной?
 - действиям и атомарным высказываниям модели должны быть сопоставлены действия и атомарные высказывания системы,
 - каждому состоянию системы должно быть сопоставлено состояние модели,
 - модель должна корректно сохранять множество начальных состояний,
 - если в системе есть переход между двумя состояниями, в модели должен быть переход по между соотв. состояниями по соотв. действию,
 - соотв. состояния в модели и системе должны быть размечены атомарными высказываниями модели одинаково.



Достаточное условие корректности

- Какими же свойствами должна обладать TS модели, чтобы быть корректной?

$$TS = \langle S, Act, \xrightarrow{a}, I, AP, L \rangle$$
$$TS' = \langle S', Act', \xrightarrow{a}', I', AP', L' \rangle$$

$$Act' \subseteq Act$$

$$AP' \subseteq AP$$

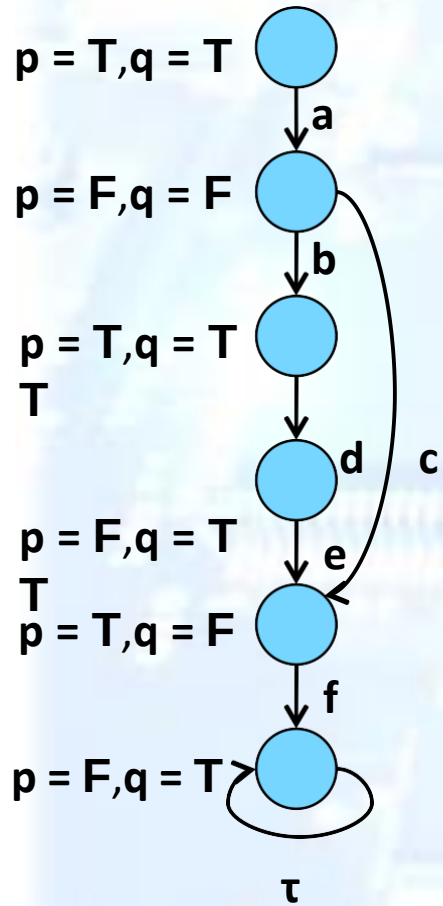
$$\exists \alpha : S \rightarrow S', s_0' = \alpha(s_0)$$

$$s_1 \xrightarrow{a} s_2 \Rightarrow \alpha(s_1) \xrightarrow{a'} \alpha(s_2)$$

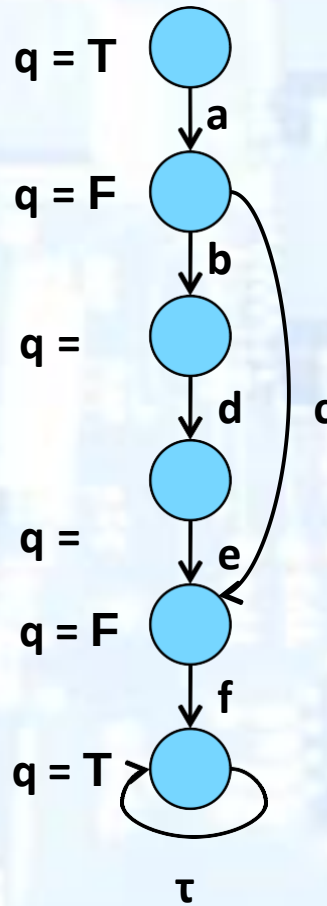
$$\forall s \in S, L'(\alpha(s)) = L(s) \cap AP'$$



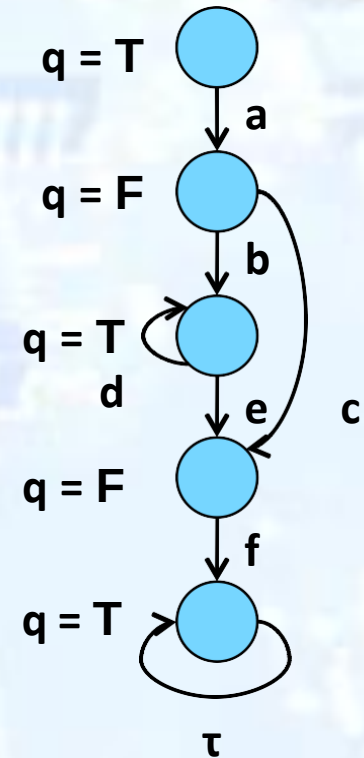
Пример корректной абстракции системы переходов



(P)



(M_1)



(M_2)

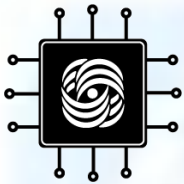
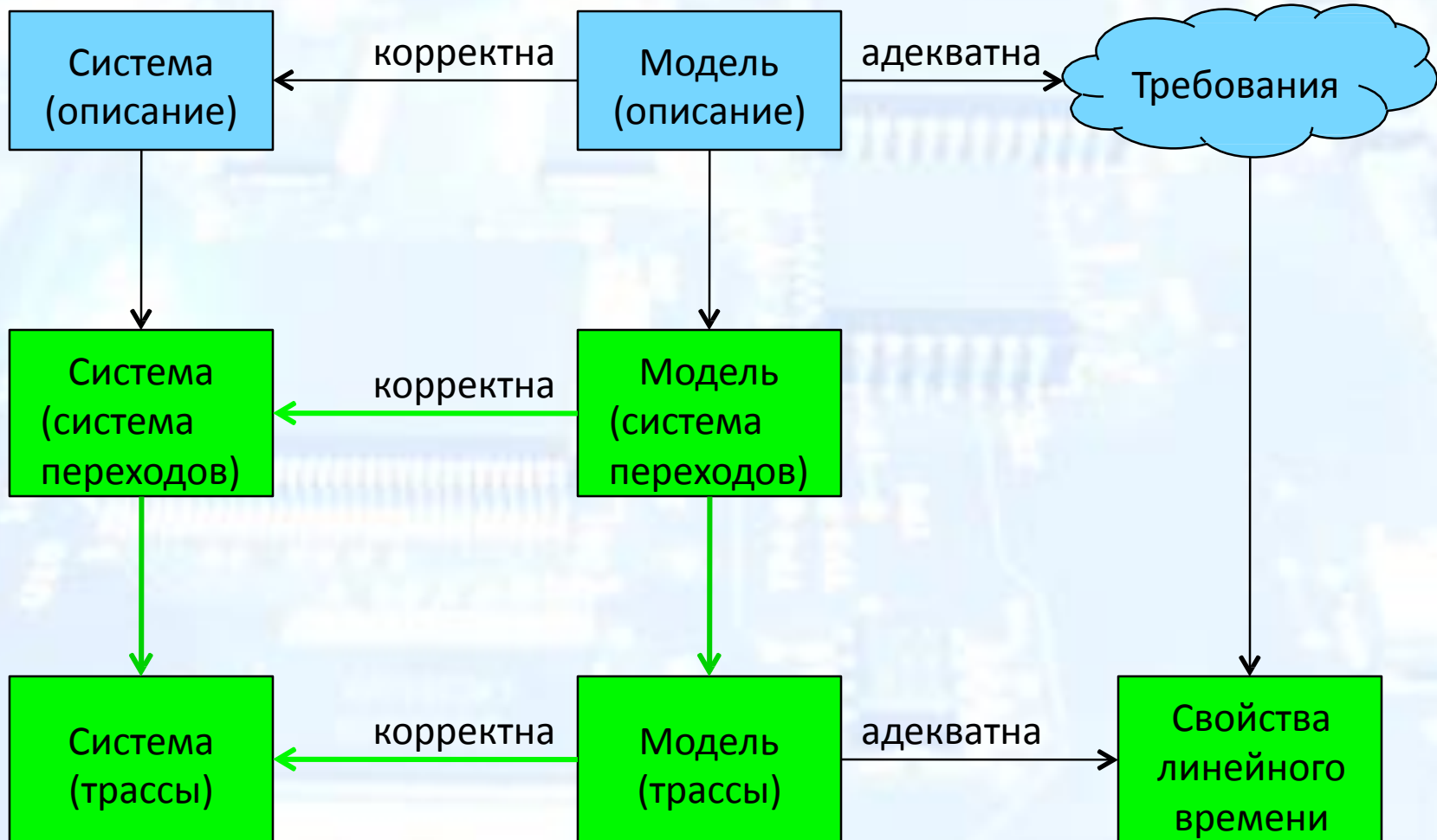
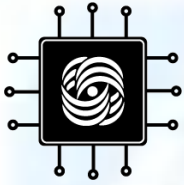


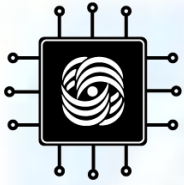
Схема понятий





Адекватность модели

- Модель называется адекватной, если:
 1. Атомарные высказывания, в терминах которых задаются свойства, присутствуют в разметке модели
 2. Из нарушения свойства на модели следует, что оно нарушается и на исходной системе



Адекватность модели

- Модель называется адекватной, если:

$$1. AP_{\varphi} \subseteq AP_M$$

необходимое условие (можно вычислить)

2.

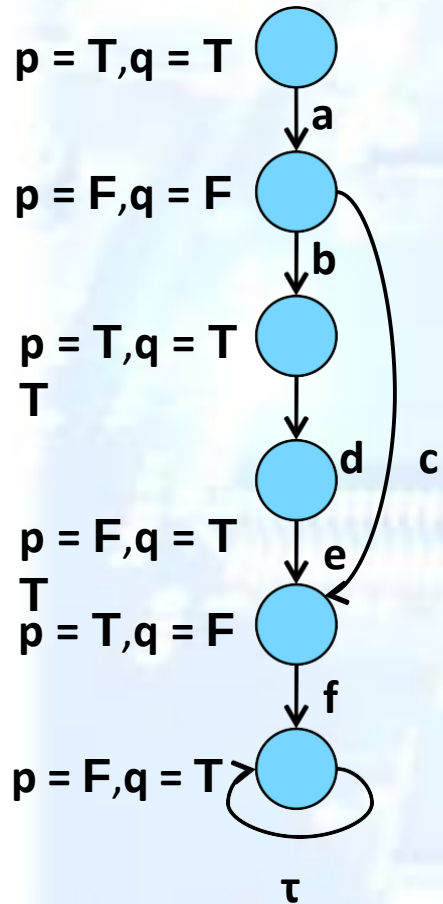
$$M \not\models \varphi \Rightarrow P \not\models \varphi$$

достаточное условие (нельзя вычислить)

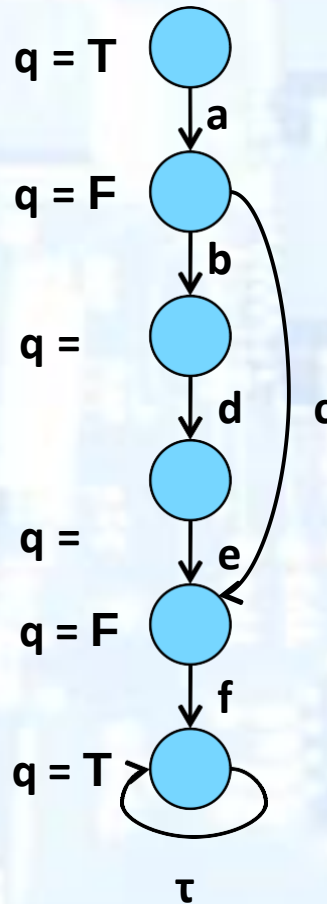
- Определить адекватность при построении нельзя, можно лишь обнаружить несоответствие и исправить модель



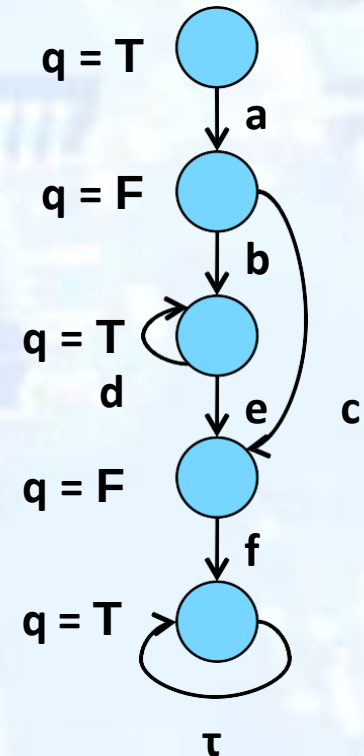
Пример корректной абстракции системы переходов



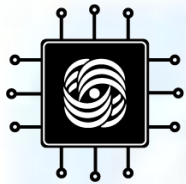
(P)



(M₁)



(M₂)



Пример – проверяемые свойства

- В любом вычислении встречается состояние, когда $p=T$ & $q=T$ – *ни одна из моделей не адекватна*,
- Для любого пути верно, что за любым $q=F$ рано или поздно встретится $q=T$ – *все модели адекватны*,
- Между двумя состояниями с $q=F$ встречается не более чем 3 состояния с $q=T$ – *модель M_1 адекватна, M_2 – нет.*

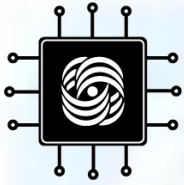
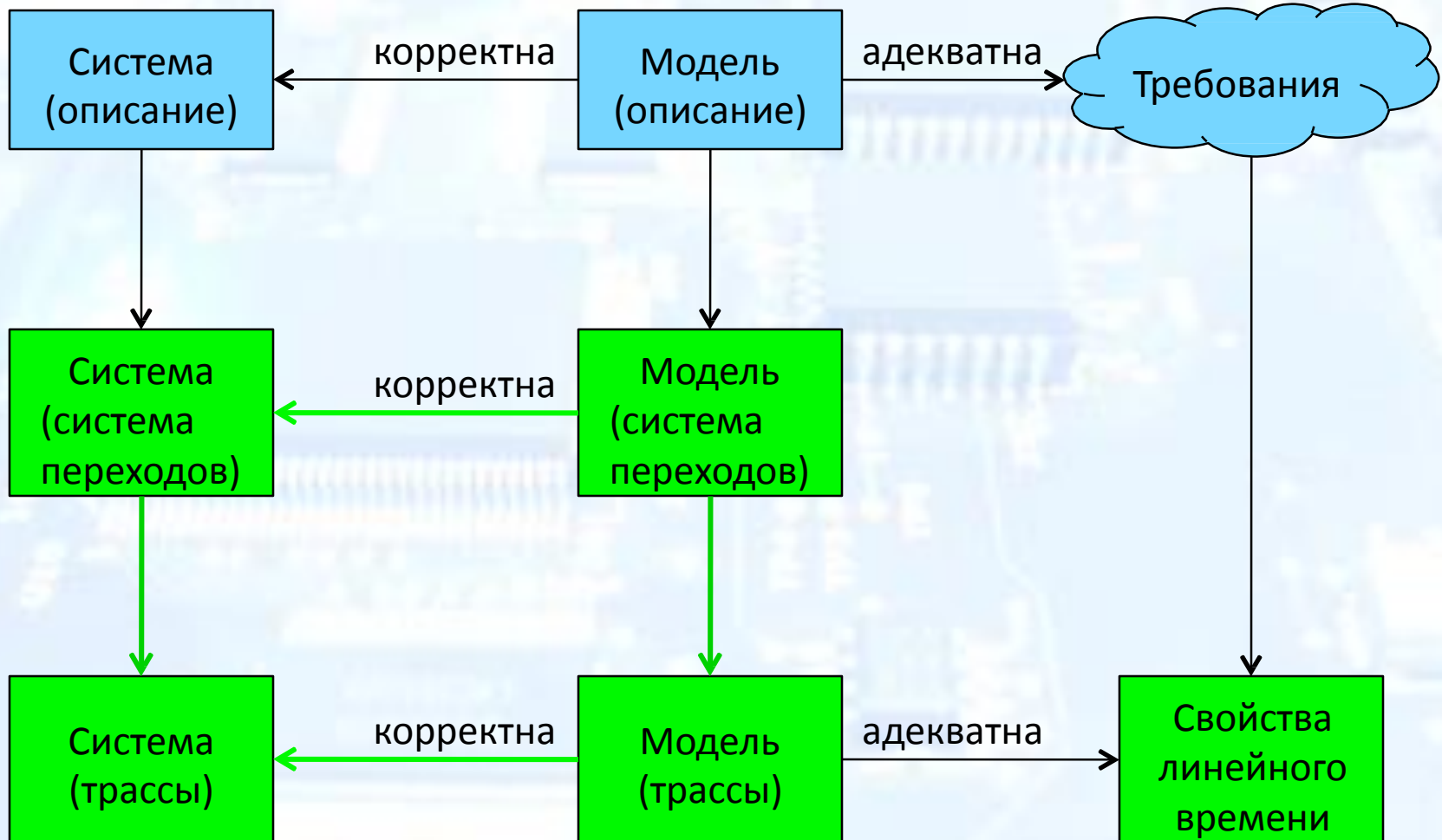
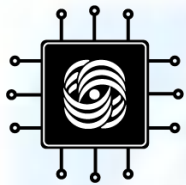


Схема понятий





Спасибо за внимание!