Верификация программ на моделях

Лекция №5

Графы программ. Системы с каналами взаимодействия. Синхронный и асинхронный параллелизм

Константин Савенков (лектор)

План лекции

- Alternating Bit Protocol
- Графы программ
- Операционная семантика графов программ:
  - последовательные процессы,
  - чередование,
  - разделяемые переменные,
  - синхронная и асинхронная передача сообщений

Alternating Bit Protocol
(Bartlett и др., 1969)

- Два процесса, отправитель и получатель;
- К каждому сообщению добавляется один *бит*;
- Получатель сообщает о доставке сообщения, возвращая бит отправителю;
- Если отправитель убедился в доставке сообщения, он отправляет новое, изменяя значение бита;
- Если значение бита не изменилось, получатель считает, что идёт повтор сообщения.

Функция eval()

Отображает текущее значение x на константу, которая служит ограничением для принимаемых сообщений

```
ch!msg(12)
ch?msg(eval(x))
```

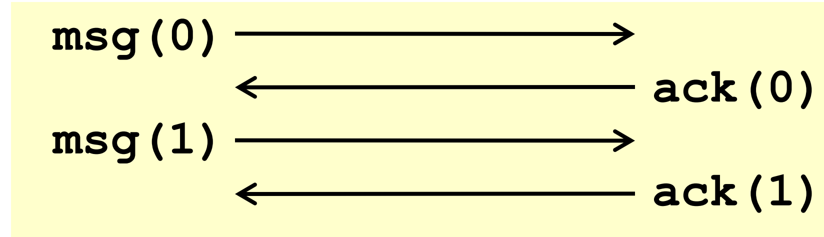
Сообщение будет принято, если значение переменной x равно 12

Модель на Promela

```
mtype = {msg, ack};
chan s_r = [2] of {mtype, bit};
chan r_s = [2] of {mtype, bit};

active proctype sender()
{ bit seqno;
  do
  :: s_r!msg,seqno ->
       if
       :: r_s?ack,eval(seqno) ->
           seqno = 1 - seqno;
       :: r_s?ack,eval(1-seqno)
       fi
  od
}

active proctype receiver()
{ bit expect, seqno;
  do
  :: s_r?msg,seqno ->
       r_s!ack,seqno;
       if
       :: seqno == expect;
          expect = 1 - expect
       ::else
       fi
  od
```



Считываем новое сообщение

Сохраняем сообщение

Игнорируем сообщение

Запускаем моделирование

```
>./spin -u20 -c abp.pml
proc 0 = sender
proc 1 = receiver
q\p   0   1
  1   s_r!msg,0
  1   .   s_r?msg,0
  2   .   r_s!ack,0
  2   r_s?ack,0
  1   s_r!msg,1
  1   .   s_r?msg,1
  2   .   r_s!ack,1
  2   r_s?ack,1
-------------
depth-limit (-u20 steps) reached
-------------
final state:
-------------
#processes: 2
                queue 1 (s_r):
                queue 2 (r_s):
 20:    proc  1 (receiver) line  19 "abp.pml"
(state 7)
 20:    proc  0 (sender) line   7 "abp.pml"
(state 7)
2 processes created
>
```

Моделируем первые 20 шагов

Верификация по умолчанию

```
>./spin -a abp.pml
> gcc -o pan pan.c
>./pan
(Spin Version 5.1.4 -- 27 January 2008)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid end states      +

State-vector 44 byte, depth reached 13, errors: 0
       14 states, stored
        1 states, matched
       15 transitions (= stored+matched)
        0 atomic steps
hash conflicts:          0 (resolved)

    2.501       memory usage (Mbyte)

unreached in proctype sender
        line 15, state 10, "-end-"
        (1 of 10 states)
unreached in proctype receiver
        line 27, state 10, "-end-"
        (1 of 10 states)
>
```

Чем и как проверяем?

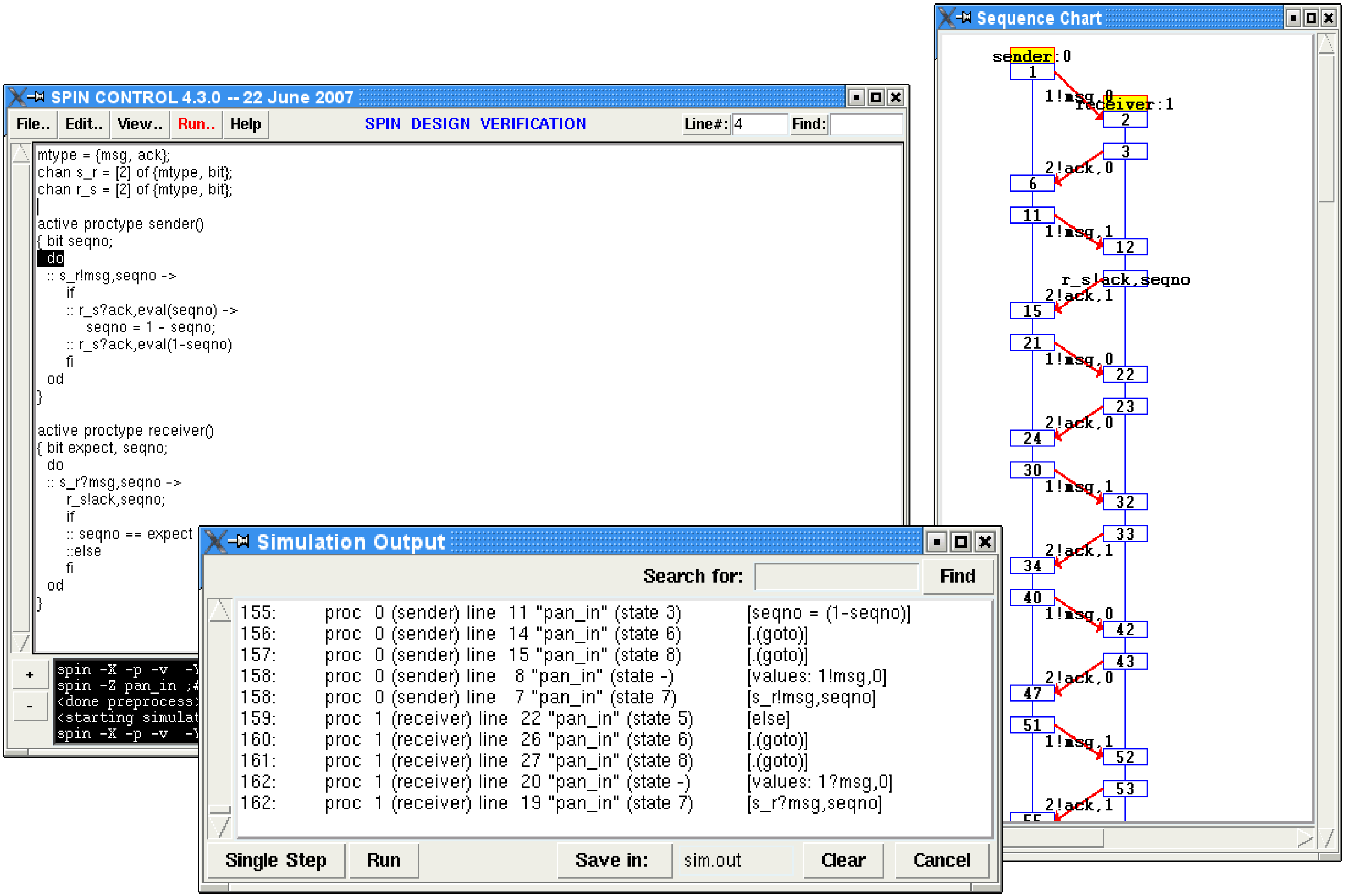
Какие свойства?

Проделанная работа

Используемая память

Обнаружен недостижимый код (процессы не завершаются)

Графический интерфейс xspin



Пространства состояний в SPIN («отладка» процесса верификации)

Полезные инструменты

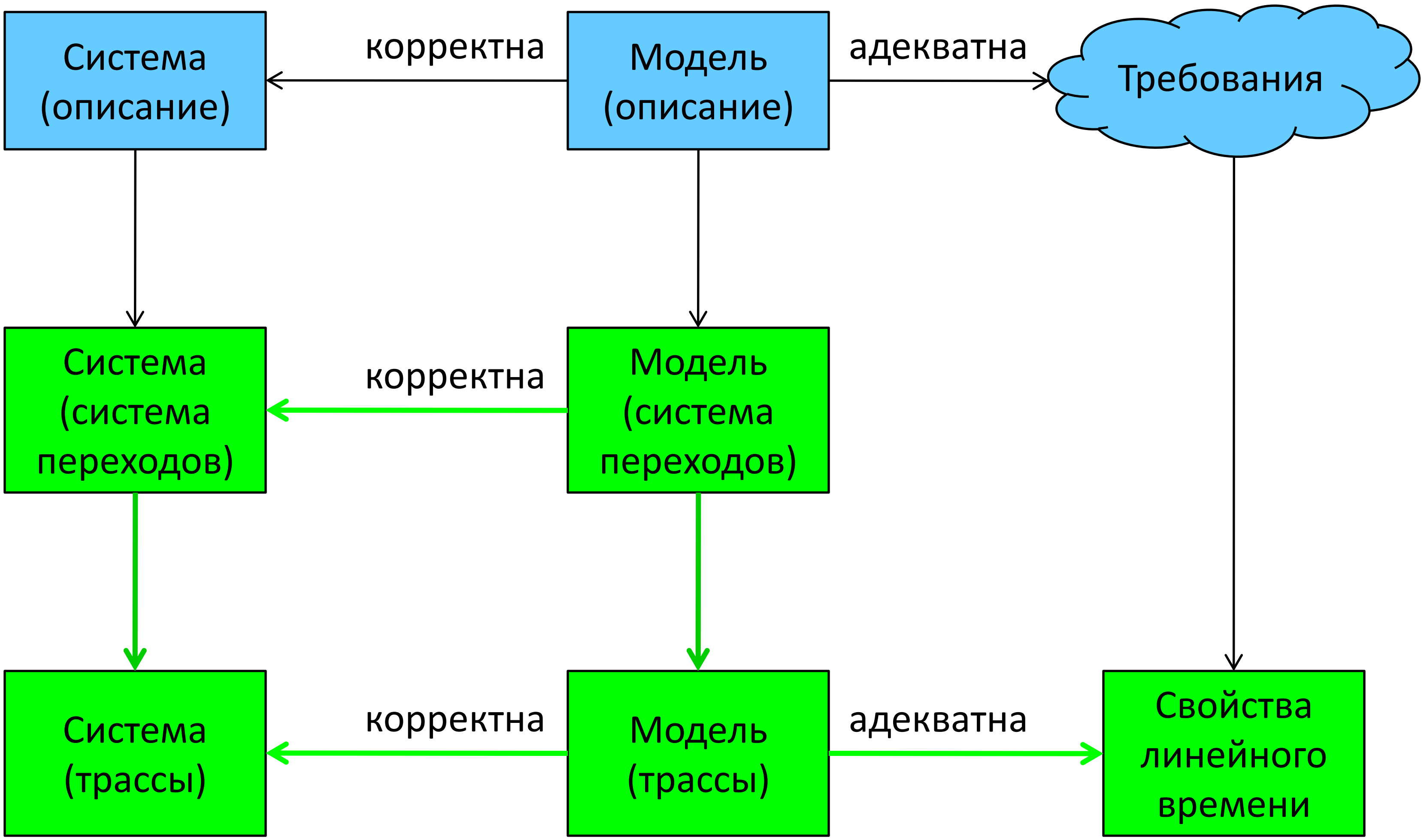
- Просмотр пространства состояний:
  - параметры компиляции pan.c:
    - -DCHECK – выводить порядок обхода пространства состояний
    - -DVERBOSE -DSDUMP – выводить вектора состояний
    - -DBFS – обход в ширину (удобнее для анализа)
  - параметры запуска pan:
    - -d – вывод графов процессов (state – номер оператора)
- Отключение оптимизаций:
  - параметры spin:
    - -o1 – отключение оптимизации потока данных,
    - -o2 – отключение удаления мертвых переменных,
    - -o3 – отключение слияния состояний
  - параметры компиляции pan.c:
    - -DNOREDUCE – отключение редукции частичных порядков

Обратите внимание:

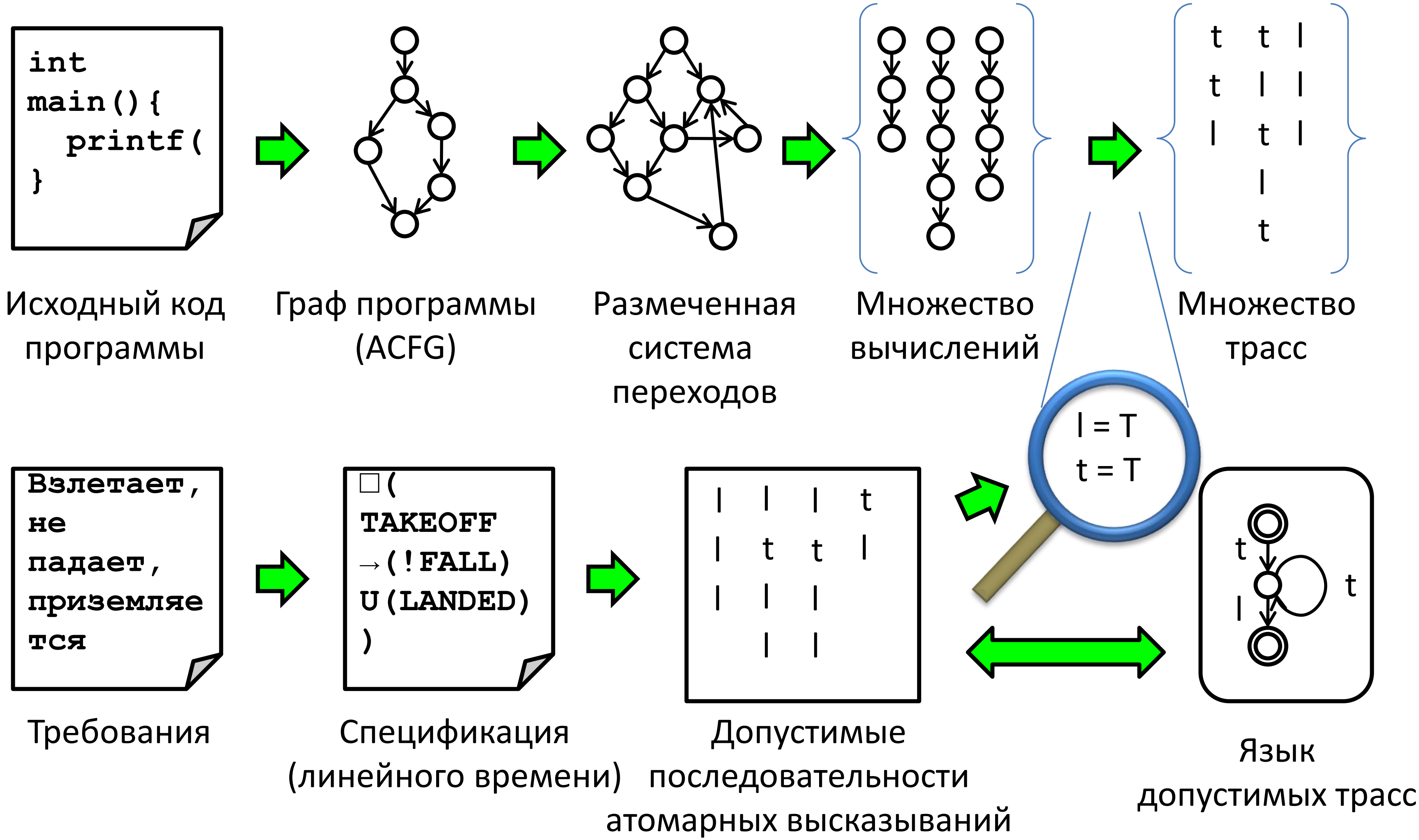
- инициализация переменной (`int x = 1`) не считается действием;
- порождение (run) и завершение процесса – действия,
  - при использовании active в начальном состоянии процесс уже запущен,
  - не только терминальное состояние, но и терминальное действие -end-;
- процессы порождаются в случайном порядке, но завершаются в только порядке, обратном порядку порождения (LIFO);
- проверка стража ветвления – действие.

Графы программ

Схема понятий



Различные представления программы



Размеченные системы переходов

(напоминание)

- описывают поведение системы;
- ориентированный граф: узлы – состояния, дуги – переходы;
- **состояние** – счётчик управления + значения переменных программы;
- **переход** (изменение состояния) – выполнение оператора программы.

Размеченные системы переходов

(напоминание)

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

- S – множество состояний,
- Act – множество действий, τ – невидимое действие,
- $\xrightarrow{a} = S \times Act \times S$ – **тотальное** отношение

переходов,

- $I \subseteq S$ – **множество** начальных состояний,
- AP – множество атомарных высказываний,
- $L : S \to 2^{AP}$ – функция разметки.

Формальное представление программы

- LTS – всевозможные состояния программы и переходы между ними;
- Однако модель строится в виде программы на специальном языке;
- Рассуждения о корректности необходимо перенести на текст программы-модели;
- Для этого понятие описания программы необходимо формализовать;
- Формальное представление программы: граф программы и его семантика.

Формальное представление программы

1. **граф**, задающий структуру программы;
2. **статическая семантика** – набор ограничений, которым должна удовлетворять структура;
3. **операционная семантика** – понятие состояния программы и изменение состояния в ходе работы программы.

(то, как по графу строится LTS)

Вспомогательные определения

- $\boxed{D_P}$ – единый абстрактный домен данных

```
bool z;
mtype {M1,M2} m = M1;

proctype EQ(byte x, byte y)
{
  if
  :: (x == y) -> z = true
  :: else -> z = false
  fi
}
```

$$D_P \equiv \mathrm{int}$$

$$bool \subset \mathrm{int}$$

$$byte \subset \mathrm{int}$$

$$mtype \subset \mathrm{int}$$

Вспомогательные определения

- $\boxed{V_P} \in Var$ – множество переменных (последовательной) программы P,
- $\forall v \in V_P, \boxed{dom(v)} = D_P^v \subseteq D_P$

```
bool z;
mtype {M1,M2} m = M1;

proctype EQ(byte x, byte y)
{
  if
  :: (x == y) -> z = true
  :: else -> z = false
  fi
}
```

$$V_{EQ} = \{z, m, x, y\}$$

Вспомогательные определения

- Функция означивания переменной:

$$\boxed{\eta : V_P \to D_P}, \forall v \in V_P, \eta(v) \in D_P^v$$

```
bool z;
mtype {M1,M2} m = M1;

proctype EQ(byte x, byte y)
{
  if
  :: (x == y) -> z = true
  :: else -> z = false
  fi
}
```

Примеры:

$$\eta(m) = \mathrm{M1} \in mtype$$

$$\eta(x) = 3 \in byte$$

$$\eta(z) = true \in bool$$

Вспомогательные определения

- $\boxed{Cond(V_P)}$ – набор булевых условий над V_P
  - – формулы пропозициональной логики

$$\left(p1 \wedge p2 \vee p3 \right)$$

  - – используются высказывания вида $\overline{x} \in \overline{X}$

$$\left(\begin{array}{c} p1 \equiv -3 < x \leq 5 \\ p2 \equiv m = M2 \\ p3 \equiv y < 2 * x \end{array} \right)$$

Вспомогательные определения

- Эффект операторов формализуется как отображение

$$\boxed{Effect : Act \times Eval(Var) \to Eval(Var)}$$

```
…
x = 17;
if
:: y = -2
:: y = 3
fi;
…
x = y + 5;
…
```

$Пусть\ \alpha \equiv x = y + 5,$

$\eta_{1,2}(x) = 17, \eta_1(y) = -2, \eta_2(y) = 3$

Тогда

$Effect(\alpha, \eta_1)(x) = \eta_1(y) + 5 = 3$

$Effect(\alpha, \eta_1)(y) = \eta_1(y) = -2$

$Effect(\alpha, \eta_2)(x) = \eta_2(y) + 5 = 8$

$Effect(\alpha, \eta_2)(y) = \eta_2(y) = 3$

Графы программ
(статическая семантика)

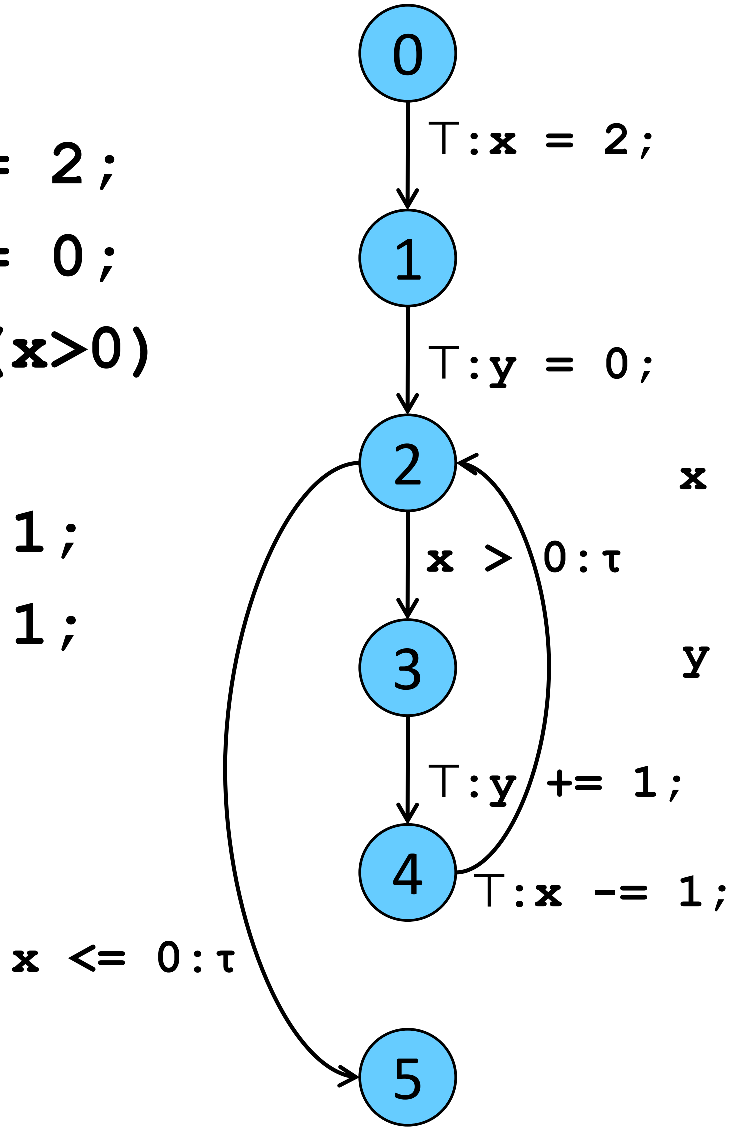
$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

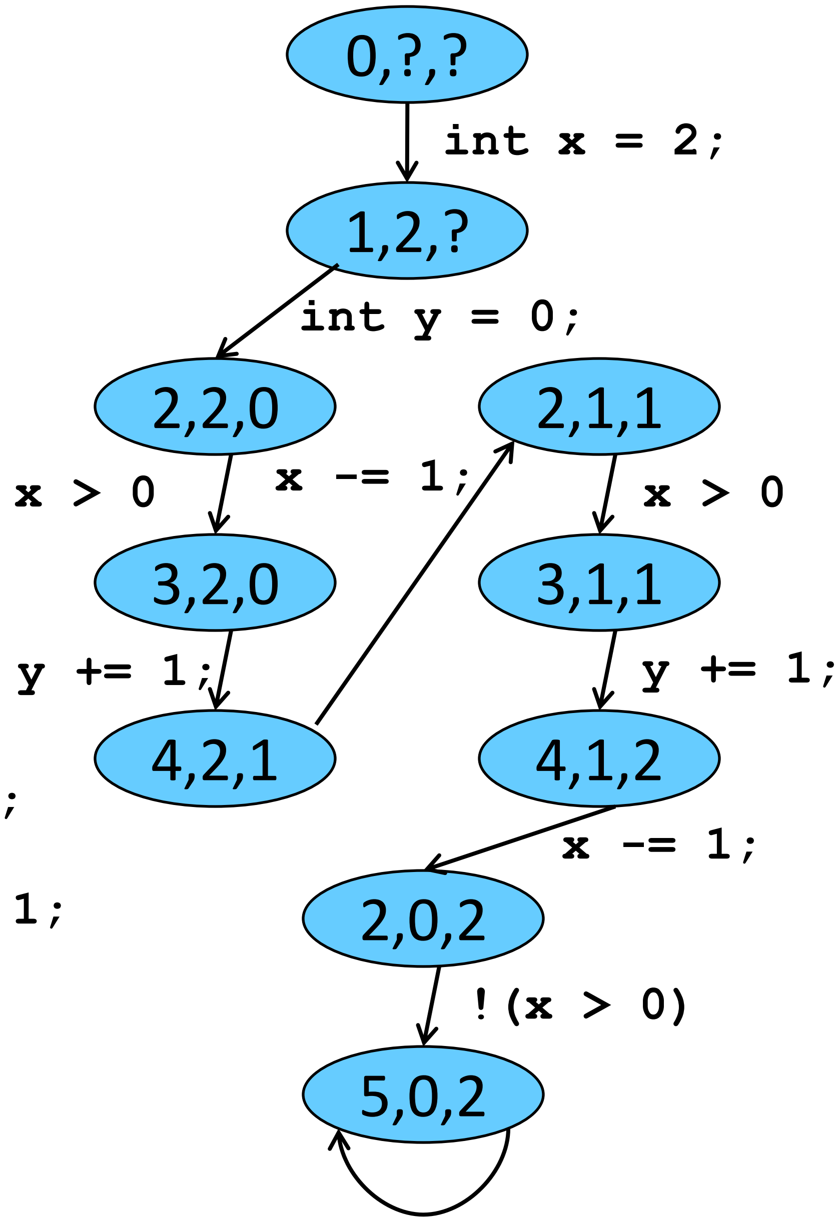
- Loc – множество *точек*, исходные точки $Loc_0 \subseteq Loc$,
- Act – множество действий,
- $Effect : Act \times Eval(Var) \rightarrow Eval(Var)$ – функция *эффекта*,
- $\rightarrow \subseteq Loc \times (Cond(V_P) \times Act) \times Loc$ – отношение перехода,
- $g_0 \in Cond(V_P)$ – начальное условие,

$$\textit{Нотация}: \quad l \xrightarrow{g:\alpha} l' \quad \textit{обозначает} \quad \langle l, g, \alpha, l' \rangle \in \rightarrow$$

Пример графа программы

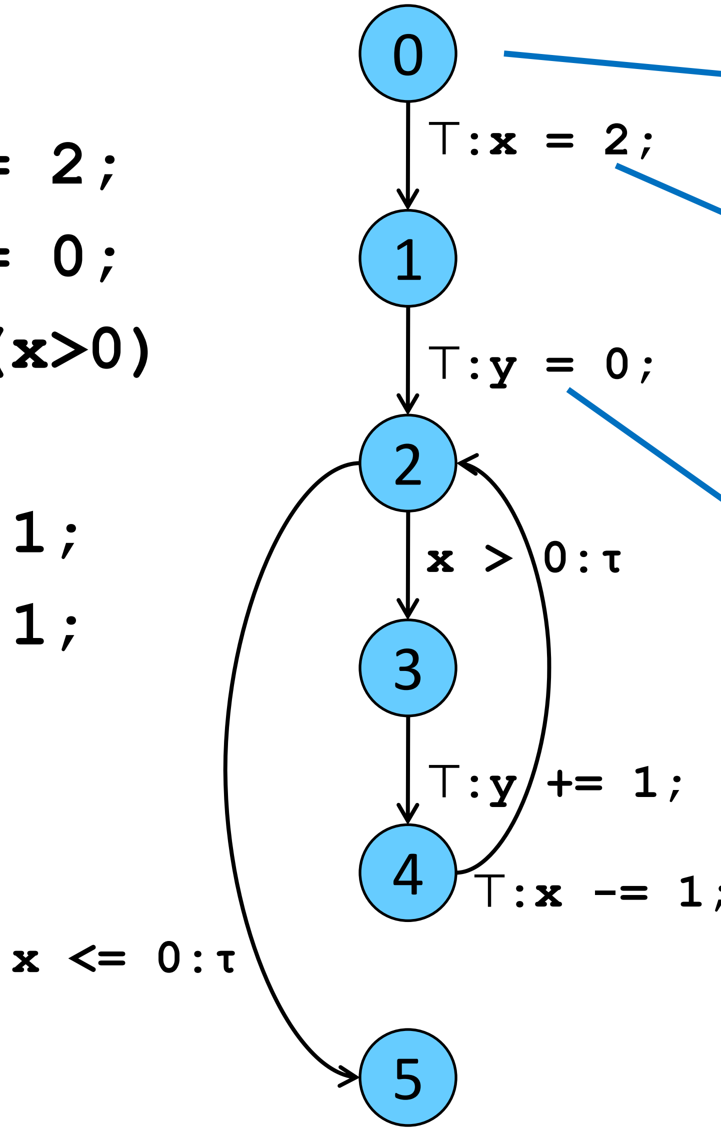
```
0: int x = 2;
1: int y = 0;
2: while (x>0)
   {
3:    y += 1;
4:    x -= 1;
   }
5:
```





Пример графа программы

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



Loc

{0, 1, 2, 3, 4, 5}

Act

{“x=2”, “y=0”, “y+=1”, “x-=1”, τ}

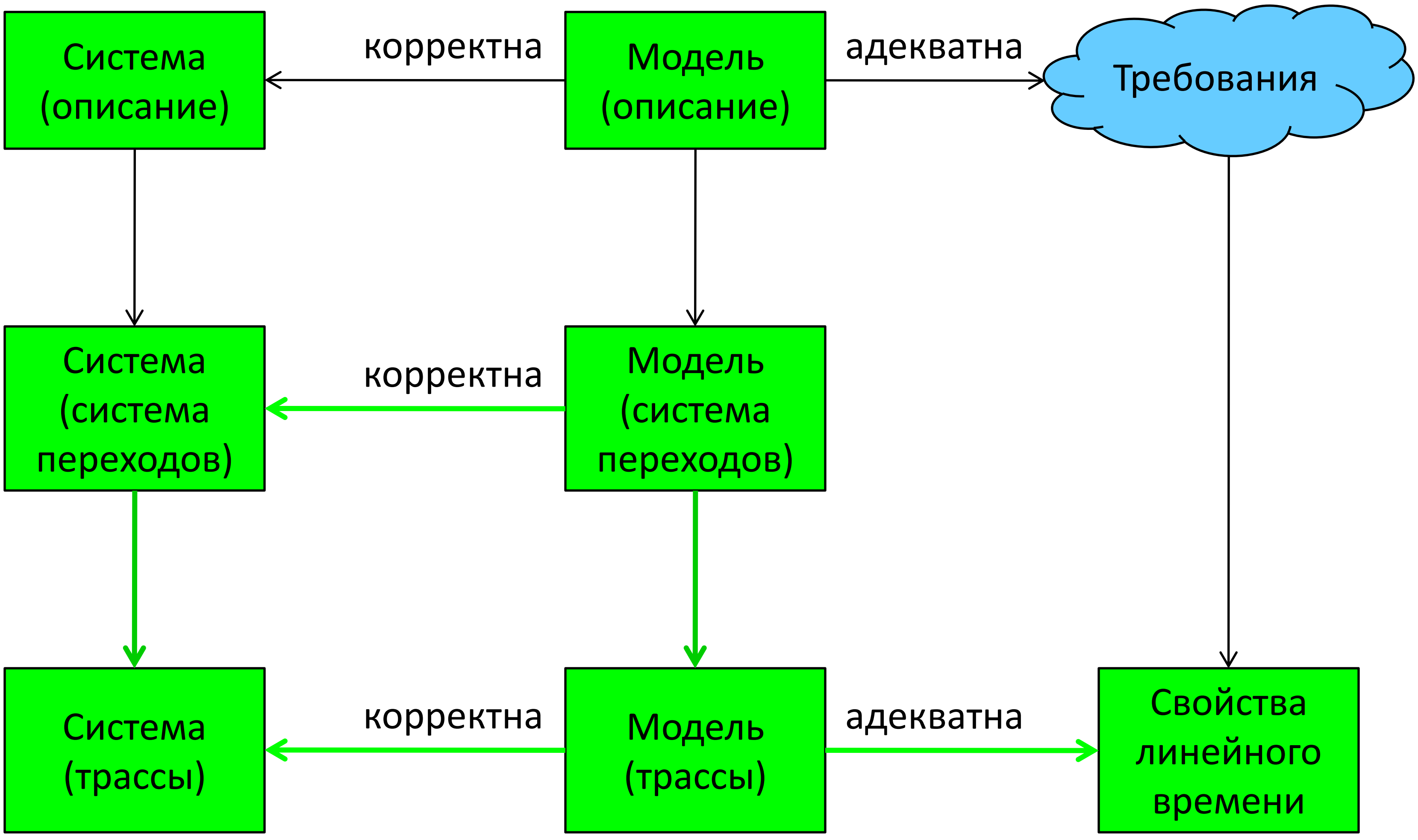
Effect

$$\rightarrow$$

$$Loc_0 = "0"$$

$$g_0 = (x = \#) \wedge (y = \#)$$

Схема понятий



Как из PG получить TS?

- Основная идея – *раскрутка*
  - состояние: точка *l* + значение данных *η*
  - начальное состояние: начальная точка + все значения данных, удовлетворяющие g_0;
- Атомарные высказывания и разметка:
  - высказывания вида: “в *l*” и “$x \in D$”, где $D \subseteq dom(x)$;
  - состояние <*l*,*η*> размечается высказыванием “в *l*” и всеми высказываниями, истинными в *η;*
- Если $l \xrightarrow{g:\alpha} l'$ и g истинно в *η,* то

$$\langle l, \eta \rangle \xrightarrow{\alpha} \langle l', Effect(\alpha, \eta) \rangle$$

Структурированная операционная семантика

$$\text{Нотация} \quad \frac{\text{посылка}}{\text{следствие}} \quad \text{означает:}$$

- если посылка истинная, то *следствие* также истинно;
- это т.н. *правило вывода*;
- если посылка тождественно равна истине, то следствие – *аксиома*.

Системы переходов графов программ

- Система переходов TS(PG) графа программы

$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

над переменными V_P описывается сигнатурой

$$TS(PG) = \langle S, Act, \rightarrow, I, AP, L \rangle, \text{ где}$$

$$S = Loc \times Eval(V_P)$$

«состояние: точка *l* + значение данных *η*»

Системы переходов графов программ

- Система переходов TS(PG) графа программы

$$PG = \langle Loc, Act, Effect, \to, Loc_0, g_0 \rangle$$

над переменными V_P описывается сигнатурой

$$TS(PG) = \langle S, Act, \to, I, AP, L \rangle, \textit{где}$$

$$I = \{\langle l, \eta \rangle \mid l \in Loc_0, \eta \models g_0\}$$

«начальное состояние: начальная точка + все значения данных, удовлетворяющие g_0»

Системы переходов графов программ

- Система переходов TS(PG) графа программы

$$PG = \langle Loc, Act, Effect, \to, Loc_0, g_0 \rangle$$

над переменными V_P описывается сигнатурой

$$TS(PG) = \langle S, Act, \to, I, AP, L \rangle, \text{где}$$

$$AP = Loc \cup Cond(V_P)$$

«высказывания вида: “в *l*” и “x∈D”, где D⊆dom(x);»

Системы переходов графов программ

- Система переходов TS(PG) графа программы

$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

над переменными V_P описывается сигнатурой

$$TS(PG) = \langle S, Act, \rightarrow, I, AP, L \rangle, \text{ где}$$

$$L(\langle l, \eta \rangle) = \{l\} \cup \{g \in Cond(V_P) \mid \eta \models g\}$$

«состояние <*l*,*η*> размечается высказыванием “в *l*” и всеми высказываниями, истинными в *η*»

Системы переходов графов программ

- Система переходов TS(PG) графа программы

$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

над переменными V_P описывается сигнатурой

$$TS(PG) = \langle S, Act, \rightarrow, I, AP, L \rangle, \text{ где}$$

$$\rightarrow \subseteq S \times Act \times S \text{ задано правилом } \quad \frac{l \xrightarrow{g:\alpha} l' \wedge \eta \models g}{\langle l, \eta \rangle \xrightarrow{\alpha} \langle l', Effect(\alpha, \eta) \rangle}$$

«Если $l \xrightarrow{g:\alpha} l'$ и g истинно в *η*, то

$\langle l, \eta \rangle \xrightarrow{\alpha} \langle l', Effect(\alpha, \eta) \rangle$ »

Системы переходов графов программ

- Система переходов TS(PG) графа программы

$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

над переменными V_P описывается сигнатурой

$$TS(PG) = \langle S, Act, \rightarrow, I, AP, L \rangle, \text{ где}$$

- $S = Loc \times Eval(V_P)$
- $\rightarrow \subseteq S \times Act \times S$ *задано правилом* $\dfrac{l \xrightarrow{g:\alpha} l' \wedge n \models g}{\langle l, \eta \rangle \xrightarrow{\alpha} \langle l', Effect(\alpha, \eta) \rangle}$
- $I = \{ \langle l, \eta \rangle \mid l \in Loc_0, n \models g_0 \}$
- $AP = Loc \cup Cond(V_P)$
- $L(\langle l, \eta \rangle) = \{l\} \cup \{ g \in Cond(V_P) \mid n \models g \}$

Параллелизм

Чередование (интерливинг)
(напоминание)

- Абстрагируемся от того, что система состоит из множества компонентов;
- Действия независимых компонентов чередуются:
  - доступен один процессор, выполнение одного действия блокирует другие;
- Порядок выполнения процессов неизвестен
  - Возможные порядки выполнения независимых процессов P и Q:

```
P Q P Q P Q Q Q P . . .
P P Q P P Q P P Q . . .
P Q P P Q P P P Q . . .
```

Чередование (интерливинг)

- Обоснование чередования:

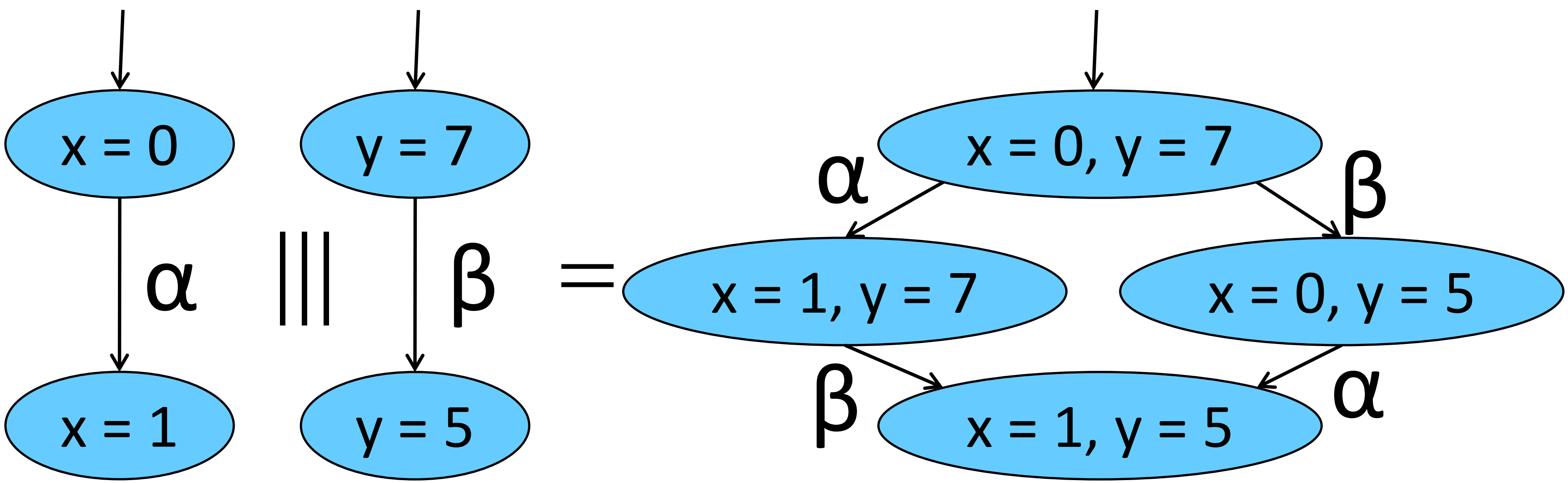
  эффект от параллельного выполнения независимых действий α и β равен эффекту от последовательного выполнения действий α и β в произвольном порядке;
- Символьная запись:

$$Effect(\alpha \,|||\, \beta, \eta) = Effect((\alpha;\beta) + (\beta;\alpha), \eta)$$

  – «$|||$» – бинарный оператор чередования,
  – «;» – оператор последовательного выполнения,
  – «+» – оператор недетерминированного выбора.

Чередование (интерливинг)

$$\underbrace{x = x + 1}_{\alpha} \;|||\; \underbrace{y = y - 2}_{\beta}$$



Чередование систем переходов

- Пусть $TS_i = \langle S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i \rangle, i = 1{,}2$

  – две системы переходов
- Система переходов $TS_1 \,|||\, TS_2$ определяется как

$$\langle S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L \rangle, \text{ где}$$

- $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$,
- отношение перехода $\rightarrow$ определяется правилами:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad и \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

Чередование графов программ

- Для графов программ PG_1 (над V_1) и PG_2 (над V_2) без разделяемых переменных (т.е. $V_1 \cap V_2 = \varnothing$), формула

$$TS(PG_1) \,|||\, TS(PG_2)$$

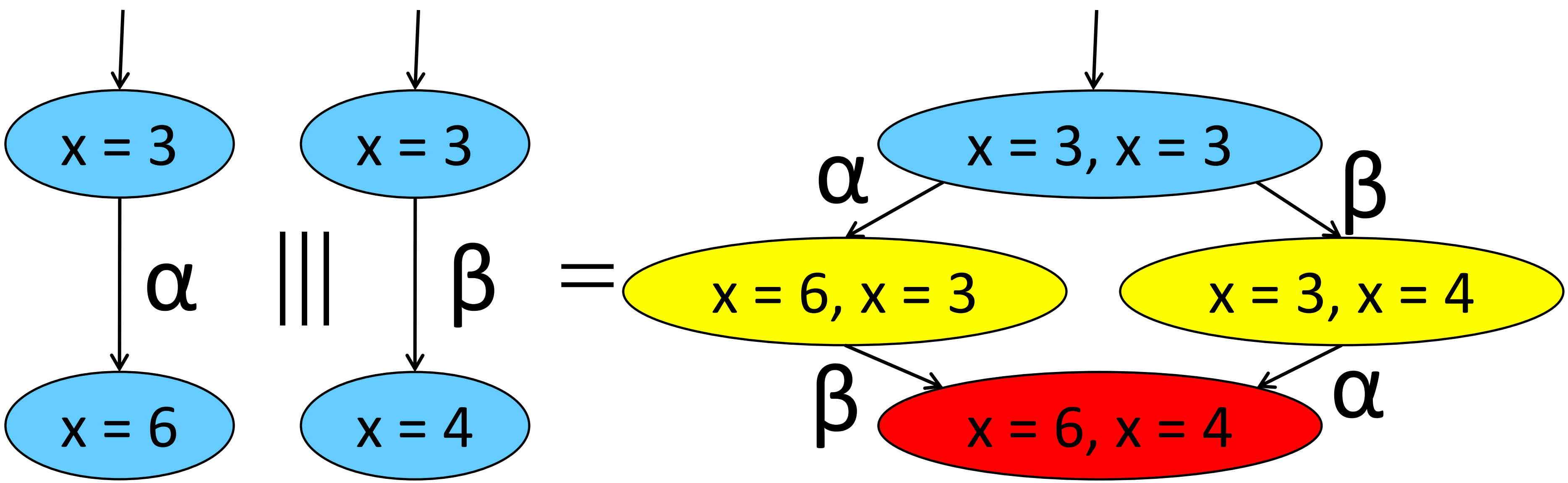
достоверно описывает параллельную композицию PG_1 и PG_2

а если разделяемые переменные есть?

Разделяемые переменные

(пытаемся сначала раскручивать, затем чередовать)

$$\underbrace{x = x * 2}_{\alpha} \quad ||| \quad \underbrace{x = x + 1}_{\beta}$$ (в начале x = 3)



Чередование графов программ

- Пусть $PG_i = \langle Loc_i, Act_i, Effect_i, \rightarrow_i, Loc_{0,i}, g_{0,i} \rangle, i = 1,2$
- Граф $PG_1 \,|||\, PG_2$ над $V_1 \cup V_2$ определяется так

$$\langle Loc_1 \times Loc_2, Act_1 \cup Act_2, \rightarrow, Effect, Loc_{0,1} \times Loc_{0,2}, g_{0,1} \wedge g_{0,2} \rangle$$

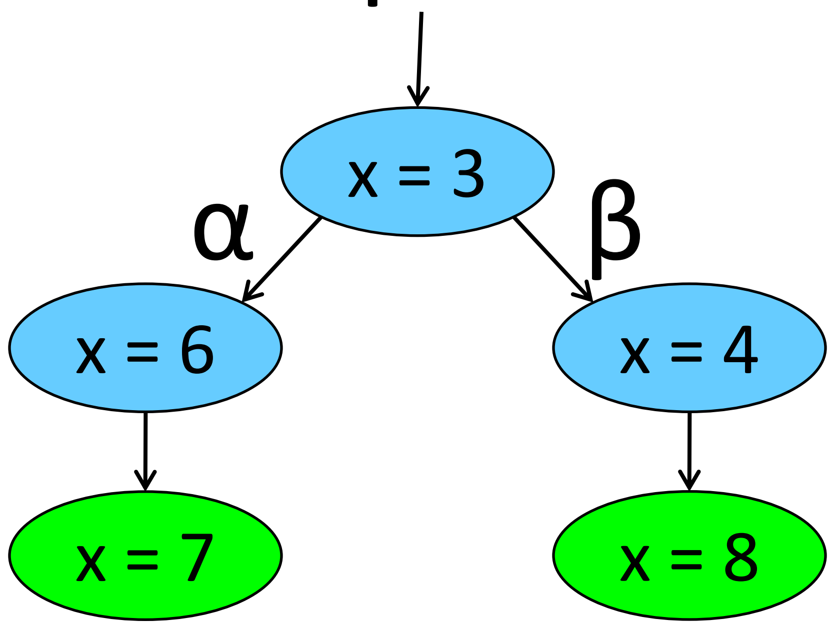
где отношение перехода $\rightarrow$ определяется правилами

$$\frac{l_1 \xrightarrow{g:\alpha}_1 l_1'}{\langle l_1, l_2 \rangle \xrightarrow{g:\alpha} \langle l_1', l_2 \rangle} \quad и \quad \frac{l_2 \xrightarrow{g:\alpha}_2 l_2'}{\langle l_1, l_2 \rangle \xrightarrow{g:\alpha} \langle l_1, l_2' \rangle}$$

$$и\ Effect(\alpha, \eta) = Effect_i(\alpha, \eta),\ если\ \alpha \in Act_i.$$

Пример

$$\underbrace{x = x * 2}_{\alpha} \;|||\; \underbrace{x = x + 1}_{\beta}$$ (в начале x = 3)



$$\boxed{TS(PG_1) \,|||\, TS(PG_2) \neq TS(PG_1 \,|||\, PG_2)}$$

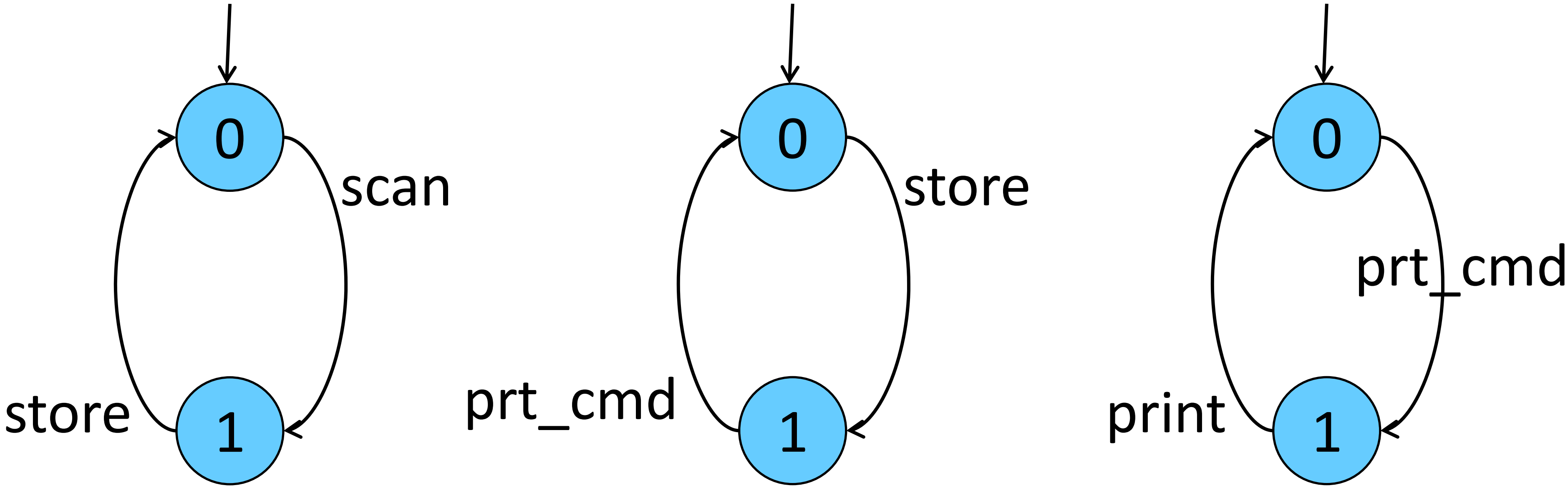
Параллелизм и рандеву

- Распределённые программы выполняются параллельно;
- Для моделирования взаимодействия необходимо придумать подходящий механизм;
- В распределённой программе разделяемых переменных нет;
- **Передача сообщений**:
  - синхронная передача сообщений (рандеву),
  - асинхронная передача сообщений (каналы).

Рандеву

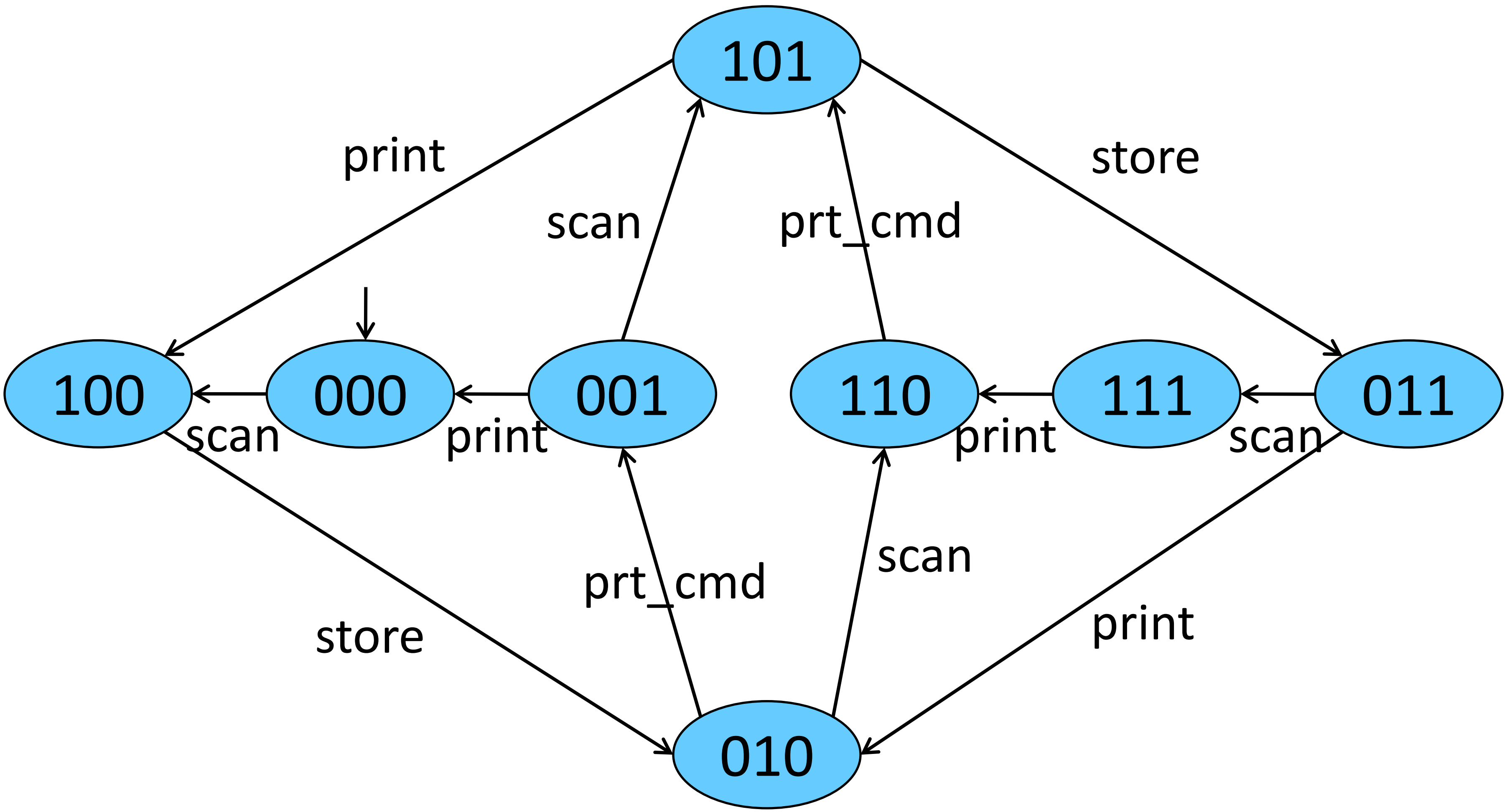
- Распределённые процессы, взаимодействующие при помощи синхронного обмена сообщениями
  - процессы вместе выполняют синхронизированные действия,
  - взаимодействие обоих процессов происходит одновременно,
  - происходит “рукопожатие”;
- Абстрагируемся от передаваемой информации;
- *H* – набор синхронизированных действий:
  - действия, не принадлежащие *H*, независимы и чередуются,
  - действия из *H* должны быть синхронизированы.

Пример рандеву



$$A \|_H B \|_H \text{Printer}$$

Параллельная композиция



Рандеву систем переходов

- Пусть $TS_i = \langle S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i \rangle, i = 1{,}2$ *и* $H \subseteq Act_1 \cap Act_2$
- Тогда $TS_1 \parallel_H TS_2$ определяется как

$$\langle S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L \rangle, \ \text{где}$$

- $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$,
- отношение перехода $\rightarrow$ определяется правилами:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \text{и} \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle} \ \text{для } \alpha \notin H$$

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle} \quad \text{для } \alpha \in H$$

{ интерливинг }

{ рандеву }

Заметим, что $TS_1 \parallel_H TS_2 = TS_2 \parallel_H TS_1$, *но* $(TS_1 \parallel_{H_1} TS_2) \parallel_{H_2} TS_3 \neq TS_1 \parallel_{H_1} (TS_2 \parallel_{H_2} TS_3)$

Попарное рандеву

- Пусть $TS_1 \| \ldots \| TS_n\ \text{для}\ H_{i,j} \subseteq Act_i \cap Act_j\ \text{с}\ H_{i,j} \cap Act_k = \varnothing\ \text{для}\ k \neq i, j$
- Пространство состояний $TS_1 \| \ldots \| TS_n$ – это декартово произведение множеств состояний TS_i

- $\text{для}\ \alpha \in Act_i \setminus (\bigcup_{0<j\leq n \wedge i \neq j} H_{i,j})\ \text{и}\ 0 < i \leq n$

$$\frac{s_i \xrightarrow{\alpha}_i s_i'}{\left\langle s_1, \ldots, s_i, \ldots, s_n \right\rangle \xrightarrow{\alpha} \left\langle s_1, \ldots, s_i', \ldots, s_n \right\rangle}$$

- $\text{для}\ \alpha \in H_{i,j}\ \text{и}\ 0 < i < j \leq n$

$$\frac{s_i \xrightarrow{\alpha}_i s_i' \wedge s_j \xrightarrow{\alpha}_j s_j'}{\left\langle s_1, \ldots, s_i, \ldots, s_j, \ldots, s_n \right\rangle \xrightarrow{\alpha} \left\langle s_1, \ldots, s_i', \ldots, s_j', \ldots, s_n \right\rangle}$$

Синхронный параллелизм

$$Пусть\ TS_i = \langle S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i \rangle, i = 1{,}2\ и$$

$$\exists Act \times Act \rightarrow Act, (\alpha, \beta) \rightarrow \alpha * \beta$$

$$TS_1 * TS_2 = \langle S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L \rangle$$

где $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ и $\rightarrow$ определяется так:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\beta}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha * \beta} \langle s_1', s_2' \rangle}$$

chan ch[0]={int}
ch!M1 – α
ch?m – β

Асинхронный параллелизм

- Процессы взаимодействуют при помощи каналов ($c \in Chan$),
- Каналы типизированы по передаваемым сообщениям ($dom(c)$),
- Каналы – FIFO буфера, хранящие сообщения (соотв. типа),
- Емкость канала $cap(c)$ – *максимальное число* сообщений, которое может буферизовать канал,
- $cap(c) = 0$ – взаимодействие сводится к рандеву.

Каналы

- Процесс P_i = граф программы PG_i + действия обмена сообщениями *Comm:*

  c!v – передача значения v по каналу c

  c?x – приём сообщения по каналу c и присвоение его переменной x

$$Comm = \{c!v, c?x \mid c \in Chan, v \in dom(c), x \in V_P, dom(x) \subseteq dom(c)\}$$

- Отправка и приём сообщений:
  - c!v помещает значение v в конец буфера c (если c не полон),
  - c?x забирает первый элемент буфера и присваивает его значение x (если c не пуст),
  - cap(c) = 0 – у канала *c* нет буфера, отправка и приём производятся одновременно (*рандеву),*
  - cap(c) > 0 – отправка и приём никогда **не происходят одновременно** (асинхронная передача сообщений).

Системы с каналами

- Граф программы *PG* над *(Var, Chan)* задаётся сигнатурой

$$PG = \langle Loc, Act, Effect, \rightarrow, Loc_0, g_0 \rangle$$

где

$$\rightarrow \subseteq Loc \times (Cond(V_P) \times Act) \times Loc \quad \cup \quad Loc \times Comm \times Loc$$

- Система с каналами *CS* над $(\bigcup_{0<i\leq n} Var_i, Chan)$ задаётся как

$$CS = [PG_1 \mid \ldots \mid PG_n]$$

где PG_i – графы программ над (Var_i, Chan)

Взаимодействие

- Рандеву
  - если *cap(c)=0*, то процесс P_i может выполнить $l_i \xrightarrow{c!v} l_i'$
  - … только если P_j может выполнить $l_j \xrightarrow{c?x} l_j'$
  - эффект соответствует распределённому $x = v$
- Асинхронная передача сообщений
  - если *cap(c) > 0*, то процесс P_i может выполнить $l_i \xrightarrow{c!v} l_i'$
  - … только если в *c* хранится меньше *cap(c)* сообщений,
  - P_j может выполнить $l_j \xrightarrow{c?x} l_j'$, только если *c* не пуст
  - … после чего первый элемент *v* извлекается из *c* и присваивается *x* (атомарно)

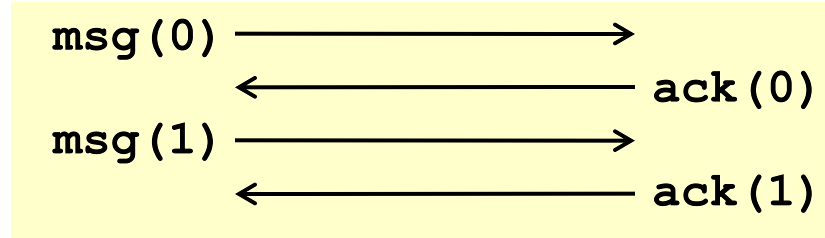
	Выполнимо, если	**Эффект**
c!v	с не полон	Enqueue(c,v)
c?x	с не пуст	$\langle$x=Front(c);Dequeue(c); $\rangle$

Модель на Promela

```
mtype = {msg, ack};
chan s_r = [2] of {mtype, bit};
chan r_s = [2] of {mtype, bit};

active proctype sender()
{ bit seqno;
  do
  :: s_r!msg,seqno ->
       if
       :: r_s?ack,eval(seqno) ->
           seqno = 1 - seqno;
       :: r_s?ack,eval(1-seqno)
       fi
  od
}

active proctype receiver()
{ bit expect, seqno;
  do
  :: s_r?msg,seqno ->
       r_s!ack,seqno;
       if
       :: seqno == expect;
          expect = 1 - expect
       ::else
       fi
  od
```

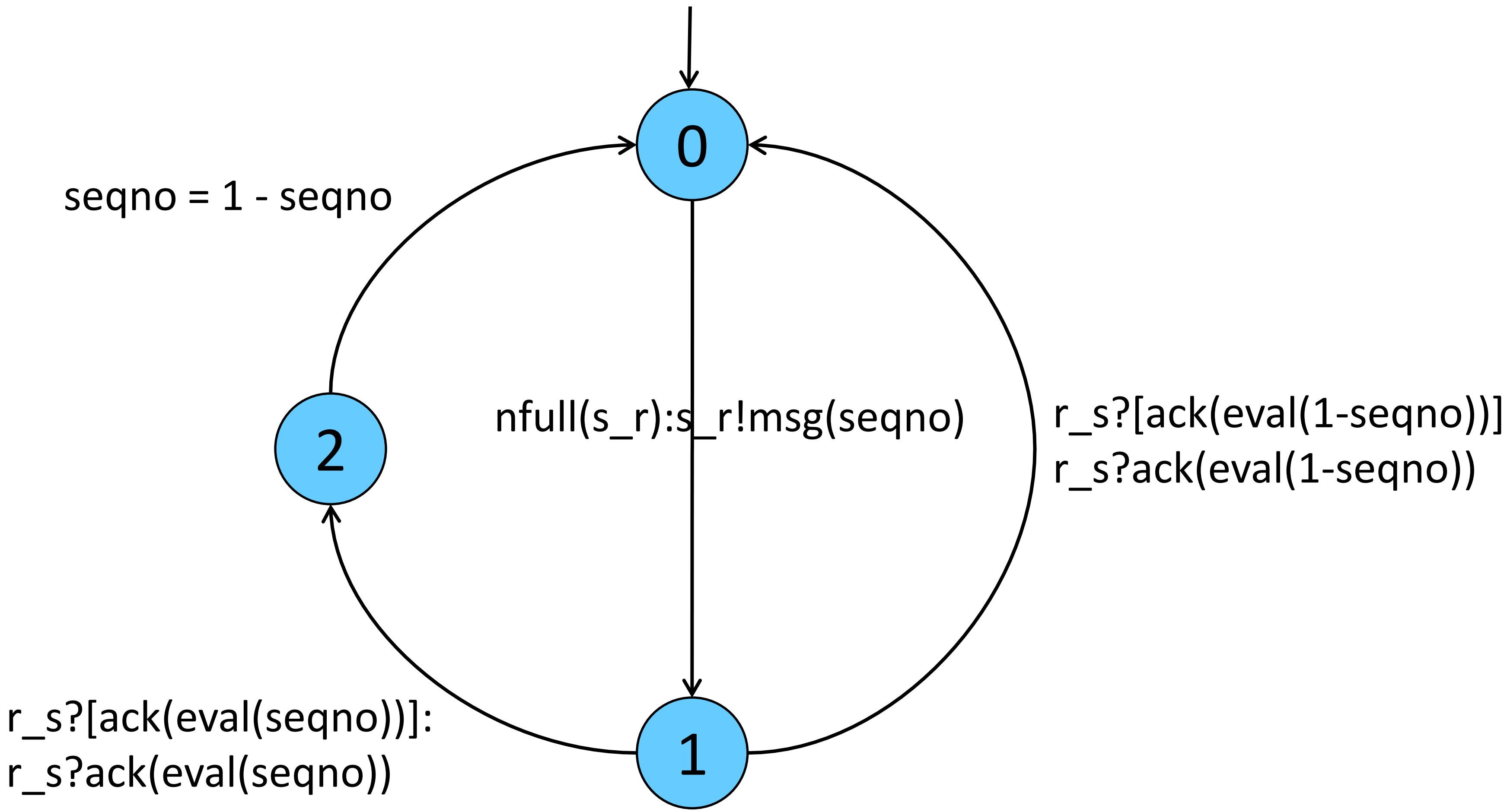


Считываем новое сообщение

Сохраняем сообщение

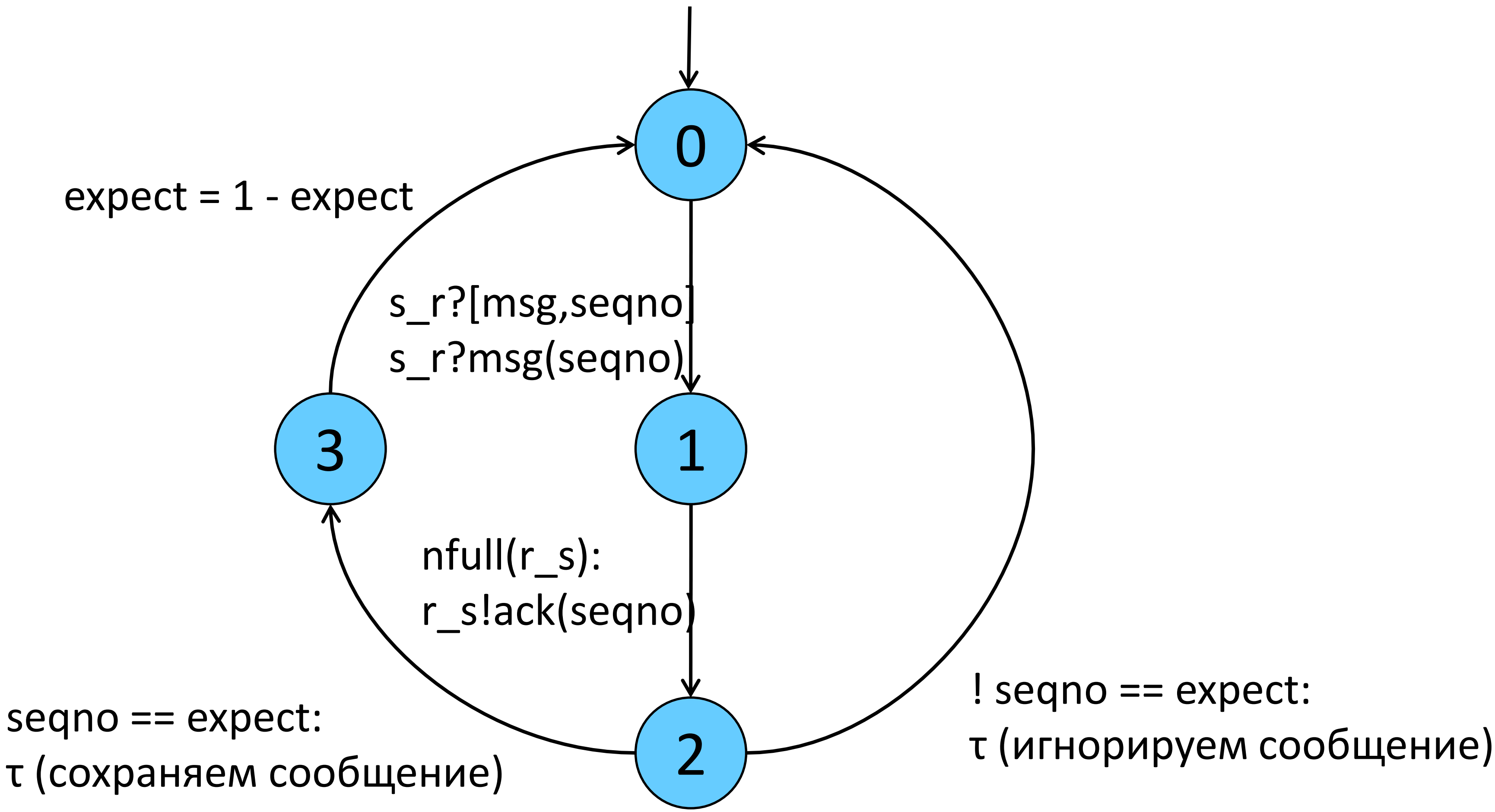
Игнорируем сообщение

Alternating Bit Protocol
(отправитель)



Alternating Bit Protocol

(получатель)



Значение канала

- Оценка ξ значения канала *c* – это
  - отображение канала на последовательность значений:

$$\xi : Chan \rightarrow dom(c)^*$$

  - такое, что длина последовательности не превосходит ёмкости канала:

$$len(\xi(c)) \leq cap(c)$$

  - при этом $\xi(c) = v_1 v_2 ... v_k$ означает, что v_1 – верхнее сообщение в буфере

- Исходная оценка $\xi_0(c) = \varepsilon \quad \forall c \in Chan$

Операционная семантика системы с каналами

- Пусть $CS = [PG_1 | \ldots | PG_n]$ – система с каналами над (*Chan*, *Var*), и

$$PG_i = \langle Loc_i, Act_i, Effect_i, \to_i, Loc_{0,i}, g_{0,i} \rangle, i = \overline{1,n}$$

- Система переходов TS(CS) описывается сигнатурой

$$TS(CS) = \langle S, Act, \to, I, AP, L \rangle, \textit{где}$$

- $S = (Loc_1 \times \ldots \times Loc_n) \times Eval(Var) \times Eval(Chan)$
- $Act = (\bigcup_{0<i\leq n} Act_i) \cup \tau$
- $\to$ определяется правилами вывода на сл. слайдах
- $I = \{\langle l_1,\ldots,l_n,\eta,\xi \rangle \mid \forall i(l_i \in Loc_{0,i} \wedge \eta \models g_0) \wedge \forall c(\xi_0(c) = \varepsilon)\}$
- $AP = (\bigcup_{0<i\leq n} Loc_i) \cup Cond(Var)$
- $L(\langle l_1,\ldots,l_n,\eta,\xi \rangle) = \{l_1,\ldots,l_n\} \cup \{g \in Cond(Var) \mid \eta \models g\}$

Правила вывода (I)

- Интерливинг для $\alpha \in Act_i$

$$\frac{l_i \xrightarrow{g:\alpha} l_i' \wedge \eta \models g}{\langle l_1,\ldots,l_i,\ldots,l_n,\eta,\xi\rangle \xrightarrow{\alpha} \langle l_1,\ldots,l_i',\ldots,l_n,\eta',\xi\rangle}$$

- Синхронная передача сообщений через $c \in Chan, cap(c) = 0$

$$\frac{l_i \xrightarrow{c?x} l_i' \wedge l_j \xrightarrow{c!v} l_j' \wedge i \neq j}{\langle l_1,\ldots,l_i,\ldots,l_j,\ldots,l_n,\eta,\xi\rangle \xrightarrow{\tau} \langle l_1,\ldots,l_i',\ldots,l_j',\ldots,l_n,\eta',\xi\rangle}$$

где $\eta = \eta[x = v]$.

Правила вывода (II)

- Асинхронная передача сообщений через $c \in Chan, cap(c) \neq 0$
  - получить значение по каналу *c* и присвоить переменной *x*:

$$\frac{l_i \xrightarrow{c?x} l_i' \wedge len(\xi(c) = k > 0) \wedge \xi(c) = v_1 \dots v_k}{\langle l_1, \dots, l_i, \dots, l_n, \eta, \xi \rangle \xrightarrow{c?x} \langle l_1, \dots, l_i', \dots, l_n, \eta', \xi' \rangle}$$

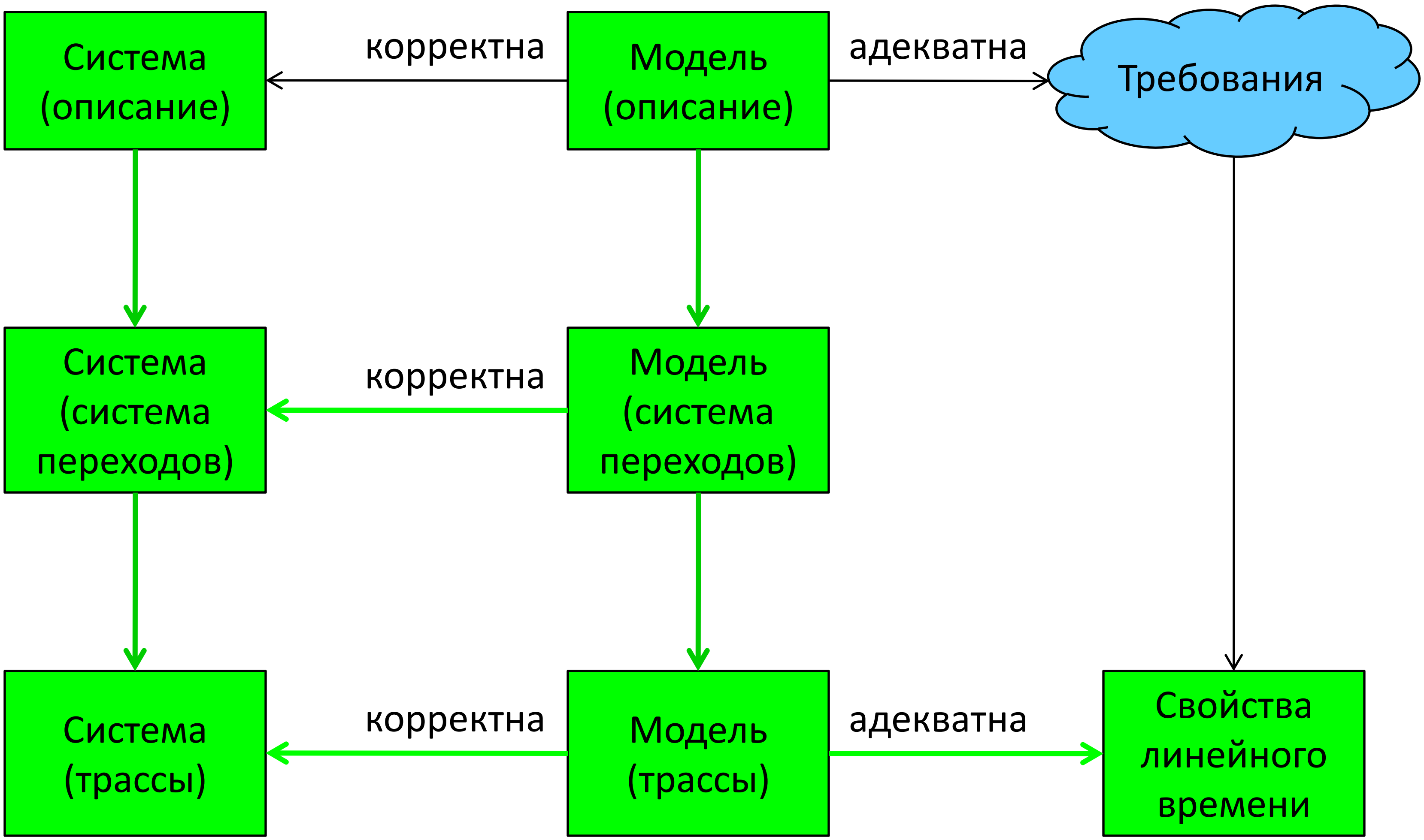
где $\eta = \eta[x = v_1]$ u $\xi' = \xi[c = v_2 \dots v_k]$.

  - передать значение $v \in dom(c)$ по каналу c:

$$\frac{l_i \xrightarrow{c!v} l_i' \wedge len(\xi(c) = k < cap(c)) \wedge \xi(c) = v_1 \dots v_k}{\langle l_1, \dots, l_i, \dots, l_n, \eta, \xi \rangle \xrightarrow{c!v} \langle l_1, \dots, l_i', \dots, l_n, \eta, \xi' \rangle}$$

где $\xi' = \xi[c = v_1 \dots v_k v]$.

Схема понятий



Спасибо за внимание!
Вопросы?