Введение в моделирование и верификацию аппаратных и программных систем

Лекция 2: Системы переходов для моделирования систем

Борис Юрьевич Конев

konev@liverpool.ac.uk

Liverpool University

Октябрь-Ноябрь 2007

Реактивные системы (reactive systems)

- Взаимодействуют с окружением не завершая своей работы
  - Протоколы, операционные системы, драйверы, сенсоры...
- Параллельное исполнение

- Разница с системами, завершающими свою работу, не столь уж велика: всегда можно ввести новое состояние системы “Finish”, войдя в которое, система перестает принимать и посылать сообщения

Возможно, термин *реагирующие* системы лучше бы отражал суть дела

Важные Вопросы при моделировании

- Что есть состояние системы?
- Моделирование параллелизма
- Крупность разбиения

Важные Вопросы при моделировании

- Что есть состояние системы?
  - Представляет информацию о программе/схеме в определенный момент.
  - Значение переменных, регистров, состояние файлов, сообщений...
- Моделирование параллелизма
  - Синхронное исполнение?
  - Асинхронное исполнение?
    - Как синхронизируются компоненты?
- Крупность разбиения
  - Строчка на C?
  - Строчка на ассемблере?
  - Изменение напряжения?

Зафиксируем

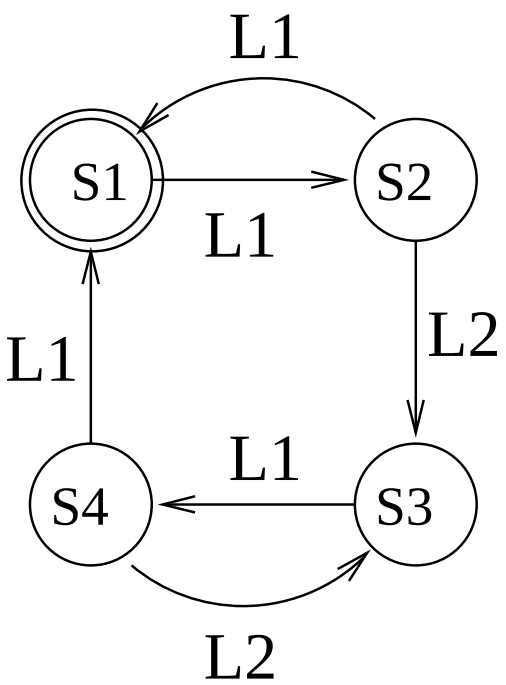
- Аппаратная/программная система может быть в **одном** из конечного множества состояний.
- Каждое состояние системы имеет конечное описание.
- Имеется выделенное начальное состояние системы.
- Система переходит из состояния в состояние.
- В каждый момент времени срабатывает лишь один переход.

Зафиксируем

- Аппаратная/программная система может быть в **одном** из конечного множества состояний.
- Каждое состояние системы имеет конечное описание.
- Имеется выделенное начальное состояние системы.
- Система переходит из состояния в состояние.
- В каждый момент времени срабатывает лишь один переход.

Граф, представляющий поведение системы

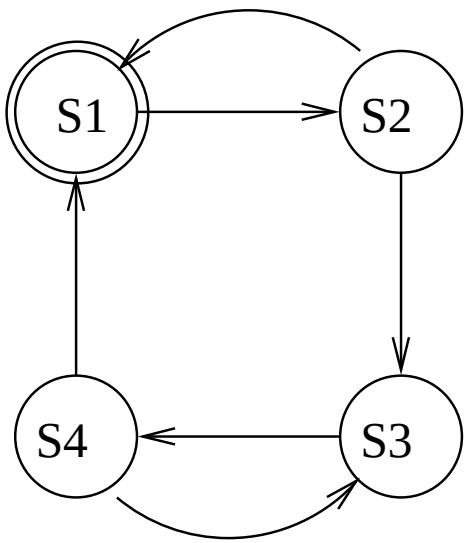
Система переходов (transition system)



Система переходов (Q, E, T, q_0, L)

- Q — конечное множество состояний
- E — конечное множество меток переходов
- $T \subseteq Q \times E \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

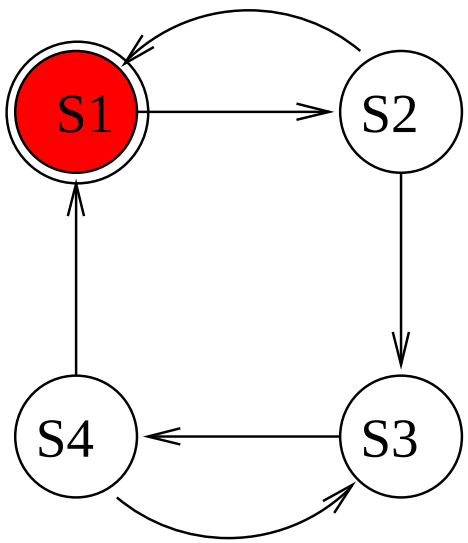
Система переходов (transition system)



Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

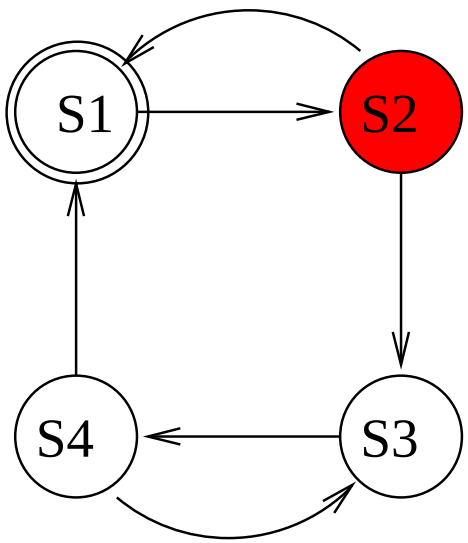
Система переходов (transition system)



Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

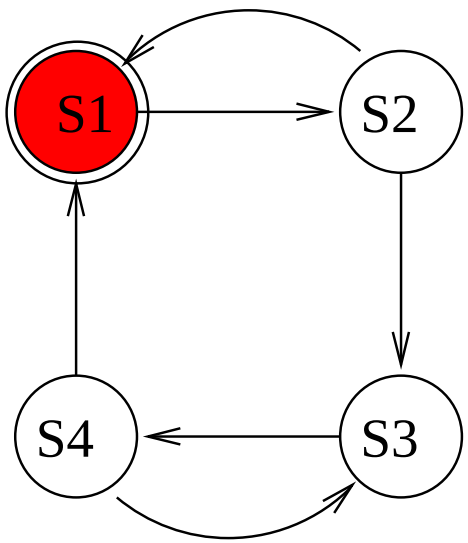
Система переходов (transition system)



Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

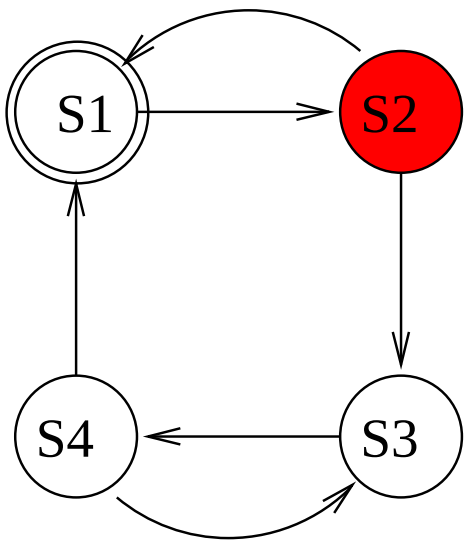
Система переходов (transition system)



Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

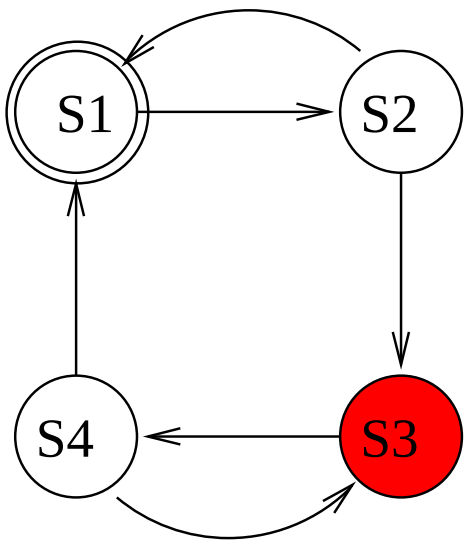
Система переходов (transition system)



Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

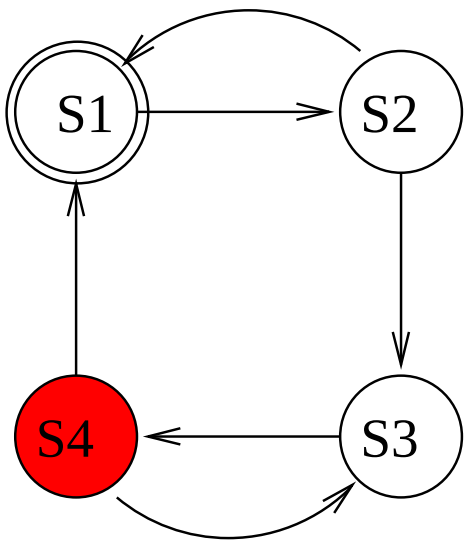
Система переходов (transition system)



Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

Система переходов (transition system)



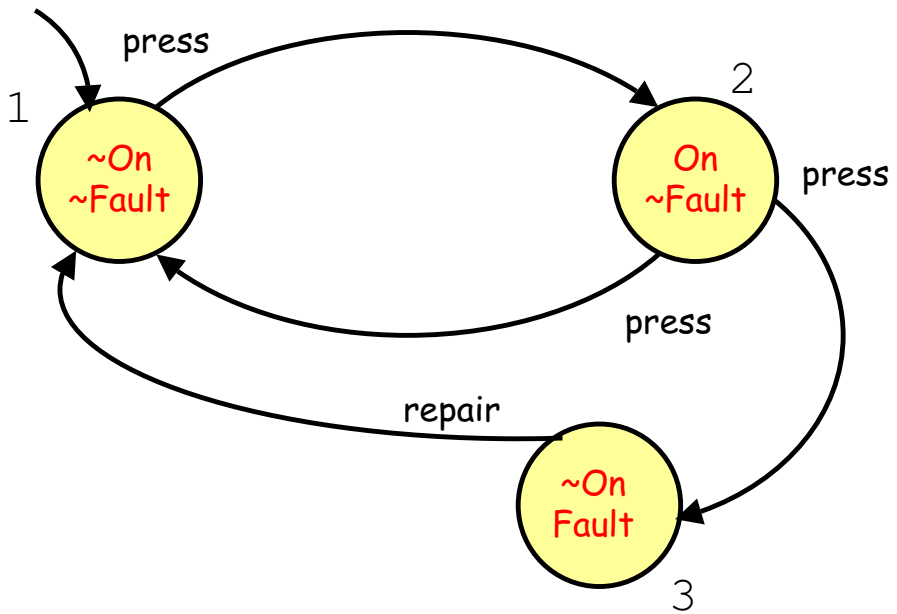
Система переходов (Q, T, q_0, L)

- Q — конечное множество состояний
- $T \subseteq Q \times Q$ — множество переходов
- q_0 — начальное состояние
- $L : Q \to Prop$ — пометка пропозициональными переменными

Пример

Модель выключателя

$$Prop = \{On, Fault\}$$
$$Q = \{1, 2, 3\}$$
$$q_0 = 1$$
$$T = \{(1, press, 2), (2, press, 1), (2, press, 3), (1, repair, 1)\}$$
$$L = \{1 \mapsto \{\}, 2 \mapsto \{On\}, 3 \mapsto \{Fault\}\}$$



Параллельная исполнение

Часто реальные системы имеют несколько компонентов, работающих одновременно

> В каждый момент времени срабатывает лишь один переход

- Независимые компоненты
- Реализация фиксирует (заранее неизвестный) порядок

- Система переходов композиции = декартово произведение систем переходов компонентов (с точностью до синхронизации)

Произведение систем переходов

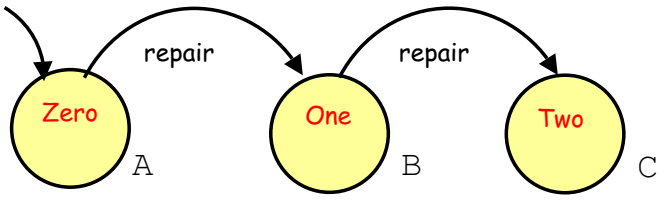
- Пусть $S_1 = (Q_1, E_1, T_1, q_{0,1}, L_1)$ $S_2 = (Q_2, E_2, T_2, q_{0,2}, L_2)$ — системы переходов
- Декартово произведение $S_1 \times S_2 = (Q, E, T, q_0, L)$
  - $Q = Q_1 \times Q_2$;
  - $E = (E_1 \cup \{-\}) \times (E_2 \cup \{-\})$;
  - $T = \left\{ ((q_1, q_2), (e_1, e_2), (q_1', q_2')) \,\middle|\, \begin{array}{l} \text{для } i = 1, 2\text{: } e_i =' - ' \text{ и } q_i' = q_i \text{ или} \\ e_i \neq' - ' \text{ и } (q_i, e_i, q_i') \in T_i \end{array} \right\}$;
  - $q_0 = (q_{0,1}, q_{0,2})$;
  - $L((q_1, q_2)) = L_1(q_2) \cup L_2(q_2)$

  - $'-'$ означает “пропуск хода”

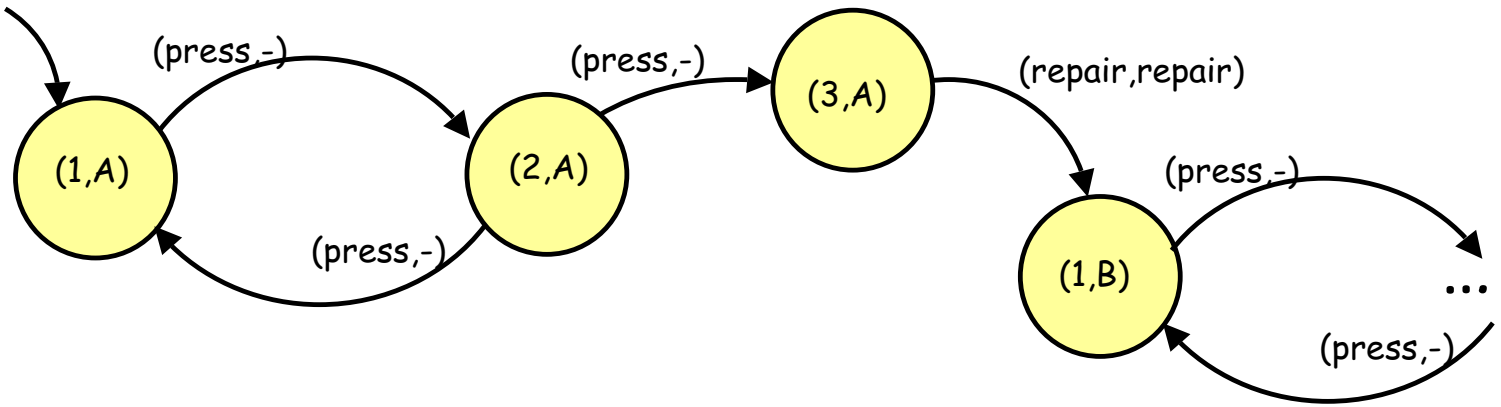
Синхронизированное произведение систем переходов

- Пусть $S_1 = (Q_1, E_1, T_1, q_{0,1}, L_1)$ $S_2 = (Q_2, E_2, T_2, q_{0,2}, L_2)$ — системы переходов
- Пусть $X \subseteq (E_1 \cup \{-\}) \times (E_2 \cup \{-\})$ — синхронизирующее множество
- Синхронизированное произведение $S_1 \times S_2 = (Q, E, T, q_0, L)$:
  - $Q = Q_1 \times Q_2$;
  - $E = (E_1 \cup \{-\}) \times (E_2 \cup \{-\})$;
  - $T = \left\{ ((q_1, q_2), (e_1, e_2), (q_1', q_2')) \middle| \begin{array}{l} (e_1, e_2) \in X \text{ и для } i = 1, 2\text{: } e_i =' - ' \\ \text{и } q_i' = q_i \text{ или } e_i \neq' - ' \text{ и } (q_i, e_i, q_i') \in \\ T_i \end{array} \right\}$;
  - $q_0 = (q_{0,1}, q_{0,2})$;
  - $L((q_1, q_2)) = L_1(q_2) \cup L_2(q_2)$

Пример



- Выключатель + счетчик
- $X = \{(press, -), (repair, repair)\}$:
- Синхронизированное произведение:



Рост числа состояний

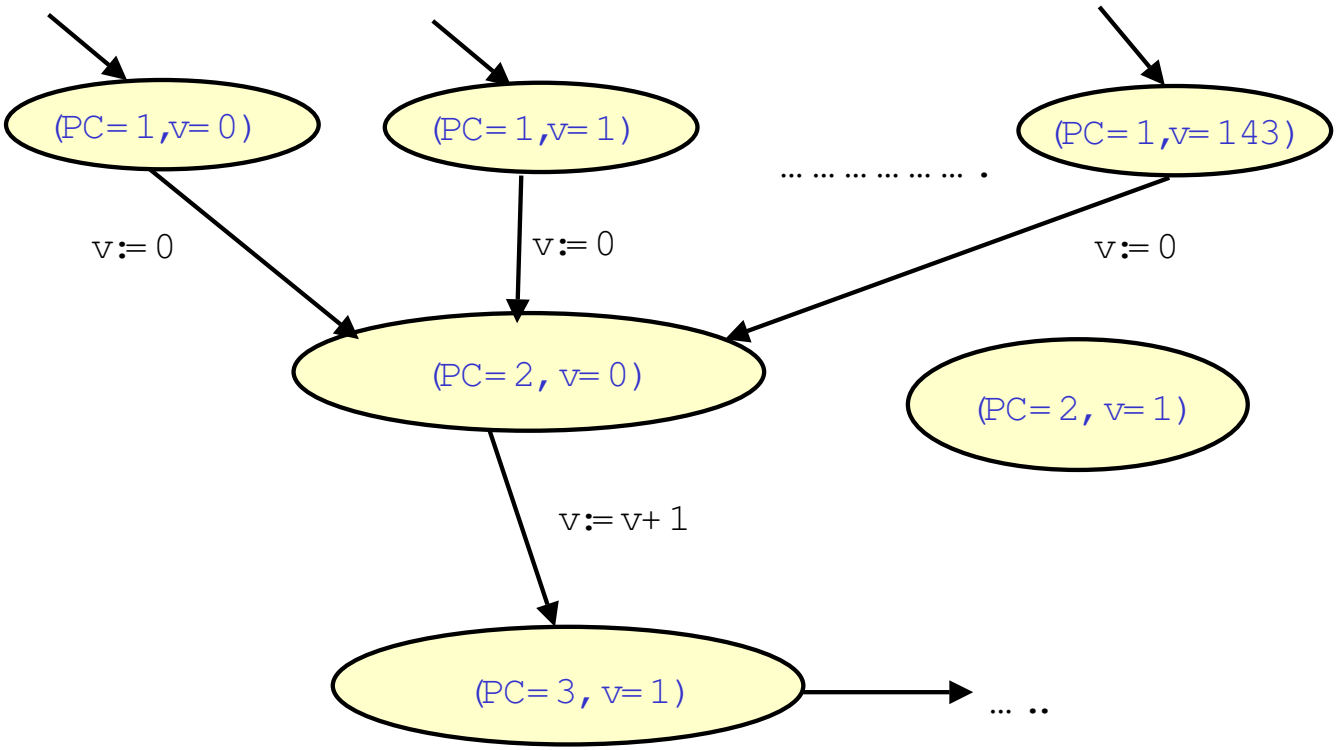
- Выключатель: 3 состояния, 4 метки перехода (включая '$-$')
- Счетчик 3 состояния, 2 метки перехода (включая '$-$')
- Произведение: $3 \times 3 = 9$ состояний, $4 \times 2 = 8$ меток перехода
- Экспоненциальный рост!

Системы переходов как модели программ

- Рассмотрим фрагмент

  $v := 0$

  $v := v + 1$
- Состояния: значения переменных, счетчик команд
  - Тип переменных int, float,. . .



Пример: взаимное исключение

Простейшая реализация взаимного исключения

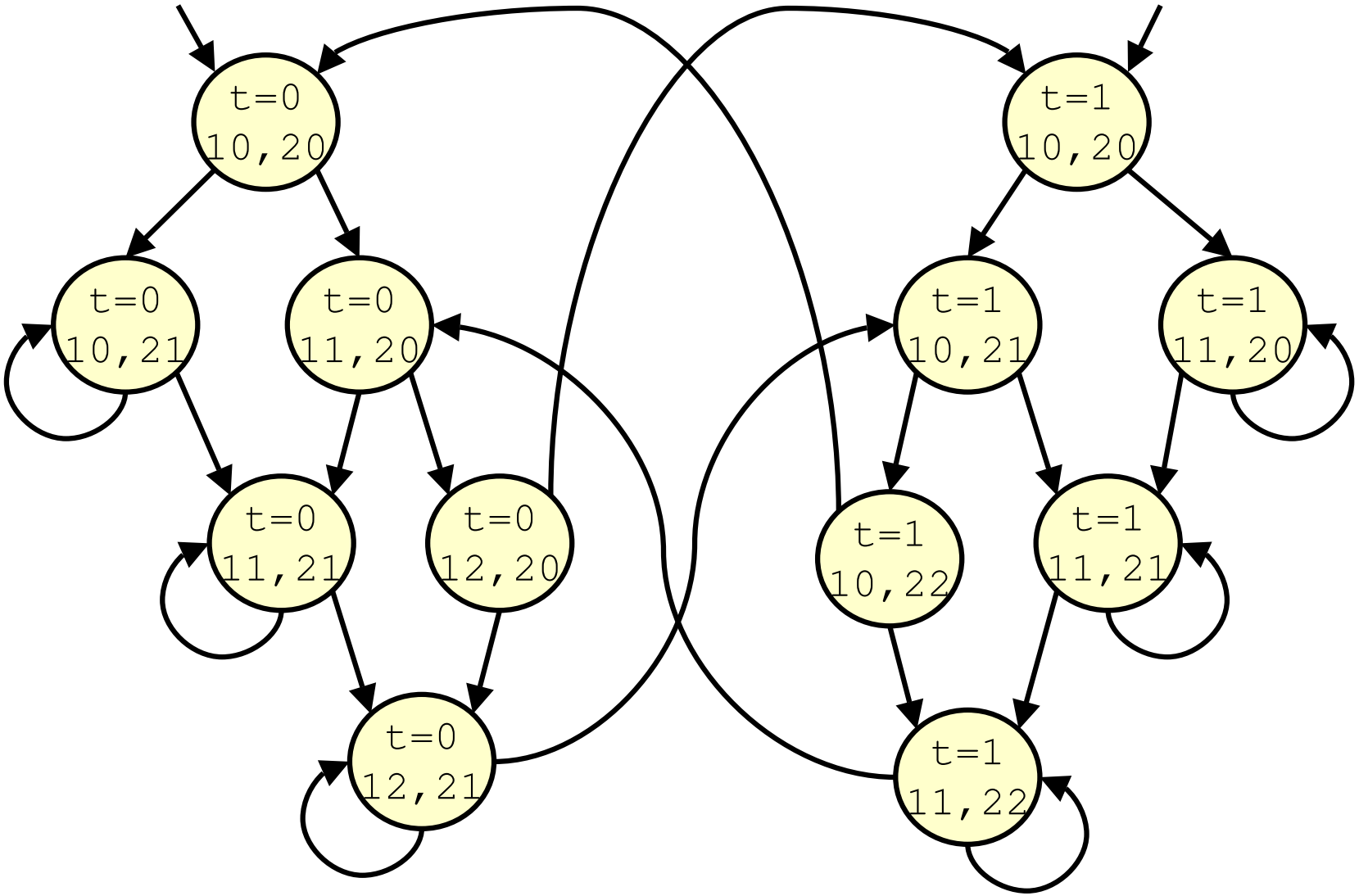
- Два процесса исполняются асинхронно
- Для синхронизации используется разделяемая переменная *turn*
- Не находятся одновременно в критической секции

```
10: while True do
11:   wait (turn = 0)
12:   turn :=1
13: end while;

||

20: while True do
21:   wait (turn = 1)
22:   turn :=0
23: end while;
```

Система переходов для взаимного исключения



Крупность разбиения (1)

Рассмотрим два процесса, исполняемых параллельно на обычном компьютере (одно ядро)

Process A	Process B
$x := x + y;$	$y := y + x;$

- Начальное состояние $x = 2$, $y = 3$
- Значения x и y хранятся в регистрах

Process A	Process B
add r1, r2	add r2, r1

- Возможный результат:

$$x = r1 = 5,\ y = r2 = 8$$
$$x = r1 = 7,\ y = r2 = 5$$

Крупность разбиения (2)

Process A	Process B
$x := x + y;$	$y := y + x;$

- Начальное состояние $x = 2$, $y = 3$
- Значения x и y хранятся в памяти

Process A	Process B
load r1, m1	load r2, m2
add r1, m2	add r2, m1
store r1, m1	store r2, m2

- Возможный результат:

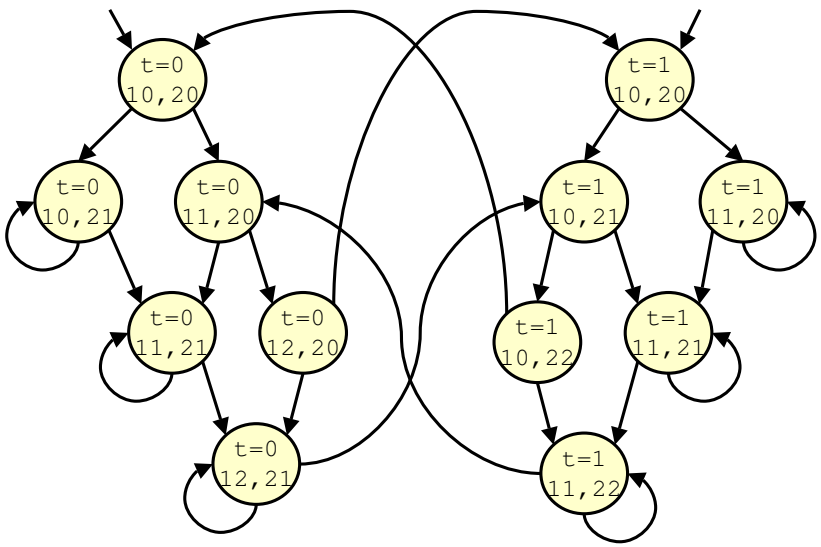
$$\mathrm{x = m1 = 5,\ y = m2 = 8,\ r1 = 5,\ r2 = 8}$$
$$\mathrm{x = m1 = 7,\ y = m2 = 5,\ r1 = 7,\ r2 = 5}$$
$$\mathrm{x = m1 = 5,\ y = m2 = 5,\ r1 = 5,\ r2 = 5}$$

Промежуточные итоги

- Реактивные системы взаимодействуют с окружением не завершая своей работы
- Для моделирования используем системы переходов
  - Состояния
  - Переходы
  - Композиция частей
  - Аппаратные системы
    - (Теоретическая) возможность анализировать программы
- Так как же их верифицировать?
  - Анализ достижимости

Идея

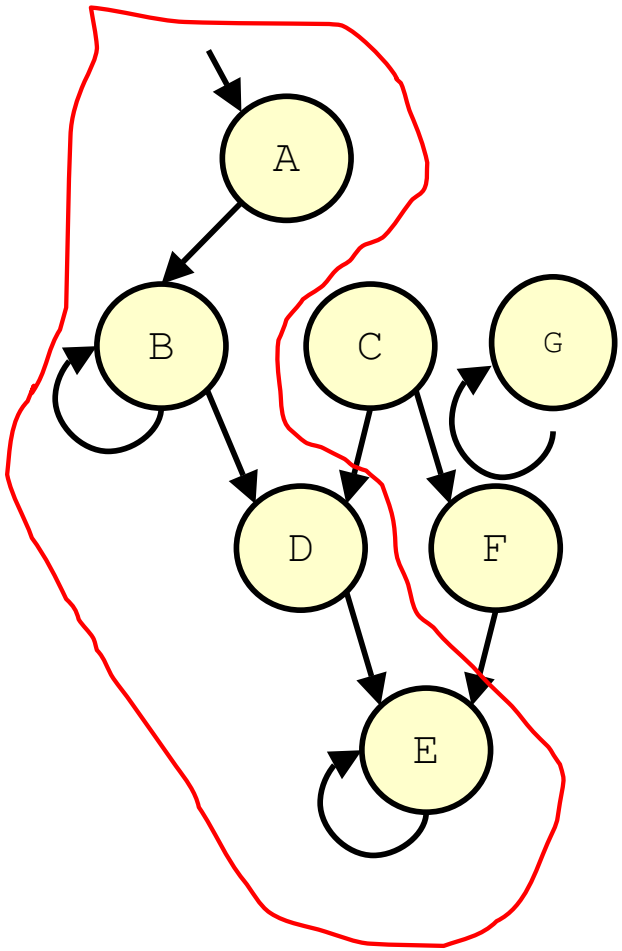
- Система переходов описывает все состояния и переходы
- Сформулируем условие корректности как достижимость “хорошего” состояния
  - Недостижимость “плохого”



Состояние, в котором PC1=12 и PC2=22 не достижимо

Достижимый подграф

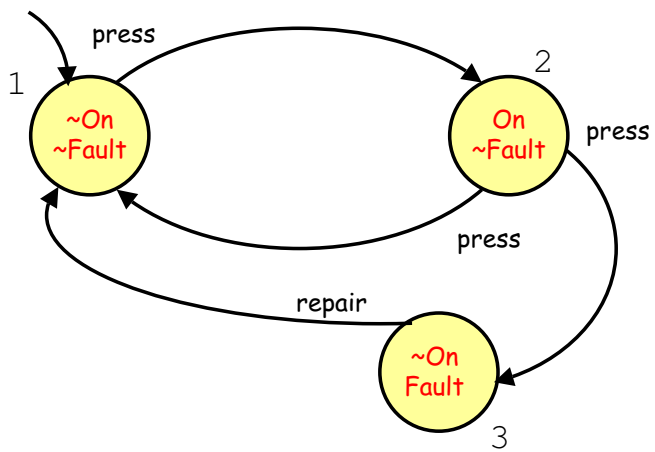
- (Q, E, T, q_0, L) — система переходов
- Состояние q достижимо, если существует путь в него из q_0
- $\mathrm{Reach}(S) = \{q | q \text{ достижимо в } S\}$
  - Достижимый подграф



Анализ достижимости

- (Q, E, T, q_0, L) — система переходов
- $P \subseteq Q$ — множество состояний
- Определим операции
  - $\mathrm{Pre}(P) = \{q \in Q \mid \exists p \in P, \exists e \in E : (q, e, p) \in T\}$
  - $\mathrm{Post}(P) = \{q \in Q \mid \exists p \in P, \exists e \in E : (p, e, q) \in T\}$

- Например,
  - $\mathrm{Pre}(\{1, 2\}) = \{1, 2, 3\}$
  - $\mathrm{Post}(\{2\}) = \{1, 3\}$



Алгоритм поиска

- Если *new* — стек, поиск в глубину
- Если *new* — очередь, поиск в ширину
- Сложность:
  - n — число состояний
  - m — число переходов
  - $O(n+m)$
- Произведение систем переходов может быть построено “на лету”

Вход: система переходов S
Выход: $\mathrm{Reach}(S)$
$new \leftarrow \{q_0\}$
$R \leftarrow \{\}$
while $new \neq \{\}$ **do**
  выбрать q из new, удалить q из new
  if $q \notin R$ **then**
    добавить q к R
    добавить $\mathrm{Post}(q)$ к new
  end if
end while
return R

Итоги

- Системы переходов как модели реактивных систем
- Анализ достижимости для верификации простых условий
  - Аналогично, можно построить алгоритм обратного поиска, который проверяет достижимость “плохого” состояния, используя $\mathrm{Pre}(P)$.
  - Сложность: порядка размера системы переходов
  - Композицию частей можно строить “на лету”
    - Экспоненциальный рост размера!
- Как верифицировать более сложные условия?
  - “Каждый процесс, который хочет войти в критическую секция, рано или поздно войдет в неё”
  - “каждое посланное сообщение будет получено”
  - “устройство готово к работе бесконечно часто”
- Требуется более богатый формализм

Что дальше?

- Временная логика LTL
- Проверка моделей на основе табличных доказательств
- NuSMV