Введение в формальные методы верификации программ

Занятие №1

Аксиома 1. *В каждой [даже отлаженной] программе есть хотя бы одна ошибка*

Аксиома 2. *Если ошибки нет в программе, она есть в используемом алгоритме*

Аксиома 3. *Если ошибки нет и в алгоритме, такая программа никому не нужна*

М.Р. Шура-Бура (1918 – 2008)

Александр Сергеевич Камкин

kamkin@ispras.ru

Контакты

- **Александр Сергеевич Камкин**
  - E-mail: askamkin@gmail, kamkin@ispras.ru
  - Phone: +7 926 1347886 (Telegram, WhatsApp)
  - Room: 211 (вход через 210)
- **Telegram-канал** (объявления и прочее)
  - t.me/model_checking

Содержание курса

- Темпоральная логика LTL (Liner-time Temporal Logic)
- Теоретико-автоматный подход к проверке моделей
- Инструмент SPIN
- Символическая проверка моделей
- Инструмент SMV
- Задача выполнимости: SAT- и SMT-решатели

Выставление оценок

- **Накопленная оценка (вес – 60%)**
  - Контрольная работа (PROMELA/SPIN)
  - Домашняя работа (SMV)
  - Контрольная работа (“на бумаге”)
  - Практикум (построение автомата Бюхи по формуле LTL, BDD, DPLL)
- **Оценка за экзамен (вес – 40%)**
  - Пара теоретических вопросов
  - Пара задач (“на бумаге”)
- *Учитывается активность на семинарах!*

Ошибки в программах

- **Ошибка – несоответствие программы требованиям**
- Две ипостаси ошибки
  - *Failure* – отклонение поведения от ожидаемого
  - *Fault* (*bug*) – причина такого отклонения
- Статистика числа ошибок
  - В среднем **10-50** ошибок на **1000** строк неотлаженного кода

С. Макконнелл. *Совершенный код*. *Мастер-класс*. М.: Издательство «Русская редакция», 2015.
(S. McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, 2004)

Примеры ошибок в компьютерных системах

- 1982 – Therac-25 (AECL)
- 1988 – Фобос-1 / Фобос-2 (СССР)
- 1994 – Pentium (Intel)
- 1996 – Ariane-5 (ESA)
- 2003 – Patriot (США)

Примеры ошибок:
Ю.Г. Карпов. *Model Checking. Верификация параллельных и распределенных программных систем*. СПб.: БХВ-Петербург, 2010.

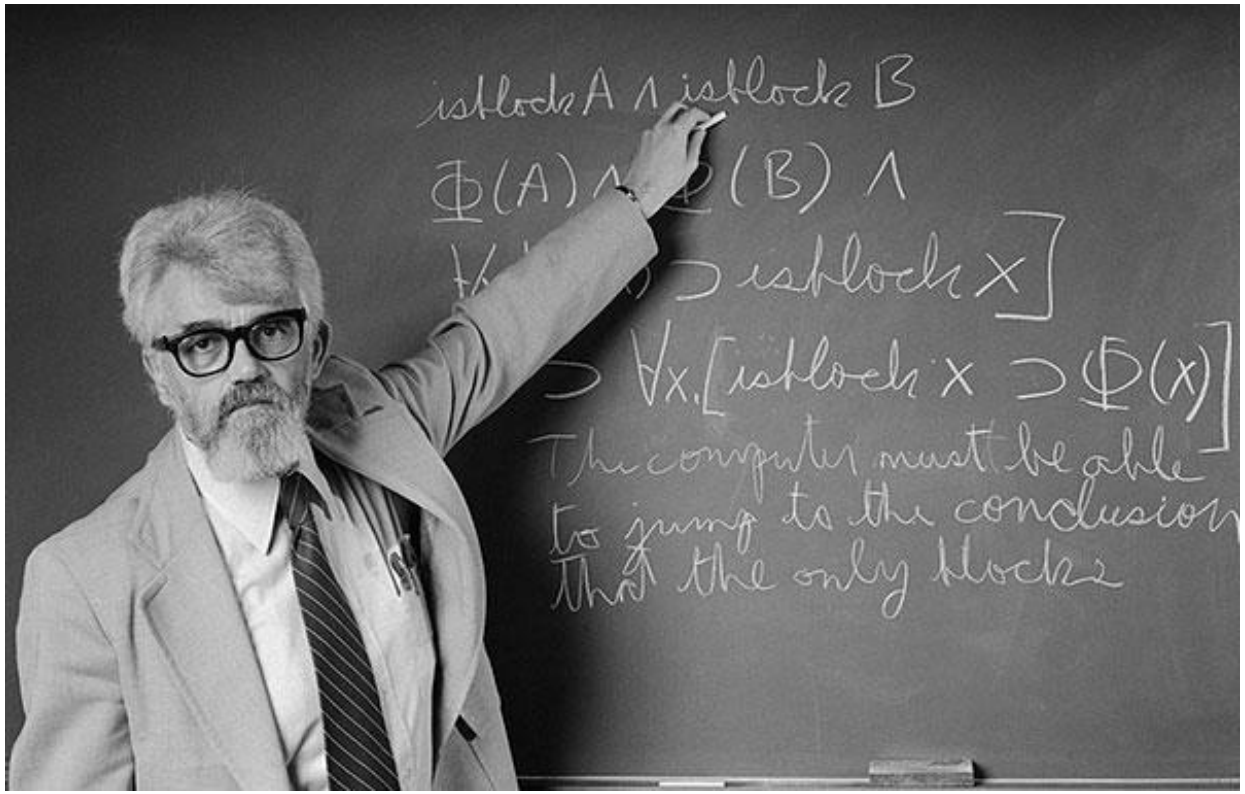
Анализ некоторых ошибок:
В. Аджиев. *Мифы о безопасном ПО: уроки знаменитых катастроф*. Открытые системы, №6, 1998 (http://www.osp.ru/os/1998/06/179592/).

Верификация программ

- **Что такое верификация?**
  - Проверка соответствия программы требованиям
    - Положительный вердикт – доказано отсутствие ошибок
    - Отрицательный вердикт – продемонстрировано наличие ошибок
    - Неопределенный вердикт – ошибки не найдены, но их отсутствие не доказано
- **Методы верификации**
  - *Формальные методы* – то, чем мы будем заниматься
  - Методы тестирования
  - Экспертиза (инспекция кода)
  - Гибридные (синтетические, полуформальные) методы

В.В. Кулямин. *Методы верификации программного обеспечения*, 2008 (http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf).

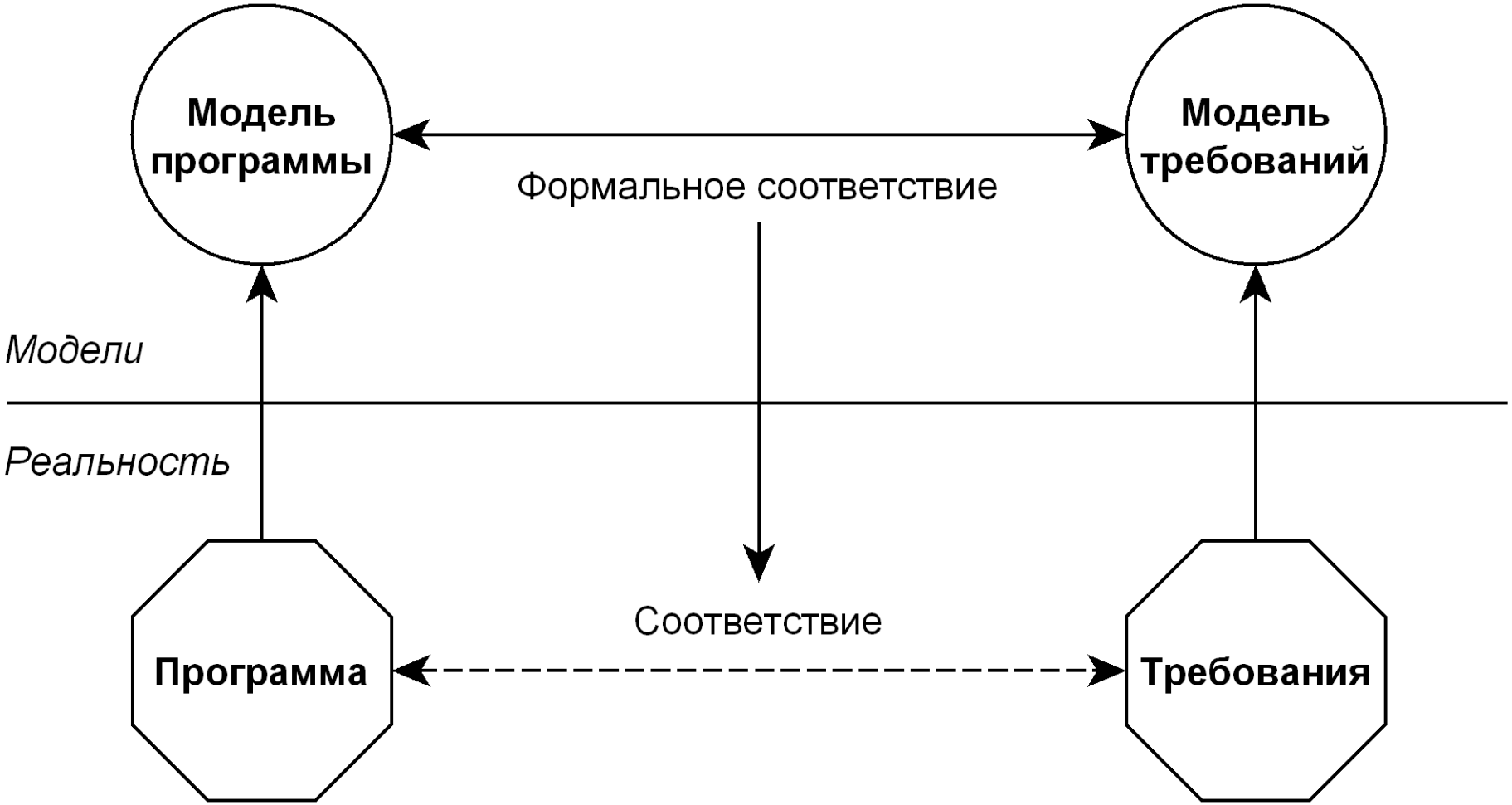
Формальные методы



Джон Маккарти (John McCarty; 1927-2011)

Разумно ожидать, что связи между вычислительной техникой и математической логикой окажутся столь же плодотворными в следующем столетии [XXI веке]*, какими были связи между математическим анализом и физикой в столетии предыдущем* [XIX веке]

Схема формальной верификации



Примеры формальных методов

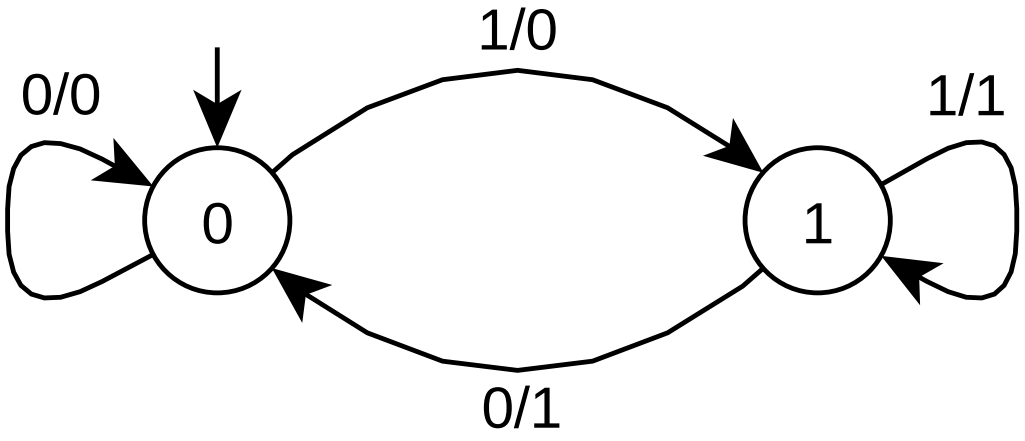
- **Проверка эквивалентности**
  - *Модель программы*: конечный автомат (программа на формализованном языке)
  - *Модель требований*: конечный автомат (программа на формализованном языке)
  - *Отношение соответствия*: (функциональная) эквивалентность
- **Проверка модели**
  - *Модель программы*: структура Крипке (размеченная система переходов)
  - *Модель требований*: формула темпоральной логики (LTL, CTL)
  - *Отношение соответствия*: истинность формулы на структуре
- **Дедуктивный анализ**
  - *Модель программы*: программа на формализованном языке
  - *Модель требований*: программный контракт (пред- и постусловие)
  - *Отношение соответствия*: частичная или полная корректность

Конечный автомат Мили

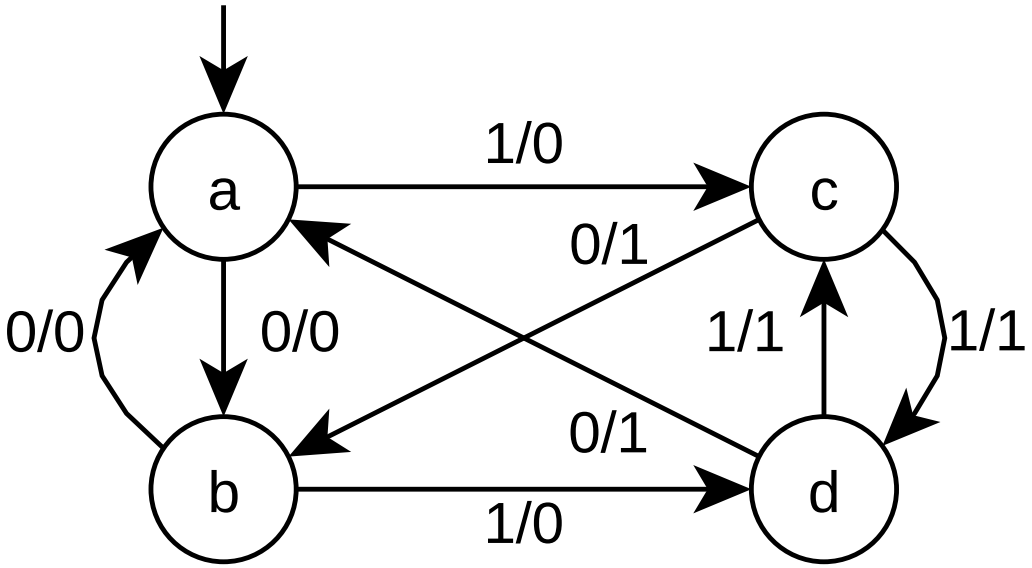
Конечный автомат Мили – $\langle S, X, Y, s_0, \delta, \lambda \rangle$, где

- S — множество состояний
- X — входной алфавит (множество стимулов)
- Y — выходной алфавит (множество реакций)
- $s_0 \in S$ — начальное состояние
- $\delta: S \times X \to S$ — функция переходов
- $\lambda: S \times X \to Y$ — функция выходов

Эквивалентность конечных автоматов



Автомат A



Автомат B

Как проверить эквивалентность автоматов?

- **Тестирование «черных ящиков»**
  - *Теорема Мура (о различении состояний автоматов)*:

    для любых двух КА и любых двух различимых состояний существует различающий эксперимент длины не большей $n + m - 1$,
    где n и m — мощности множеств состояний данных КА
- **Статический анализ отношений переходов**
  - *Теорема Мура (об эквивалентности автоматов)*:

    два КА A и B эквивалентны тогда и только тогда, когда в любом достижимом состоянии их прямого произведения (s_A, s_B) для всех стимулов x выполнено равенство $\lambda_A(s_A, x) = \lambda_B(s_B, x)$
- **Эквивалентные преобразования**

Эквивалентность программ

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while y ≠ 0 **do**
	  z := x;
	  x := y;
	  y := z % y
	end;
	c := x

Как проверить эквивалентность программ?

- **Догадаться, что обе программы вычисляют** $\text{НОД}(a, b)$:

$$\exists m, n \in \mathbb{N}\big((a = c \cdot m) \wedge (b = c \cdot n)\big) \wedge$$
$$\forall c', m', n' \in \mathbb{N}\left(\big((a = c' \cdot m') \wedge (b = c' \cdot n')\big) \rightarrow (c' \leq c)\right)$$

  - Доказать, что каждая программа удовлетворяет «контракту»
    - Дедуктивная верификация (будет рассмотрена позже)
- **Применить эквивалентные преобразования**
  - Преобразовать одну программу к другой (или обе к третьей)
    - Этот подход в курсе не рассматривается

Преобразование программы Q к P

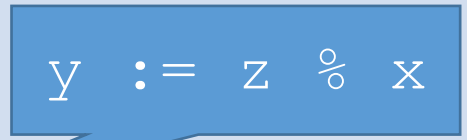
Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while y ≠ 0 **do**
	  z := x;
	  x := y;
	  y := z % y
	end;
	c := x

Преобразование программы Q к P

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while y ≠ 0 **do**
	  z := x;
	  x := y;
	  y := z % y
	end;
	c := x



Преобразование программы Q к P

Программа P	Программа Q

`x := a;`
`y := b;`
`while x ≠ y do`
`  if x > y then`
`  x := x - y`
`  else`
`  y := y - x`
`  end`
`end;`
`c := x`
	`x := a;`
	`y := b;`
	`while y ≠ 0 do`
	`  z := x;`
	`  x := y;`
	`  y := z % x`
	`end;`
	`c := x`

Преобразование программы Q к P

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while y ≠ 0 **do**
	  z := x;
	  x := y;
	  y := z % x
	end;
	c := x

```
y := z;
while y ≥ x do
  y := y − x
end
```

Преобразование программы Q к P

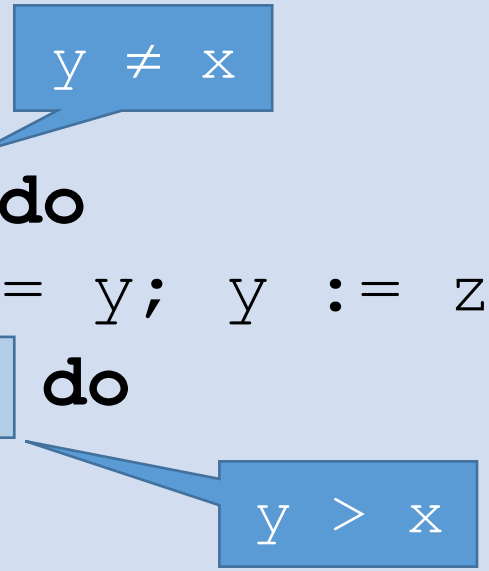
Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while y ≠ 0 **do**
	  z := x; x := y; y := z;
	  while y ≥ x **do**
	  y := y − x
	  end
	end;
	c := x

Преобразование программы Q к P

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while y ≠ 0 **do**
	  z := x; x := y; y := z;
	  while y ≥ x **do**
	  y := y − x
	  end
	end;
	c := x



Преобразование программы Q к P

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while x ≠ y **do**
	  z := x; x := y; y := z;
	  while y > x **do**
	  y := y − x
	  end
	end;
	c := x

Преобразование программы Q к P

Программа P

```
x := a;
y := b;
while x ≠ y do
 if x > y then
  x := x − y
 else
  y := y − x
 end
end;
c := x
```

Программа Q

```
x := a;
y := b;
while x ≠ y do
 z := x; x := y; y := z;
 while y > x do
  y := y − x
 end
end;
c := x
```

Перестановка переменных

Преобразование программы Q к P

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
  if x > y **then**
  x := x − y
  else
  y := y − x
  end
end;
c := x
	x := a;
	y := b;
	while x ≠ y **do**
	  while x > y **do**
	  x := x − y **end**
	  while y > x **do**
	  y := y − x
	  end
	end;
	c := x

Преобразование программы Q к P

Программа P

```
x := a;
y := b;
while x ≠ y do
 if x > y then
  x := x − y
 else
  y := y − x
 end
end;
c := x
```

Программа Q

```
x := a;
y := b;
while x ≠ y do
 while x > y do
  x := x − y end
 while y > x do
  y := y − x
 end
end;
c := x
```

Замена на if-then-else

Преобразование программы Q к P

Программа P	Программа Q

x := a;
y := b;
while x ≠ y **do**
if x > y **then**
x := x − y
else
y := y − x
end
end;
c := x
	x := a;
	y := b;
	while x ≠ y **do**
	if x > y **then**
	x := x − y
	else
	y := y − x
	end
	end;
	c := x

Структура Крипке

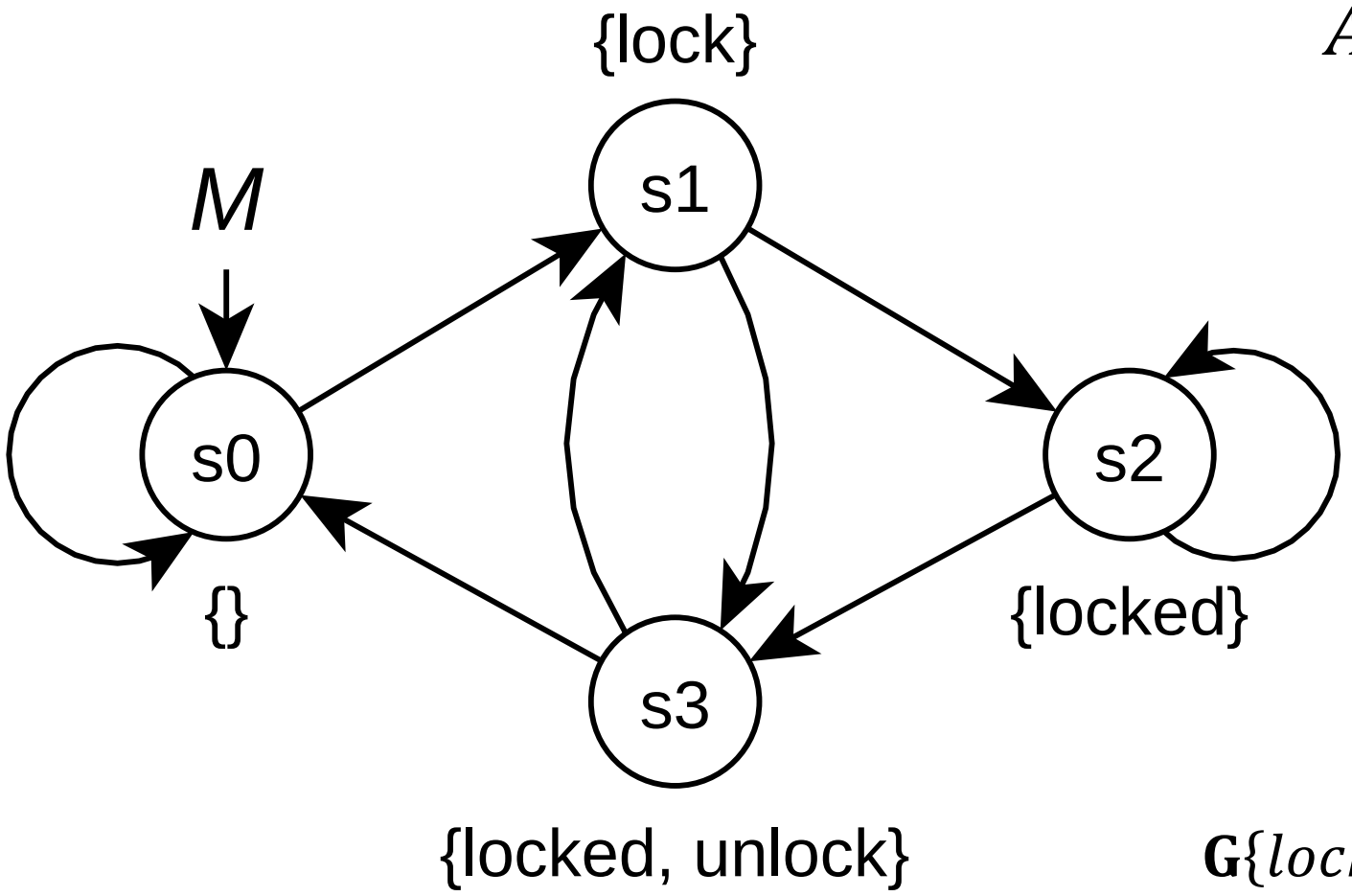
Структура Крипке над множеством AP – $\langle S, S_0, R, L \rangle$, где

- S — множество состояний
- $S_0 \subseteq S$ — множество начальных состояний
- $R \subseteq S \times S$ — отношение переходов
- $L: S \to 2^{AP}$ — функция разметки, помечающая каждое состояние множеством истинных в этом состоянии высказываний

Логика LTL: Linear-time Temporal Logic

- p — элементарное высказывание
- $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$ и т.д. – классические связки
- $\mathbf{G}\varphi$ — формула φ истинна *всегда*
- $\mathbf{F}\varphi$ — φ истинна *сейчас* или станет истинной *в будущем*
- $\mathbf{X}\varphi$ — φ станет истинной *в следующий момент времени*
- $\varphi\ \mathbf{U}\ \psi$ — φ истинна *до тех пор, пока* не станет истинной ψ

Пример структуры Крипке и формул LTL



$$AP = \{locked, lock, unlock\}$$

$$\mathbf{G}\{\neg(lock \wedge unlock)\}$$

$$\mathbf{G}\{\neg(locked \wedge lock)\}$$

$$\mathbf{G}\{\neg(\neg locked \wedge unlock)\}$$

$$\mathbf{G}\{lock \rightarrow \mathbf{X}(locked)\}$$

$$\mathbf{G}\{unlock \rightarrow \mathbf{X}(\neg locked)\}$$

$$\mathbf{G}\{locked \rightarrow \neg(\neg unlock\ \mathbf{U}\ \neg locked)\}$$

Частичная и полная корректность

- *Программный контракт* – предусловие и постусловие
- Программа P *частично корректна* относительно предусловия φ и постусловия ψ ($\{\varphi\}\ P\ \{\psi\}$), если

$$\varphi \Rightarrow (P \text{ завершается} \Rightarrow \psi)$$

- Программа P *полностью корректна* относительно предусловия φ и постусловия ψ ($\langle\varphi\rangle\ P\ \langle\psi\rangle$), если

$$\varphi \Rightarrow (P \text{ завершается} \wedge \psi)$$

Пример программного контракта

```
{(a ≥ 0) ∧ (b > 0)}
q := 0;
r := a;
while r ≥ b do
   q := q + 1;
   r := r - b
end
{(a = q * b + r) ∧ (0 ≤ r < b)}
```

Учебное пособие

http://sp.cs.msu.ru/courses/vmp/kamkin_mc2018.pdf

Что почитать

- Ю.Г. Карпов. *Model Checking. Верификация параллельных и распределенных программных систем*. — СПб.: БХВ-Петербург, 2010.
- G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.
- C. Baier, J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
- Э.М. Кларк, О. Грамберг, Д. Пелед. *Верификация моделей программ. Model Checking*. — М.: МЦНМО, 2002.
- M. Ben-Ari. *Mathematical Logic for Computer Science*. Springer, 2012.