

Математические методы верификации схем и программ

mk.cs.msu.ru → Лекционные курсы
→ Математические методы верификации схем и программ

Блок 7

Модели Крипке

Особенности моделирования систем

Лектор:

Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

Модели Крипке

Обычно модель в рамках метода model checking устроена так:

- ▶ Моделью задаётся множество **состояний**: «слепков» системы, в которых записаны рассматриваемые особенности системы в заданные моменты времени выполнения
- ▶ Состояния могут изменяться посредством выполнения **переходов**, изменяющих текущее состояние согласно выполнению заданных **действий** системой
- ▶ Выполнение системы в неограниченном времени соответствует **вычислению** модели: бесконечной последовательности состояний, получающейся из заданного состояния выполнением переходов

Модели Крипке





Model checking применяется *в основном* для анализа систем с **конечным** числом состояний


Это ещё один недостаток метода, затрудняющий его широкое использование: на практике число состояний системы нередко бесконечно или конечно, но настолько велико, что можно считать его практически бесконечным

Тем не менее, существуют и важные классы систем, заведомо обладающие конечным числом состояний: **контроллеры**, **драйверы**, многие **коммуникационные протоколы**, не слишком объёмная **аппаратура**, ...





Модели Крипке

Обсуждение моделей вычислительных систем начнём немного издалека, с классической головоломки про волка, козу и капусту

На левом берегу реки располагаются волк () , коза () , капуста () и лодочник с лодкой ()






 может переправиться на противоположный берег в лодке, взяв с собой не более одного пассажира (, , )

Оставшись на берегу без  ,  может съесть  , а  может съесть 

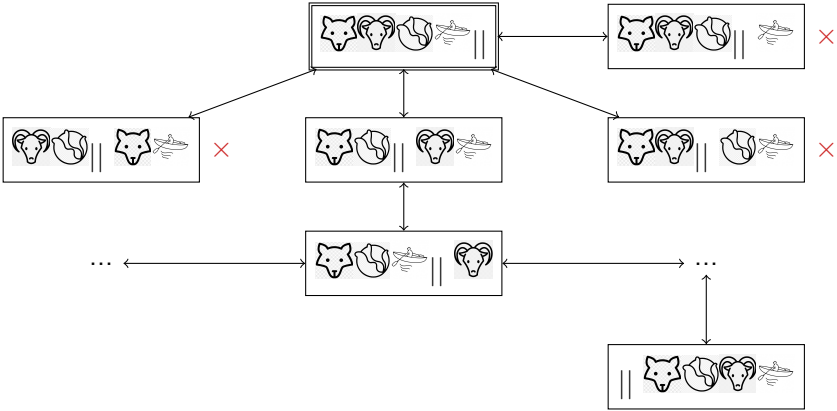
Как  может безопасно переправить ,  и  на правый берег?

Модели Крипке

Чтобы решить эту головоломку, достаточно

- ▶ перебрать всевозможные варианты расположения  ,  ,  и  , получающиеся из начального расположения согласно всевозможным действиям 
- ▶ это **состояния** системы
- ▶ разделить состояния на “плохие” (кто-то кого-то может съесть) и “хорошие” (остальные)
 - ▶ пометим плохие состояния символом ✕
- ▶ посмотреть, как достичь состояния “все на правом берегу”, ни разу не встретив ✕

Модели Крипке



□ — состояния системы

▣ — начальное состояние

→ — переходы системы

× — атомарное высказывание: свойство состояний системы, которое мы по тем или иным причинам посчитали необходимым для рассмотрения

Модели Крипке

Для множества X записью 2^X будем обозначать множество всех подмножеств X

Модель Крипке над множеством **атомарных высказываний** AP — это система $M = (S, S_0, \rightarrow, L)$, где:

- ▶ S — множество **состояний**
- ▶ S_0 — множество **начальных** состояний, $S_0 \subseteq S$
- ▶ $\rightarrow \subseteq S \times S$ — **тотальное** отношение **переходов**
- ▶ $L : S \rightarrow 2^{AP}$ — **функция разметки**

Тотальность отношения переходов означает, что для любого состояния s существует состояние s' , такое что $s \rightarrow s'$

Событием будем называть произвольное множество атомарных высказываний (элемент семейства 2^{AP})

Модели Крипке

$(M = (S, S_0, \rightarrow, L)$ — модель Крипке над AP)

Соотношение $L(s) = \sigma$ можно понимать так: состояние s обладает свойствами, отвечающими атомарным высказываниям из σ , и не обладает остальными свойствами, отвечающими атомарным высказываниям

Будем говорить, что модель M **конечна**, если конечны множества S и AP

Модель M представляет собой особый размеченный ориентированный граф: S — это вершины, \rightarrow — это дуги, остальное — это метки вершин

В связи с этим будем графовые обозначения и графовую терминологию по отношению к моделям Крипке

Путь, исходящий из начального состояния модели, будем называть **начальным**

Бесконечный начальный путь будем называть **вычислением** модели

Модели Крипке

$(M = (S, S_0, \rightarrow, L)$ — модель Крипке над AP)

Трассой будем называть бесконечную последовательность событий

Трассой пути $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$ будем называть трассу, состоящую из событий, помечающих состояния этого пути:

$$L(s_0), L(s_1), L(s_2), \dots$$

Для последовательности $\Sigma = (x_0, x_1, x_2, \dots)$ (в том числе для пути и для трассы) будем использовать такие обозначения:

- ▶ $\Sigma[i] = x_i$ — i -й элемент последовательности, нумерация с нуля
- ▶ $\Sigma^i = (x_i, x_{i+1}, \dots)$ — **суффикс** последовательности, начинающийся с i -го элемента

Особенности моделирования систем

Моделирование программ

Рассмотрим **императивную программу** π , выполняющуюся в интерпретации \mathcal{I} на произвольной оценке данных множества Θ

Модель $M_{\pi, \mathcal{I}, \Theta} = (S, S_0, \rightarrow, L)$, отвечающая такому выполнению, может быть устроена так:

- ▶ S — это множество всех **состояний вычисления** программы
- ▶ $S_0 = \{\langle \pi \mid \theta \rangle \mid \theta \in \Theta\}$
- ▶ \rightarrow — **отношение переходов программы**, в которое добавлены всевозможные пары вида $\langle \emptyset \mid \theta \rangle \rightarrow \langle \emptyset \mid \theta \rangle$
- ▶ AP — всевозможные пары (x, d) , где x — переменная программы и d — предмет из \mathcal{I}
- ▶ $(x, d) \in L(\langle \pi' \mid \theta \rangle) \Leftrightarrow \overline{\theta(x)} = d$

Фрагментами такой модели Крипке являются **вычисления программы** в \mathcal{I} на θ , $\theta \in \Theta$

Для других видов программ модель можно устроить дословно так же, если для программ определена **операционная семантика**

Особенности моделирования систем

Моделирование программ

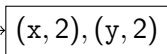
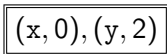
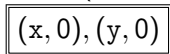
Для примера рассмотрим программу, в которой в бесконечно (в цикле) выполняется присваивание

$$x := x + y;$$

в двухбитовой арифметике с переполнением в модели императивных программ, но с заменой термов t в оценках данных на соответствующие предметы \bar{t}

Пусть известно, что в начале работы программы $x = 0$, а значение y может быть любым

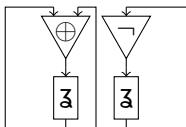
Тогда некоторые компоненты связности соответствующей модели Крипке устроены так (*можете представить себе и остальные по аналогии*):



Особенности моделирования систем

Моделирование схем

(кто знает термин «последовательная схема» — можете представить её вместо схемы из функциональных элементов с задержкой, СФЭЗ)



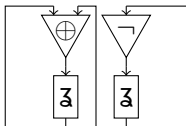
Модель $M = (S, S_0, \rightarrow, L)$, отвечающая СФЭЗ (последовательной схеме), может быть устроена так:

- ▶ Все элементы задержки пронумерованы: $1, 2, \dots, n$
- ▶ $S = \{0, 1\}^n$ (все состояния схемы)
- ▶ $S_0 = \{(0, 0, \dots, 0)\}$ (начальное состояние схемы)
- ▶ $s \rightarrow s' \Leftrightarrow$ при переходе к следующему моменту времени (по переднему фронту тактового сигнала) возможна такая смена состояния схемы
- ▶ $AP = \{1, 2, \dots, n\} \times \{0, 1\}$ (номер и состояние регистра)
- ▶ $(i, b) \in L(b_0, b_1, \dots, b_n) \Leftrightarrow b_i = b$

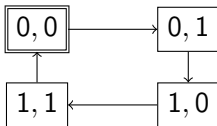
Особенности моделирования систем

Моделирование схем

(кто знает термин «последовательная схема» — можете представить её вместо схемы из функциональных элементов с задержкой, СФЭЗ)



Например, модель Крипке для этой схемы может быть устроена так:



Особенности моделирования систем

Моделирование параллелизма

Современные вычислительные системы зачастую состоят из набора компонентов, исполняющихся одновременно (параллельно) и взаимодействующих друг с другом

В зависимости от природы системы, при построении модели используется один из двух видов параллелизма (или их комбинация):

- ▶ **Асинхронное исполнение** (**чередующееся исполнение; семантика чередующихся вычислений; interleaving**): шаг вычисления системы отвечает одному шагу **одного** компонента, а остальные компоненты не делают ни одного шага
- ▶ **Синхронное исполнение**: шаг вычисления системы отвечает одновременному выполнению шага вычисления **всех** компонентов

Особенности моделирования систем

Моделирование параллелизма

Параллельная композиция моделей Крипке $M = (S, S_0, \rightarrow, L)$ и $M' = (S', S'_0, \mapsto, L')$ над непересекающимися множествами атомарных высказываний — это модель $M|M' = (S \times S', S_0 \times S'_0, \rightsquigarrow, \mathcal{L})$, где:

- ▶ $\mathcal{L}(s, s') = L(s) \cup L'(s')$
- ▶ Отношение переходов \rightsquigarrow определяется видом параллелизма

Синхронное исполнение характерно для аппаратных систем и других имеющих встроенные средства синхронизации компонентов

Переходы синхронной композиции моделей (без взаимодействия компонентов) определяются так:

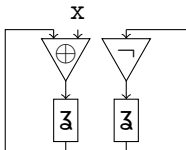
$$(s_1, s'_1) \rightsquigarrow (s_2, s'_2) \Leftrightarrow s_1 \rightarrow s_2 \text{ и } s'_1 \mapsto s'_2$$

Особенности моделирования систем

Моделирование параллелизма

$$M = (S, S_0, \rightarrow, L) \quad M' = (S', S'_0, \mapsto, L') \quad M|M' = (S \times S', S_0 \times S'_0, \rightsquigarrow, \mathcal{L})$$

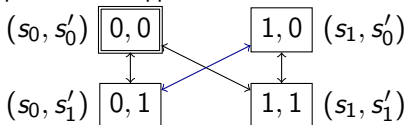
Например:



Модели Крипке, описывающие поведение левой и правой задержек:



Синхронная композиция этих моделей:



Особенности моделирования систем

Моделирование параллелизма

Асинхронное исполнение характерно для систем без встроенных средств синхронизации компонентов, в том числе (с «примесью» синхронности) для программных систем

Переходы асинхронной композиции моделей (без взаимодействия компонентов) определяются так:

$$(s_1, s'_1) \rightsquigarrow (s_2, s'_2) \Leftrightarrow (s_1 \rightarrow s_2 \text{ и } s'_1 = s'_2) \text{ или } (s_1 = s_2 \text{ и } s'_1 \mapsto s'_2)$$

Особенности моделирования систем

Моделирование параллелизма

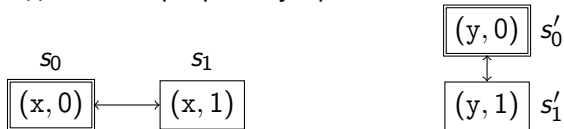
Для примера рассмотрим две параллельно работающие программы, в цикле выполняющие одно присваивание:

$$\pi_1 : x := x + 1;$$

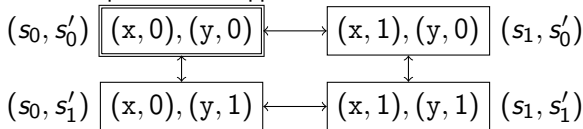
$$\pi_2 : y := y + 1;$$

Для простоты будем считать, что эти программы выполняются в условиях однобитовой арифметики с переполнением

Модели Крипке для этих программ устроены так:



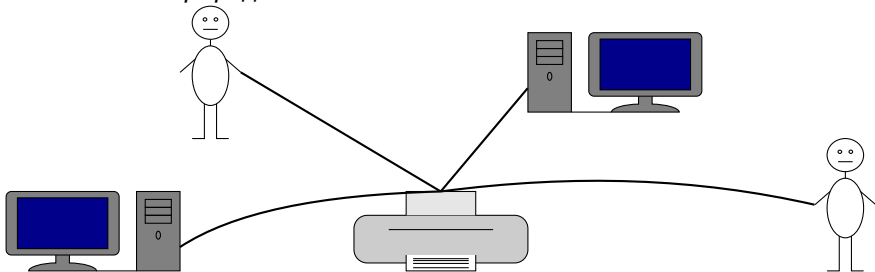
Асинхронная композиция этих моделей:



Особенности моделирования систем

Моделирование взаимодействия

Пример: с сетевым принтером пытаются взаимодействовать участники *неизвестной природы*



Принтер работает последовательно: принимает информацию и производит печать согласно содержащейся в нём *программе*

Программы остальных участников, *если они есть*, неизвестны

Особенности моделирования систем

Моделирование взаимодействия

Предположим, что в контроллере принтера есть однобитовый регистр R , доступный на чтение и запись всем желающим послать запрос на печать:

$$R = \text{т} \Leftrightarrow \text{принтер свободен для печати}$$

Тогда программу π , посредством которой можно организовать взаимодействие участника с принтером, можно устроить так:

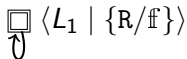
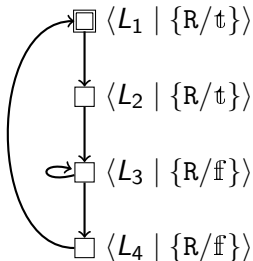
```
while т do
  L1 : while ¬R do ∅ od
  L2 : R := f;
  L3 : послать данные для печати
  L4 : R := т;
od
```

Особенности моделирования систем

Моделирование взаимодействия

Модель Крипке для π :

(функция разметки опущена)



Особенности моделирования систем

Моделирование взаимодействия

Программа в вычислительной системе может взаимодействовать с другими программами: **общие переменные**, **обмен сообщениями**, **сигналы**, ...

Такое взаимодействие выражается в том, что состояние вычисления программы может измениться под воздействием её **окружения**

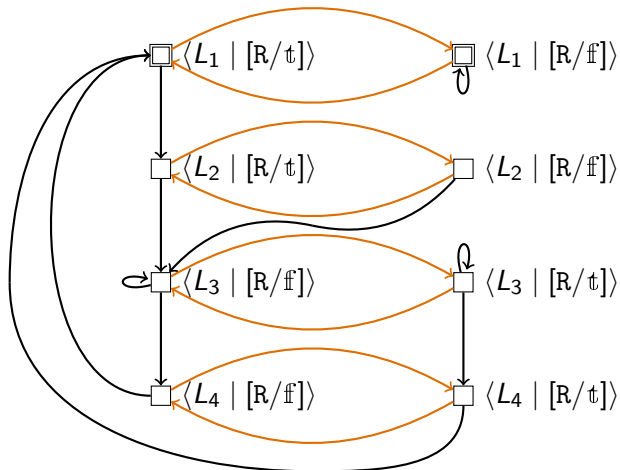
Например, регистр R в примере может быть изменён любым участником

Чтобы учесть такое изменение, следует добавить в модель Крипке переходы, отвечающие всем возможностям окружения повлиять на состояние вычисления программы

Особенности моделирования систем

Моделирование взаимодействия

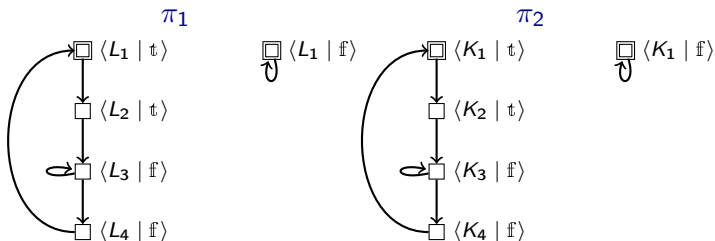
Модель Крипке для программы π с окружением, способным произвольно переключать значение регистра R:



Особенности моделирования систем

Моделирование взаимодействия

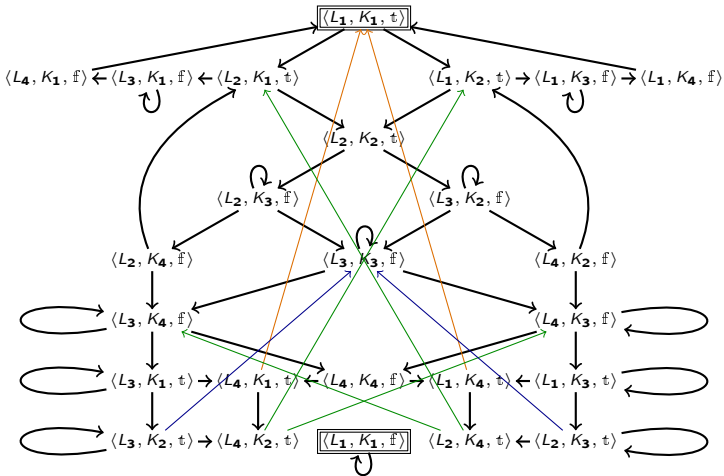
Рассмотрим две (одинаковые) программы взаимодействия с сетевым принтером, выполняющиеся согласно следующим моделям Крипке:



Особенности моделирования систем

Моделирование взаимодействия

Модель Крипке, описывающая взаимодействие π_1 , π_2 согласно семантике чередующихся вычислений с общим регистром R , выглядит так:



Особенности моделирования систем

Гранулярность и атомарность в моделях

При моделировании параллельных систем большую роль играет **гранулярность переходов**, то есть степень их детализации

Переход должен соответствовать **атомарному** действию:

- ▶ такому, в выполнение которого не может вмешаться другое действие
- ▶ не имеющему промежуточных наблюдаемых состояний
- ▶ не подразбивающемуся на более простые действия

Если детализация переходов будет слишком мала (переходы слишком «крупны»), то в модели могут отсутствовать некоторые ошибки, наблюдающиеся при частичном выполнении и «перекрытии» действий

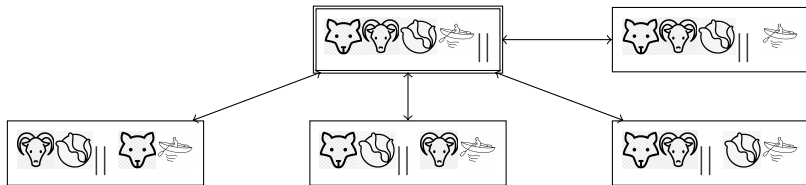
Если же детализация переходов будет слишком велика, то это может

- ▶ существенно увеличить размер модели и снизить эффективность её верификации и
- ▶ добавить в композицию такие состояния, которых не бывает в реальной системе

Особенности моделирования систем

Гранулярность и атомарность в моделях

Например,



Нужно ли рассматривать отдельное действие «плавание по реке»?

А «посадка в лодку» и «высадка из лодки»?

Можно ли в качестве атомарного выбрать действие плавания туда и обратно?

Особенности моделирования систем

Гранулярность и атомарность в моделях

Другой пример

Рассмотрим две параллельно выполняющиеся программы, каждая из которых выполняет одну команду

$$\pi_1 : x := x + y;$$
$$\pi_2 : y := x + y;$$

Устроит ли нас, если эти две команды будут считаться атомарными?

Из оценки данных $\{x/2, y/3\}$ в вычислениях можно достичь только оценок $\{x/5, y/3\}$, $\{x/5, y/8\}$, $\{x/2, y/5\}$ и $\{x/7, y/5\}$

Реализация этих команд на языке ассемблера может содержать и более одной команды:

```
load $1, x
```

```
load $2, y
```

```
add $1, $2
```

```
store $1, x
```

```
load $3, x
```

```
load $4, y
```

```
add $3, $4
```

```
store $3, y
```

Если атомарными считать ассемблерные команды, то достижимы и другие оценки — например, $\{x/5, y/5\}$, и такой результат может быть и нежелательным