Математическая логика и логическое программирование

mk.cs.msu.ru → Лекционные курсы
→ Математическая логика и логическое программирование (3-й поток)

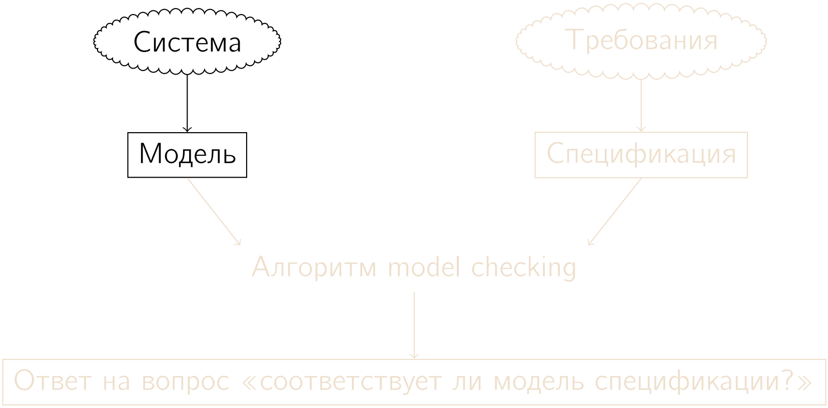
Блок 56

Размеченные системы переходов

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2023/2024, осенний семестр

Вступление (краткая схема model checking)



Размеченные системы переходов

Чтобы лучше понять, как (*и почему именно так*) устроена модель вычислительной системы, используемая в model checking, рассмотрим **для примера** такую систему

Система состоит из кофейного автомата и покупателя

Кофейный автомат имеет приёмник монет и кнопки «чай» и «кофе» и запрограммирован на такое поведение:

- В режиме ожидания приёмник монет открыт
- После приёма монеты
  - приёмник закрывается, ожидается нажатие на одну из кнопок,
  - после нажатия на кнопку
    соответствующий напиток выдаётся покупателю,
    монета удаляется из приёмника
    и автомат переходит в режим ожидания

Покупатель в зависимости от своего желания может кидать монеты в приёмник и нажимать на кнопки

Размеченные системы переходов

Поведение кофейного автомата можно представить себе так

В каждый момент времени он находится в некотором состоянии: «ожидает монету», «ожидает нажатия кнопки», «выдаёт чай», «выдаёт кофе»

Иногда автомат переходит из одного состояния в другое согласно внешним обстоятельствам и своей программе

Некоторое состояние (начальное) отвечает запуску программы

Чтобы проверить правильность работы автомата, достаточно выделить набор *простых* свойств состояний (атомарных высказываний) и проанализировать изменение этих свойств с течением времени: «приёмник открыт», «в приёмнике есть монета», «выдаётся чай», «выдаётся кофе»

Размеченные системы переходов

AP — так будем обозначать множество атомарных высказываний

В качестве модели системы будем использовать размеченную систему переходов (СП)[1] $M = (S, S_0, \mapsto, L)$ над AP, устроенную так:

- $(S, \to, L)$ — это модель Крипке над переменными AP, в которой
  - миры из S называются состояниями,
  - отношение переходов $\mapsto \subseteq S \times S$ тотально: для каждого состояния s существует состояние s', такое что $s \mapsto s'$ — и
  - оценку переменных $L : S \to 2^{\mathrm{AP}}$ принято называть функцией разметки состояний
- S_0 — множество начальных состояний, $S_0 \subseteq S$

СП будем называть конечной, если конечны множества её состояний и атомарных высказываний

1 На самом деле СП — это более широкое понятие, и модель Крипке с начальными мирами (состояниями) является частным случаем такой системы, но сейчас нет смысла всё переусложнять

Размеченные системы переходов

СП представляет собой размеченный ориентированный граф, вершинами которого являются состояния, а дугами — переходы, и поэтому будем использовать для СП терминологию теории графов

Путь в СП будем называть начальным, если он исходит из начального состояния

Бесконечный начальный путь будем называть вычислением СП

Вычисления СП отвечают (*потенциально*) бесконечным сценариям выполнения моделируемой системы

Размеченные системы переходов

Пример: СП кофейного автомата

Атомарные высказывания:

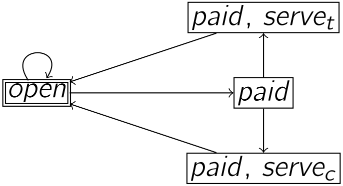
$open$ = «приёмник открыт»

$paid$ = «в приёмнике есть монета»

$serve_t$ = «выдаётся чай»

$serve_c$ = «выдаётся кофе»

СП:



□ — состояние

⧈ — начальное состояние

Высказывания, размечающие состояние, записаны внутри этого состояния

Система переходов программы

Математически строгий анализ императивной программы π (и программ других парадигм в рамках операционной семантики), как правило, основан на понятиях, которые уже появлялись в лекциях:

- Состояние управления: то, какую часть программы осталось выполнить
- Состояние данных: то, как устроены данные, преобразуемые программой на каждом шаге (например, оценки переменных или запросы)
- Состояние вычисления, включающее в себя состояние данных и состояние управления
- Отношение $\to_\pi$ шага вычисления программы π

Система переходов программы

Программе π отвечает СП $(S, S_0, \mapsto, L)$ над AP, где:

- S — множество всех состояний вычисления π
- S_0 — состояния, с которых начинаются вычисления π на интересующих входных данных
- $\mapsto = \to_\pi$ с поправкой на требуемую тотальность: если из состояния s не исходит ни одного перехода, то в $\mapsto$ «насильно» добавляется переход $s \mapsto s$
- AP суть интересующие свойства состояний вычисления, например:
  - «$\mathrm{x} = 2$», «$\mathrm{x} > 2$», «$\mathrm{x} > \mathrm{y}$»
  - «Состояние управления — это пустая команда»
  - «Предикатный символ левой подцели — P»
- $L(s)$ состоит из всех атомарных высказываний, истинных для s согласно естественной трактовке их записи

Система переходов программы

Пример

$$\pi = \text{счёт} := \text{счёт} + 1\,000;$$

Достижимый фрагмент СП этой модельной императивной программы для интерпретации $Ar_{\mathbb{Z}}$, начальной оценки $[\text{счёт}/0]$ и атомарных высказываний «$\text{счёт} < 100$» и «$\text{счёт} = 1\,000$»:

$$\begin{tikzcd} \boxed{\boxed{\text{счёт} < 100}} \arrow[r] & \boxed{\text{счёт} = 1\,000} \arrow[loop right] \\ \langle \pi \mid [\text{счёт}/0] \rangle & \langle \emptyset \mid [\text{счёт}/1\,000] \rangle \end{tikzcd}$$

Система переходов программы

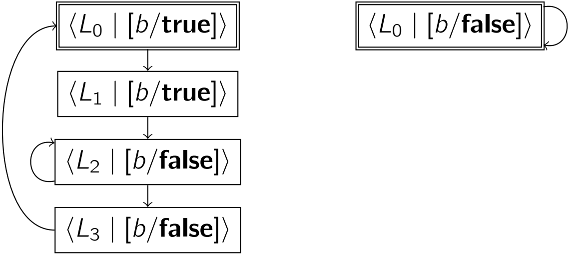
Другой пример

Пусть в сетевом принтере содержится булев регистр $\mathtt{b}$,
обозначающий *занятость* принтера,
и доступ к нему осуществляется при помощи такой программы π:

$$\textbf{while (true) } \{ \quad L_0 : \textbf{while } (!\mathtt{b}); \quad L_1 : \mathtt{b} = \textbf{false}; \quad L_2 : \mathit{EXCHANGE} \quad L_3 : \mathtt{b} = \textbf{true}; \quad \}$$

($\mathit{EXCHANGE}$ — подпрограмма обмена данными для печати)

Фрагмент СП для π, достижимый из начальных состояний
с произвольным начальным значением $\mathtt{b}$, может быть устроен так
(*для простоты считаем, что метка состояния — это оно само*):



Система переходов программы в окружении

Программа в распределённой системе может
взаимодействовать со своим **окружением** при помощи
разделяемых переменных, сообщений, сигналов, ...

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

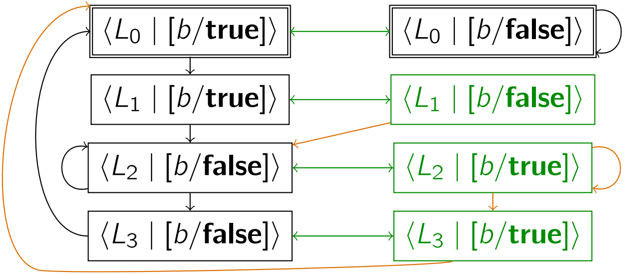
СП программы в окружении — это СП программы,
в которую добавлены переходы, отвечающие возможностям окружения

Система переходов программы в окружении

Пример

$$\textbf{while}\ (\textbf{true})\ \{ \quad L_0 : \textbf{while}\ (!\mathtt{b});\quad L_1 : \mathtt{b} = \textbf{false};\quad L_2 : \mathit{EXCHANGE}\quad L_3 : \mathtt{b} = \textbf{true};\quad \}$$

СП для программы доступа к сетевому принтеру в окружении, способном произвольно изменять значение регистра $\mathtt{b}$, может быть устроена так:



Параллельная композиция систем переходов

Наиболее популярный способ композиции СП параллельно выполняющихся программ — это асинхронная композиция (согласно семантике чередующихся вычислений; *interleaving*)

Это способ композиции устроен так:

- Состояние композиции компонентов представляет собой набор локальных частей их состояний и *включённую один раз* общую (разделяемую) часть
- Переход в композиции отвечает
  - произвольному выбору одного из компонентов и перехода в нём и
  - выполнению этого перехода с изменением локальной части состояния компонента и общей части состояния согласно устройсту СП компонента системы

В вычислении построенной так композиции произвольно чередуется выполнение переходов компонентов системы

Параллельная композиция систем переходов

Пример

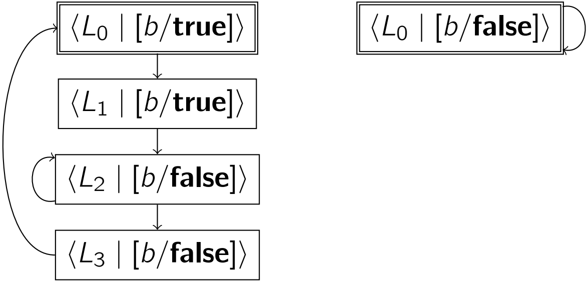
$$\pi_1 = \text{счёт} := \text{счёт} + 1\,000; \qquad \pi_2 = \text{счёт} := \text{счёт} + 1\,000\,000;$$

Достижимый фрагмент асинхронной композиции СП
этих двух модельных императивных программ
с общей переменной «счёт»
для интерпретации $Ar_{\mathbb{Z}}$ и начальной оценки $[\text{счёт}/0]$ устроен так:

$$\begin{tikzcd}
& \boxed{\boxed{\langle \pi_1, \pi_2 \mid [\text{счёт}/0]\rangle}} \arrow[dl] \arrow[dr] & \\
\boxed{\langle \emptyset, \pi_2 \mid [\text{счёт}/1\,000]\rangle} \arrow[dr] & & \boxed{\langle \pi_1, \emptyset \mid [\text{счёт}/1\,000\,000]\rangle} \arrow[dl] \\
& \boxed{\langle \emptyset, \emptyset \mid [\text{счёт}/1\,001\,000]\rangle} \arrow[loop below] &
\end{tikzcd}$$

Параллельная композиция систем переходов

Другой пример



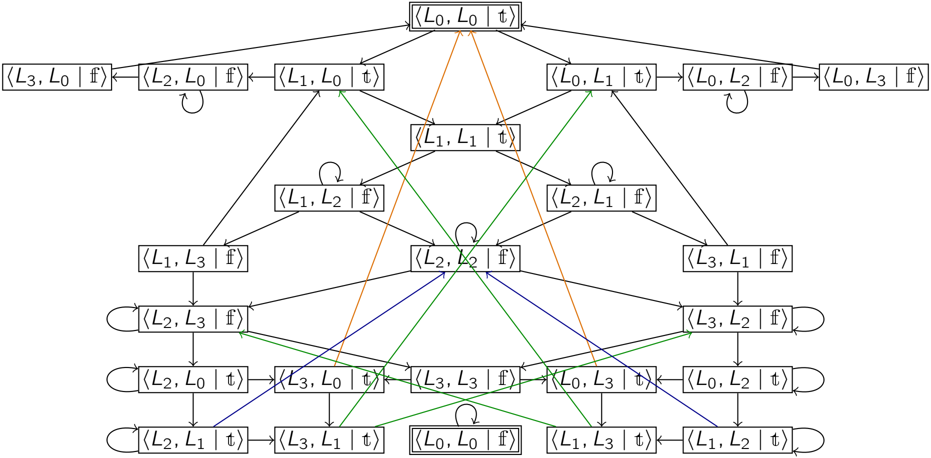
Предположим, что

- с сетевым принтером взаимодействуют две программы с одинаковыми СП (как изображено выше) и
- регистр b является общим для обеих программ

Тогда асинхронная композиция СП, отвечающая параллельному выполнению двух программ доступа к принтеру, устроена так ...

Параллельная композиция систем переходов

Другой пример



Согласно методу model checking,
построение и анализ композиции СП производятся автоматически,
так что «нечитаемость» композиции не считается недостатком