Формальные модели программ

Менее известные модели вычислений. Тезис Чёрча-Тьюринга. Теорема Райса о неразрешимости нетривиальных свойств.

Формальные системы. Истинность и ложность в формальных системах. Полнота и непротиворечивость. Теоремы Гёделя о неполноте. Неразрешимые задачи для формальных систем.

Теоретико-модельные и теоретико-доказательные подходы к верификации.

maxim.krivchikov@gmail.com

Материалы курса: https://maxxk.github.io/formal-models-2015/

Клеточный автомат

«Таблица» (любой размерности) из ячеек, каждая из которых может находиться в конечном числе состояний.

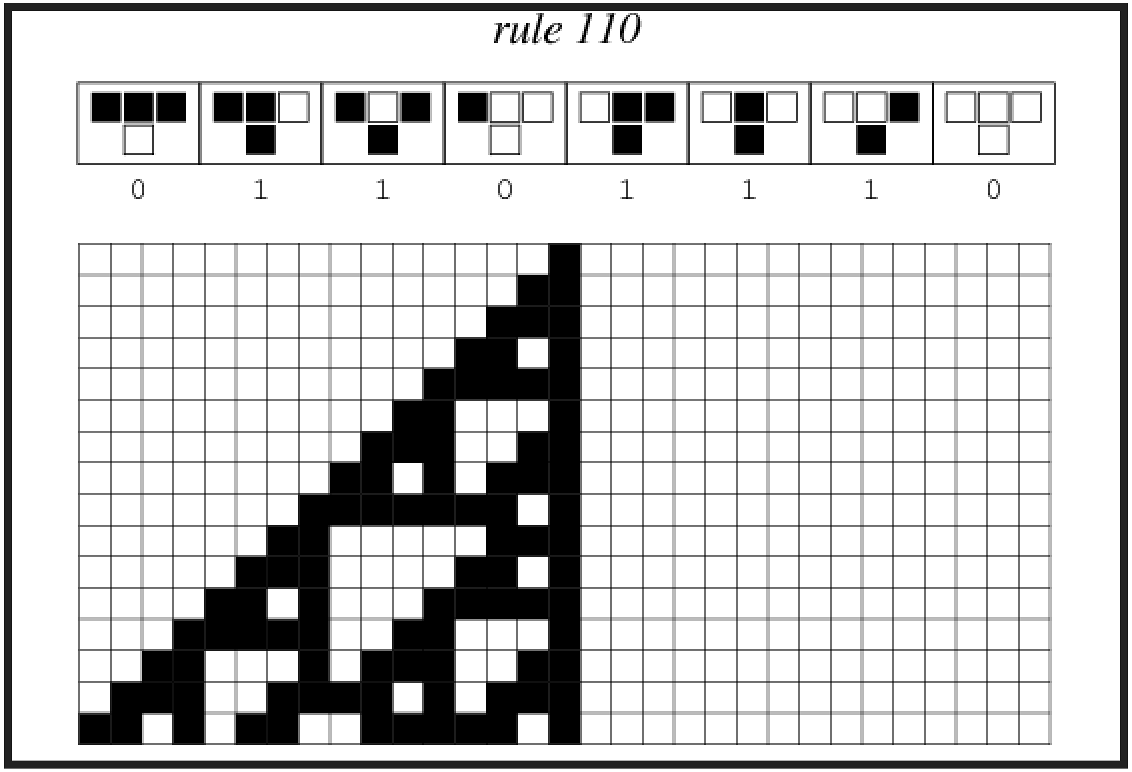
На каждом шаге следующее состояние каждой ячейки определяется по заданному фиксированному правилу, которое может использовать текущее состояние ячейки и текущие состояния ячеек в некоторой окрестности.

Примеры:

1. Игра «Жизнь» Конвэя (и ещё одна схема)
2. «Rule 110» — одномерный клеточный автомат

Rule 110

Из Wolfram MathWorld: http://mathworld.wolfram.com/Rule110.html



Существующее оборудование

1. URISC (Ultimate Restricted Instruction Set Computer) — машина с одной инструкцией. Для того, чтобы иметь возможность эмулировать машину Тьюринга, память должна быть бесконечной, а адреса и ячейки — произвольными целыми числами (возможно, неотрицательными)
   - `subleq a, b, c` — посчитать `M[b] - M[a]`, записать в `M[b]` и перейти на адрес `c`. В памяти хранятся тройки `a, b, c`
   - `dln a, b` — уменьшить на 1 значение `M[a]`; если получился `0`, перейти по адресу `b`

2. Механизм защиты памяти в x86-процессорах допускает реализацию `subleq` с помощью вложенных исключений page fault

3. Magic: The Gathering (и другие примеры на странице Accidentally Turing-complete)

Turing tar-pit

(«тьюрингова трясина»)

Десятое правило Гринспена

любая достаточно сложная программа на языках C или Fortran содержит самодельную, не имеющую формального описания, наводнённую багами медленную реализацию Common Lisp

Тьюрингова трясина

языки программирования, на которых можно написать машину Тьюринга, но очень сложно решать какие-то практичекие задачи (Brainfuck, Whitespace)

(развитие идеи) — чрезмерная универсализация разрабатываемой программы (например, язык конфигурации, который затем становится Тьюринг-полным; так получилось с SQL)

Теорема Райса о неразрешимости нетривиальных свойств программ

Исходная формулировка — в терминах частично-рекурсивных функций Клини.
Для любого нетривиального свойства частичных функций не существует общего эффективного метода разрешения того, вычисляет ли данный алгоритм частичную функцию с таким свойством.
Тривиальные свойства — свойства, которые выполняются для всех вычислимых функций (или не выполняются ни для одной).

Частичная функция из X в Y — функция, область определения которой не совпадает со всем X. Вычислимые функции частичны, т.к. можно представить машину Тьюринга, которая останавливается для некоторых входных значений и не останавливается для других.

Теорема Райса

Следствие для практики

Невозможно создать алгоритм, который определяет, обладает ли программа некоторым нетривиальным свойством.

$\Rightarrow$ невозожна автоматическая формальная верификация

В терминах распознающих машин Тьюринга

Пусть S — нетривиальное множество языков, т.е. существуют две машины Тьюринга — распознающая, что язык находится в S и распознающая, что язык не находится в S. Тогда для произвольной данной машины Тьюринга вопрос о том, принадлежит ли распознаваемый ей язык S является неразрешимым.

Теорема Райса

Схема доказательства

Сводим утверждение теоремы к проблеме останова. Пусть:

- a — строка-программа
- $a \mapsto \mathbf{F}_a$ — частичная функция, соответствующая программе a
- $a \mapsto P(a)$ — данный алгоритм, разрешающий некоторое нетривиальное свойство
- n — строка-программа, которая не завершается ни для какого входа
- $P(n) = 0$
- b — строка-программа, для которой $P(b) = 1$

Покажем, что из этих элементов можно создать алгоритм $H(a, i)$, решающий проблему останова для программы a на входе i.
Построим t — строку-программу, которая для некоторого j выполняет вычисление $\mathbf{F}_a(i)$, затем — $\mathbf{F}_b(j)$ и возвращает последний результат.

Тогда $P(t)$ решает проблему останова.

Программа Гильберта

Искомые свойства математики:

1. Полнота (мы можем переписыванием получить из аксиом все истинные утверждения математики)
2. Непротиворечивость (мы не можем получить переписыванием из аксиом ложное утверждение)
3. ~~Разрешимость (руководствуясь простым набором правил можно для любой формулы получить, выводима она или нет).~~

Формальная система

— конструкция, отражающая концепцию формализма: теории представляются в виде формул, которые можно менять по заданным правилам

Составные части:

1. Конечный *алфавит*, из символов которого составляются *формулы* — строки.
2. *Грамматика* — набор правил, описывающий как составить корректно сформированные формулы (*well-formed formulas*, **wf**). Обычно под грамматикой подразумевается процедура разрешения, является ли формула корректно сформированной.
3. *Схемы аксиом* — набор корректно сформированных формул (или схем формул — формул с «метапеременными», вместо которых можно подставлять произвольные формулы).
4. *Правила вывода* — правила, по которым из нескольких корректно сформированных формул, удовлетворяющих некоторой схеме, можно получить новую корректно сформированную формулу.

Пример: исчисление высказываний

(по лекциям Лупанова и Угольникова)

1. Алфавит:
   - *переменные* ($a, b, c, \ldots, a_1, b_1, c_1, \ldots$)
   - *логические операторы* ($\neg, \wedge, \vee, \oplus, \rightarrow, \leftrightarrow$)
   - *технические символы* (скобки — $()$, $[]$)
2. Грамматика: переменная **wf**; формула в скобках **wf**; отрицание — унарный префиксный оператор с высшим приоритетом связывания; остальные — бинарные операторы, приоритет и ассоциативность других операторов не определены, если не указано обратное (нужно всё брать в скобки)

Исчисление высказываний

3. Аксиомы (11 штук, далее ссылаемся с префиксом А):
   1. $A \to (B \to A)$,
   2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$,
   3. $(A \land B) \to A$,
   4. $(A \land B) \to B$,
   5. $[A \to B] \to [(A \to C) \to (A \to (B \land C))]$,
   6. $A \to (A \lor B)$,
   7. $B \to (A \lor B)$,
   8. $[A \to C] \to [(B \to C) \to ((A \lor B) \to C)]$,
   9. $(A \to B) \longrightarrow (\neg B \to \neg A)$,
   10. $A \to \neg\neg A$,
   11. $\neg\neg A \to A$ (аксиома двойного отрицания, double negation) или $A \lor \neg A$ (закон исключённого третьего, law of excluded middle, LEM)

Исчисление высказываний

4. Правило вывода (*modus ponens*, дословно «правило вывода», MP)

$$\frac{P \qquad P \to Q}{Q}$$

Интересуют нас общезначимые формулы — формулы, истинные при любых наборах переменных.

Определение. Формула *выводима*, если её можно построить из аксиом и правил вывода. *Вывод* — конечная последовательность формул, каждый из элементов которой является аксиомой, либо получается применением правила вывода к нескольким из предыдущих формул.

Для краткости правила вывода и вообще отношение выводимости записывают с помощью значка ⊢ (кстати говоря, ⊥ — «ложь», «перевёрнутый на 180» — «истина»)

$$(\mathbf{MP}) \quad P, P \to Q \vdash Q$$

Исчисление высказываний

Пример. $A \to A$ выводима. Приведём вывод (ссылаемся на предыдущие с префиксом F):

1. $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$ [A2: A,C := A; B := (A → A)]
2. $A \to ((A \to A) \to A)$ [A1: A := A, B := (A → A)]
3. $(A \to (A \to A)) \to (A \to A)$ [MP: P:= F2; (P → Q) := F1]
4. $A \to (A \to A)$ [A1: A, B := A]
5. $A \to A$ [MP: P := F4, (P → Q) := F3]

Исчисление высказываний

Основные свойства

Теорема. Любая выводимая формула общезначима.

Определение. Исчисление *непротиворечиво*, если не существует **wf**, выводимой формулы, для которой выводимо её отрицание.

Следствие. (о непротиворечивости исчисления высказываний) Исчисление высказываний непротиворечиво.

Определение. Исчисление *полно*, если в нём выводимы все истинные формулы.

Теорема. Исчисление высказываний полно — все общезначимые формулы в нём выводимы.

Определение. Система аксиом *независима*, если в ней нет аксиом, которые можно вывести из других входящих в систему.

Пример: исчисление предикатов

логика первого порядка

1. Алфавит — логические и не логические символы
    - логические: квантификаторы ∀, ∃; связки (→, ¬, возможно, дополнительные); технические знаки (скобки, пунктуация), переменные, символ равенства
    - не логические: n-арные предикатные символы функциональные символы
        - предикат это «высказывание об объектах», функция — «преобразование объектов»
2. Синтаксис: термы и формулы
    - термы — переменные или функции от термов со всеми аргументами
    - формулы — как в исчислении высказываний, а также:
        - предикатные символы от термов со всеми заданными аргументами
        - квантификаторы, связывающие переменную (∀x P(f(x)), ∃y Q(g(y)))

Логика первого порядка

3. Аксиомы
   1. A → (B → A)
   2. (A → (B → C)) → ((A → B) → (A → C))
   3. (¬A →¬B) → (B → A)
   4. ∀x.A(x) → A(y)
   5. A(x) → ∃y.A(y)
4. Правила вывода
   1. $$\frac{A \to B, A}{B}$$
   2. $$\frac{B \to A(x)}{B \to \forall x. A(x)}$$
      , где x — не свободная переменная B
   3. $$\frac{A(x) \to B}{\exists x. A(x) \to B}$$
      , где x — не свободная переменная B
   4. Переименование переменных.

Разрешимость логики первого порядка

Логика первого порядка неразрешима — задача сводится к проблеме разрешимости.

Истинность и ложность в формальной системе

Интерпретация логики первого порядка — соответствие предикатных и функциональных символов некоторым предикатам и функциям (а также множество допустимых значений переменных).

Теория первого порядка — собственный набор аксиом.

Модели формальной системы

Модель формальной системы — интерпретация, в которой все аксиомы общезначимы.

Если выводимая формула верна во всех моделях — она общезначима.
Если выводимая формула неверна во всех моделях — система противоречива.
Если для данная формула верна не во всех моделях, она невыводима и является независимой от системы аксиом.

Первая теорема Гёделя о неполноте

Если формальная система достаточно выразительна для описания элементарной арифметики, она не может быть одновременно непротиворечива и полна. Для непротиворечивой теории существуют утверждения (формулы), которые не могут быть ни выведены, ни опровергнуты.

Вторая теорема Гёделя о неполноте

Если в формальной системе можно представлять утверждения об арифметике и о формальной доказуемости, и если такая система включает формулу, представляющую непротиворечивость этой системы, то такая система противоречива.

(непротиворечивость формальной системы нельзя показать внутри самой этой системы)

Теоремы Гёделя о неполноте

Схемы доказательств и более подробная информация:
http://plato.stanford.edu/entries/goedel-incompleteness/

Программа Гильберта

Искомые свойства математики:

1. ~~Полнота (мы можем переписыванием получить из аксиом все истинные утверждения математики)~~
2. ~~Непротиворечивость (мы не можем получить переписыванием из аксиом ложное утверждение)~~
3. ~~Разрешимость (руководствуясь простым набором правил можно для любой формулы получить, выводима она или нет).~~

Теория моделей и теория доказательств

Теория моделей — изучает связь синтаксиса (формальных систем) и семантики (моделей для таких систем)

Теоретико-модельные подходы к формальной верификации: программа представляется в виде некоторой модели, допускающей исчерпывающую проверку интересующих свойств (enumerative and symbolic model checking). Обычно — автоматные модели, а также методы абстрактной интерпретации (символьные вычисления и т.п.).

Теория доказательств — изучает структуру формальных доказательств

Дедуктивные (теоретико-доказательные) подходы к формальной верификации — программа и свойства записываются в виде формул в формальной системе, доказательства строятся как вывод в формальной системе.