Математические методы верификации схем и программ

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

Блок 24

Символьные представления моделей

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

ВМК МГУ, 2025, сентябрь–декабрь

Вступление

Базовый алгоритм проверки моделей для CTL

- идеологически прост: сводит задачу проверки выполнимости формулы на модели к набору хорошо известных задач теории графов:
    - выделение компонент сильной связности,
    - проверка достижимости вершин
    - ...
- теоретически эффективен: его сложность полиномиальна с невысокой степенью (без учёта алгоритма вычисления н.к.с.с. — линейна) относительно размеров модели и формулы

Вступление

Но у базового алгоритма есть и недостатки

Для примера можно представить *относительно небольшую* систему из 20 процессов, в каждом из которых содержится 20 булевых переменных

В такой системе будет более 10^{120} состояний (*чтобы осознать, насколько это много: больше, чем гуго́л*)

В базовом алгоритме граф модели хранится **явно и полностью**, что приводит к неразумному и даже невозможному расходу памяти

А можно ли сократить расход памяти, представив модель более эффективно?

Вступление

Напоминание: конечная модель Крипке $M = (S, S_0, \to, L)$ — это

- конечное множество состояний S
- конечное множество начальных состояний S_0
- отношение переходов на паре конечных множеств состояний $\to$
- функция разметки, сопоставляющая каждому состоянию конечное множество L

Другое напоминание:

- $\mathbb{B} = \{0, 1\}$ (будем использовать 0 как синоним $\mathbb{f}$ и 1 как синоним $\mathbb{t}$)
- Булева функция местности n — это функция $f : \mathbb{B}^n \to \mathbb{B}$

Символьные представления

Символьные представления — это семейство эффективных представлений структур (графовых и не только), становящееся всё более популярным в практике «умного» программирования, и коротко можно описать его так:

1. Представляемый объект переписывается (кодируется) как совокупность множеств наборов нолей и единиц заданной длины
2. Каждое множество наборов нолей и единиц длины представляется булевой функцией, принимающей значение 1 на наборах этого множества
3. Выбирается подходящее представление получающихся булевых функций в зависимости от природы исходных объектов и от того, как с ними предполагается взаимодействовать (*этот пункт необязателен, пока речь не идёт о программной реализации*)
4. Символьное представление объекта — это совокупность представлений булевых функций, кодирующих объект, и реализация требуемых операций над объектом в терминах булевых функций

Символьное представление графа

Начнём с простого **примера**: опишем символьное представление конечного ориентированного графа $G = (V, E)$, где V — это конечное множество вершин и $E \subseteq V \times V$ — множество дуг (двуместное отношение на множестве вершин)

Для этого научимся символьно представлять конечные множества (как здесь V) и конечные двуместные отношения на представленных символьно множествах (как здесь E)

Для начала представим символьно множество V

Сопоставим элементам V уникальные числа, их номера:

- если нумерация уже задана, то используем её,
- а иначе пронумеруем элементы по своему усмотрению

Например, $V = \{v_0, v_1, \ldots, v_n\}$, и вершине v_i отвечает номер $\mathfrak{n}(v_i) = i$ (*дальше в примерах будем использовать эту нумерацию*)

Как-либо выберем такое количество разрядов k, в которое вмещается двоичная запись наибольшего рассматриваемого числа

Например, $k = \lfloor \log_2(n) \rfloor + 1$

Символьное представление графа

Замéним каждый номер i на его двоичную запись $(i)_2^k$ в k разрядах

В результате получим множество $V_{\mathbb{B}}$ наборов нолей и единиц длины k, представляющее множество V

Например, $V_{\mathbb{B}} = \{(0)_2^k, (1)_2^k, (2)_2^k, \ldots, (n)_2^k\}$

Характеристическая функция множества $X \subseteq \mathbb{B}^k$ — это функция $f : \mathbb{B}^k \to \mathbb{B}$, такая что $f(\widetilde{\alpha}) = 1 \Leftrightarrow \widetilde{\alpha} \in X$

Перейдём от множества $V_{\mathbb{B}}$ к его характеристической функции f_V

Например, для $V = \{v_0, v_1, \ldots, v_n\}$ эта функция задаётся так:

$$f_V((i)_2^k) = 1 \Leftrightarrow i \leq n$$

Будем считать, что переменные этой функции устроены так:

$$f_V(x_{k-1}, \ldots, x_1, x_0)$$

Представим как-либо эту функцию — (*но так обычно не делают, но*) **например**, в виде совершенной ДНФ

Такое представление множества V (какая-либо форма представления функции f_V) будем называть стандартным символьным

Символьное представление графа

Теперь научимся представлять символьно отношение $E \subseteq V \times V$, имея нумерацию элементов V

Перейдём от множества пар E к множеству номеров элементов этих пар: $E_{\mathbb{B}} = \{(\mathfrak{n}(v), \mathfrak{n}(w)) \mid (v, w) \in E\} \subseteq \mathbb{B}^k \times \mathbb{B}^k$

Характеристическая функция двуместного отношения $R \subseteq \mathbb{B}^k \times \mathbb{B}^k$ — это функция $f : \mathbb{B}^{2k} \to \mathbb{B}$, такая что $f(\widetilde{\alpha}, \widetilde{\beta}) = 1 \Leftrightarrow (\widetilde{\alpha}, \widetilde{\beta}) \in R$

Перейдём от отношения $E_{\mathbb{B}}$ к его характеристической функции f_E

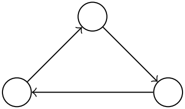
Будем считать, что переменные этой функции устроены так:

$$f_E(x_{k-1}, \dots, x_1, x_0, x'_{k-1}, \dots, x'_1, x'_0)$$

Какое-либо представление этой функции будем называть стандартным символьным представлением отношения E

Символьное представление графа

Например:



Закодируем вершины числами 0 (левая), 1 (верхняя) и 2 (правая)

Запишем эти числа двоично в двух разрядах: $(0)_2^2 = (00)$, $(1)_2^2 = (01)$, $(2)_2^2 = (10)$

Характеристическую функцию множества $\{(00), (01), (10)\}$, отвечающего вершинам, можно представить формулой

$$\neg(x_1 \mathbin{\&} x_0)$$

Характеристическую функцию отношения

$$\{((00), (01)), ((01), (10)), ((10), (00))\},$$

отвечающего дугам, можно представить формулой

$$\neg x_1 \mathbin{\&} \neg x_0 \mathbin{\&} \neg x_1' \mathbin{\&} x_0' \vee \neg x_1 \mathbin{\&} x_0 \mathbin{\&} x_1' \mathbin{\&} \neg x_0' \vee x_1 \mathbin{\&} \neg x_0 \mathbin{\&} \neg x_1' \mathbin{\&} \neg x_0'$$

Символьное представление графа — это пара написанных выше формул

Символьное представление модели Крипке

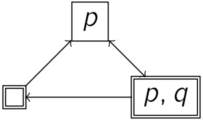
Конечную модель Крипке $M = (S, S_0, \to, L)$ над множеством AP можно представить как набор следующих конечных множеств и отношений:

- S — конечное множество
- $S_0 \subseteq S$
- $\to \subseteq S \times S$
- Множество $S_p = \{s \mid s \in S,\ p \in L(s)\} \subseteq S$ для каждого $p \in$ AP

Символьным представлением модели Крипке будем называть совокупность стандартных представлений для S, S_0, $\to$ и S_p для каждого $p \in$ AP для общей нумерации элементов S и общего числа разрядов k

Символьное представление модели Крипке

Например (AP $= \{p, q\}$)



Пример символьного представления этой модели Крипке для нумерации состояний 0 (левое), 1 (верхнее), 2 (правое) и двух разрядов:

- Множество состояний: $\neg(x_1 \mathbin{\&} x_0)$
- Множество начальных состояний: $\neg x_0$
- Отношение переходов:
$$\neg x_1 \mathbin{\&} (\neg x_0 \mathbin{\&} x_1' \mathbin{\&} \neg x_0' \vee \neg x_0 \mathbin{\&} x_1' \mathbin{\&} x_0') \vee x_1 \mathbin{\&} \neg x_0 \mathbin{\&} \neg x_1'$$
- Множество S_p: $x_0 \oplus x_1$
- Множество S_q: $x_1 \mathbin{\&} \neg x_0$

Операции над символьными представлениями

Мало уметь записывать множества как булевы функции, обычно требуется уметь их строить и применять к ним теоретико-множественные операции

Стандартное представление $\emptyset$: $\mathbb{f}$

Стандартное представление $\{i\}$ — это элементарная конъюнкция, представляющая характеристическую функцию этого множества

Если φ_1 — представление множества S_1 и φ_2 — представление множества S_2, то:

- $S_1 \cup S_2$ представляется как $\varphi_1 \vee \varphi_2$
- $S_1 \cap S_2$ представляется как $\varphi_1 \,\&\, \varphi_2$
- $S_1 \setminus S_2$ представляется как $\varphi_1 \,\&\, \neg\varphi_2$

Откуда берутся символьные представления (пример)

Чтобы развеять впечатление о том, что символьное представление модели Крипке — это «неестественная» конструкция, необходимая только в технических целях, приведём пример, когда такое представление возникает прежде «явного» (*а ещё лучше это будет заметно в обязательном задании по средству NuSMV*)

Представим себе императивную программу над переменными $V = \{v_1, \ldots, v_n\}$, каждая из которых принимает значения из конечного множества D (домена)

Считая переменные упорядоченными по номерам, будем считать состоянием данных набор значений $(d_1, \ldots, d_n) \in D^n$

Состоянием управления будем считать значение счётчика команд — особой переменная pc, принимающей значения из конечного множества D_{pc} и значение которой — это *номер* команды, которая будет выполняться следующей

Тогда состояния вычисления — это элементы множества $D^n \times D_{pc}$

Откуда берутся символьные представления (пример)

Каждое «элементарное» утверждение о значениях переменных и счётчике команд: $v_i = k$, $v_i = v_j$, $pc = k$ — можно трактовать как сокращение для булевой формулы, описывающей такое равенство для двоичных записей значений переменных

Например, если переменной v_2 сопоставлены переменные x_3, x_4, x_5 (от младшего разряда к старшему), то выражение $v_2 = 3$ — это сокращение для формулы $\neg x_5 \,\&\, x_4 \,\&\, x_3$

Остальные («производные») соотношения между значениями переменных и счётчика команд аналогично можно считать сокращениями для соответствующих булевых формул

Состоянию данных и состоянию вычисления естественно сопоставляются конъюнкции таких выражений

Например, состоянию данных $(2, 3, 5)$ для переменных v_1, v_2, v_3 соответствует формула $\varphi = (v_1 = 2) \,\&(v_2 = 3) \,\&(v_3 = 5)$, а этой оценке и значению счётчика команд 5 — формула $\varphi \,\&(pc = 5)$

Откуда берутся символьные представления (пример)

Если состояние данных, с которого программа начинает вычисление, однозначно определено, то множество начальных состояний модели задаётся формулой, отвечающей этому состоянию и начальному значению счётчика команд (обычно — 0)

Если нет, то семейство допустимых оценок нередко *естественно* записывается в виде формулы (предусловия)

Самый простой и при этом полезный вид атомарных высказываний для рассматриваемой программы — это высказывания вида $v_i = k$

Множество состояний S_p, которые размечены высказыванием $p = (v_i = k)$, задаётся формулой $v_i = k$

Откуда берутся символьные представления (пример)

Множество переходов, определяемых для команды C операционной семантикой, как правило, легко выражается в виде формулы ψ_C над двумя комплектами переменных: «обычные» для текущих значений переменных и текущего значения счётчика команд, и «штрихованные» — для значений переменных и счётчика команд после выполнения C

Например, переходы для команды $\mathtt{x} := \mathtt{y} + 1$; программы над $\{\mathtt{x}, \mathtt{y}\}$ задаются формулой $(\mathtt{x}' = \mathtt{y} + 1)\,\&(\mathtt{y}' = \mathtt{y})\,\&(\mathtt{pc}' = \mathtt{pc} + 1)$:

- Следующее значение переменной $\mathtt{x}$ — это текущее значение переменной $\mathtt{y}$ плюс один
- Значение переменной $\mathtt{y}$ не изменяется
  - **Важно!** — если вычеркнуть множитель «$\mathtt{y}' = \mathtt{y}$», то это будет означать «выбор следующего значения $\mathtt{y}$ не влияет на истинность формулы», то есть «значение $\mathtt{y}$ может произвольно измениться при выполнении команды»
- Управление передаётся команде со следующим номером

Откуда берутся символьные представления (пример)

Для примера рассмотрим программу $\mathtt{x} := \mathtt{x} + 1;\ \mathtt{y} := \mathtt{x} + \mathtt{y};$ над переменными $\mathtt{x}$, $\mathtt{y}$ с доменом $\mathbb{B}$, и условимся, что значение счётчика команд 0 указывает на первое присваивание, 1 — на второе, и 2 означает, что программа завершилась

Счётчику команд $\mathtt{pc}$ сопоста́вим булевы переменные $\mathtt{pc}_0$ (младший разряд) и $\mathtt{pc}_1$ (старший разряд)

Тогда:

- Множество состояний представляется формулой $\neg(\mathtt{pc} = 3)$
- Множество переходов представляется формулой
$$\begin{aligned}&(\mathtt{pc} = 0)\,\&(\mathtt{x}' \oplus \mathtt{x})\,\&(\mathtt{y}' \leftrightarrow \mathtt{y})\,\&(\mathtt{pc}' = \mathtt{pc} + 1)\vee\\ &(\mathtt{pc} = 1)\,\&(\mathtt{x}' \leftrightarrow \mathtt{x})\,\&(\mathtt{y}' \leftrightarrow(\mathtt{x} \oplus \mathtt{y}))\,\&(\mathtt{pc}' = \mathtt{pc} + 1)\vee\\ &(\mathtt{pc} = 2)\,\&(\mathtt{x}' \leftrightarrow \mathtt{x})\,\&(\mathtt{y}' \leftrightarrow \mathtt{y})\,\&(\mathtt{pc}' = \mathtt{pc})\end{aligned}$$
- Утверждения о значениях $\mathtt{pc}$ «раскодируются» так:
$$\begin{aligned}&(\mathtt{pc} = 0) \sim (\neg\mathtt{pc}_1\,\&\,\neg\mathtt{pc}_0) \quad (\mathtt{pc} = 1) \sim (\neg\mathtt{pc}_1\,\&\,\mathtt{pc}_0)\\ &(\mathtt{pc} = 2) \sim (\mathtt{pc}_1\,\&\,\neg\mathtt{pc}_0) \quad (\mathtt{pc} = 3) \sim (\mathtt{pc}_1\,\&\,\mathtt{pc}_0)\\ &(\mathtt{pc}' = \mathtt{pc}) \sim ((\mathtt{pc}_1 \leftrightarrow \mathtt{pc}_1')\,\&(\mathtt{pc}_0 \leftrightarrow \mathtt{pc}_0'))\\ &(\mathtt{pc}' = \mathtt{pc} + 1) \sim ((\mathtt{pc}_1' \leftrightarrow(\mathtt{pc}_1 \oplus \mathtt{pc}_0))\,\&(\mathtt{pc}_0' \oplus \mathtt{pc}_0))\end{aligned}$$

Откуда берутся символьные представления (пример)

Для самостоятельного размышления:

а как выглядят формулы, задающие семантику всех других команд императивных программ из блока 3 для «естественной» арифметики с переполнением?

(*Это осознать не очень сложно, но весьма полезно*)