Математические методы верификации схем и программ

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

Блок 19

Символьные представления моделей

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

ВМК МГУ, 2022/2023, осенний семестр

Вступление

Базовый алгоритм model checking 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: линеен относительно размера модели и относительно размера 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 числá гугóл*)

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 S наборов нолей и единиц длины n расценивается как характеристическое множество n-местной булевой функции $\mathfrak{f}_S$: $\mathfrak{f}_S(\widetilde{\alpha}) = 1 \Leftrightarrow \widetilde{\alpha} \in S$
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)

Пронумеруем вершины графа: $V = \{v_0, v_1, , \ldots, v_n\}$

Сопоста́вим каждой вершине уникальное число — например, её номер: $V_{\mathbb{N}} = \{0, 1, \ldots, n\}$

Как-либо выберем такое количество бит k, в которое вмещается двоичная запись наибольшего рассматриваемого числа — например, $k = \lfloor \log_2(n) \rfloor + 1$

Символьное представление графа

Заме́ним каждое число i на его двоичную запись $(i)_2^k$ в k битах

В результате получим множество наборов нолей и единиц длины k, представляющее множество V:
$V_{\mathbb{B}} = \{(00\ldots00), (00\ldots01), (00\ldots10), \ldots, (n)_2^k\}$

Перейдём от этого множества к булевой функции: $f_{V_{\mathbb{B}}}((i)_2^k) \Leftrightarrow i \leq n$

Будем считать, что это функция над переменными $x_0, \ldots, x_{k-1}$, от младшего (правого) бита двоичной записи к старшему (левому)

Представим как-либо эту функцию — например (*но так обычно не делают*), в виде совершенной ДНФ: $\varphi_V = \neg x_{k-1} \,\&\, \ldots \,\&\, \neg x_1 \,\&\, \neg x_0 \vee \neg x_{k-1} \,\&\, \ldots \,\&\, \neg x_1 \,\&\, x_0 \vee \neg x_{k-1} \,\&\, \ldots \,\&\, x_1 \,\&\, \neg x_0 \vee \neg x_{k-1} \,\&\, \ldots \,\&\, x_1 \,\&\, x_0 \vee \ldots$

Такое представление множества (какая-либо форма задания соответствующей булевой функции) будем называть стандартным символьным

Символьное представление графа

Отношение $E \subseteq V \times V$ перепишем как соответствующее отношение $E_{\mathbb{B}} \subseteq \mathbb{B}^k \times \mathbb{B}^k$

Перейдём от этого отношения к его характеристической функции: $(2k)$-местной булевой функции $\mathfrak{f}_E$, такой что $f(\widetilde{\alpha}, \widetilde{\beta}) = 1 \Leftrightarrow (\widetilde{\alpha}, \widetilde{\beta}) \in E$

Будем считать, что это функция над двумя комплектами переменных: $x_0, \ldots, x_{k-1}$ для первого набора (начала дуги) и $x'_0, \ldots, x'_{k-1}$ для второго (конца дуги)

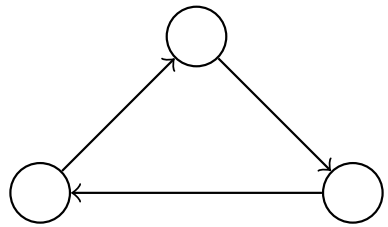
Например, дуга $(0010) \to (1011)$ может быть представлена как конъюнкция $\neg x_3 \,\&\, \neg x_2 \,\&\, x_1 \,\&\, \neg x_0 \,\&\, x'_3 \,\&\, \neg x'_2 \,\&\, x'_1 \,\&\, x'_0$

А отношение $E_{\mathbb{B}}$ можно представить как дизъюнкцию представлений дуг

Такое представление двуместных отношений (какая-либо форма задания соответствующей булевой функции над двумя комплектами переменных) будем называть стандартным символьным

Символьное представление графа

Например:



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

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

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

$$\neg(x_0 \,\&\, x_1)$$

Характеристическую функцию отношения

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

отвечающего дугам, можно представить формулой

$$\neg x_1 \,\&\, \neg x_0 \,\&\, \neg x_1' \,\&\, x_0' \vee x_1 \,\&\, \neg x_0 \,\&\, x_1' \,\&\, \neg x_0' \neg x_1 \,\&\, \vee x_0 \,\&\, \neg x_1' \,\&\, \neg x_0'$$

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:

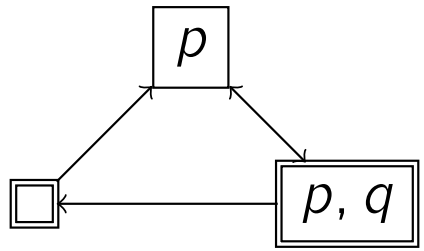
- S — конечное множество, самое большое по включению множество
- S_0 — конечное множество, $S_0 \subseteq S$
- $\to$ — двуместное отношение, $\to \subseteq S \times S$
- Для каждого атомарного высказывания p, $p \in \mathrm{AP}$, — множество всех вершин, в которых p истинно:

$$S_p = \{s \mid s \in S,\ p \in L(s)\}$$

Символьным представлением модели Крипке будем называть совокупность стандартных представлений S, S_0, $\to$ и S_p для каждого $p \in \mathrm{AP}$ для заданного подходящего числа бит k

Символьное представление модели Крипке

Например ($\mathrm{AP} = \{p, q\}$)



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

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

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, отвечающая i

Если φ_1 — представление множества S_1 и φ_2 — представление множества φ_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)

Откуда берутся символьные представления (пример)

Оценка переменных — это отображение $\xi : V \to D$, которым задаётся набор значений переменных в заданный момент времени выполнения программы

Оценку переменных ξ иногда записывают в такой форме (*табличной*): $[v_1/\xi(v_1), \ldots, v_n/\xi(v_n)]$

Также у программы есть счётчик команд — особая переменная pc, принимающая значения из конечного множества D_{pc} и задающая своим значением команду, которая будет выполняться следующей

Состояниями такой программы будут элементы множества $D^n \times D_{pc}$ — например, это можно воспринимать как представление состояния вычисления императивной программы согласно модели из начала этого курса

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 бита к старшему), то выражение $v_2 = 3$ — это сокращение для формулы $x_5 \,\&\, \neg x_4 \,\&\, \neg x_3$

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

Оценке переменных и состоянию программы естественным образом сопоставляются конъюнкции таких выражений

Например, оценке $[v_1/2, v_2/3, v_3/5]$ 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 значения переменных в начале выполнения программы однозначно определены (оценкой ξ), то множесто начальных состояний модели задаётся формулой, отвечающей оценке ξ 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 «следующее значение переменной $\mathtt{y}$ любым», 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 бит) и $\mathtt{pc}_1$ (старший бит)

Тогда:

- Множество состояний представляется формулой $\neg(\mathtt{pc} = 3)$
- Множество переходов представляется формулой

$$(\mathtt{pc} = 0)\,\&(\mathtt{x}' \oplus \mathtt{x})\,\&(\mathtt{y}' \leftrightarrow \mathtt{y})\,\&(\mathtt{pc}' = \mathtt{pc} + 1) \vee$$
$$(\mathtt{pc} = 1)\,\&(\mathtt{x}' \leftrightarrow \mathtt{x})\,\&(\mathtt{y}' \leftrightarrow (\mathtt{x} \oplus \mathtt{y})\,\&(\mathtt{pc}' = \mathtt{pc} + 1) \vee$$
$$(\mathtt{pc} = 2)\,\&(\mathtt{x}' \leftrightarrow \mathtt{x})\,\&(\mathtt{y}' \leftrightarrow \mathtt{y})\,\&(\mathtt{pc}' = \mathtt{pc})$$

- Утверждения о значениях $\mathtt{pc}$ «раскодируются» так:

$$(\mathtt{pc} = 0) \sim (\neg\mathtt{pc}_1\,\&\,\neg\mathtt{pc}_0) \quad (\mathtt{pc} = 1) \sim (\neg\mathtt{pc}_1\,\&\,\mathtt{pc}_0)$$
$$(\mathtt{pc} = 2) \sim (\mathtt{pc}_1\,\&\,\neg\mathtt{pc}_0) \quad (\mathtt{pc} = 3) \sim (\mathtt{pc}_1\,\&\,\mathtt{pc}_0)$$
$$(\mathtt{pc}' = \mathtt{pc}) \sim ((\mathtt{pc}_1 \leftrightarrow \mathtt{pc}_1')\,\&(\mathtt{pc}_0 \leftrightarrow \mathtt{pc}_0'))$$
$$(\mathtt{pc}' = \mathtt{pc} + 1) \sim ((\mathtt{pc}_1' \leftrightarrow (\mathtt{pc}_1 \oplus \mathtt{pc}_0))\,\&(\mathtt{pc}_0' \oplus \mathtt{pc}_0))$$

Откуда берутся символьные представления (пример)

Для самостоятельного размышления: а как выглядят формулы, задающие семантику **всех** команд императивных программ из блока 2 для «естественной» арифметики с переполнением?

(*Это не очень сложно, но полезно*)