Обзор статьи

Никита Костливцев

Февраль 2018

1 Описание задачи

Здесь представлен обзор статьи «Minimal Synthesis of String To String Functions From Examples». В данной статье описывается один из способов решения задачи генерации программы по предоставленным входным-выходным данным. Входные и выходные данные будем представлять в виде строк. Авторы статьи решают задачу путем генерации выходного детерминированного автомата (Output-deterministic automaton или ODA), согласующегося с входными/выходными данными. ODA представляет из себя детерминированный конечный автомат, алфавит которого составляет декартово произведение входного и выходного алфавитов, а также со следующим ограничением. Находясь в некотором состоянии, данный автомат принимает следующие символы входной и выходной строки и выдает состояние, в которое перейдет автомат. Нужно иметь ввиду, что данный автомат является детерминированным только относительно своего алфавита. Если же алфавит ODA сузить на входной алфавит, то он будет являться недетерминированным. Но из соображений здравого смысла в ODA по любой входной строке хотелось бы получать единственную выходную строку. Таким образом уже видны некоторые ограничения описанного в статье решения: входная и выходная строки должны иметь одинаковую длины, а также каждая входная строка должна быть сопоставлена единственной выходной строке. Далее авторы статьи предлагают генерировать минимальный полный ODA, согласующийся с входными/выходными данными, что является хорошим способом обобщения входных-выходных данных.

Но генерация минимального ODA, согласующегося с входными/выходными данными, как было доказано в статье, является NP-полной задачей. А именно, данную задачу можно свести к проблеме выбора: существует ли ODA, согласующийся с входными/выходными данными, в котором ровно k состояний? В статье было показано, что необходимо рассмотреть не более линейного от размера входных данных количество состояний. Таким образом, если доказать, что проблема выбора является NP-полной, то и исходная задача будет являться NP-полной.

2 Формальная постановка задачи

Теперь опишем задачи более формально.

Определение 1. *Алфавит* Σ *– не пустое конечное множество.*

Определение 2. Σ^n *– множество слов длины* n *из алфавита* Σ*.*

Определение 3. $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$*.*

Определение 4. $|.| : \Sigma^* \to \mathbb{Z}^+$*.* $|u| = n$*, если* $u \in \Sigma^n$*.*

Определение 5. *Недетерминированный конечный автомат – это кортеж* $(\Sigma, Q, q_init, \delta, F)$*, где* Σ *– алфавит автомата,* Q *– конечное множество состояний,* $q_init \in Q$ *– начальное состояние,* $\delta \subseteq Q \times \Sigma \times Q$ *– функция перехода,* F *– принимающие состояния.*

Определение 6. $\mathcal{L}(A)$ *– множество всех слов, которые принимает автомат* A*.*

Определение 7. *Автомат называется недвойственным, если любое слово принимается автоматом не более чем одним состоянием.*

Определение 8. *Автомат называется детерминированным, если* $\forall q_1 \in Q, \forall a \in \Sigma \ \exists! q_2 \in Q$*, что* $(q_1, a, q_2) \in \delta$

Определение 9. $|A| = n$*, если автомат* A *имеет* n *состояний.*

Определение 10. *Пусть* $u = u_1...u_n$*,* $v = v_1...v_n$*.*
Тогда $u * v = (u_1, v_1)...(u_n, v_n)$*.*

Определение 11. *Пусть* Σ *– алфавит входных слов,* Γ *– алфавит выходных слов. Выходной детерминированный автомат(ODA) – конечный детерминированный автомат над алфавитом* $\Sigma \times \Gamma$*, для которого выполняется* $\forall u \in \Sigma^*, \forall v_1, v_2 \in \Gamma^*$*, если* $u * v_1 \in \mathcal{L}(A), u * v_2 \in \mathcal{L}(A)$ *выполнено* $v_1 = v_2$*.*

Определение 12. *Можно заметить, что для ODA существует частичная функция* $\overline{A} : \Sigma^* \to \Gamma*$*, что* $\forall u * v \in \mathcal{L}(A)$ $\overline{A}(u) = v$*. Тогда ODA называется полным, если полна* $\overline{A}$*.*

Обозначим формально задачи:

Задача 1. *Пусть* $E \subseteq \Sigma^* \times \Gamma^*$ *– множество входных-выходных строк. Найти полный ODA, согласующийся с* E*, размер которого минимален.*

Задача 2. *Пусть* $E \subseteq \Sigma^* \times \Gamma^*$ *– множество входных-выходных строк. Найти полный ODA, согласующийся с* E*, размер которого не привышает заданного* n*.*

В статье было приведено три доказательства NP-полноты для трех разных случаев. Все эти случаи доказывались по одному принципу – сведением этих задач к задаче One-In-Three Sat, которая является NP-полной.

Задача 3. *Пусть дано множество из переменных V и множество троек переменных $C \subseteq V^3$. Существует ли отображение $f : V \to 0, 1$ такое, что для каждой тройки $(x, y, z) \in C$ только одна из переменных x, y, z отображается в 1 под действием отображения f? Данная задача называется One-In-Three Sat задачей.*

Теорема 1. *Задача 2 является NP-полной, когда количество состояний фиксировано, выходной алфавит Γ фиксирован, и существует единственный входной/выходной пример.*

Теорема 2. *Задача 2 является NP-полной, когда входной алфавит Σ, выходной алфавит Γ фиксированы, и существует единственный входной/выходной пример.*

Заметим, что оставшийся случай, когда фиксированы оба алфавита и количество состояний является тривиальным и не является NP-полной задачей. В данном случае необходимо просто перебрать всевозможные автоматы и проверить их на согласование с примерами и полноту, что можно сделать за полиномиальное время.

Также в статье авторы доказали даже более сильное утверждение: было доказано, что даже если входные/выходные примеры имеют одинаковую длину, задача все равно остается NP-полной.

Определение 13. *Назовем недетерминированный конечный автомат $A = (\Sigma, Q, q_init, \delta, F)$ l-слоистым для $l \in \mathbb{N}$, если автомат принимает только слова длины l, то есть $\mathcal{L}(A) \subseteq \Sigma^l$. Тогда такой автомат будет являться l-полным, если областью определения функции $\overline{A}$ будет являться Σ^l.*

Задача 4. *Пусть Σ – входной алфавит, Γ – выходной алфавит и $l \in \mathbb{N}$. Тогда $u_1 * v_1, u_2 * v_2 ... u_k * v_k$ – множество входных/выходных примеров, таких что $u_i \in \Sigma^l$ и $v_i \in \Gamma^l$*

Существует ли l-слоистый и l-полный ODA, который принимает все $u_i * v_i$ $1 \leq i \leq k$ и имеет не более n состояний?

Теорема 3. *Задача 4 является NP-полной, если входной алфавит Σ и выходной алфавит Γ являются фиксированными.*

Далее, чтобы полностью закончить доказательство, что и Задача 1 является NP-полной авторы доказали лемму.

Лемма 1. *Пусть $E \subseteq (\Sigma \times \Gamma)*$ – корректные множество входных/выходных примеров. Тогда существует полный ODA, согласующийся с E, в котором не более $2 + \Sigma_{w \in E}|w|$ состояний.*

Таким образом, чтобы решить Задачу 1, мы можем решить Задачу 2 не более линейного от входных данных количества раз. Это означает, что Задача 1 тоже является NP-полной.

3 Решение с помощью SMT-решателя

Теперь рассмотрим решение авторов статьи с помощью SMT-решателя. Пусть E – набор входных/выходных примеров. Пусть $\phi_{E,k}$ – формула, которая выполняется тогда и только тогда, когда существует полный ODA с k состояниями, согласующийся с E. Тогда будем проверять с помощью SMT-решателя выполнимость формулы $\phi_{E,k}$. Свободными переменными в $\phi_{E,k}$ будут функции, описывающие детерминированный конечный автомат A с k вершинами, а именно: $\delta : Q \times (\Sigma \times \Gamma) \to Q$ – функция, описывающая переходы в A, $isFinal : Q \to 0, 1$ – функция, определяющая принимающие состояния в A и δ_{in} – отображение, которое будет являться проекцией δ на входной алфавит Σ.

Разобьем формулу $\phi_{E,k}$ на несколько составных частей.

$$\phi_{E,k} = AcceptExamples \ \& \ Projection \ \& \ Unambiguous \ \& \ Total. \quad (1)$$

Формула $AcceptExamples$ проверяет, что каждый входной/выходной пример будет приниматься автоматом. Или же каждый их пробег в автомате будет заканчиваться в состоянии q, что

$$isFinal(q) = 1 \quad (2)$$

Формула $Projection$ проверяет, что δ_{in} действительно является проекцией δ на алфавит Σ.

Формула $Unambiguous$ проверяет, что проекция автомата на алфавит Σ не является двойственным. Сделать это не прибегая к использованию кванторов можно следующим образом. Будем строить множество пар замкнутое в себе относительно операции добавления новой пары. А именно, эта операция берет любую пару (q_1, q_2) из множества, просматривает все символы a из алфавита Σ добавляет во множество новую пару (q_3, q_4) такую, что $\delta_{in}(q_1, a, q_3)$ и $\delta_{in}(q_2, a, q_4)$. Такое множество начинаем строить таким образом: просмотрим все достижимые состояния q в графе, просмотрим все символы перехода a из них и добавим в множество пары (q_1, q_2) такие, что $\delta_{in}(q, a, q_1)$ и $\delta_{in}(q, a, q_2)$. Таким образом, чтобы проверить недвойственность, нужно просмотреть все пары множества и проверить, что для любой пары (q_1, q_2) не более чем одно состояние из пары является принимающим или должно выполняться

$$\neg(isFinal(q_1) = 1 \ \& \ isFinal(q_2) = 1). \quad (3)$$

Формула $Total$ проверяет полноту автомата A. Положим A' – проекция A на алфавит Σ. Утверждается следующее, чтобы A был полным необходимо и достаточно, чтобы A' принимал все слова из Σ^l для всех $0 \leq l \leq |Q|$. Можно посчитать, сколько слов длины l принимает каждое из состояний. Тогда автомат будет являться полным, если для каждого $0 \leq l \leq |Q|$

$$\Sigma_{q \in Q} isFinal(q) * c_{l,q} = |\Sigma|^l \quad (4)$$

, где $c_{l,q}$ – количество слов длины l, оканчивающихся в состоянии q.

4 Решение с помощью SAT-решателя

Теперь приведем решение, которое использует SAT-решатель. Обозначим за Q множество состояний, за $E = \{ex_1, ex_2...ex_{|E|}\}$ – множество входных-выходных примеров. Как и в SMT-решателе, у нас будут переменные $\delta(i, a, j)$, где $i, j \in Q$, $a \in (\Sigma \times \Gamma)$, означающие, что между состоянием i и состоянием j есть переход по символу a, если $\delta(i, a, j) = 1$. Также есть переменные $\delta_{in}(i, b, j)$, где $i, j \in Q$, $a \in \Sigma$, которые являются проекциями переходов δ на алфавит Σ. И также, как в SMT, есть переменные $isFinal(i)$, где $i \in Q$, означающие, что i – принимающее состояние, если $isFinal(i) = 1$. (В дальнейшем «добавление булевой формулы» будет означать, что следующую формулу мы приконкатенируем к окончательной формуле при помощи конъюнкции «$\wedge$». Изначально окончательная формула будет равна $\mathbf{1}$).

Для начала добавим следующие булевы формулы:

Пусть $i, j, k \in Q, j \neq k$, $a \in (\Sigma \times \Gamma)$. Тогда

$$\neg\delta(i, a, j) \vee \neg\delta(i, a, k) \tag{5}$$

Это ограничивает автомат таким образом, чтобы по символу a было не более одного перехода.

$$\phi_{E,k} = AcceptExamples \wedge Projection \wedge Unambiguous \wedge Total \tag{6}$$

Каждую из составных частей $\phi_{E,k}$ приведем к булевой формуле.

4.1 AcceptExamples

Введем булевы переменные $t_{i,l,k}$, которые означают, что после просмотра первых l символов примера с номером i, ODA окажется в состоянии с номером k. Тогда, чтобы проверить, что ODA принимает все примеры, добавим следующие булевы выражения:

Пусть $k \in Q$, $ex_i \in E$, $0 \leq l < |ex_i|$, $a \in (\Sigma \times \Gamma)$ – l-ый символ примера ex_i. Тогда добавим формулы:

$$t_{i,l,k} \implies \vee_{j \in Q}(\delta(k, a, j) \wedge t_{i,l+1,j}) \tag{7}$$

(Если автомат, после промотра его первых l символов примера находился в состоянии k, то оно после перехода по символу a, автомат должен находиться в состоянии j)

Пусть $k \in Q$, $ex_i \in E$, $l = |ex_i|$. Тогда

$$t_{i,l,k} \implies isFinal(k) \tag{8}$$

(Это условие заставляет принимать примеры).

Пусть $ex_i \in E$. Тогда необходимо добавить формулы: $t_{i,0,1}$ (Не умаляя общности, считаем, что стартовое состояние – 1. Тогда ODA заканчивает в стартовом состоянии, просмотрев в любом из примеров 0 символов).

4.2 Projection

Проверить, что δ_{in} является проекцией δ не составляет особых усилий:

Пусть $i, j \in Q$, $a \in \Sigma$, $b \in \Gamma$, $(a, b) \in (\Sigma \times \Gamma)$. Тогда

$$\delta(i, (a, b), j) \implies \delta_{in}(i, a, j) \tag{9}$$

$$\delta_{in}(i, a, j) \implies \vee_{b \in \Sigma} \delta(i, (a, b), j) \tag{10}$$

4.3 Unambiguous

Перед тем, как проверить автомат на недвойственность, введем переменные r_i, где $i \in Q$. $r_i = 1$, если состояние i достижимо из стартового состояния. Тогда добавим формулы:

$$r_1 \tag{11}$$

$$r_i \iff \vee_{j \in Q, a \in \Sigma, i \neq j} \delta_{in}(j, a, i) \tag{12}$$

Проверим получающийся автомат на недвойственность. Будем действовать способом, описанным в главе «Решение с помощью SMT-решателя». Введем переменные: $\tau_{i,j}$. Если $\tau_{i,j} = 1$, то существует слово w, что пробег слова по δ_{in} мог закончиться и в состоянии i, и в состоянии j (т.к. δ_{in} – функция перехода недетерминированного конечного автомата), пройдя по различным путям. Тогда:

Пусть $i, j, k \in Q, a \in \Sigma, b, c \in \Gamma, b \neq c$. Добавим формулы:

$$r_i \wedge \delta(i, (a, b), j) \wedge \delta(i, (a, c), k) \implies \tau_{j,k} \tag{13}$$

(Если состояние i достижимо и из этого состояния есть переходы в состояния j и k, то добавляем пару (i, j). Это является «инициализацией» в описанном выше алгоритме).

Пусть $i, j, q, w \in Q, a \in \Sigma$. Тогда добавим формулы:

$$\tau_{i,j} \wedge \delta_{in}(i, a, q) \wedge \delta_{in}(j, a, w) \implies \tau_{q,w} \tag{14}$$

(Формируем и добавляем новые пары. Заметим, что в данном случае нет требований $q \neq w$ и $i \neq j$, т.к. возможна ситуация, когда слово прошло разными путями и оказалось в одном и том же состоянии).

Пусть $i, j \in Q$. Добавим последнюю булеву формулу:

$$\neg(\tau_{i,j} \wedge isFinal(i) \wedge isFinal(j)) \tag{15}$$

(У слова не должно быть двух путей, оканчивающихся в терминальном состоянии).

4.4 Total

Самое сложное – проверка, что автомат получился полный. Будем использовать конструкцию, предложеную авторами статьи, с одним лишь тем изменением, что вместо $c_{l,q}$ будем хранить список переменных, которые будут соответствовать битам в двоичной записи $c_{l,q}$. Также придется реализовать сложение двух двоичных чисел. Как упоминалось раньше, достаточно проверить, что подходят все слова из Σ^l, где $0 \leq l \leq |Q|$. Определим $Size$:

$$Size = |Q| * log_2(max(|Q|, |\Sigma|)) + 1 \tag{16}$$

Будем обозначать за $\mathbf{a}$ список переменных $a_1, a_2 ... a_{Size}$

Будем считать, что + принимает $\mathbf{a}$ и $\mathbf{b}$ и возвращает некоторый $\mathbf{d}$, где $\mathbf{d}$ – новый список перемнных(то есть переменные из этого списка еще не встречались в окончательной формуле), а также добавляет следующие формулы:

Пусть **rest** –список переменных, которые еще не встречались в окончательной формуле, $i \in [1..Size - 1]$. Тогда добавим формулы:

$$d_i \iff a_i \oplus b_i \oplus rest_i \tag{17}$$

$$rest_{i+1} \iff med(a_i, b_i, rest_i) \tag{18}$$

$$rest_1 \iff \mathbf{0} \tag{19}$$

Данные операции описывают сложение двух бинарных чисел. Напомним, что $\oplus$ от трех аргументов можно записать следующим образом:

$$q \oplus w \oplus e \equiv (q \wedge w \wedge e) \vee (\neg q \wedge \neg w \wedge e) \vee (\neg q \wedge w \wedge \neg e) \vee (q \wedge \neg w \wedge \neg e) \tag{20}$$

$med(q, w, e)$ можно записать так:

$$med(q, w, e) \equiv (q \wedge w) \vee (q \wedge e) \vee (w \wedge e) \tag{21}$$

Функцию + также можно обобщить на n списков переменных, положив

$$+(\mathbf{a}^1, \mathbf{a}^2, ...\mathbf{a}^n) = (..((\mathbf{a}^1 + \mathbf{a}^2) + \mathbf{a}^3) + ...) + \mathbf{a}^n) \tag{22}$$

Теперь опишем операцию =, которая принимает $\mathbf{a}$ и $\mathbf{b}$. Эта операция исключительно добавляет булевы формулы:

Пусть $i \in [1..Size - 1]$. Тогда добавим формулы:

$$a_i \iff b_i \tag{23}$$

И еще потребуется операция $And(q, \mathbf{a})$, которая принимает булеву переменную и список переменных, возвращает новый список переменных $\mathbf{d}$ и добавляет формулы:

$$d_i \iff (q \wedge a_i) \tag{24}$$

Пересчитывать $\mathbf{c}_{l,q}$ будем следующим образом. Вызовем функцию + со следующими аргументами

$$And(\delta_{in}(j,a,q),\mathbf{c}_{l-1,j})\forall j \in Q, \forall a \in \Sigma \tag{25}$$

А дальше возьмем результат этой функции **res** и применим операцию

$$\mathbf{res} = \mathbf{c}_{l,q} \tag{26}$$

Когда сделали операции из предыдущего пункта для всех l и q, необходимо проделать следующие операции для любого l:

Применим операцию + ко всем переменным вида

$$And(isFinal(q),\mathbf{c}_{l,q})\forall q \in Q \tag{27}$$

Возьмем результат **res** предыдущей операции и применим операцию

$$\mathbf{res} = FromIntToBooleanList(|\Sigma|^l) \tag{28}$$

Функция $FromIntToBooleanList(x)$ возвращает новый список $\mathbf{d}$ и добавляет булевы формулы:

Если в двоичной записи числа x на позиции i стоит 1:

$$d_i \iff \mathbf{1} \tag{29}$$

Иначе:

$$d_i \iff \mathbf{0} \tag{30}$$

Последнее, о чем необходимо позаботиться – об «инициаллизации»: применим следующие операции.

$$\mathbf{c}_{0,1} = FromIntToBooleanList(1) \tag{31}$$

$$\mathbf{c}_{0,i} = FromIntToBooleanList(0) \tag{32}$$

где $i \in Q$ и $i \neq 1$

Таким образом мы проверим, что автомат является полным.