Анализ атрибутивной политики безопасности с использованием методов автоматического планирования

Афонин С.А.[1], Бонюшкина А.Ю.[2]

В работе рассматривается задача проверки возможности получения пользователем информационной системы доступа к выделенному объекту при заданной атрибутивной политике безопасности. Показывается, что при некоторых ограничениях на модель информационной системы и правила политики эта задача сводится к задаче интеллектуального планирования. Определяется древовидная структура, построение которой соответствует планированию в пространстве планов и позволяет учитывать специфику решаемой задачи при построении эвристических алгоритмов проверки доступа. Доказывается, что существование такой структуры является необходимым и достаточным условием возможности получения доступа к целевому объекту.[1 2]

Работа выполнена при поддержке гранта РФИИ 18-07-01055.

Ключевые слова: ABAC, проверка доступа, интеллектуальное планирование.

1. Введение

Управление правами доступа является важной составляющей информационных систем. В многопользовательских системах необходимо ограничивать возможность выполнения *субъектами* доступа (пользователями или программами) тех или иных операций с *объектами* доступа. В общем случае доступ зависит от свойств объекта и субъекта доступа, вида операции и *контекста*. Примером контекста является время выполнения

[1] *Афонин Сергей Александрович* — доцент каф. дискретной математики мех.-мат. ф-та МГУ, e-mail: serg@msu.ru.

Afonin Sergey Aleksandrovich — associate professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Numerical Analysis.

[2] *Бонюшкина Антонина Юрьевна* — аспирант каф. дискретной математики мех.-мат. ф-та МГУ, e-mail: abonush@yandex.ru.

Bonyushkina Antonina Yuryevna — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Numerical Analysis.

операции, текущее физическое положение пользователя или другие подобные данные. Доступ субъектов к объектам определяется правилами о допустимых операциях субъекта над объектом. Например, в правиле «Сотрудник отдела кадров имеет право изменять карточку работника с 10 до 17 часов» объектом доступа является карточка работника (запись в базе данных), субъектом — сотрудник отдела кадров, операция — редактирование записи. При этом накладывается дополнительное ограничение на контекст — время выполнения операции. Набор правил доступа такого вида называется *политикой безопасности*.

На практике политика безопасности описывается документом на естественном языке. Для реализации политики безопасности в коде системы правила доступа представляются в терминах некоторой *модели доступа*. Использование формальной модели позволяет проводить анализ политики на предмет выполнения некоторых свойств. Например, рассматриваются такие свойства как непротиворечивость правил политики друг другу, доступность объектов доступа определенным пользователям, эквивалентность политик и другие [4, 3].

Алгоритмическая разрешимость и сложность задач анализа политики доступа зависит от выразительной силы модели доступа. В литературе описывается множество моделей доступа. К числу наиболее известных и распространенных моделей относятся дискреционная (DAC), мандатная (MAC) и ролевая (RBAC) модель доступа. В последнее время значительную популярность приобретают модели типа ABAC [8], которые позволяют использовать свойства (атрибуты) объектов и субъектов доступа при определении правил доступа, что позволяется расширить спектр возможных правил по сравнению с ролевой моделью. При этом атрибутивные модели включают некоторые классические модели доступа [1].

В данной работе мы рассматриваем задачу проверки возможности получения пользователем доступа к выделенному объекту в результате выполнения последовательности операций, разрешенных ему политикой CBAC. Мы считаем, что в информационной системе, состоящей из набора взаимосвязанных объектов, работает один пользователь (злоумышленник), который пытается получить доступ к выделенному объекту, выполняя только разрешенные ему операции.

Сформулируем задачу более формально. Пусть задано множество *переменных* (или *объектов* информационной системы) $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. Переменные принимают значения из множества X. *Состоянием системы* является вектор $x \in X^N$ значений переменных. С каждой переменной $\mathbf{x}_i$ связано *множество доступности* $A_i \subseteq X^N$, которое определяет условия на значения переменных, при которых значение переменной $\mathbf{x}_i$ может быть изменено. Если текущее состояние x принадлежит A_i, то пользо-

ватель может перевести систему в состояние x', которое отличается от x только в i-ой компоненте. Такое действие назовем *допустимым* и будем обозначать как $x \mapsto_i x'$. За одну операцию можно изменять только одну переменную. Задача состоит в следующем. Для заданной системы множеств $\mathcal{P} = \langle A_1, A_2, \ldots, A_N \rangle$, индекса $t \in \{1, \ldots, N\}$ (определяющего переменную, к которой мы хотим получить доступ) и начального состояния $x^0 \in X^N$ требуется проверить, что существует конечная последовательность допустимых действий, которая переводит состояние x^0 в некоторое состояние $x' \in A_t$. Другими словами можно сказать, что в переходной системе, заданной системой множеств $\mathcal{P}$, требуется проверить достижимость множества A_t из начального состояния x^0.

Очевидно, что если множества A_i произвольны, то задача проверки доступа не имеет алгоритмического решения. Например, пусть заданы переменные $\{u, v, w\}$ и два рекурсивно перечислимых множества $U = \{u_1, u_2, \ldots\}$ и $V = \{v_1, v_2, \ldots\}$. Определим множества доступности переменных следующим образом:

$$\begin{array}{lcl} A_u & = & \{(u_i, v_i, 0) \mid i > 0\}, \\ A_v & = & \{(u_{i+1}, v_i, 0) \mid i > 0\}, \\ A_w & = & \{u^*, v^*, 0\}. \end{array}$$

При начальном состоянии $x^0 = (u_1, v_1, 0)$ единственно возможная бесконечная последовательность состояний системы имеет вид $(u_1, v_1, 0) \mapsto_u (u_2, v_1, 0) \mapsto_v (u_2, v_2, 0) \mapsto_u (u_3, v_2, 0) \mapsto_v \ldots$. Доступ к переменной w возможен тогда и только тогда, когда либо $u^* \in U$, либо $v^* \in V$. В силу нерекурсивности множеств U и V выполнимость этих условий не может быть проверена.

Одно из возможных ограничений на структуру множеств доступности рассматривается в работе [2], где эти множества представляются в виде конечного объединения декартовых произведений некоторых множеств. В этом случае множество состояний системы разделяется на конечное число классов эквивалентности и задача проверки доступа сводится к поиску пути в конечном графе. Этот результат показывает разрешимость задачи, но число вершин этого графа ограничено 2^{2^N}, что не позволяет использовать данный метод на практике.

Целью данной работы является решение задачи проверки доступа в условиях [2] методами интеллектуального планирования [7]. Для систем с конечным числом состояний универсальные алгоритмы классического планирования, такие как STRIPS, также сводятся к перебору возможных последовательностей действий и имеют экспоненциальную сложность [6]. Возможность разработки прикладных решений связана либо с рассмотрением специальных подклассов исходной задачи, либо с учетом различных эвристик и планированием в пространстве частичных

целей [5]. В данной работе предлагается алгоритм построения плана, который использует свойства задачи проверки доступа.

В следующем разделе приводится основные определения и формальная постановка задачи проверки корректности политики. В разделе 3 описывается интерпретация задачи в виде графа зависимостей и доказываются некоторые его свойства. Метод проверки корректности политики путем выделения промежуточных целей и его связь с задачей интеллектуального планирования описывается в разделе 4. В разделе 5 вводится понятие дерева порядков и доказывается основная теорема — критерий возможности получения доступа.

2. Определения и постановка задачи

Пусть задана система из N *переменных* (будем также называть их *объектами*)

$$\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}.$$

Переменные могут принимать значения из множества X. *Состоянием системы* является вектор $x = (x_1, \ldots, x_N) \in X^N$ значений переменных.

С каждой переменной $\mathbf{x}_i$ связано непустое *множество доступности* $A_i \subseteq X^N$, которое определяет условия на все переменные, при которых значение переменной $\mathbf{x}_i$ может быть изменено: если текущее состояние x принадлежит A_i, то систему можно перевести в состояние x', изменив значение переменной $\mathbf{x}_i$:

$$x = (x_1, \ldots, x_i, \ldots, x_N), x' = (x_1, \ldots, x'_i, \ldots, x_N),$$

причем на её новое значение x'_i не накладывается никаких ограничений. В один момент времени можно изменять одну переменную. Новое состояние x' назовем *непосредственно достижимым* из x.

Условие «$x \in A_i$» называется *условием доступа* к переменной $\mathbf{x}_i$, и если оно выполняется, то изменение значения переменной $\mathbf{x}_i$ *допустимо* (к переменной есть доступ), и такое изменение называется *доступной операцией*. Условие доступа будем записывать через индикаторную функцию множества A_i:

$$I_{A_i}(\mathbf{x}) = \begin{cases} 1, & \text{если } \mathbf{x} \in A_i \\ 0, & \text{иначе.} \end{cases}$$

Таким образом, система непустых множеств $\mathcal{P} = \langle A_i | i = 1, \ldots, N \rangle$ определяет политику доступа к переменным для операции изменения.

Доступ к $\mathbf{x}_i$ *существенно зависит* от переменной $\mathbf{x}_j$, если существует пара состояний $x = (x_1, \ldots, x_j, \ldots, x_N)$ и $x' = (x_1, \ldots, x'_j, \ldots, x_N)$, отличающихся только в j-й координате, такие что $x \in A_i, x' \notin A_i$. Если таких

состояний не существует, то зависимость *фиктивная*. Из непустоты A_i следует, что множество A_i содержит все значения фиктивной $\mathbf{x}_j$. Существенные для доступа к $\mathbf{x}_i$ переменные (объекты) назовём *подобъектами* $\mathbf{x}_i$.

Доступную операцию изменения состояния обозначим $x \mapsto_i x'$: состояние x переходит в состояние x' изменением в i-той компоненте. Если из контекста ясно, какая переменная меняет значение, или это не важно, переход в другое состояние будем обозначать как $x \mapsto x'$.

Состояние x' назовем *достижимым* из состояния x^0, если существует конечная последовательность состояний $x^1, \ldots, x^k = x'$, таких что $x^i \mapsto x^{i+1}\ \forall i \in \{0, 1, 2, \ldots, k-1\}$. Здесь каждое состояние отличается от предыдущего в одной переменной. Подчеркнем, что все изменения в цепочке должны быть доступны (то есть каждое состояние принадлежит множеству доступности переменной, которая меняется для перехода к следующему состоянию). Переход в достижимое состояние обозначим $x \mapsto^* x'$.

Множество S назовем *достижимым* из состояния x, если существует состояние $x' \in S$, достижимое из x: $x' \in S : x \mapsto^* x'$, обозначение $x \mapsto^* S$.

Замечание. Если X^N — это аффинное пространство, каждая переменная — координата, а состояние — это точка, то непосредственно достижимое из x состояние x', отличающееся в i-той координате получено «сдвигом по прямой вдоль i-той координаты». Множество A_i является цилиндрической поверхностью вдоль фиктивной для $\mathbf{x}_i$ переменной.

Задача проверки доступа Будем считать, что у каждой переменной $\mathbf{x}_j$ есть k_j существенных переменных $\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_{k_j}}$, а множество доступности A_j задается индикаторной функцией вида

$$I_{A_j}(\mathbf{x}) = \prod_{i=1}^{k_j} I_{A_{j,j_i}}(\mathbf{x}_{j_i}), \tag{1}$$

где $A_{j,j_i} \subseteq X$ есть множество значений переменной $\mathbf{x}_{j_i}$, при которых возможен доступ к $\mathbf{x}_j$. Это означает, что условия доступа $\mathbf{x}_j$ проверяются «независимо» для каждой из её существенных переменных и доступ к $\mathbf{x}_j$ есть, если выполняется каждое из условий справа в (1). Такие условия и функции назовем *разделимыми*.

Множества $A_{i,j}$ могут быть произвольными. Для реализации алгоритмов потребуем, чтобы они были «эффективно определены», то есть известны алгоритмы:

1) проверки пустоты пересечения $A_{i,j}$ со всеми $A_{k,j}$;

2) выбора представителя этого пересечения, если оно не пусто.

То есть мы должны знать, какие условия на $\mathbf{x}_j$ могут быть выполнены одновременно и уметь присваивать $\mathbf{x}_j$ такие значения. Для простоты будем считать, что эта информация известна и нам не надо тратить время на проверку и поиск представителя.

Задача состоит в следующем. Для заданной системы «эффективно определенных» множеств $\mathcal{P}$, удовлетворяющих условию (1), индекса $t \in \{1, \ldots, N\}$ и начального состояния $x^0 \in X^N$ требуется проверить, что $x \mapsto^* A_t$. Переменную $\mathbf{x}_t$ и множество A_t будем называть соответственно *целевой* переменной и *целевым* множеством.

Для удобства обозначим $I_{i,j}(\mathbf{x}) = I_{A_{i,j}}(\mathbf{x}_j)$, подразумевая, что условие принадлежности множеству $A_{i,j}$ накладывается только на $\mathbf{x}_j$. Без ограничения общности можно считать, что целевой переменной является переменная $\mathbf{x}_0$.

Таким образом, если пользователю нужно получить доступ к $\mathbf{x}_0$, необходимо, чтобы выполнялись индикаторные условия на его подобъекты. Если какие-то из подобъектов $\mathbf{x}_0$ не удовлетворяют условиям, пользователь должен изменить их значения на требующиеся. Для этого у него должен быть доступ к этим объектам, который также определяется индикаторными условиями на их подобъекты, и так далее.

3. Графическое представление

Совокупность переменных с условиями доступа к ним можно представить ориентированным графом, который мы назовем *графом зависимостей.*

Определение 1. Полным графом зависимостей для состояния x *называется ориентированный граф* $G(x) = \langle \mathbf{x}, E, \mathbb{I}(\mathcal{P}), \mu \rangle$ *с помеченными ребрами, который удовлетворяет следующим условиям:*

- *вершинами* G *являются переменные* $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$*;*
- *пара вершин* $(\mathbf{x}_i, \mathbf{x}_j)$ *образует ориентированное ребро* $e_{i,j} \in E$*, если* $\mathbf{x}_j$ *является существенной переменной для* $\mathbf{x}_i$*;*
- *задан набор множеств доступности* $\mathcal{P}$*, пересечения которых мы умеем проверять на непустоту и брать представителя. Задан соответствующий ему набор разделимых функций доступа* $\mathbb{I} = \{I_{i,j}(\mathbf{x})\}$*;*
- разметкой *полного графа зависимостей по текущему состоянию* $x \in X^N$ *назовем функцию* $\mu : E \times X^N \to \{0, 1\}$*, такую, что* $\mu(e_{i,j}, x) = I_{i,j}(x)$*. Для определенности будем считать, что если ребра* $e_{i,j}$ *не существует, то* $\mu((\mathbf{x}_i, \mathbf{x}_j), x) \equiv 1$ $\forall x$ *(то есть*

условие доступа всегда выполнено, что и следует из фиктивной зависимости между этими вершинами).

В случае, когда понятно, какое состояние текущее, будем обозначать метку ребра в нем $\mu(e_{i,j}, x) = \mu(e_{i,j})$.

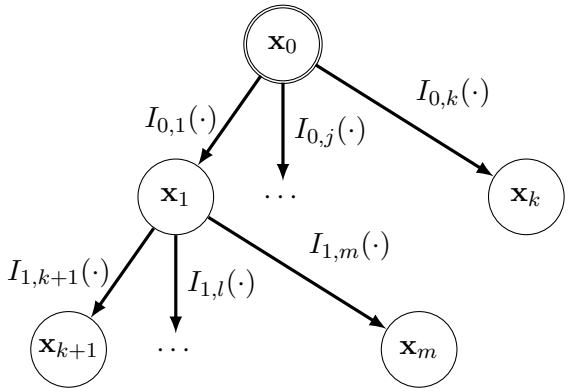


Рис. 1. Построение графа зависимостей методом поиска в ширину.

Для удобства обозначим через $out(\mathbf{x}_i)$ множество всех существенных переменных объекта $\mathbf{x}_i$, а $in(\mathbf{x}_i)$ — множество переменных, для которых $\mathbf{x}_i$ является существенным. Таким образом доступ к $\mathbf{x}_i$ зависит только от объектов из множества $out(\mathbf{x}_i)$, а сам $\mathbf{x}_i$ влияет только на доступ к переменным множества $in(\mathbf{x}_i)$.

В фиксированном состоянии x индикаторные функции на ребрах принимают значения 0 или 1, что соответствует булевой разметке графа. Изменение значений объектов не меняют структуру графа, изменяя только разметку.

Для проверки доступа важно не то, какие именно значения принимают переменные, — а только, выполняются ли зависящие от них индикаторные функции. Поэтому для удобства будем рассматривать не значения объектов, а классы эквивалентности их значений через разметку: $\forall \mathbf{x}_i$ занумеруем элементы множества $in(\mathbf{x}_i) = \{1, \ldots, k\}$ и значение $\mathbf{x}_i$ запишем k-разрядным числом в двоичной системе счисления, где в j-том разряде стоит значение $I_{j,i}(\mathbf{x})$. Таким образом в двоичной записи представлено, какие условия на объект выполняются, а какие нет. Это число обозначим $val.in(\mathbf{x}_i)$. Обозначим $val.in(\mathbf{x}_i)|_x$ это значение в состоянии x. Если по контексту понятно, что рассматривается текущее состояние, то «$|_x$» будем опускать.

Аналогично поступим с исходящими ребрами объекта: перенумеровав их, запишем натуральное число в двоичной записи, каждому разряду которого соответствует значение, приписанное некоторому исходящему

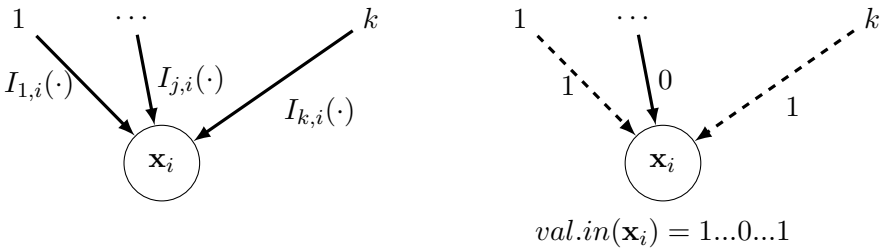


Рис. 2. Кодирование разметрки рёбер значением вершины.

ребру. Это число обозначим $val.out(\cdot)$. Обозначим $val.out(\cdot)|_x$ это значение в состоянии x. Если по контексту понятно, что рассматривается текущее состояние, то «$|_x$» будем опускать.

Доступ к объекту есть, только если его $val.out()$ состоит из одних единиц (обозначим такое значение $\overline{1}$) , и тогда мы можем менять его $val.in()$. При этом вполне возможно, что далеко не все значения $val.in()$ можно получить: то есть, не всегда можно подобрать такое значение, которое удовлетворяет всем условиям $I_{j,i}()$ (что соответствует $val.in() = \overline{1}$), поэтому задача и сложна). Набор значений, которые мы можем получить, напрямую связан с непустотой пересечения множеств $A_{i,j}$. Для решения нам важно знать, какие значения могут быть у всех $val.in()$. Для этого нужно знать, не пусто ли пересечение любого конечного числа $A_{i,j}, \mathbf{x}_i \in in(\mathbf{x}_j)$, и уметь брать представителя этого пересечения, чтобы присваивать его значение переменной $\mathbf{x}_j$.

Для удобства будем называть ребра, помеченные единицей, *зелеными*, а нулем — *черными*. Ориентированный путь, который является последовательностью черных ребер, назовём *черным путем*. Ориентированный цикл из черных ребер назовем *черным циклом*.

Заметим, что для наших целей — проверить, можно ли получить доступ к $\mathbf{x}_0$ — весь полный граф зависимостей с разметкой не нужен. Мы будем рассматривать такой подграф полного графа зависимостей, в который входит целевая вершина $\mathbf{x}_0$; все вершины, в которые ведет черный путь из $\mathbf{x}_0$; все ребра, которые соединяют эти вершины (и черные, и зеленые). Считаем, что число вершин в этом графе равно n. Получившийся граф D будем называть просто *графом зависимостей* для доступа к $\mathbf{x}_0$.

3.1. Свойства графа зависимостей

Первое, что мы понимаем о графе зависимостей, — это то, что он не содержит лишних рёбер. Если доступ к целевой вершине можно получить, то каждое ребро графа зависимостей на некотором шаге процесса получения доступа должно принять метку 1. Аналогично и каждая вершина

на некотором шаге должна стать доступной. В противном случае получится, что к вершинам, которые требовалось изменить, не было получено доступа, а значит изменить их не удалось, и доступ к целевой вершине не мог быть получен. Это наблюдение формально записано ниже.

Утверждение 1. *Пусть состояние $x^m \in A_0$ достижимо по цепочке состояний x^0, x^1, ..., x^m. Тогда:*
(1) для каждого ребра e_{ij} графа D существует $k \in \{0, \dots, m\}$, такое что $\mu(e_{ij}, x^k) = 1$;
(2) для любой вершины $\mathbf{x}_j$ графа D существует k, такое что $val.out(\mathbf{x}_j)|_{x_k} = \overline{1}$.

Доказательство. Следует из построения графа зависимостей. □

Утверждение 2. *Если в графе зависимостей в состоянии x есть цикл из черных ребер, то доступ к целевой вершине получить невозможно.*

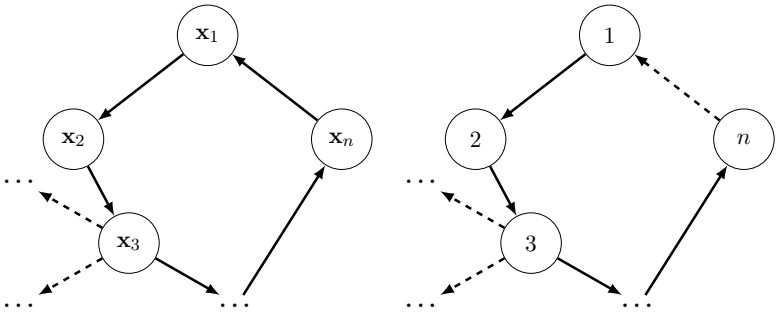
Доказательство. Черный цикл — это последовательность вершин $\mathbf{x}_1$, ..., $\mathbf{x}_k$, что $\mathbf{x}_{k+1} = \mathbf{x}_1$ и $\mu((\mathbf{x}_i, \mathbf{x}_{i+1}), x) = 0$, $i = 1, \dots, k$ (рис.3 (a)). Для получения доступа к любой вершине цикла необходимо изменить значения ее подобъектов, для получения доступа к подобъекту нужно изменить его подобъекты, и так далее. Приходим к тому, что для получения доступа к вершине ее саму необходимо изменить, что невозможно, так как доступа нет. Далее заметим, что из утверждения 1 следует, что если целевое множество достижимо, то каждая вершина должна по крайней мере один раз изменить значение (поскольку в графе в каждую вершину ведёт хотя бы одно черное ребро, которое должно на некотором шаге выполниться, а значит эту вершину придётся менять). □

Получается, в случае с черным циклом мы знаем, что у задачи нет решения (доступ получить нельзя). Ниже доказано, что если в графе зависимостей для начального состояния нет черных циклов, то они не могут появиться при выполнении допустимых операций.

Утверждение 3. *Пусть состояние x^m достижимо из состояния x^0. Если в графе зависимостей $D(x^0)$ нет черных циклов, то и граф $D(x^m)$ не содержит черных циклов.*

Доказательство. Так как в один момент времени можно изменить значение только в одной вершине, достаточно показать, что нельзя выполнить один последний шаг, чтобы замкнуть цикл (рис.3 (b)).

Пусть есть цепочка пронумерованных вершин $1, \dots, n$, соединенных ребрами $(i, i + 1 \bmod n)$, $i = 1, ..n$, причем все указанные ребра, кроме $(n, 1)$ черные, а $(n, 1)$ — зеленое. Тогда его нельзя сделать черным, так как для этого необходимо изменить значение в вершине 1 на такое чтобы $\mu(n, 1) = 0$. А для этого нужен доступ к вершине 1, а для того чтобы его



(a) Цикл из чёрных ребер. (b) Дополнение до цикла.

Рис. 3. Циклы черных ребер в графе зависимостей.

получить, нужно чтобы $val.out(1) = \bar{1}$, в том числе $\mu(1,2) = 1$. Для этого нужен доступ к вершине 2, и т.д., нужен доступ к вершине n, допустим, что он есть, тогда изменяем ее так что $\mu(n-1,n) = 1\ldots$, $\mu(1,2) = 1$, теперь можно положить $\mu(n,1) = 0$ (теперь вершина n недоступна), но даже если последовательно сделать черными ребра $\mu(1,2) = 0,\ldots,$ $\mu(n-2,n-1) = 0$, тогда $\mu(n-1,n)$ изменить нельзя, так как n недоступна, оно останется зеленым, и ситуация та же, что была изначально: остался цикл без одного ребра. □

3.2. Порядок вершин графа зависимостей

Теперь мы знаем, что когда в графе зависимостей есть черный цикл, доступ получить нельзя, и что если в начальном состоянии черного цикла нет, то появиться он не может. Поэтому предполагаем, что мы умеем проверять наличие черного цикла в начальном состоянии графа и далее мы будем рассматривать только графы без черных циклов. Это позволяет упорядочить вершины естественным образом. Расположим вершины D по уровням, введя частичный порядок $\preceq$, порожденный ориентированным черным путем при начальной разметке: если есть черный путь из $\mathbf{x}_i$ в $\mathbf{x}_j$, то $\mathbf{x}_j \preceq \mathbf{x}_i$. При этом любая вершина $\preceq \mathbf{x}_0$, которая находится на уровне с максимальным номером, а на самом нижнем уровне находятся вершины, доступные в начальном состоянии.

Далее мы без ограничения общности будем считать, что вершины графа зависимостей занумерованы в соответствии с отношением $\preceq$, то есть из $\mathbf{x}_i \preceq \mathbf{x}_j$ следует строгое неравенство $i < j$. Несравнимые вершины, находящиеся на одном уровне, пронумеруем между собой случайным образом, но в соответствии с частичным порядком относительно вершин других уровней.

Ниже показана корректность такой нумерации в смысле отсутствия черных рёбер, ведущих на уровень с бо́льшим номером.

Утверждение 4. *Пусть вершины графа зависимостей D занумерованы в соответствии с отношением $\preceq$, и пара вершин $(\mathbf{x}_i, \mathbf{x}_j)$ образует ребро. Тогда $\mu(e_{ij}, x^0) = 0 \implies i < j$.*

Доказательство. Противоречие с тем, что путь до вершины, в которую ведет обратное ребро, самый длинный. □

Следствие 1. *При таком порядке вершин $\preceq$ для получения доступа к каждой вершине из начального состояния может потребоваться менять только вершины с меньшим номером (непосредственно следует из отсутствия обратных черных ребер).*

Приведенные выше свойства графа зависимостей обосновывают корректность его построения из полного графа G. По построению D мы не взяли в него те вершины, в которые не ведут черные ребра из вершин, уже входящих в D. Это вершины, в которые входят только зеленые ребра из D или не входят никакие (могут быть входящие ребра из G, но не D). Если в вершину $\mathbf{x}_j$ не входят ребра из D, то другие вершины из D от нее не зависят, и изменение $\mathbf{x}_j$ не повлияет на доступ к ним, а значит нет причин получать к ней доступ и знать, от чего она зависит. Если в вершину $\mathbf{x}_j$ входят только зеленые ребра из D, значит все условия, зависящие от нее, уже выполнены, и ее также незачем менять. Таким образом, все вершины, которые не были включены в граф зависимостей целевой вершины, отделены от него ориентированными зелеными ребрами, которые никогда не понадобится менять.

Теперь мы имеем граф зависимостей, в котором последовательно будем менять значения вершин, попутно изменяя цвет ребер и доступность вершин. А значит порядок, связанный с длиной черного пути до вершины, будет меняться. Тем не менее, покажем, что при некоторых условиях можно использовать начальный порядок (и свойство, что для доступа к вершине нужны только вершины с меньшим номером в этом порядке) до момента получения доступа.

Прежде всего заметим, что любая цепочка допустимых действий обратима:

Утверждение 5. *Если состояние x' было достигнуто из начального состояния x^0 цепочкой допустимых действий и соответствующей ей цепочкой состояний, то систему можно вернуть в состояние x^0 цепочкой обратных действий.*

Доказательство. Поскольку доступ к вершине не зависит от нее самой, а каждое действие — изменение значения одной вершины, не меняющее

ее доступность, то каждое действие в отдельности может быть обращено, то есть состояние переведено в предыдущее, в котором может быть обращен уже предыдущий шаг, и так далее до x^0. □

Свойство обратимости верно для любой цепочки допустимых действий. Справедливо и более сильное утверждение: в определенных случаях систему можно перевести в состояние, которое отличается от начального значением одной вершины.

Утверждение 6. *Пусть состояние* $x' = (x'_1, \ldots, x'_n, x_{n+1}, \ldots, x_N)$ *графа зависимостей* D *было достигнуто из начального состояния* $x^0 = (x^0_1, \ldots, x^0_n, x_{n+1}, \ldots, x_N)$ *цепочкой допустимых действий и соответствующей ей цепочкой состояний, в которой менялись без ограничения общности только первые* n *вершин* $\mathbf{x}_1, \ldots, \mathbf{x}_n$. *Пусть состояние* x' *таково, что в нем существует вершина* $\mathbf{x}_i, i \in \{1, \ldots, n\}$, *принимающая значение* x'_i, *такое что* $\mu((\mathbf{x}_j, \mathbf{x}_i), x'_i) = 1$, $j \in \{1, \ldots, n\}, j \neq i$. *Тогда систему можно перевести в состояние* $x^{0'} = (x^0_1, \ldots, x'_i, \ldots, x^0_n, x_{n+1}, \ldots x_N)$, *отличающееся от начального в одной вершине, цепочкой обратных действий.*

Условие $\mu((\mathbf{x}_j, \mathbf{x}_i), x'_i) = 1$, $j \in \{1, \ldots, n\}, j \neq i$ значит, что в этом состоянии выполняются условия доступа для вершин, зависящих от $\mathbf{x}_i$ и изменяющихся в данной цепочке действий.

Доказательство. По условию мы меняли только вершины подграфа D_n из первых n вершин графа зависимостей D. Новое значение x'_i удовлетворяет условиям доступа ко всем вершинам D_n. Значит присваивание вершине этого значения не повлияло на доступность других вершин D_n. Значит, за исключением нового значения $\mathbf{x}_i$, мы можем обратить все изменения для этих вершин к начальному состоянию, выполнив в обратном порядке все совершенные до сих пор изменения вершин. На каждом шаге единственное, что отличается от состояния, через которое проходил прямой процесс — это значение $\mathbf{x}_i$, но его изменение не повлияло на доступ к вершинам, которые изменяются, поэтому обратный шаг можно сделать. □

Следствие 2. *Очевидно, что порядок* $\preceq$ *на всех вершинах за исключением* $\mathbf{x}_i$, *в состоянии* $x^{0'}$ *«не испортится» в сравнении с* x^0, *в том смысле что для каждой вершины множество вершин* $\{\mathbf{x}_j | \mathbf{x}_j \preceq \mathbf{x}_i\}$ *в новом состоянии, не увеличилось.*

Доказательство. Поскольку значения вершин вернулись к прежним, а значит метки всех ребер, кроме тех, что входят в $\mathbf{x}_i$, остались прежними.

Могли добавиться только черные ребра, входящие в $\mathbf{x}_i$, но эта вершина не рассматривается. □

Следствие 3. *Аналогично можно из состояния $x^{0'}$ снова вернуться в состояние x^0, изменив в x' вершину $\mathbf{x}_i$ на прежнее значение.*

4. Проверка доступа как задача планирования

Коротко опишем задачу классического планирования [7]. Пусть задана переходная система $\Sigma = \{S, A, \gamma\}$, где S — конечное множество состояний, A — конечное множество действий, $\gamma : S \times A \to S$ — функция переходов состояний.

Пусть также фиксировано начальное состояние $s_0 \in S$ и множество целевых состояний $S_g \subseteq S$. Задача классического планирования заключается в том, чтобы найти последовательность действий (план) $(a_1, a_2, \ldots, a_k)$ и соответствующую ему последовательность состояний $(s_0, s_1, \ldots, s_k)$, такую что $s_1 \in \gamma(s_0, a_1)$, $s_2 \in \gamma(s_1, a_2)$, $\ldots s_k \in \gamma(s_{k-1}, a_k)$, и $s_k \in S_g$. Действиям могут дополнительно ставиться в соответствие стоимость их реализации. В этом случае требуется найти план минимального (или максимального) веса.

Несложно заметить, что задача проверки доступа в точности является задачей классического планирования: состояния системы — это вектор значений вершин графа зависимостей для $\mathbf{x}_0$ в смысле класса эквивалентностей $\{val.in(\mathbf{x}_0), \ldots, val.in(\mathbf{x}_n)\}$. Количество классов эквивалентности конечно, поскольку это вектор из $n + 1$ вершины, у каждой из которых не более 2^n возможных значений. Действием будет изменение значения доступной вершины, переход в другой класс, то есть изменение $val.in()$ этой вершины на другое возможное значение. Функция переходов состояний по состоянию и действию определяет новое состояние — с новым значением измененной вершины.

Для поиска оптимального плана используются методы *планирования в пространстве состояний* и *в пространстве планов*. В первом случае на каждом шаге алгоритм выбирает конкретное действие. Во втором — сначала строится множество промежуточных целей, последовательное достижение которых приводит к достижению состояния из целевого множества S_g, а потом производится поиск действий, переводящих систему от одной промежуточной цели к другой.

Планирование в пространстве планов допускает наличие достаточно эффективных эвристических алгоритмов поиска решения. Для использование этого метода требуется определь промежуточные цели решения

задачи. Для этого разберемся, какие условия на значения вершин должны выполняться в процессе получения доступа.

Предположим, что существует последовательность S переменных $\mathbf{x}_i$, такая что изменение значений вершин в этой последовательности приводит к получению доступа к целевой вершине. Она конечна, поскольку завершается с получением доступа. Поэтому в этой последовательности каждая вершина на каком-то шаге встретится в последний раз. Это значит, что на этом шаге она в последний раз поменяет значение и оно «фиксируется». Порядок вершин, в котором они принимают окончательное значение, назовем *порядком фиксации* (это подпоследовательность S). Номер вершины в этом порядке назовем *номером фиксации*.

Для каждой вершины $\mathbf{x}_i$ в последовательности S можно выделить фрагмент с начала последовательности до момента, когда эта вершина встречается впервые. Такой фрагмент S соответствует «получению доступа к нецелевой вершине $\mathbf{x}_i$» и определяет некоторый «локальный» порядок фиксации, который может отличаться от порядка фиксации для всей последовательности S: вершины, которые фиксируются в этом порядке, могут быть использованы позже.

Для получения доступа к некоторой вершине из начального состояния с его порядком на вершинах потребуется менять только вершины, которые меньше данной в введенном частичном порядке $\preceq$. Поэтому порядок фиксации для доступа к нецелевой вершине $\mathbf{x}_i$ начального размеченного графа зависимостей D — это порядок фиксации на подграфе, для которого $\mathbf{x}_i$ является корневой.

Следующие утверждения определяют условия, которым должны удовлетворять последние значения вершин.

Утверждение 7. *Если $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_n}$ — порядок фиксации вершин графа зависимости для доступа к целевой вершине $\mathbf{x}_0$, то после фиксации в состоянии x^j вершины $\mathbf{x}_{i_j}, j < n$ нефиксированные вершины $\mathbf{x}_{i_k}, k > j$ достижимы по черному пути от целевой.*

То есть нефиксированные вершины не могли оказаться ненужными для дальнейшего процесса получения доступа.

Доказательство. Пусть в нефиксированную вершину k в состоянии x^j нет черных путей из $\mathbf{x}_0$. Значит, она уже не понадобится для получения доступа к целевой вершине. Значит, нам не потребуется менять ее значение. Значит, ее значение стало фиксированным при последнем изменении, противоречие с тем, что она нефиксированная. □

Лемма 1 (Критерий про черные пути)**.** *Пусть $\mathbf{x}_{i_1}$, $\mathbf{x}_{i_2}, \ldots$, $\mathbf{x}_{i_n}$ — порядок фиксации вершин, приводящий к доступу к целевой вершине, а*

$x^1, \ldots, x^n$ — *состояния в момент фиксации значений этих вершин. Тогда для любого* $k \leq n$ *и* $j > k$ *в графе* $D(x^k)$ *нет черных путей из вершины* $\mathbf{x}_{i_j}$ *в вершину* $\mathbf{x}_{i_k}$.

Доказательство. Утверждение означает, что последнее принимаемое вершиной значение должно выбираться таким образом, чтобы не возникало чёрных путей из вершин с бо́льшим номером фиксации.

Пусть у вершины f фиксировано значение и есть черный путь в неё из нефиксированной вершины g. Так как значение g ещё не фиксировано, её ещё надо будет менять, а для этого к ней нужно будет получить доступ. Значит её $val.out(g)$ должен будет состоять только из единиц, в том числе первое ребро черного пути из неё в f должно будет стать зелёным. А для этого нужно изменить соответствующий подобъект, а для этого к нему надо получить доступ. И так далее, все ребра черного пути из g в f придется выполнить, в том числе входящее в f, а это значит, что f надо будет изменить, что противоречит тому, что её значение фиксировано. □

Лемма 2 (О выборе фиксированного значения)**.** *В условиях предыдущей леммы отсутствие черных путей из нефиксированных вершин в вершину* $\mathbf{x}_{i_k}$ *в состоянии* x^k *— это в точности условие, что для любого* $j > k$ $\mu((\mathbf{x}_{i_j}, \mathbf{x}_{i_k}), x^j) = 1$.

То есть все ребра, непосредственно ведущие из вершин, находящимся справа от данной в порядке фиксации, должны быть зелеными, и вершине $\mathbf{x}_{i_k}$ должно быть присвоено значение, удовлетворяющее условиям, приписанным этим ребрам.

Доказательство. Проверим по критерию. Разделим множество всех вершин, из которых есть ребра в $\mathbf{x}_{i_k}$ непосредственно, на фиксированные (множество $fix(\mathbf{x}_{i_k})$) и нефиксированные ($notfix(\mathbf{x}_{i_k})$).

Существующие ребра непосредственно из нефиксированных вершин в вершину $\mathbf{x}_{i_k}$ зеленые, так как не должно быть черных путей в $\mathbf{x}_{i_k}$ из нефиксированных вершин. Есть ли черные пути из нефиксированных вершин в $\mathbf{x}_{i_k}$? Если такой путь проходит через $notfix(\mathbf{x}_{i_k})$, то ближайшее к $\mathbf{x}_{i_k}$ ребро (между нефиксированной вершиной и $\mathbf{x}_{i_k}$) зеленое, противоречие.

Если такой путь проходит через $fix(\mathbf{x}_{i_k})$, то на этом пути (он конечен) будет ребро из нефиксированной вершины в фиксированную, оно зеленое по предположению для этой фиксированной вершины, значит путь также не черный. □

Получается, первая фиксированная вершина должна принять такое значение, что все возможные условия, зависящие от нее, выполняются.

Поэтому после фиксации доступ к ней не понадобится. Можно рассматривать подграф зависимостей без неё и уже в нём искать вершину, на которую все условия (на меньшем графе) могут быть выполнены. По сути, именно этим мы и будем заниматься для получения доступа к целевой вершине: последовательно «выкидывать» из графа зависимостей вершины, у которых есть «хорошее» значение, пока не останется только целевая вершина.

После фиксации вершины v и возвращения остальных вершин к начальным значениям на подграфе $D \setminus \{v\}$ возникает индуцированный порядок $\preceq_{fix}$, являющийся подпорядком начального порядка $\preceq$ в том смысле, что если $p \preceq_{fix} k$, то и $p \preceq k$ (по тому же черному пути). Здесь fix — множество фиксированных вершин, состоящее пока из одной вершины v, $fix = \{v\}$.

По начальному состоянию мы можем вычислить какой порядок $\preceq_{sub}$ будет на любом подграфе D и по следствию из Утверждения 6 он будет совпадать с порядком $\preceq_{fix}$ на подграфе $D \setminus fix$ после фиксации соответствующих вершин и возвращения остальных вершин к начальным значениям. Поэтому для доступа к любой вершине подграфа нужны в точности те вершины, которые меньше нее в индуцированном порядке (это те вершины, в которые ведут черные пути из данной).

Следствие 4. *После фиксации первой вершины v граф зависимостей можно вернуть в состояние, в котором для доступа к нефиксированной вершине k нужны только вершины $p \preceq_{fix} k$.*

Доказательство. Следует из Утверждения 6 и Леммы 2. □

То есть после фиксации вершины, поскольку все оставшиеся «хорошо» зависят от нее, мы исключаем ее из рассмотрения и переходим к подграфу.

По Утверждению 7 мы знаем, что в момент фиксации некоторой вершины все нефиксированные вершины еще будут нужны для доступа к целевой. Однако после возвращения нефиксированных вершин к начальным значениям порядок $\preceq_{fix}$ на них может быть таким, что подграф зависимости для целевой вершины по черным ребрам не содержит те вершины, которые раньше были достижимы по черному пути, проходящему через фиксированную вершину. Это значит, что эти вершины на самом деле были фиксированы в ходе выполнения обратной последовательности действий (возвращения нефиксированных вершин к начальным значениям). Рассмотрим вершину p, которая при возвращении к начальным значениям окажется недостижимой по черному пути от целевой вершины $\mathbf{x}_0$. В ходе выполнения обратной цепочки действий будет состояние, в котором она поменяет значение в последний раз, и оно останется таким

же в состоянии с начальными значениями нефиксированных вершин, в котором мы видим, что менять ее больше не потребуется. Значит, по Лемме 2 это значение удовлетворяет всем условиям нефиксированных вершин и p необходимо добавить в множество фиксированных вершин fix. То есть на обратном ходе нам нужно собрать все вершины, которые принимают значения, удовлетворяющие всем условия от нефиксированных вершин и они войдут в порядок фиксации.

Теперь множество fix действительно содержит все фиксированные вершины, а в подграфе из нефиксированных вершин все сравнимы в целевой в порядке $\preceq_{fix}$.

5. Дерево порядков

До этого момента мы предполагали, что известна последовательность изменений значений переменных, которая является решением задачи проверки доступа. Были выяснены необходимые условия получения доступа, которые мы используем для нахождения этой последовательности. Будем использовать следующую структуру.

Определение 2. Деревом порядков по графу D *назовем упорядоченное дерево, удовлетворяющее следующим условиям:*

- *узлами дерева являются упорядоченные подмножества вершин графа зависимостей D;*
- *если узлом дерева является набор длины n, то этот узел имеет $n-1$ дочерний узел, каждый из которых связан со своим элементом из первых $n-1$ элементов данного узла;*
- *последним элементом набора k-ого дочернего узла является k-й элемент набора родительского узла;*
- *последним элементом набора корневой вершины дерева является целевая вершина t.*

Схематично узел дерева порядков представлен на рис. 4.

Чтобы избежать терминологического замешательства, будем вершины дерева порядков называть узлами или наборами, а слово «вершина» использовать только для элементов $\mathbf{x}$ вершин графа зависимостей. Листья дерева порядков — наборы, состоящие из одной вершины. Последнюю вершину набора будем называть *целевой* вершиной этого набора. Упорядоченный набор $N = \langle \mathbf{x}_{N1}, \ldots, \mathbf{x}_{Nk} \rangle$ вершин D *реализуем*, если для любой его вершины $\mathbf{x}_{Ni}$ существует допустимое значение, такое что $\mu(\mathbf{x}_{Nj}, \mathbf{x}_{Ni}) = 1$ для всех $j > i$. *Предками* узла N дерева порядков будем

называть все элементы последовательности родительских узлов вплоть до корня (у каждого узла только один родитель).

Фиксированными вершинами узла N для его дочернего узла Ni назовем все вершины $\mathbf{x}_{Nj}$, $j < i$ узла N. *Фиксированными вершинами дерева порядков для узла N* назовем все фиксированные вершины всех предков N. Это множество обозначим $F(N)$. Если по контексту понятно, какой узел текущий, обозначим просто F.

Узел дерева порядков *согласован*, если не содержит фиксированные вершины дерева порядков. Дерево порядков назовем *согласованным*, если все его узлы согласованы. Узлу N дерева порядков соответствует текущий порядок вершин $\preceq_{F(N)}$, индуцированный порядком $\preceq$ на подграфе $D \setminus F(N)$.

Набор N дерева порядков *возрастающий*, если для любого $i \leq k$ верно $\mathbf{x}_i \preceq_{F(N)} \mathbf{x}_k$ (все вершины в индуцированном порядке меньше последней) и он содержит все вершины $\mathbf{x}_i \preceq_{F(N)} \mathbf{x}_k$.

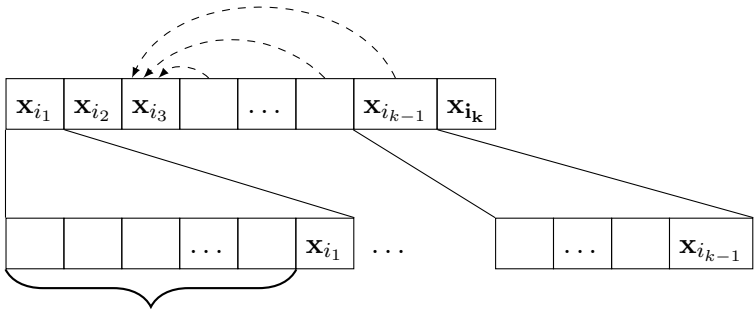


Рис. 4. Схематичное представление узла дерева порядков.

Пояснение. Каждый узел дерева порядков — это порядок фиксации для доступа к последней вершине этого узла. Условие реализуемости набора — это в точности условие выбора фиксированного значения по Лемме 2 в терминах дерева порядка: значение фиксируемой вершины должно быть выбрано таким образом, чтобы условия доступа к нефиксированным вершинам выполнялись (все нефиксированные вершины в узле находятся справа от текущей). Согласованность нужна, чтобы не пытаться изменить те вершины, которые уже фиксированы ранее. Набор возрастающий, поскольку граф зависимостей упорядочен и для получения доступа к вершине k могут понадобиться только вершины меньшего порядка в текущем графе зависимостей. Листовые вершины соответствуют тем вершинам графа зависимостей, которые доступны в начальном состоянии.

Определение 3. *Дерево порядков назовем* корректным, *если (1) для каждого его узла набор вершин реализуем и (2) для любого узла набор его вершин возрастающий.*

5.1. Критерий корректности дерева порядков

Теорема 1. *Корректное и согласованное дерево порядков, построенное по графу $D(x^0)$, существует тогда и только тогда, когда доступ к целевой переменной $\mathbf{x}_0$ может быть получен из начального состояния x^0.*

Доказательство. Пусть задано корректное согласованное дерево порядков. Покажем, что в этом случае целевая переменная достижима. Сначала опишем цепочку изменений вершин, которую мы строим по дереву порядков, а ниже покажем, что все действия в ней допустимы.

Через v будем обозначать узел дерева порядков. Обозначим вектор дочерних узлов узла v Через $d(v) = \langle d(v)_0, \ldots, d(v)_{N(v)} \rangle$. В узле v им соответствуют вершины $(v_0, \ldots, v_i, \ldots v_{N(v)})$. Через $v.t$ обозначим последнюю (целевую) вершину набора v, а через $val(v.t)$ — её значение. Для листового узла v $d(v) = \varnothing$, а набор v состоит только из $v.t$. Через F обозначается текущее множество фиксированных вершин.

Весь процесс описывается рекурсивным алгоритмом GAESBR (Алгоритм 5.1). Название является аббревиатурой Get Access, Edit the Subject, get Back the Rest.

Algorithm 5.1 Получение доступа к последней вершине узла v дерева порядков T.

function GAESBR(T, v, *newvalue*)
Изменение значения вершины $v.t$ на newvalue.
$history \leftarrow \varnothing$
if $d(v) \neq \varnothing$ **then**
 for $i \leftarrow 0$ **to** $N(v)$ **do**
  выбрать новое значение $newval_i$ для i-ой вершины согласно Лемме 2
  добавить в стек $history$ тройку $\langle v, i, val(v_i) \rangle$
  GAESBR(T, $d(v)_i$, $newval_i$)
$val(v.t) \leftarrow newvalue$
Отменить все остальные шаги
while $history \neq \varnothing$ **do**
 извлечь из стека последнее действие $\langle v, i, oldval \rangle$
 $val(v_i) \leftarrow oldval$
return

Будем присваивать вершинам значения в порядке обхода дерева в глубину: доходим до доступной вершины и присваиваем ей такое значение, которое нужно в текущем узле дерева порядков. После этого все действия, совершенные для получения доступа к этой вершине, отменяются: значения в обратном порядке возвращаются на предыдущие.

Если v — листовой узел, то $d(v)$ пусто и мы сразу присваиваем вершине $v.t$ значение, нужное в родительском узле — такое, благодаря которому набор родительского узла реализуем: что если ребро из каждой вершины, которая в родительском наборе находится справа от $v.t$, существует, то соответствующая ему метка должна выполняться (иметь значение 1). То есть все условия на эту вершину должны выполняться, и ее значение локально фиксируется: до конца этого родительского узла.

Если v не листовой узел, то мы последовательно присваиваем нецелевым вершинам набора новые значения в соответствии с реализуемостью набора v. Когда мы доходим до целевой вершины узла, у нас есть к ней доступ. Почему: рассмотрим все вершины, от которых она зависит.

Для получения доступа к вершине $v.t$ по определению индуцируемого порядка нужны только вершины $\preceq_{F(v)} v.t$. Все эти вершины входят в узел v и в ходе выполнения алгоритма для узла v принимали значения согласно реализуемости набора v, а значит от них $v.t$ зависит с меткой 1. Значит, $v.t$ доступна. Присваиваем ей значение в соответствии с реализуемостью родительского узла.

А теперь нам нужно вернуть все остальные вершины узла v к тем значениям, которые у них были в начальном состоянии. В родительском узле $v.t$ — первая фиксированная вершина на подграфе из $v.t$ и нефиксированных вершин этого узла с индуцированным на них порядком. По следствию 4 систему можно привести в такое состояние, что порядок на нефиксированных вершинах $\preceq_F$ таков, что для получения доступа к любой нефиксированной вершине понадобятся только вершины меньше нее в порядке $\preceq_F$ на подграфе без $v.t$.

Значит, после каждого возвращения значений нефиксированных вершин к начальным доступ к каждой из них снова зависит только от вершин с меньшим порядком на текущем подграфе, что является предположением для начала действия функции.

Таким образом, с помощью описанной функции по корректному согласованному дереву порядков мы получим доступ к целевой вершине.

Пусть теперь известно, что целевая переменная достижима. Тогда существует порядок фиксаций, приводящий к доступу к целевой вершине. Этот порядок и будет набором корневой вершины дерева. Значения, которые вершины принимают в момент фиксации, соответствуют реализуемому набору по лемме 2. Аналогично, и для каждой вершины графа

существует порядок фиксации, приводящий к доступу к ней. Вопрос в том, какие вершины используются в этом порядке.

По утверждению 6 после получения доступа к некоторой вершине корневого набора остальные вершины можно откатывать в начальное состояние, так что частичный порядок на нефиксированных вершинах является подпорядком начального порядка. Поэтому после получения доступа к каждой из вершин набора корня можно произвести обратную последовательность действий для возвращения нефиксированных вершин к начальным значениям. И после этого для доступа к каждой вершине набора корневого узла понадобятся только вершины индуцированным порядком меньше нее. Это условие означает, что дочерние узлы корня возрастающие.

Оно же значит, что узлы согласованы: в них используются только нефиксированные вершины.

Все это верно, если к каждому узлу дерева относиться как к корневому на подграфе, содержащем вершины этого узла. Таким образом, дерево порядков достраивается до листов и является корректным и согласованным. □

5.2. Построение дерева порядков

Задано множество вершин $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ и для каждой вершины $\mathbf{x}_i \in \mathbf{x}$ задан набор $S_{i1}, S_{i2}, \ldots, S_{ik_i}$ подмножеств $\mathbf{x}$. Порядком на $\mathbf{x}$ назовем взаимно-однозначного отображение $f : \mathbf{x} \to \{1, \ldots, n\}$. Через $succ_f(\mathbf{x}_i)$ обозначим множество вершин, которые находятся правее $\mathbf{x}_i$ в порядке f, то есть $succ_f(\mathbf{x}_i) = \{\mathbf{x}_j \in \mathbf{x} \mid f(\mathbf{x}_j) > f(\mathbf{x}_i)\}$. Требуется найти порядок f при котором выполняется следующее условие:

$$\forall i \le n \; \exists m \le k_i \;\; succ_f(\mathbf{x}_i) \subseteq S_m. \tag{2}$$

То есть для каждой вершины $\mathbf{x}_i$ найдется набор S_{im}, содержащий все вершины, которые в последовательности f находятся после $\mathbf{x}_i$. Порядок f, удовлетворяющий условию (2) будем называть *корректным*.

Здесь множества S_{ij} соответствуют возможным значениям $val.in(\mathbf{x}_i)$, где единицы стоят в разрядах, отвечающих вершинам, входящим в S_{ij}.

Утверждение 8 (Сдвиг влево)**.** *Пусть задано корректное отображение f. Если существует вершина $\mathbf{x}_i$, для которой:*

1) *$f(\mathbf{x}_i) > 1$ ($\mathbf{x}_i$ имеет левого соседа в порядке f) и*

2) *найдется набор S_{im}, такой что $\{f^{-1}(f(\mathbf{x}_i) - 1)\} \cup succ_f(\mathbf{x}_i) \subseteq S_{im}$ (набор S_{im} содержит левого соседа $\mathbf{x}_i$ и все вершины, которые стоят справа от $\mathbf{x}_i$), то*

порядок f', отличающийся перестановкой $\mathbf{x}_i$ с его левым соседом, является корректным.

Доказательство. Множества $succ_{f'}(\mathbf{x}_j)$ для всех вершин с номерами $j \in (1, \dots, f(\mathbf{x}_i) - 2, f(\mathbf{x}_i) + 1, \dots n)$ остались прежними.

Множество $succ_{f'}(\mathbf{x}_i) \subseteq S_{im}$ по условию, значит для $\mathbf{x}_i$ выполняется условие корректности f'.

Для $succ_{f'}(f^{-1}(f(\mathbf{x}_i) - 1))$ можно взять то же множество S', которое использовалось в f: оно покрывает все вершины, находящиеся правее в новом порядке. □

Алгоритм нахождения корректного порядка Если для каждого i задано единственное множество S_{i1}, то есть $k(i) = 1$, то задача в точности является задачей топологической сортировки ориентированного графа и решается за линейное время [9].

Для произвольного набора множеств применяем аналогичный алгоритм. Находим все вершины $\mathbf{x}_i$, для которых существует набор S_{im}, такой что $(\mathbf{x} \setminus \{\mathbf{x}_i\}) \subseteq S_{im}$. Они будут первыми в искомом порядке f. «Удаляем их из рассмотрения» и решаем аналогичную задачу на меньшем множестве вершин. Таким образом строится корневой порядок фиксации вершин. После этого надо перейти на подграф для доступа к каждой вершине этого порядка и найти порядок фиксации на нем, проверив аналогичные условия. Проделать это с каждым порядком.

Если входными данными задачи считать множество вершин $\mathbf{x}$ и наборы множеств S_{ij}, и предположить, что множество S_{ij} проверяется на полноту за $O(1)$, то сложность алгоритма для нахождения порядка одного узла $O\left((\sum_{i=1}^{n} k_i)^2\right)$ — квадратичная от количества подмножеств S_{ij}. Возьмем одну вершину, проверим все ее множества на полноту, перейдем к другой. В худшем случае нам придется проверить все $\sum_{i=1}^{n} k_i$ множеств. Если одна вершина подойдет, то ее берем первой (без ограничения общности она имеет номер 1) и на второе место нам надо проверить уже $\sum_{i=2}^{n} k_i$, и так далее. Всего проверок будет не больше квадрата от числа подмножеств.

Трудность в том, что подмножеств S_{ij} может быть много. Если от вершины $\mathbf{x}_i$ зависят все остальные вершины, то множеств S_{ij} может быть 2^N, где $N+1$ — число вершин (если есть все возможные пересечения множеств, удовлетворяющих условиям вершин). Однако вообще говоря нам не требуются все множества: если частично упорядочить все множества

для одной вершины по вложению, достаточно из каждой цепи взять максимальный элемент (такое множество, очевидно, можно использовать в наборе вместо вложенного в него). Тогда их не больше, чем длина максимальной антицепи $C_N^{[N/2]}$, что все еще экспоненциально много.

Напомним, что мы считаем, что сами множества S_{ij} уже построены, но вообще говоря их построение может внести дополнительную сложность. Вопрос в том, как строятся множества S_{ij}, нас до сих пор не интересовал. Вспомним, что они связаны с пересечением множеств доступности A_{ij}, которые как-то задаются, и сложность задания этих множеств влияет на количество S_{ij} и на сложность выполнения алгоритма: построение пересечений множеств доступности и поиск значения, принадлежащего пересечению нескольких множеств доступности, может быть трудоемкой процедурой при некоторых способах задания множеств (например, если множества заданы как решение уравнений или как удовлетворяющие некоторому предикату).

5.3. Сложность построения дерева порядков в частных случаях графа зависимостей

Если множества A_{ij} — отрезки, то даже в предположении, что $\mathbf{x}_i$ зависит от всех остальных вершин, экспоненты в количестве S_{ij} не возникает, поскольку классов пересечения n отрезков на прямой не более $2n$, значит количество множеств S_{ij} не больше $2N$, где N — число вершин. Таким образом, экспоненциальная сложность алгоритма построения дерева порядков может не являться препятствием для практического применения.

Если множества A_{ij} в исходных данных задаются, например предикатами, то на «сложность» множеств влияет сложность формул. Отрезок задается простой формулой $(\mathbf{x}_i > a)\&(\mathbf{x}_i < b)$. Но если предикаты могут быть какими угодно, понятно, что формула может быть сложной и алгоритм с ней будет работать ожидаемо долго.

Входные данные могут оказаться простыми и в другом смысле. Если граф зависимостей D является ациклическим ориентированным графом (то есть упорядочивается так, что обратных ребер вообще нет), то доступ к корневой всегда можно получить. Предположим, что граф — полное дерево, то есть содержит все возможные ребра (если в исходном графе отсутствовало некоторое ребро, то добавим фиктивное ребро, метка которого тождественно равна 1). Тогда вершины нумеруются $1 \preceq 2 \ldots \preceq N$. Можно выбрать порядок фиксации для каждой вершины k следующим образом: $k-1, k-2, \ldots, 2, 1, k$. В этом порядке единственное существующее ребро, ведущее из вершины справа — ребро из вершины k, а значит всегда существует значение, удовлетворяющее условию этого ребра. Несложно проверить, что количество действий в последовательности из-

менений значений вершин не превосходит $O(2^N)$, где N — число вершин, и эта оценка является точной.

6. Заключение

В работе рассмотрена задача проверки невозможности получения доступа к выделенному объекту информационной системы с атрибутивной политикой безопасности. Показано, что при некоторых ограничениях на структуру правил доступа, задача сводится к задаче интеллектуального планирования, что дает возможность применять для ее решения известные эффективные методы планирования с использованием эвристических характеристик задачи [7].

В данной работе доказан критерий возможности получения доступа в зависимости от входных данных задачи. Вводится понятие дерева порядков — структуры, с помощью которой проверяется выполнение критерия. Предложен алгоритм получения доступа с помощью дерева порядков.

Предложен алгоритм, отвечающий на вопрос «возможно ли получение доступа к заданному объекту». Этот алгоритм имеет экспоненциальную оценку сложности в зависимости от числа объектов информационной системы. При этом показано, что при выполнении некоторых дополнительных условий на структуру графа зависимостей или множеств доступности объектов, которые соответствуют правилам политики безопасности информационной системы, сложность этого алгоритма может быть уменьшена до полиномиальной.

Возможными направлениями дальнейших исследований могут быть: поиск кратчайшей последовательности действий при заданном корректном дереве порядков, поиск оптимального порядка вершин в узле дерева порядков в процессе построения корректного дерева, оценка алгоритмической сложности проверки корректности дерева порядков.

Список литературы

[1] А.В. Галатенко, Плетнёва В.А., “Выразимость моделей безопасности take–grant и невлияния в модели CBAC”, *Программная инженерия*, №1, 40–46.

[2] Sergey Afonin, Antonina Bonushkina., “Validation of safety-like properties for entity-based access control policies.”, *Advances in Soft and Hard Computing*, Advances in Intelligent Systems and Computing, **889**, Springer International Publishing, 259–271.

[3] Sandra Alves, Maribel Fernández, “A graph-based framework for the analysis of access control policies”, *Theoretical Computer Science,*, **685**, 3–22.

[4] Clara Bertolissi, Maribel Fernández., “A metamodel of access control for distributed environments: Applications and properties”, *Information and Computation*, **238**, 187–207.

[5] Avrim L Blum, Merrick L Furst, "Fast planning through planning graph analysis.", *Artificial intelligence*, **90**:1-2, 281–300 ..

[6] Tom Bylander, "The computational complexity of propositional STRIPS planning.", *Artificial Intelligence*, **69**:1-2, 165–204.

[7] Malik Ghallab, Dana Nau, Paolo Traverso, *Automated Planning: theory and practice*, Elsevier.

[8] Vincent C Hu, D Richard Kuhn, David F Ferraiolo, Jeffrey Voas, "Attribute-based access control", *Computer*, **48**:2, 85–88.

[9] A. B. Kahn, "Topological sorting of large networks", *Communications of the ACM*, **5**:11, 558–562.

Attribute-based access control policy analysis using automated planning technique

Afonin S.A., Bonushkina A. Yu.

The paper considers the problem of testing the possibility of a user of an information system gaining access to the selected object with a given attribute-based security policy. It is shown that under some restrictions on information system model and policy rules, this task is reduced to the task of automated planning. There is a tree structure determined, the construction of which corresponds to planning in the space of plans and allows to take into account the specifics of the problem when constructing heuristic algorithms for checking access. It is proved that the existence of such a structure is a necessary and sufficient condition for the possibility of getting an access to the target.

This work was supported by RFBR Grant 18-07-01055.

Keywords: ABAC, access control, automated planning.