*Математические структуры и моделирование*
2009, вып. 20, с. 160–172

УДК 65.012.810(075.8)

# АЛГОРИТМЫ АНАЛИЗА БЕЗОПАСНОСТИ СОСТОЯНИЙ КОМПЬЮТЕРНОЙ СИСТЕМЫ ДЛЯ МОДЕЛИ TAKE-GRANT

**Д.М. Бречка**

В статье рассматриваются алгоритмы поиска $tg$-путей, островов и мостов в графах доступов для дискреционной модели Take-Grant.

## Введение

Современные системы защиты информации создаются и тестируются исходя из постулата, впервые предложенного в знаменитой «Оранжевой книге»: «Все вопросы безопасности компьютерных систем могут быть разрешены в терминах доступов субъектов к объектам». Таким образом, основной целью любой системы защиты является разграничение доступа, позволяющего разделить компоненты компьютерной системы таким образом, что пользователи системы получают доступ лишь к той информации, которая необходима им для выполнения своих функциональных обязанностей.

При моделировании систем с дискреционным разграничением доступа строится система, отслеживающая все возможные доступы субъектов компьютерной системы к объектам компьютерной системы. Одним из представлений множества возможных доступов является граф доступов. Наиболее развитой моделью распространения прав доступа по графу доступов является Take-Grant [1]. Центральным понятием этой модели служит «безопасное состояние». В рамках модели Take-Grant выделены критерии безопасности состояния компьютерной системы. Однако остается открытым вопрос об алгоритмах проверки безопасности конкретного состояния. В данной статье предложены некоторые алгоритмы, позволяющие однозначно ответить, является ли заданное состояние компьютерной системы безопасным.

## 1. Описание модели Take-Grant

Модель Take-Grant — это модель дискреционного разграничения доступа в информационной системе. Данная модель является расширением классической модели HRU, ориентированной на анализ распространения прав в системе [1, 2].


Омский государственный университет имени Ф.М. Достоевского.
E-mail: dbrechkawork@yandex.ru

### 1.1. Основные положения

Согласно модели Take-Grant, информационная система представляется в виде ориентированного графа, в котором вершинами являются субъекты и объекты (активные и пассивные сущности системы соответственно), а дуги представляют собой установленные права субъектов на объекты. При этом субъекты рассматриваются как активизированные объекты, то есть могут существовать дуги между субъектами. Помимо прав, присутствующих в модели HRU, в Take-Grant добавляются два дополнительных права: Take (t) — право брать права доступа у какого-либо субъекта и Grant (g) — право предоставлять права доступа на какой-либо объект.

Состояния системы изменяются под воздействием четырех видов команд:

1. $Take(\alpha, x, y, z)$ — субъект $x$ берет права $\alpha$ у объекта $y$ на объект $z$.
2. $Grant(\alpha, x, y, z)$ — субъект $x$ дает права $\alpha$ объекту $y$ на объект $z$.
3. $Create(\alpha, x, y)$ — субъект $x$ создает объект $y$ с правами $\alpha$ на него.
4. $Remove(\alpha, x, y)$ — субъект $x$ удаляет права $\alpha$ на объект $y$.

При анализе безопасности системы исследуется возможность получения доступа каким-либо субъектом на какой-либо объект. При этом анализируются установленные на начальный момент времени отношения в системе и рассматриваются возможности перехода системы в различные состояния путем изменения прав доступа на объекты. Имеется два возможных варианта получения прав доступа в системе: санкционированное получение прав и похищение прав.

### 1.2. Получение прав доступа

**Определение 1.** Предикат *«возможен доступ»*$(\alpha, x, y, G_0)$ истинен тогда и только тогда, когда существуют графы доступов $G_1$, $G_2$, ...$G_n$ такие, что $G_1 \vdash_{c_1} G_2 \vdash_{c_2} ... \vdash_{c_n} G_n$, где $G_0$ — исходный граф системы, $G_1$, $G_2$, ...$G_n$ — последовательные преобразования исходного графа под воздействием команд $c_1$, $c_2$, ...$c_n$.

**Определение 2.** Вершины графа доступов называются $tg$-связными (соединены $tg$-путем), если в графе существует такой путь между этими вершинами, что каждая дуга пути содержит право Take либо Grant без учета направления дуг.

Для графа доступов, вершинами которого являются только субъекты, справедлива следующая теорема:

**Теорема 1.** *В графе доступов $G_0$, содержащим только вершины-субъекты, предикат* «возможен доступ»$(\alpha, x, y, G_0)$ *истинен тогда и только тогда, когда выполнены следующие условия:*

1. *В графе $G_0$ существуют субъекты $s_1, s_2, ..., s_m$, связанные дугами с вершиной $y$.*

2. *Субъект $x$ в графе $G_0$ соединен $tg$-путем с каждым субъектом $s_i$ для $i = 1, 2, \ldots m$.*

Доказательство приводится в [1].

Подобная теорема существует и для произвольного графа доступов. Для того чтобы сформулировать теорему, необходимо прежде определить несколько понятий.

**Определение 3.** Островом в произвольном графе доступов называется его максимальный $tg$-связный подграф, состоящий только из вершин субъектов.

**Определение 4.** Мостом в произвольном графе доступов называется $tg$-путь, проходящий через вершины-объекты, концами которого являются вершины-субъекты; при этом словарная запись $tg$-пути должна иметь вид $\overrightarrow{t}^*$, $\overleftarrow{t}^*$, $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$, $\overrightarrow{t}^*\overleftarrow{g}\overleftarrow{t}^*$, где символ $^*$ означает многократное (в том числе нулевое) повторение.

**Определение 5.** Начальным пролетом моста в произвольном графе доступов называется $tg$-путь, проходящий через вершины-объекты, началом которого является вершина-субъект, а концом — вершина-объект; при этом словарная запись пути имеет вид $\overrightarrow{t}^*\overrightarrow{g}$.

**Определение 6.** Конечным пролетом моста в произвольном графе доступов называется $tg$-путь, проходящий через вершины-объекты, началом которого является вершина-субъект, а концом — вершина-объект; при этом словарная запись пути имеет вид $\overrightarrow{t}^*$.

**Теорема 2.** *В произвольном графе доступов для объектов $x$ и $y$ предикат* «возможен доступ»$(\alpha, x, y, G_0)$ *истинен тогда и только тогда, когда выполнены следующие условия:*

1. *В исходном графе $G_0$ существуют объекты $s_1, s_2, ..., s_m$, соединенные дугами с объектом $y$.*

2. *В исходном графе $G_0$ существуют субъекты $x'_1, x'_2, ..., x'_m$ и $s'_1, s'_2, ..., s'_m$ такие, что:*

   - *$x'_i = x$ или $x'_i$ соединен с $x$ начальным пролетом моста в графе $G_0$ для $i = 1, 2, ..., m$;*
   - *$s'_i = s$ или $s'_i$ соединен с $s$ конечным пролетом моста в графе $G_0$ для $i = 1, 2, ..., m$.*

3. *В исходном графе $G_0$ для каждой пары $(x'_i, s'_i)$, где $i = 1, 2, ..., m$, существуют острова $I_{i,1}, ..., I_{i,u_i}$ при $u_i \geq 1$ такие, что $x'_i \in I_{i,1}$, $s'_i \in I_{i,u_i}$, и существуют мосты между островами $I_{ij}$ и $I_{ij+1}$ при $j = 1, 2, ..., u_{i-1}$.*

Доказательство приводится в [1].

## 2. Граф доступов из вершин-субъектов

Согласно Теореме 1 для того, чтобы выяснить возможность доступа одного субъекта к другому в графе, который содержит только вершины-субъекты, необходимо выяснить, существует ли в графе между этими субъектами $tg$-путь. При этом направление дуг не учитывается. Для поиска $tg$-пути можно использовать алгоритм Дейкстры [3] при некоторой модификации графа доступов.

## 3. Поиск $tg$-путей

Для начала приведем описание самого алгоритма Дейкстры. Алгоритм позволяет найти кратчайший путь в графе между двумя выделенными вершинами $s$ и $t$. Перед началом работы алгоритма все вершины и дуги не помечены. В ходе выполнения каждой вершине $x$ присваивается число $d(x)$, равное длине кратчайшего пути между $s$ и $x$, включающего только помеченные вершины и дуги. Шаги алгоритма:

1. Присвоить $d(s) = 0$ и $d(x) = \infty$ для всех $x$, отличных от $s$. Пометить вершину $s$ и присвоить $y = s$. Где $y$ - последняя помеченная вершина.

2. Для каждой непомеченной вершины $x$ пересчитать величину $d(x)$:

   $d(x) = min\{d(x), d(y) + a(y, x)\}$, где $a(y, x)$ — длина дуги от $y$ к $x$.

   Если $d(x) = \infty$ для всех непомеченных $x$, то закончить процедуру: в графе нет пути из $s$ в непомеченные вершины. Иначе — пометить ту из вершин $x$, для которой $d(x)$ минимальна. Вместе с вершиной $x$ окрашивается дуга, ведущая в данную вершину. Присвоить $y = x$.

3. Если $y == t$, то закончить процедуру: путь из $s$ в $t$ найден — состоит из помеченных дуг и вершин. Иначе — перейти на шаг 2.

Для того, чтобы можно было применить данный алгоритм для поиска $tg$-путей, произведем модификацию графа доступов описанным ниже образом.

Во-первых, будем считать, что направления дуг не имеют значения (согласно определению 2 направления дуг в $tg$-пути не учитываются).

Во-вторых, присвоим всем дугам, не содержащим прав Take или Grant, вес, равный бесконечности ($d(e) = \infty$).

В-третьих, для всех ребер, содержащих права Take или Grant, присвоим вес, равный единице ($d(e) = 1$), так как не существует принципиальной разницы, используется право Take или Grant. При этом если между двумя вершинами существует больше одного такого ребра, то необходимо оставить лишь одно из них.

После такой модификации к графу можно применять алгоритм Дейкстры. При этом будет найден кратчайший $tg$-путь между указанными вершинами, но для нас нет принципиальной разницы — любой из существующих путей подойдет.

## 4. Произвольный граф доступов

Согласно Теореме 2 для того, чтобы выяснить возможность доступа одного субъекта к другому в произвольном графе доступов, необходимо, чтобы:

1) в графе существовали мосты;

2) исследуемые субъекты были соединены в графе доступов начальным и конечным пролетом моста с островами;

3) сами острова в графе были соединены мостами.

### 4.1. Поиск островов

Для поиска островов в графе можно воспользоваться алгоритмом Флойда [3], при этом граф доступов необходимо определенным образом модифицировать.

Приведем описание алгоритма Флойда. Алгоритм предполагает, что перед началом его работы все вершины графа должны быть пронумерованы. Длина кратчайшего пути из вершины $i$ в вершину $j$ обозначается через $d_{ij}^m$, где $m$ — число промежуточных вершин пути. Если между вершинами $i$ и $j$ не существует ни одного пути, то $d_{ij}^m = \infty$. Величина $d_{ii}^0 = 0$.

Формально алгоритм Флойда имеет следующие шаги:

1. Пронумеровать все вершины исходного графа целыми числами от 1 до $N$. Определить матрицу $D^0$ такую, что каждый ее элемент $(i, j)$ соответствует величине $d_{ij}^0$, то есть, каждый элемент матрицы $D^0$ выражает длину дуги между вершинами $i$ и $j$ в исходном графе. Если в исходном графе вершины не соединены дугами, то $d_{ij}^0 = \infty$. Для каждого $i$ $d_{ii}^0 = 0$.

2. Для $m$, последовательно принимающего значения $1, 2, ..., N$, определить по величинам элементов матрицы $D^{m-1}$ величины матрицы $D^m$, используя отношение $d_{ij}^m = min\{d_{im}^{m-1} + d_{mj}^{m-1}, d_{ij}^{m-1}\}$.

По окончании данной процедуры величина элемента $(i, j)$ матрицы $D^N$ будет определять длину кратчайшего пути из вершины $i$ в вершину $j$.

Для применения алгоритма Флойда необходимо модифицировать граф доступов таким же образом, что и для применения алгоритма Дейкстры.

1. Удалим направления дуг.

2. Присвоим всем ребрам, не содержащим права Take и Grant, бесконечный вес. Заметим, что по условию алгоритма вес ребра между двумя вершинами равен бесконечности, если ребра между вершинами не существует. Фактически это означает, что мы удаляем из графа все ребра, не содержащие прав Take или Grant.

3. Всем ребрам, содержащим права Take и Grant, присвоим вес, равный единице.

Для модифицированного графа будем применять алгоритм Флойда следующим образом:

1. Выберем вершину из множества всех вершин графа ($E$), применим для нее алгоритм Флойда. Алгоритм обнаружит все вершины в графе, связанные с выбранной $tg$-путем. Множество вершин, выбранное алгоритмом, будет являться островом. Обозначим его через $I_1$.

2. Множество вершин графа без $I_1$ обозначим через $E/I_1$. Выберем вершину из множества $E/I_1$ и применим для нее алгоритм Флойда. Алгоритм обнаружит все вершины, принадлежащие множеству $E/I_1$, связанные с выбранной $tg$-путем. Множество вершин, выбранное алгоритмом, обозначим через $I_2$. Это множество будет являться вторым мостом в графе.

Будем применять данную процедуру до тех пор, пока все вершины графа не будут распределены по своим островам. На некотором шаге $N$ множество $E/I_{N-1}$ станет пустым — это значит, что обнаружены все мосты в графе доступов.

### 4.2. Поиск мостов

После того как найдены все острова в графе доступов, необходимо выяснить, соединены ли они мостами. Согласно определению 4, мостом в графе доступов называется $tg$-путь, проходящий через вершины-объекты, концами которого являются вершины-субъекты; при этом словарная запись $tg$-пути имеет вид $\overrightarrow{t}^*$, $\overleftarrow{t}^*$, $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$, $\overrightarrow{t}^*\overleftarrow{g}\overleftarrow{t}^*$.

Задачу отыскания моста в графе можно сформулировать следующим образом: необходимо построить алгоритм, который, анализируя вершины графа доступов и установленные права между ними, мог определить $tg$-путь заданного вида. Похожая задача решается для построения алгоритмов-распознавателей грамматик. Построение распознавателей грамматик подробно описывается в [4].

Построим распознаватель мостов в графе доступов. При этом мост $\overrightarrow{t}^*$ будем рассматривать как частный случай моста $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$, то есть для поиска этих двух типов мостов будем использовать один алгоритм.

Введем ряд обозначений. Начальную и конечную вершину моста будем обозначать $S(e)$ и $F(e)$ соответственно. Вершину, в которой мы находимся в данный момент, будем обозначать $C(e)$, а рассматриваемую вершину — $W(e)$. Дугу между вершинами $e_1$ и $e_2$, содержащее право $\overrightarrow{g}$, обозначим $\overrightarrow{g}(e_1, e_2)$. Для дуг с правами $\overrightarrow{t}$ и $\overleftarrow{t}$ также введем соответствующие обозначения $\overrightarrow{t}(e_1, e_2)$ и $\overleftarrow{t}(e_1, e_2)$.

Для начала рассмотрим мосты $\overrightarrow{t}^*$ и $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$.

Будем искать мост между островами графа $I_1$ и $I_2$. Пусть наш распознаватель находится в начальном состоянии, для которого определены вершины $S(e)$ и $F(e)$ ($S(e) \in I_1$, $F(e) \in I_2$). В самом начале работы $C(e) = S(e)$. Рассмотрим возможные варианты функционирования распознавателя. Очевидно,

что прямой дуги между $C(e)$ и $F(e)$, содержащей право $t$ или $g$, быть не может, так как в этом случае обе вершины принадлежали бы одному острову. Для поиска моста между данными вершинами нам необходимо рассмотреть вершины-объекты графа доступов. Обозначим множество всех объектов через $O$. Выберем из этого множества вершину $W(e)$ и рассмотрим, каким образом эта вершина может быть связана с $C(e)$. Так как мы ищем мост $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$, то нас интересует, существует ли дуга $\overrightarrow{t}$ или $\overrightarrow{g}$ между $C(e)$ и $W(e)$. Если такой дуги нет, то нам необходимо выбрать для рассмотрения следующую вершину $W(e)$ из $O$. Возможна ситуация, когда мы рассмотрели все вершины из $O$, но подходящей так и не нашли, в этом случае нам следует выбрать другую вершину $S(e) \in I_1$. Если же мы рассмотрели все вершины из $I_1$ и $O$, но так и не нашли ни одной подходящей нам пары, то, очевидно, моста не существует и распознаватель переходит в конечное состояние с отрицательным результатом. Назовем это состояние $E^-$. Допустим, что, перебирая пары $C(e)$ - $W(e)$, мы обнаружили дугу содержащую право $\overrightarrow{t}$ ($\overrightarrow{t}(C(e), W(e))$, это значит, что мы нашли дугу, которая может являться частью моста. Нам необходимо запомнить эту дугу, для этого пометим ее и вершину $W(e)$ *положительно*, сделаем рассматриваемую вершину текущей ($W(e) = C(e)$) и переведем распознаватель в новое состояние, которое будем называть $\overrightarrow{T}^+$. Нами осталась не рассмотрена ситуация, когда, перебирая пары $C(e)$ - $W(e)$, мы обнаружили дугу содержащую право $\overrightarrow{g}$ ($\overrightarrow{g}(C(e), W(e))$. В этом случае пометим дугу и вершину $W(e)$ положительно, выполним присвоение $W(e) = C(e)$ и переведем распознаватель в новое состояние, которое будем называть $\overrightarrow{G}^+$.

Допустим, что в ходе своего функционирования распознаватель перешел в состояние $\overrightarrow{T}^+$. В первую очередь нам необходимо проверить, существуют ли $tg$-дуги между текущей и конечной вершиной моста. Пусть существует дуга $\overrightarrow{t}(C(e), F(e))$, это означает, что распознаватель обнаружил мост типа $\overrightarrow{t}^*$; необходимо пометить эту дугу и перевести распознаватель в новое состояние, которое назовем $E^+$. Другой вариант предполагает, что существует дуга $\overrightarrow{g}(C(e), F(e))$, это означает, что распознаватель обнаружил мост типа $\overrightarrow{t}^*\overrightarrow{g}$. Пометим данную дугу и вершину положительно и переведем распознаватель в состояние $E^+$. Рассмотрим варианты, когда $tg$-дуги между текущей и конечной вершиной моста не существует. В этом случае распознаватель выбирает вершину $W(e) \in O$ из тех, что еще не помечены. Пусть существует дуга $\overrightarrow{t}(C(e), W(e))$ — пометим дугу и вершину $W(e)$ положительно, сделаем рассматриваемую вершину текущей ($C(e) = W(e)$); распознаватель остается в состоянии $\overrightarrow{T}^+$ и выбирает следующую вершину $W(e) \in O$. Если существует дуга $\overrightarrow{g}(C(e), W(e))$ — пометим дугу и вершину положительно, сделаем рассматриваемую вершину текущей и переведем распознаватель в состояние $\overrightarrow{G}^+$. Наконец, возможна ситуация, когда мы перебрали все непомеченные вершины из множества $O$ и не нашли дуг $\overrightarrow{t}$ или $\overrightarrow{g}$ — это значит, что мы зашли в тупик. Попробуем вернуться к предыдущей вершине и выбрать другую, отличную от текущей. Для этого переведем распознаватель в состояние, которое будем называть $\overrightarrow{T}^-$.

Рассмотрим состояние $\overrightarrow{T}^-$. Распознаватель попадает в него из состояния $\overrightarrow{T}^+$

в том случае, если, рассмотрев все непомеченные вершины множества $O$, он не нашел дуг $\overrightarrow{t}$ или $\overrightarrow{g}$ между вершиной $C(e)$ и $W(e)$. Попробуем выбрать вершину, отличную от текущей. Для этого пометим $C(e)$ *отрицательно*, сделаем текущей предыдущую помеченную положительно вершину ($C(e) = C(e-1)$) и вернем распознаватель в состояние $\overrightarrow{T}^+$. Текущую вершину необходимо пометить отрицательно для того, чтобы распознаватель вновь не выбрал ее в состоянии $\overrightarrow{T}^+$. Возможна ситуация, когда, выбирая предыдущую вершину, мы попадем в вершину $S(e)$. Это, очевидно, означает, что между вершинами $S(e)$ и $F(e)$ не существует моста и необходимо вернуть распознаватель в начальное состояние для выбора других $S(e)$ и $F(e)$.

Пусть в ходе своей работы распознаватель перешел в состояние $\overrightarrow{G}^+$. Проверим, существует ли дуга $\overleftarrow{t}$ между вершинами $C(e)$ и $F(e)$. Если да, то это означает, что обнаружен мост $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$. Пометим данную дугу положительно и переведем распознаватель в состояние $E^+$. Если нужной нам дуги между $C(e)$ и $F(e)$ не существует, то необходимо выбрать непомеченную вершину $W(e)$ из множества $O$. Если существует дуга $\overleftarrow{t}\,(C(e), W(e))$, то пометим эту дугу и вершину $W$(e) положительно, присвоим $C(e) = W(e)$, и переведем распознаватель в новое состояние, которое будем называть $\overleftarrow{T}^+$. Все прочие варианты нас не интересуют, однако возможна такая ситуация, когда мы перебрали все непомеченные вершины из множества $O$, но так и не нашли дуги, содержащей право $\overleftarrow{t}$. В этом случае нам придется пометить текущую вершину отрицательно, выбрать в качестве текущей предыдущую помеченную положительно ($C(e) = C(e-1)$) и вернуться в состояние $\overrightarrow{T}^+$.

Если распознаватель попал в состояние $\overleftarrow{T}^+$, то нам следует проверить, существует ли дуга $\overleftarrow{t}\,(C(e), F(e))$. Если это так, то это значит, что обнаружен мост $\overrightarrow{t}^*\overrightarrow{g}\overleftarrow{t}^*$; пометим данное ребро положительно и перейдем в состояние $E^+$. В противном случае выберем непомеченную вершину $W(e) \in O$ и проверим, существует ли дуга $\overleftarrow{t}\,(C(e), W(e))$. Если такая дуга существует, то пометим вершину $W(e)$ и дугу положительно, присвоим $C(e) = W(e)$ и вновь перейдем в состояние $\overleftarrow{T}^+$. В случае, когда, перебрав все непомеченные вершины из $O$, мы не нашли нужной дуги (попали в тупик), необходимо попробовать вернуться к предыдущей помеченной положительно вершине. Для этого переведем распознаватель в новое состояние, которое назовем $\overleftarrow{T}^-$.

Опишем состояние $\overleftarrow{T}^-$. Распознаватель переходит сюда из состояния $\overleftarrow{T}^+$, если, перебрав все непомеченные вершины из $O$, он не смог обнаружить дуги $\overleftarrow{t}$ между $C(e)$ и $W(e)$. Пометим текущую вершину отрицательно, вернемся к предыдущей помеченной положительно вершине ($C(e) = C(e-1)$) и вновь перейдем в состояние $\overleftarrow{T}^+$. Если при выборе предыдущей вершины мы попадем в вершину, которой закончилась дуга $\overrightarrow{g}$, то нам придется пометить ее отрицательно и вернуться в состояние $\overrightarrow{T}^+$.

Графическая схема работы распознавателя представлена на рисунке 1.

Для поиска моста $\overrightarrow{t}^*\overleftarrow{g}\overleftarrow{t}$ можно воспользоваться описанным распознавателем, если заменить состояние $\overrightarrow{G}^+$ на состояние $\overleftarrow{G}^+$.

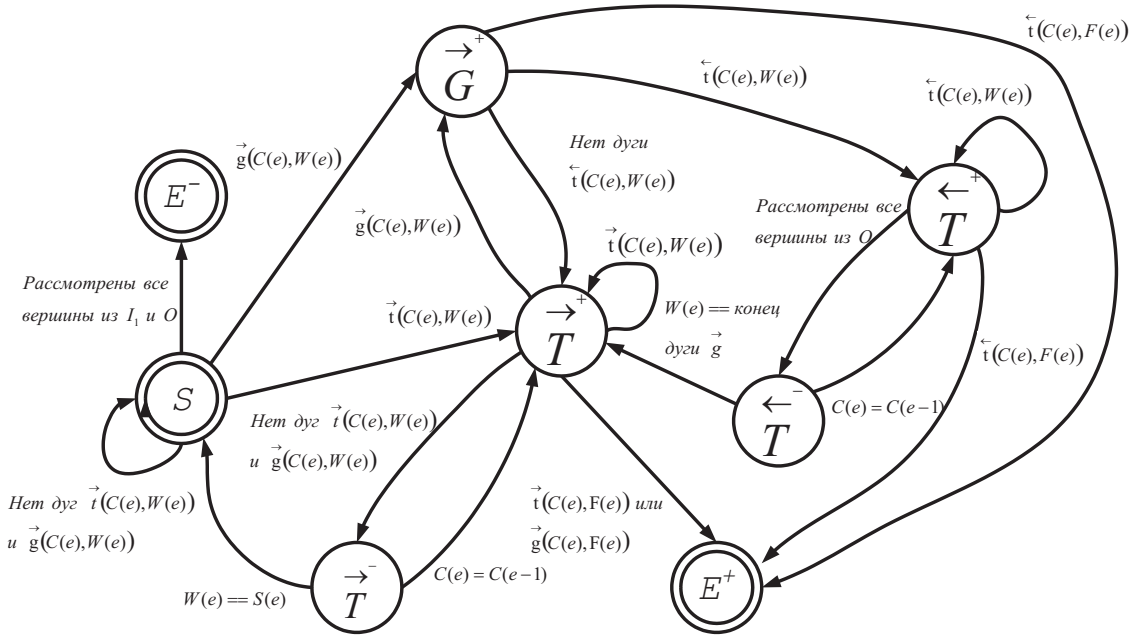


Рис. 1. Распознаватель моста $\vec{t}^{\,*}\ \vec{g}\ \overleftarrow{t}^{\,*}$

Построим распознаватель моста $\overleftarrow{t}^{\,*}$. Будем искать мост между островами $I_1$ и $I_2$. В начальном состоянии выберем $S(e) \in I_1$, $F(e) \in I_2$, присвоим $C(e) = S(e)$. Очевидно, что в данном случае не существует дуги $\overleftarrow{t}\,(C(e), F(e))$, иначе вершины $S(e)$ и $F(e)$ принадлежали бы одному острову. Выберем вершину $W(e)$ из множества $O$ и проверим, существует ли дуга $\overleftarrow{t}\,(C(e), W(e))$. Если дуга существует, то пометим данную дугу и вершину $W(e)$ положительно, присвоим $C(e) = W(e)$ и переведем распознаватель в состояние $\overleftarrow{T}^{+}$; иначе выберем следующую вершину $W(e)$. Если мы рассмотрели все вершины из $O$, но при этом так и не нашли ни одной дуги $\overleftarrow{t}\,(C(e), W(e))$, то попробуем выбрать другую вершину $S(e) \in I_1$. Наконец, когда мы перебрали все вершины из $I_1$ и не нашли ни одной дуги $\overleftarrow{t}\,(C(e), W(e))$ — это значит, что моста не существует; переведем распознаватель в состояние $E^{-}$.

Пусть распознаватель попал в состояние $\overleftarrow{T}^{+}$. Проверим, существует ли дуга $\overleftarrow{t}\,(C(e), F(e))$. Если да, то это значит, что обнаружен мост $\overleftarrow{t}^{\,*}$, пометим дугу положительно и переведем распознаватель в состояние $E^{+}$. Если такой дуги нет, то выберем вершину $W(e) \in O$ и проверим, существует ли дуга $\overleftarrow{t}\,(C(e), W(e))$. Если такая дуга есть, то пометим ее и вершину $W(e)$ положительно, присвоим $C(e) = W(e)$ и вновь переведем распознаватель в состояние $\overleftarrow{T}^{+}$; иначе выберем следующую вершину $W(e)$. Когда мы рассмотрим все вершины из $O$, но при этом не обнаружим ни одной дуги $\overleftarrow{t}\,(C(e), W(e))$, то попробуем вернуться к предыдущей помеченной положительно вершине, для этого переведем распознаватель в состояние $\overleftarrow{T}^{-}$.

В состоянии $\overleftarrow{T}^{-}$ необходимо пометить текущую вершину отрицательно и вернуться к предыдущей положительной вершине ($C(e) = C(e-1)$), после чего

нужно перевести распознаватель обратно в $\overleftarrow{T}^+$. Если при переходе к предыдущей вершине мы оказались в вершине $S(e)$, то необходимо вернуться в начальное состояние для выбора другой $S(e) \in I_1$.

Графическая схема работы распознавателя для поиска моста $\overleftarrow{t}^*$ представлена на рисунке 2.

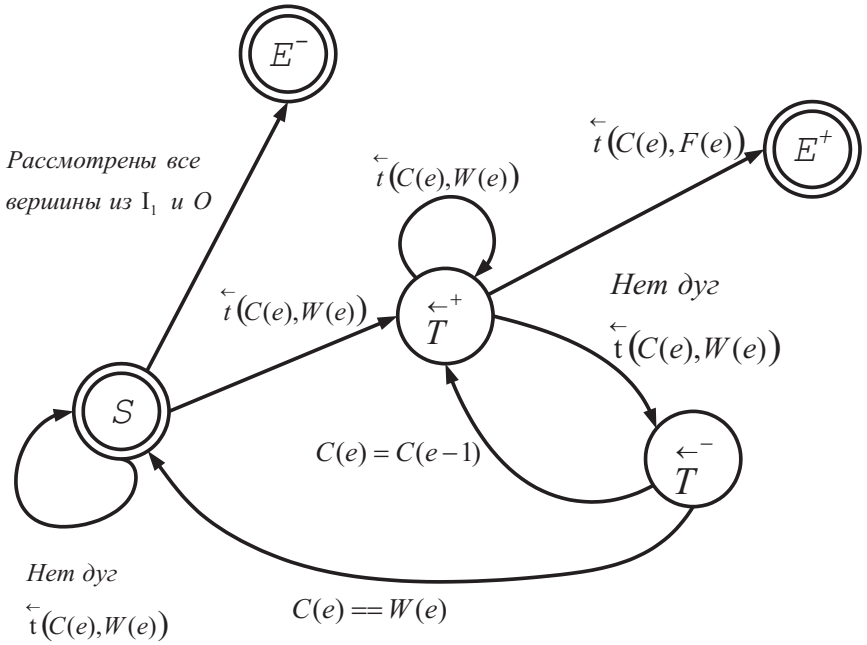


Рис. 2. Распознаватель моста $\overleftarrow{t}^*$

### 4.3. Поиск начального и конечного пролета моста

Нами остались не рассмотрены алгоритмы поиска начального и конечного пролета моста.

Рассмотрим поиск начального пролета. По определению 5 начальный пролет моста — это $tg$-путь, проходящий через вершины-объекты, началом которого является вершина-субъект, а концом — вершина-объект; при этом словарная запись пути имеет вид $\overrightarrow{t}^*\overrightarrow{g}$. Построим распознаватель начального пролета моста.

В первую очередь нам необходимо определить вершину-объект, которая будет являться концом пролета. Обозначим эту вершину $F(e)$. Будем искать начальный пролет моста от острова $I$ до вершины $F(e)$. В начальном состоянии нам необходимо выбрать вершину $S(e) \in I$. Также нужно выбрать рассматриваемую вершину, в начальном состоянии это будет вершина $S(e)$ $(C(e) = S(e))$. После этого проверим, существует ли дуга $\overrightarrow{g}(C(e), F(e))$. Если такая дуга существует — это значит, что начальный пролет моста обнаружен, пометим дугу положительно и переведем распознаватель в состояние $E^+$. Если такой дуги нет, то выберем вершину $W(e) \in O$ и проверим, существует ли дуга $\overrightarrow{t}(C(e), W(e))$.

Если дуга есть, то пометим ее и вершину $W(e)$ положительно, выполним присвоение $C(e) = W(e)$ и перейдем в состояние $\overrightarrow{T}^+$. Если дуги нет, то выберем следующую вершину $W(e)$. Если при переборе всех вершин из $O$ дуги $\overrightarrow{t}(C(e), W(e))$ не обнаружится, то выберем другую вершину $S(e)$. Когда будут рассмотрены все вершины из $I$, но при этом распознаватель будет находиться в начальном состоянии — мы выясним, что начального пролета между островом $I$ и вершиной $F(e)$ не существует; переведем распознаватель в состояние $E^-$.

Если распознаватель находится в состоянии $\overrightarrow{T}^+$, то в первую очередь нужно проверить, существует ли дуга $\overrightarrow{g}(C(e), F(e))$. Если существует, то, очевидно, мы нашли начальный пролет моста; пометим дугу положительно и перейдем в состояние $E^+$. Если такой дуги нет, то выберем следующую непомеченную вершину $W(e)$ из $O$, проверим, существует ли дуга $\overrightarrow{t}(C(e), W(e))$. Так же, как и в предыдущих рассмотренных случаях, если такая дуга есть, то помечаем ее и рассматриваемую вершину положительно, выполняем присвоение $C(e) = W(e)$ и вновь переходим в состояние $\overrightarrow{T}^+$. Если же дуги нет, то переходим в состояние $\overrightarrow{T}^-$.

Состояние $\overrightarrow{T}^-$ также аналогично ранее рассмотренным. Здесь мы помечаем текущую вершину отрицательно, переходим к предыдущей положительной вершине ($C(e) = C(e-1)$) и возвращаемся в состояние $\overrightarrow{T}^+$. Если $C(e-1) == S(e)$, то переходим в начальное состояние.

Граф работы распознавателя представлен на рисунке 3.

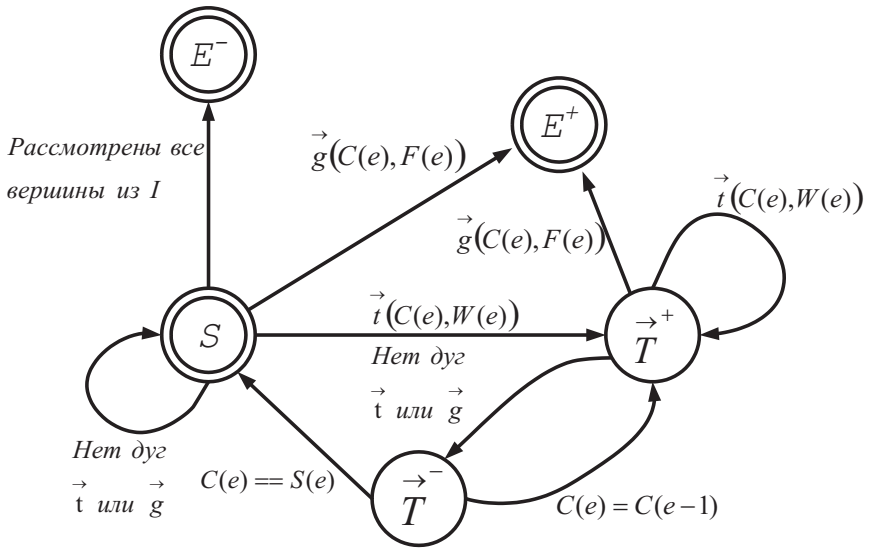


Рис. 3. Распознаватель начального пролета моста

Наконец, рассмотрим алгоритм для поиска конечного пролета моста. По определению 6, конечный пролет моста — это $tg$-путь, проходящий через вершины-объекты, началом которого является вершина-субъект, а концом — вершина-объект; при этом словарная запись пути имеет вид $\overrightarrow{t}^*$.

Как и в предыдущих случаях, построим распознаватель для данного типа $tg$-пути. Для начала выберем конечную вершину моста из $F(e) \in O$ и началь-

ную вершину $S(e) \in I$. В начальном состоянии присвоим $C(e) = S(e)$ проверим наличие дуги $\overrightarrow{t}(C(e), F(e))$. Если дуга есть, то пометим ее положительно и перейдем в состояние $E^+$ (конечный пролет найден), если нет — выберем вершину $W(e) \in O$. Проверим наличие дуги $\overrightarrow{t}(C(e), W(e))$, если такая дуга есть — перейдем в состояние $\overrightarrow{T}^+$, в противном случае выберем следующую $W(e)$. Вновь стоит учесть ситуацию, когда при переборе всех вершин из $O$ мы не нашли нужной дуги, в этом случае нужно выбрать следующую вершину $S(e) \in I$. Не забудем также, что в случае перебора всех субъектов из $I$ и всех объектов из $O$ мы можем не найти ни одной дуги $\overrightarrow{t}(C(e), W(e))$ и тогда, следует перевести распознаватель в состояние $E^-$.

Состояние $\overrightarrow{T}^+$ аналогично ранее рассмотренным. Здесь проверяется наличие дуги $\overrightarrow{t}(C(e), F(e))$, и в положительном случае распознаватель переводится в состояние $E^+$. В отрицательном случае проверяется наличие дуги $\overrightarrow{t}(C(e), W(e))$, и если дуга есть, то она и $W(e)$ помечаются положительно, рассматриваемая вершина делается текущей ($C(e) = W(e)$), и распознаватель возвращается в $\overrightarrow{T}^+$. Если перебрали все вершины из $O$, а дуги $\overrightarrow{t}(C(e), W(e))$ не нашли, то переходим в состояние $\overrightarrow{T}^-$.

В состоянии $\overrightarrow{T}^-$ переходим к предыдущей помеченной положительно вершине и возвращаемся в $\overrightarrow{T}^+$. Также не следует забывать про возможность перехода в начальное состояние в случае, когда $C(e-1) == S(e)$.

Граф работы распознавателя представлен на рисунке 4.

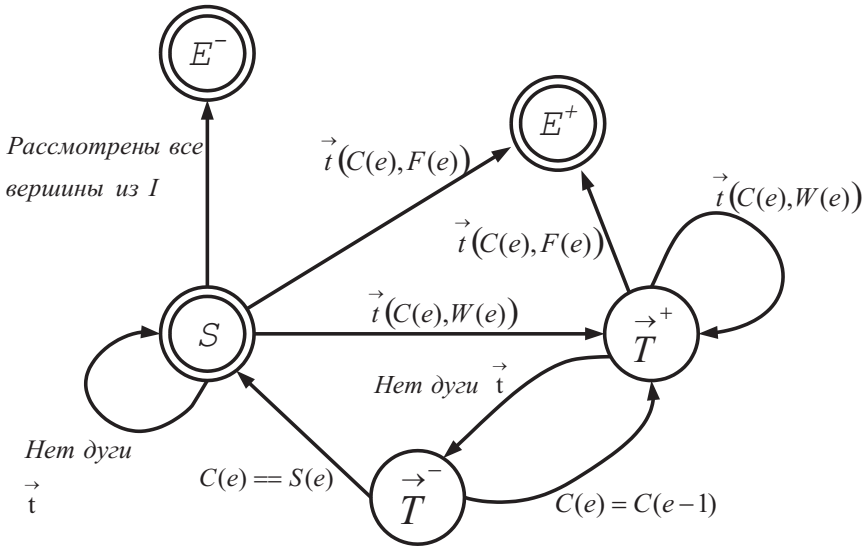


Рис. 4. Распознаватель конечного пролета моста

Мы рассмотрели алгоритмы для поиска $tg$-путей, островов и мостов в графе доступов. Теперь мы можем построить формальный автомат, который смог бы выяснить, является ли предикат *«возможен доступ»*$(\alpha, x, y, G_0)$ истинным для данного состояния системы. То есть этот автомат может показать, является ли данное состояние компьютерной системы безопасным.

## Литература

1. Девянин, П.Н. Модели безопасности компьютерных систем: Учебное пособие для студентов высших учебных заведений / П.Н. Девянин. – М.: Издательский центр «Академия», 2005.
2. Гайдамакин, Н.А. Разграничение доступа к информации в компьютерных системах / Н.А. Гайдамакин. – Е.: Издательство Уральского университета, 2003.
3. Майника, Э. Алгоритмы оптимизации на сетях и графах / Э. Майника – М.: Издательство «Мир», 1981.
4. Гордеев, А.В. Системное программное обеспечение. Учебник / А.В. Гордеев, А.Ю. Молчанов – СПб.: «Питер», 2001.