Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

М. В. Леонов, А. Г. Никитин

ЭФФЕКТИВНЫЙ АЛГОРИТМ, РЕАЛИЗУЮЩИЙ ЗАМКНУТЫЙ НАБОР БУЛЕВЫХ ОПЕРАЦИЙ НАД МНОЖЕСТВАМИ МНОГОУГОЛЬНИКОВ НА ПЛОСКОСТИ

Препринт
46

Новосибирск 1997

В данной работе описывается простой и эффективный алгоритм, реализующий замкнутый набор булевых операций {*объединение*, *пересечение*, *разность*, *симметрическая разность*} над множествами гранично заданных многоугольников (*boundary represented polygons*) на плоскости. Замкнутость набора операций подразумевает корректную обработку многоугольников с самокасаниями (или кратными вершинами), т.е. результирующие многоугольники, полученные с помощью предлагаемого алгоритма, удовлетворяют предъявляемым требованиям к входным данным алгоритма. Асимптотическая оценка времени выполнения алгоритма не превышает $O(n \log^* n + k + z \log n)$, где n — общее число вершин многоугольников-операндов, k — число точек пересечения их рёбер, z — общее число контуров многоугольников-операндов. Таким образом, предложенный алгоритм по эффективности не уступает большинству алгоритмов, известных на сегодняшний день.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

Michael V. Leonov, Alexey G. Nikitin

AN EFFICIENT ALGORITHM FOR A CLOSED SET OF BOOLEAN OPERATIONS ON POLYGONAL REGIONS IN THE PLANE

Preprint 46

Novosibirsk 1997

This paper presents a simple and fast algorithm for computing union, intersection, difference and symmetrical difference of boundary represented polygonal regions in the plane. The algorithm explicitly copes with degenerate cases and vertices of high degree so that the output of the algorithm satisfies its input restrictions. An expected running time of the algorithm is $O(n\ log^*n + k + z\ log\ n)$, where n (respectively z) is the total number of edges (respectively contours) in polygons describing input regions and k is the number of intersections between the edges. The presented algorithm outperforms most of known algorithms for polygon set operations and is relatively easy to implement.

ВВЕДЕНИЕ

Каждый простой двумерный многоугольник ограничивает некоторую область на плоскости. Под выполнением булевой операции *op* ∈ {*объединение*, *пересечение*, *разность*, *симметрическая разность*} над такими многоугольниками понимается выполнение операции *op* над описываемыми ими областями и получение множества многоугольников, описывающего полученную область. Основная проблема заключается в том, что результатом некоторой операции даже над двумя простыми выпуклыми многоугольниками в общем случае может являться множество невыпуклых многоугольников с возможными отверстиями и самокасаниями (например, $A - B$ на рис. 1).

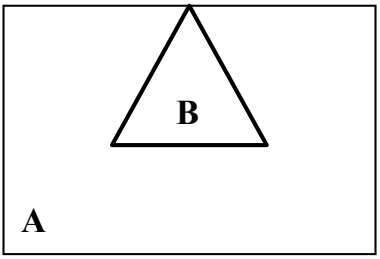


Рис. 1

К сожалению, традиционные алгоритмы ограничиваются тем, что разбивают сложные многоугольники результата на простые, тем самым исключая возможность повторного выполнения операций над результатом. Для задач отсечения по экрану и выяснения критических участков интегральных схем этого достаточно, но в системах автоматизированного проектирования и управления географическими базами данных требуется реализация *замкнутого* набора операций над множествами многоугольников. Это означает, что результат, полученный с помощью алгоритма, должен удовлетворять его же предусловию. Также желательно, чтобы алгоритм допускал отверстия и неограниченную кратность вершин. Проведём краткий обзор существующих алгоритмов.

Вейлер (Weiler) и Азертон (Atherton) [5, 12, 13] предложили следующий алгоритм. Ребра исходных многоугольников дублируются, и между ними вычисляются пересечения. Затем производится перераспределение ссылок в вершинах пересекающихся ребер с целью выделения минимальных контуров. Для этого используется довольно сложный набор правил для различных случаев пересечения рёбер. На основе полученных контуров создаются результирующие многоугольники для операций объединения, пересечения и вычитания. Перечислим основные недостатки этого подхода:

— результат булевой операции не всегда состоит из минимальных контуров (см. 0, результат операции вычисления симметрической разности $A \neq B$);

— не освещена проблема описания многоугольников с самокасаниями, которые могут быть получены в результате работы алгоритма;

— результат операции может не удовлетворять требованиям на входные данные, причем расширение алгоритма для обработки кратных пересечений крайне затруднительно.

Шутте (Schutte) [9, 10] модифицировал алгоритм Вейлера более четким разделением этапов и новым алгоритмом маркировки ребер (edge labeling), однако его алгоритм обладает более существенными ограничениями на входные данные.

Итак, приведенные выше алгоритмы реализуют набор операций, не являющийся замкнутым, так как в результате их выполнения могут получаться контуры с самокасаниями, что неприемлемо в качестве исходных данных для обоих алгоритмов, и с отверстиями, которые не допускаются алгоритмом Шутте.

Сравнительно недавно Гютинг (Gueting) и Шнайдер (Schneider) [6 - 8] разработали ROSE-алгебру (RObust Spatial Extension algebra), реализующую замкнутый набор операций над двумерными объектами, в том числе регионами (regions). Предложенное ими описание также имеет ряд недостатков:

— координаты вершин ROSE-объектов принадлежат дискретному множеству, в отличие от общепринятой в вычислительной геометрии модели вещественнозначной РАМ [13];

— сложный и громоздкий в реализации алгоритм связывания сегментов (half-segments) региона в корректные внешние и внутренние циклы (R-cycles), приводящий к снижению эффективности процедур обработки регионов.

В данной работе описывается простой и эффективный алгоритм, который свободен от перечисленных выше недостатков и реализует замкнутый набор операций {*объединение*, *пересечение*, *разность*, *симметрическая разность*} над множествами многоугольников на плоскости. В разд. 0 даются основные определения, в разд. 0 описывается собственно алгоритм, в последующих разделах анализируется его эффективность и обсуждаются некоторые детали реализации.

1. ОПРЕДЕЛЕНИЯ И ОБОЗНАЧЕНИЯ

Далее *ребром* называется направленный отрезок с началом в точке a и концом в точке b, не совпадающей с a, котрый обозначается как $E(a, b)$.

При этом будем говорить, что для точки b ребро E является *входящим*, а для точки a — *выходящим*. Множество точек x, удовлетворяющих условиям $0 < \angle(bax) < \varphi$ и $2\pi - \varphi < \angle(abx) < 2\pi$, назовем *левой* φ-*окрестностью ребра* E[1]. Множество точек x, удовлетворяющих условиям $0 < \angle(abx) < \varphi$ и $2\pi - \varphi < \angle(bax) < 2\pi$, назовем *правой* φ-*окрестностью ребра* E (см. рис. 2а).

Определение 1. *Контуром* называется упорядоченное множество n ребер $C = \{E_0, E_1, \dots, E_{n-1}\}$, $n \geq 3$ такое, что $\forall\, i \in \{0 \dots n-1\}$ $E_{i-1} = E(v_{i-1}, v_i)$ и $E_i = E(v_i, v_{i+1})$[2].

Точку v_i, для которой ребра E_{i-1} и E_i являются соответственно входящим и выходящим, будем называть i-й вершиной контура C, ребра E_{i-1} и E_i — соседними или *смежными*. Порядок обхода ребер контура $C = \{E_0, E_1, \dots, E_{n-1}\}$ с увеличением индекса ($i \to i + 1$) будем называть *прямым* направлением обхода, а противоположный ему — *обратным*.

Заметим, что одной точке плоскости может соответствовать несколько вершин контура, такие вершины будем называть *кратными*. *Углом входящего* в вершину v *ребра* $E(a,v)$ будем называть направленный угол между векторами e_1 (орт абсцисс) и $E(v,a)$. *Углом выходящего* из вершины v *ребра* $E(v,b)$, будем называть направленный угол между векторами e_1 и $E(v,b)$.

Рассмотрим пару соседних ребер E_{i-1} и E_i контура C: $0 \leq i \leq n - 1$. Множество точек x, удовлетворяющих условиям $0 < \angle(v_{i+1}\, v_i\, x) < \angle(v_{i+1}\, v_i\, v_{i-1})$ и $0 < |x - v_i| < \varepsilon$, назовем *внутренней* ε-*окрестностью вершины* v_i и будем обозначать $\varepsilon^+(v_i)$. Множество точек x, удовлетворяющих условиям $0 < \angle(v_{i-1}\, v_i\, x) < \angle(v_{i-1}\, v_i\, v_{i+1})$ и $0 < |x - v_i| < \varepsilon$ назовем *внешней* ε-*окрестностью вершины* v_i и будем обозначать $\varepsilon^-(v_i)$ (см. рис. 2б).

Рассмотрим евклидово пространство размерности 2, наделенное декартовыми координатами с ориентирующим репером e_1, e_2, т. е. плоскость с традиционным "правым" репером.

Определение 2. *Областью* называется ограниченное полигональное открытое связное множество на евклидовой плоскости, определенное с точностью до множества меры нуль.

---

[1] $\angle(ABC)$ обозначает направленный угол между векторами BA и BC, лежащий в интервале $[0,2\pi)$.

[2] Здесь и далее при вычислении индексов вершин и ребер используется сложение и вычитание по модулю n, где n — число ребер рассматриваемого в данный момент контура.

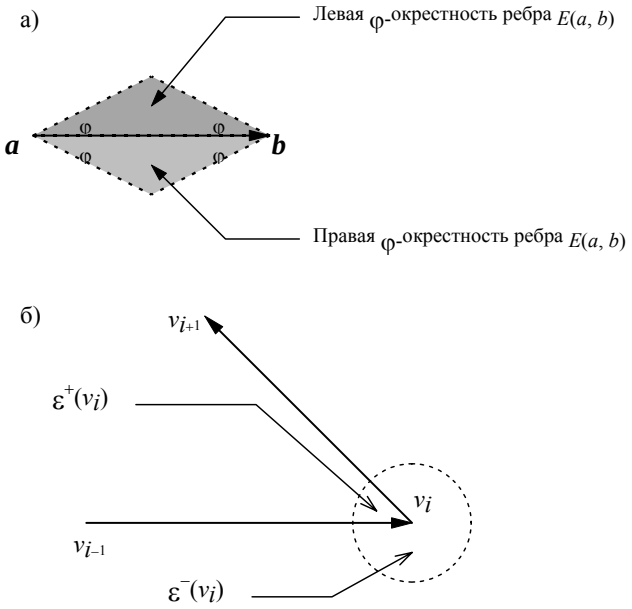


Рис. 2

Под полигональностью понимается то, что граница области состоит из замкнутых ломаных. Определение области с точностью до множества меры нуль позволяет рассматривать области без “разрезов” и “дыр” в точке. Заметим, что область может быть как одно-, так и многосвязной.

Рассмотрим границу dA некоторой области A. Если ориентация dA согласована с ориентацией A, мы приходим к соответствию между границей области и набором контуров на плоскости. Для того, чтобы оно было взаимно-однозначным, добавим к традиционному определению два ограничения. Итак, рассмотрим контур C, содержащий n ребер.

Определение 3. Контур C называется *ограничивающим контуром* области A, если выполняются следующие условия:

1) $C \subset dA$;
2) обход C в прямом направлении является положительным обходом, то есть обходом, при котором область A остается слева;
3) $\forall\, i \in \{0 \ldots n-1\}\ \exists \quad \varepsilon > 0\colon \forall\ x \in \varepsilon^{+}(v_i) \quad x \in A;$

   $\forall\, i \in \{0 \ldots n-1\}\ \forall \quad \varepsilon > 0\colon \exists\ \ x \in \varepsilon^{-}(v_i) \quad x \notin A.$

Определение 4 *Полигоном* называется совокупность всех ограничивающих контуров некоторой области.

Внешним контуром области *A* назовем ограничивающий ее контур, положительный обход которого производится против часовой стрелки. *Внутренним* контуром области *A* назовем ограничивающий ее контур, положительный обход которого производится по часовой стрелке. Очевидно, что в произвольном полигоне может быть один и только один внешний контур.

Определение 5. *Регионом* называется набор полигонов, описывающий множество непересекающихся областей.

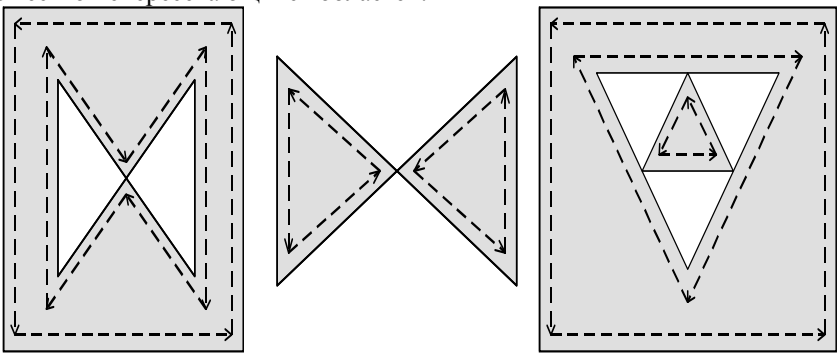


Рис. 3

Таким образом, установлено взаимно-однозначное соответствие между множеством непересекающихся областей и множеством ограничивающих их контуров. На рис. 3 можно увидеть удовлетворяющие данным выше определениям описания областей (закрашенные) регионами (пунктирными линиями показаны направление обхода и связность ограничивающих контуров).

2. ОПИСАНИЕ АЛГОРИТМА

Входные данные алгоритма: 2 региона *A* и *B*, операция *op* = {*объединение*, *пересечение*, *разность*, *симметрическая разность*}.

Выходные данные алгоритма: регион *R* = *A op B*.

Алгоритм состоит из четырех этапов.

1. Обработка пересечений ребер.
2. Маркировка контуров и ребер.
3. Получение результирующих контуров.

4. Создание результирующего региона.

Рассмотрим в качестве примера регионы A и B, изображенные на рис. 4. С учетом наших определений регион A состоит из внешнего и внутреннего контуров, а регион B — из внешнего контура, касающегося самого себя в точке.

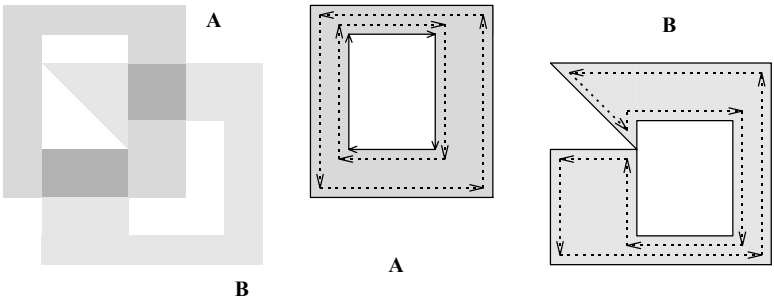


Рис. 4

2.1. Обработка пересечений ребер

Найдем все пары не смежных пересекающихся ребер регионов A и B. Если пересечение ребер не точка, а отрезок, будем считать концы этого отрезка двумя точками пересечения. Если точки пересечения не совпадают с концевыми точками ребер, в исходные регионы добавляются новые вершины с координатами соответствующих точек. Отметим, что добавление точек пересечения не изменяет области, описываемые регионами A и B. Далее везде под регионами A и B имеются в виду исходные регионы A и B со всеми добавленными точками пересечения. Вершины, соответствующие точкам пересечения ребер, назовем *вершинами-пересечениями* (отмечены жирными точками на рис. 5). Отметим, что одной геометрической точке пересечения ребер соответствуют как минимум две вершины-пересечения.

Рассмотрим вершину-пересечение v одного из регионов. Если за $E^+(v)$ обозначим входящее в v ребро, а за $E^-(v)$ — выходящее, тогда обработка вершины-пересечения выполняется следующим образом.

1. Создаются два *дескриптора вершин-пересечений* $D^+(v)$ и $D^-(v)$, отвечающие соответственно $E^+(v)$ и $E^-(v)$. В дескрипторы помещается информация об углах их ребер, а также о том, являются ли эти ребра входящими или выходящими относительно v.

2. Между v и дескрипторами $D^+(v)$ и $D^-(v)$ устанавливаются двусторонние связи, которые позволяют определить соответствующую некоторой вершине-пересечению пару дескрипторов и наоборот.

3. $D^+(v)$ и $D^-(v)$ помещаются в *список связности* $L(x)$ *точки пересечения* x. Данный список является кольцевым списком дескрипторов вершин-пересечений, соответствующих точке x, отсортированным по углам ребер. В дальнейшем список $L(x)$ будет использоваться для поиска ребер, ближайших к данному в направлениях по часовой и против часовой стрелки. Порядок в списке связности дескрипторов с одинаковыми углами ребер не специфицирован.

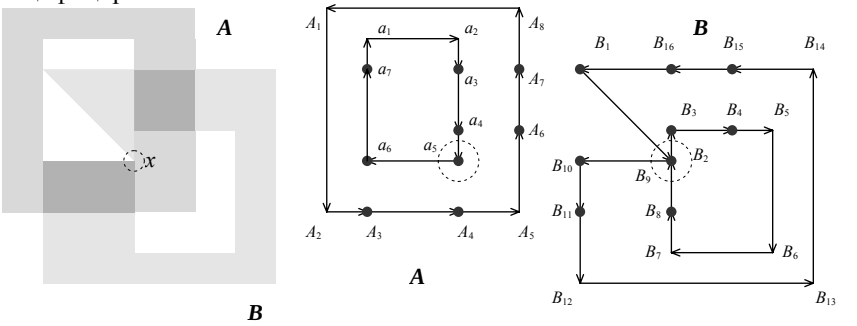


Рис. 5

На рис. 6 приведен пример списка связности точки x, выделенной на рис. 5. Ей соответствуют три вершины-пересечения — a_5, B_2 и B_9.

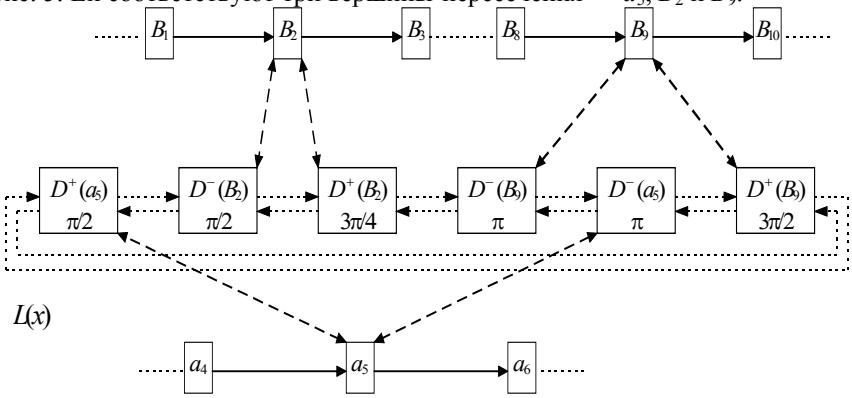


Рис. 6

Далее будем говорить, что вершина-пересечение v присутствует в некотором списке связности, если в нем находятся ее дескрипторы $D^+(v)$ и $D^-(v)$.

Так как *v* может присутствовать только в списке связности геометрически соответствующей ей точки, можно употреблять понятие *списка связности вершины-пересечения*.

2.2. Маркировка контуров и ребер

Рассмотрим некоторый ограничивающий контур *C* одного из регионов *A* и *B*. Обозначим за *M* регион, к которому не принадлежит *C*.

На предыдущем этапе все точки пересечения были добавлены в исходные регионы. Отсюда, с учетом определений 3, 4 и 5, с очевидностью вытекает следующее утверждение.

Утверждение 1. После выполнения первого этапа алгоритма:

1) ребра регионов *A* и *B* не имеют никаких общих точек друг с другом, кроме концевых;

2) произвольное ребро контура *C* может либо геометрически совпадать с одним из ребер *M* (такие ребра будем называть *сопряженными*), либо находиться целиком (за исключением, может быть, концевых точек) внутри или вне области, описываемой *M*;

3) если контур *C* не содержит вершин-пересечений, то он лежит целиком внутри или вне области, описываемой *M*.

Меткой ребра E, принадлежащего контуру *C*, будем называть его атрибут, имеющий значение из множества {*SHARED1, SHARED2, INSIDE, OUTSIDE*} и отражающий его геометрическое положение относительно региона *M* следующим образом: *INSIDE* — ребро *E* (может быть, за исключением его концевых точек) лежит внутри *M*, *OUTSIDE* — ребро *E* лежит вне *M*, *SHARED1* — ребро *E* и сопряженное ему ребро из *M* сонаправлены, *SHARED2* — ребро *E* и сопряженное ему ребро из *M* направлены в противоположные стороны.

Меткой контура C мы будем называть его атрибут, имеющий значение из множества {*ISECTED, INSIDE, OUTSIDE*} и отражающий его геометрическое положение относительно *M* следующим образом: *ISECTED* — содержит вершины-пересечения, *INSIDE* — *C* лежит внутри *M*, *OUTSIDE* — *C* лежит вне *M*.

Корректность определений меток контура и ребра следует из утверждения 1.

Приведем алгоритм маркировки контура *C* и его ребер.

1. *C не содержит вершин-пересечений.*

Если *C* лежит внутри *M*, то *C* помечается как *INSIDE,* в противном случае — как *OUTSIDE*. Маркировки ребер *C* не производится.

2. *C содержит вершины-пересечения.*

C помечается как *ISECTED* и производится последовательная маркировка всех его ребер E_i, $i \in \{0 \ldots n-1\}$, где n — число ребер в *C*. Пусть $E_i(a, b)$ — маркируемое ребро.

2.1 *E_i не содержит вершин-пересечений.*

1) $i \neq 0$. Значение метки копируется у ребра E_{i-1}.
2) $i = 0$. Если *a* лежит внутри *M*, то E_i помечается как *INSIDE*, в противном случае E_i помечается как *OUTSIDE*.

2.2 *E_i содержит вершины-пересечения.*

1) $\exists$ *ребро* $F(c, d)$: $F \in M \wedge a = c \wedge b = d$. E_i помечается как *SHARED1*.
2) $\exists$ *ребро* $F(c, d)$: $F \in M \wedge a = d \wedge b = c$. E_i помечается как *SHARED2*.
3) *В списке связности вершин-пересечений не присутствует вершин M.*

   Такое возможно, если контур *C* не пересекается с *M* и касается самого себя. Метка ставится так же, как и в случае 2.1.
4) *В списке связности вершин-пересечений присутствуют вершины M.*

   Для каждой из таких вершин w_k определяется, лежит ли E_i внутри $\angle(w_{k-1}w_kw_{k+1})$. Если E_i попало во внутренность хотя бы одного из таких "маркирующих" углов, то оно лежит внутри *M* и помечается как *INSIDE*. В противном случае E_i помечается как *OUTSIDE*(см. рис. 7).

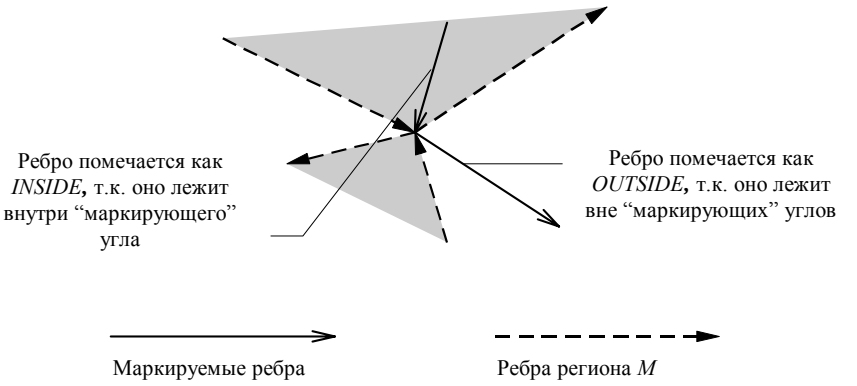


Рис. 7

2.3. Получение результирующих контуров

2.3.1. Предварительные рассуждения

Одна из ключевых идей данного алгоритма заключается в том, что ограничивающие контуры региона R (далее *результирующие контуры*) собираются на основе *меток* контуров и ребер, а не координат их вершин. Так как одно и то же ребро не может входить в регион более одного раза, нам нужно найти правила, по которым ребра и контуры включаются в результат.

На рис. 8 приведен результат маркировки регионов A и B нашего примера.

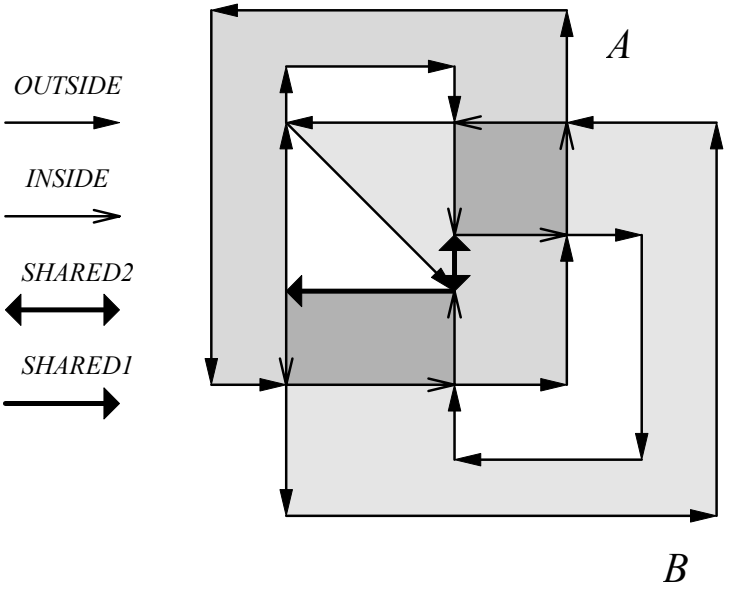


Рис. 8

2.3.1.1. Условия включения ребер.

Рассмотрим некоторое ребро E, принадлежащее A или B. Выберем наибольшее $\varphi > 0$ такое, что ни левая, ни правая φ-окрестности ребра E не будут содержать никаких вершин и ребер регионов A и B. Назовем эти окрестности левой и правой *минимальными окрестностями ребра E* и обозначим их соответственно $\varphi^{+}(E)$ и $\varphi^{-}(E)$. По выбору φ все точки мини-

мальной окрестности ребра одновременно принадлежат или не принадлежат области, описываемой регионом A или B.

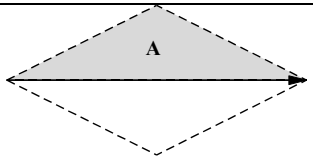
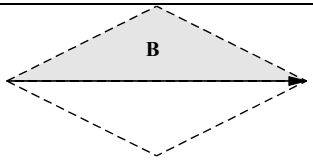
Принадлежность некоторого ребра результату можно установить по принадлежности результирующей области минимальных окрестностей ребра следующим образом:

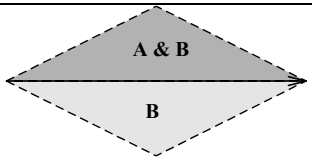
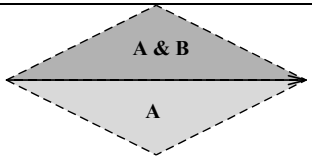
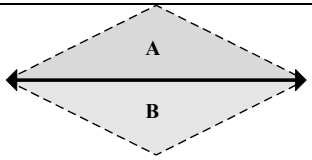
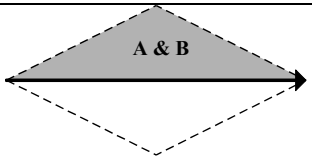
1. Если $\varphi^+(E)$ входит в результат, а $\varphi^-(E)$ — нет, E войдет в результат с исходной ориентацией.
2. Если $\varphi^-(E)$ входит в результат, а $\varphi^+(E)$ — нет, E войдет в результат с обратной ориентацией.
3. Если обе минимальные окрестности E не входят в результат, E не войдет в результат.
4. Если обе минимальные окрестности E входят в результат, E не войдет в результат (иначе возникает противоречие пункту 4 определения 3).

Заметим, что из пары сопряженных ребер в результат можно включить только одно с нужной ориентацией.

В табл. 1 рассмотрены условия включения для всех возможных случаев меток ребер обоих регионов.

Таблица 1

	**Случай 1.** E принадлежит региону A и имеет метку *OUTSIDE;* $\varphi^+(E)$ принадлежит региону A, $\varphi^-(E)$ не принадлежит ни одному из регионов; E войдет в результат операций *объединение*, *разность* и *симметрическая разность* с исходной ориентацией.
	**Случай 2.** E принадлежит региону B и имеет метку *OUTSIDE*; $\varphi^+(E)$ принадлежит региону B, $\varphi^-(E)$ не принадлежит ни одному из регионов; E войдет в результат операций *объединение* и *симметрическая разность* с исходной ориентацией.

	**Случай 3.** *E* принадлежит региону *A* и имеет метку *INSIDE*; $\varphi^+(E)$ принадлежит обоим регионам, $\varphi^-(E)$ принадлежит региону *B*; *E* войдет в результат операции *пересечение* с исходной ориентацией, а в результат операции *симметрическая разность* — с обратной.
	**Случай 4.** *E* принадлежит региону *B* и имеет метку *INSIDE*; $\varphi^+(E)$ принадлежит обоим регионам, $\varphi^-(E)$ принадлежит региону *A*; *E* войдет в результат операции *пересечение* с исходной ориентацией, а в результат операций *разность* и *симметрическая разность* — с обратной.
	**Случай 5.** Пара двух геометрически совпадающих и направленных в разные стороны ребер $E \in A$ и $F \in B$ с метками *SHARED2*; $\varphi^+(E)$ принадлежит только региону *A*, а $\varphi^+(F)$ принадлежит только региону *B*. Одно из ребер войдет в результат операции *разность* с направлением, совпадающим с направлением ребра *E*.
	**Случай 6.** Пара двух геометрически совпадающих и направленных в одну сторону ребер регионов *A* и *B* с метками *SHARED1*. Левая минимальная окрестность ребер принадлежит обоим регионам, а правая — ни одному из них. Одно из ребер войдет в результат операций *объединение* и *пересечение* с исходным направлением.

2.3.1.2. Процедура перехода в вершине-пересечении

Если в процессе сборки ребер результата мы пришли в вершину-пересечение *v*, возникает проблема выбора дальнейшего направления обхо-

да. Пусть x — точка плоскости, соответствующая v. Рассмотрим все вершины-пересечения, геометрически совпадающие с x. Пусть их число равно m, $m \geq \mathbf{2}$. Тогда число их дескрипторов равно $2m$ (см. 0). С помощью $L(x)$ построим множество ребер F_j, $j \in \{\mathbf{0}\ldots 2m-\mathbf{1}\}$ такое, что $F_0 = E^+(v)$ и дескриптор, соответствующий ребру F_{j+1} является ближайшим в направлении по часовой стрелки к дескриптору, соответствующему ребру F_j.

Выберем наибольшее $\varepsilon > 0$ такое, что ни одна из внутренних или внешних ε-окрестностей рассматриваемых вершин не будут содержать никаких вершин и ребер, отличных от F_j. Далее рассмотрим сектора, на которые геометрическая ε-окрестность точки x разбивается ребрами F_j. Назовем такие сектора *минимальными*. Очевидно, что все точки минимального сектора (за исключением, может быть, его границ) одновременно принадлежат или не принадлежат области, описываемой регионом A или B. Если точки соседних минимальных секторов должны войти в результат, то по определению 3 их должен ограничивать один и тот же контур результирующего региона.

Таким образом, нужно, начиная с F_0 и перебирая F_j в направлении возрастания j, найти ребро, удовлетворяющее условию включения для выполняемой операции. Оно и будет искомым ребром для продолжения обхода. Отметим, что для поиска не требуется информации о координатах вершин, так как дескрипторы отсортированы в списке связности по углам соответствующих ребер.

2.3.2. Алгоритм сборки результирующих контуров

Так как предыдущие рассуждения справедливы для всех операций, сборка результата осуществляется для всех операций единообразно. На данном этапе последовательно рассматривается каждый контур из регионов A и B. Обозначим рассматриваемый контур за C.

Метку контура или ребра X будем обозначать *X.Flags*. Далее обход ребер контура в прямом и обратном направлениях (см. определение 1) обозначим как *FORWARD* и *BACKWARD* соответственно. Приведем алгоритм сборки результирующих контуров.

1. Если $C.Flags \neq ISECTED$, то контур включается в результат на основе своей метки с нужным направлением. В табл. 2 приведены *условия включения для контура C* в зависимости от выполняемой операции *op*.

Таблица 2

op	Условие включения *C* в результат	Направление
$\cap$	(*C.Flags* = *INSIDE*)	*FORWARD*
$\cup$	(*C.Flags* = *OUTSIDE*)	*FORWARD*
–	(*C.Flags* = *OUTSIDE*) $\wedge$ (*C* $\in$ *A*) (*C.Flags* = *INSIDE*) $\wedge$ (*C* $\in$ *B*)	*FORWARD* *BACKWARD*
$\neq$	(*C.Flags* = *OUTSIDE*) (*C.Flags* = *INSIDE*)	*FORWARD* *BACKWARD*

2. Если *C.Flags* = *ISECTED*, нужно совершить обход всех ребер *C*, чтобы найти ребра, с которых начнется сборка результирующих контуров. Если *n* — число вершин контура *C*, то алгоритм поиска подходящих ребер выглядит следующим образом:

```
for i := 0 to n–1 do
begin
    if (EdgeRule(Ei, dir) and (Ei не помечено как пройденное))
        if (dir = FORWARD)
            r := Collect(vi, dir);
        else
            r := Collect(vi+1, dir);
    Включить r в набор результирующих контуров;
end;
```

где **function EdgeRule(E:ребро; var dir:(FORWARD, BACKWARD)): boolean** — функция-параметр, соответствующая условию включения ребра в результат выполняемой операции (см. табл. 3).

Таблица 3

op	Условие включения *E* в результат	Направление
$\cap$	(*E.Flags* = *INSIDE*) $\vee$□(*E.Flags* = *SHARED*)	*FORWARD*
$\cup$	(*E.Flags* = *OUTSIDE*) $\vee$□(*E.Flags* = *SHARED1*)	*FORWARD*
–	((*E.Flags* = *OUTSIDE*) $\vee$ (*E.Flags* = *SHARED2*)) $\wedge$ (*E* $\in$ *A*) ((*E.Flags* = *INSIDE*) $\vee$ (*E.Flags* = *SHARED2*)) $\wedge$ (*E* $\in$ *B*)	*FORWARD* *BACKWARD*
$\oplus$	(*E.Flags* = *OUTSIDE*) (*E.Flags* = *INSIDE*)	*FORWARD* *BACKWARD*

Процедура *сборки* результирующего контура *r*, начиная с вершины *v*, выглядит следующим образом:

```
function Collect(v:вершина; dir:(FORWARD, BACKWARD)):контур;
begin
      Создать пустой контур r;
    repeat
          Включить v в r;
          if (dir = FORWARD)
                E := выходящее из v ребро;
          else
                E := входящее в v ребро;
          Пометить E как пройденное;
          if ((E.Flags = SHARED1) or□(E.Flags = SHARED2))
                пометить сопряженное E ребро как пройденное;
          v := вершина, следующая относительно v в направлении dir;
          if (v является вершиной-пересечением) then
                Jump(v, dir);
      until (текущее ребро помечено как пройденное);
      return r;
end;

procedure Jump(var v:вершина; var dir:(FORWARD, BACKWARD));
begin
      if (dir = FORWARD) then
            d := prev(D+(v));
      else
            d := prev(D–(v));
      {За prev(D) мы обозначили дескриптор, ближайший по часовой стрел-
      ке к D}
      found := FALSE;
    repeat
          e := ребро, соответствующее d;
            if ((e не помечено как пройденное) and EdgeRule(e, newdir)) then
          begin
                if ((e входящее) and (newdir = BACKWARD) or
                  (e выходящее) and (newdir = FORWARD)) then
                  begin
                        dir := newdir;
                        v := вершина-пересечение, соответствующая d;
```

```
                found := TRUE;
            end;
        end;
          d := prev(d);
    until (found);
end;
```

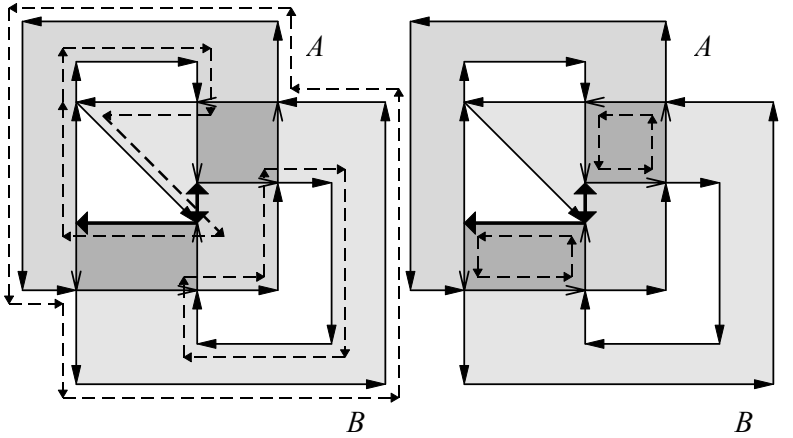
На рис. 9 приведены результирующие контуры различных операций над регионами *A* и *B* примера.

Результат операции *объединение* состоит из одного внешнего и двух внутренних контуров, *разность* и *пересечение* — из двух внешних контуров, *симметрическая разность* — из трех внешних контуров.

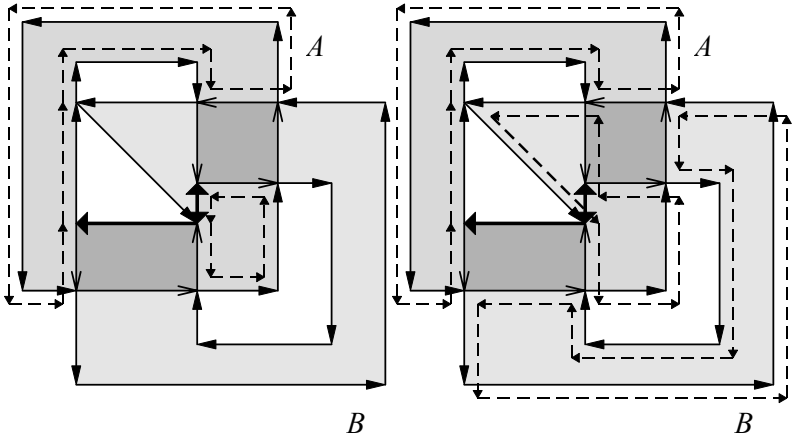
2.4. Создание результирующего региона

Итак, мы получили набор ограничивающих контуров *R*. В отличие от многих других алгоритмов, задача правильного формирования *R* очень проста, так как полученные на предыдущем этапе внешние и внутренние контуры имеют правильную ориентацию.

Для каждого внешнего контура создается отдельный полигон, внешним контуром которого он и будет являться. Внутренние контуры на предыдущем этапе сохраняются во временном списке, а затем помещаются в полигоны результата. Найти нужный полигон несложно, для этого достаточно определить минимальный из внешних контуров, содержащих данный внутренний контур. Регион *R* будет состоять из множества полученных таким образом полигонов.



$A \cup B$ $A \cap B$



$A - B$ $A \oplus B$

Рис. 9

3. АНАЛИЗ ЭФФЕКТИВНОСТИ АЛГОРИТМА

Проанализируем эффективность описанного выше алгоритма. Пусть исходные регионы A и B содержат всего n вершин и z контуров.

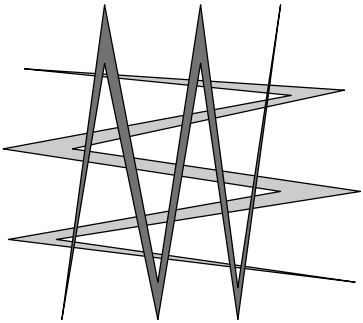


Рис. 10

На этапе 0 максимальное число новых вершин k составляет $O(n^2)$ (см. рис. 10). Время выполнения данного этапа определяется используемым алгоритмом нахождения пересечений. Вычислительная сложность наиболее простого алгоритма попарного перебора ребер составляет $O(n^2 + k)$. Существует ряд более эффективных алгоритмов [1 - 3], время выполнения которых составляет $O(n \log n + k)$. В работе Финке (Finke) и Хинрикса (Hinrichs) [4] описывается алгоритм, который можно использовать для нахождения пересечения двух многосвязных подразбиений плоскости (path connected planar subdivisions) за время $O(n \log^* n + k + z \log n)$ (пусть $n > 0$, $\log^{(0)} n = n$, $\log^{(i)} n = \log(\log^{(i-1)} n)$, тогда $\log^* n$ — это наибольшее целое число s, такое что $\log^{(s)} n \geq 1$).

Этап 0 выполняется за время $O(n + z\, p(n))$, где $p(n)$ — время, необходимое для выяснения, находится ли некоторая точка внутри региона, состоящего из n вершин. Без использования дополнительных структур данных $p(n) = O(n)$. Для ускорения выполнения таких запросов можно использовать структуру, полученную при выполнении трапезоидации (trapezoidation) регионов, которую можно построить за $O(n \log^* n + z \log n)$ [11]. С помощью этого приема можно достичь оптимального времени $p(n) = O(\log n)$.

Так как на этапе 0 каждое ребро проходится не более двух раз, он выполняется за время $O(n + k)$. Время выполнения этапа 0 не превышает $O(z\, p(n + k))$.

За счет использования минимальных описывающих прямоугольников, которые вычисляются за время $O(n)$, можно достичь дополнительного уменьшения вычислительных затрат на всех этапах алгоритма.

Таким образом, время выполнения алгоритма не превышает $O(n \log^* n + k + z \log n)$, и затраты на память, очевидно, составляют $O(n + k)$. Эти результаты на сегодняшний день являются оптимальными как с теоретической, так и с практической точки зрения.

ЗАКЛЮЧЕНИЕ

Перечислим основные результаты данной работы.

- Даны четкие определения, позволяющие однозначно описать полигональные области на плоскости с отверстиями и самокасаниями.
- Разработан метод обработки кратных вершин-пересечений, позволивший решить проблему вырожденности (degeneracy) точек, характерную для традиционных алгоритмов.
- Введен набор ясных и интуитивно понятных (обращаем внимание на симметрию условий включения в результат контуров и ребер) правил для сборки результирующих контуров, допускающих эффективную и компактную реализацию алгоритма.

Главным достижением данной работы является *замкнутость* полученного набора операций.

Авторская реализация алгоритма занимает около 850 строк на языке C, причем в случае больших полигональных моделей реальное время выполнения несколько меньше теоретически предсказанного ввиду использования различных оптимизирующих техник (см. разд. 0).

В заключение хочется поблагодарить Ральфа Гютинга, Раймунда Зейделя и Джека Сноуинка за предоставленные материалы, и Кламера Шутте, с готовностью обсуждавшего с авторами плюсы и минусы своего алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. **Further** comparison of algorithms for geometric intersection problems / D. S. Andrews, J. Snoeyink, J. Boritz a. o. // Proc. 6th Internat. Sympos. on Spatial Data Handling. — 1994. — P. 709—724.
2. **Balaban I. J.** An optimal algorithm for finding segments intersections // Proc. 11th Annual ACM Sympos. on Computational geometry. — 1995. — P. 211—219.
3. **Chan T.** iA simple trapezoid sweep algorithm for reporting red/blue segment ntersections // Proc. 6th Canadian Conf. on Computational Geometry. — 1994. — P. 263—268.
4. **Finke U. , Hinrichs K. H.** Overlaying simply connected planar subdivisions in linear time // Proc. 11th Annual ACM Sympos. on Computational geometry. — 1995. — P. 119—126.
5. **Computer** graphics: principles and practice / J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes. — Reading: Addison-Wesley, 1990.— Section 15.7.2.

6. **Gueting R. H., Schneider M.** Realms: a foundation for spatial data types in database systems // Proc. 3rd Internat. Sympos. on Large Spatial Databases. — 1993. — P. 14—35.
7. **Gueting R. H. , Schneider M.** Realm-based spatial data types: the ROSE algebra // VLDB Journal 4. — 1995. — P. 100—143.
8. **Gueting R. H. , Schneider M., de Ridder Th.** Implementation of the ROSE algebra: efficient algorithms for realm-based spatial data types // Proc. 4th Internat. Sympos. on Large Spatial Databases. — 1995. — P. 216—239.
9. **Schutte K.** An edge labeling approach to concave polygon clipping // публикация в Internet [http://www.ph.tn.tudelft.nl/~klamer/clip.ps.gz].
10. **Schutte K.** Knowledge based recognition of man-made objects: Ph.D. Thesis. — University of Twente, ISBN 90-9006902-X, 1994. — Appendix A.
11. **Seidel R.** A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons // Computational Geometry: Theory and Applications. — 1991. — Vol. 1, № 1. — P. 51—64.
12. **Weiler K., Atherton P.** Hidden surface removal using polygon area sorting // Computer Graphics. — 1977. — Vol. 11, № 2. — P. 214—222.
13. **Weiler K.** Polygon comparison using a graph representation // Computer Graphics. — 1980. — P. 10—18.
14. Вычислительная геометрия: введение / Ф. Препарата, М. Шеймос. — М.: Мир, 1989.

М. В. Леонов, А. Г. Никитин

ЭФФЕКТИВНЫЙ АЛГОРИТМ, РЕАЛИЗУЮЩИЙ ЗАМКНУТЫЙ НАБОР БУЛЕВЫХ ОПЕРАЦИЙ НАД МНОЖЕСТВАМИ МНОГОУГОЛЬНИКОВ НА ПЛОСКОСТИ

Препринт
46

Рукопись поступила в редакцию 27.11.97
Рецензент Н. В. Шилов
Редактор Л. А. Карева

Подписано в печать 10.12.97
Формат бумаги 60 × 84 1/16 Объем 1.5 уч.-изд.л., 1.6 п.л.
Тираж 75 экз.

Отпечатано на ризографе “AL Group”
630090, г. Новосибирск: пр. Акад. Лаврентьева, 3