5. Half-plane intersection

Introduction. In this chapter, we will show you how to describe and find the intersection of n half-planes in the plane effectively. We want to proceed recurrently according to the "divide and conquer"strategy. That is, we divide the set

$$H = \{h_1, h_2, \dots, h_n\}$$

of given half-planes into two parts H_1 and H_2 of roughly same size and we compute the intersection

$$C = \bigcap_{h_i \in H} h_i$$

as an intersection $C_1 \cap C_2$, where

$$C_1 = \bigcap_{h_i \in H_1} h_i, \qquad C_2 = \bigcap_{h_i \in H_2} h_i.$$

A major role for the gradual computing of intersections $C_1 \cap C_2$ has the way how we describe these intersections.

Description of half-plane intersection. Intersections of half-planes are convex sets that can be both bounded and unbounded. We exclude the case that among given half-planes there are two opposite to each other. In this case, the intersection would be a subset of their common line, and this would lead to a one-dimensional task of the intersection of half-lines on the line.

If the intersection is bounded and 2-dimensional, it is a convex polygon and we can describe it using a double-connected edge list with two faces, one bounded and one unbounded. In case the intersection is unbounded, we need to modify this description. For this purpose, we will use a lexicographic arrangement of points in the plane defined as follows:

$$p > q \quad \text{iff} \quad p_y > q_y \ \text{ or } \ (p_y = q_y \ \text{ and } \ p_x < q_x).$$

Let us consider a non-empty convex set C, which is the intersection of the half-planes and is not a subset of a line. There is just one of the following options:

(1) The set C has a maximum point p and a minimum point q in the lexicographic arrangement. They both lie on the boundary and divide it into the left and right parts, abbreviated to the left and right boundaries. Each of these is a polygonal chain determined by the sequence of vertices and segments. We denote these sequences $L(C)$ and $P(C)$, respectively.

FIGURE 5.1 The left boundary of C is $L(C) = (p = v_1, e_1, v_2, e_2, v_3, e_3, q = v_4)$, the right boundary is $P(C) = (p = v_1, e_7, v_7, e_6, v_6, e_5, v_5, e_4, q = v_4)$.

(2) The set C contains the maximum point p but does not contain the minimum point. In this case the point p divides the boundary of C again into two parts, left and right. Each of them is determined by a sequence of vertices and segments terminated by a half-line.

FIGURE 5.2 The left boundary $L(C) = (p = v_1, e_1, v_2, e_2, v_3, e_3)$, the right boundary $P(C) = (p = v_1, e_7, v_7, e_6, v_6, e_5, v_5, e_4)$.

(3) C contains a minimum point q but does not have a maximum point. In this case q divides the boundary again into two parts, left and right. These are sequences, starting with a half-line followed by vertices and segments.

FIGURE 5.3 $L(C) = (e_1, v_2, e_2, v_3, e_3, q = v_4)$, $P(C) = (e_7, v_7, e_6, v_6, e_5, v_5, e_4, q = v_4)$.

(4) The set C has neither a maximum point nor a minimum point. In this case, there are two possibilities. Either the boundary has left and right parts formed by parallel lines or the boundary has only one part, left or right. This is determined by a single line or a sequence beginning and ending with a half-line with a sequence of vertices and segments between them. We will speak about the other part of the boundary as empty.

FIGURE 5.4 Left: $L(C) = (e_1)$, $L(C) = (e_2)$. Right: $L(C) = (e_1, v_2, e_2, v_3, e_3)$, $P(C) = (\,)$.

Algorithm for half-plane intersection. Let C_1 and C_2 be convex sets that have been created as intersections of disjoint sets of half-planes. Therefore they are determined by their left and right boundaries. We will describe an algorithm which will create two sequences describing the left and right boundary of the intersection $C_1 \cap C_2$ out of the sequences $L(C_1)$, $P(C_1)$, $L(C_2)$, $P(C_2)$.

To do this, it is sufficient to realize that the vertices on the left boundary of $C_1 \cap C_2$ are the following:

- vertices of $L(C_1)$, which lie inside C_2,
- vertices of $L(C_2)$, which lie inside C_1,
- points of intersection of the left boundaries of C_1 and C_2,
- points of intersection of the left boundary of C_i and the right boundary of C_j for $i \neq j$. They form in the given lexicographic order a maximal or a minimal vertex in $L(C_1 \cap C_2)$.

FIGURE 5.5 The vertices on the left boundary of the intersections $C_1 \cap C_2$ are p – the point of the intersection of the left boundary of C_2 and the right boundary of C_1, $w_2 \in L(C_2)$ lying inside C_1, q – the point of the intersection of the left boundaries, $v_3 \in L(C_1)$ lying inside C_2 and r – the point of the intersection of the left boundary of C_1 and the right boundary of C_2.

Analogously, the points of the right boundary of $C_1 \cap C_2$ are:

- vertices of $P(C_1)$ lying inside C_2,
- vertices of $L(C_2)$ lying inside C_1,
- points of intersection of the right boundaries of C_1 and C_2,
- points of intersection of the left boundary of C_i and the right boundary of C_j for $i \neq j$ They are maximal or minimal vertices of $P(C_1 \cap C_2)$.

To create lists $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$ we have to select the appropriate vertices of C_1 and C_2, to calculate the intersections of the boundaries and to organize chosen vertices lexicographically. Our algorithm will be similar to the algorithm for segment intersection in Chapter 1. So we use again sweep line method.

Events will be the vertices of C_1 and C_2 and the calculated intersections of boundaries. At the beginning of the algorithm, we set the boundary vertices lexicographically into the queue. Because the vertices are already arranged at both left and right boundaries, the creation of the queue takes time proportional to the number of vertices. Gradually, we will include the calculated intersections of boundaries into the event queue. Since we know on which segments or half-lines these intersections lie, their queuing will take only a constant time.

If none of the convex sets C_1 and C_2 have a maximum point, we calculate the intersections of the half-lines that begin the left and right boundaries of both sets and we will place the obtained points into the queue. Since C_1 and C_2 are intersections of disjoint sets of half-planes, the set of the boundary intersections will be finite.

Horizontal sweep line l moves from top to bottom. It starts at a position above the highest event. For every event we will record on which boundary it lies, and which lines, half-lines or segments pass through it. (For each of the three types of objects we will use a common notation *edge*.) We will also record the order in which the edges of each boundary cross the sweep line. Since there are no more than four edges crossing the sweep line, there is no need to keep the order in a binary balanced tree. When passing the event v, we do the following actions:

(1) We include or do not include the event v as a vertex in the sequence $L(C_1 \cap C_2)$ or $P(C_1 \cap C_2)$ using criteria given above.

(2) If v is the first event in the left or right boundary of the intersection, and if there are some half-lines going upwards from it, we decide which one will be in the right and the left boundary of the intersection.

FIGURE 5.6 The vertex v is the first event in $L(C_1 \cap C_2)$. The half-line f_1 will be a part of $L(C_1 \cap C_2)$ above v, and hence $L(C_1 \cap C_2) = (f_1, v, \dots)$.

(3) We decide which of the edges going downwards from the event v is in the left and which is in the right boundary of the intersection.

FIGURE 5.7 The edges e and f are going down the event $v \in L(C_1 \cap C_2)$. The edge e will be included in $L(C_1 \cap C_2)$ below the event v, while f is not any more a part of the boundary below v.

(4) We compute the intersection of the left edge e_l originating from the event v downwards with the adjacent left edge s_l of the boundary of the second set, and the intersection of the right edge e_p originating from the event v downwards with the adjacent right edge s_p of the boundary of the second set. If intersections exist and lie below v, we put them in the queue.

FIGURE 5.8 The choice of edges e_l, e_p, s_l, s_p for the sweep line below the event v. We put the point $p = e_l \cap s_l$ in the queue. The intersection $e_p \cap s_p$ is empty.

(5) If the event v is the minimum vertex of the left and right boundary of the intersection, we will remove all events from the queue as the description of both boundaries of the intersection is already complete. Otherwise, only the event v is removed from the queue.

The algorithm ends when the queue is empty.

Pseudocodes and their running time. A framework for the whole algorithm is provided by the following pseudocode:

Algoritmus 1: HalfplanesIntersection(H)

```
Input: A set $H = \{h_1, h_2, \dots, h_n\}$ of $n$ half-planes in the plane.
Output: Lists $L(C)$ and $P(C)$ describing the left and the right boundaries of
        the intersection $C$ of all half-planes from $H$.
if $n = 1$ then
    determine the left and the right boundaries.
else
    put $H_1 = \{h_1, h_2, \dots, h_{[n/2]}\}$, $H_2 = H \setminus H_1$.
    $C_1 \leftarrow$ HalfplanesIntersection($H_1$).
    $C_2 \leftarrow$ HalfplanesIntersection($H_2$).
    $C \leftarrow$ IntersectionOfTwo($C_1, C_2$).
end
```

The intersection of two convex sets can be obtained in this way:

Algoritmus 2: IntersectionOfTwo(C_1, C_2)

```
Input: Convex sets $C_1$ a $C_2$ described using the left and right boundaries.
Output: The intersection $C = C_1 \cap C_2$ described using a list $L(C)$ for the left
        boundary and a list $P(C)$ for the right boundary.
Put $L(C) = (\ )$ a $P(C) = (\ )$.
if $C_1$ a $C_2$ are half-planes then
    compute $L(C)$ a $P(C)$.
else
    make an event queue out of vertices of $C_1$ and $C_2$.
end
if $C_1$ a $C_2$ are not bounded from above then
    compute intersections of boundary half-planes or lines which are at the
      beginnings of boundaries of both sets, and insert them into the queue.
end
while the queue of events is not empty do
    take its first vertex $v$
    HandleEvent($C_1, C_2, v$)
end
return $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$.
```

The passage of the sweep line through an event is recorded in the pseudocode:

Algoritmus 3: HandleEvent(C_1, C_2, v)

Input: A vertex v on the boundary of C_1 or C_2 and lists $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$ for the left and right boundaries of the intersection above v.

Output: Updated lists $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$, updated queue Q.

```
1: if $v \in L(C_i)$ lies between the left and right boundaries of $C_j$, $i \neq j$ then
2:     insert $v$ into $L(C_1 \cap C_2)$.
3: end
4: if $v \in P(C_i)$ lies between the left and right boundaries of $C_j$, $i \neq j$ then
5:     insert $v$ into $P(C_1 \cap C_2)$.
6: end
7: if $v$ lies in the intersection of the left boundaries then
8:     insert $v$ into $L(C_1 \cap C_2)$.
9: end
10: if $v$ lies in the intersection of the right boundaries then
11:     insert $v$ into $P(C_1 \cap C_2)$.
12: end
13: if $v$ lies in the intersection of the left boundary of $C_i$ and the right boundary of
      $C_j$, $i \neq j$ then
14:     insert $v$ into $L(C_1 \cap C_2)$ and also into $P(C_1 \cap C_2)$.
15: end
16: if $v$ lies in $L(C_1 \cap C_2)$ or in $P(C_1 \cap C_2)$ then
17:     if $v$ is the first vertex in $L(C_1 \cap C_2)$ or in $P(C_1 \cap C_2)$ then
18:         find which halflines with lower vertex $v$ belong to $L(C_1 \cap C_2)$ or
              $P(C_1 \cap C_2)$.
19:     end
20:     Find which edges with the upper vertex $v$ belong to $L(C_1 \cap C_2)$ or
        $P(C_1 \cap C_2)$.
21: end
22: From the edges which go downwards from the vertex $v$ denote $e_l$ the mostleft
    one and $e_p$ the mostright one. To $e_l$ find a left adjacent edge $s_l$ from the
    boundary of the other convex set. To $e_p$ find a right adjacent edge $s_p$ from the
    boundary of the other set.
23: Compute intersections $s_l \cap e_l$ and $s_p \cap e_p$ below $v$ and insert them into the
    queue.
24: if $v$ is the last member of the sequence $L(C_1 \cap C_2)$ or/and $P(C_1 \cap C_2)$ (i.e. it is
      not followed by an edge) then
25:     remove all events from the queue
26: else
27:     remove $v$ from the queue
28: end
29: return $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$.
```

The running time of the algorithm from the last pseudocode is constant. Let the convex sets C_1 and C_2 have n_1 and n_2 vertices, respectively. Then the running time of the algorithm from the pseudocode $\textsc{IntersectionOfTwo}(C_1, C_2)$ is $O(n_1 + n_2)$.

The running time $T(n)$ of the whole algorithm for the computation of the intersection of n half-planes is given by the recurrent relation

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n).$$

This leads to the resulting running time

$$T(n) = O(n \log n).$$

Animation. An example of the computation of the intersection of two convex sets C_1 and C_2 according to the algorithms described by the second and the third pseudocodes is captured in the following animation.

ANIMATION

(1) The algorithm creates the queue $Q = (w_2, w_3, w_5, v_2, v_4, v_3, w_4)$. There are no intersections of half-lines e_1, e_4, f_1, f_5 above w_2. The queue remains unchanged. The position of sweep line is above w_2.

CORRECTION of the pictures. The vertex in the middle denoted w_4 in this first picture of animation should be denoted w_5. See the second picture where the description is correct. Nevertheless, all the remaining pictures contain the same mistake. Please, correct.

(2) The sweep line passes $w_2 \in L(C_2)$. The vertex does not lie in C_1, hence $L(C_1 \cap C_2) = P(C_1 \cap C_2) = (\ )$, $e_p = e_l = f_2$, $s_l = e_4$, $s_l \cap e_l = \{p\}$. We insert p into the queue.

(3) The sweep line passes p. This point is in the intersection of the left boundary of C_2 and the right boundary of C_1. Hence $L(C_1 \cap C_2) = P(C_1 \cap C_2) = (p)$. Next we find edges which are beyond the point p in the left and right boundaries of the intersection. $L(C_1 \cap C_2) = (p, f_2)$, $P(C_1 \cap C_2) = (p, e_4)$. $e_l = f_2$, $e_p = e_4$, $s_l = e_1$, $s_p = f_5$. $e_p \cap s_p = \{q\}$, $e_l \cap s_l = \emptyset$. We insert q into the queue.

(4) The sweep line passes $w_3 \in L(C_2)$ which lies in C_1. Hence $L(C_1 \cap C_2) = (p, f_2, w_3, f_3)$. $e_l = e_p = f_3$, $s_l = e_1$, $s_p = e_4$, $s_l \cap e_l = e_p \cap s_p = \emptyset$.

(5) The sweep line passes q which is the intersection of the right boundary of C_1 and the right boundary of C_2. Hence $P(C_1 \cap C_2) = (p, e_4, q, f_5)$. $e_l = f_5$, $e_p = e_4$, $s_l = f_3$, $s_l \cap e_l = \emptyset$.

(6) The sweep line passes $w_5 \in P(C_2)$ which lies in C_1. Hence $P(C_1 \cap C_2) = (p, e_4, q, f_5, w_5, f_4)$. $e_l = e_p = f_4$, $s_l = e_1$, $s_p = e_4$, $s_l \cap e_l = s_p \cap e_p = \emptyset$.

(7) The sweep line passes $v_2 \in L(C_1)$ which does not lie in C_2. Hence $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$ remain unchanged. $e_l = e_p = e_2$, $s_p = f_3$, $e_p \cap s_p = \{r\}$. We insert r into the queue. $Q = (r, v_4, v_3, w_4)$.

(8) The sweep line passes r which is an intersection of the left boundaries. We update $L(C_1 \cap C_2) = (p, f_2, w_3, f_3, r, e_2)$. $e_l = f_3$, $e_p = e_2$, $s_p = f_4$, $e_p \cap s_p = \{t\}$. We insert t into the queue. $Q = (v_4, t, v_3, w_4)$.

(9) The sweep line passes $v_4 \in P(C_1)$ which does not lie in C_2. Hence $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$ remain unchanged. $e_l = e_p = e_3$, $s_l = f_4$, $e_l \cap s_l = \emptyset$. The queue is $Q = (t, v_3, w_4)$.

(10) The sweep line passes t which is an intersection of the left boundary of C_1 and the right boundary of C_2. Hence t is the last member both in $L(C_1 \cap C_2)$ and $P(C_1 \cap C_2)$. We get $L(C_1 \cap C_2) = (p, f_2, w_3, f_3, r, e_2, t)$ and $P(C_1 \cap C_2) = (p, e_4, q, f_5, w_5, f_4, t)$. We remove the remaining points from the queue. The algorithm is finished.