Lecture 6 - Half-plane intersection

Consider a set $H = \{h_1, \ldots, h_n\}$ of half-planes

$h_i : a_i x + b_i y \leq c_i$

Goal: compute intersection $C = \bigcap_{h_i \in H} h_i$

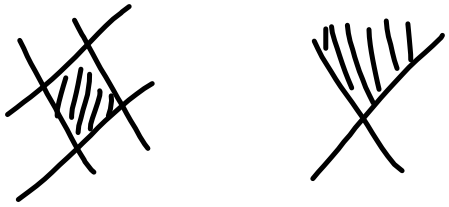
Approach: recursive (divide & conquer)

ie. divide H into two sets H_1, H_2 of roughly same size, compute $C_1 = \bigcap_{h_i \in H_1} h_i$, $C_2 = \bigcap_{h_i \in H_2} h_i$

& then calculate $C = C_1 \cap C_2$

certain convex sets

- Half-plane  is convex set.
- Intersection of convex sets is convex:

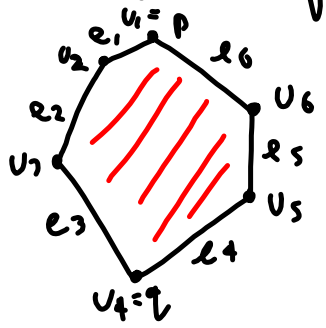
Eg. 

- What are the possible shapes of convex sets that can arise as the intersection of finitely many half-planes?
- How to represent them computationally?
- Consider lex. ordering

$$p > q \iff p_y > q_y \text{ or } (p_y = q_y \;\&\; p_x < p_x).$$

Non-empty convex subsets of plane which are ints of finitely many half-planes but not subsets of a line:

1) C has max p & min q, wrt lex ordering.



Boundary of P splits into left & right path L(C) & R(C) from top to bottom:

$L(C) = (v_1, e_1, v_2, e_2, v_3, \ell_3, v_4)$

$R(C) = (v_1, \ell_6, v_6, \ell_5, v_5, \ell_4, v_4)$

C is determined by LC & RC.

In each case, we will represent C by two sequences LC, RC of vertices, edges, half-edges or unbounded edges.

2) C has max, no min :

LC, RC are sequences beginning w' vertex & ending with a half-edge.

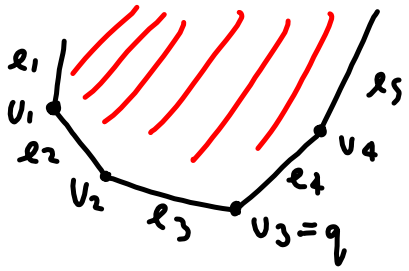
eg. $LC = (v_1, e_1, v_2, e_2, v_3, e_3)$

3) C has no max, has min

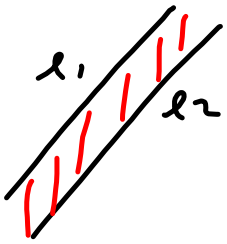
LC, RC begin with half-edges & end in vertices

$RC = (e_5, v_4, e_4, v_3)$

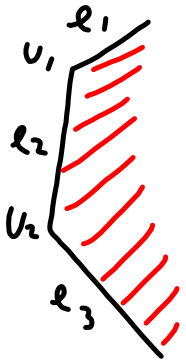
half-edge



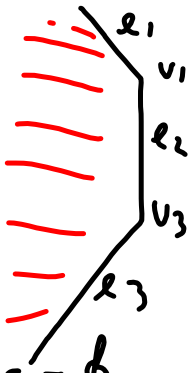
4) C has no min or max .



$LC = (\ell_1)$, $RC = (\ell_2)$ where e_1, e_2 unbounded edges .



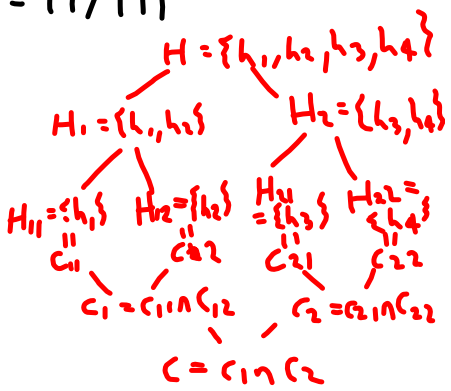
$LC = (\ell_1, v_1, \ell_2, v_2, \ell_3)$
$RC = \emptyset$



$LC = \emptyset$
$RC = (\ell_1, v_1, \ell_2, v_2, \ell_3)$

Algorithm : Half-plane Int (H)

- Input : $H = \{h_1, \ldots, h_n\}$ set of half-planes
- Output : Intersection C of H described using sequences LC, RC of vertices, edges, half-edges & unbounded edges, describing left & right paths of C.
- If $n=1$, determine LC, RC.
- Else, put $H_1 = \{h_1, \ldots, h_{\lfloor n/2 \rfloor}\}$ & $H_2 = H / H_1$.
- Set C_1 = Half-plane Int (H_1)
  C_2 = ... (H_2)
  C = Intersection of Two (C_1, C_2)



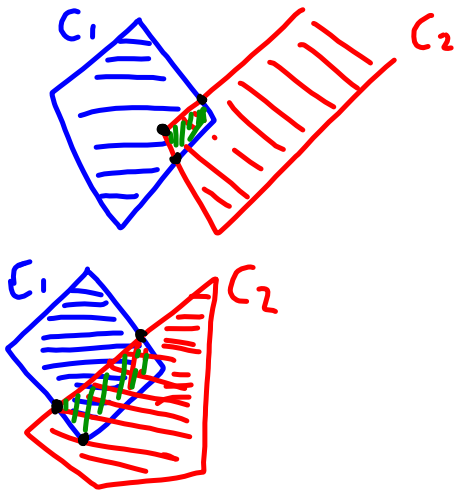
Algorithm : IntOfTwo(C_1, C_2)

- Input C_1, C_2 : int of sets of half-planes desc. using lists LC_1, RC_1, LC_2, RC_2.
- Output - $C_1 \cap C_2$ described using lists $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$.

Important : understand vertices of $C_1 \cap C_2$.

⊛ Vertices of $L(C_1 \cap C_2)$:

- vertices of $L(C_2)$ inside C_1
- vertices of $L(C_1)$ inside C_2
- points of intersection of $L(C_1)$ & $L(C_2)$.
- points of int of left path of one & right path of the other - these will be max, min points of intersection.
- Similarly $R(C_1 \cap C_2)$.



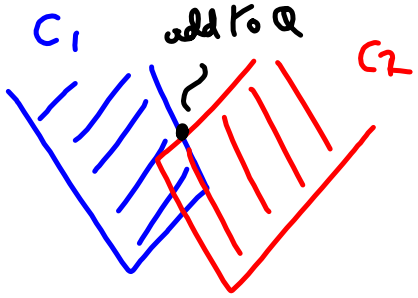
Use sweep-line method:

- Ordered sequence of edges T intersecting the sweep-line (no need for bin bal. tree - at most 4 edges intersecting sweep-line.)
- Queue Q of events - certain vertices of C_1 & C_2, plus points of intersection: only those that are uppermost vertices on left & right paths, endpoints of edges in T, or intersections of edges in T.

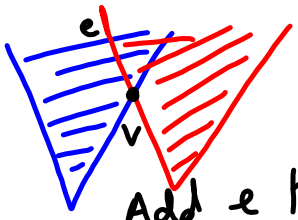
Certainly no more than 16!!! in Q

① Add largest vertices on $L(C_1)$, $R(C_1)$, $L(C_2)$ & $R(C_2)$ to Q, if they exist.

- If neither C_1, C_2 has max, compute the intersections of upper lines, half-edges & insert into Q.



(2) - At event point v,
decide whether $v \in L(C_1 \cap C_2)$ or $v \in R(C_1 \cap C_2)$ using (*), & if so, add v to appropriate path.

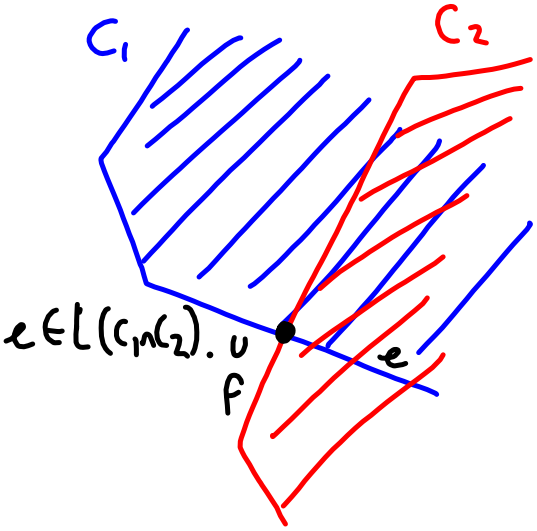


Add e to $L(C_1 \cap C_2)$ so $\{e, v\}$.

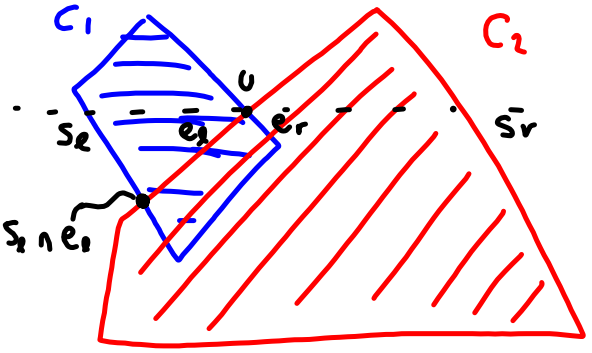
- If v is first element of $L(C_1 \cap C_2)$ or $R(C_1 \cap C_2)$, find edges w' v as lower endpoint & decide which belong to $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$. Add to path.

eg. if e appears before v in $L(C_1)$ it belongs to $L(C_1 \cap C_2)$ $\Leftrightarrow$ it belongs to C_2.

- Look at edges going downwards from v - decide which lies in $C_1 \cap C_2$ & on which path $L(C_1 \cap C_2)$ or $R(C_1 \cap C_2)$. Add to path.
- Add all lower endpoints of edges going downwards from v into Q.



- Let e_ℓ, e_r be leftmost & rightmost edges going down from v.



- Find left edge s_ℓ to e_ℓ from other set.
- Find right edge s_r to e_r from other set.
- Calculate $s_\ell \cap e_\ell$ & $s_r \cap e_r$ & add them to Q.
- If no edges come down from v (so that v is last element in $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$) then we have computed the intersection — therefore we empty queue (if necessary).
- Otherwise, delete v from Q.

IntersectionOfTwo(C_1, C_2)

n_1 vertices n_2 vertices

has complexity $O(n_1+n_2)$ – updates to Q, T take constant time.

$T(n)$ complexity of half-plane int. alg.

$T(n) = 2T(n/2) + \text{Time}(\text{IntOfTwo}(n/2, n/2))$

$= 2T(n/2) + O(n)$

$= O(n \log n)$.