Alumno:	Erik Rangel Limón	**Programación Declarativa**
No. Cuenta:	318159287	*2023-1*
Correo:	erikrangel.014@ciencias.unam.mx	PROPUESTA PROYECTO

Cierre convexo para un conjunto de discos

Este semestre estoy cursando la optativa de *"Geometría Computacional"* para la cual voy a realizar una exposición sobre un artículo que explica un algoritmo para calcular un cierre convexo.

El problema a resolver es el siguiente:

Se tiene un conjunto de S discos en el plano de tamaño arbitrario que posiblemente se intersectan; se desea calcular el cierre convexo de S, es decir, la región convexa más pequeña que contiene a todos los discos en S.

La solución a este problema se realiza mediante un algoritmo con una estrategia "divide y vencerás" que divide el conjunto de discos y calcula recursivamente su cierre convexo, para posteriormente unir el cierre convexo con un algoritmo *"merge"*.

Algorithm Hull

1. Split S arbitrarily into two disjoint subsets of discs, P and Q, such that $|P|$ and $|Q|$ differ by at most one.
2. Recursively find $CH(P)$ and $CH(Q)$.
3. Use algorithm merge to merge $CH(P)$ and $CH(Q)$ resulting in $CH(S)$.

Algorithm Merge

Input: $CH(P)$ and $CH(Q)$.
Output: $CH(S) = CH(P \cup Q)$
{Initialization}
Let L_p and L_q denote lines supporting P and Q respectively, and tangent to the sites $p \in P$ and $q \in Q$ such that both L_p and L_q are parallel to a given line L^*. Initialize $CH(S)$ empty.

{We make use of a procedure Advance to advance to the next arc in either $CH(P)$ or $CH(Q)$.}

repeat
  if dom(L_p, L_q) **then** Add($CH(S)$, p); Advance(L^*, p, q);
    else {dom(L_q, L_p)} Add($CH(S)$, q); Advance(L^*, q, p);
  $L_p \leftarrow$ line parallel to L^* and tangent to P at p;
  $L_q \leftarrow$ line parallel to L^* and tangent to Q at q;
until every arc in $CH(P)$ and $CH(Q)$ has been visited.

procedure Advance (L^*, x, y);
{Find common lines of support and advance on the minimum angle.}
{We test for edges that bridge the two hulls, that is, edges of the form $t(x, y)$ and $t(y, x)$.}
{If $L(x, y)$ does not exist then $\alpha(L^*, L(x, y))$ is undefined.}
  $a_1 \leftarrow \alpha(L^*, L(x, y))$; $a_2 \leftarrow \alpha(L^*, L(x, \text{succ}(x)))$;
  $a_3 \leftarrow \alpha(L^*, L(y, \text{succ}(y)))$; $a_4 \leftarrow \alpha(L^*, L(y, x))$;
  if $a_1 = \min(a_1, a_2, a_3)$ **then** Add($CH(S)$, y); {$t(x, y)$ is a bridge}
    if $a_4 = \min(a_4, a_2, a_3)$ **then** Add($CH(S)$, x); {and $t(y, x)$ is a bridge too}
  if $a_2 < a_3$ **then** $L^* \leftarrow L(x, \text{succ}(x))$; $x \leftarrow \text{succ}(x)$;
    else {$a_3 < a_2$} $L^* \leftarrow L(y, \text{succ}(y))$; $y \leftarrow \text{succ}(y)$;

Figura 1: Algoritmo para calcular el cierre convexo

El algoritmo parece corto dado que utiliza algunas primitivas geométricas, por ejemplo $L(x, y)$ nos regresa la línea de soporte para S que es tangente a x y y si existe tal línea; $\alpha(L_1, L_2)$ nos da el ángulo entre dos líneas y $succ(x)$ nos regresa al elemento siguiente del cierre convexo al que pertenece x.

Propuesta

Mi propuesta es hacer una implementación de éste algoritmo en *haskell* definiendo los discos, las primitivas necesarias, y el algoritmo en sí.

La implementación de éste algoritmo será suficiente para mi exposición; sin embargo para el proyecto de ésta materia quiero hacer una interfaz sencilla en *"elm"* para dibujar los círculos, y a partir de ellos genere un archivo con los círculos descritos por identificador, su centro y su radio, éste archivo será procesado en *haskell* y devolverá otro archivo con las líneas tangentes entre los círculos que se cargará en *"elm"* y así poder visualizar el cierre convexo.