

# Boolean Operations in Practice

Sherif Ghali

Department of Computing Science  
University of Alberta

## 1 Introduction

The CGAL library [2] provides (release 3.2.1, July 2006) two data structures and accompanying algorithms for the computation of boolean operations on point sets in the plane. Nef polyhedra [1] capture linearly bounded (and possibly irregular) point sets in the plane and make it possible to compute boolean operations on such sets [4, 5]. The Boolean set operations (BSO) package [2, Chapter 12] computes regularized boolean operations on polygons with holes.

A third fundamental technique is known for representing point sets. A Labeled-Leaf Binary Space Partitioning (LLBSP) tree is a BSP tree whose leaves are labeled with a boolean flag. LLBSP trees can be used to compute boolean operations on point sets in the plane [6].

Our recent implementation of robust polygon splitting made it possible to compute boolean operations using LLBSP trees robustly. Using CGAL's extended kernels [3] makes it possible to handle both bounded and unbounded point sets.

Figure 1 shows an example of computing a boolean expression on three polygons. Convex regions corresponding to leaf nodes labeled true are shaded and each splitting line is drawn to the boundary of the convex region it splits.

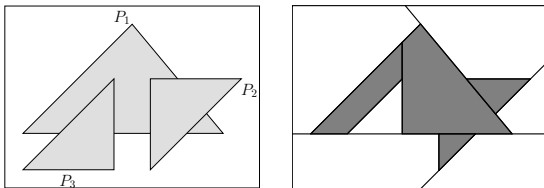


Figure 1: The (implicit) tree on the right results from computing  $(P_1 \cup P_2) \setminus P_3$  on the set of polygons on the left.

## 2 Comparison

Two comparisons of the performance of the three implementations are shown: a comparison of the time it takes to

compute boolean union on a set of triangles and the time it takes to query the resulting set for point inclusion. Even though the point sets used for the boolean operations and for the queries are identical, there are several differences in how the comparisons are executed.

- BSO computes regularized boolean operations. LLBSP\_2 captures only regular sets. Nef\_2 captures and computes boolean operations on sets with arbitrary neighborhoods.
- Nef\_2 and LLBSP\_2 use CGAL's extended kernels, which make it possible to capture and compute boolean operations on unbounded sets.
- Different geometric kernels [2] are used. Nef\_2 and LLBSP\_2 use `Filtered_extended_homogeneous<Gmpz>` whereas BSO uses `Cartesian<Gmpq>`.

Figures 2 and 3 show the construction and query graphs. The source code for LLBSP trees and for generating the graphs can be downloaded from <http://www.cs.ualberta.ca/~ghali/LLBSP.2>.

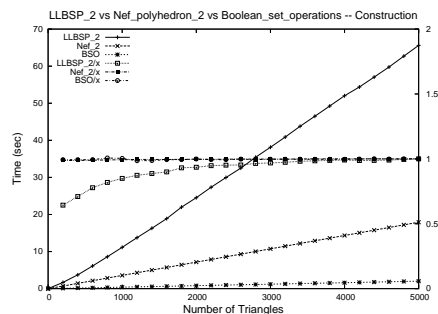


Figure 2: Time for computing identical boolean operations using the three methods. The kernel `Filtered_extended_homogeneous<Gmpz>` is used for both LLBSP\_2 and Nef\_polyhedron\_2 and the kernel `Cartesian<Gmpq>` is used for Boolean\_set\_operations.

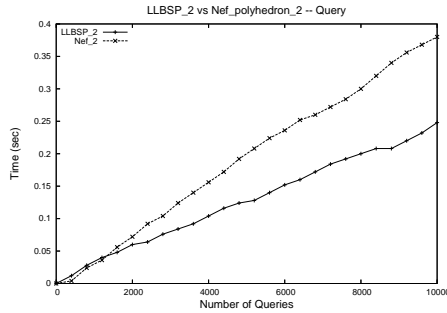


Figure 3: Query time in the resulting point sets in the plane.

### 3 Polygon splitting

The traditional way to implement a polygon class in a geometric system is to represent the vertices explicitly and to represent the edges only implicitly, but such a representation is insufficient for our purposes:

1. The edges need to be promoted and captured in a general polygon class. In the case of an LLBSP\_2 implementation, this makes it possible to flag those edges that result from a polygon split. Such edges do not need to be represented when constructing a leaf node in the LLBSP tree as constructing them would introduce nodes representing empty sets.
2. Representing a polygon using its vertex coordinates means that the number of bits needed for its coordinates under an exact kernel is in the worst case proportional to the depth of the node in the tree where the vertex was generated. The growth of the number of bits (even with filtering) makes it impossible to handle arbitrarily large sets. What is needed is to define the edges using input line segments. The vertex coordinates are determined only when splitting is needed, and the polygon fragments are then also stored using input line segments. The pseudo-code is shown in Figure 4.

### 4 Acknowledgements

Helpful feedback from Hervé Brönnimann is gratefully acknowledged. This work was supported by a grant from NSERC.

### References

[1] H. Bieri and W. Nef. Elementary set operations with  $d$ -dimensional polyhedra. In *Computational Geometry and its*

#### Split(Polygon P, Line L)

```

returns positive_polygon, negative_polygon: Polygon
determine vertices of P
classify vertices of P with respect to L
if no vertex lies in  $L^-$ 
or no vertex lies in  $L^+$ 
    for each bounding line of P
        if line coincides with L
            set the corresponding edge flag
    if no vertex lies in  $L^-$ 
        copy P with new edge flags into positive_polygon
    if no vertex lies in  $L^+$ 
        copy P with new edge flags into negative_polygon
    return
if all vertices of P lie on L
    return
vector_of_lines positive_lines, negative_lines
vector_of_flags flags_of_positive_lines, flags_of_negative_lines
for each bounding edge e of P
    if e.source is not in  $L^-$  and e.target is not in  $L^-$ 
        insert e to positive_lines
        insert flag of e to flags_of_positive_lines
        if e.target lies on L
            insert L to positive_lines
            insert true to flags_of_positive_lines
    else if e.source is not in  $L^+$  and e.target is not in  $L^+$ 
        // symmetric case
    else // segment straddles the splitting line; split
        find intersection point I
        if e.source lies in  $L^+$  and e.target lies in  $L^-$ 
            insert e to positive_lines
            insert flag of e to flags_of_positive_lines
            insert L to positive_lines
            insert true to flags_of_positive_lines
            insert e to negative_lines
            insert flag of e to flags_of_negative_lines
        // The symmetric next case is included for completeness
        if e.source lies in  $L^-$  and e.target lies in  $L^+$ 
            // symmetric case
construct positive_polygon from positive_lines and flags_of_positive_lines
construct negative_polygon from negative_lines and flags_of_negative_lines

```

Figure 4: Robust polygon splitting

*Applications*, volume 333 of *Lecture Notes Comput. Sci.*, pages 97–112. Springer-Verlag, 1988.

[2] *CGAL Reference Manual*, CGAL 3.2.1 edition. <http://www.cgal.org/>.

[3] K. Mehlhorn and M. Seel. Infimaximal frames: A technique for making lines look like segments. *Int. J. Comput. Geometry Appl*, 13(3):241–255, 2003.

[4] M. Seel. Implementation of planar nef polyhedra. Research Report MPI-I-2001-1-003, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, August 2001.

[5] M. Seel. *Planar Nef Polyhedra and Generic Higher-Dimensional Geometry*. PhD thesis, Universität des Saarlandes, August 2001.

[6] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. *SIGGRAPH '87*, 21(4):153–162, 1987.