A Convex Deficiency Tree Algorithm for Curved Polygons

Vadim Shapiro
Mechanical Engineering and Computer Sciences
University of Wisconsin, Madison, WI 53706 USA
vshapiro@engr.wisc.edu

January 3, 2001

Abstract

Boolean set representations of curved two-dimensional polygons are expressions constructed from planar halfspaces and (possibly regularized) set operations. Such representations arise often in geometric modeling, computer vision, robotics, and computational mechanics. The *convex deficiency tree* (CDT) algorithm described in this paper constructs such expressions automatically for polygons bounded by linear and curved edges that are subsets of convex curves. The running time of the algorithm is not worse than $O(n^2 \log n)$ and the size of the constructed expressions is linear in the number of polygon edges. The algorithm has been fully implemented for polygons bounded by linear and circular edges.

1 Background

1.1 Motivation

The problem of converting a boundary representation to a constructive geometry representation is classic in geometric modeling. Historically, the problem arose because both humans and many computer applications tend to describe geometric shapes by specifying their boundaries, while representations of the same shapes by Boolean set expressions offer a number of computational advantages. A detailed account of this problem in solid modeling as well as a general approach to solving it can be found in reference [12]. The problem has been largely solved for several important classes of geometric objects, including three-dimensional solids bounded by second-degree surfaces [14] and two-dimensional shapes considered in this paper[13]. Informally, the proposed solution involves inducing halfspaces from faces of the solid, partitioning the whole space into homogeneous cells (two- or three-dimensional, depending on the dimensionality of the space), and constructing a Boolean expression by "gluing together" those cells that classify to be inside the given shape. The expression is subsequently optimized using Boolean minimization methods.

The above approach to constructing Boolean expressions has at least two undesirable properties. It is relatively computationally intensive, since the running time is at least $\Omega(n^{d+1})$ in d-dimensional space E^d, even before performing Boolean optimization. Furthermore, the constructed Boolean expressions are not always well-formed[11] in the sense that additional neighborhood analysis is required to determine whether certain points belong to the boundary of the solid. Such expressions must further be regularized[1] in solid modeling systems to filter out the points on "dangling boundaries" [16].

Alternative methods of constructing Boolean expressions are known for restricted classes of shapes. One of the more popular approaches for planar linear polygons is based on the concept of convex defficiency tree (CDT) and appears to have been reinvented many times. The first version of the algorithm known to the author appeared in [9]. In English literature O'Rourke [6] attributes a similar algorithm for constructing a convex deficiency tree to [2], and other versions of the algorithm have been described in [20, 17]. The basic $O(n^2)$ algorithm has been improved to $O(n \log n)$ in [3]. This efficient and elegant algorithm also

[1] Regularization of a set X is *closure (interior(X))*.

produces very natural Boolean expressions: the halfspaces in the Boolean expression are in the one-to-one correspondence with the polygon's edges and no further regularization is required [11]. Additional improvements to the basic algorithm are reported in [8] and [4].

Several attempts to generalize and extend the CDT algorithm for decomposing more general geometric domains ran into difficulties. Adaptation of CDT to three-dimensional polyhedra are described in [19], but the algorithm cannot be applied to all polyhedra. The causes of failure of the CDT algorithm in three dimensions have been analyzed by Kim [5] who also showed that additional decomposition of certain polyhedra may be required in order for the CDT algorithm to work. Peterson [7] discussed the importance and the difficulties of extending the CDT algorithm to curved polygons, and a heuristic decomposition algorithm based on pattern matching for the same class of objects is described in [18].

This paper shows that the CDT algorithm naturally extends to a large class of curved two-dimensional polygons while preserving many of the attractive properties of its linear counterpart. The remainder of this section gives a brief background on CDT algorithm and the difficulties in applying it to curved polygons. Section 2 describes the generalized version of CDT algorithm and estimates its running time and the size of constructed Boolean expressions. A complete implementation of the algorithm which heuristically improves the Boolean expressions is discussed in Section 3, which is followed by a brief conclusion.

1.2 Algorithms for linear polygons

A *polygon* is a homogeneously two-dimensional subset of a plane bounded by a closed piecewise smooth curve. In this paper we will deal only with simply connected polygons, which means that the polygon's boundary is a single loop of smooth edges (a simple connected 1-cycle). We refer to a polygon as linear when all its edges are line segments, and curved otherwise.

The essence of the convex-hull based method rests on the observation that any linear polygon P can be represented as its convex hull minus[2] a finite number of "concavities"(Figure 1). Each of the concavities is processed recursively by computing its convex hull and its concavities, and so on until all concavities are convex. In Figure 1 polygon P is represented as its convex hull minus polygon P_1, which in turn is represented as convex hull of P_1 minus its concavities P_2 and P_3, etc. At termination, the original polygon P is represented as a Boolean combination of convex polygons. Each convex polygon is immediately representable as an intersection of the halfplanes associated with the polygon's edges. The running time of this CDT algorithm is $O(n^2)$ including $O(n)$ to compute the convex hull of a simple polygon [17]. Using $(+)$ to denote union and $(\cdot)$ intersection, the resulting Boolean expression for the polygon P in Figure 1 is:

$$aH_1nopq - (\overline{h}H_1H_2\overline{m}\overline{l}H_3 - ((H_2bcdH_4 - (\overline{g}H_5H_4 - efH_5)) + H_3ijk)),$$

where lower case letters represent linear halfplanes associated with the polygon's edges, H_i are added convex hull halfplanes that close concavities P_i, and $\overline{x}$ is the regularized complement of halfplane x.

Every time a concavity is closed with a convex hull edge, the representation of the polygon leads to two additional construction halfplanes: one to bound the convex hull of the original polygon and one to close the concavity. These construction halfplanes are denoted by H_i in Figure 1. A slight modification of the basic CDT algorithm yields Boolean expressions that do not use any additional construction halfplanes [3]. The main idea is illustrated in Figure 2: the polygon can also be viewed as the intersection of two polygonal (and possibly unbounded) regions: one bounded by the polygonal chain of edges on the original polygon and another, associated with the *complement* of the concavity and bounded by the polygonal chain of the concavity edges. For precise definitions and detailed discussion the reader is referred to [3]. DeMorgan's laws of Boolean algebra and recursive application of this argument leads to the conclusion that the polygon can be represented by an expression of the form:

$$h_1h_2\ldots(h_j + h_{j+1} + \ldots(\cdots(\cdots)\cdots)\ldots + h_i)h_{i+1}\ldots h_n,$$

where h_i is the linear halfplane associated with the ith edge of the polygon, and addition and multiplication denote respectively union and intersection. For the polygon in Figure 1, the resulting Boolean expression is:

$$a(bc(d(ef + g) + h) + ijk + l + m)nopq.$$

[2] Throughout the paper, we assume that the set operations (union, intersection, and difference) are regularized.

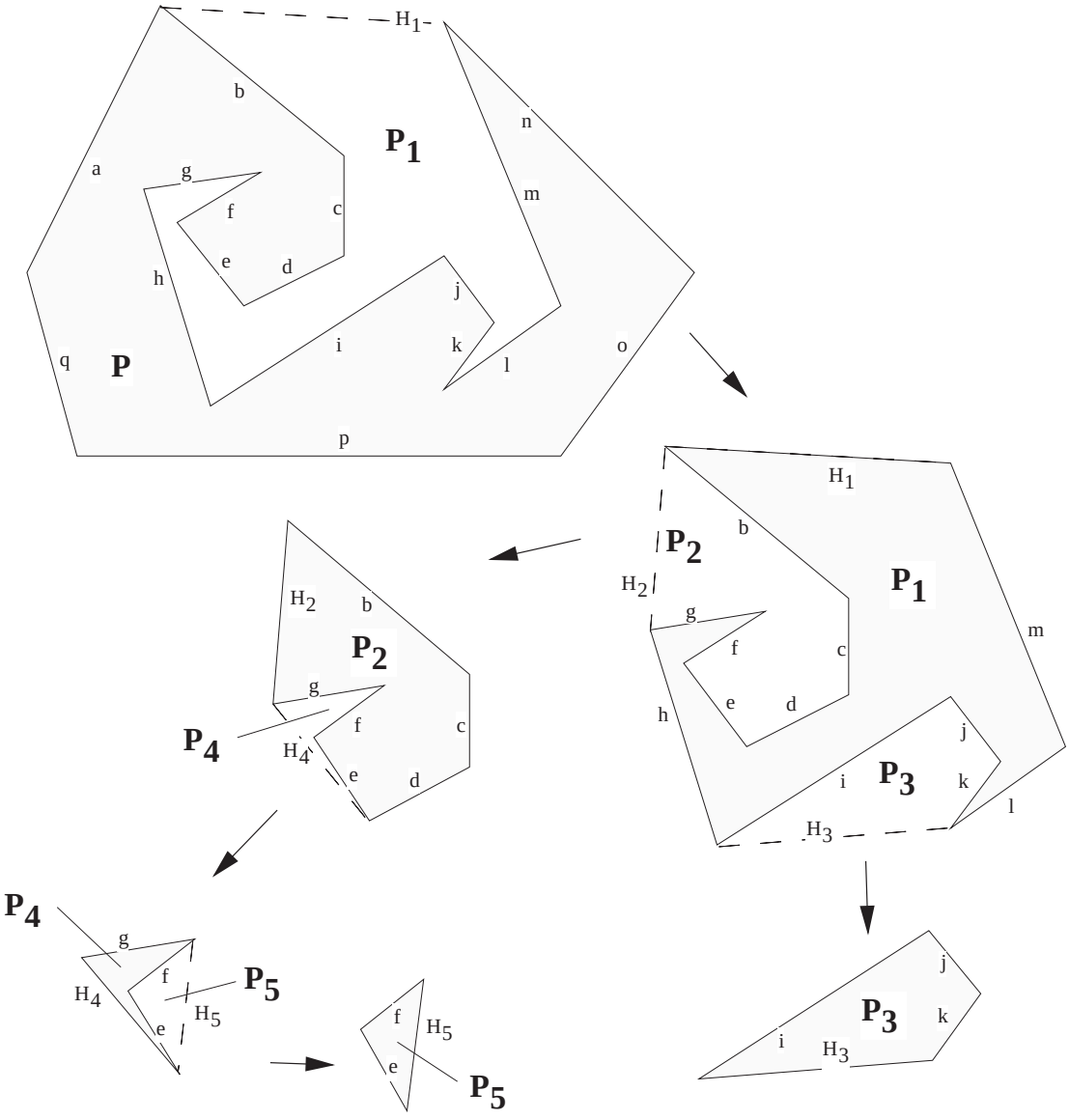


Figure 1: A convex defficiency tree algorithm for a simple linear polygon

Note that the halfplanes appear in the same order as edges on the original polygon and each halfplane appears exactly once. Furthermore, every pair of parentheses corresponds to the alternation between operations of union (+) and intersection (·); and the correct placement of parentheses corresponds exactly to the vertices connected by the convex hull edges computed at every level of recursion[9]. However, it should be noted that the convex hull edges computed for the regions bounded by semi-infinite chains can be *different* from the convex hull edges computed for the bounded polygon. Simply discarding the convex hull edges in the original CDT algorithm may produce an incorrect Boolean expression as shown in [7].[3] Concise and elegant proofs of the above facts can also be found in [3], where it is also shown that the resulting Boolean expression can be constructed efficiently in $O(n \log n)$ time.

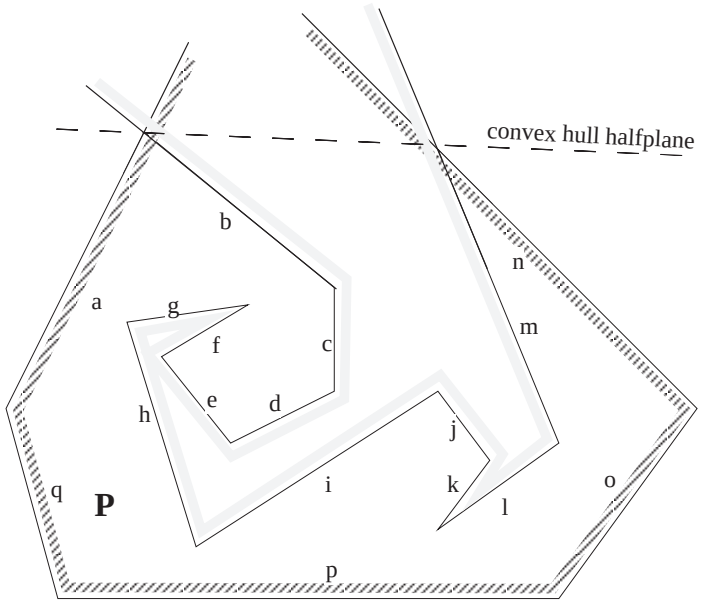


Figure 2: A linear polygon can be represented as the intersection of polygonal chains that meet at the vertices of convex hull

1.3 Problems with curved polygons

It is not difficult to see that the same CDT algorithm does not work on curved polygons. While it is true that any curved polygon can be represented as its convex hull minus a finite number of concavities, Figure 3 shows an example of a curved polygon for which the CDT algorithm does not terminate. After any number of decomposition steps, the boundary of the concavity will contain the inflection point, thus remaining non-convex.

A Boolean expression for this particular polygon can be easily constructed by examination, as shown in Figure 4. This example suggests that curved polygons may be represented as linear polygons that are "deformed" by adding or subtracting curved sectors associated with the straight edges. A number of such special situations have been analyzed by Peterson [7] but, as we explain below, a general implementation of this approach is not trivial.

Properties of curved polygons were also studied by Souvaine in her Ph.D. thesis [15].[4] She proposed a notion of a linear *carrier* polygon which is obtained for any curved polygon by simply connecting the vertices of the polygon in the order they occur on the polygon's boundary. Figure 5 shows that the carrier of a curved polygon does not always have the same topology. The curved polygon in Figure 5(a) has a non-simple carrier shown in Figure 5(b), i.e. its boundary is a self-intersecting polygonal curve. Notice that the boundary of

[3] The author is grateful to John Woodwark for drawing his attention to this fact.

[4] In [15] curved polygons are called "splinegons."

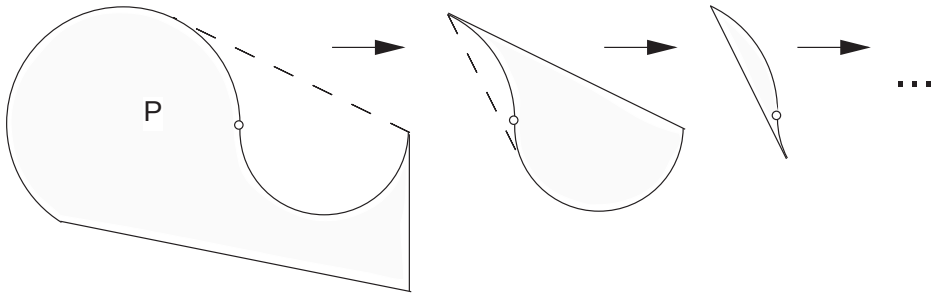


Figure 3: The classical CDT algorithm does not terminate for this curved polygon

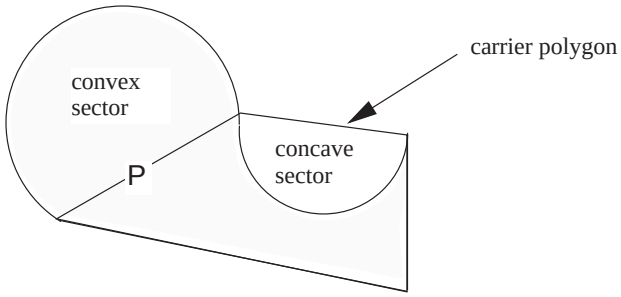


Figure 4: A curved polygon can be represented as a linear carrier polygon with curved sectors attached to the carrier: convex sectors are unioned and concave sectors are differenced

the carrier has to preserve the same orientation as the original polygon. Thus, the carrier of the polygon in Figure 5(c) is the unbounded plane with a polygonal hole and is not simple, even though its boundary is a simple closed curve shown in 5(d).

Even if the carrier is a simple polygon, it may not be possible to deform it into the given curved polygon by unioning and/or subtracting curved sectors. An example from [7] is shown in Figure 6. Here every convex/concave curved sector invades some other concave/convex sector in the cyclical order, implying that no correct sequence of unions/subtractions exists. Souvaine showed that every curved polygon can be decomposed into $O(n)$ monotone curved polygons, each of which can be decomposed into the carrier polygon and a number of curved "concave-in" and "concave-out" sectors. This in turn implies existence of $O(n)$ Boolean expressions for any curved polygon, which is asymptotically optimal.

The above examples show that constructing Boolean representations using carriers does not work for all curved polygons; nevertheless we will use this technique below as a basis for a more general algorithm. In what follows we describe a generalization of the linear CDT algorithm for a curved polygon using the properties of its carrier polygon to assure termination and a reasonable size of the computed Boolean expressions.

2 Generalized CDT algorithm

2.1 Curved polygons with convex carriers

The presented algorithms, examples, and specific implementation deal with polygons bounded by linear and circular edges, but we explain in the concluding section that the described algorithms may be applicable to polygons bounded by more general curves. The two end points of a curved edge e_i form a line segment, which we will call a *chord*. We will *assume* that every curved edge e_i is a subset of some connected curve C_i forming the boundary of some halfplane h_i and intersecting the chordal line only at the end points of the edge. The material (interior) side of the halfplane h_i is chosen to coincide with that of the polygon. In addition, we will assume that curve C_i has the constant sign of curvature; if the normal pointing to the

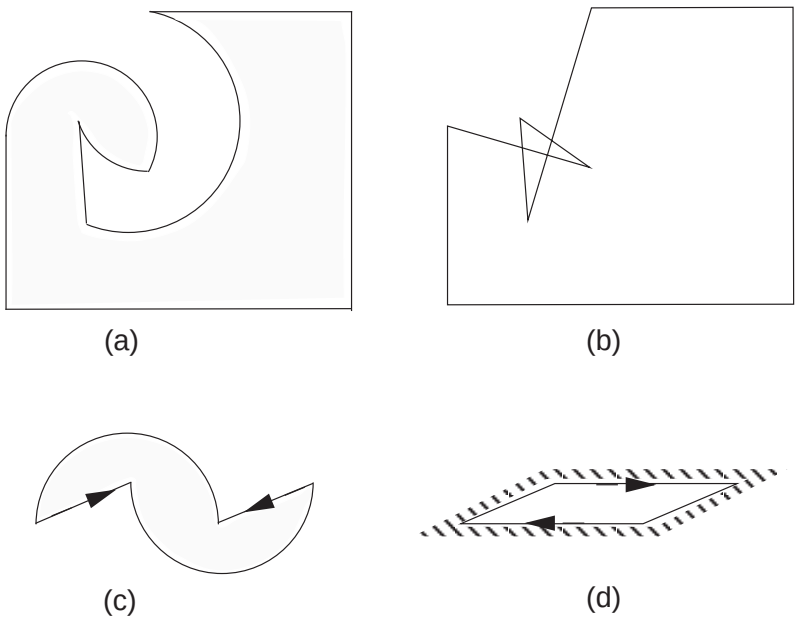


Figure 5: Curved polygons (a) and (c) have non-simple carriers (b) and (d) respectively, because some edges invade curved sectors of the polygons

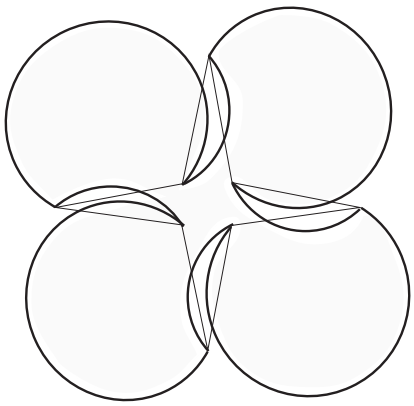


Figure 6: The carrier of this curved polygon is simple, but the eight sectors (four convex and four concave) cannot be attached to the carrier in any order

center of curvature coincides with the material side of halfplane h_i then the curve C_i is *convex*; otherwise it is *concave.*

The curved sector S_i associated with a curved edge e_i has a simple Boolean representation as the intersection of the curved halfplane and the chordal halfplane g_i associated with e_i. The two types of sectors possible with circular edges is illustrated in in Figure 7. For brevity, S_i will be called a *convex sector* if it is associated with a convex edge e_i on the curved polygon and *concave sector* if e_i is concave.

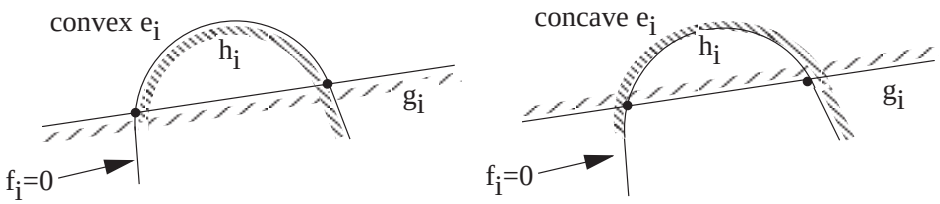


Figure 7: For each curved edge e_i, there is a curved sector: $S_i = \overline{g}_i \cap h_i$ if e_i is convex and $S_i = g_i \cap \overline{h_i}$ if e_i is concave

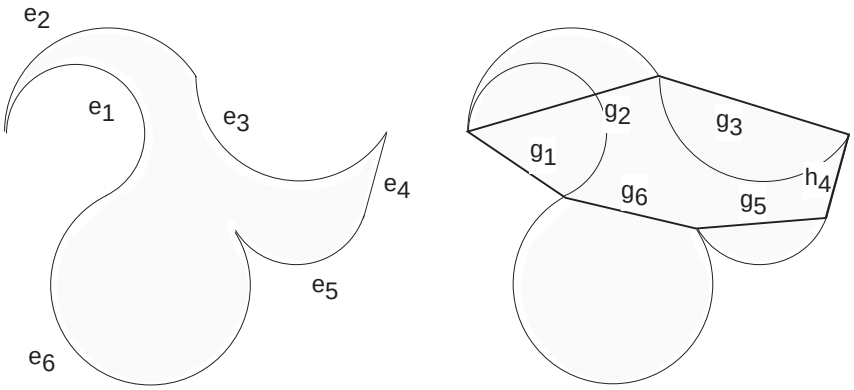
---

When edge $e_i \cap interior(S_j) \neq \emptyset$ we will say that edge e_i (and sector S_i) *invade* sector S_j. When sectors do not invade each other they can be added (unioned) or subtracted (differenced) from the carrier polygon in any order. On the other hand, every sector of polygon in Figure 6 is invaded by some other sector; this implies that no ordering exists for attaching the sectors to the carrier.

Proposition 1 *On a curved polygon with a convex carrier, no convex sector invades a concave sector.*

This follows immediately from the fact that convex sectors contain only points outside of the carrier, and sectors of the same type (convex or concave) cannot invade each other. Clearly, concave sectors can invade the convex sectors even if the carrier is convex. This observation immediately suggests how to construct a Boolean representation of a curved polygon with convex carrier.

Proposition 2 *If the carrier of a curved polygon P is convex, than P can be represented by the carrier polygon union all convex sectors and minus all concave sectors,* in that order *(Figure 8).*



(a) polygon with a convex carrier

(b) union of carrier and convex sectors

Figure 8: A polygon with the convex carrier can be represented by as:
(carrier $\cup$ {all convex sectors}) $-$ {all concave sectors}.

The Boolean representation for the polygon is obtained by observing that the convex carrier is defined by intersection of its supporting linear halfplanes associated with its edges, while each sector is represented as

shown in Figure 7. Thus, for the polygon in Figure 8, the resulting Boolean representation is

$$(g_1 g_2 g_3 h_4 g_5 g_6 + h_2 \overline{g}_2 + h_5 \overline{g}_5 + h_6 \overline{g}_6) - \overline{h}_1 g_1 - \overline{h}_3 g_3,$$

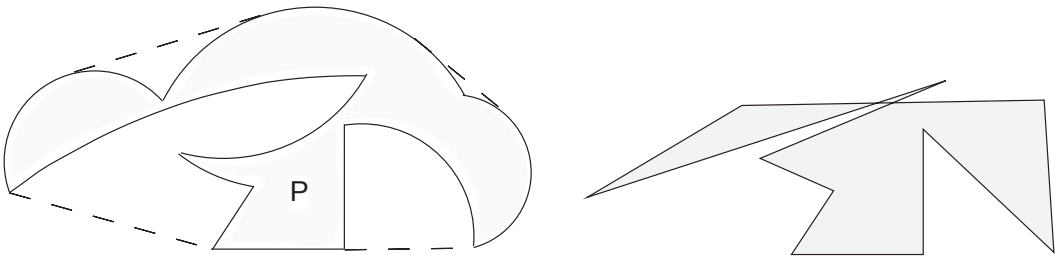
where h_i are the halfplanes associated with the edges e_i, and g_i are the corresponding chordal halfplanes. Within each group (convex or concave) the sectors may be taken in an arbitrary order. As we shall see in Section 3, sometimes more efficient constructions are possible, but in the worst case each curved edge translates into three primitives in the final Boolean expression: two primitives h_i, g_i to represent the curved sector and one more primitive $\overline{g_i}$ to represent the carrier polygon. Finally, we add the obvious observation:

Proposition 3 *A convex curved polygon has a convex carrier.*

We do not give detailed proofs for the above propositions, but they are not difficult; some of them can be found or implied in [15]. These observations provide a foundation for developing a general CDT decomposition algorithm. The basic idea is simple: recursively decompose a given curved polygon into the difference of a polygon with a convex carrier and a finite number of smaller curved polygons.

2.2 The recursive decomposition step

If the carrier of a curved polygon P is convex, then a Boolean representation of P can be constructed according to Proposition 2. If the carrier is not convex, then P is not convex either, by proposition 3. The boundary of the convex hull of P must include a finite number of some new linear hull edges (Figure 9(a)). Each of these edges will close a concavity of P. As a first approximation, we can simply apply the decomposition step of the CDT algorithm for linear polygons: compute the convex hull of P and represent P as the difference of convex hull of P and a finite number of concavities of P. Recursive application of this decomposition immediately yields an algorithm which is identical to the linear CDT algorithm described in section 1.2, except for the termination step. Instead of terminating when the polygon is convex, the recursion stops when the *carrier polygon* is convex. Since every linear polygon is its own carrier, by definition, the classical CDT is a special case of this new algorithm and works without modification in the special case when all edges are linear.



(a) polygon P and its convex hull

(b) carrier of the polygon P

Figure 9: A polygon with non-convex carrier and its convex hull

---

Figure 10 shows execution of the proposed recursive algorithm on the polygon P in Figure 9. It should be clear that all polygons with convex carriers, including the convex hulls of P and all terminal concavities P_i, have Boolean representations as described above. However, it is not obvious that this algorithm actually terminates in all cases and for all polygons. Also notice that each decomposition step may be accompanied by splitting of some curved edges into two, which implies the growth in size of the resulting Boolean representation. The next Section (2.3) shows that indeed the algorithm always terminates and estimates that the worst case growth of the output is linear in the number of edges of the original polygon. The following Section 3 describes a number of modifications to this basic algorithm aimed at further improving the constructed Boolean representations in terms of their size and structure.

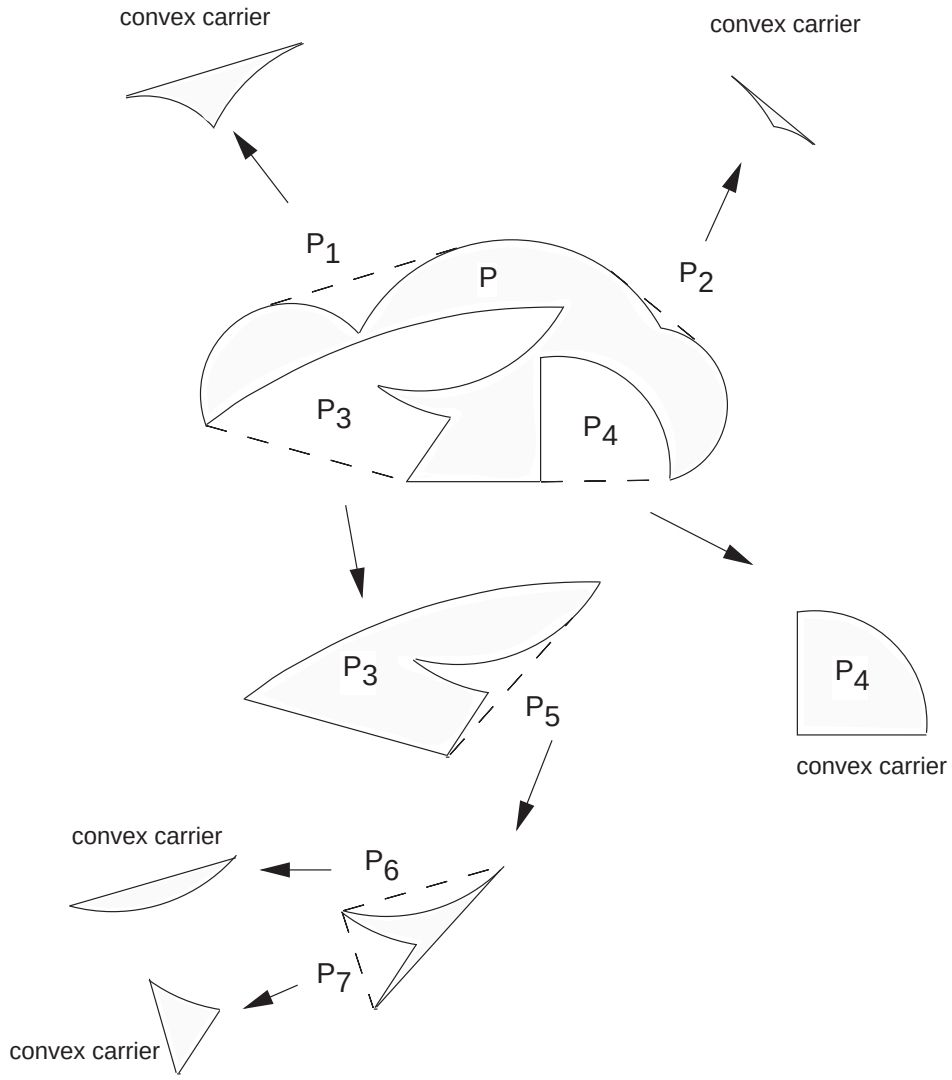


Figure 10: Execution of the generalized CDT algorithm on polygon in Figure 9

2.3 Why the algorithm works

The classical CDT algorithm may not terminate for a curved polygon P because newly computed convex hull edges split some curved arcs bounding P, introducing new curved edges *ad infinitum*. The generalized CDT algorithm proposed above also splits curved edges during the decomposition step. To show that this new algorithm terminates, as well as to estimate its running time and the size of the constructed Boolean representation, we need to bound the total number of possible edge splits over the execution of the algorithm. We do not attempt the rigorous proof or analysis, but provide sketches of proofs to support the estimates below.

It is easy to show that a single curved edge can be split $O(n)$ times. In the example of Figure 11 the single circular arc is repeatedly split during the recursive decomposition steps. This might suggest that the total number of splits can be at least $O(n^2)$ or, worse, may grow unbounded in some other cases. A more careful analysis reveals that this is not the case.

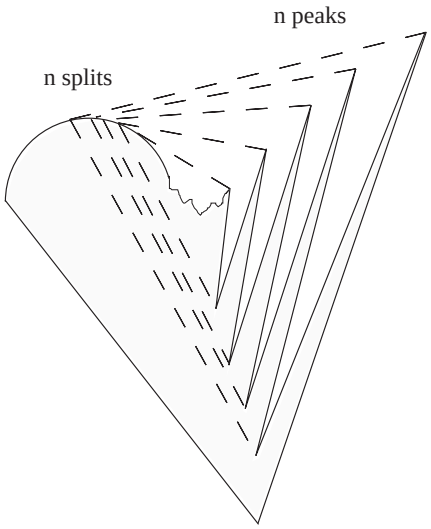


Figure 11: A single curved edge can be split $O(n)$ times during the execution of the CDT algorithm

---

Instead of counting the number of splits on individual edges, we will estimate the *total* number of splits arising *between* the edges of the original polygon P. We will say that edge a *is split by* edge b when during some recursive decomposition step a new convex hull edge from edge b splits edge a. The edge b may or may not be split as well, i.e. the convex hull edge may start at any point of b including its vertices.

Proposition 4 *Edge a can be split by edge b at most twice.*

Only convex edges can be split during the computation of the convex hull. There are at most two tangents to a convex edge a that is being split, and each of the tangents may be a new convex hull edge. Consider one of them from edge b to edge a. Without loss of generality, suppose edge a and b are split into edges a_1, a_2, b_1, b_2 as shown in Figure 12. Edge a_2 will never be split again, because it lies on the convex curved polygon P. Edge a_1 may be split again, but *not* by edges b_1 or b_2. The first one is obvious, because b_1 is on the convex hull of P, while a_1 is on the concavity. Both edges a_1 and b_2 are on the same concavity Q, which will be processed recursively. There are two possibilities: either both a_1 and b_2 are on the convex carrier of Q, or a_1 belongs to a concavity of Q that does not include b_2. Thus, a will be split by b only once for every tangent from b to a, and there are at most two such tangents. Furthermore, since both a and b may be split by each other, we can now estimate a total number of splits.

Proposition 5 *Every pair of edges of P contributes at most four splits over the execution of the algorithm.*

Every edge split is accompanied by an introduction of a new a convex hull edge that is used twice: once in the convex hull of the original polygon P and second time to close the concavity. Thus after a finite number

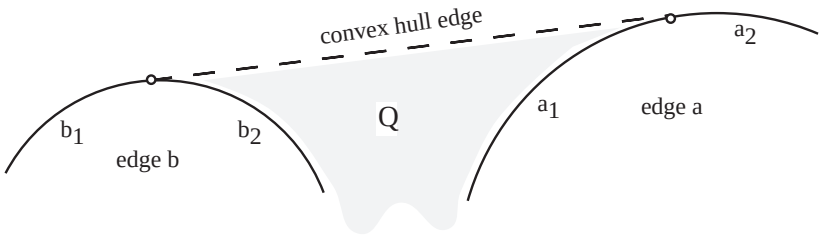


Figure 12: A split of edge a by convex hull edge from edge b

of edge splits every concavity will either become convex or will have a convex carrier. This immediately proves that the proposed algorithm terminates.

Proposition 6 [5] *The total number of splits is $O(n)$.*

After the algorithm terminates, let us represent all performed splits by a planar graph. The original edges of the polygon P are represented as nodes arranged in the order on a circle. The splits will be represented by arcs connecting the nodes on the circle. Each interaction between edges a and b may be represented by an arc connecting a and b. If the splitting tangent is contained outside the original polygon; if the splitting tangent is contained inside the polygon, then the interaction is represented by an arc inside the circle. With this arrangement, no two arcs may intersect, because this would indicate that two edges that had been separated by a split to different carriers are now interacting again. Thus, at termination, we have a *planar* graph with n nodes and at most two arcs per pair of vertices, implying a total number of $O(n)$ arcs.

Thus, the resulting Boolean expression will involve no more than linear number of curved sectors. Since each curved sector corresponds to at most three primitives in the Boolean expression, the proposed algorithm generates Boolean expressions of $O(n)$ size. It takes $O(n)$ time to compute the convex hull of a curved polygon[10, 1] and to extract one or more concavities; this procedure is repeated at most $O(n)$ times. The combined running time of the generalized CDT algorithm would be $O(n^2)$, if it were not for the need to check for self-intersections in the edges bounding the convex carrier polygons. Assuming a standard $O(n \log n)$ test for intersecting line segments, the total worst case asymptotic running time of the algorithm is $O(n^2 \log n)$. The algorithm could be further improved to $O(n^2)$ by noticing that each decomposition step can introduce or eliminate at most a linear number of self-intersections in the carrier chains. Thus, initially all self-intersections of the carrier polygon can be computed at in $O(n^2)$ time, and then can be updated in linear time at every decomposition step.

The above time and size estimates are quite crude and conservative. In a typical curved polygon, edge interactions are likely to produce only few edge splits for some of its curved edges. In this case, the Boolean expressions computed by the CDT algorithm could be more concise than those produced by the decomposition methods described in [15]. The following section discusses a implementation of the algorithm that includes a number of further improvements with respect to the size and the structure of the constructed Boolean expressions.

3 Variant algorithms and implementation

3.1 Reducing the number of split edges

When a curved edge e_i is split into two new edges, the associated halfplane h_i (and possibly the two chordal halfplanes g_i, $\overline{g_i}$) will appear twice in the constructed Boolean expression. Examples in Figures 10 and 11 show that the proposed algorithm may split a curved edge many times. Such a fragmentation is clearly

[5] The author is grateful to Jack Snoeyink for suggesting this proof.

undesirable, and we now examine a number of modifications to the basic CDT algorithm aimed at minimizing the total number of edge splits.

The first significant improvement requires only a minor modification of the generalized CDT algorithm and is based on the observation that the edges on the convex *portion* of the carrier do not need to be split. For example in Figure 10, neither P_1 nor P_2 need to be subtracted from the original polygon, because they split edges that are already on the convex portion of the carrier of P. Algorithmically, for every newly computed convex hull edge we can define the corresponding polygonal chain in the carrier polygon (for short, the *carrier chain*) as follows. If the hull edge starts (ends) at a vertex of the original curved contour P, then the chain starts (ends) at that vertex; otherwise the edge splits an arc and the carrier chain starts (ends) from the previous (following) vertex. We will say that the carrier chain is *convex* if the neighborhood of every vertex in the chain with respect to the carrier polygon is a sector less than π. It is easy to see in Figure 10 that the carrier chains corresponding to P_1 and to P_2 are each two edges long and are convex, while the carrier chains corresponding to P_3 (four edges) and P_4 (two edges) are not convex; the carrier chain corresponding to P_5 (three edges) and P_7 (two edges) are not convex either, but the carrier chain of P_6 (one edge) is convex.

Proposition 7 *The carrier of a polygon P is not convex if and only if at least one of the edges in the convex hull of P corresponds to the carrier chain that is not convex.*

The proposition is based on the fairly obvious observation that convex deficiency of the carrier polygon is equivalent to presence of some non-convex carrier chain in its boundary. Since no deformation of the carrier edges (with fixed vertices) can remedy this convex deficiency, this concavity in the carrier polygon will be always identified by some convex hull edge of the original polygon P.

In the general CDT algorithm described in Section 2 each new convex hull edge closes a concavity of the curved polygon which is then processed recursively. But only concavities corresponding to the non-convex carrier chains actually need to be processed. All convex hull edges that connect edges on the convex carrier chains can be safely ignored. It is *not* wrong to process them, but it is wasteful in terms of both running time and the size of the resulting Boolean representation. Thus, most of the hull edges and accompanying edge splits shown in Figure 10 are unnecessary (only one of splits corresponding to subtraction of P_5 will remain).

On the other hand, none of the splits in Figure 11 will be eliminated. And in fact, the splitting cannot be entirely eliminated (recall the example in Figure 6) but can be further minimized with some additional processing. In a curved polygon P with a convex carrier, no concave sector can be invaded by another edge of P. This observation provided an easy sequencing mechanism for adding curved sectors to the convex carrier. More generally, for any P with convex or non-convex carrier, we will say that sector S_i is *detachable* if it is not invaded by some other edge of P; in this case P can be decomposed as a "locally straightened" polygon P' and the curved sector S_i (unioned to or differenced from P') of edge e_i. The modified polygon P' is obtained from P by simply replacing the curved edge e_i with its chord. The decomposition step of CDT may generate several hull edges that correspond to non-convex carrier chains and split some curved edges. It may make sense to perform an $O(n)$ test per split edge to see if the corresponding convex sectors can be detached *before* being split. Detaching sectors will completely eliminate splitting in the examples of Figure 10 and 11, but will not work for the polygon in Figure 6, because none of the sectors are detachable.

3.2 Eliminating construction halfplanes

The general CDT algorithm uses two types of auxiliary linear halfplanes in order to construct the Boolean expressions: those associated with computed hull edges and those associated with chords of curved edges. Not all of these halfplanes need to appear in the final Boolean expression.

Every new convex hull edge of the curved polygon P induces two linear halfplanes: one to construct the convex hull of P and one to represent the closed concavity. Recall that neither of these halfplanes need to appear in the Boolean expression for a linear polygon if we represent the decomposition step by intersection of two chains: one corresponding to the convex hull of P and another corresponding to the complement of the concavity (refer to Figure 2). It may be tempting to apply the same technique to the carriers of the convex hull of a curved polygon P and the concavity. Indeed, the carrier of the concavity

can be safely extended beyond the convex hull of P. Thus, the linear halfplane closing the concavity can be replaced with the universal set in the representation of its carrier.[6] On the other hand, Figure 13 shows a counterexample where the linear halfplane associated with the convex hull edge cannot be simply dropped from the representation for the carrier of P: changing the carrier of P into the respective unbounded convex region would produce incorrect results. Thus, the linear halfplane bounding the carrier of P may be dropped only when additional conditions are satisfied.

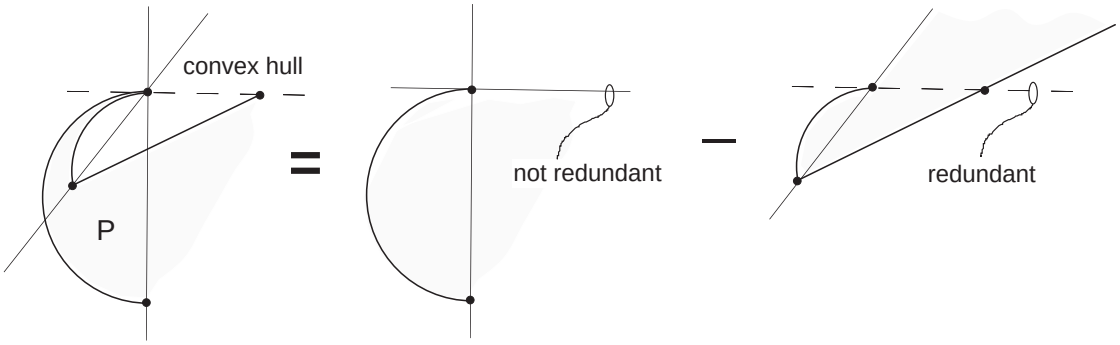


Figure 13: Some convex hull edges correspond to halfplanes that must appear in the final Boolean expression

Recall that in general each curved sector S_i requires two additional linear halfplanes associated with the chord of edge e_i: one to represent the sector and one to represent the carrier. However there is a number of special situations illustrated in Figure 14. If convex halfplane h_i is *containing*, i.e. $P \subset h_i$, neither of the linear halfplanes is needed: we can simply treat e_i as if it were a linear edge on the carrier of P. Alternatively it may happen that convex $h_i \subset P$ or concave $h_i \subset \overline{P}$, in which case we will refer to the halfplane h_i as *contained*. The contained halfplanes can be attached to the carrier directly via union or difference operation, without constructing their respective sectors. In this case only one of the linear chordal halfplanes is needed to represent the carrier.

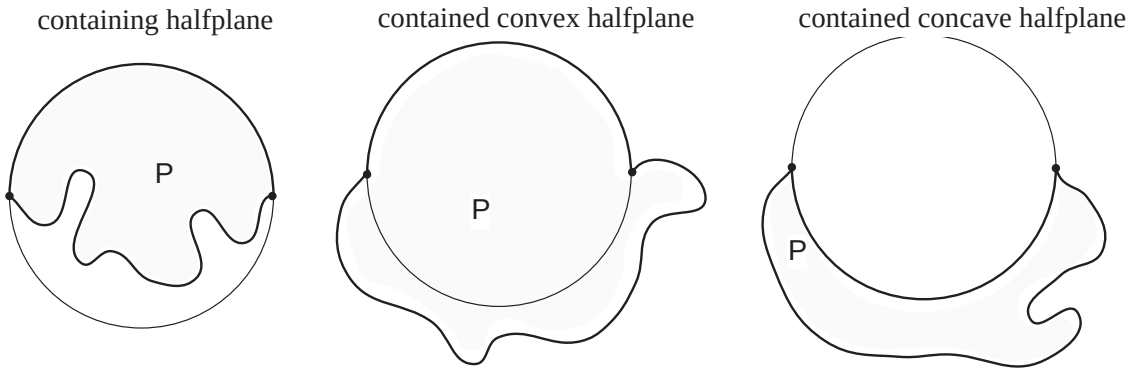


Figure 14: Special situations when only one or none of the chordal halfplanes are needed

3.3 Putting it all together

The above constructions are all local improvement to the basic CDT algorithm described in section 2 and can be used to minimize the size of the constructed Boolean representations, with some additional processing. The

[6] In other words, this halfplane can be simply omitted, since the carrier is represented as the intersection of its supporting halfplanes

non-controversial improvements include processing only concavities corresponding to the non-convex carrier chains, and eliminating unnecessary linear halfplanes as described above. We can also merge coincident edges (i.e. edges that are subsets of the same curve $f(x, y) = 0$) in some special situations, for example when such edges are adjacent on a convex carrier, or when the edges are all subsets of an containing halfplane boundary.

On the other hand, detaching sectors from a polygon P with a non-convex carrier in order to avoid split edges, or rescheduling the processing of contained and containing halfplanes in order to minimize the number of additional construction halfplanes, fundamentally changes the nature of the CDT algorithm and may have subtle and unpredictable consequences:

- The larger P is, the more likely it is that a given convex halfplane is contained with respect to P; similarly, the smaller P is, the higher the likelihood that a concave halfplane is contained. Each contained halfplane reduces the final Boolean expression by one primitive.
- The smaller P is, the more chances are that a given convex halfplane is containing; containing edges reduce the final Boolean expression by two (chordal) halfplanes.
- Detaching the sector of a would-be-split arc could reduce the Boolean expression by one to three primitives, depending on whether the halfplanes associated with the split edges would become containing or contained.
- Closing a concavity on P, and detaching sectors or splitting edges may change the status of the remaining edges, e.g. from invaded to detachable, or from contained to containing, or vice versa.
- Every time we detach a convex (concave) sector, the remaining polygon gets smaller (larger); and so on.

These observations imply that, in contrast to the pure CDT algorithm, the size of the constructed Boolean representation will strongly depend on the ordering of the individual decomposition steps that may include removal of concavities, detachment of would-be-split arcs, and processing of contained or containing halfplanes. Sequencing and balancing these events is a delicate task, where locally optimal decisions may not produce the smallest possible Boolean expression. Design of an effective heuristic strategy must also include considerations of the required computational overhead.

Figures 15 and 16 show the execution of the fully implemented algorithm on two examples. This particular implementation identifies the contained halfspaces and detaches sectors in order to minimize the number of split arcs. The recursive CDT decomposition processes one concavity at a time. The containing halfspaces are detected for polygons with convex carriers, and only the necessary linear halfspaces are included. Each box in the table corresponds to a single (recursive) decomposition step of the program. The picture in the box shows the curved polygon before the step is Executed; the particular decomposition sequence is indicated under the picture. For example, $a \rightarrow b, c$ indicates that the polygon instance a is decomposed into two instances b and c. No decomposition is indicated for single sectors and polygons with convex carriers.

Figure 15 shows the processing of the polygon in Figure 6. A single convex hull edge splits two arcs in step 1, removing a concavity 22. In step 2, a convex contained halfspace (circular disc) is detached; a convex sector 21 is detached in step 3 in order to avoid splitting an arc. Another convex hull edge splits the arc and removes a concavity 16 in step 4; and so on. This particular execution required a total of three arcs splits, but also identified four contained halfspaces (corresponding to the decompositions $2 \rightarrow 3$, $5 \rightarrow 6$, $7 \rightarrow 8$, $22 \rightarrow 23$). The total execution time for this example is 0.015 seconds on a 200 Mhz Pentium class computer.

The second example shown in Figure 16 took 0.031 seconds on the same computer. Three concave contained halfspaces were identified and removed in step 1; two more convex contained halfspaces are detached in step 2. Steps 3 and 4 detach convex sectors in an attempt to avoid splitting arcs, but the arc split in step 5 cannot be avoided. However, this in the only arc split in this particular execution sequences.

3.4 Well-formed representations

Boolean representations of a polygon completely classify the points of E^2 as in, on, or out of P. However, a point classification is not always easy to compute. For an arbitrary point q, it suffices to classify q with

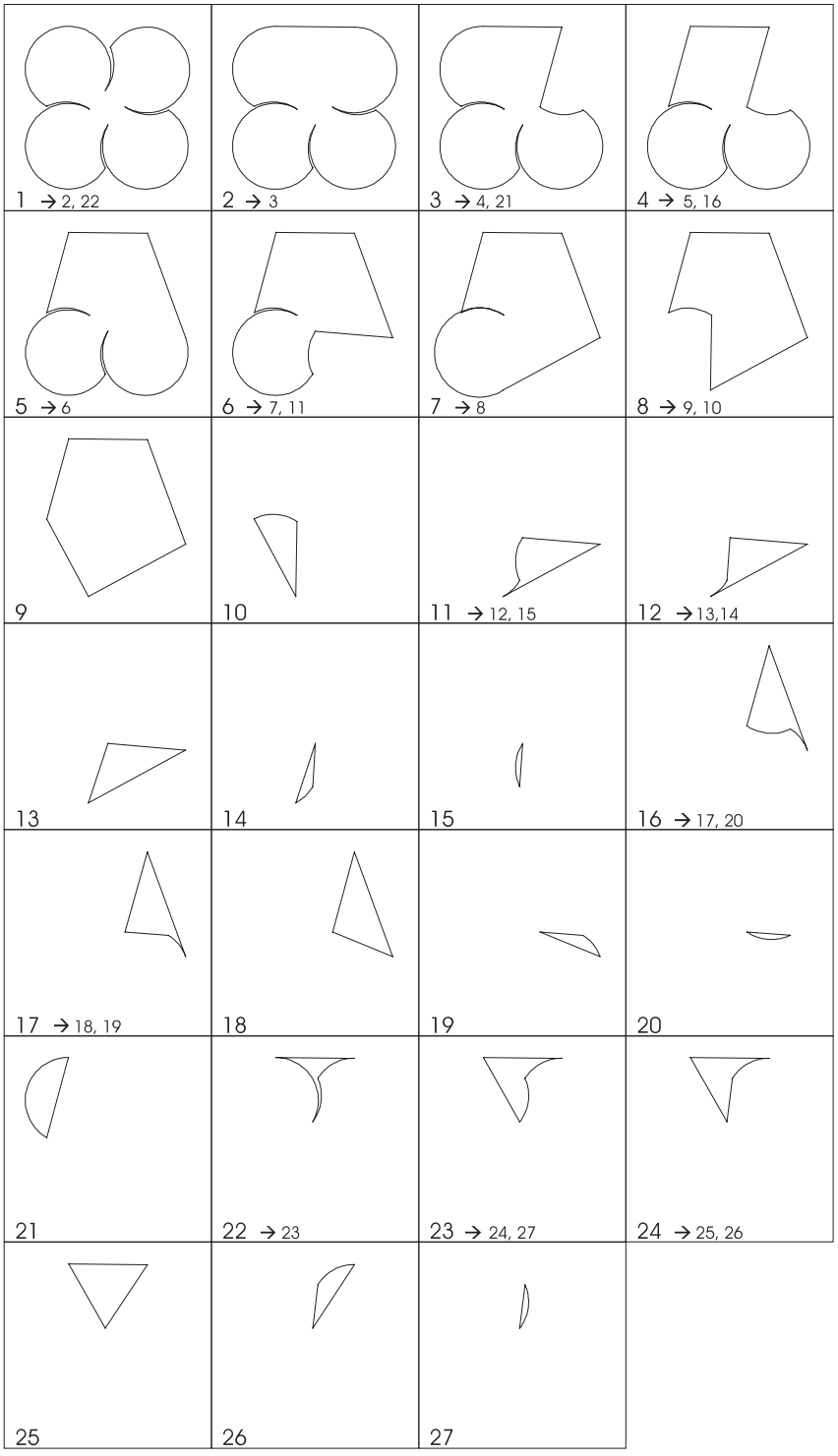


Figure 15: Execution of the implemented algorithm on the curved polygon in Figure 6

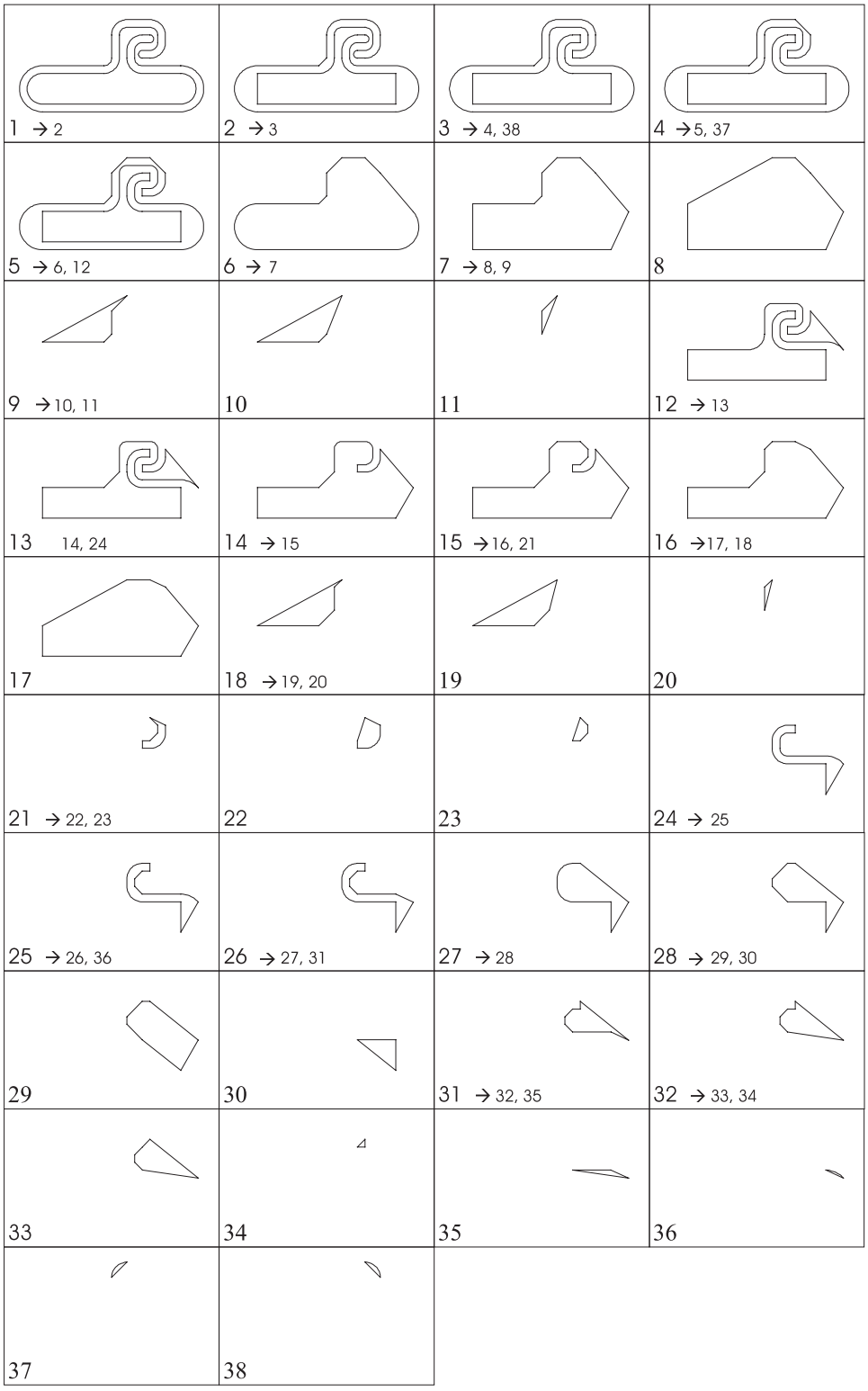
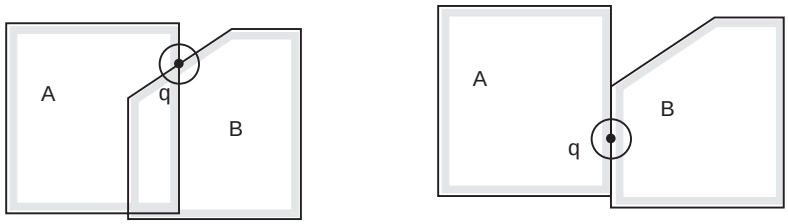


Figure 16: Execution of the implemented algorithm on a typical cross-section of a sheet-metal part

respect to all the halfplanes occurring in the Boolean expression (this involves evaluating the signs of the corresponding functions $f(x, y)$) and to combine the results of the classification according to the Boolean expression. The combine procedure may not be always straightforward, because Boolean (binary logic) functions are not sufficient to distinguish the three states: *in, on, out.* The well known problem of "on-on ambiguities" is illustrated in Figure 17[16]. When a point q classifies *on* A and *on* B, it may or may not classify *on* with respect to sets $A \cup B$ and $A \cap B$. Distinguishing between the situations in Figure 17(a) and 17(b) requires additional neighborhood analysis of point q.



(a) Point p is on A and p is on B implies p is on (A <op> B)

(b) Point q is on A and q is on B but q is not on (A <op> B)

Figure 17: On/on ambiguities of Boolean set representations

It is also possible to construct Boolean set representations such that resolving the on-on ambiguities *always* as *on* produces the correct result. Such representations are called *well-formed* in [11]. Well-formed representations have attractive computational properties that are useful in solid modeling, graphics, robotics, and boundary value problems. We showed in [11] that the linear CDT algorithm always produces well-formed Boolean expressions. In general, the Boolean expressions computed by the generalized CDT for curved polygons are not well-formed. We now show that a slight modification of the CDT algorithm will assure well-formedness, but at the expense of increasing the size of the constructed Boolean expression.

Examine the constructed Boolean expressions in the reverse order of their construction, bottom-up. Each expression is a combination of sub-expressions for curved polygons with convex carriers. The convex carrier of a polygon is a convex set defined as the intersection of the polygon's supporting halfplanes. It should be clear that this representation is well-formed, since on-on points correspond only to the vertices of the carrier. But union (or difference) of this set with a convex (or concave) sector S_i is not well-formed, because the points on the chord of the curved edge e_i are on S_i and on the carrier, but are in (or out for concave) the constructed curved polygon. An alternative method of construction always gives a well-formed Boolean expression for the same set.

Suppose a convex halfspace h_i is to be attached to the convex carrier as shown in Figure 18(a). Computing the convex hull of the union of the carrier with the convex sector S_i identifies two convex hull edges connecting the curved edge and the carrier. The end-points of these edges subdivide the carrier into the *upper* carrier chain u and the *lower* carrier chain l. The chains u and l represent the unbounded convex regions, such that the original carrier is simply $u \cap l$.

Proposition 8 *The union of the carrier and the convex sector can be also represented as* $(u \cup h_i) \cap l$.

This construction can be derived by writing the union of the carrier $(u \cap l)$ and the curved sector $(h_i \cap \overline{u})$ as

$$ul + h_i\overline{u} = (l + h_i)(u + h_i)(l + \overline{u})(u + \overline{u}) = (u + h_i)(l + \overline{u}h_i)$$

But by construction, $\overline{u}h_i \subset l$, which implies that $(l + \overline{u}h_i) = l$. Intuitively, this construction means that the convex sector is attached to the upper carrier, which is then combined with the lower carrier. This set representation is clearly well-formed because the boundaries of u, l, and h_i intersect only at the boundary points of the constructed curved polygon. On the other hand, it is not efficient, because sets u and l are different for distinct edges e_i. When several convex edges appear on the same carrier, the proposed well-formed construction will require that some carrier halfplanes appear more than once in the resulting Boolean expression.

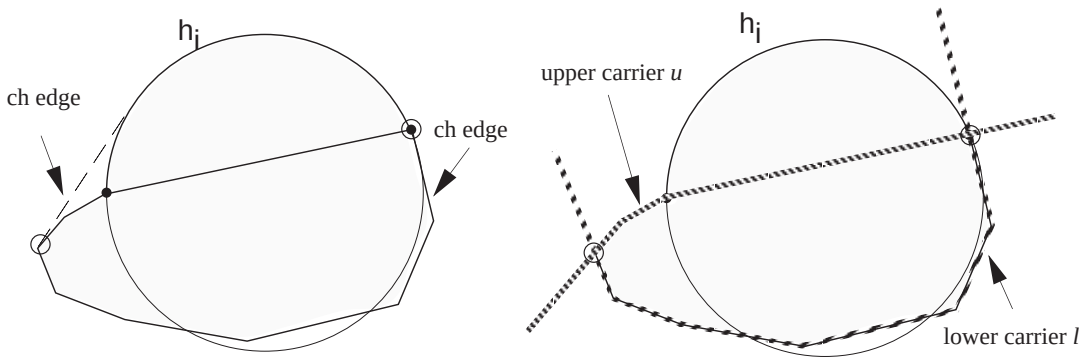


Figure 18: Well-formed attachment of the convex halfspace using upper and lower carrier chains

Similarly, to make the difference of the concave sectors from a polygon P well-formed operation, we subtract not the sector, but the portion of the curved halfspace h_i that is contained in the complement of P. This requires trimming h_i by all convex halfspaces that were previously attached to the convex carrier.

So far we described how to construct well-formed Boolean expressions for curved polygons with convex carrier. If we can assure that the recursive decomposition step preserves well-formedness, then the final constructed expressions will be also well-formed. Recall from section 3.2 that the decomposition step introduces only one new linear halfplane associated with the convex hull edge. This implies that the only points classifying on with respect to both convex hull of P and the concavities are the two vertices connected by the convex hull edge. Clearly, they also classify on with respect to P. Thus, the constructed Boolean expressions indeed are guaranteed to be well-formed.

4 Conclusion

Boolean representations of curved polygons can be constructed in many different ways. All known methods have advantages and drawbacks, and the right choice of the algorithm is typically a compromise between simplicity, speed of computation, and structural properties of the constructed Boolean expressions. The decomposition methods described in [15] are simple, efficient, and asymptotically optimal in size. The cell-based procedures described in [12, 13] are more complicated and computationally intensive, but they compute near-minimal Boolean representations and generalize to a much richer class of objects (for example, higher dimensional semi-algebraic solids). Distinguishing characteristics of CDT algorithms include conceptual simplicity, elegance, and intuitive appeal. They are easy to implement, are quite fast, and produce relatively efficient Boolean representations. CDT algorithms can be easily modified to produce Boolean expressions that are well-formed or satisfy additional requirements. Their weaknesses include seemingly restricted geometric domain, some edge fragmentation, and inability to take advantage of the collinear and cocircular edges.

The proposed CDT algorithm is also intuitively appealing because it can be viewed as a general method to represent curved objects as local (Boolean) deformations of linear carriers. In this sense, splitting curved edges is simply a method for splitting invaded sectors into smaller detachable sectors. This approach may generalize to other geometric domains. For example, every algebraic curve $f_i(x, y) = 0$ of degree k can be subdivided into a finite sequence of convex and/or concave edges[1], and every associated sector is a connected component of the intersection between the chordal halfplane and set $f_i(x, y) \geq 0$. Using techniques from [14] to construct Boolean representations for such sectors and the convex hull procedure described in [1] would extend the described CDT algorithm to polygons bounded by arbitrary piecewise algebraic curves.

In conclusion, we note that the emphasis of this work was on automatic construction of compact and well-formed Boolean expressions, and not on the computational complexity of the proposed algorithms. Many improvements are likely to be possible. It is possible that, using techniques similar to those in [3], the running time of the proposed generalized CDT algorithm may be improved by reformulating it in terms of recursive decomposition of the carrier chains. But such an algorithm may be difficult to reconcile with our

goals of further reducing the number of split edges, minimizing size of the expressions, and constructing only well-formed representations.

Acknowledgments

This work was supported in part by the National Science Foundation grants DMI-9502728 and DMI-9522806. An earlier version of the algorithm described in this paper was designed and implemented by the author in 1988, while he was a member of Cornell Programmable Automation project, for the publicly available Contour-to-CSG software preprocessor to the PADL-2 solid modeling system. The author is grateful to Eric Collins for his help in preparing Figures 15 and 16 and to Jack Snoeyink for numerous suggestions, including the proof of Proposition 6.

References

[1] C. Bajaj and M.-S. Kim. Convex hulls of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 1991.

[2] B. G. Batchelor. Hierarchical shape description based upon convex hulls of concavities. *Journal of Cybernetics*, 10:205–210, 1980.

[3] D. Dobkin, L. Guibas, J. Hershberger, and J. Snoeynik. An efficient algorithm for finding the CSG representation of a simple polygon. *Computer Graphics*, 22(4):31–40, 1988.

[4] J. Hershberger and J. Snoeyink. Cartographic line simplification and polygon CSG formulæ in $O(n \log^* n)$ time. In *Algorithms and Data Structures (WADS '97), number 1100 in Lecture Notes in Computer Science*, pages 530–540. Springer-Verlag, 1997.

[5] Y.S. Kim and D.J. Wilde. A convergent convex decomposition of polyhedral objects. *Transactions of ASME, Journal of Mechanical Design*, 114:468 – 476, 1992.

[6] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1994.

[7] D.P. Peterson. Boundary to constructive solid geometry mappings: a focus on 2D issues. *Computer-Aided Design*, 18(1):3–14, 1986.

[8] A. Rappoport. An efficient algorithm for constructing the convex differences tree of a simple polygon. *Comput. Graph. Forum*, 11(4):235–240, October 1992.

[9] V. L. Rvachev, L. V. Kurpa, N. G. Sklepus, and L. A. Uchishvili. *Method of R-functions in Problems on Bending and Vibrations of Plates of Complex Shape.* Naukova Dumka, 1973. In Russian.

[10] A. A. Schäffer and C. J. Van Wyk. Convex hulls of piecewise-smooth Jordan curves. *J. Algorithms*, 8:66–94, 1987.

[11] V. Shapiro. Well-formed set representations of solids. *International Journal of Computational Geometry and Applications*, 9(2):125–150, 1999.

[12] V. Shapiro and D. L. Vossler. Construction and optimization of CSG representations. *Computer-Aided Design*, 23(1):4–20, January/February 1991.

[13] V. Shapiro and D. L. Vossler. Efficient CSG representations of two-dimensional solids. *Transaction of ASME, Journal of Mechanical Design*, 113:292–305, September 1991.

[14] V. Shapiro and D. L. Vossler. Separation for boundary to CSG conversion. *ACM Transactions on Graphics*, 12(1):35–55, January 1993.

[15] D. L. Souvaine. *Computational geometry in a curved world.* Ph.D. thesis, Princeton Univ., Princeton, NJ, 1986.

[16] R. B. Tilove. Set membership classification: A unified approach to geometric intersection problems. *IEEE Transactions on Computer*, C-29(10):874–883, October 1980.

[17] S.B. Tor and A.E. Middleditch. Convex decomposition of simple polygons. *ACM Transactions on Graphics*, 3(4):244–265, 1984.

[18] D. L. Vossler. Sweep-to-csg conversion using pattern recognition techniques. *IEEE Computer Graphics and Applications*, 5(8):61 – 68, 1995.

[19] T. C. Woo. Feature extraction by volume decomposition. In *Proc. Conference on CAD/CAM Technology in Mechanical Engineering*, Cambridge, MA, March 24-26 1982.

[20] J. R. Woodwark and A. L. Wallis. Graphical input to a boolean solid modeller. In *Proc. CAD '82*, pages 681–688, Brighton, U.K., March 30 - April 1, 1982.