Computer Graphics 6: Viewing In 2D

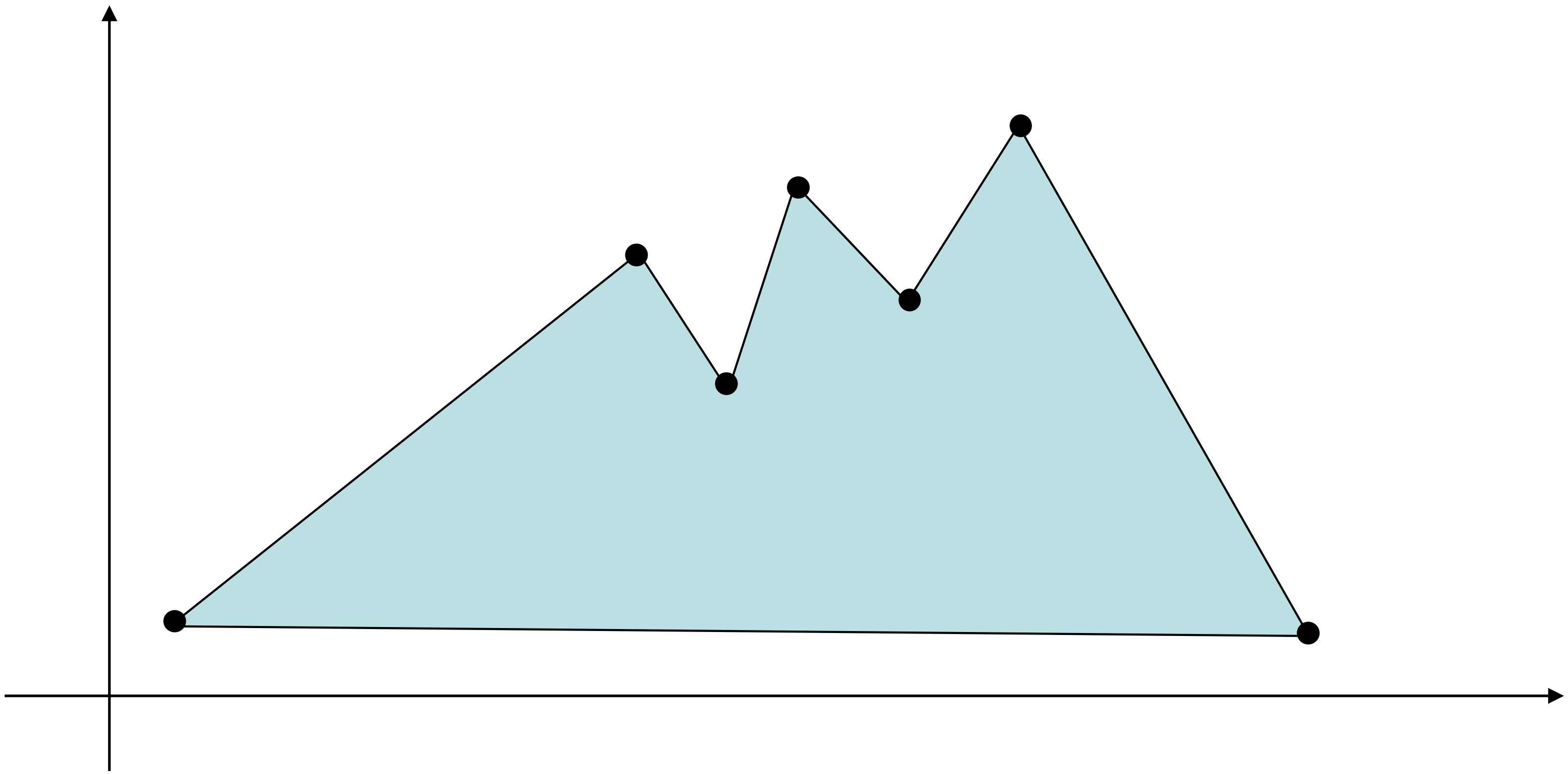
Subhadip Basu, Ph.D
Dept. of CSE, Jadavpur University

Contents

- Windowing Concepts
- Clipping
  - Introduction
  - Brute Force
  - Cohen-Sutherland Clipping Algorithm
- Area Clipping
  - Sutherland-Hodgman Area Clipping Algorithm

Windowing I

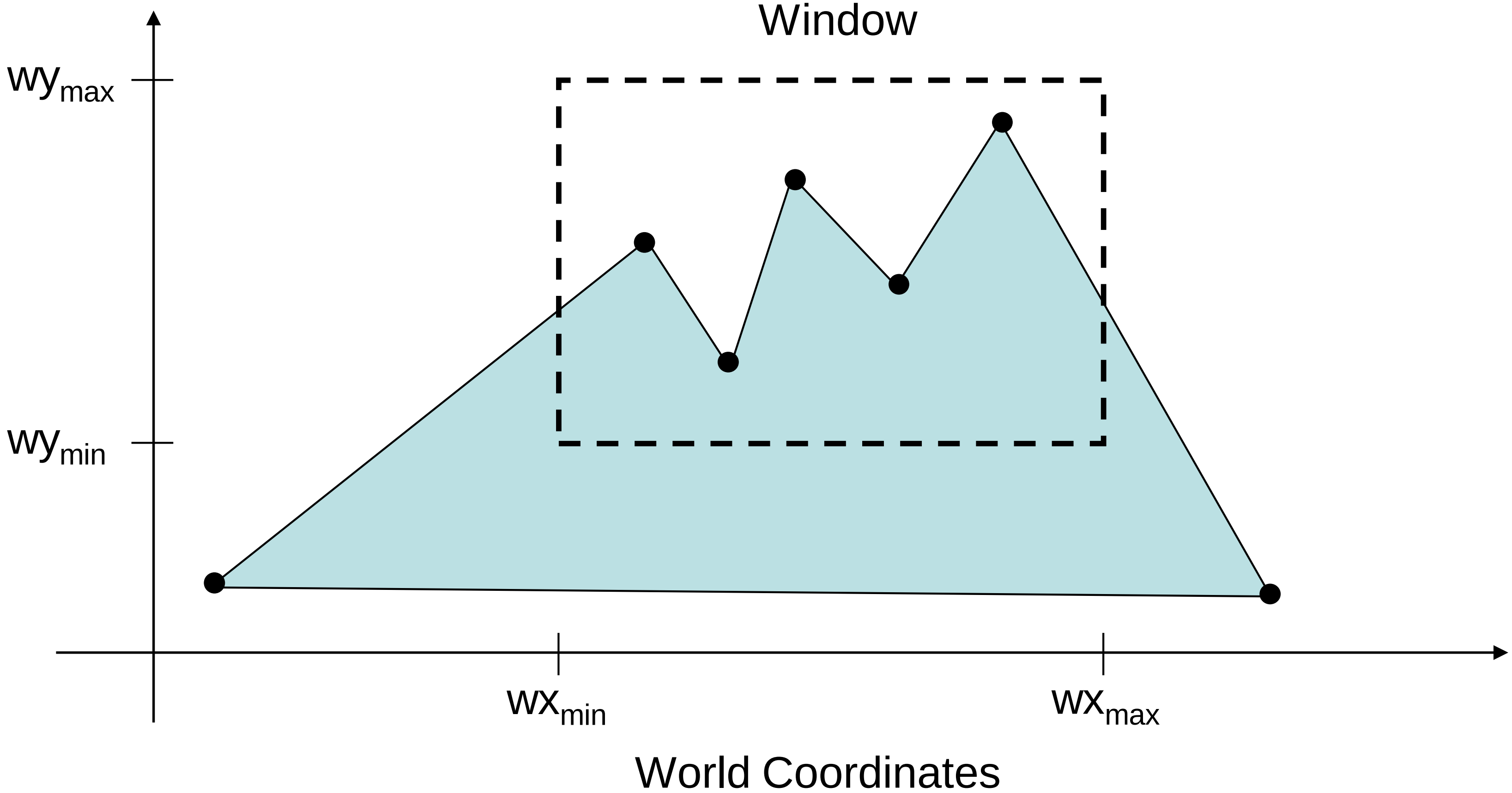
A scene is made up of a collection of objects specified in world coordinates



World Coordinates

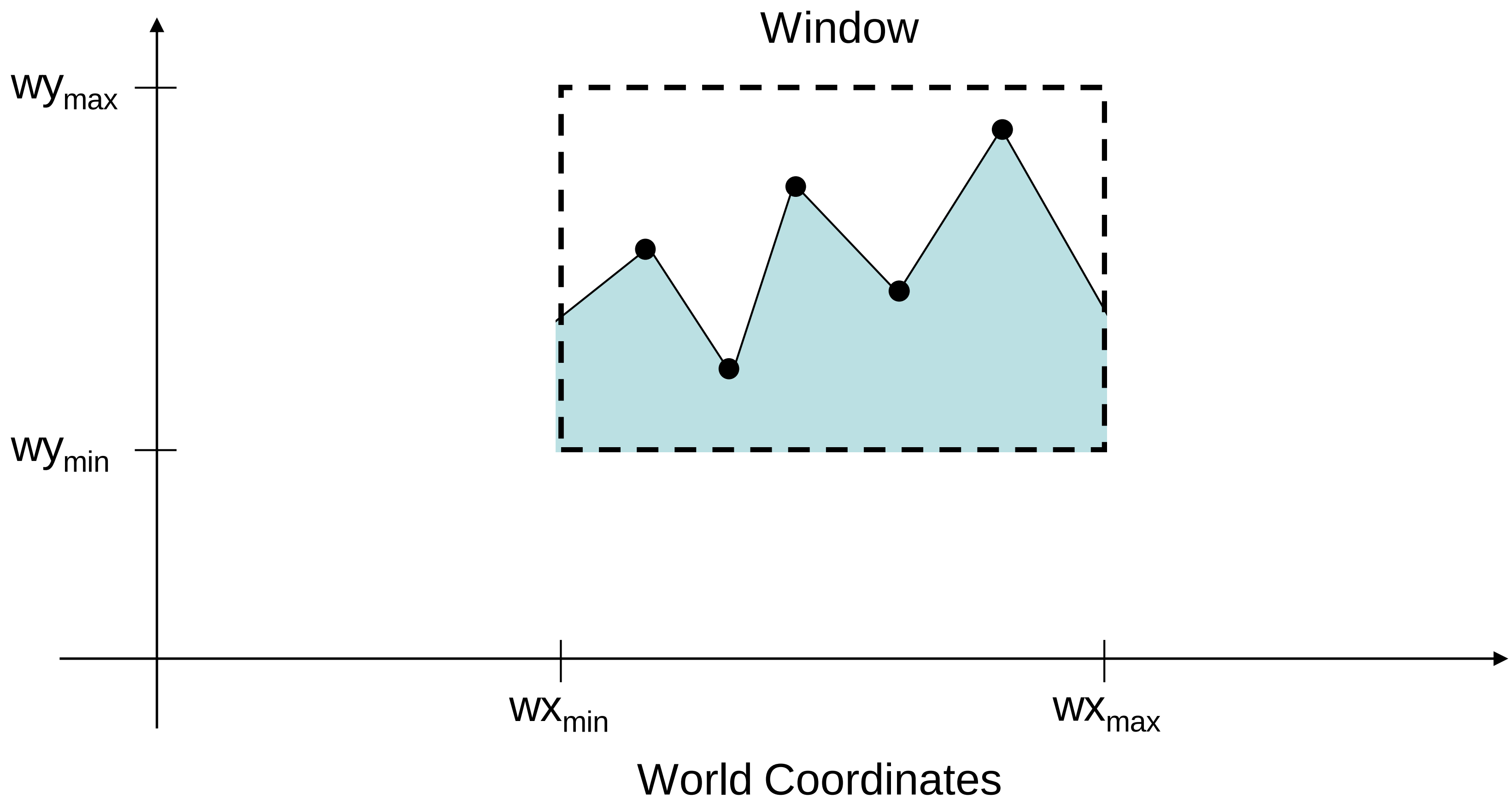
Windowing II

When we display a scene only those objects within a particular window are displayed



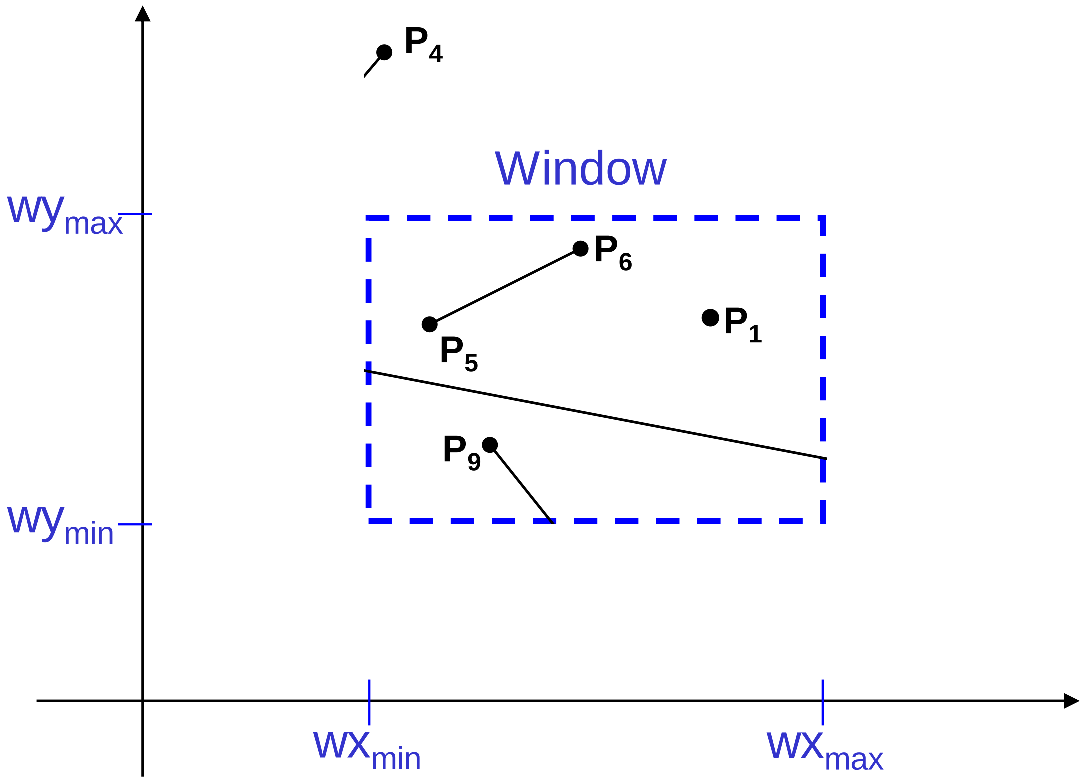
Windowing III

Because drawing things to a display takes time we *clip* everything outside the window



Clipping

For the image below consider which lines and points should be kept and which ones should be clipped

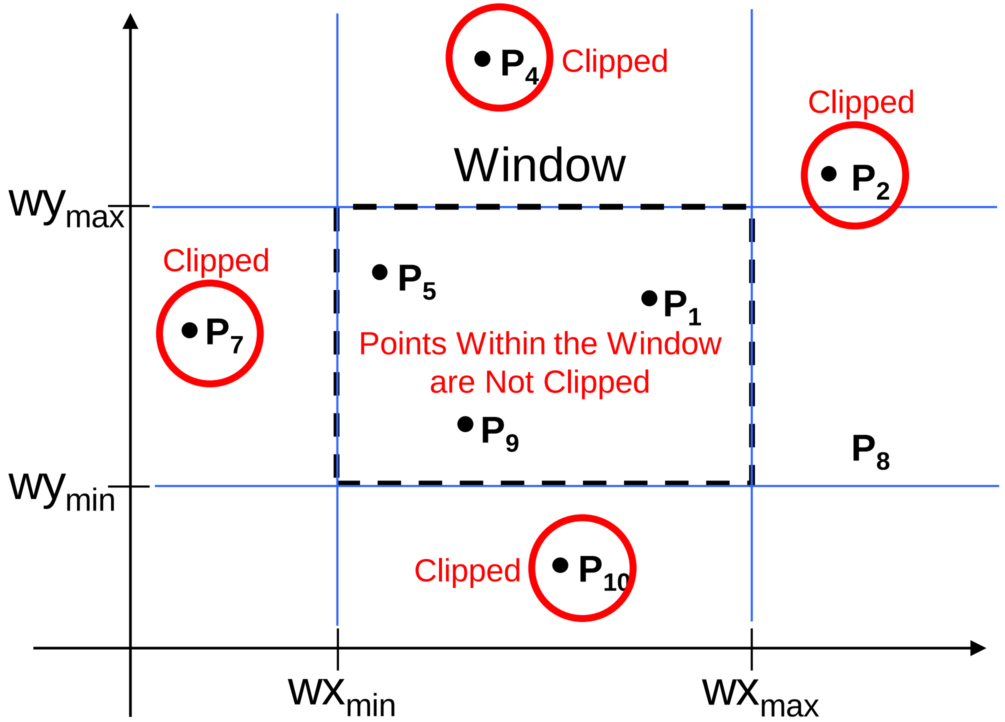


Point Clipping

Easy - a point (x,y) is not clipped if:

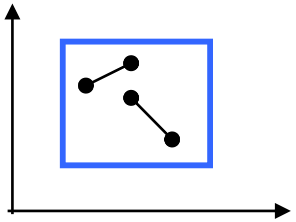
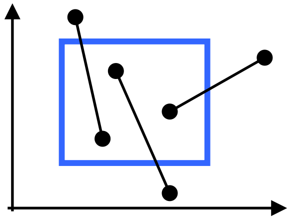
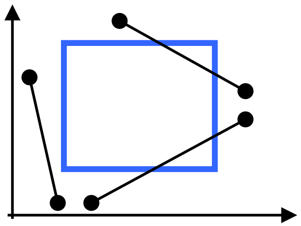
$$wx_{min} \leq x \leq wx_{max} \text{ AND } wy_{min} \leq y \leq wy_{max}$$

otherwise it is clipped



Line Clipping

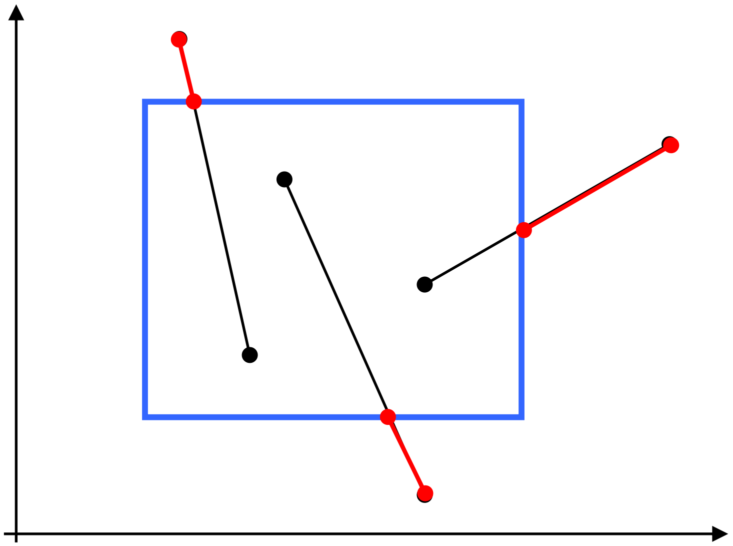
Harder - examine the end-points of each line to see if they are in the window or not

Situation	**Solution**	**Example**
Both end-points inside the window	Don't clip	
One end-point inside the window, one outside	Must clip	
Both end-points outside the window	Don't know!	

Brute Force Line Clipping

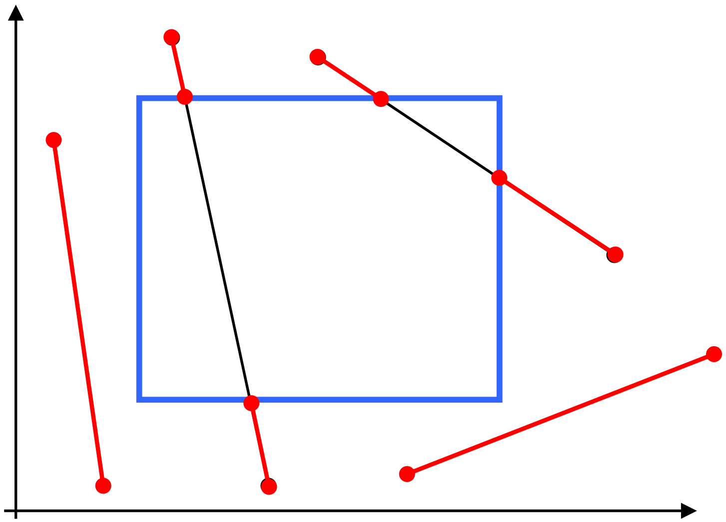
Brute force line clipping can be performed as follows:

- Don’t clip lines with both end-points within the window
- For lines with one end-point inside the window and one end-point outside, calculate the intersection point (using the equation of the line) and clip from this point out



Brute Force Line Clipping (cont…)

- For lines with both end-points outside the window test the line for intersection with all of the window boundaries, and clip appropriately



However, calculating line intersections is computationally expensive

Because a scene can contain so many lines, the brute force approach to clipping is much too slow

Cohen-Sutherland Clipping Algorithm

An efficient line clipping algorithm

The key advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated

Dr. Ivan E. Sutherland co-developed the Cohen-Sutherland clipping algorithm. Sutherland is a graphics giant and includes amongst his achievements the invention of the head mounted display.

Cohen is something of a mystery – can anybody find out who he was?

Cohen-Sutherland: World Division

World space is divided into regions based on the window boundaries

- Each region has a unique four bit region code
- Region codes indicate the position of the regions with respect to the window

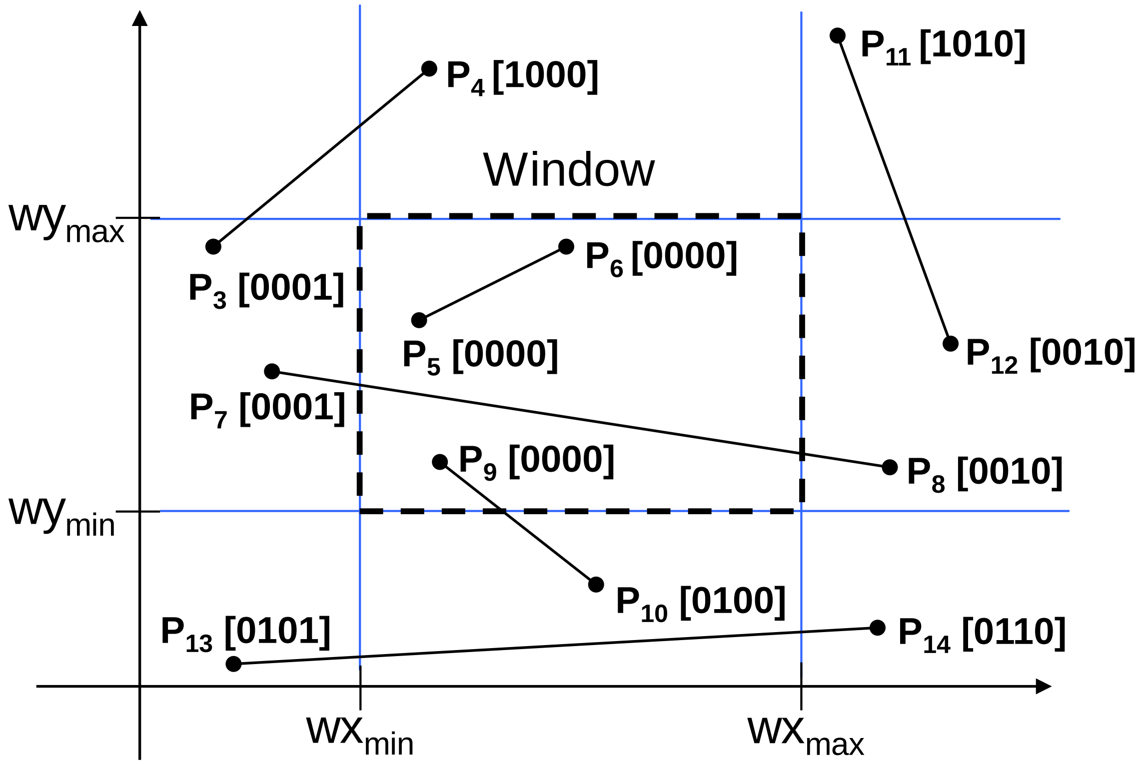
3	2	1	0
above	below	right	left

Region Code Legend

1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

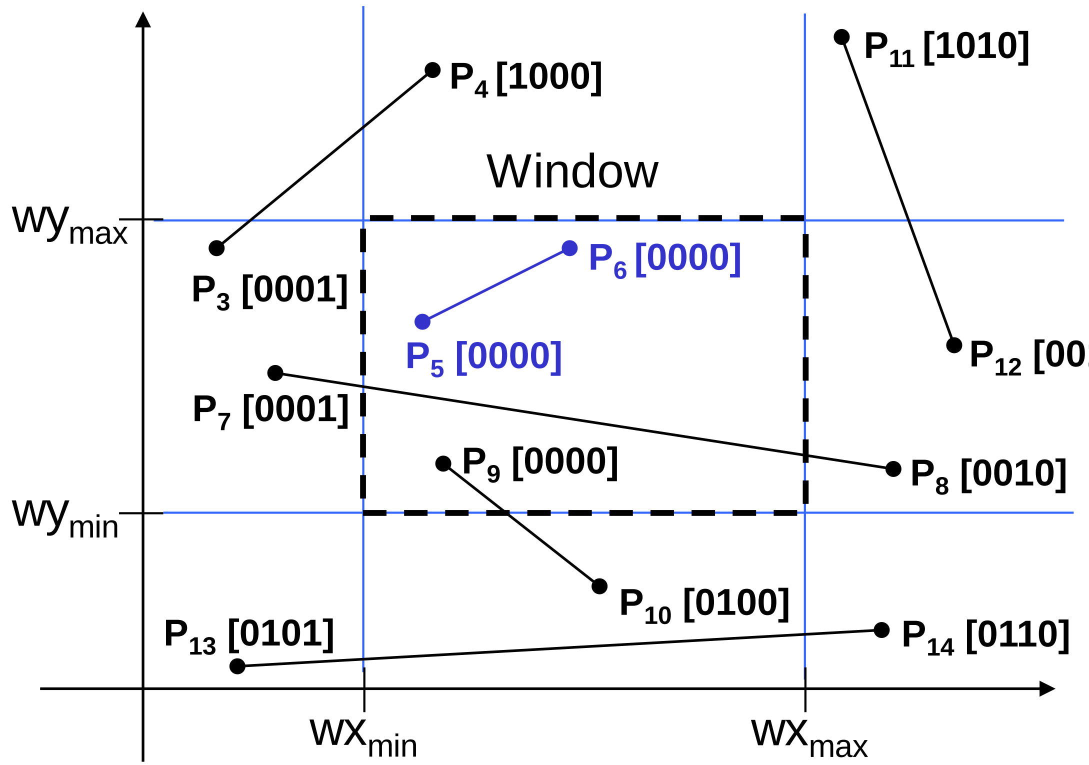
Cohen-Sutherland: Labelling

Every end-point is labelled with the appropriate region code



Cohen-Sutherland: Lines In The Window

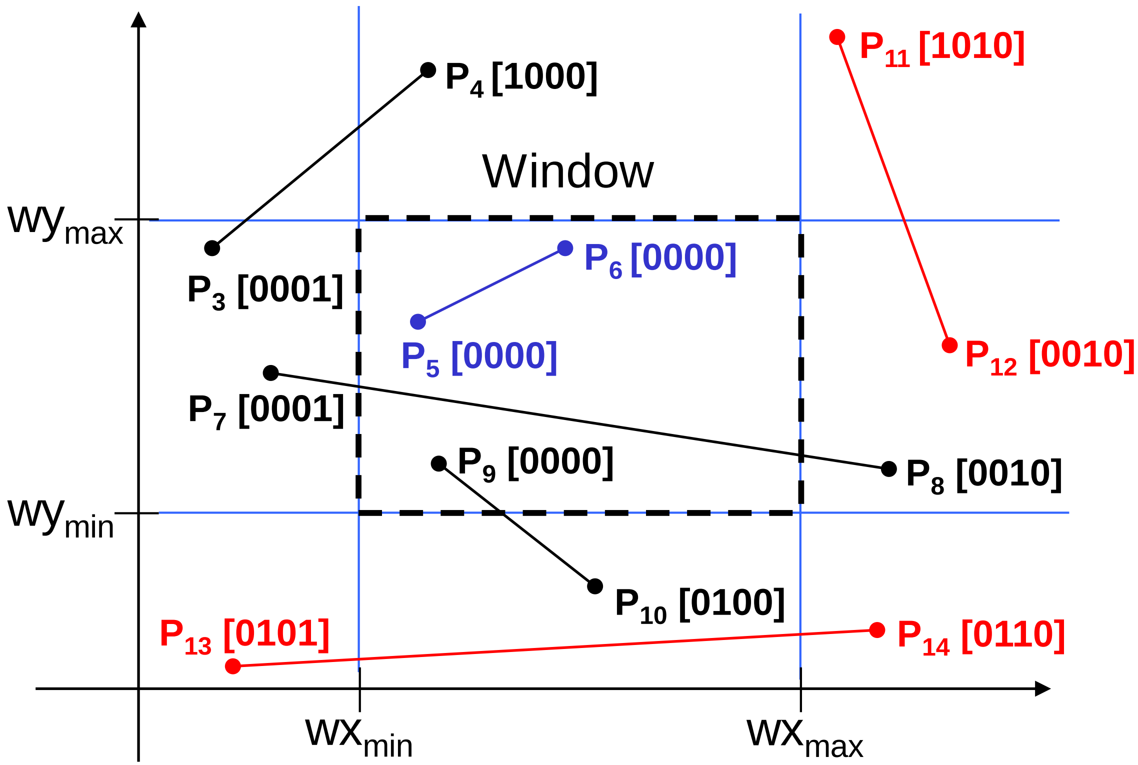
Lines completely contained within the window boundaries have region code [0000] for both end-points so are not clipped



Cohen-Sutherland: Lines Outside The Window

Any lines with a common set bit in the region codes of both end-points can be clipped

– The AND operation can efficiently check this



Cohen-Sutherland: Other Lines

Lines that cannot be identified as completely inside or outside the window may or may not cross the window interior

These lines are processed as follows:

- Compare an end-point outside the window to a boundary (choose any order in which to consider boundaries e.g. left, right, bottom, top) and determine how much can be discarded
- If the remainder of the line is entirely inside or outside the window, retain it or clip it respectively

Cohen-Sutherland: Other Lines (cont…)

- Otherwise, compare the remainder of the line against the other window boundaries
- Continue until the line is either discarded or a segment inside the window is found

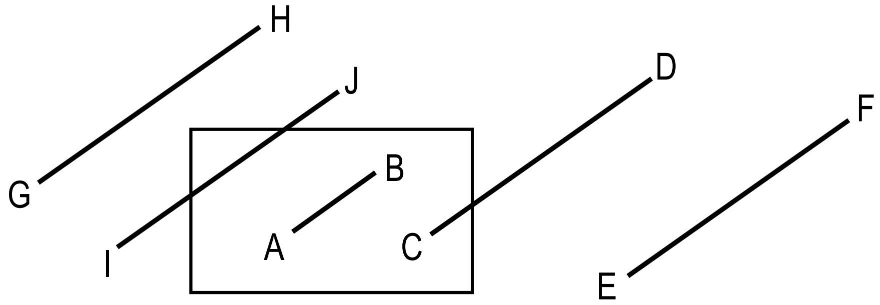
We can use the region codes to determine which window boundaries should be considered for intersection

- To check if a line crosses a particular boundary we compare the appropriate bits in the region codes of its end-points
- If one of these is a 1 and the other is a 0 then the line crosses the boundary

Cohen-Sutherland: (cont…)

Four Cases

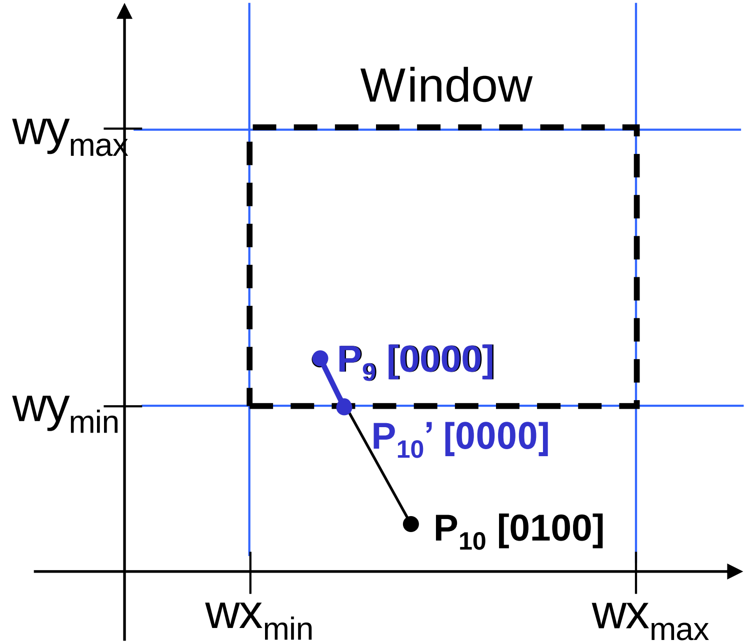
- outcode1 = outcode2 = 0000 --> Accept all
- outcode1 && outcode2 != 0000 --> Discard
- outcode1 != 0000 , outcode2 = 0000 (Vice versa) --> Shorten
- outcode1 && outcode2 = 0000 --> Discard or Shorten



Cohen-Sutherland Examples

Consider the line P_9 to P_{10} below

- Start at P_{10}
- From the region codes of the two end-points we know the line doesn't cross the left or right boundary



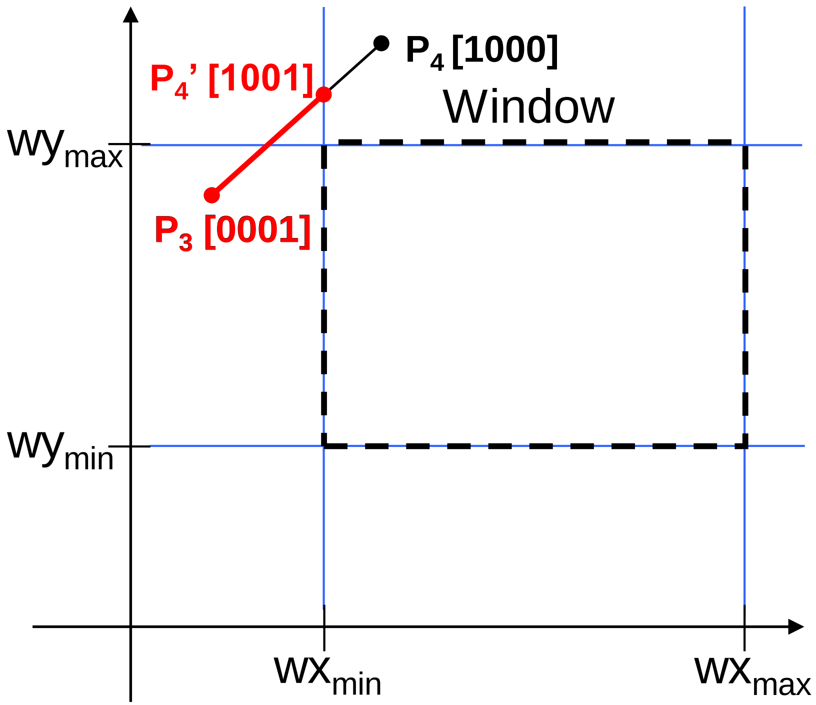
- Calculate the intersection of the line with the bottom boundary to generate point P_{10}'
- The line P_9 to P_{10}' is completely inside the window so is retained

Cohen-Sutherland Examples (cont…)

Consider the line P_3 to P_4 below

– Start at P_4

– From the region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_4'

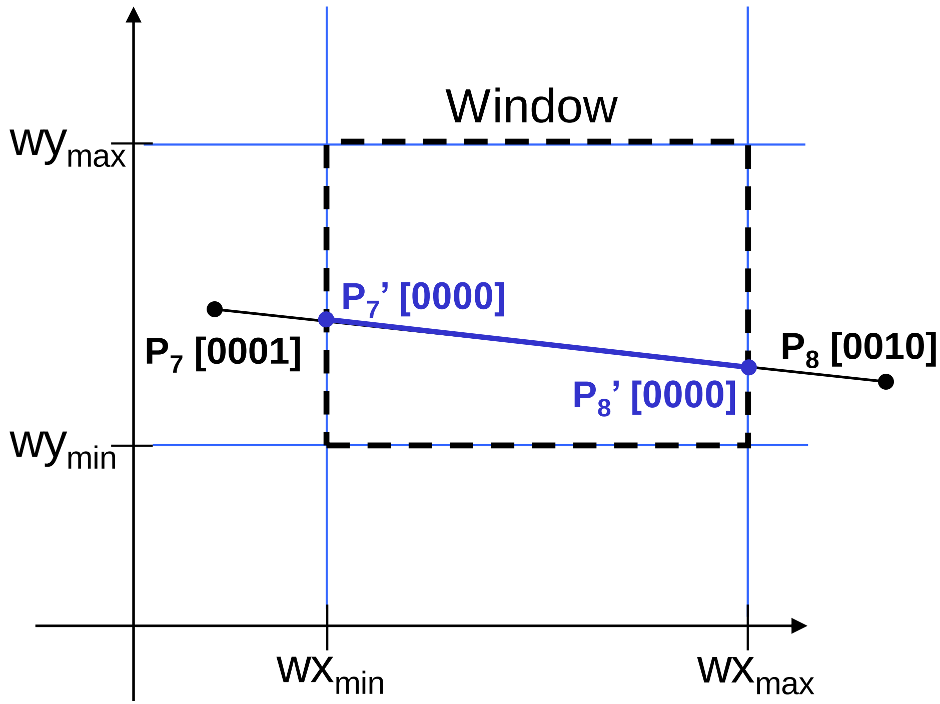


– The line P_3 to P_4' is completely outside the window so is clipped

Cohen-Sutherland Examples (cont…)

Consider the line P_7 to P_8 below

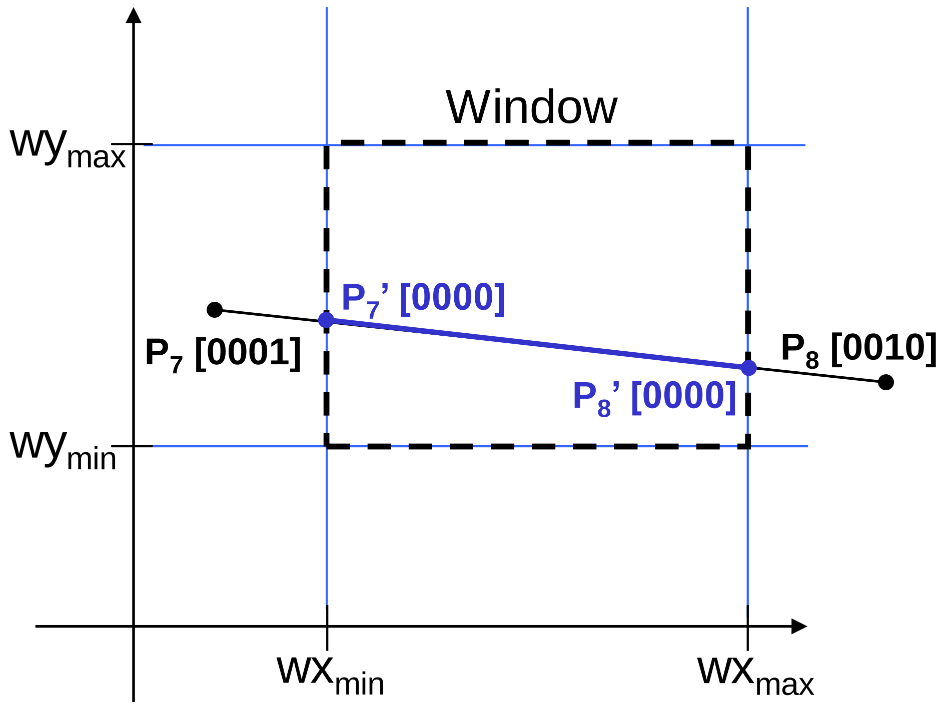
- Start at P_7
- From the two region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_7'



Cohen-Sutherland Examples (cont…)

Consider the line P_7’ to P_8

- Start at P_8
- Calculate the intersection with the right boundary to generate P_8’
- P_7’ to P_8’ is inside the window so is retained



Calculating Line Intersections

Intersection points with the window boundaries are calculated using the line-equation parameters

– Consider a line with the end-points (x_1, y_1) and (x_2, y_2)

– The y-coordinate of an intersection with a vertical window boundary can be calculated using:

$$y = y_1 + m\,(x_{boundary} - x_1)$$

where $x_{boundary}$ can be set to either wx_{min} or wx_{max}

Calculating Line Intersections (cont…)

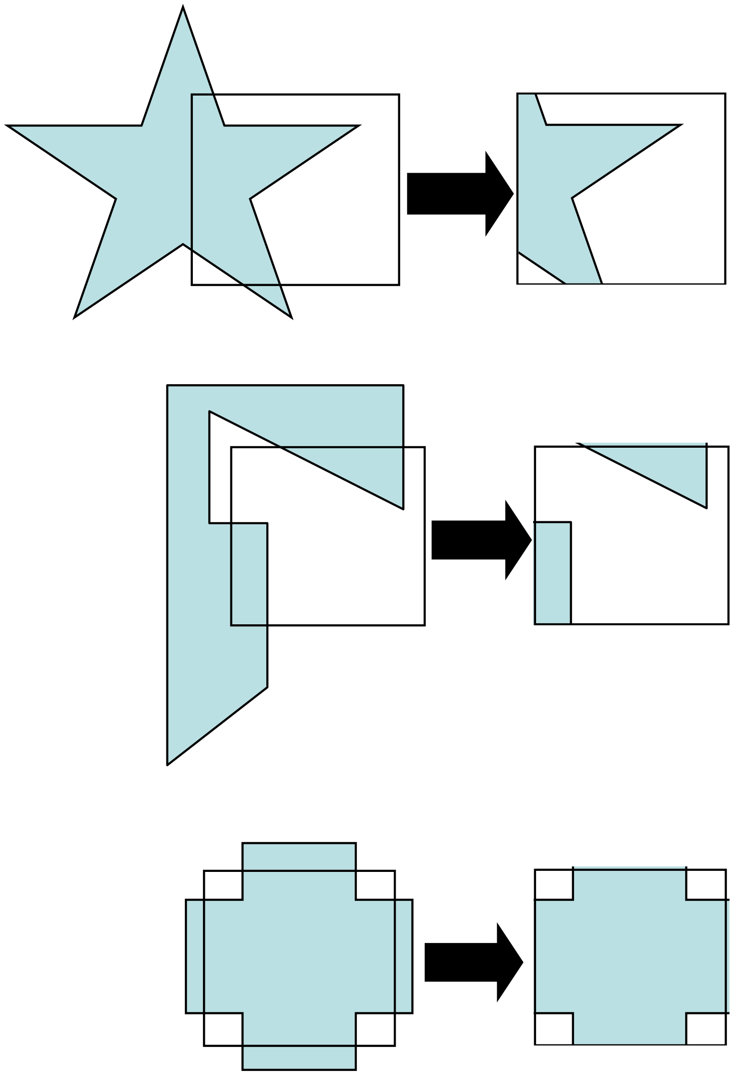
– The x-coordinate of an intersection with a horizontal window boundary can be calculated using:

$$x = x_1 + (y_{\text{boundary}} - y_1) / m$$

where $y_{boundary}$ can be set to either wy_{min} or wy_{max}

– m is the slope of the line in question and can be calculated as $m = (y_2 - y_1) / (x_2 - x_1)$

Area Clipping



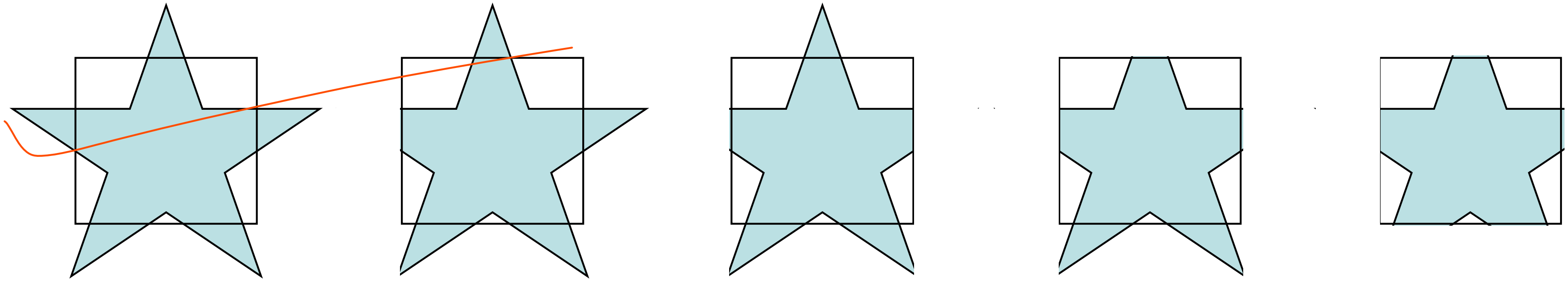
- Similarly to lines, areas must be clipped to a window boundary
- Consideration must be taken as to which portions of the area must be clipped

Sutherland-Hodgman Area Clipping Algorithm

A technique for clipping areas developed by Sutherland & Hodgman

Put simply the polygon is clipped by comparing it against each boundary in turn

Sutherland turns up again. This time with Gary Hodgman with whom he worked at the first ever graphics company Evans & Sutherland



Original Area   Clip Left   Clip Right   Clip Top   Clip Bottom

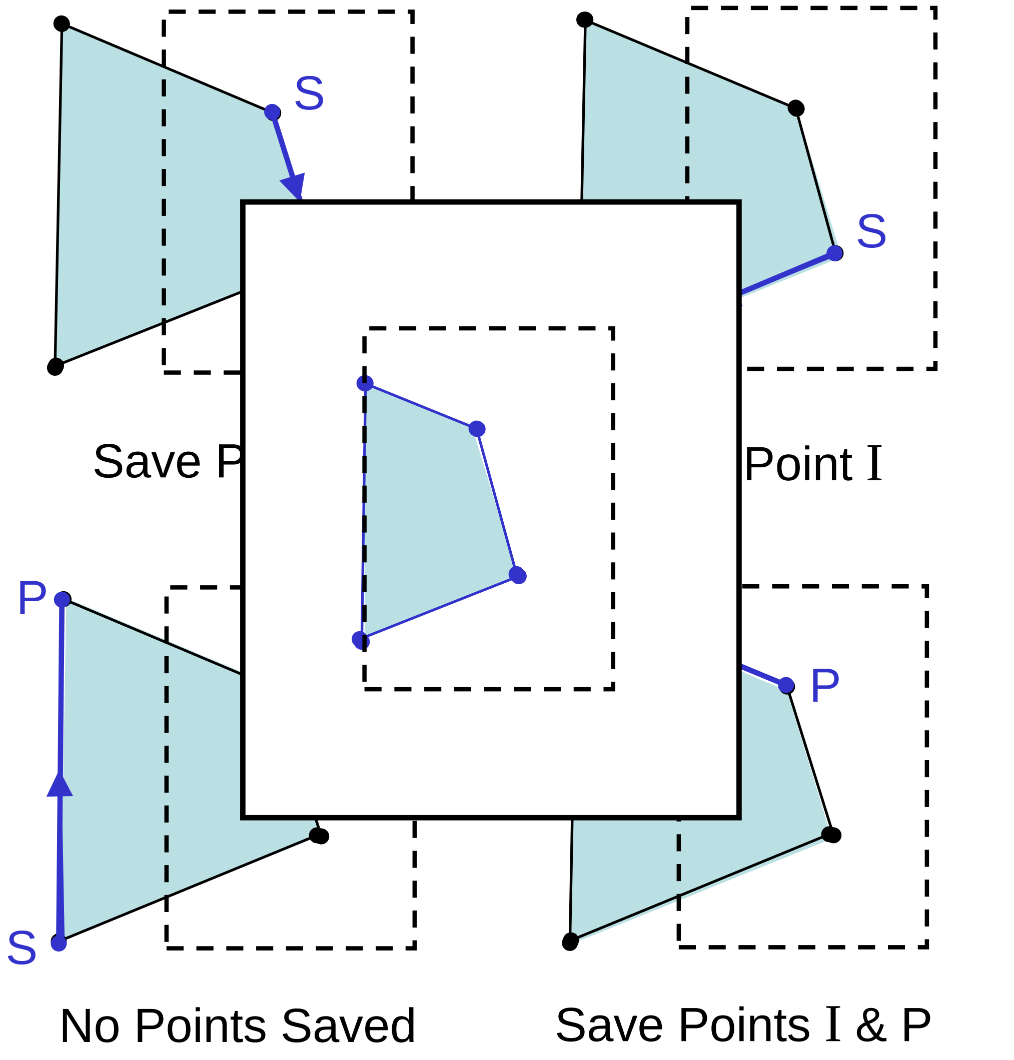
Sutherland-Hodgman Area Clipping Algorithm (cont…)

To clip an area against an individual boundary:

- Consider each vertex in turn against the boundary
- Vertices inside the boundary are saved for clipping against the next boundary
- Vertices outside the boundary are clipped
- If we proceed from a point inside the boundary to one outside, the intersection of the line with the boundary is saved
- If we cross from the outside to the inside intersection point and the vertex are saved

Sutherland-Hodgman Example

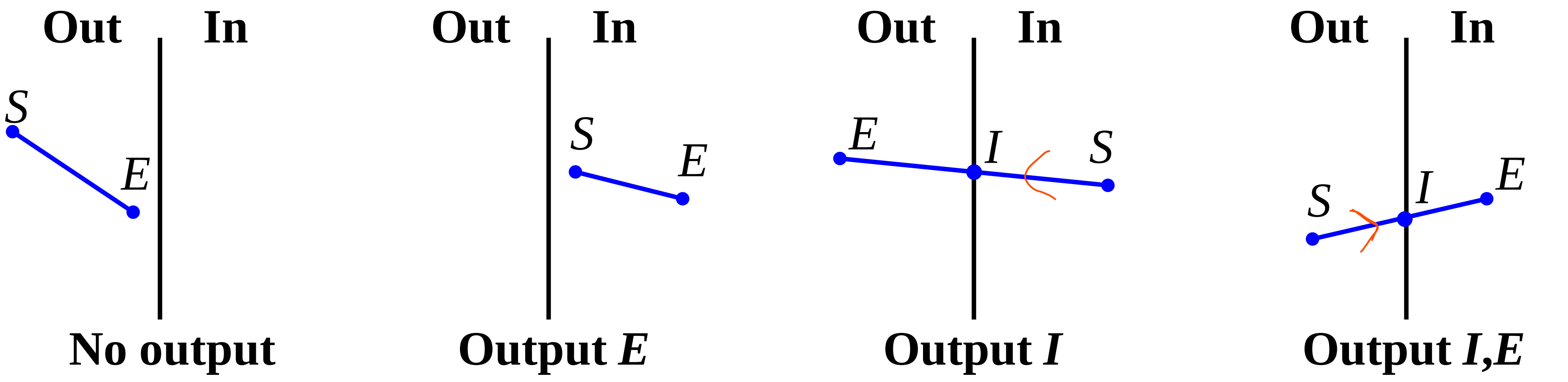
- Each example shows the point being processed (P) and the previous point (S)
- Saved points define area clipped to the boundary in question



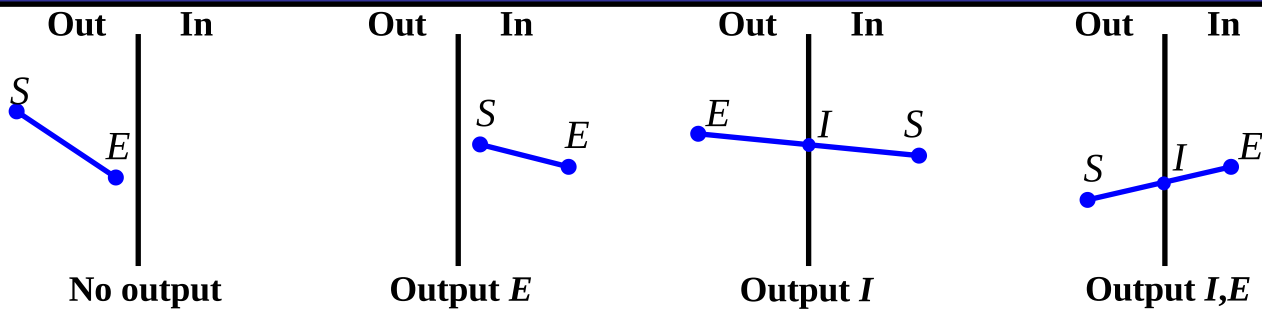
Sutherland-Hodgman Algorithm (Example)

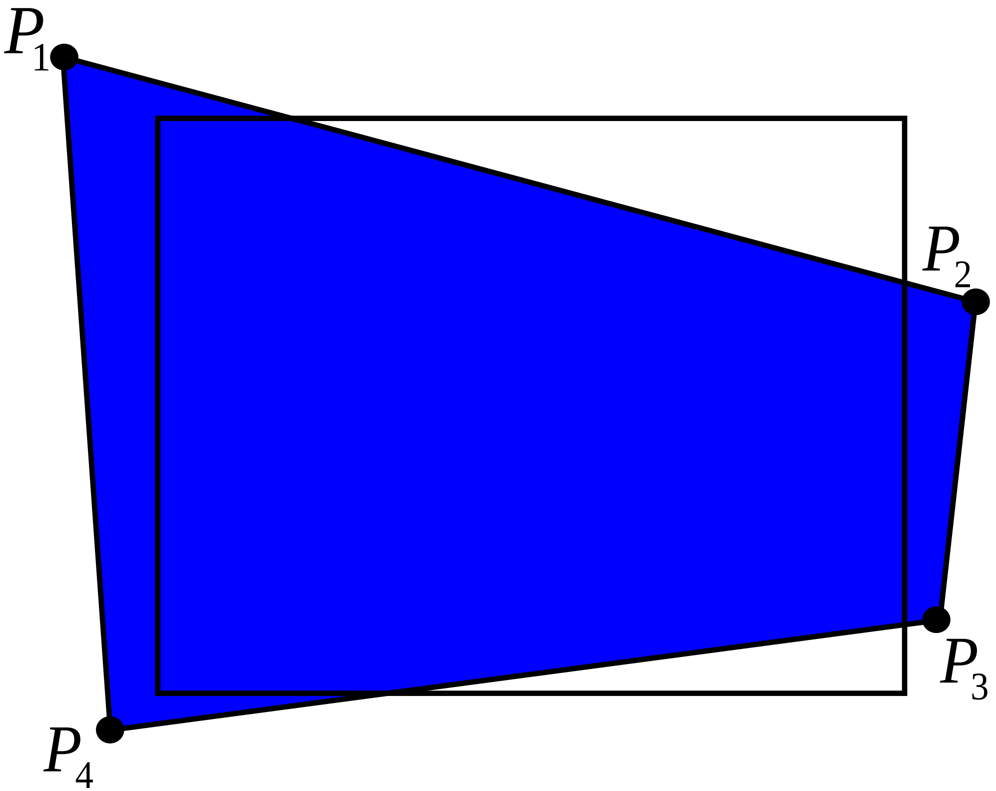
For each boundary, clip all edges to that boundary

Four clipping boundaries: L, R, B, T

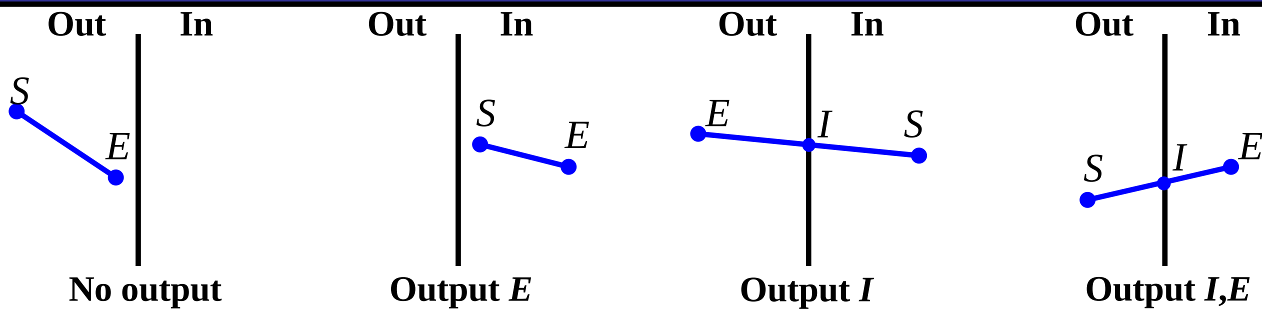


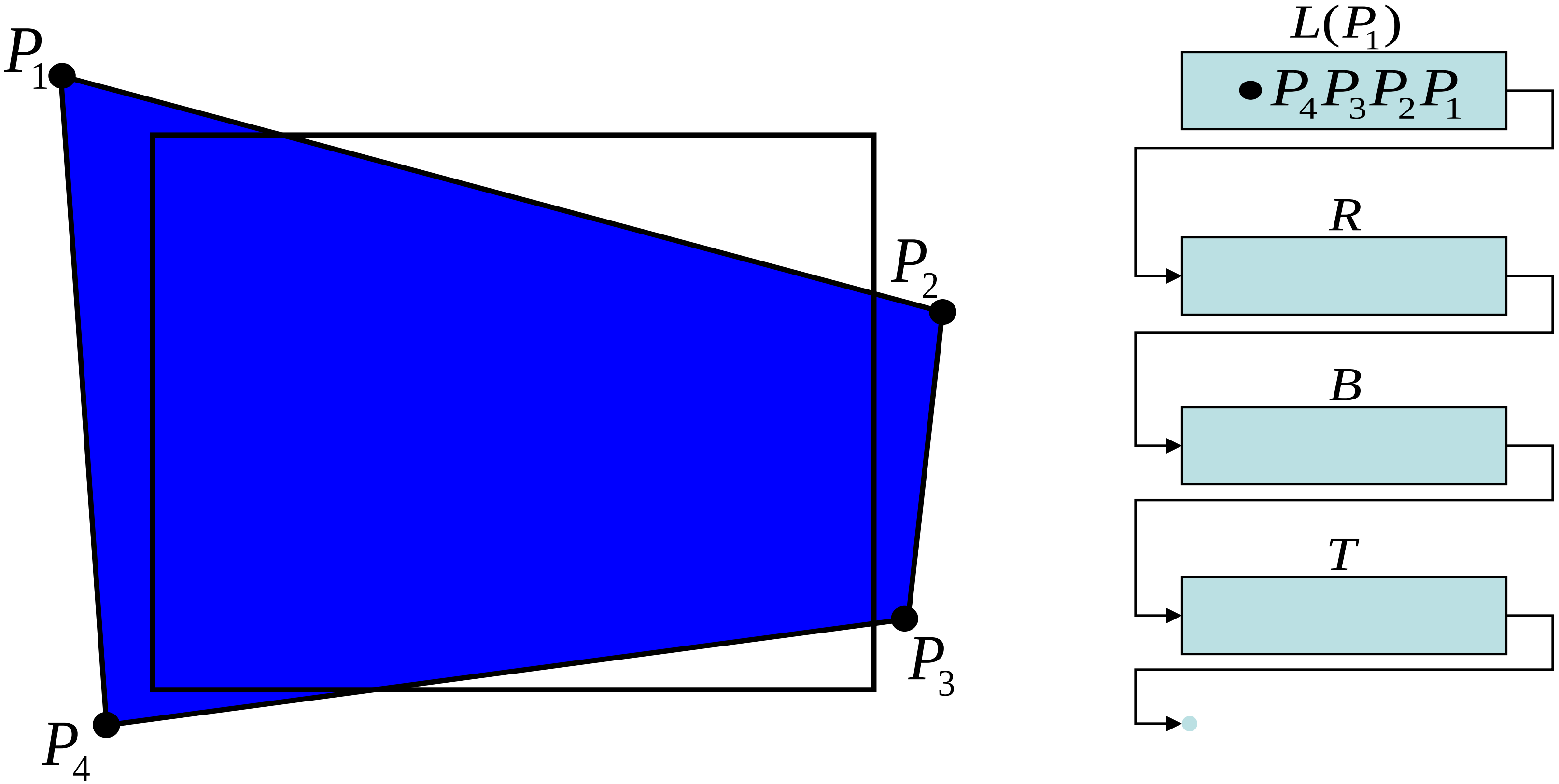
Sutherland-Hodgman Algorithm



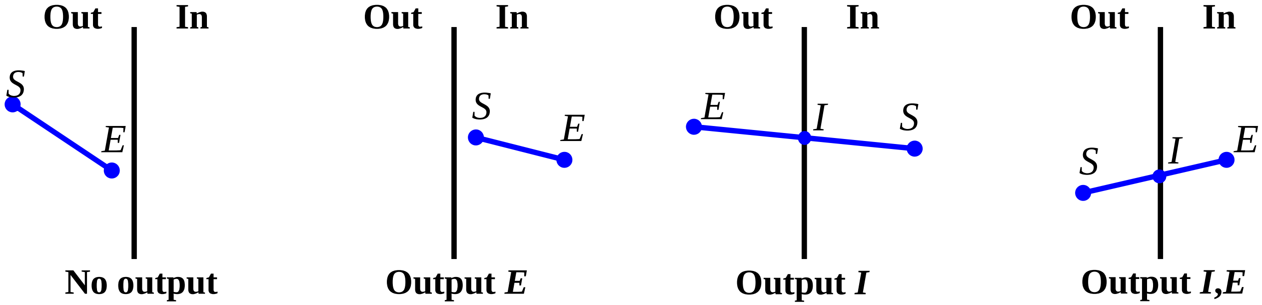


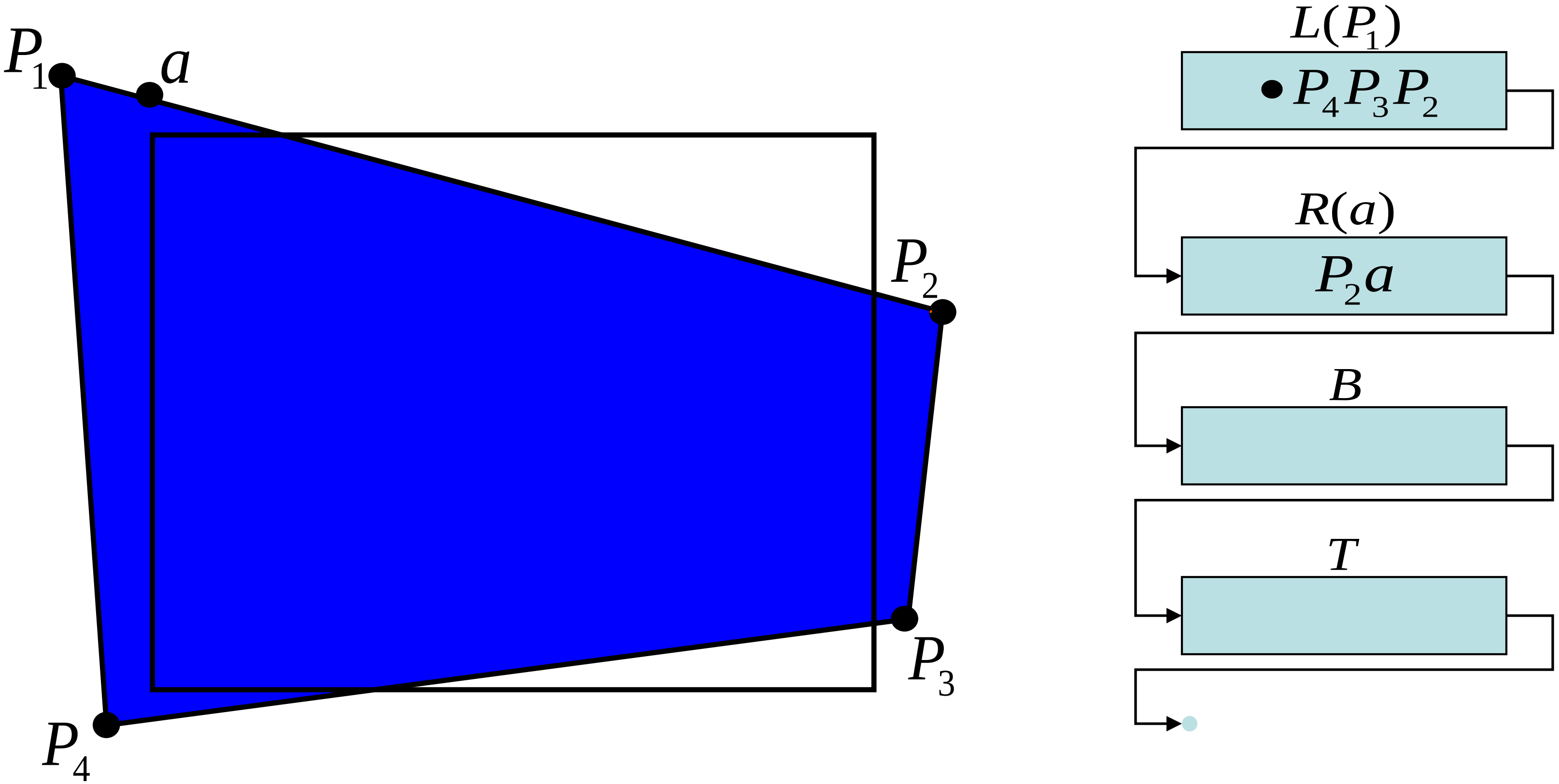
Sutherland-Hodgman Algorithm



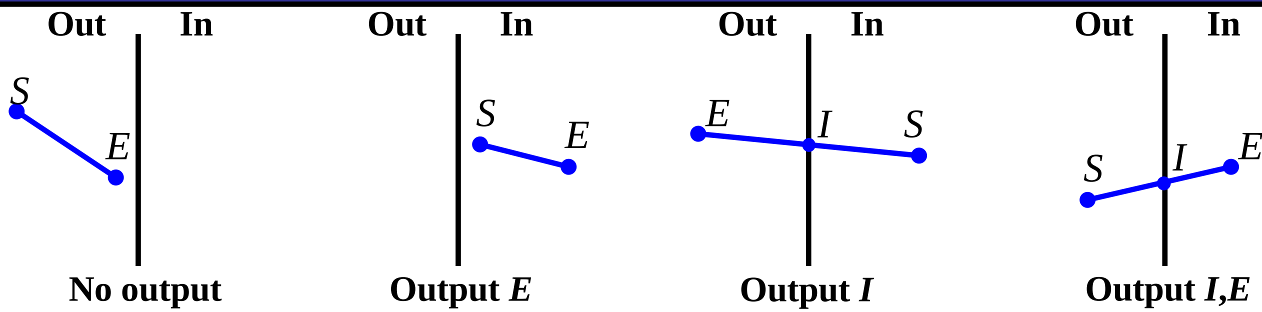


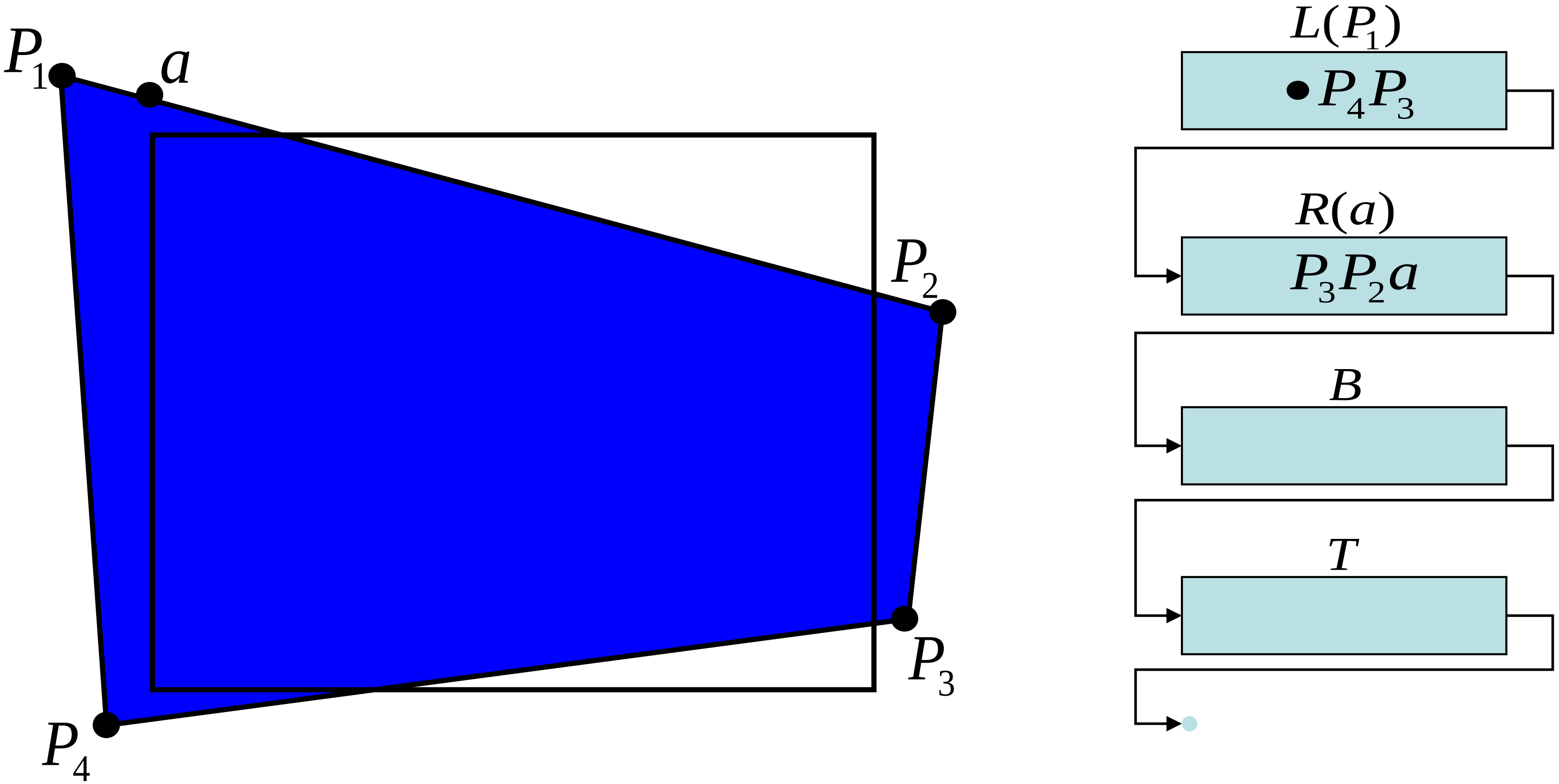
Sutherland-Hodgman Algorithm



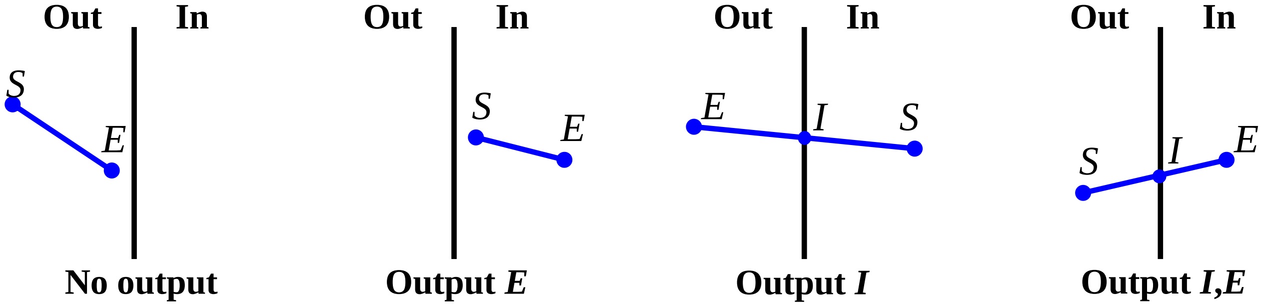


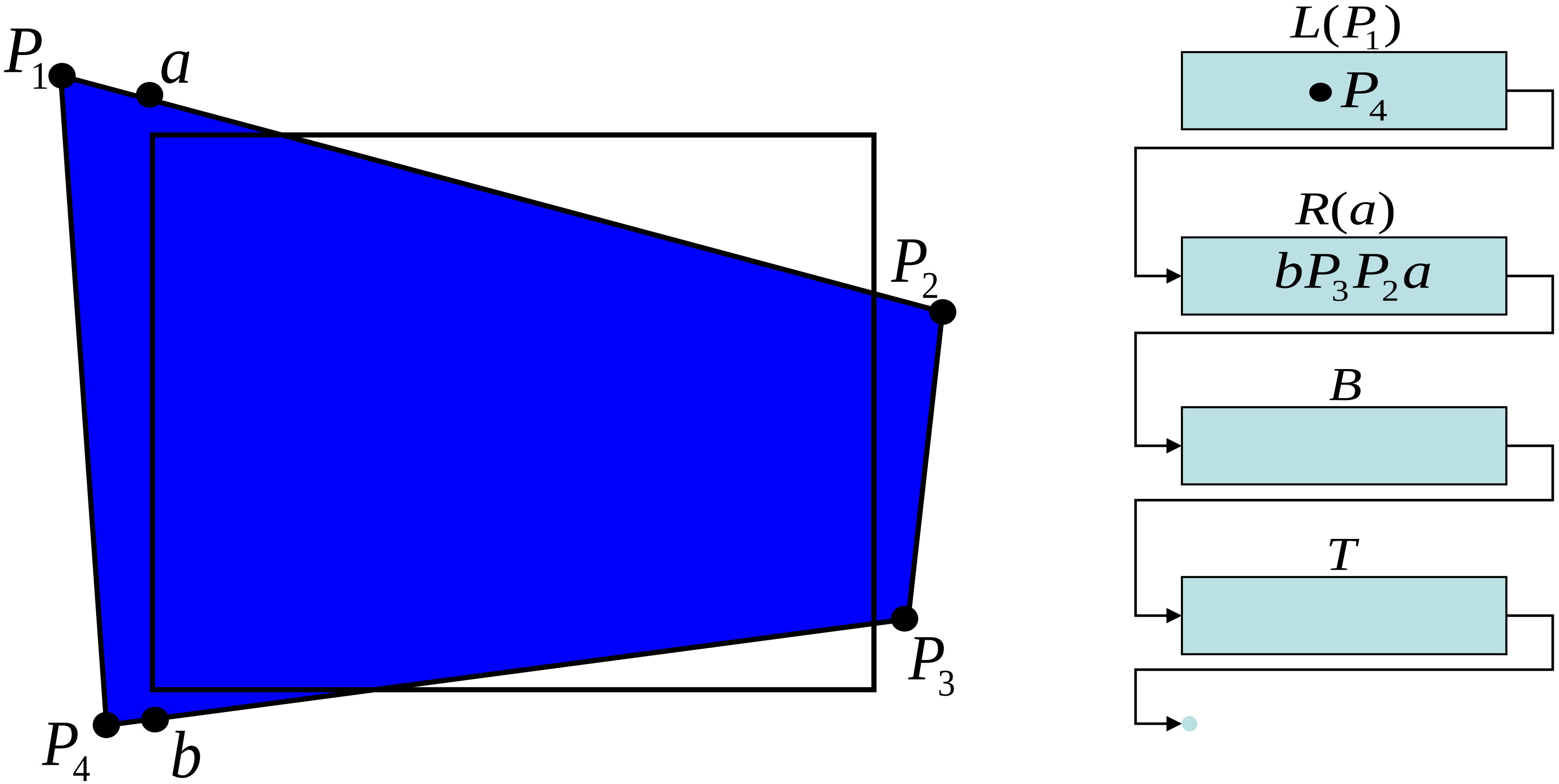
Sutherland-Hodgman Algorithm



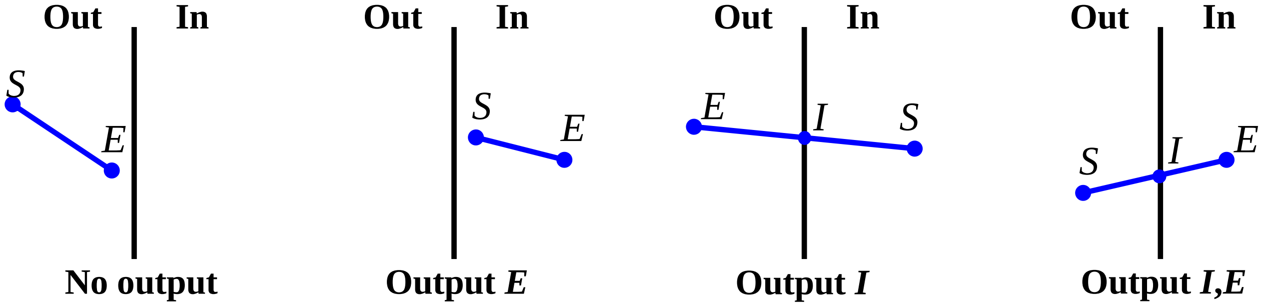


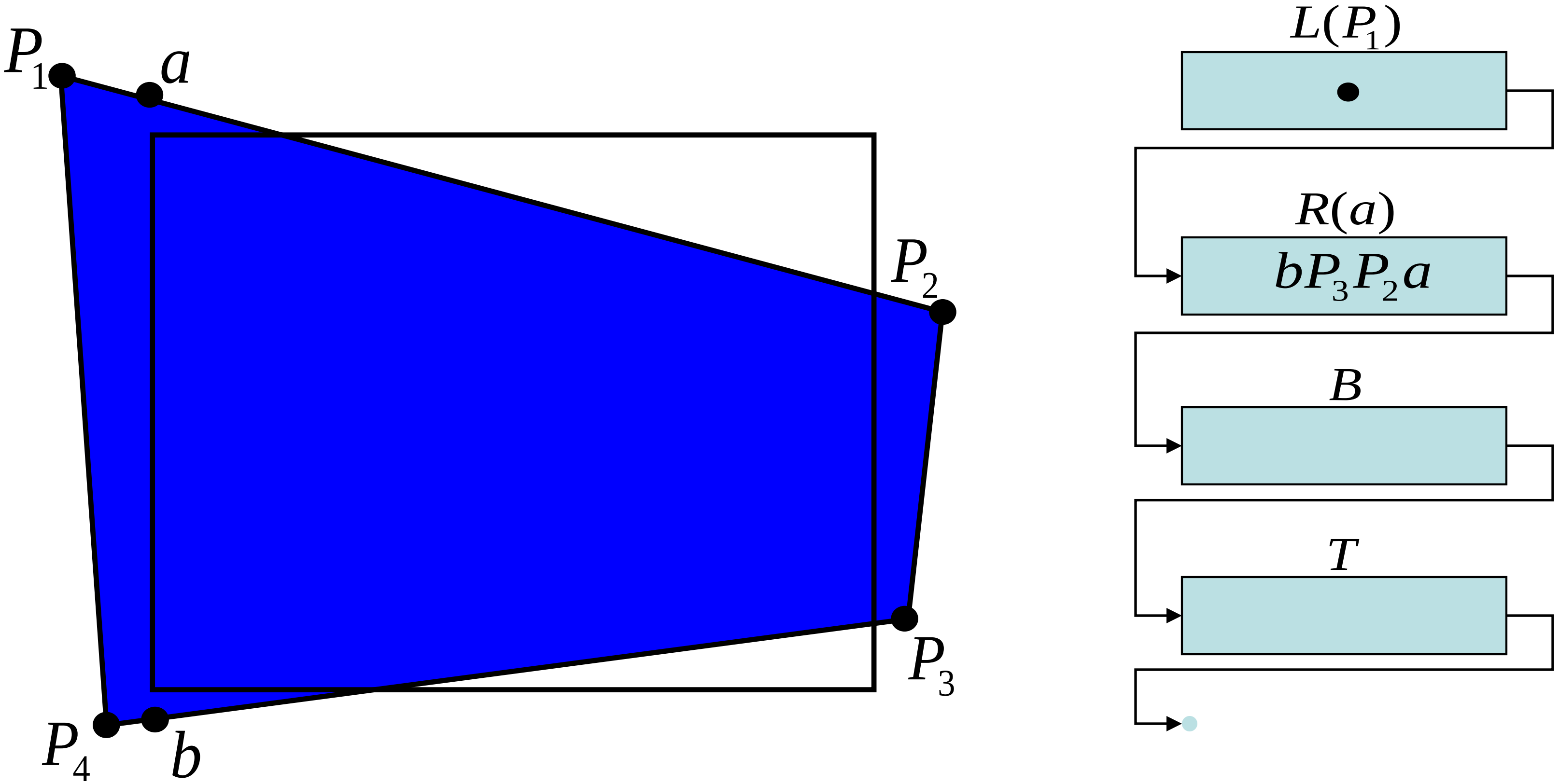
Sutherland-Hodgman Algorithm



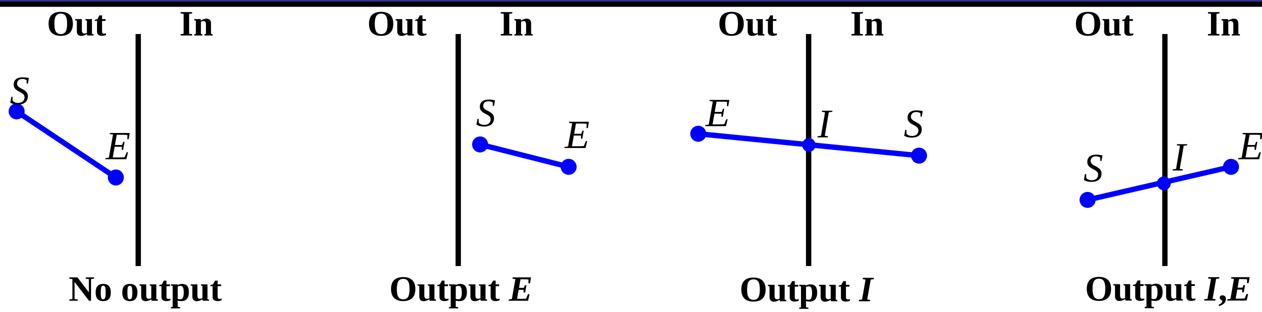


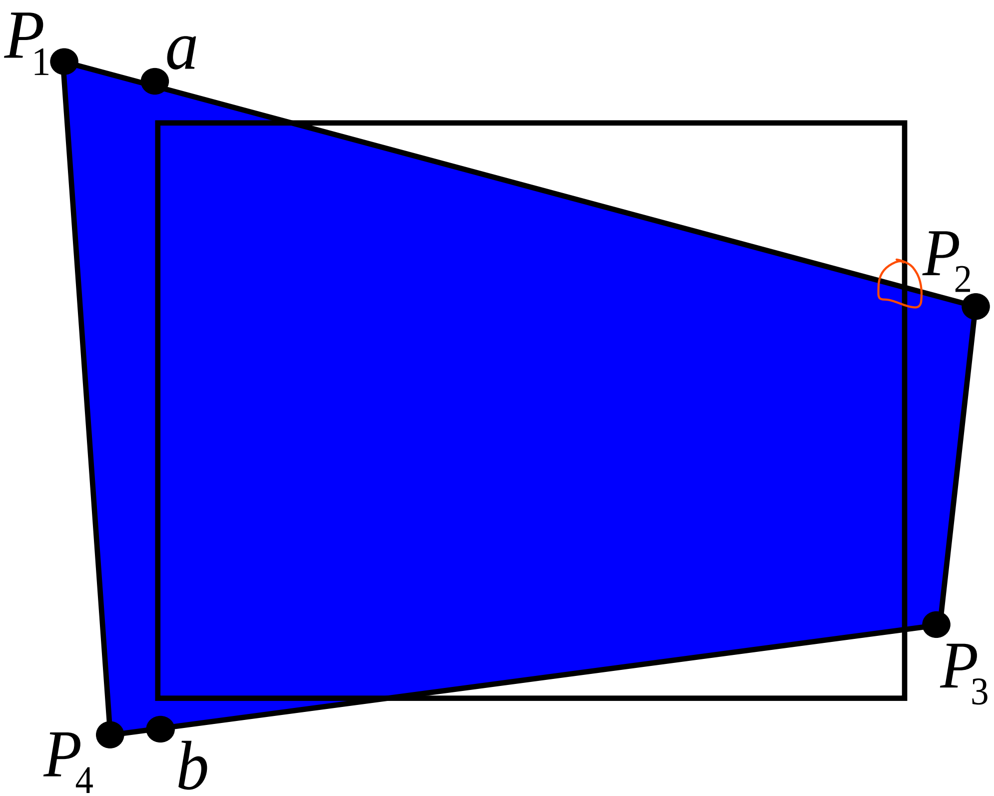
Sutherland-Hodgman Algorithm

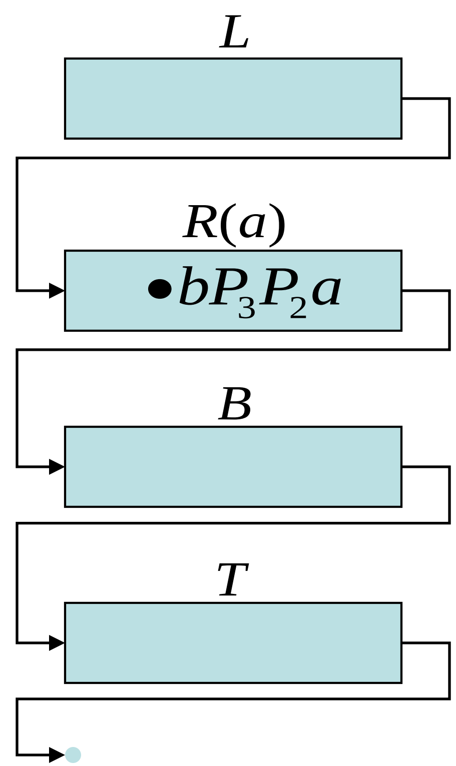




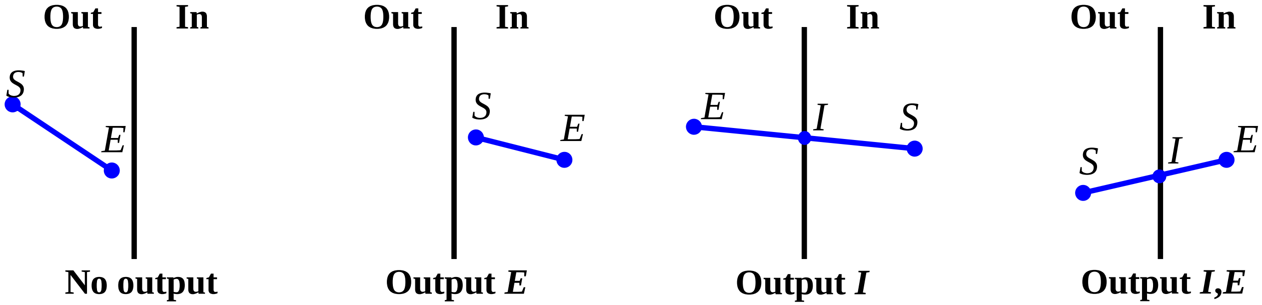
Sutherland-Hodgman Algorithm

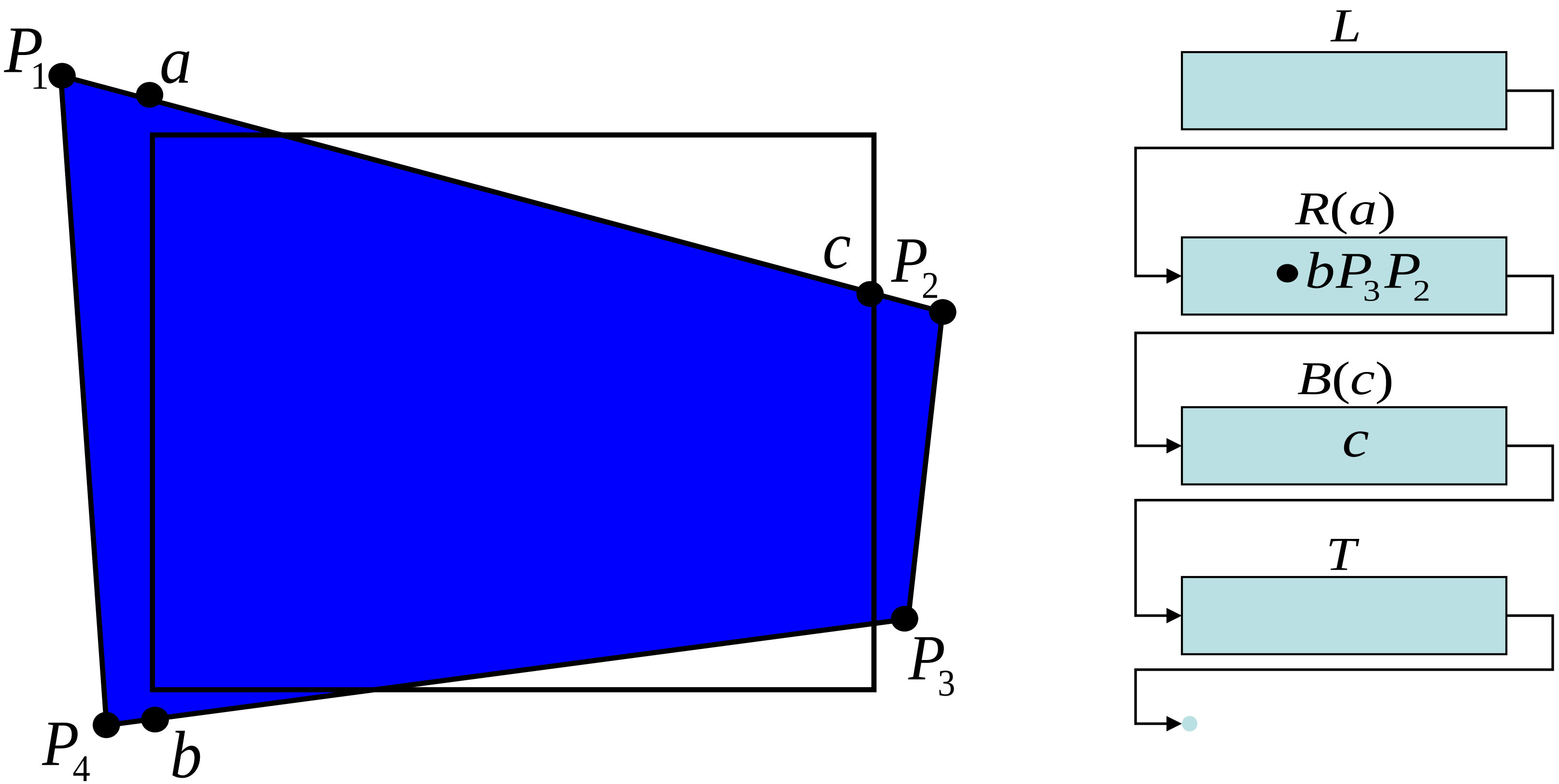




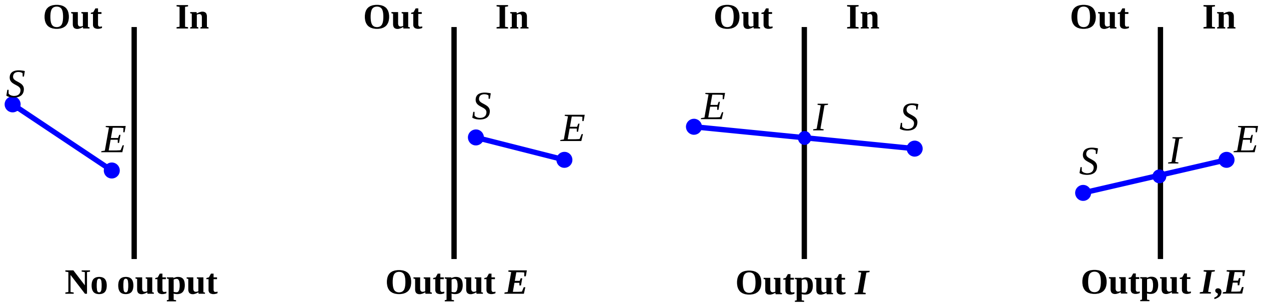


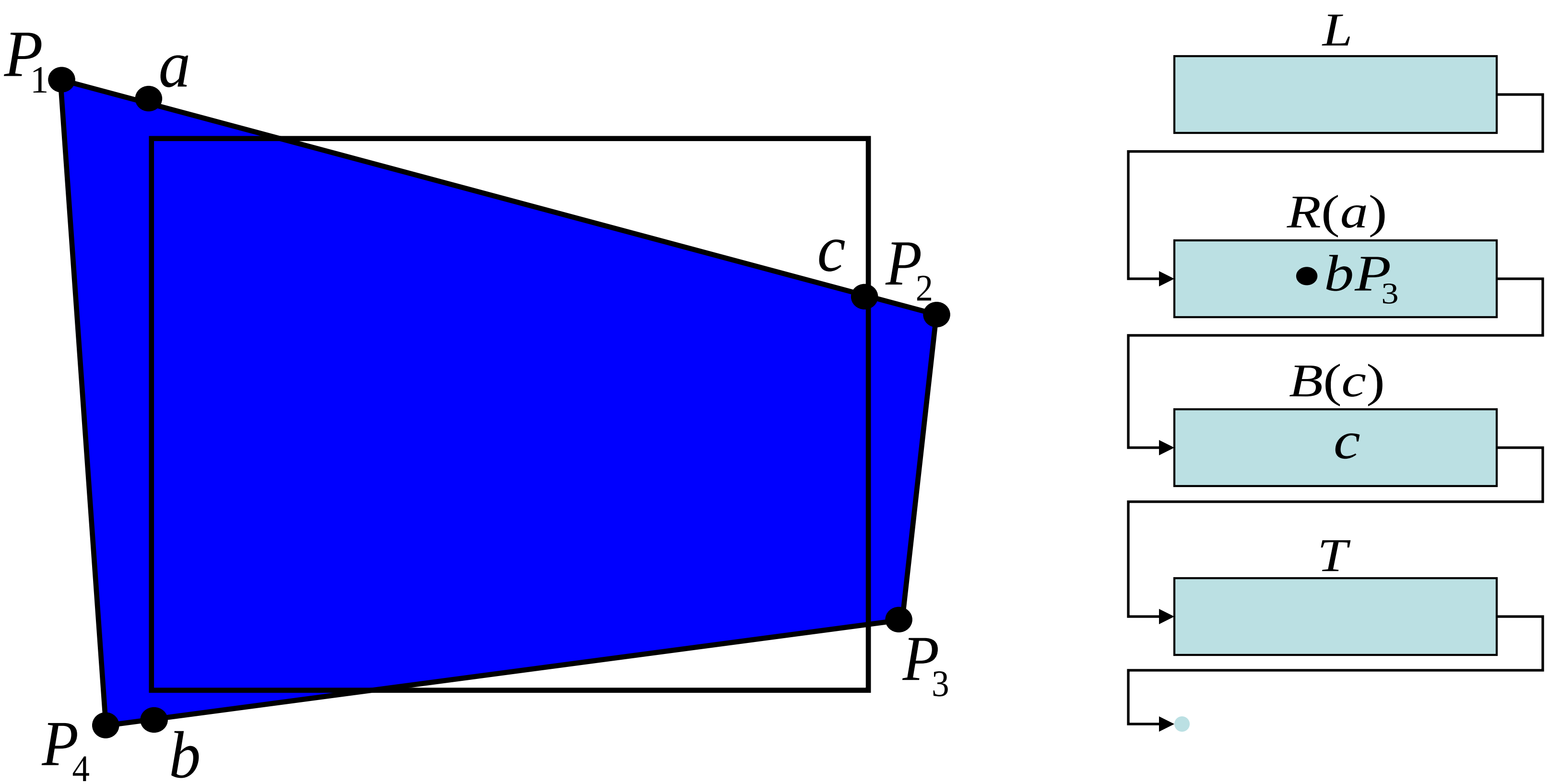
Sutherland-Hodgman Algorithm



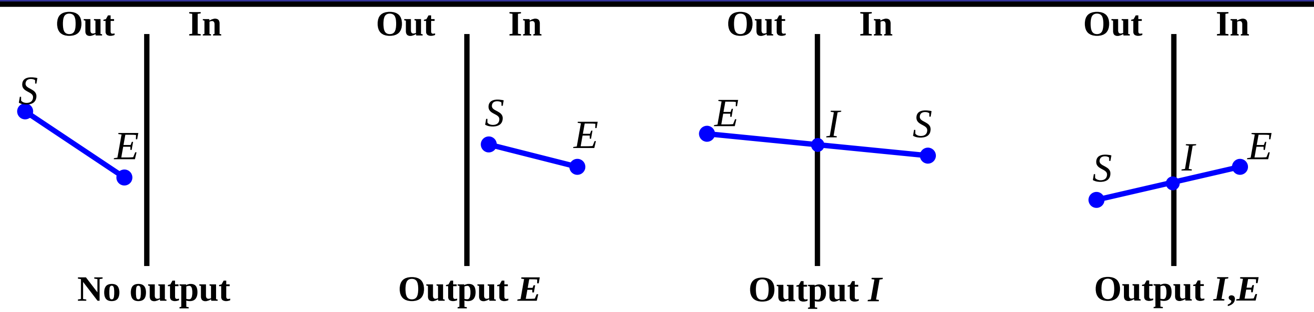


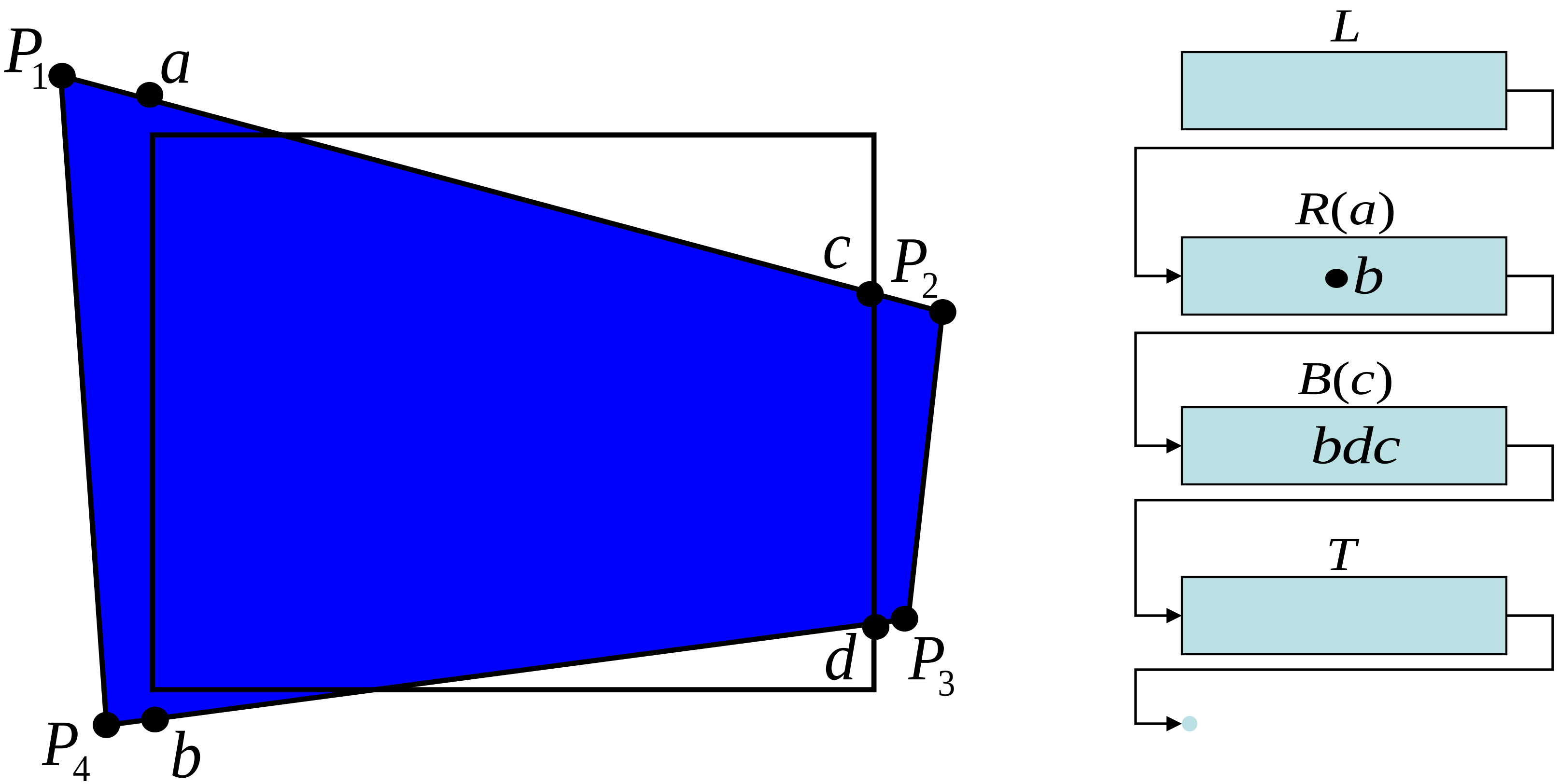
Sutherland-Hodgman Algorithm



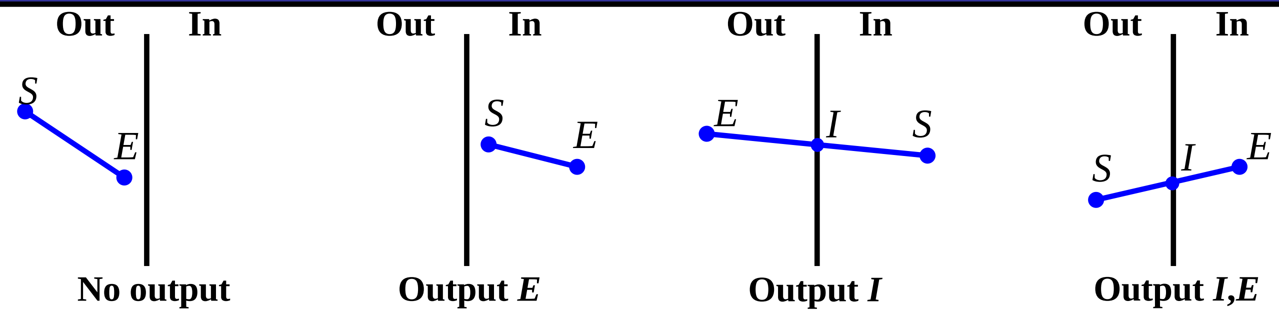


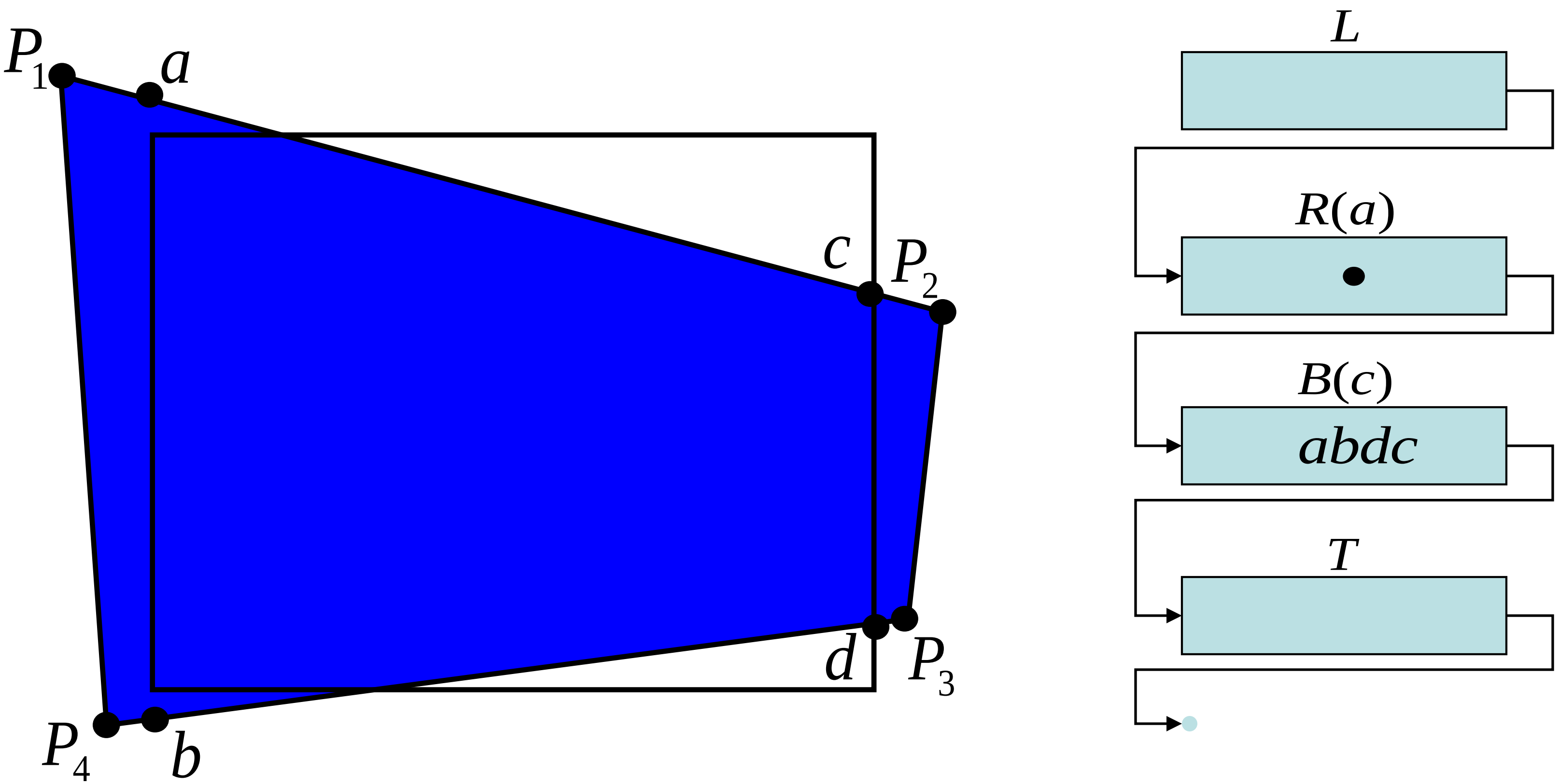
Sutherland-Hodgman Algorithm



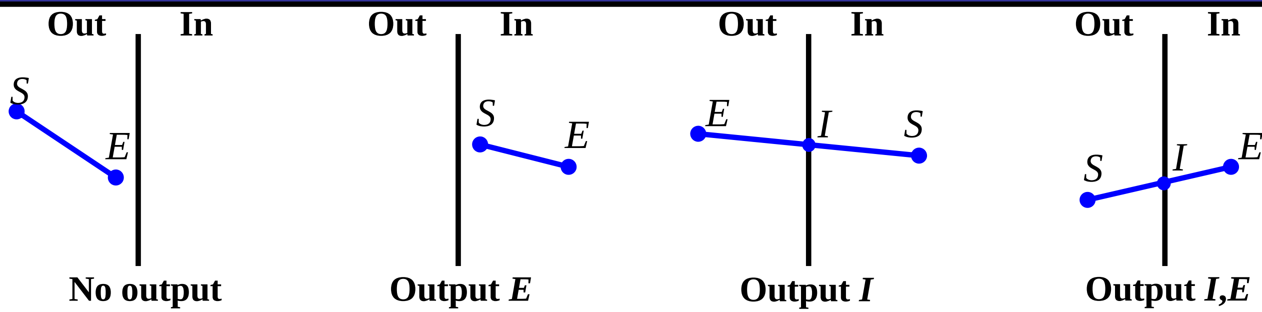


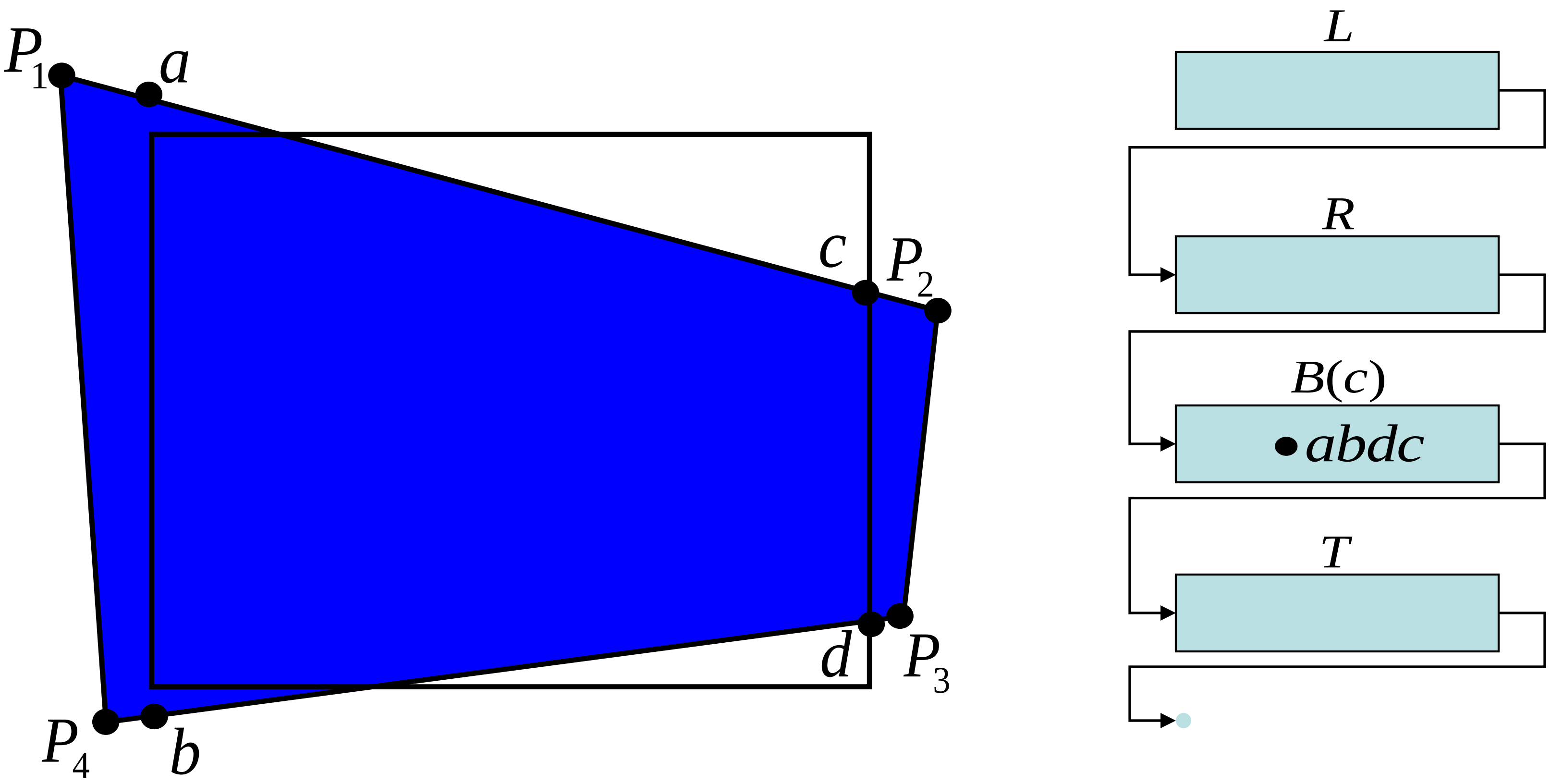
Sutherland-Hodgman Algorithm



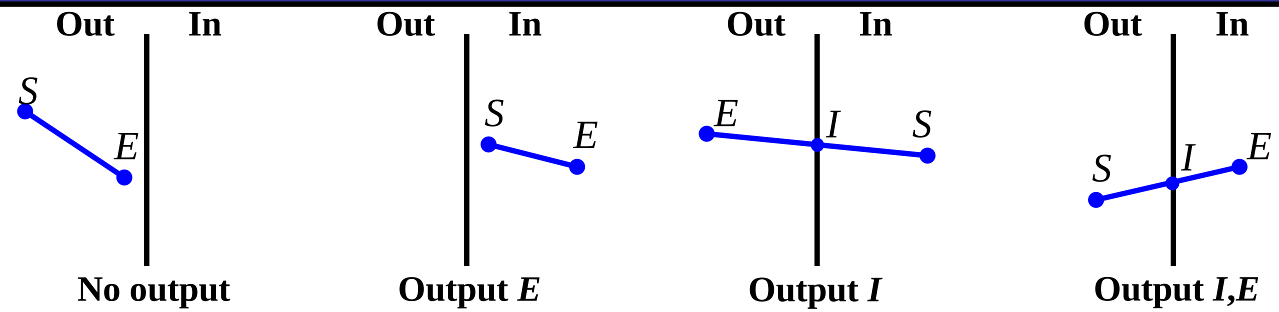


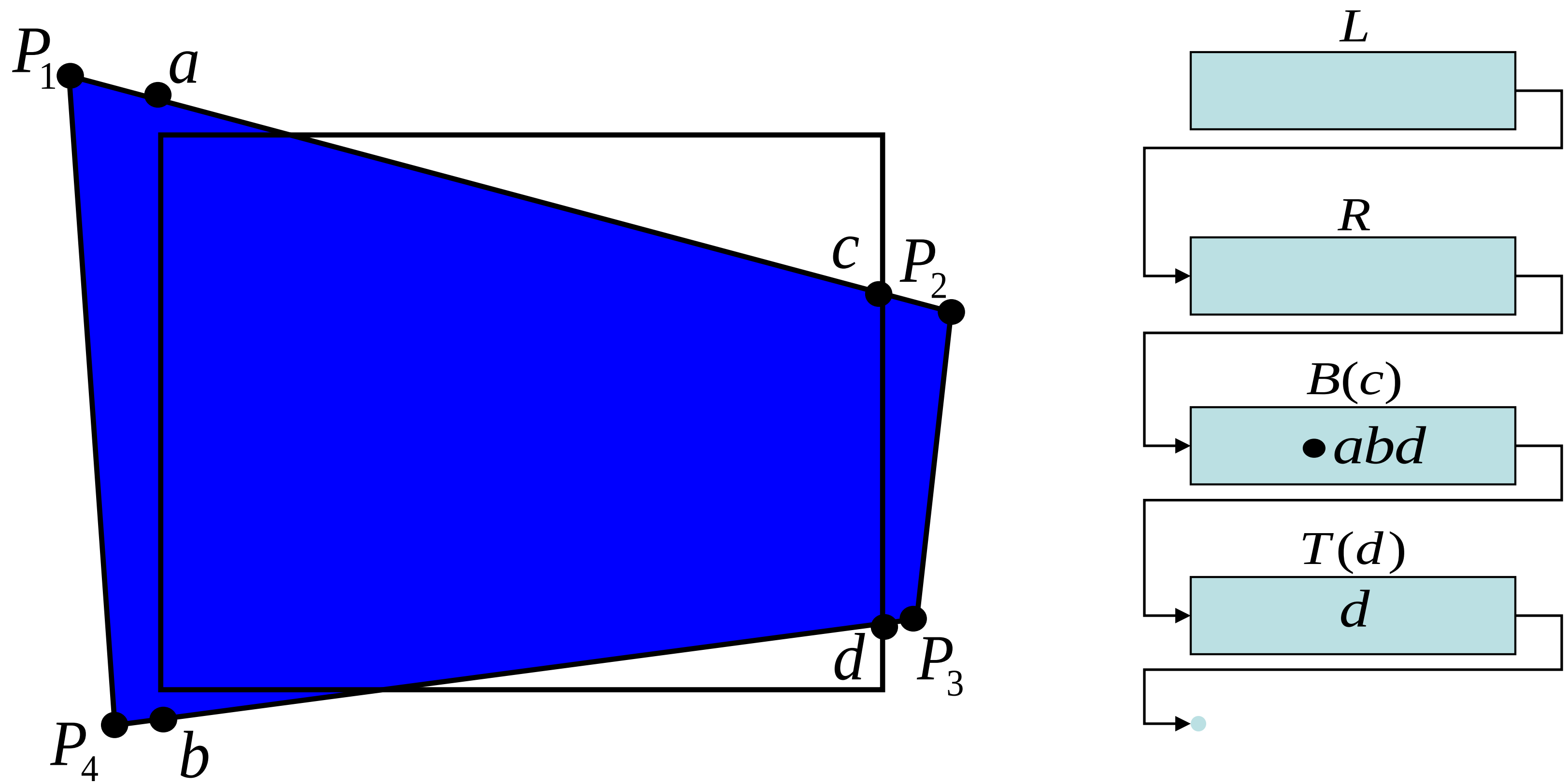
Sutherland-Hodgman Algorithm



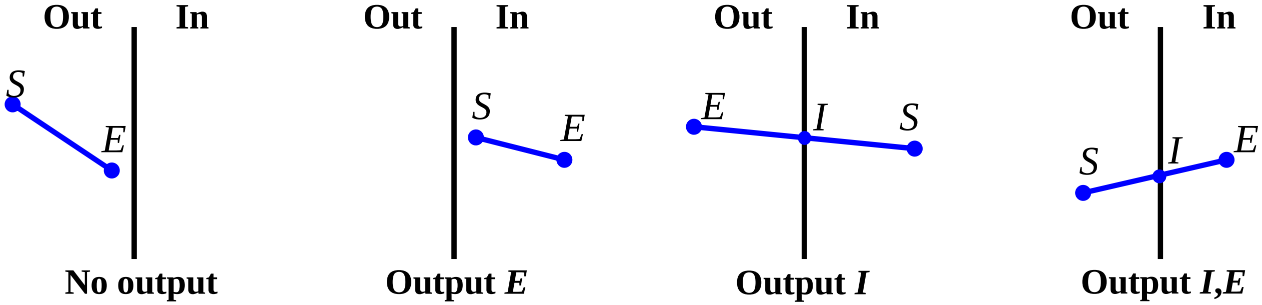


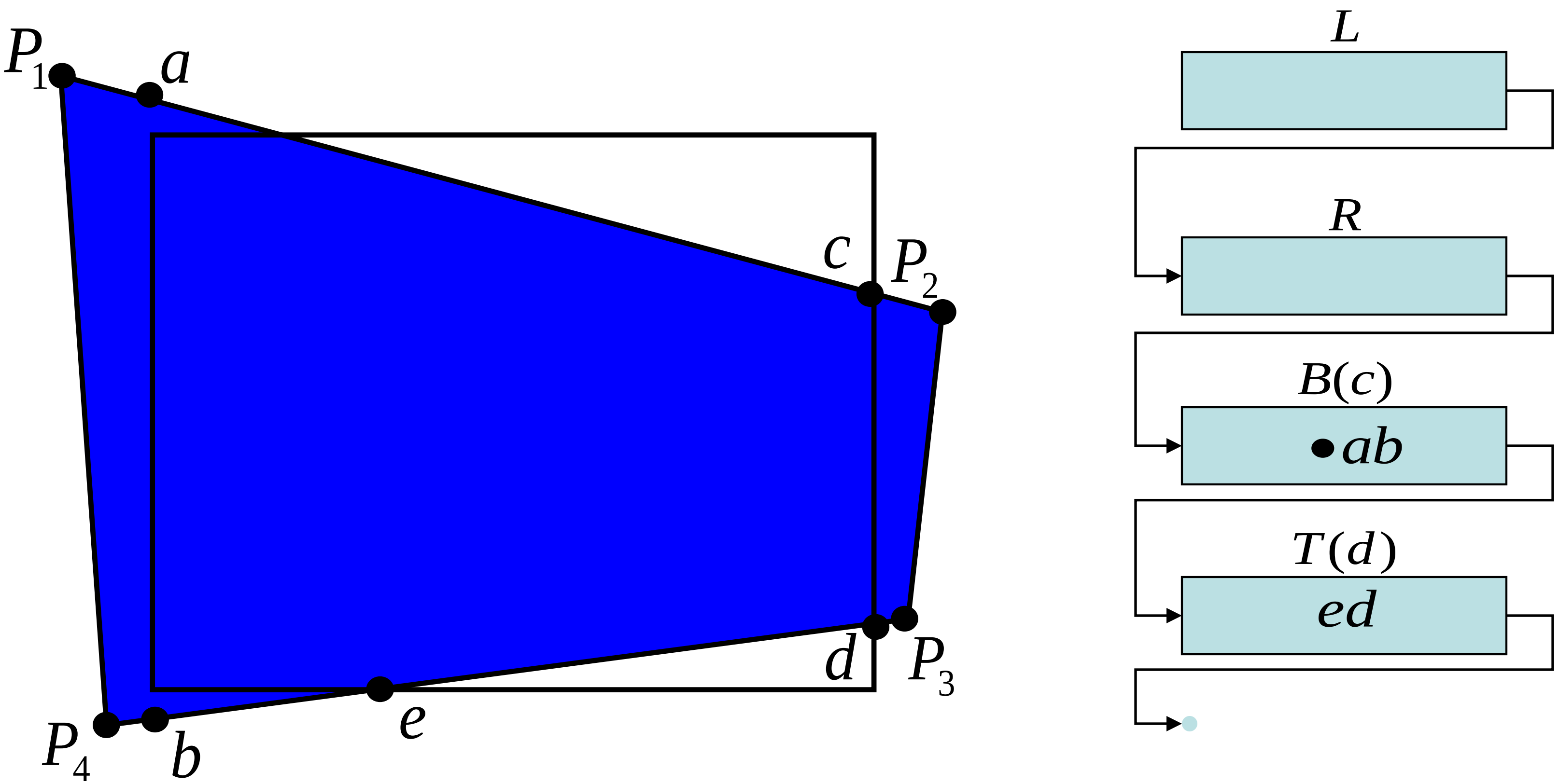
Sutherland-Hodgman Algorithm



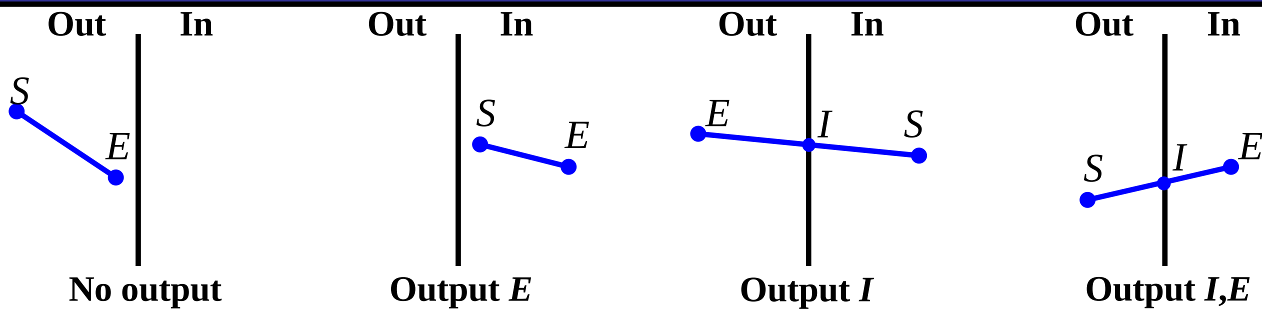


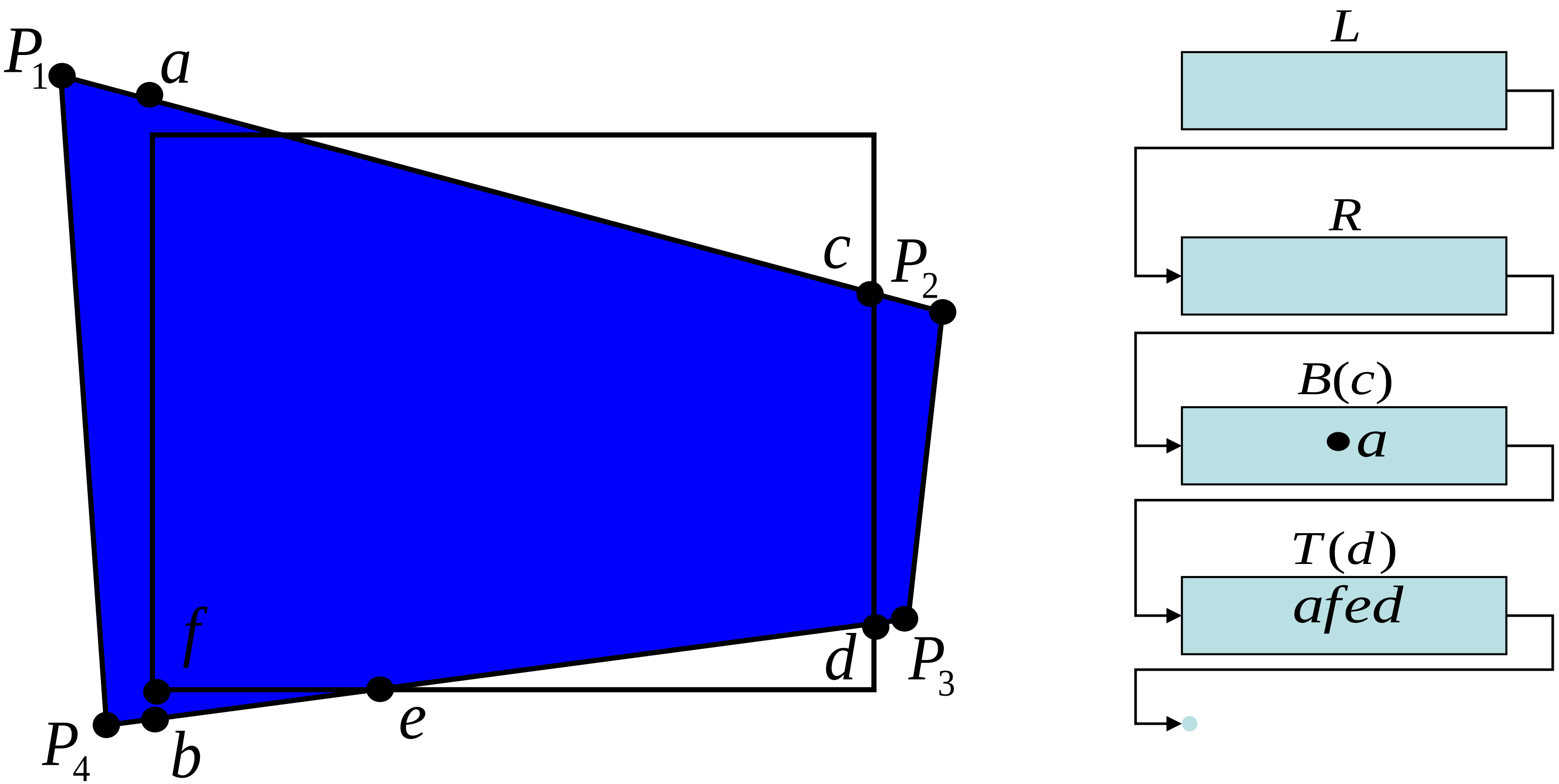
Sutherland-Hodgman Algorithm



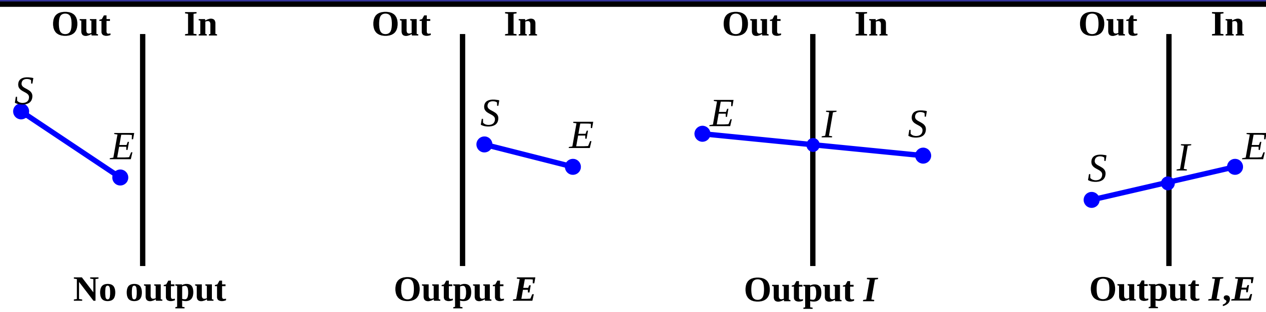


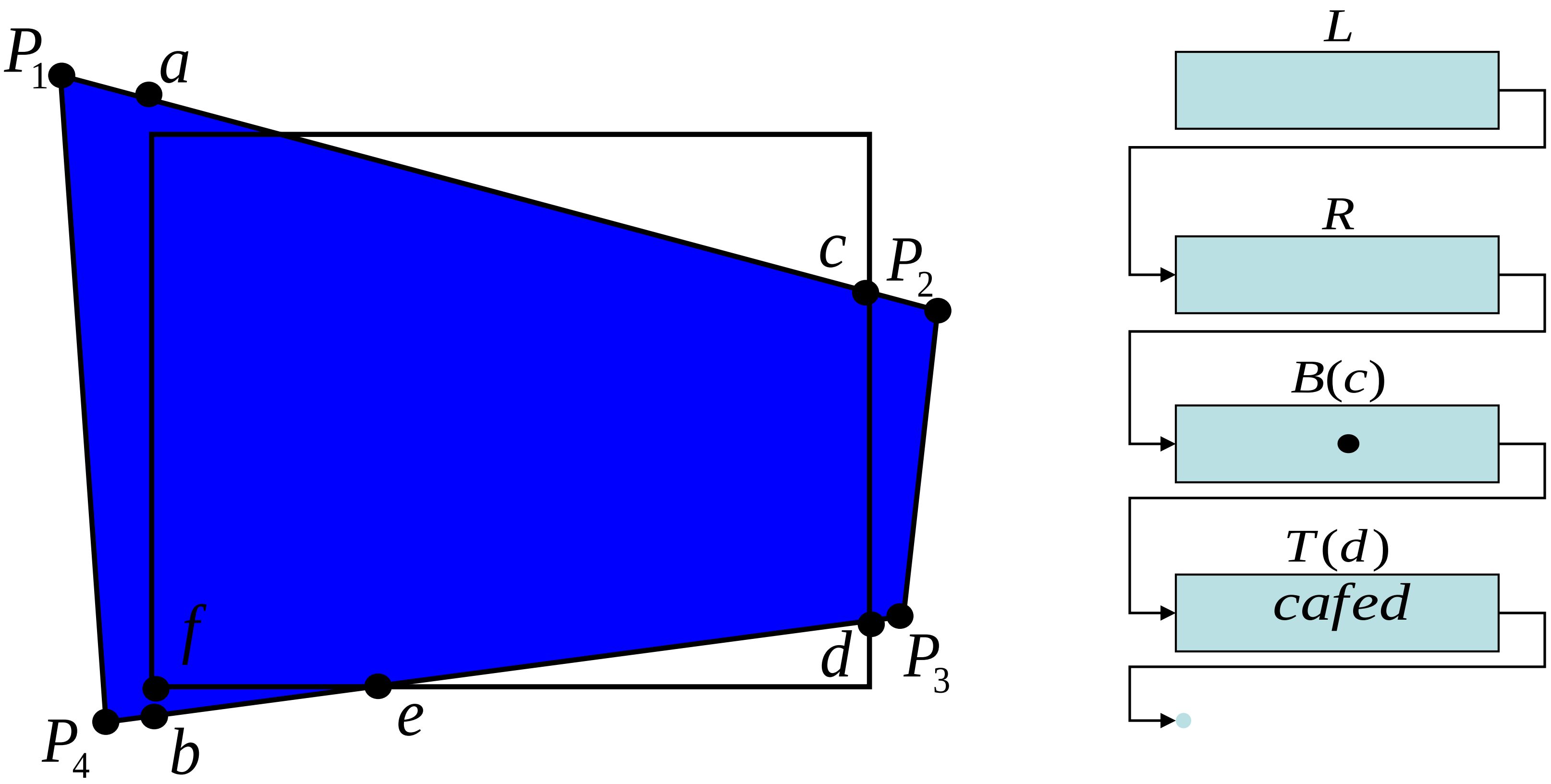
Sutherland-Hodgman Algorithm



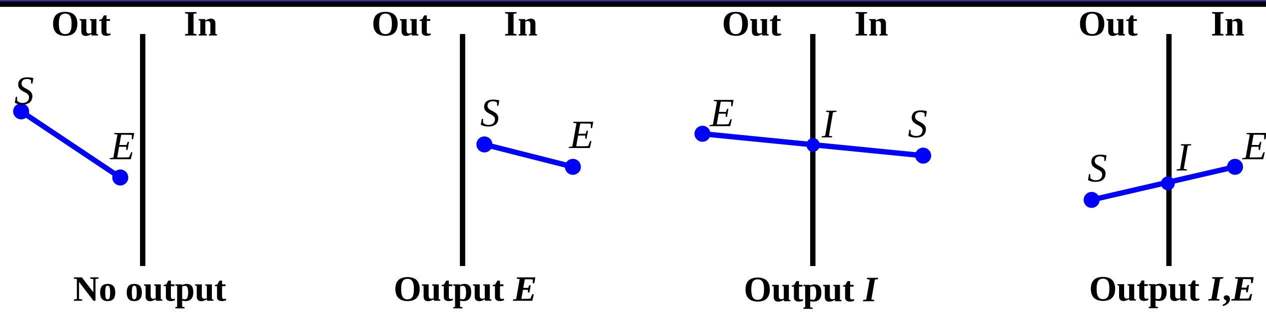


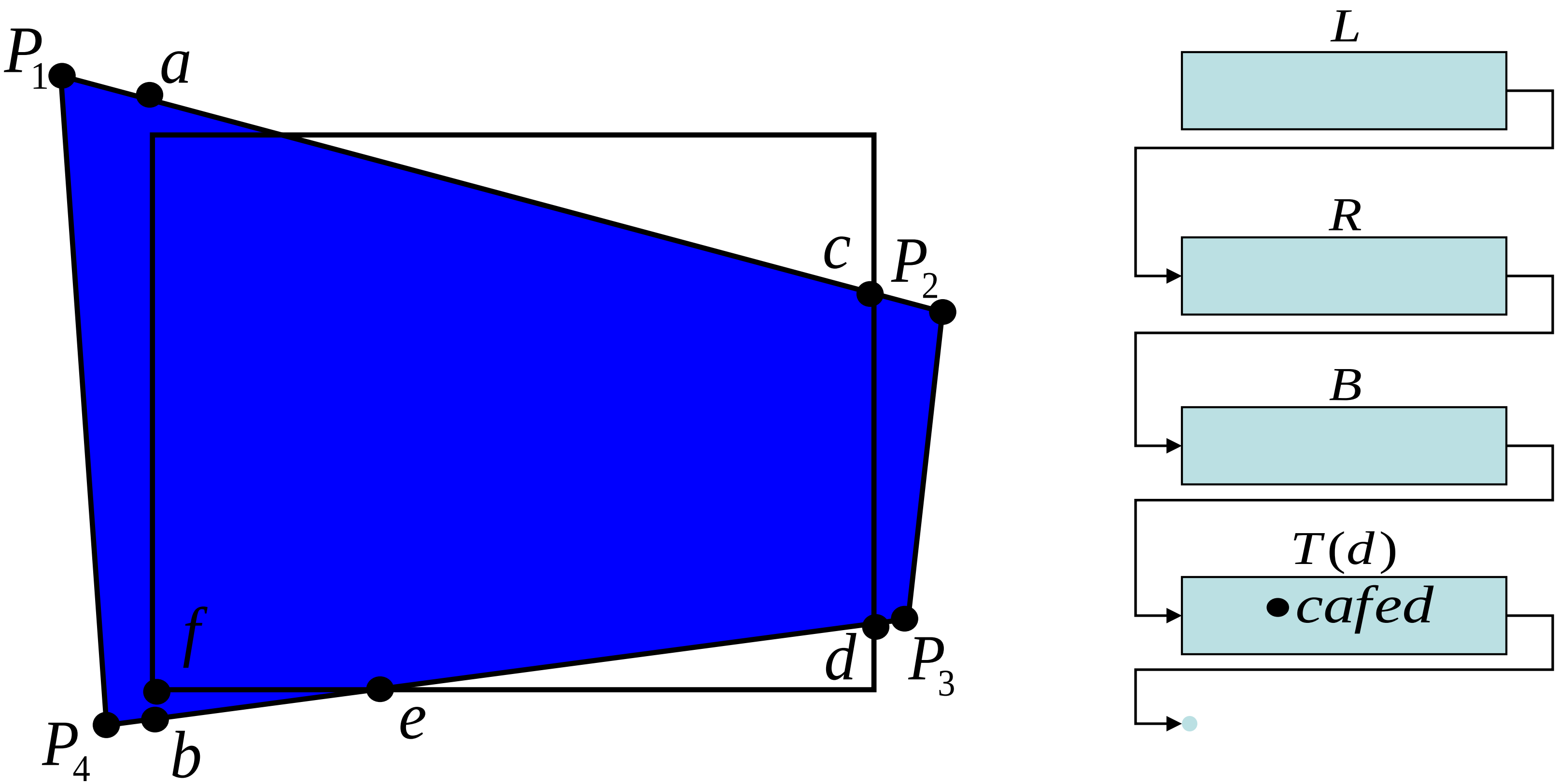
Sutherland-Hodgman Algorithm



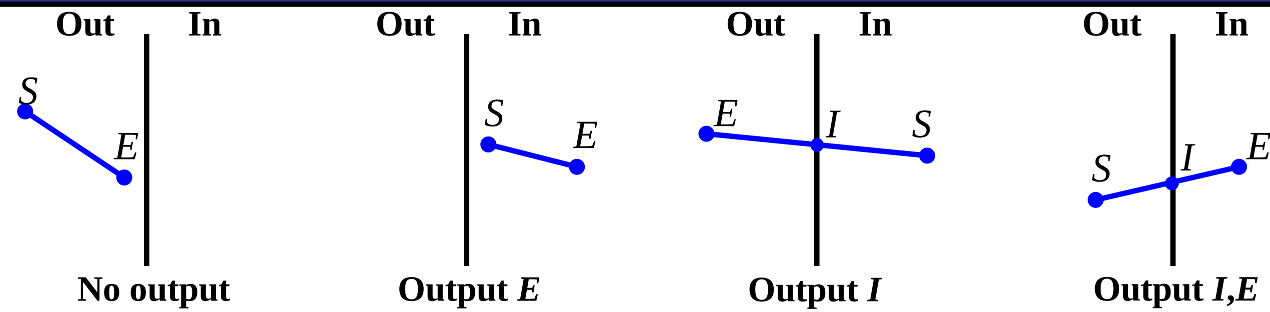


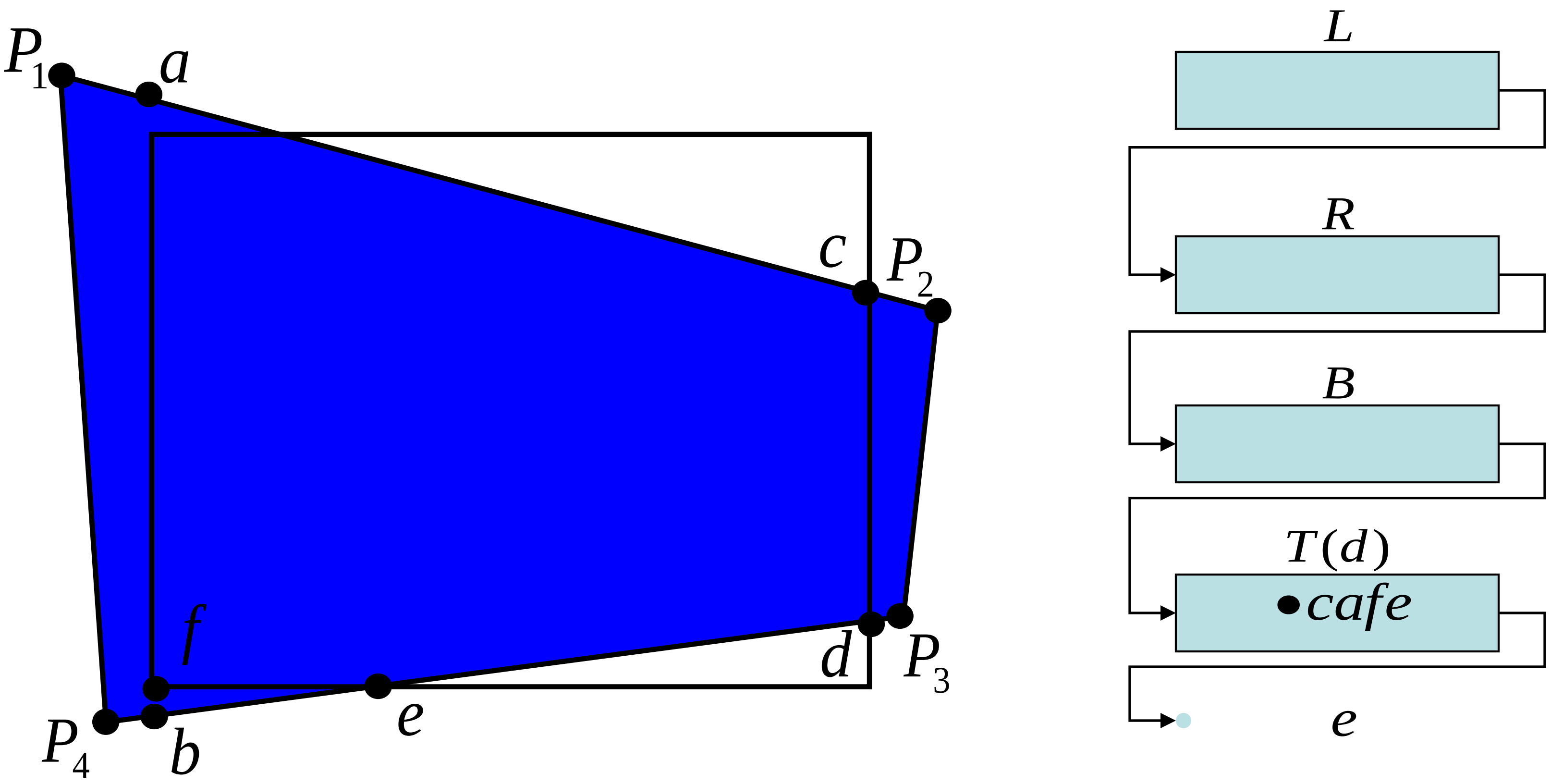
Sutherland-Hodgman Algorithm



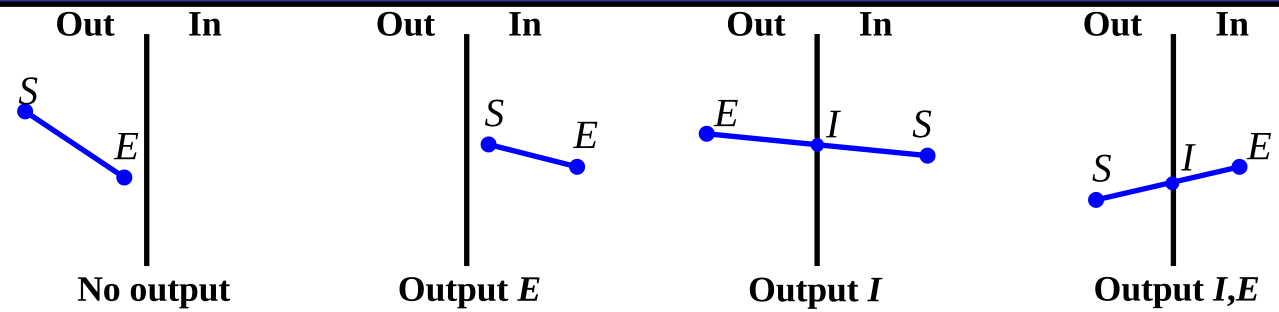


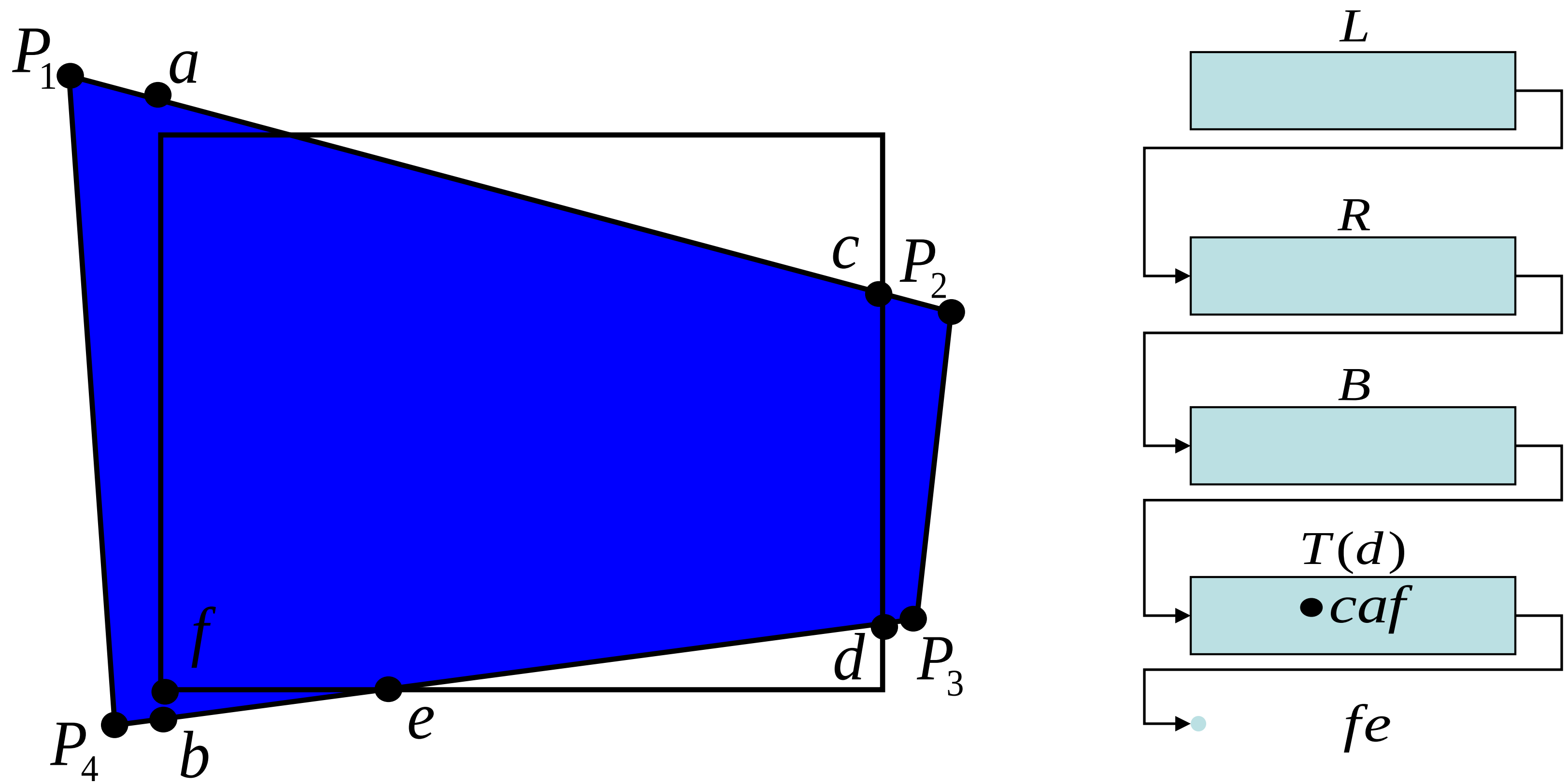
Sutherland-Hodgman Algorithm



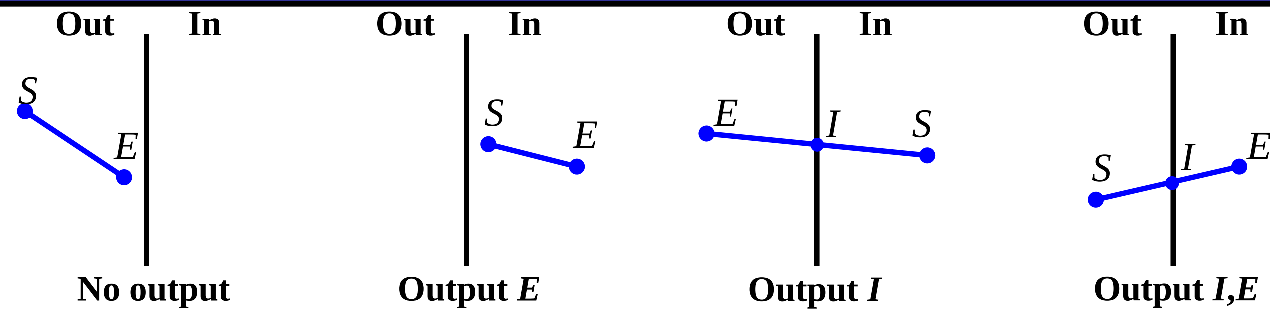


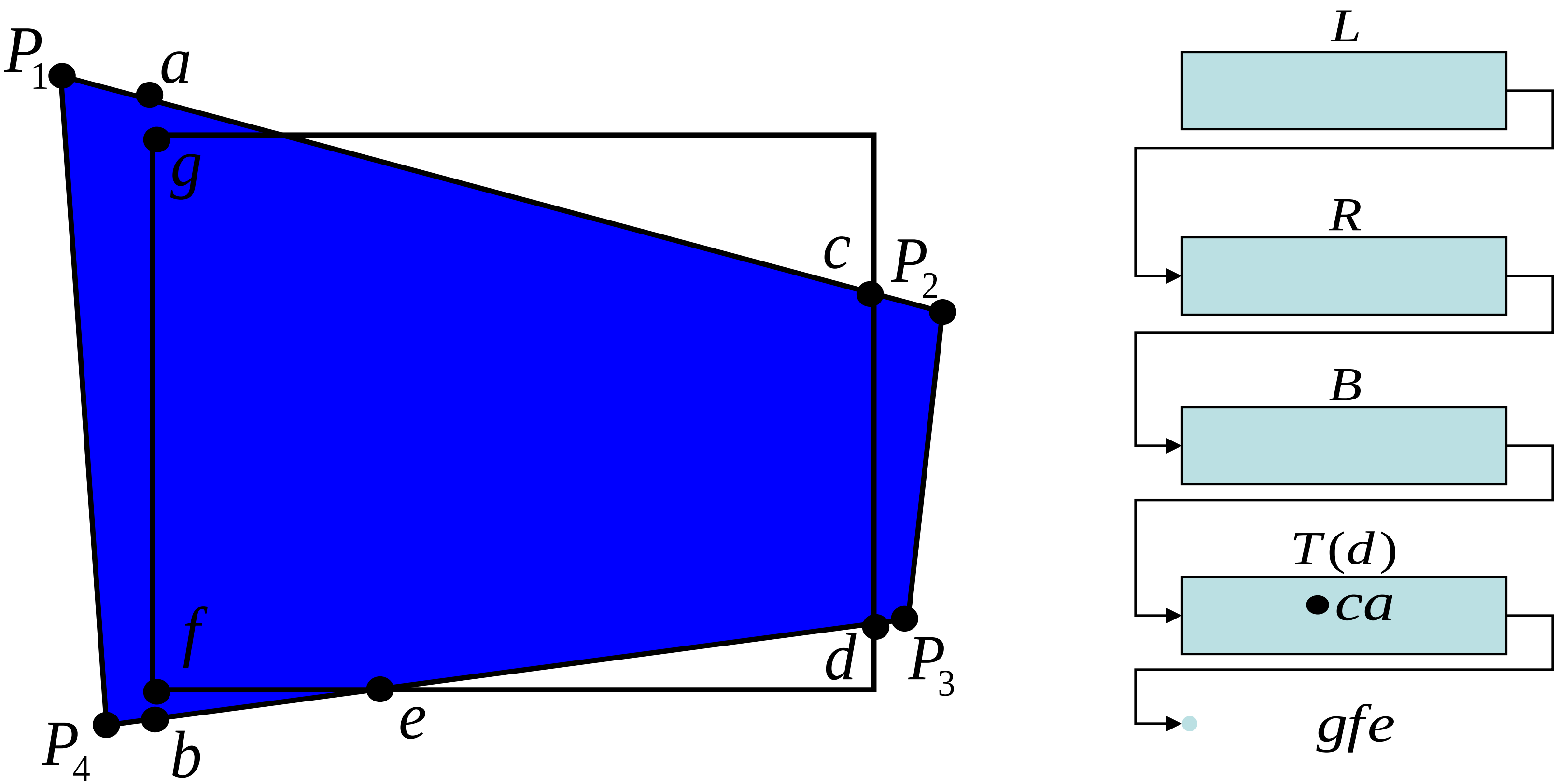
Sutherland-Hodgman Algorithm



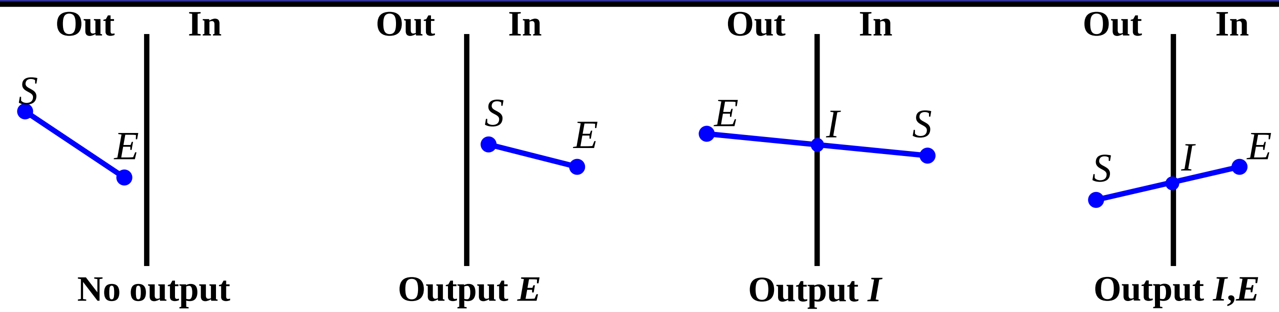


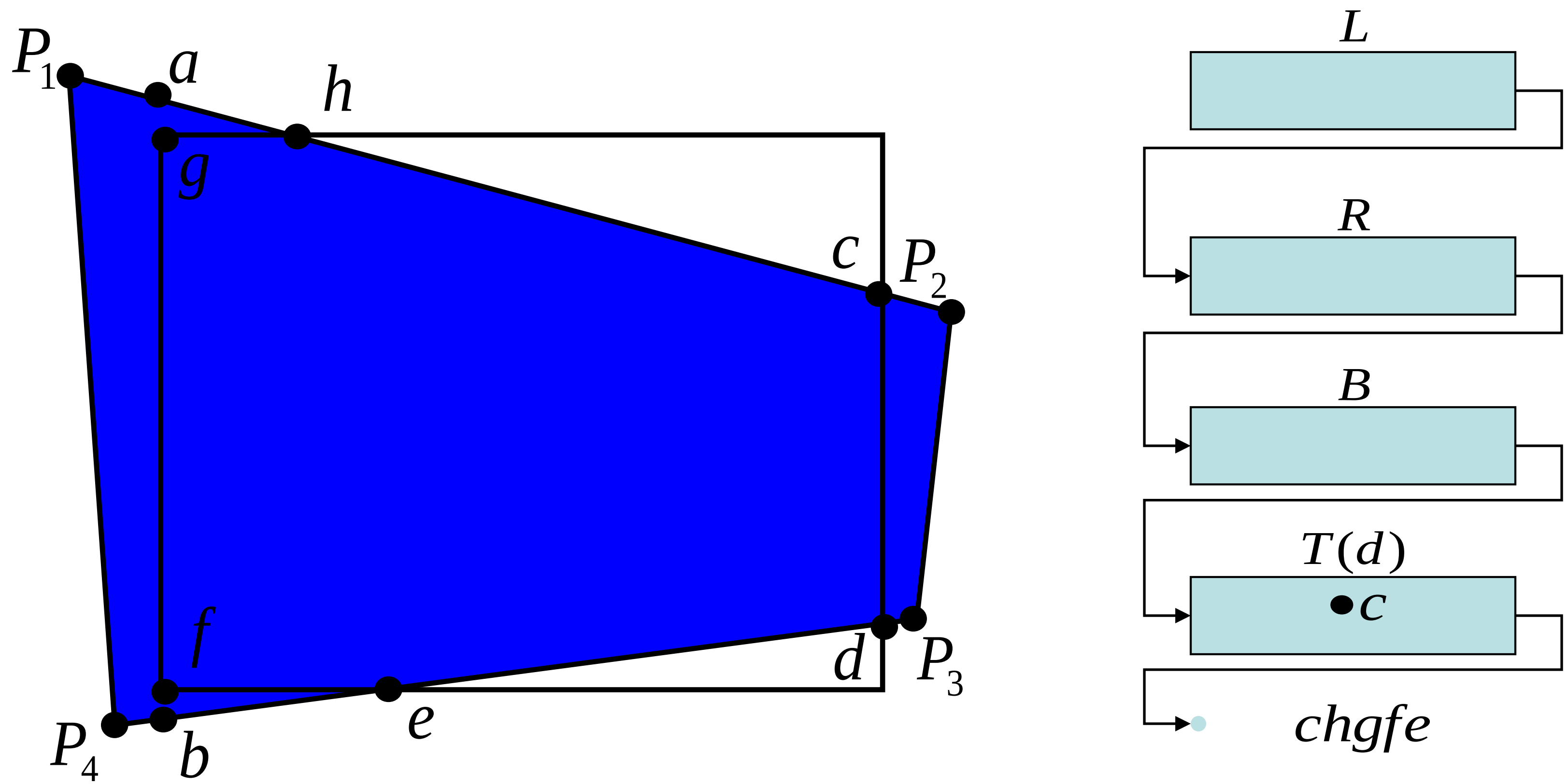
Sutherland-Hodgman Algorithm



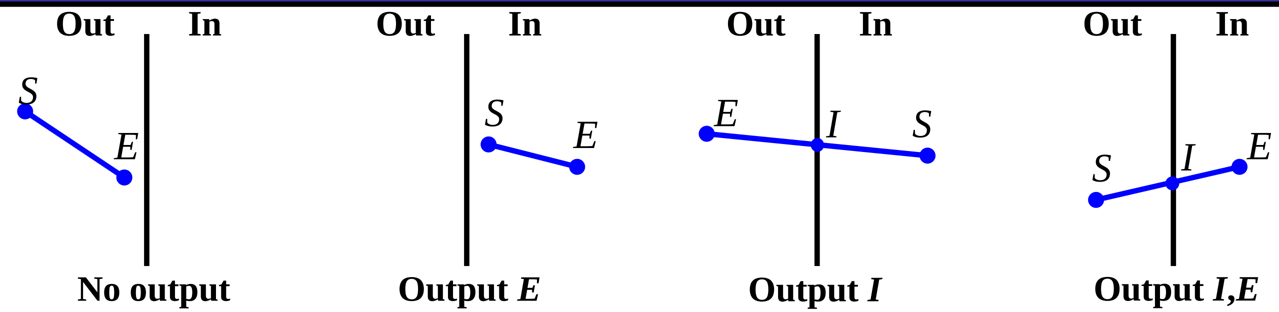


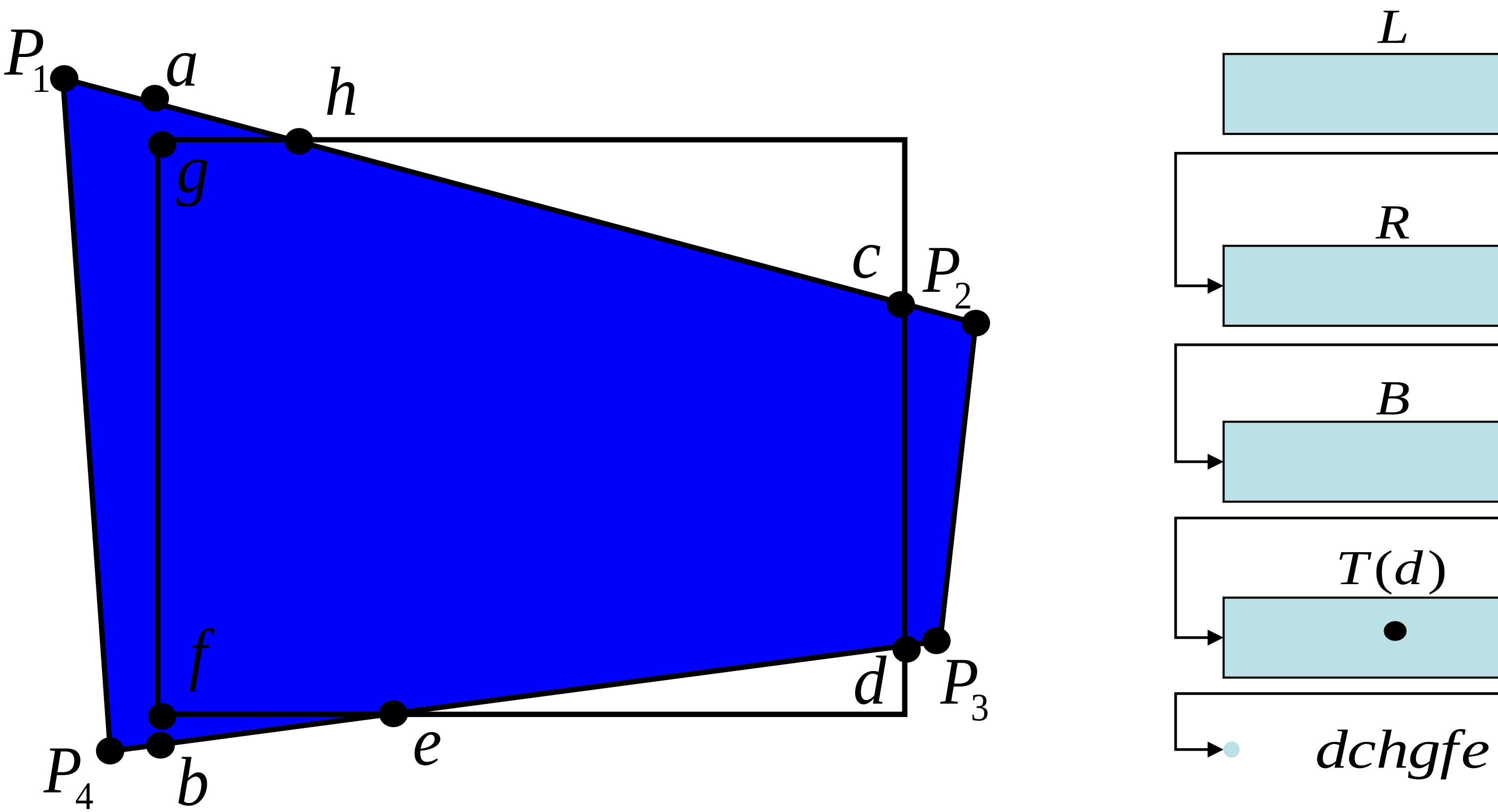
Sutherland-Hodgman Algorithm



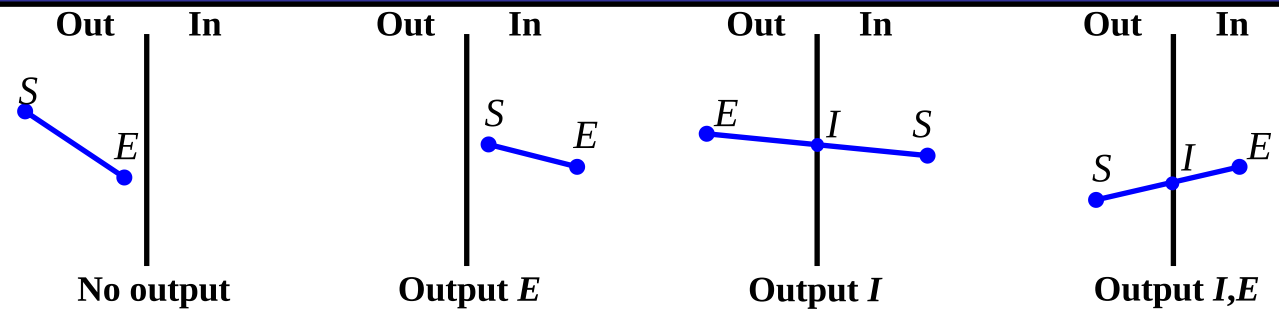


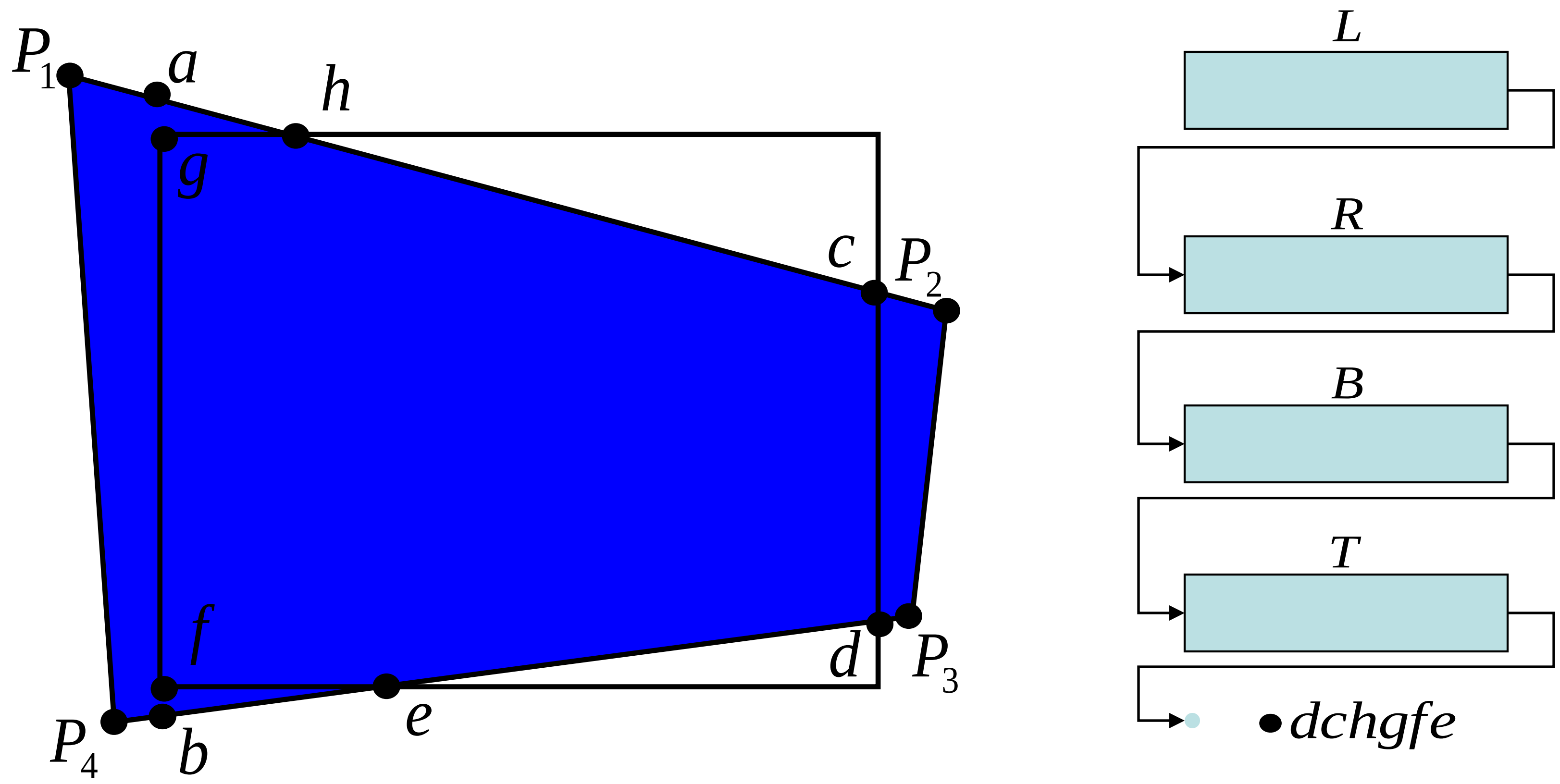
Sutherland-Hodgman Algorithm





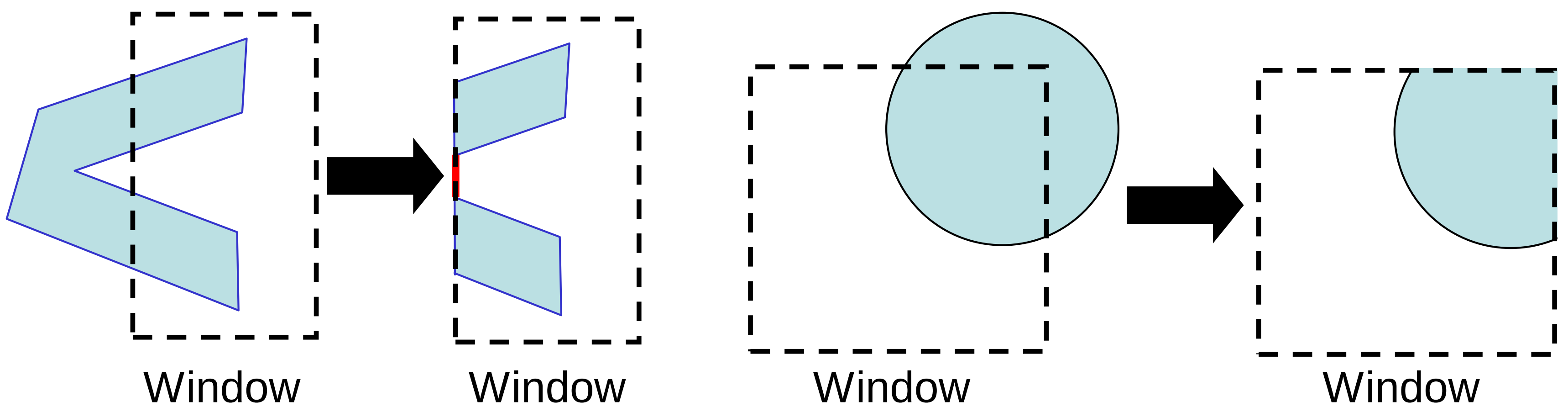
Sutherland-Hodgman Algorithm





Other Area Clipping Concerns

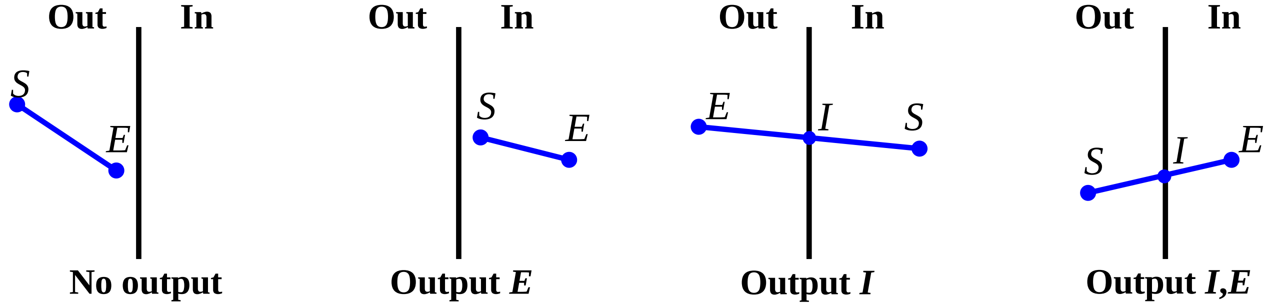
Clipping concave areas can be a little more tricky as often superfluous lines must be removed

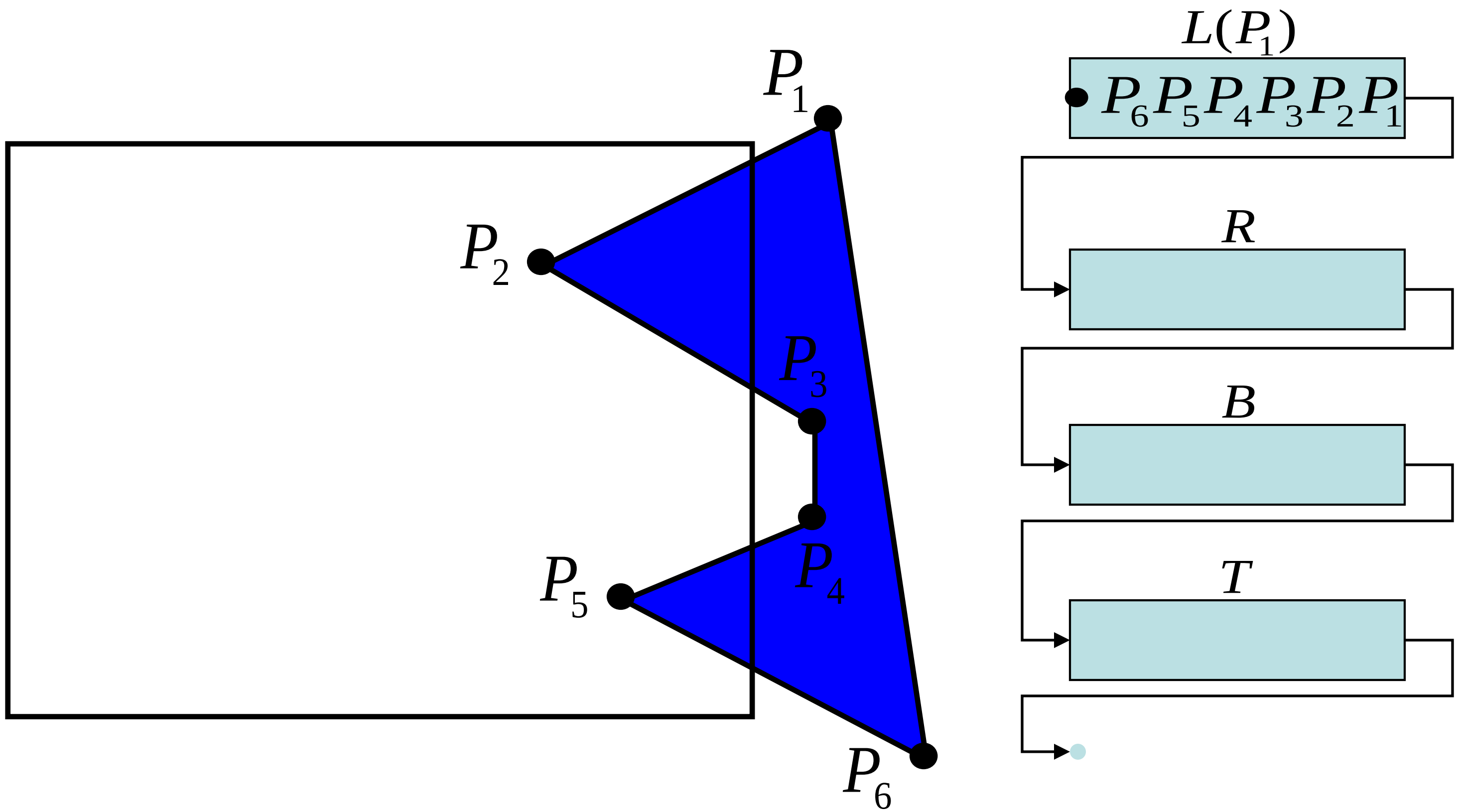


Clipping curves requires more work

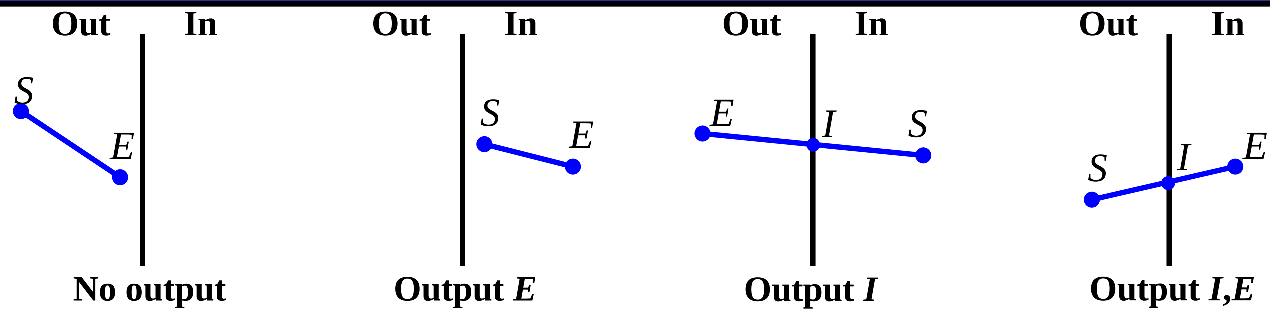
– For circles we must find the two intersection points on the window boundary

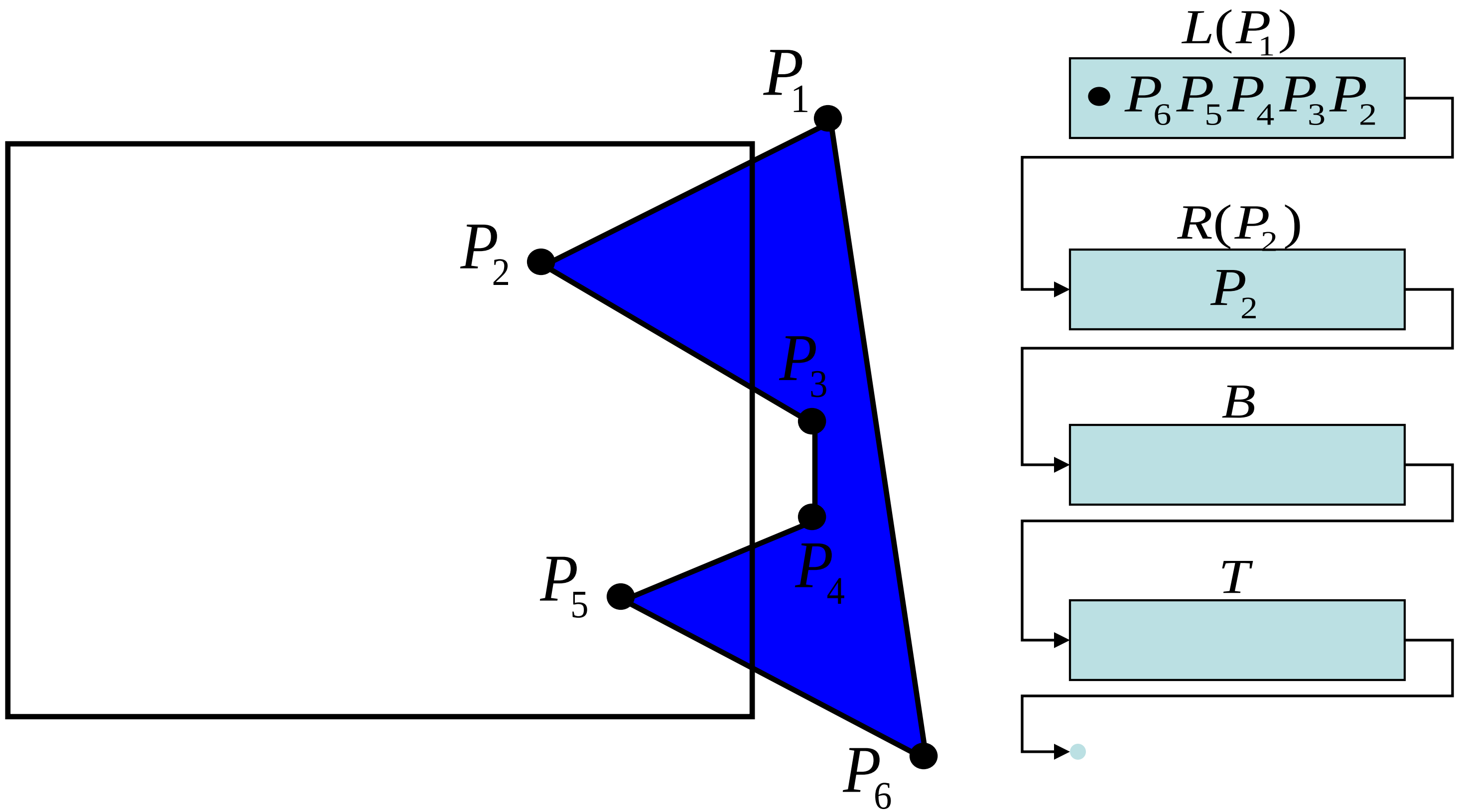
Sutherland-Hodgman Algorithm



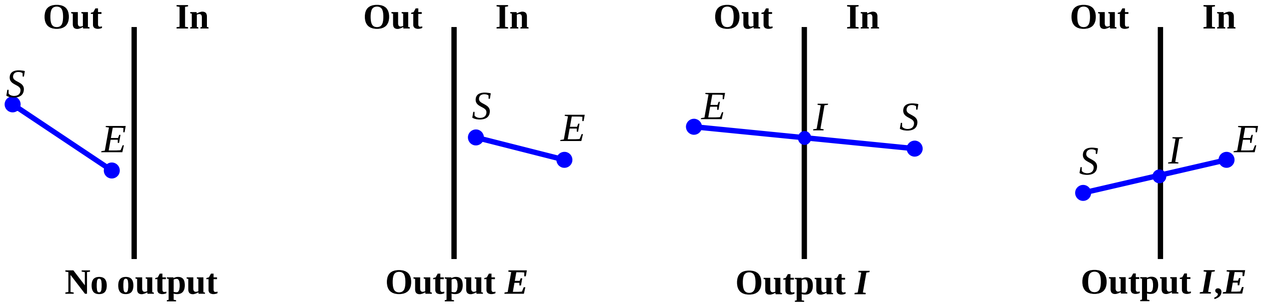


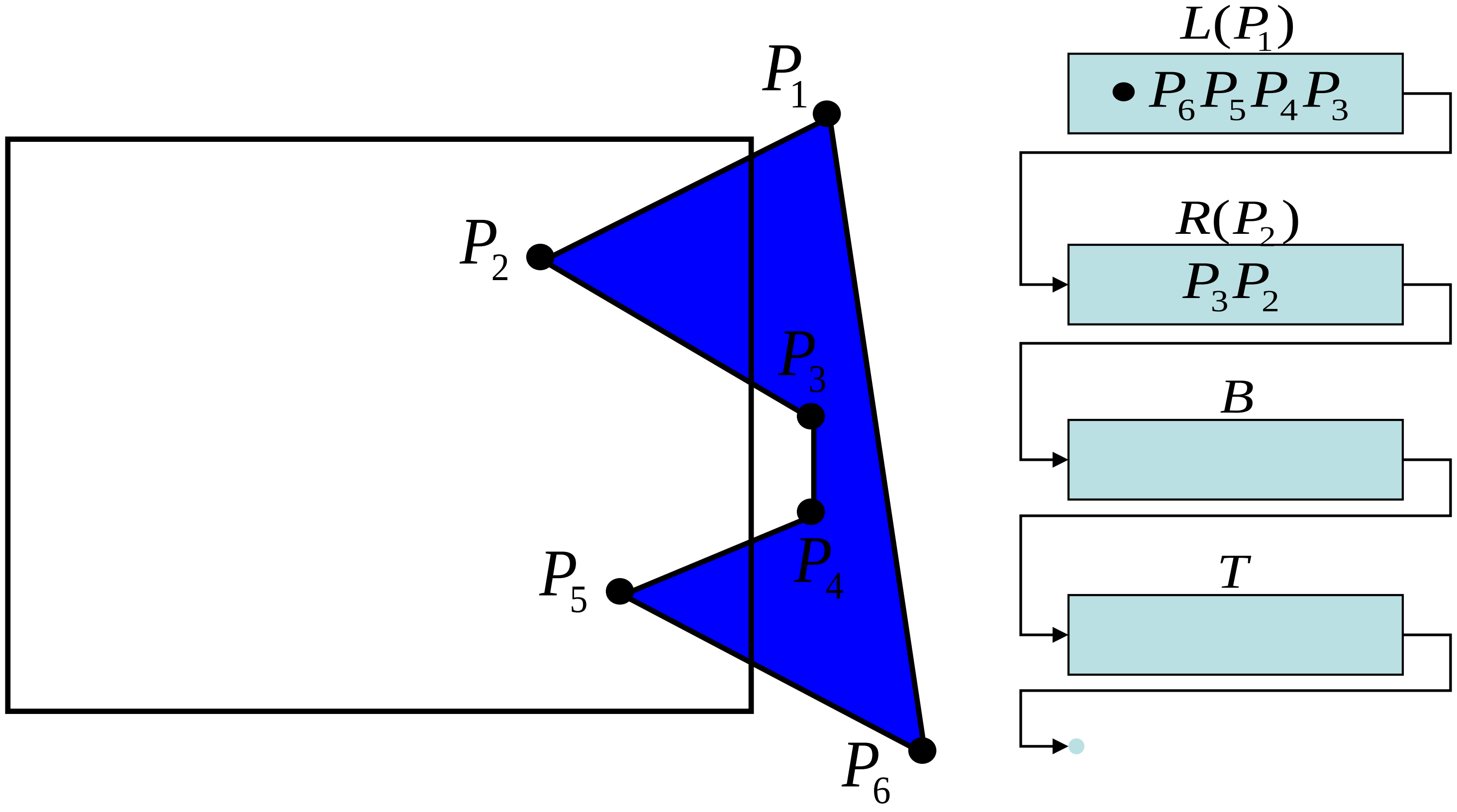
Sutherland-Hodgman Algorithm



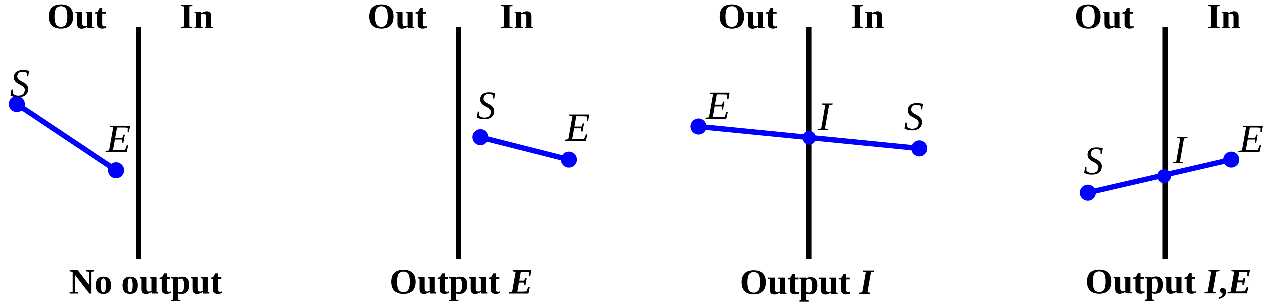


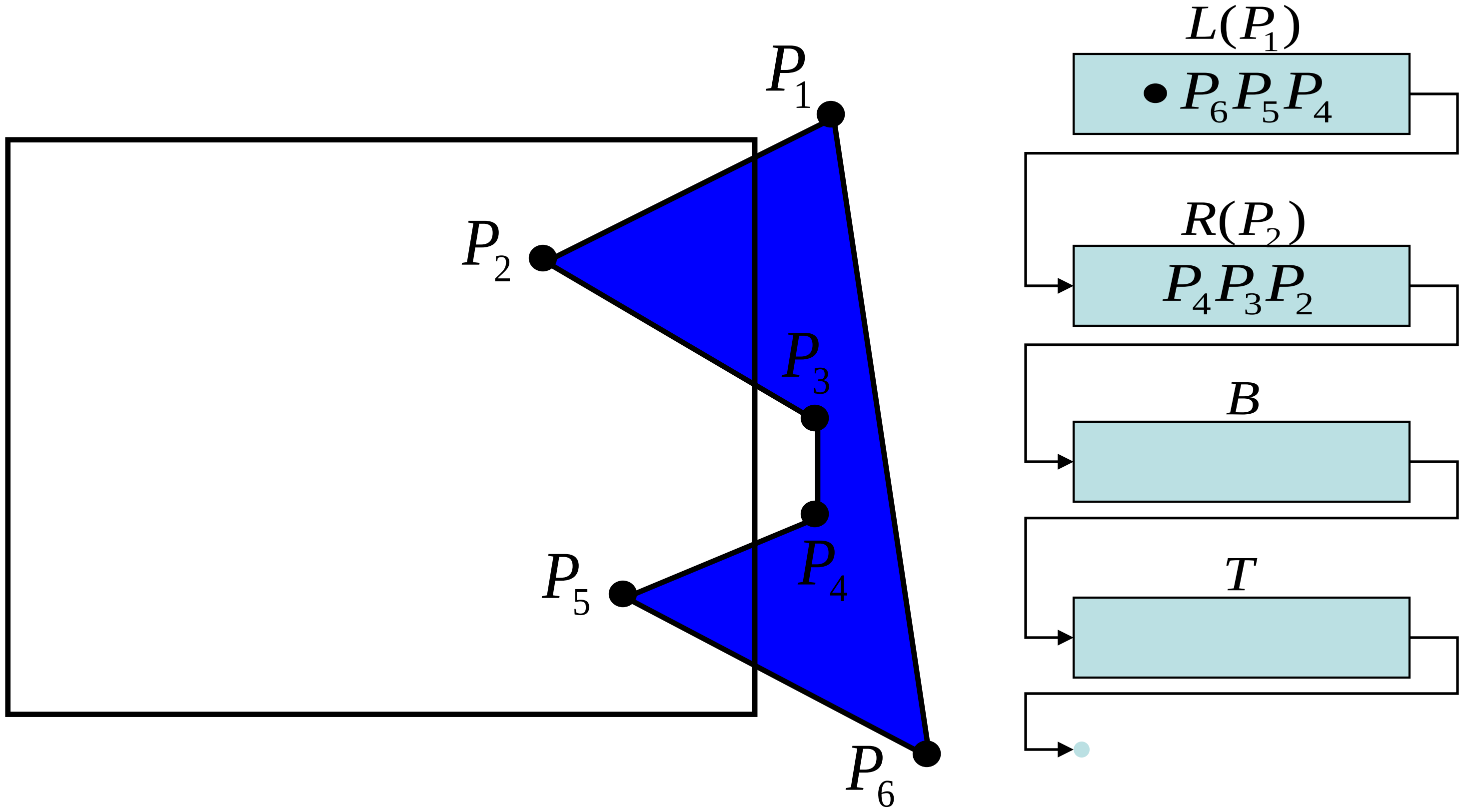
Sutherland-Hodgman Algorithm



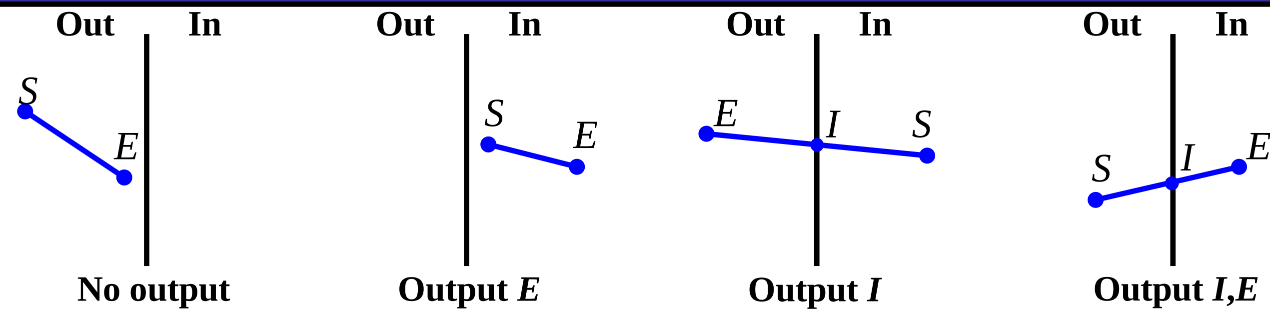


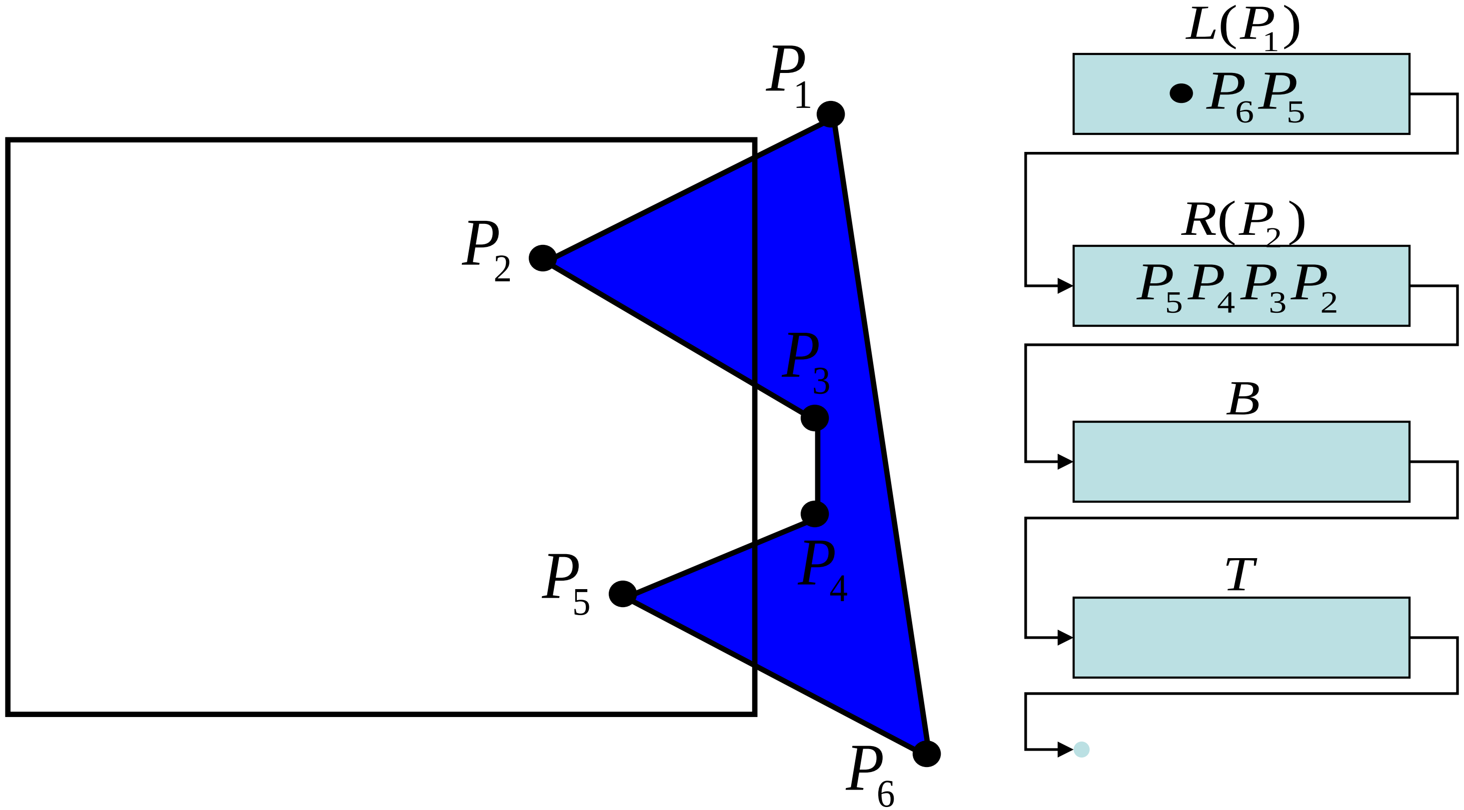
Sutherland-Hodgman Algorithm



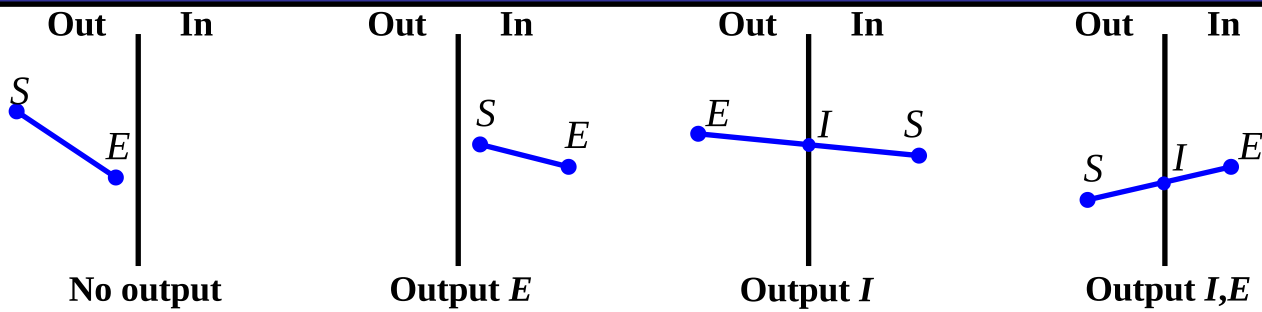


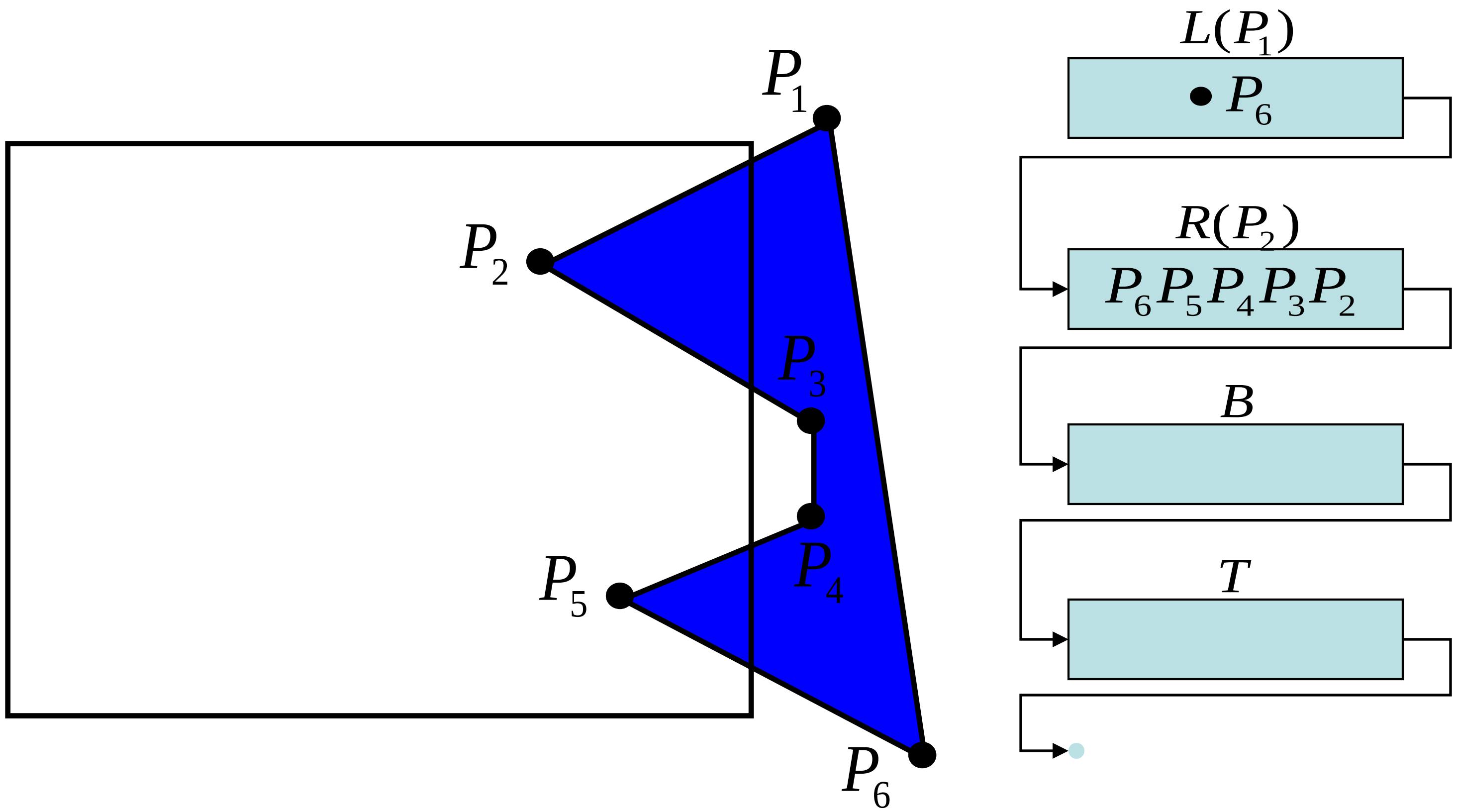
Sutherland-Hodgman Algorithm



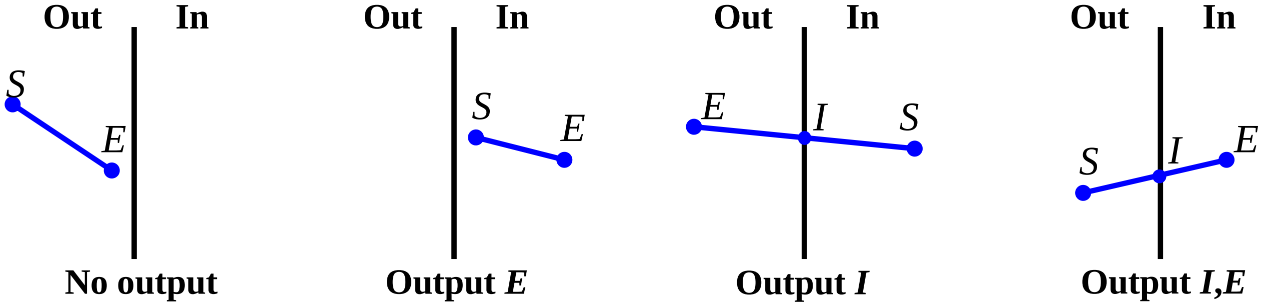


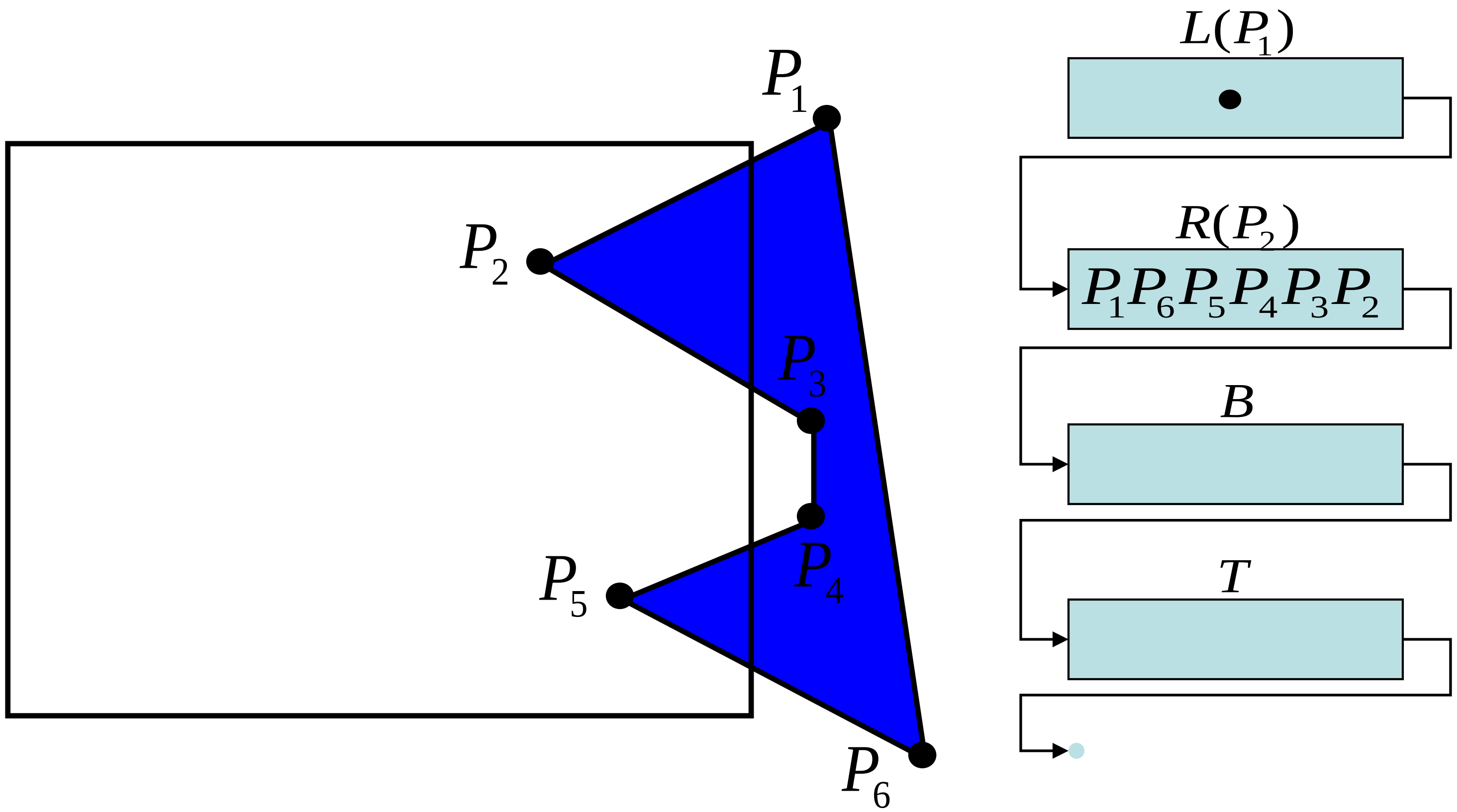
Sutherland-Hodgman Algorithm



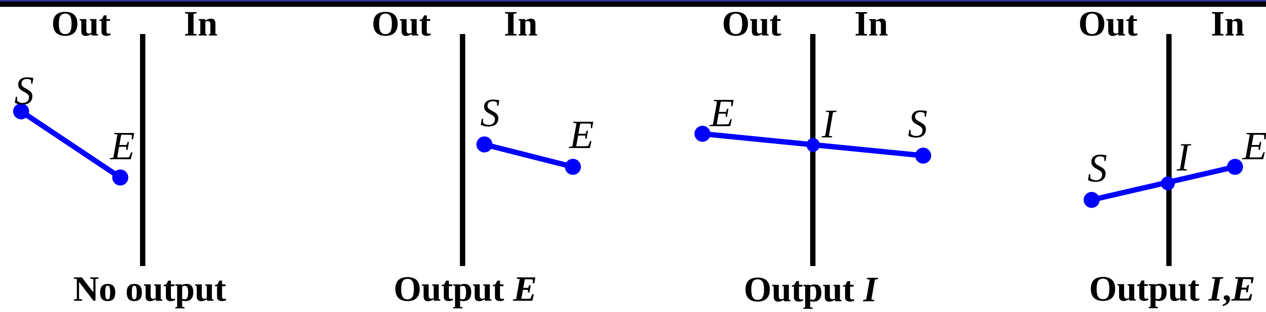


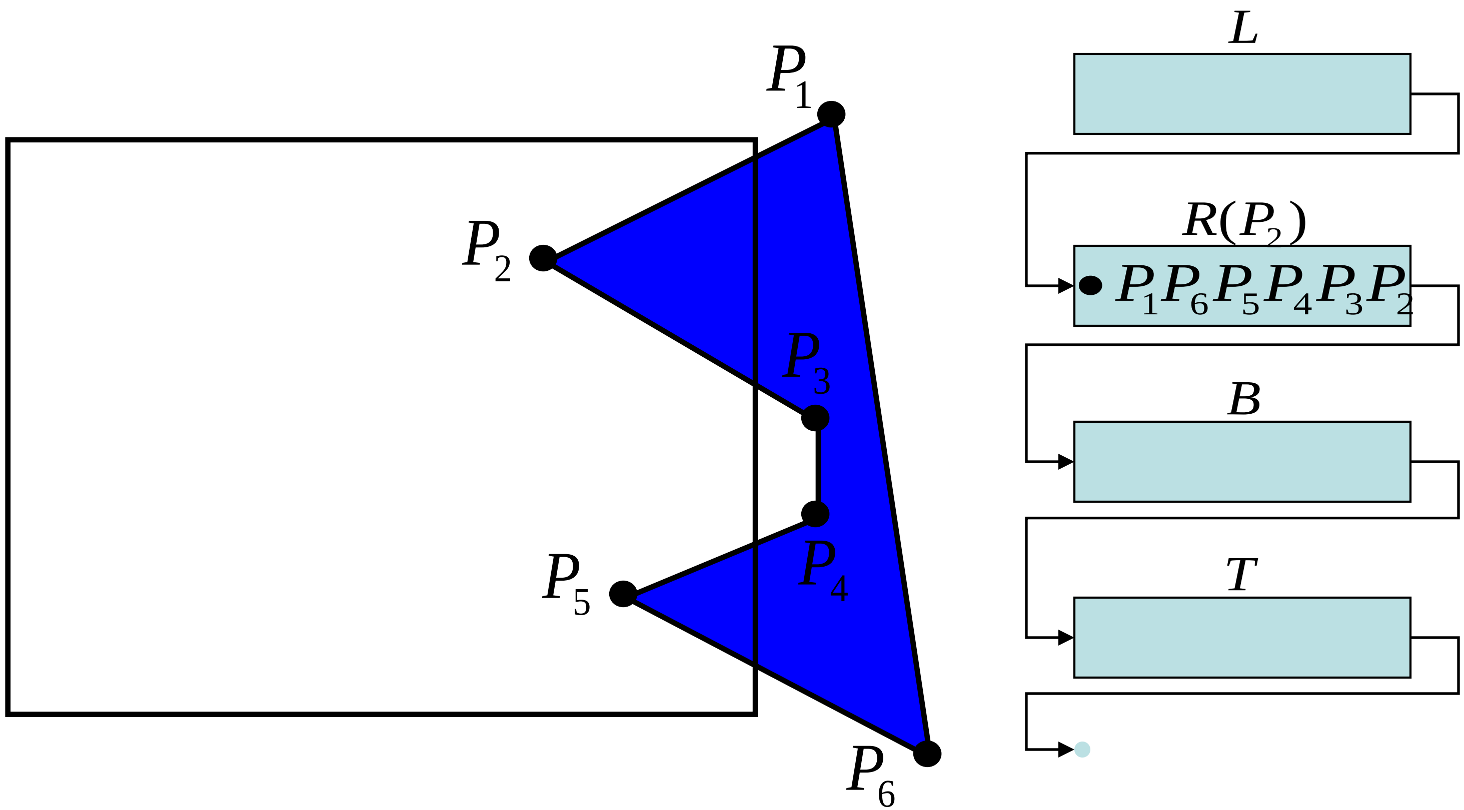
Sutherland-Hodgman Algorithm



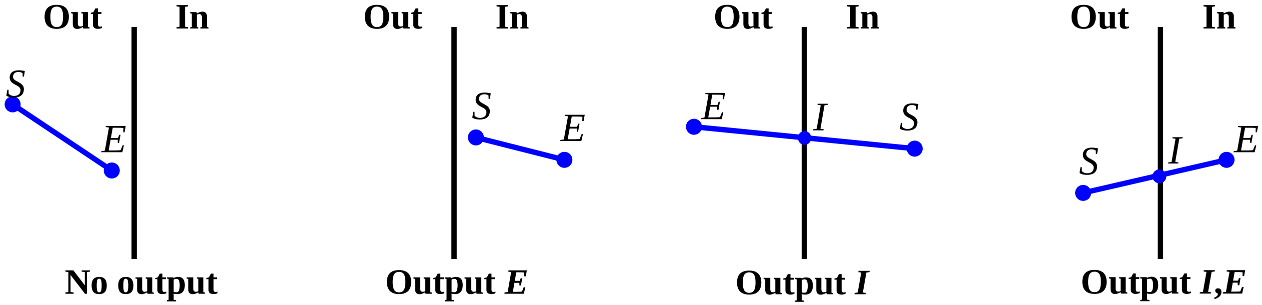


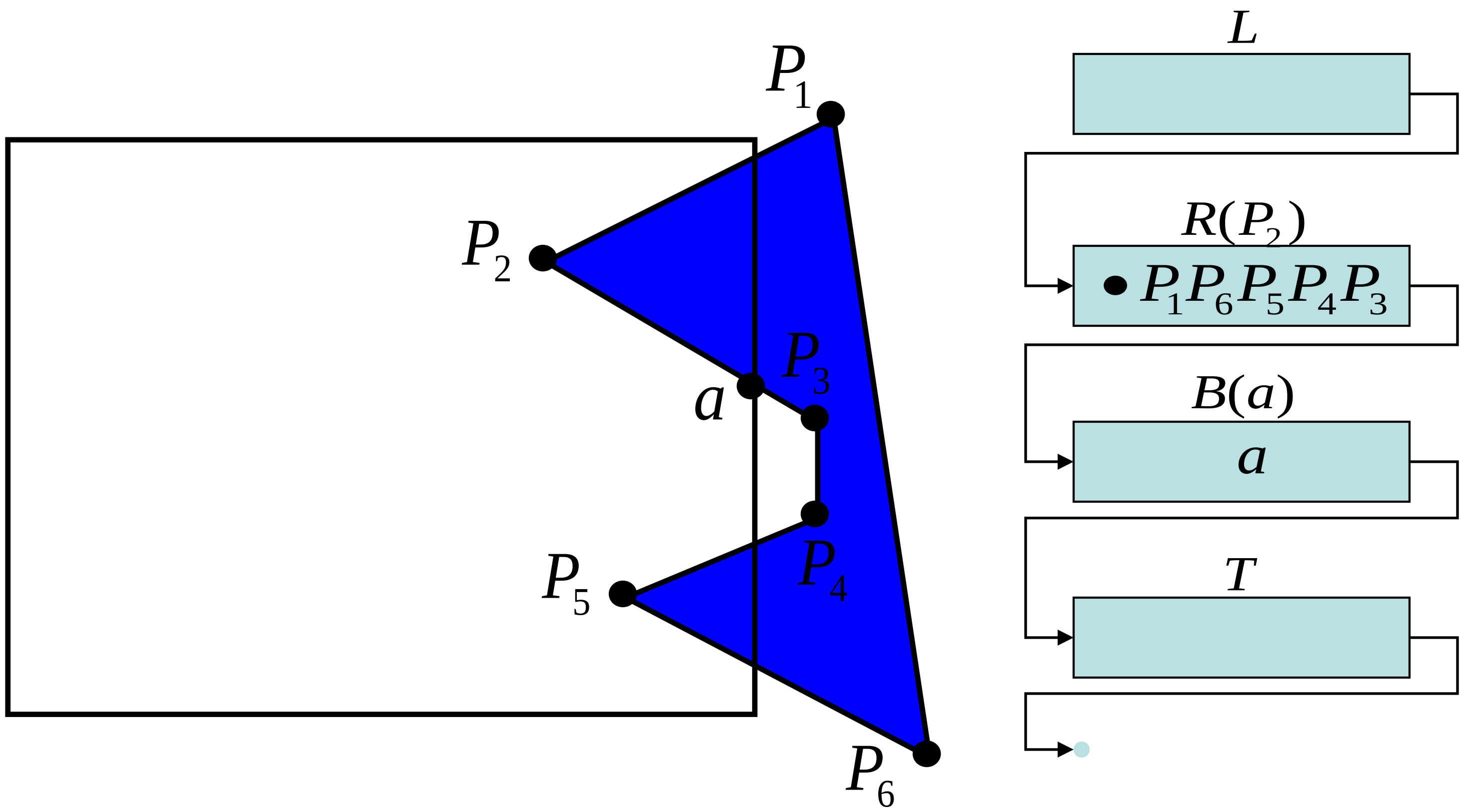
Sutherland-Hodgman Algorithm



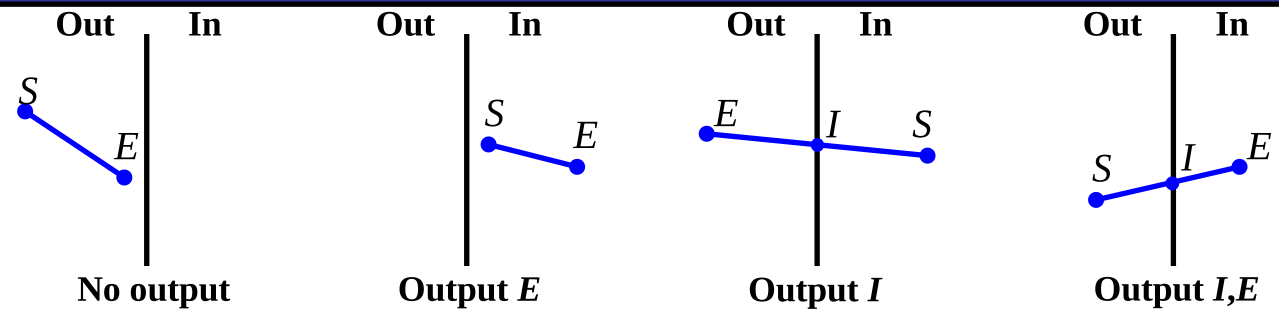


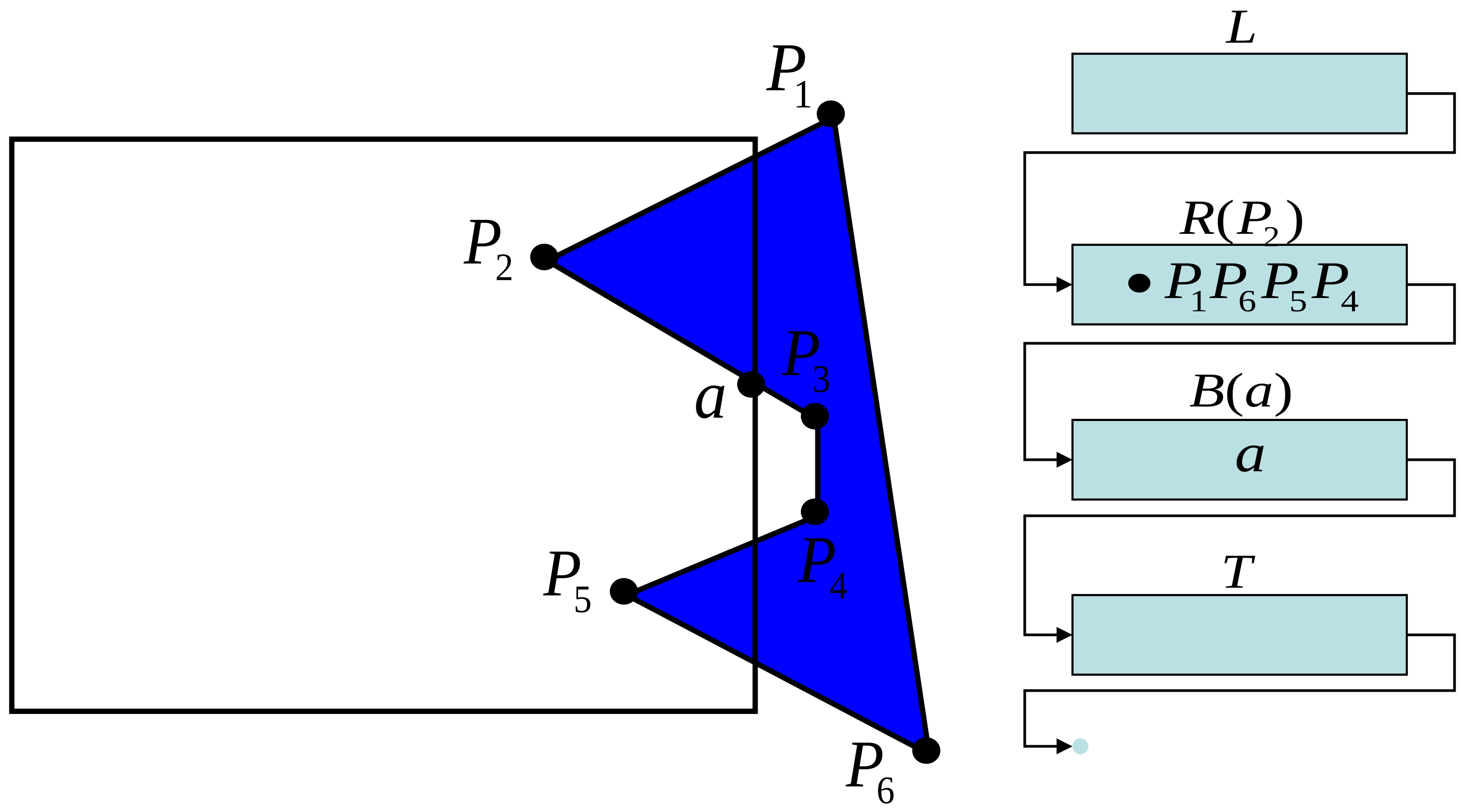
Sutherland-Hodgman Algorithm



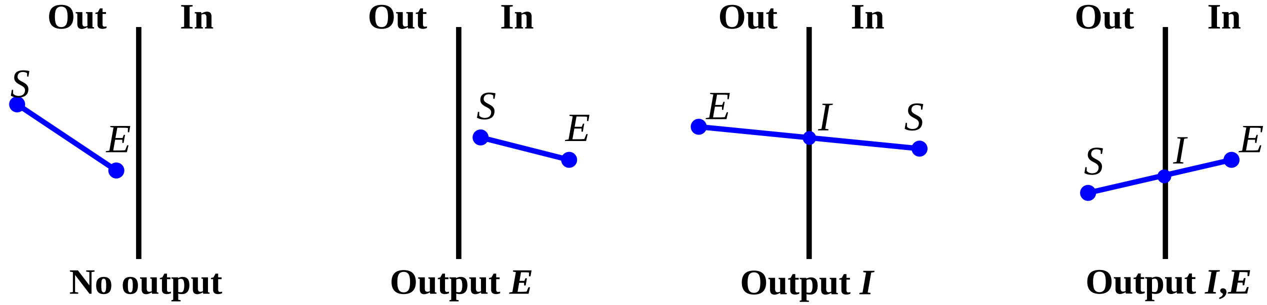


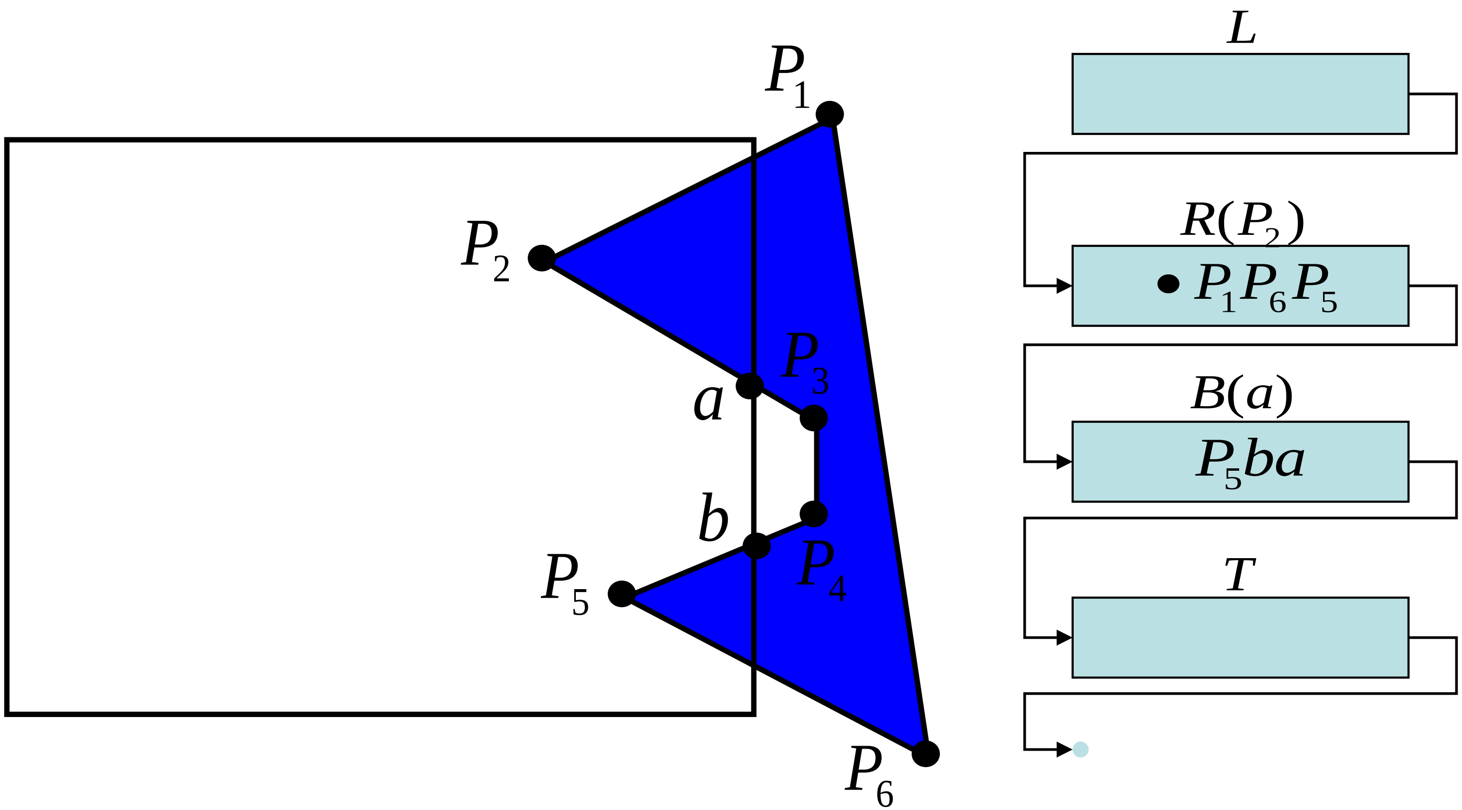
Sutherland-Hodgman Algorithm



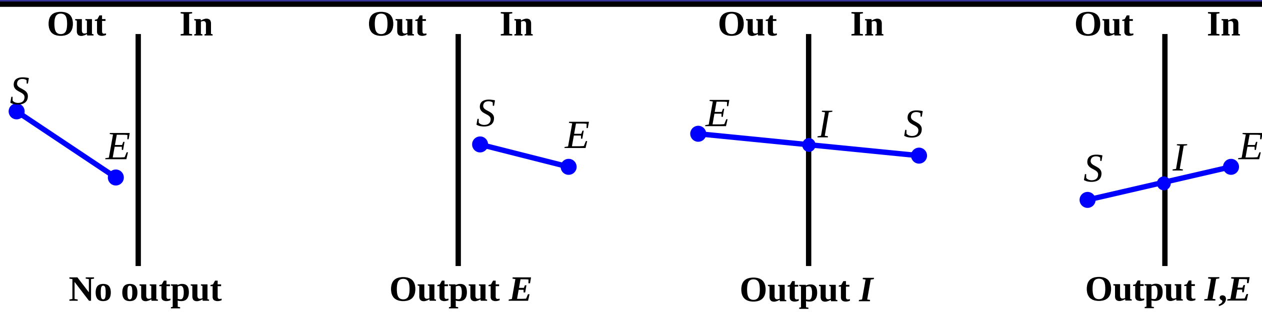


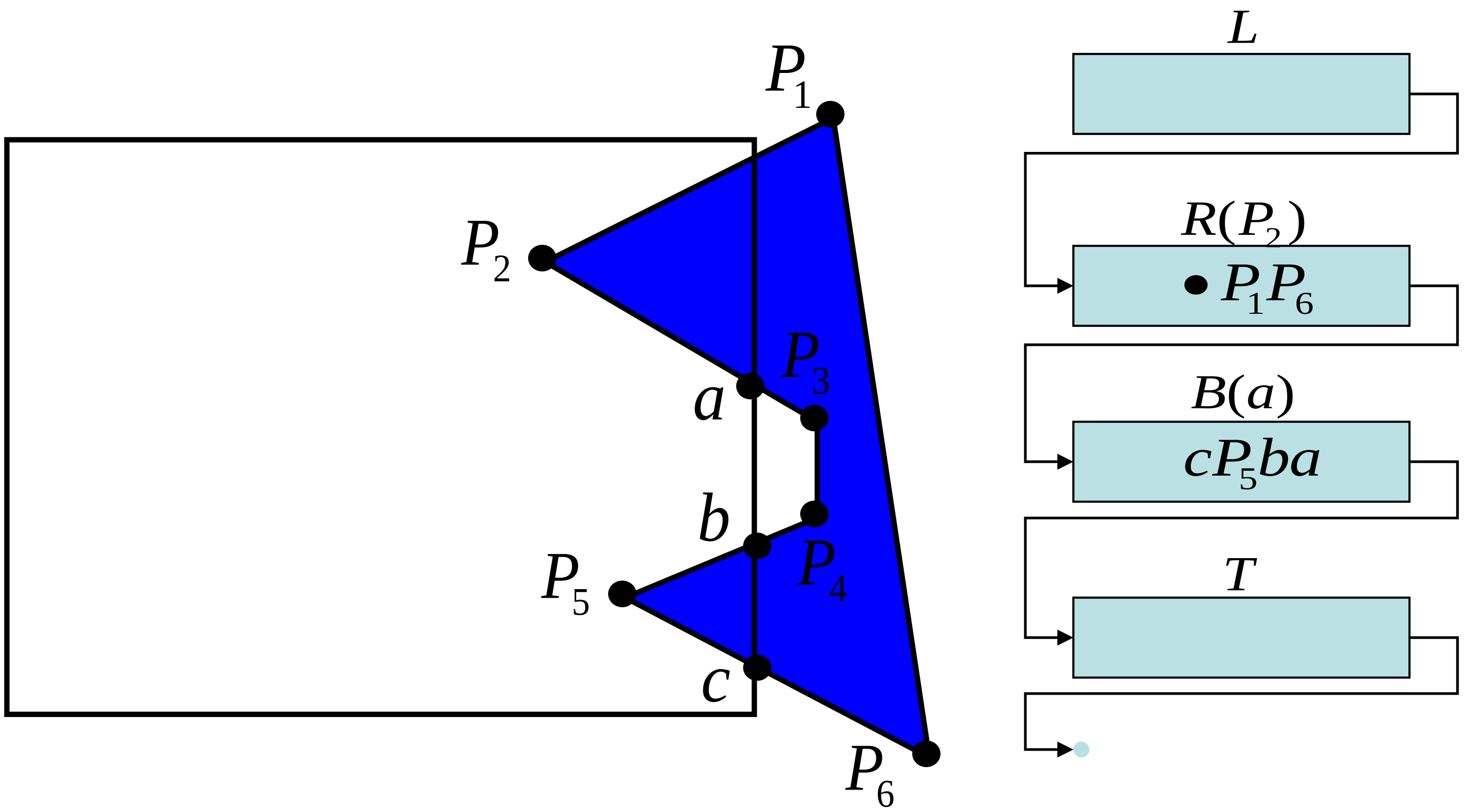
Sutherland-Hodgman Algorithm



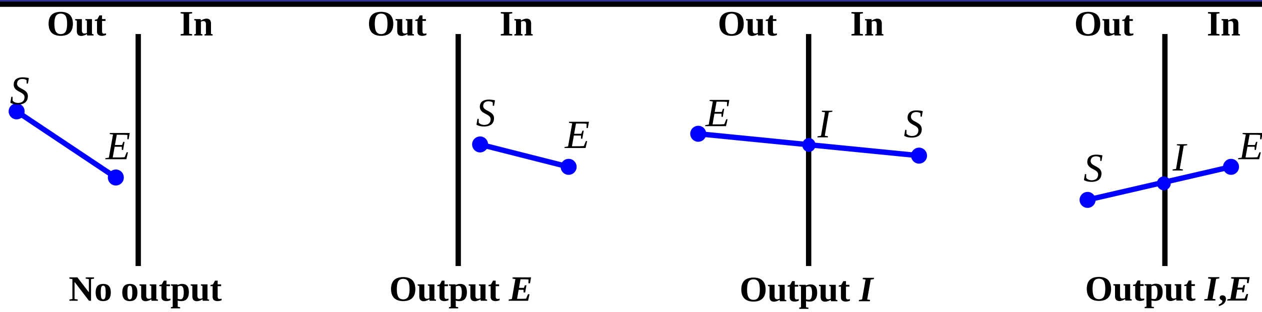


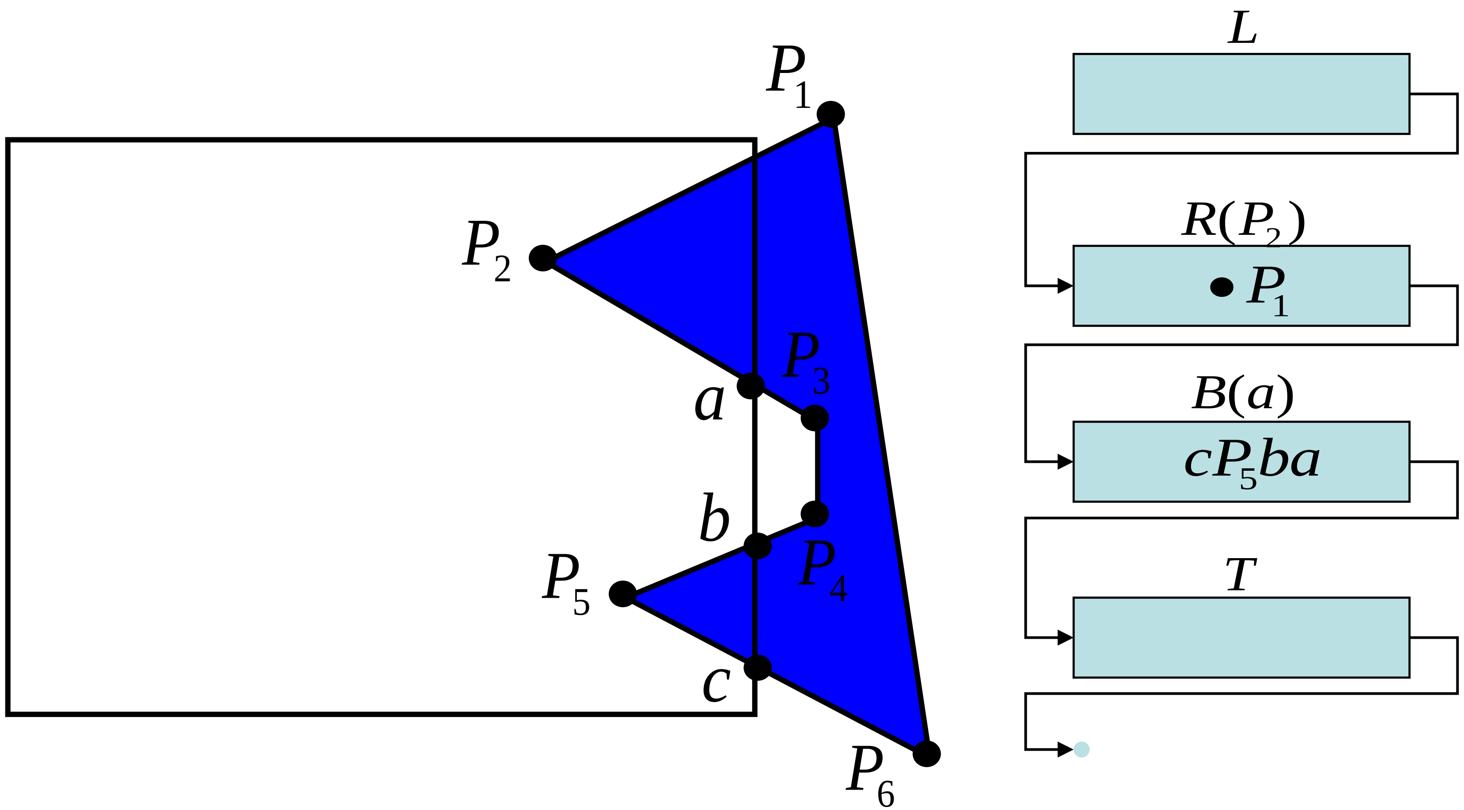
Sutherland-Hodgman Algorithm



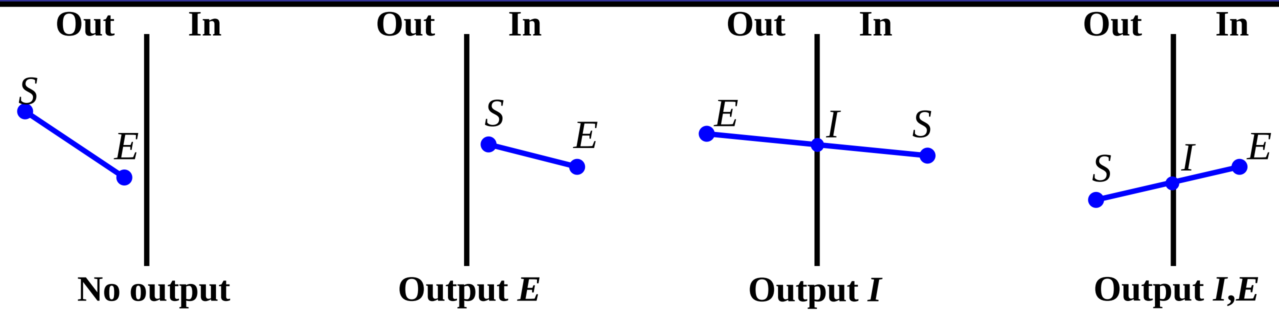


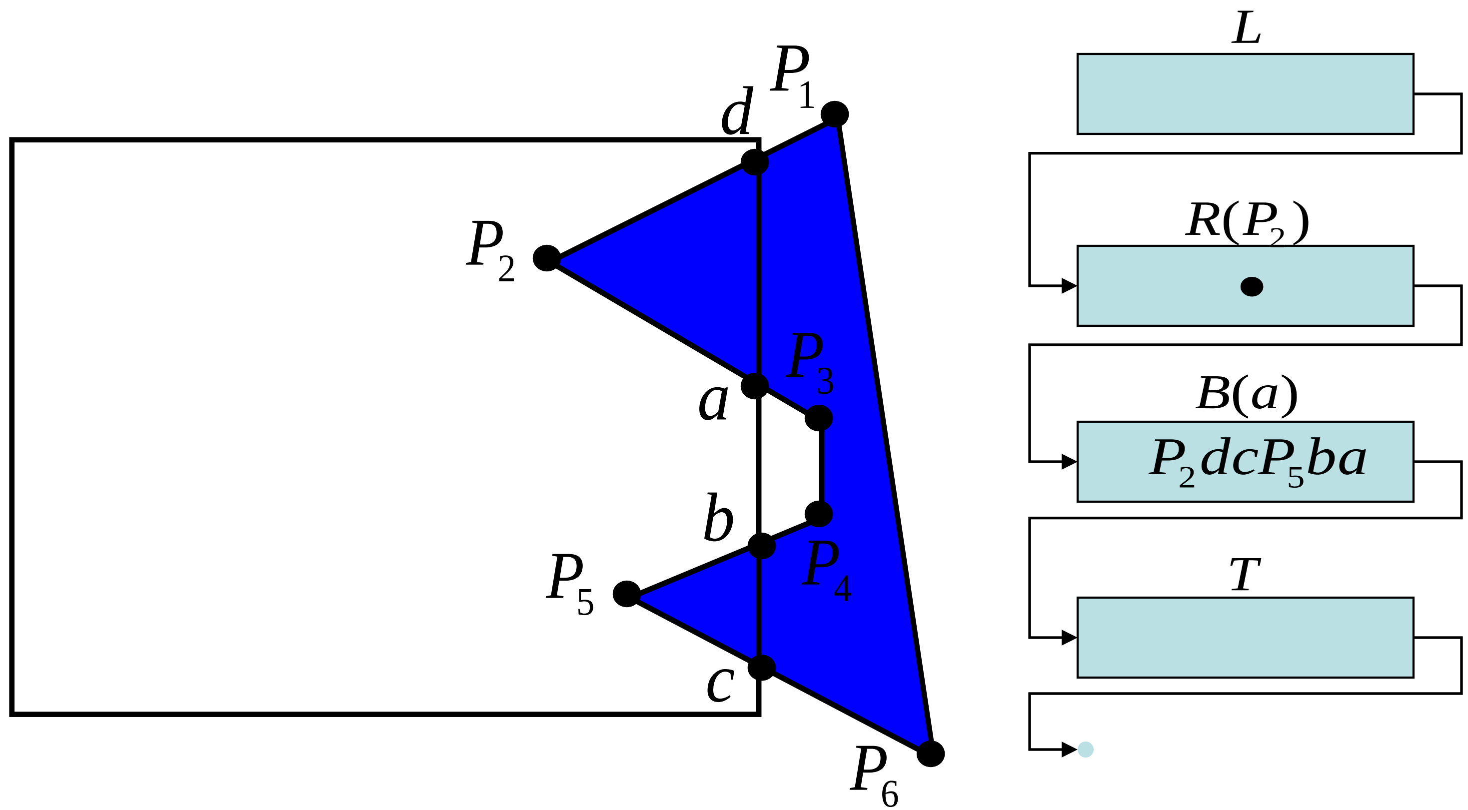
Sutherland-Hodgman Algorithm



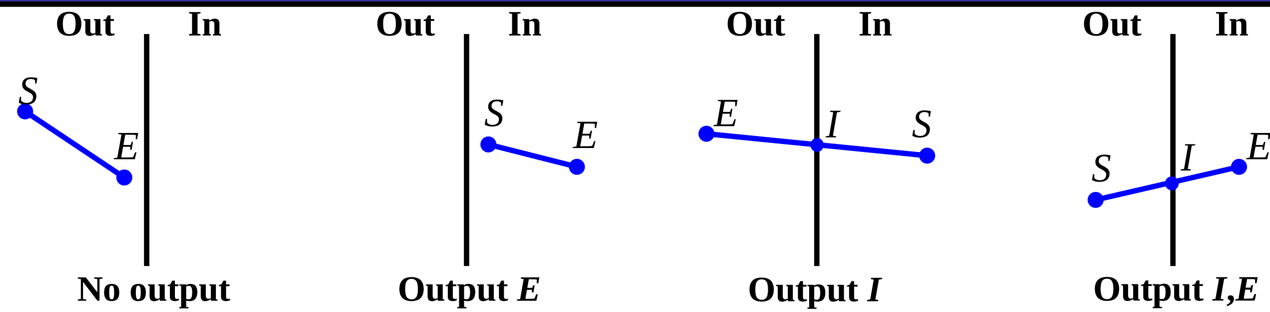


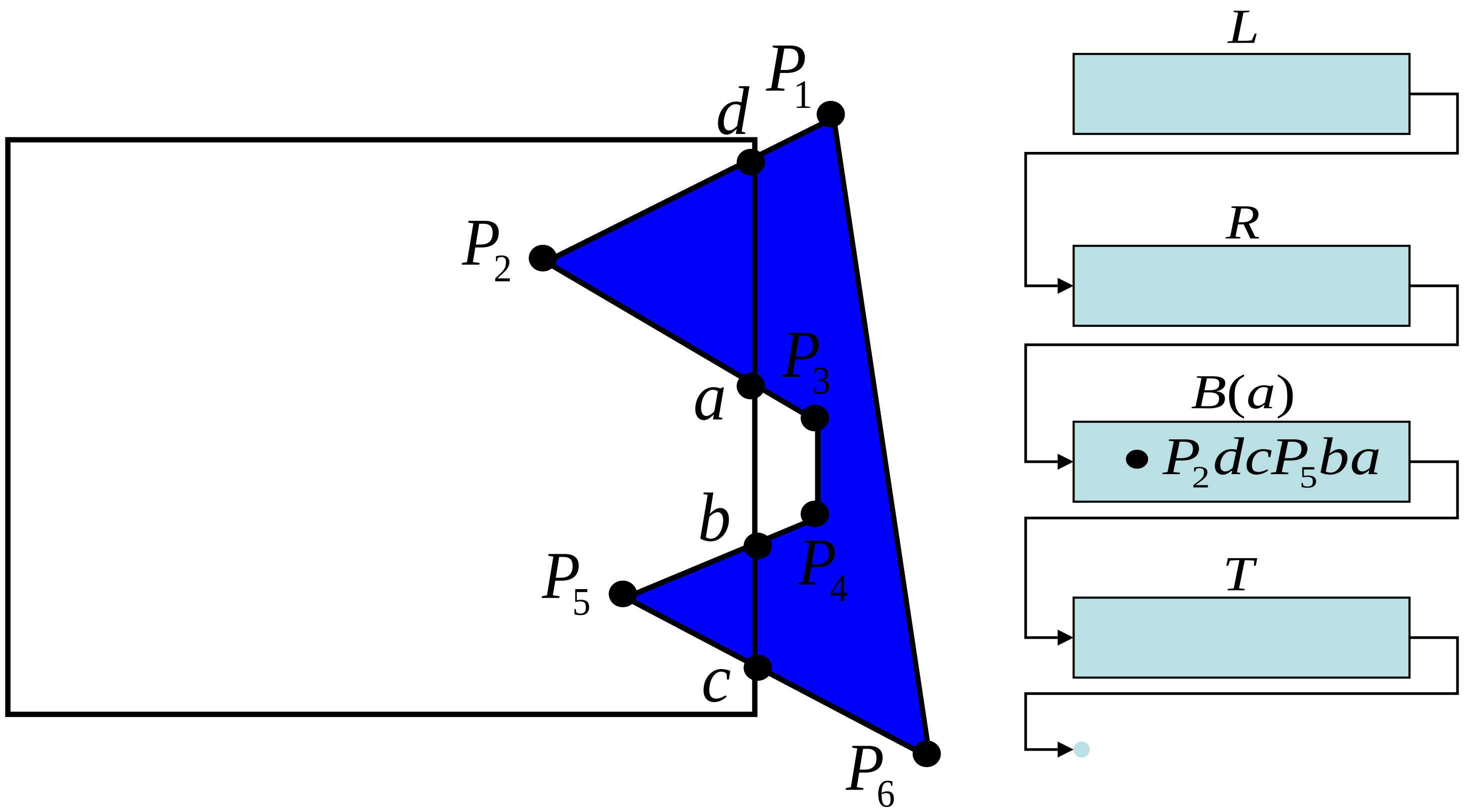
Sutherland-Hodgman Algorithm



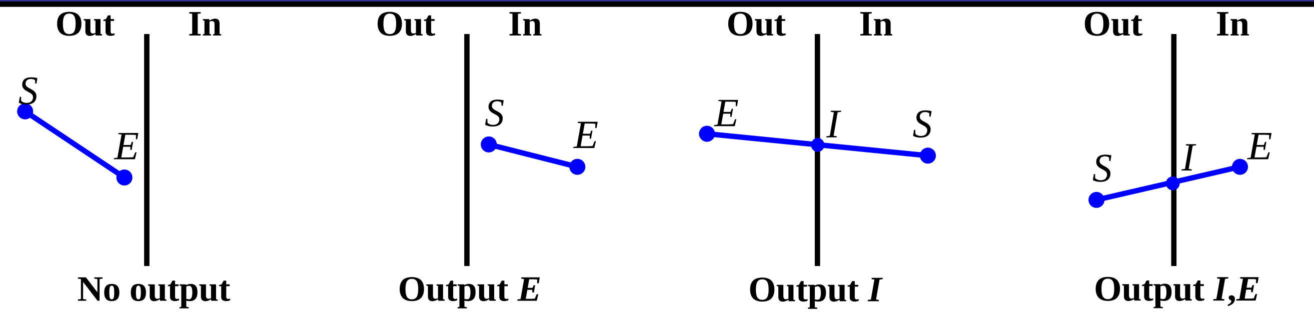


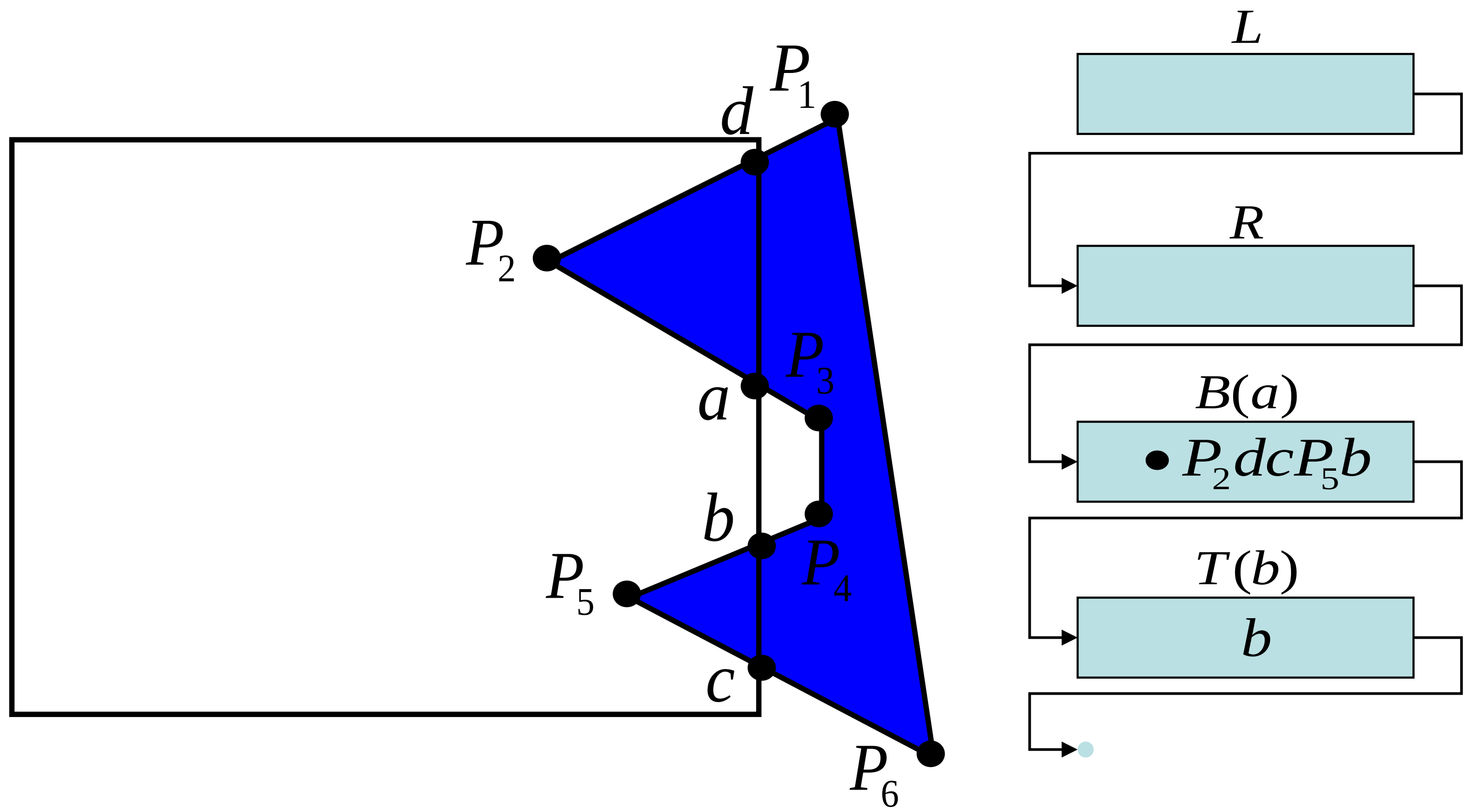
Sutherland-Hodgman Algorithm



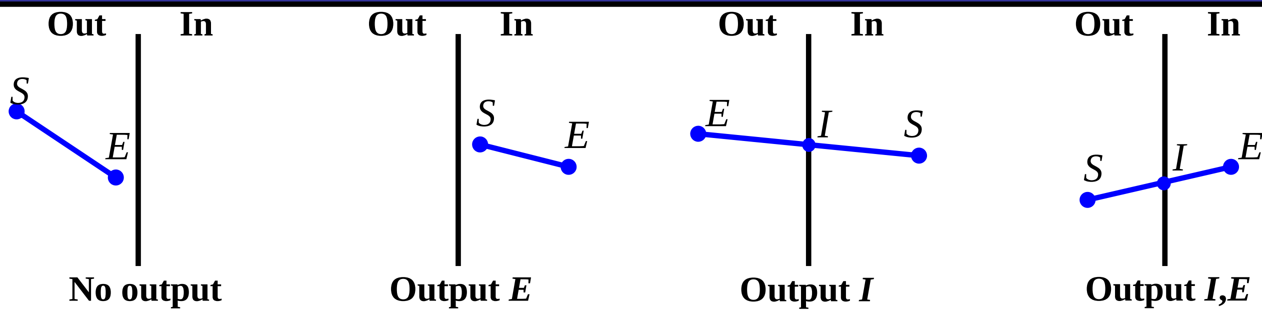


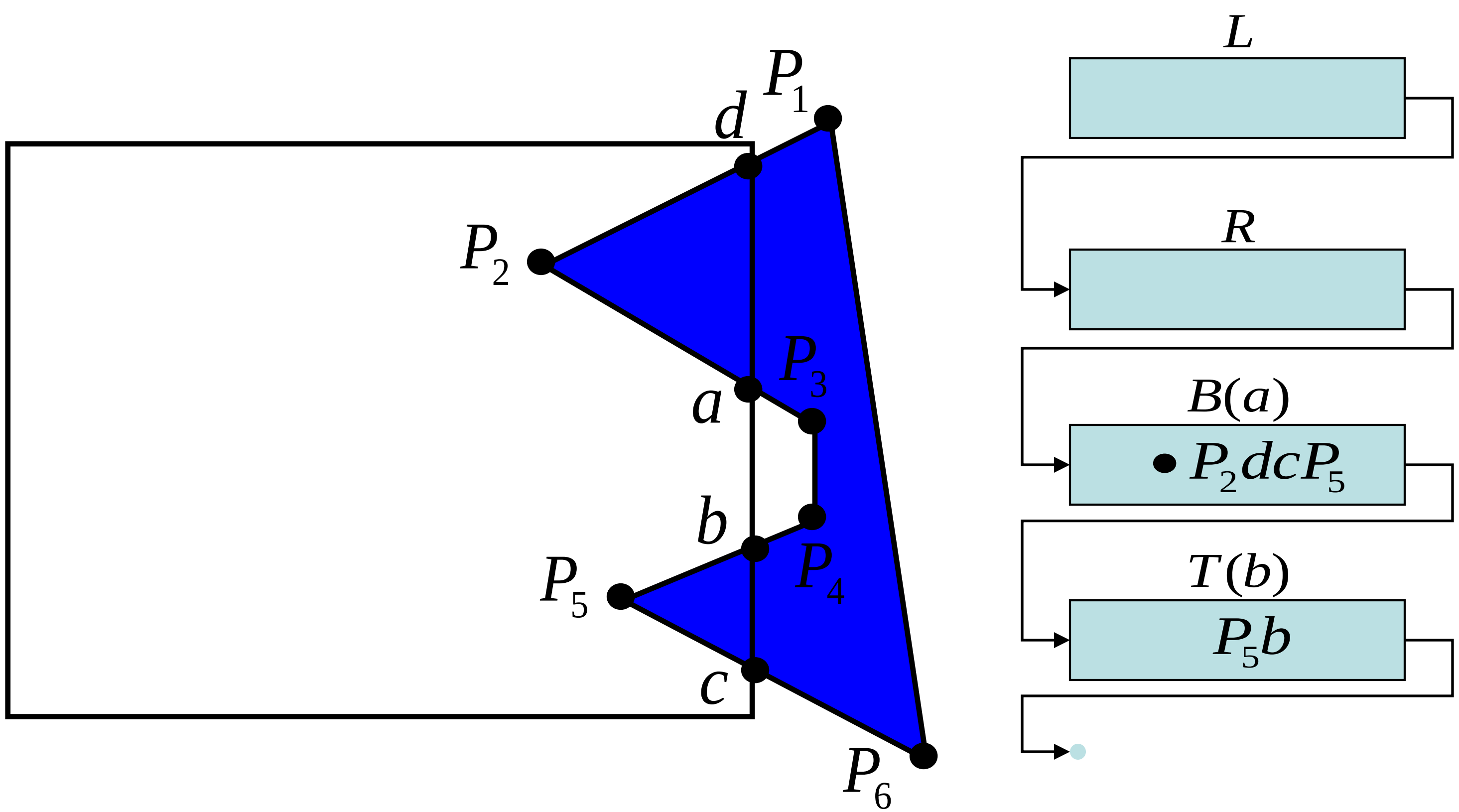
Sutherland-Hodgman Algorithm



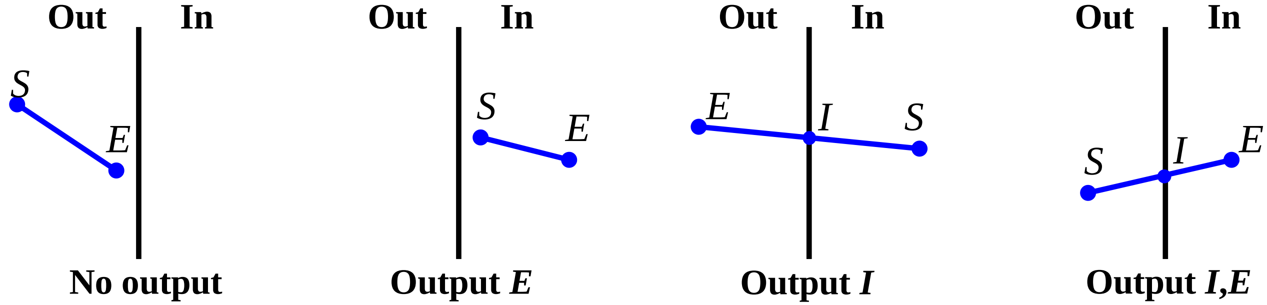


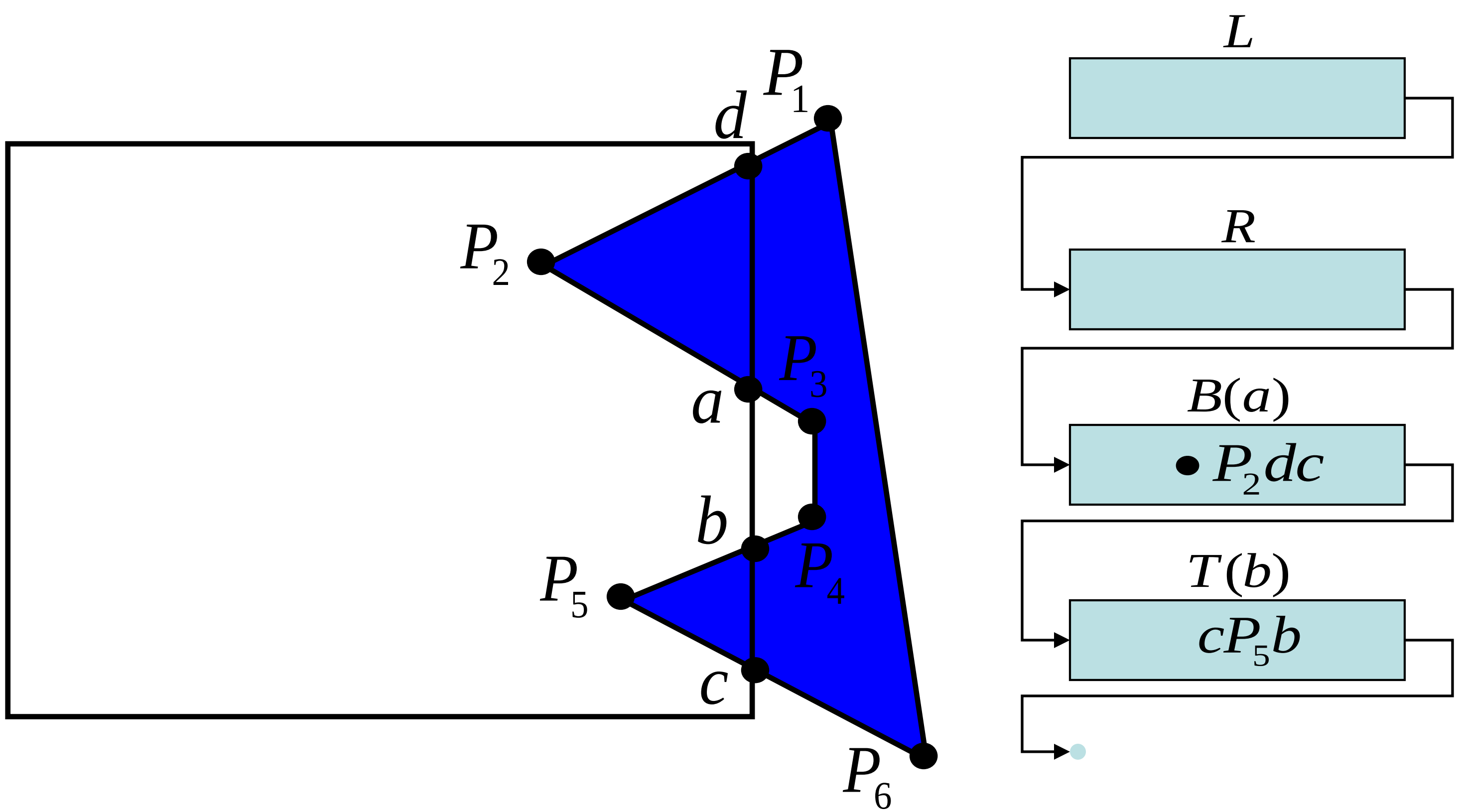
Sutherland-Hodgman Algorithm



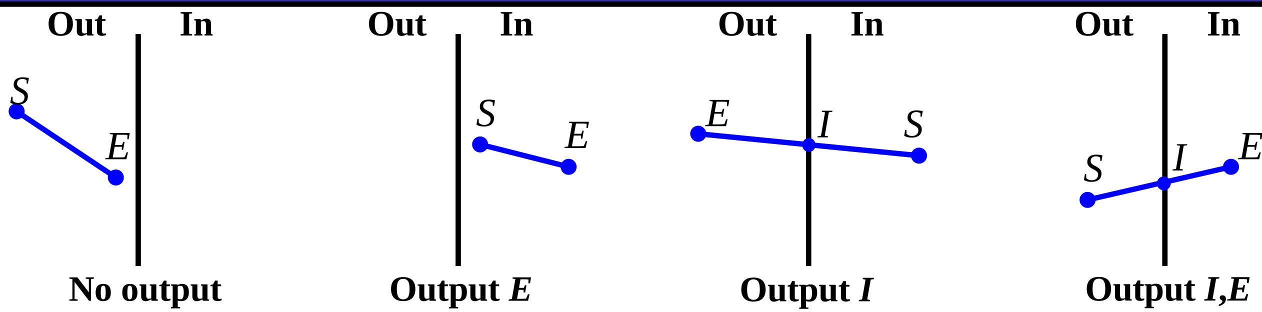


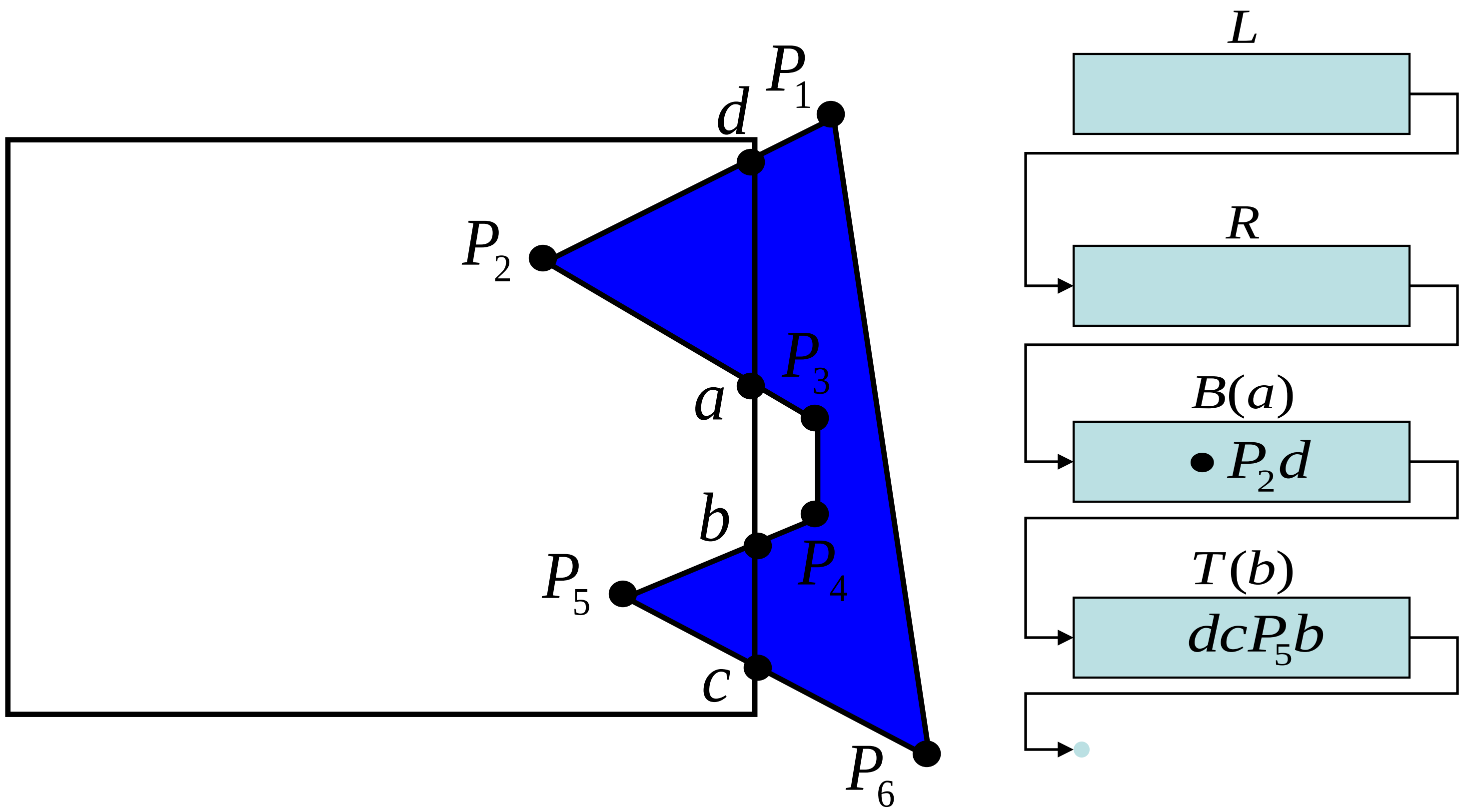
Sutherland-Hodgman Algorithm



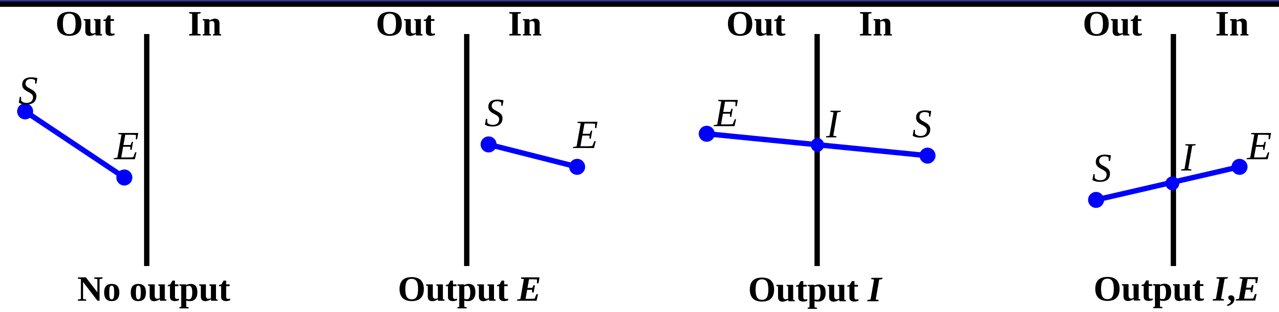


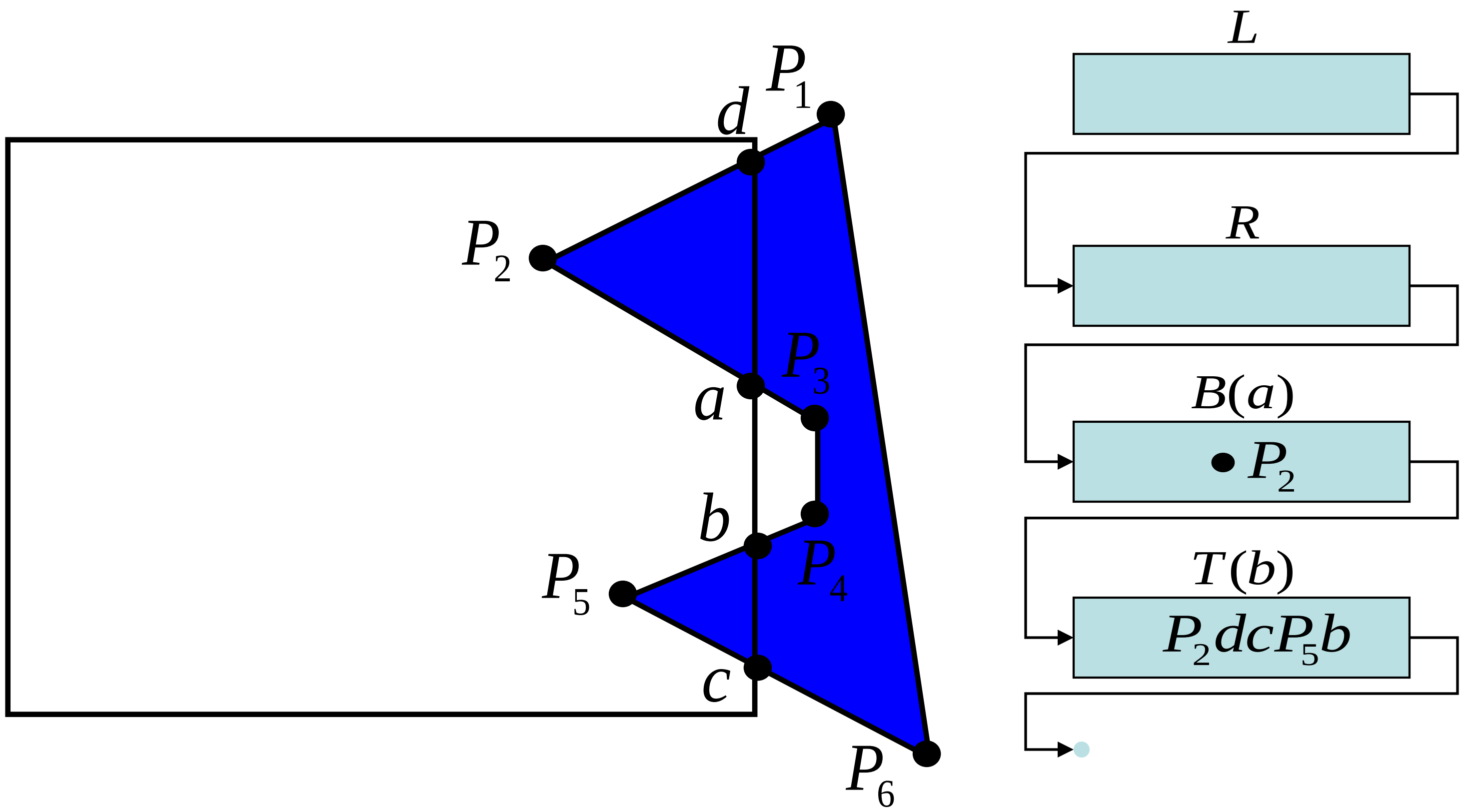
Sutherland-Hodgman Algorithm



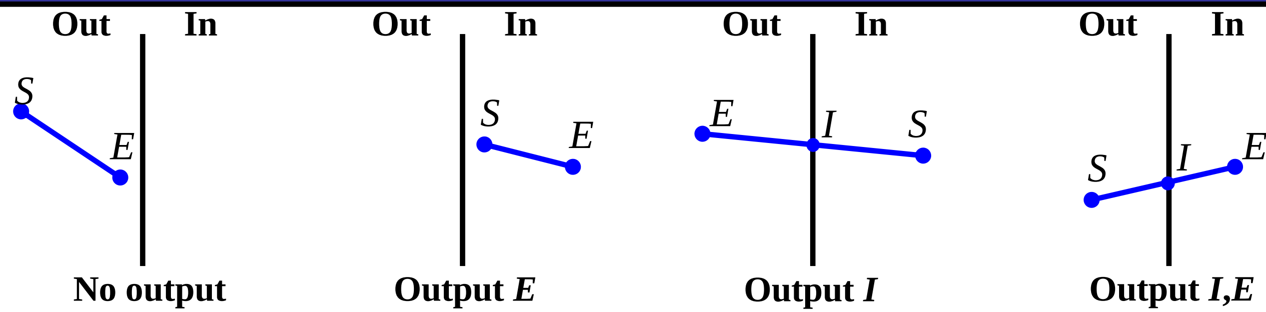


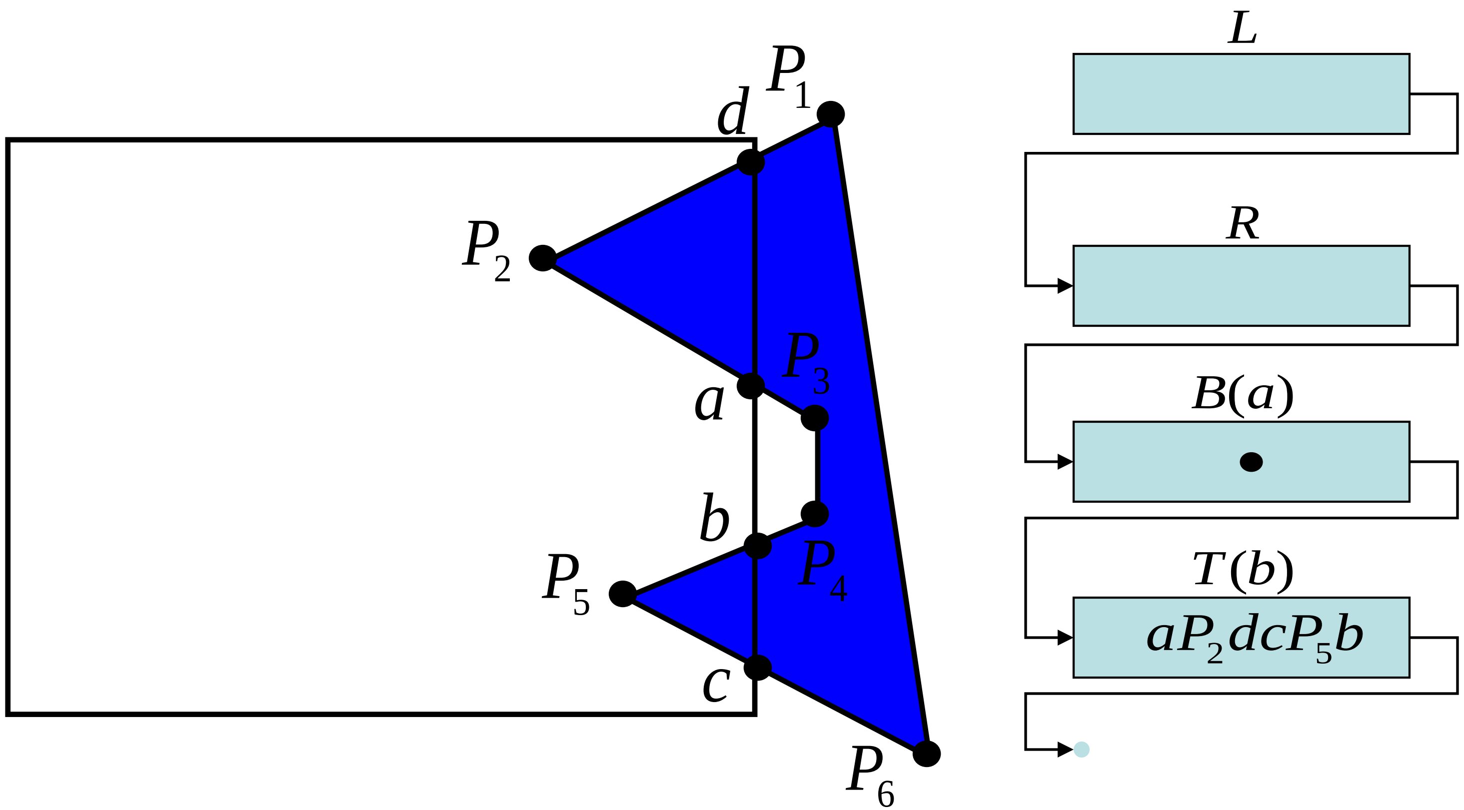
Sutherland-Hodgman Algorithm



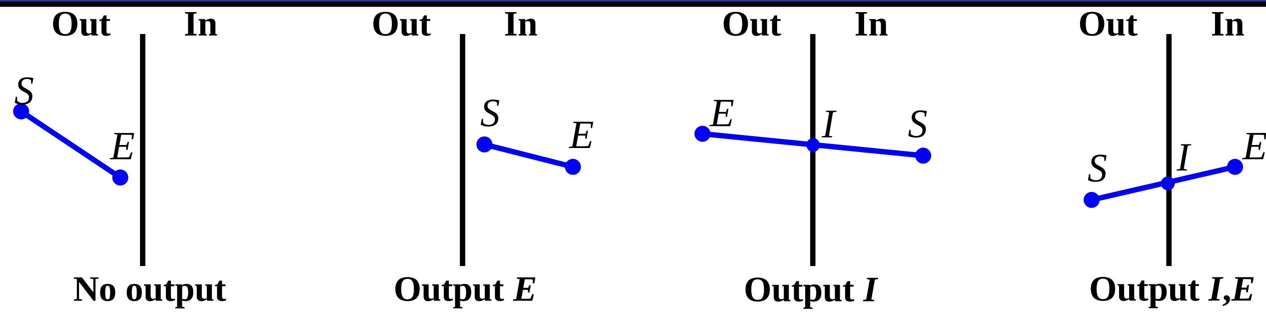


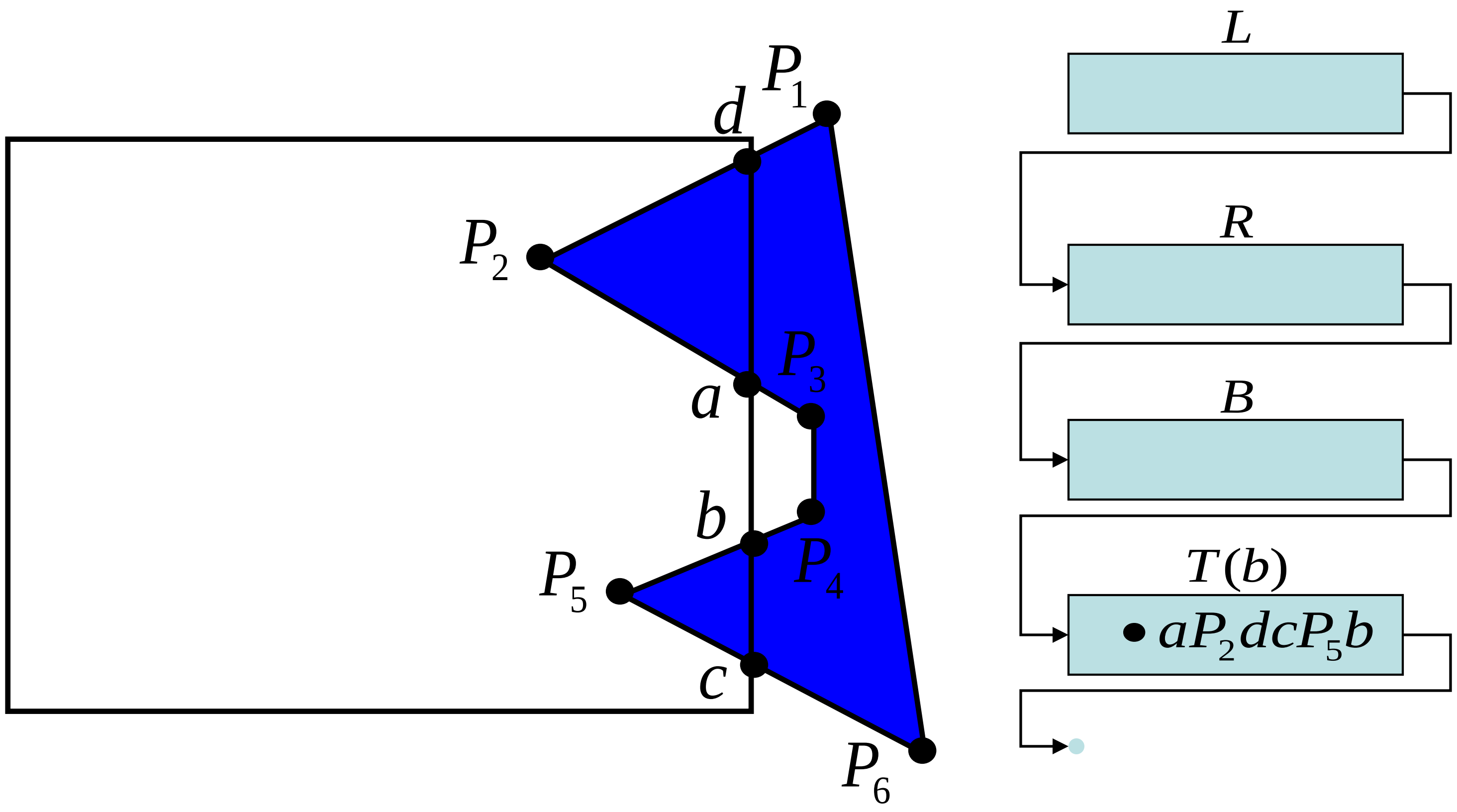
Sutherland-Hodgman Algorithm



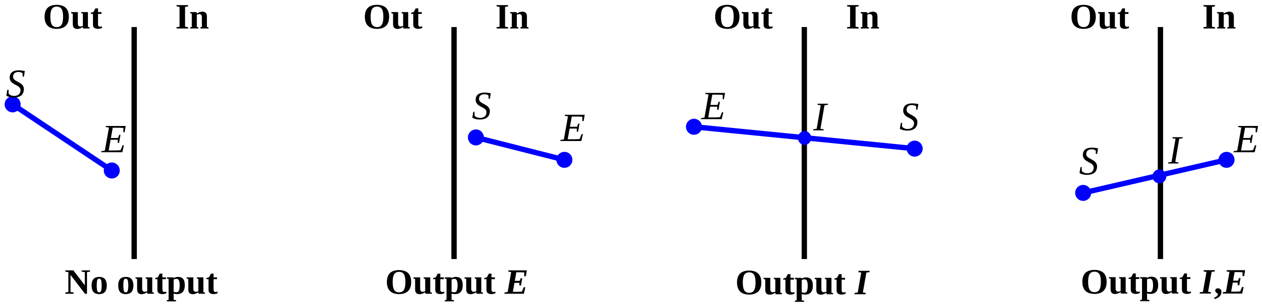


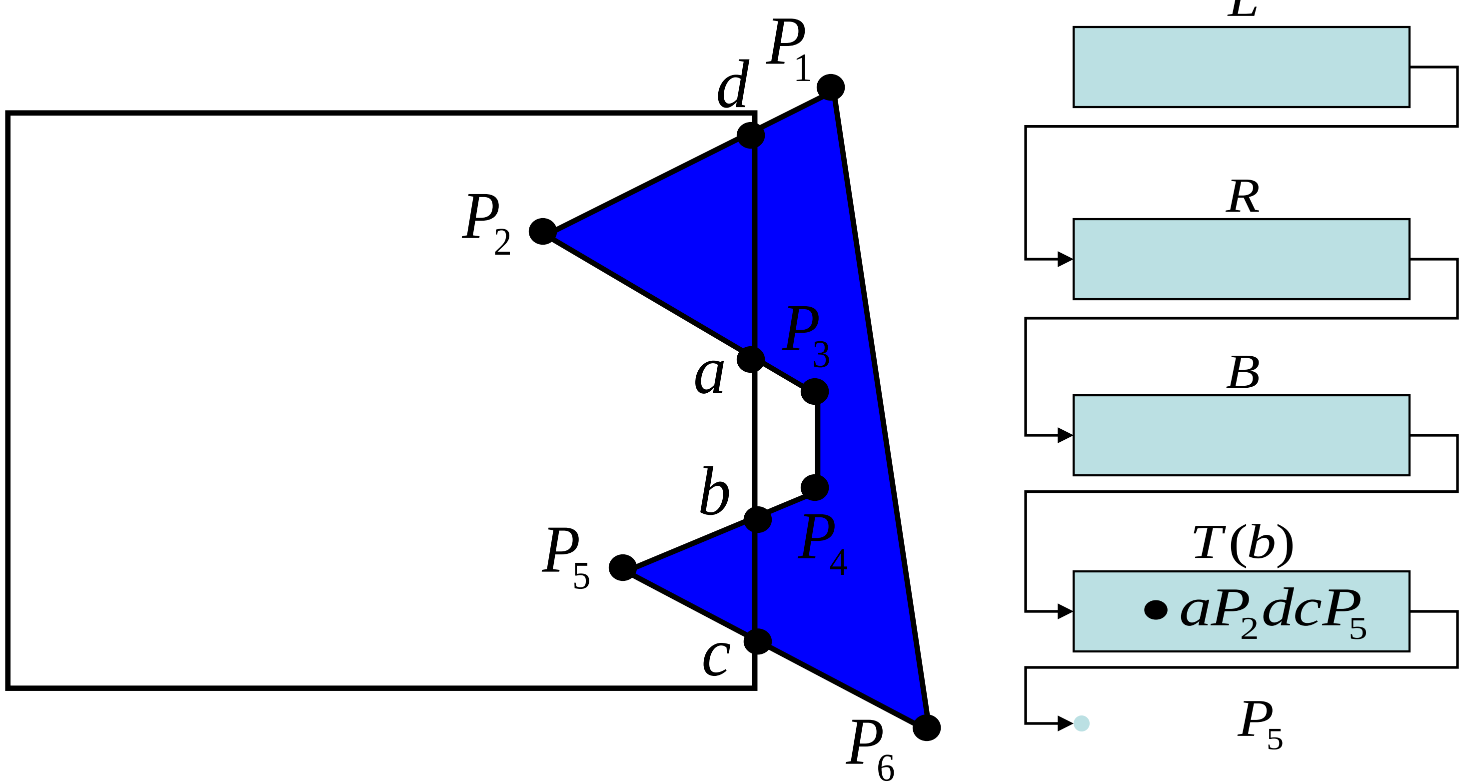
Sutherland-Hodgman Algorithm



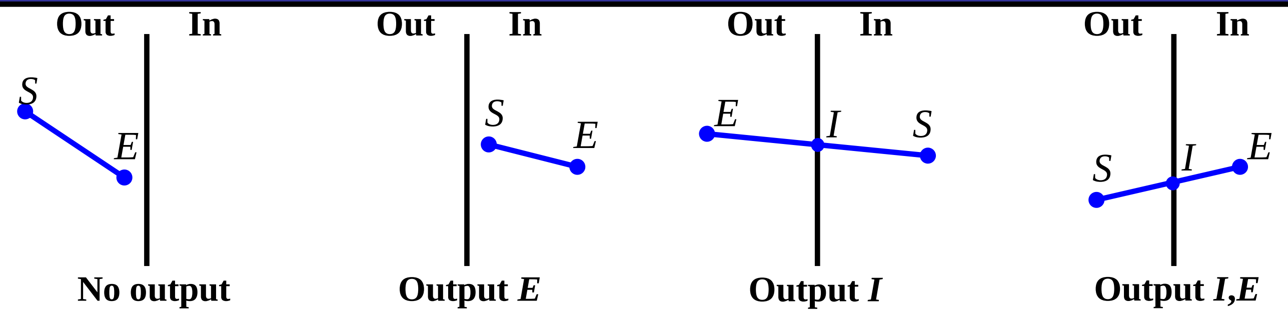


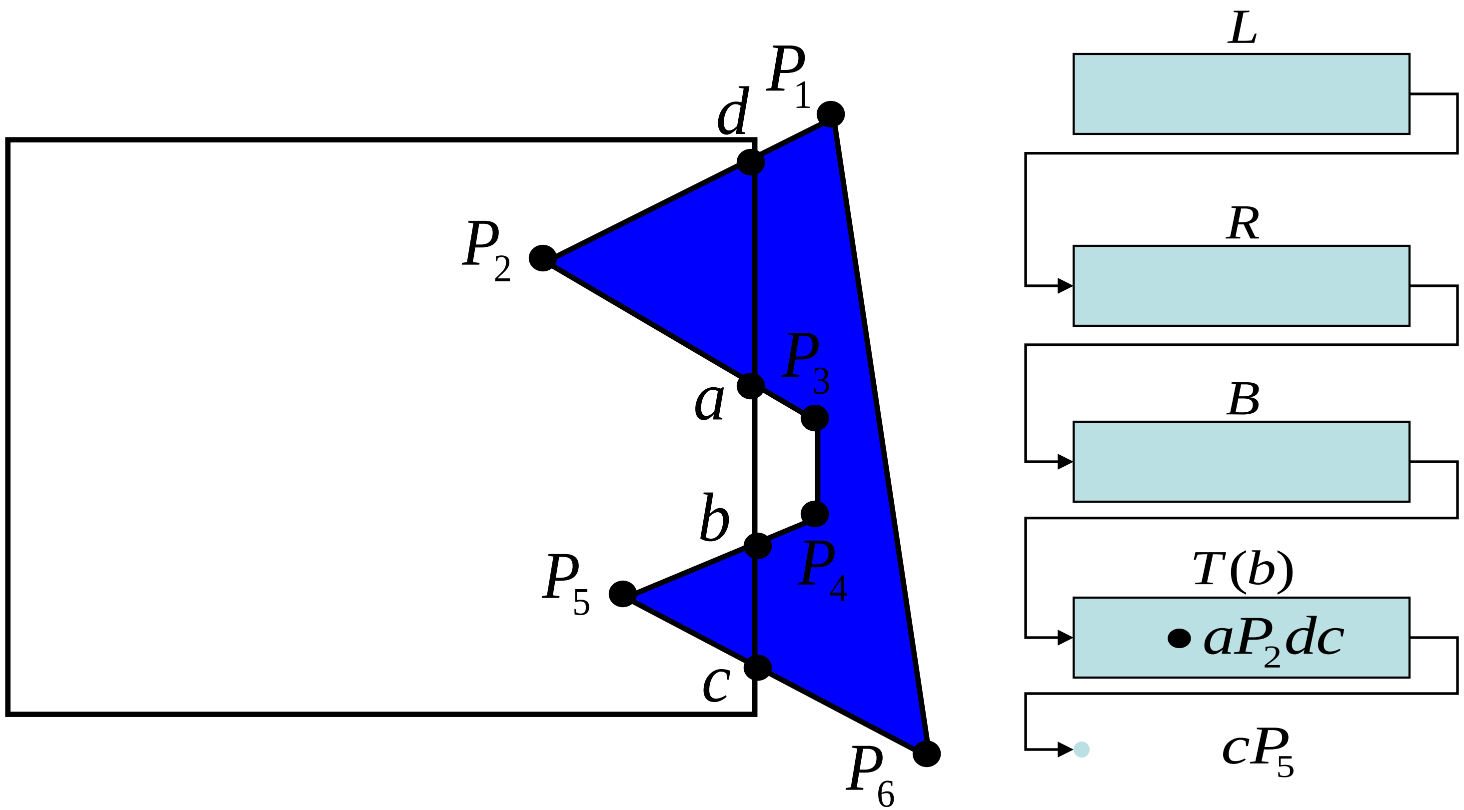
Sutherland-Hodgman Algorithm



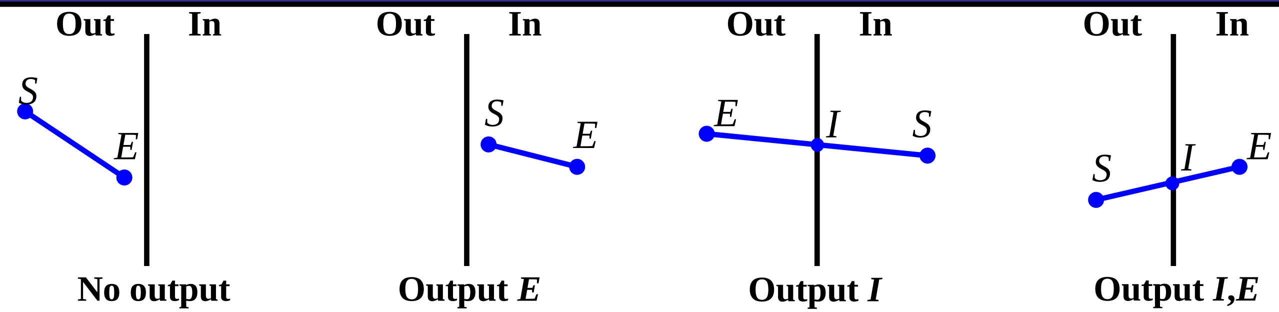


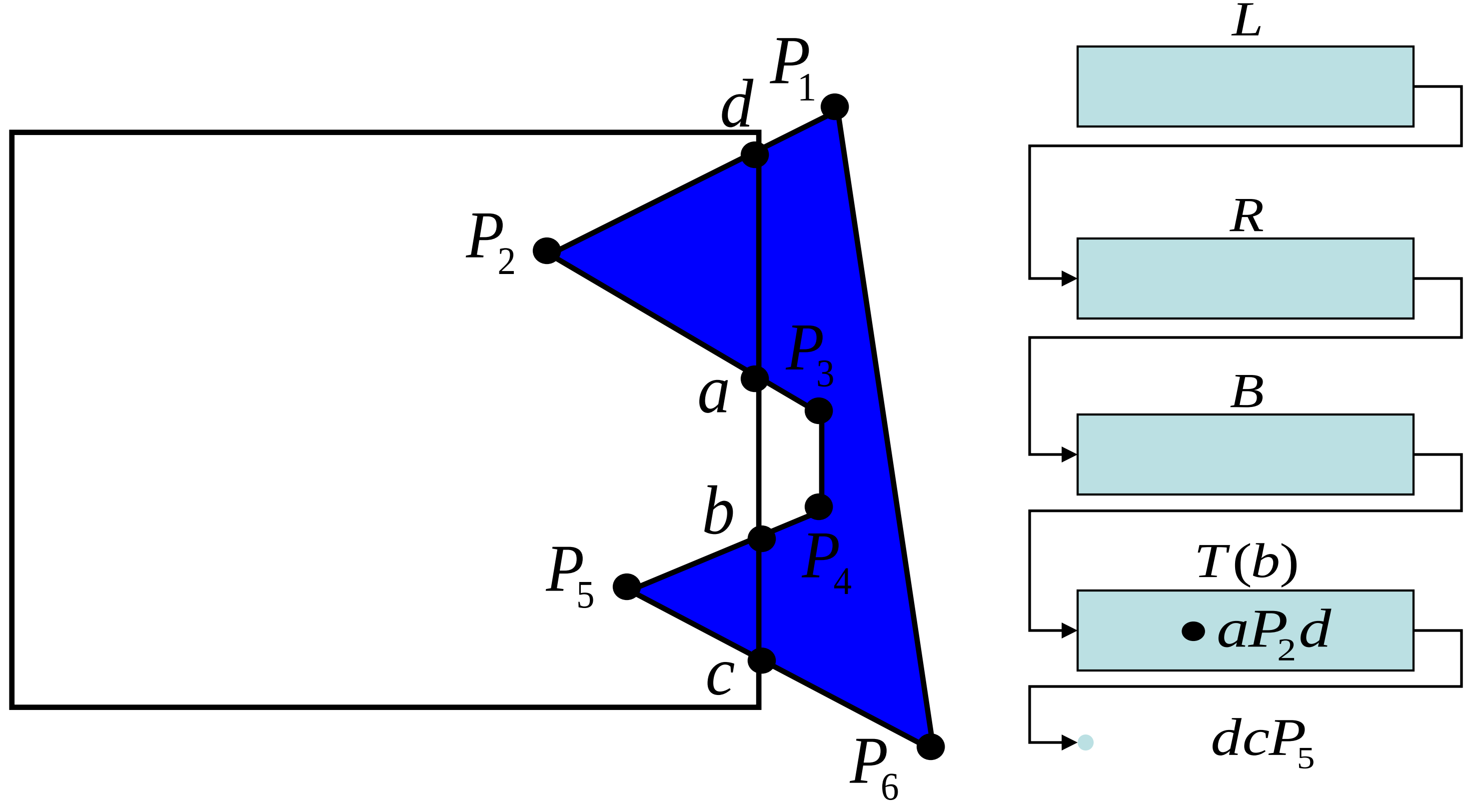
Sutherland-Hodgman Algorithm



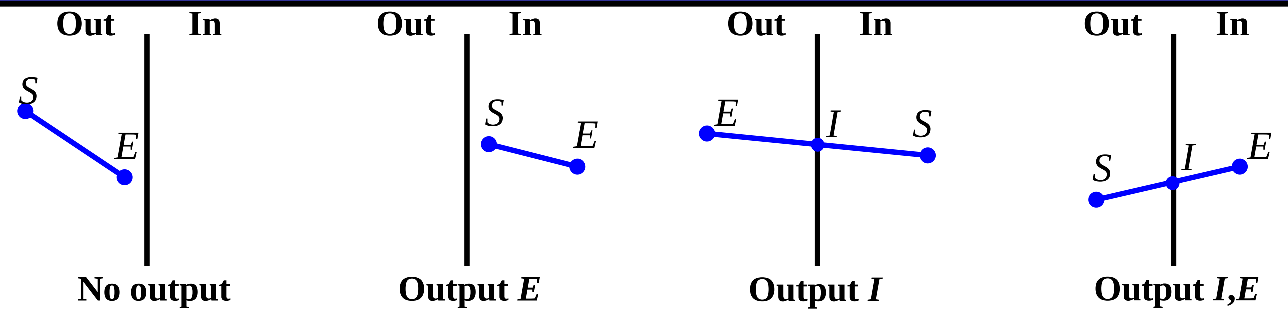


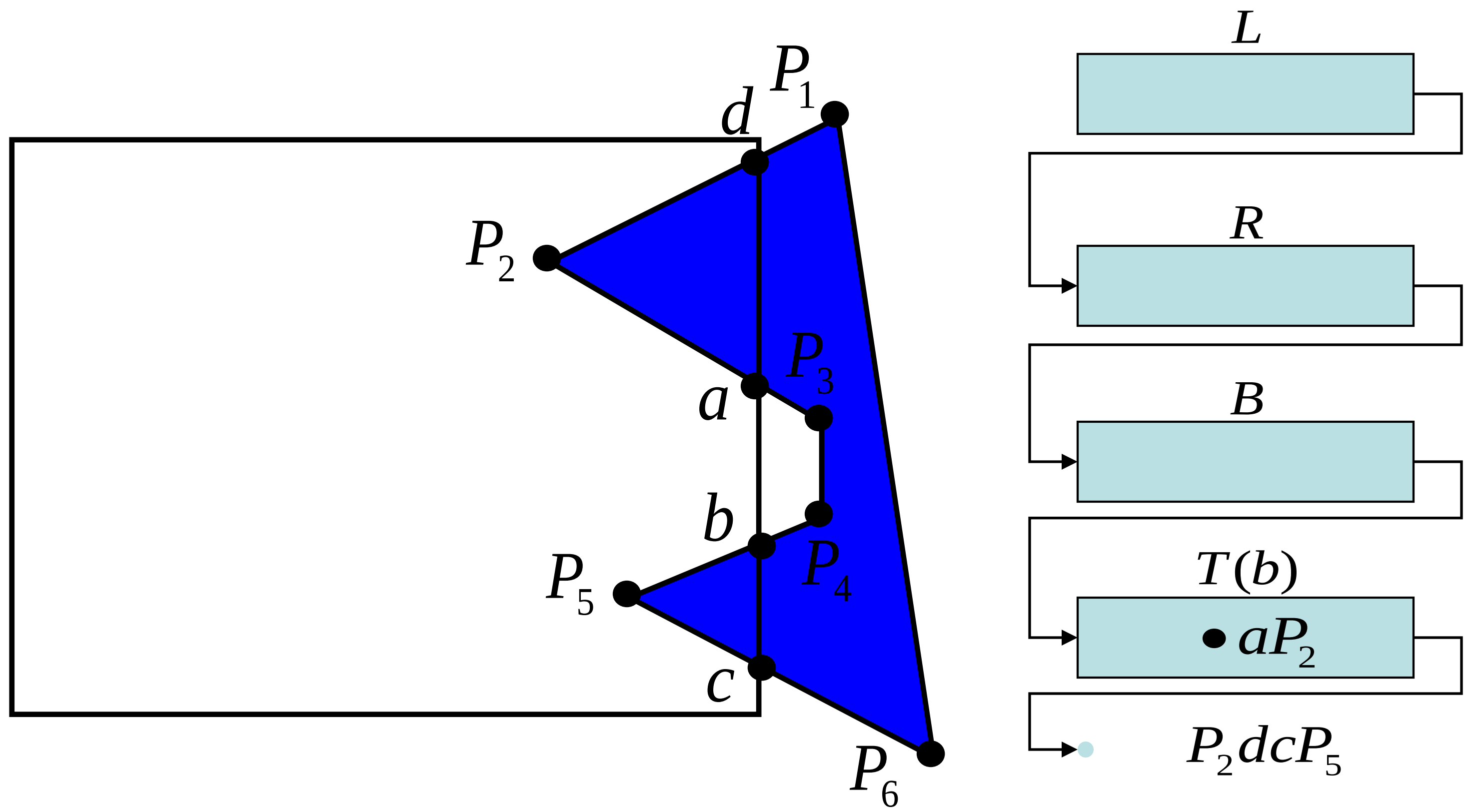
Sutherland-Hodgman Algorithm



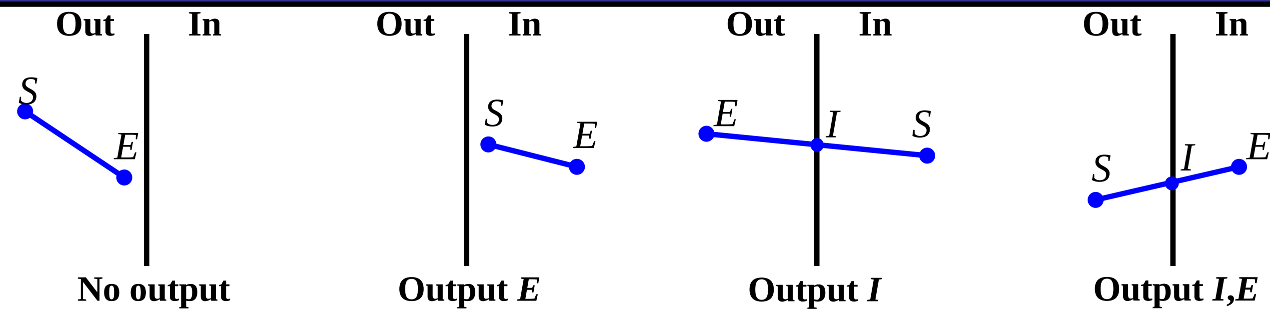


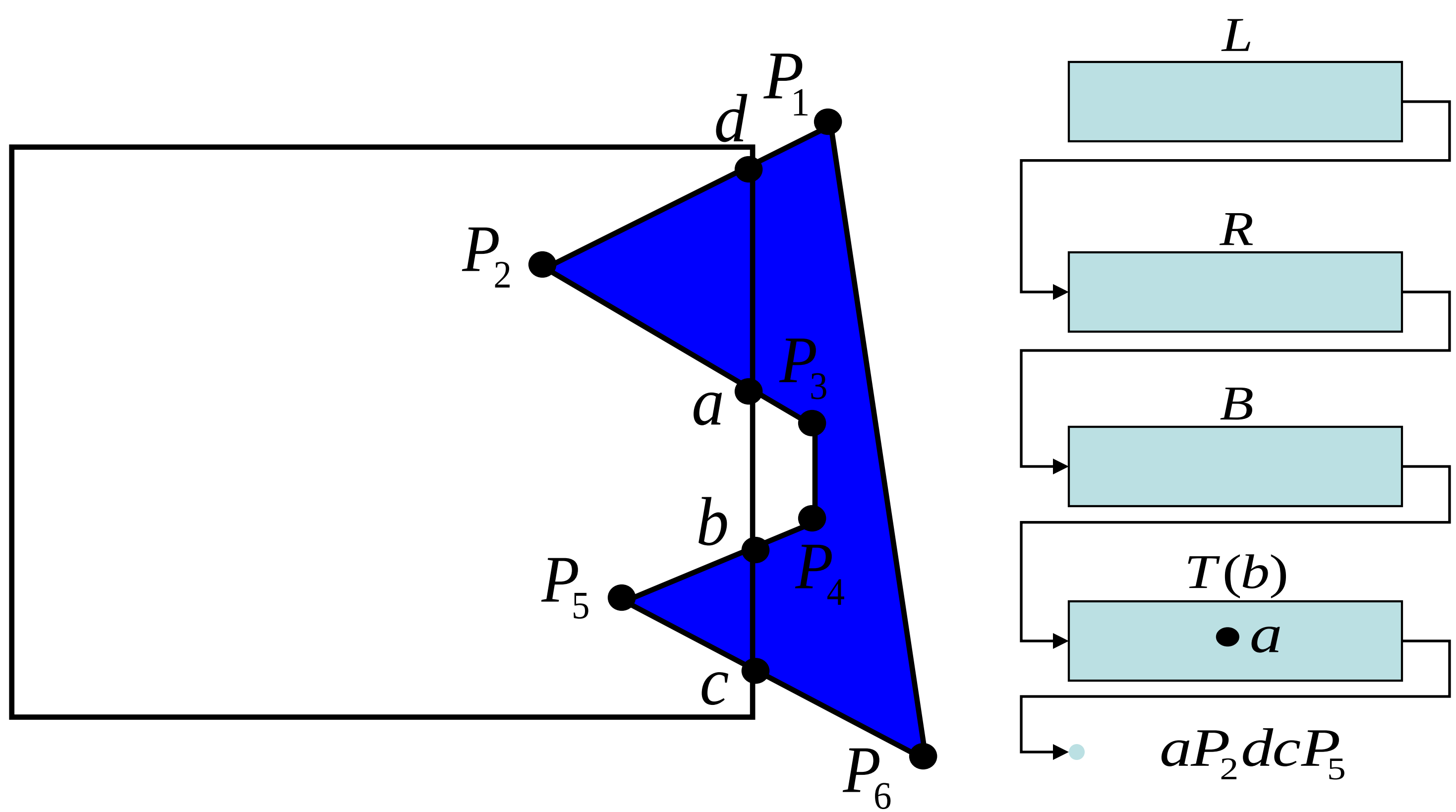
Sutherland-Hodgman Algorithm



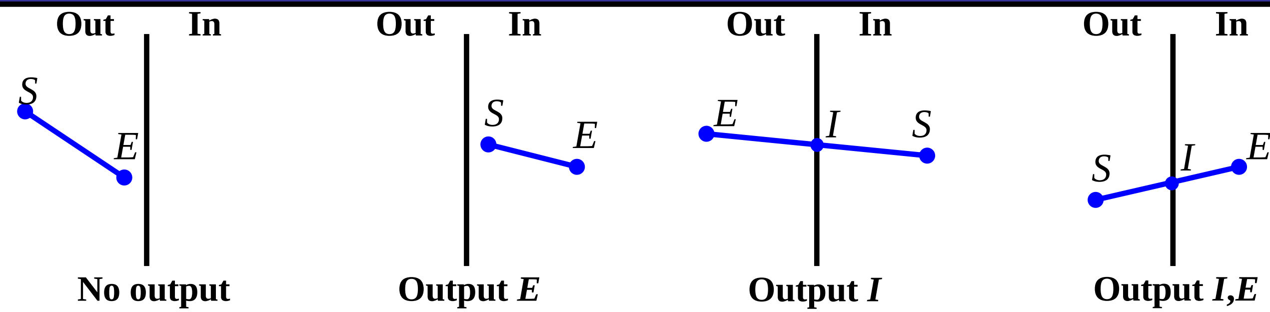


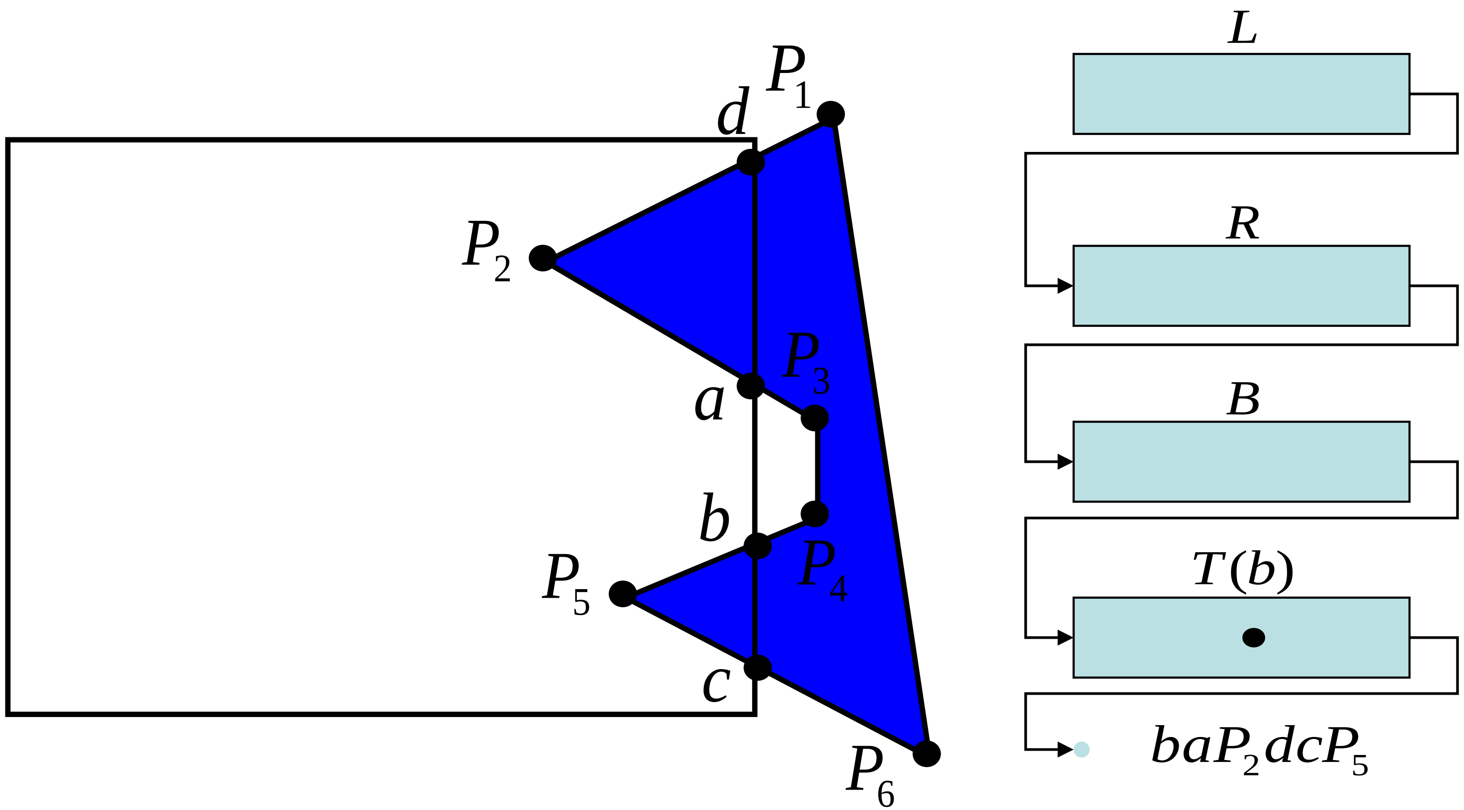
Sutherland-Hodgman Algorithm



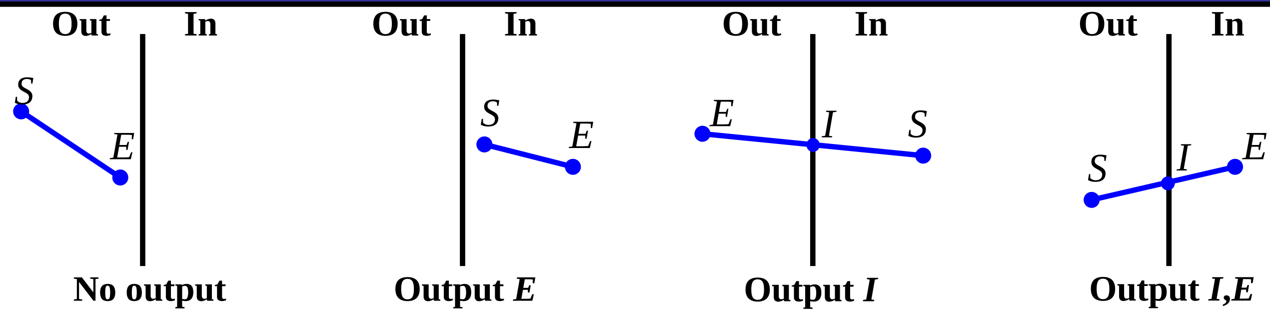


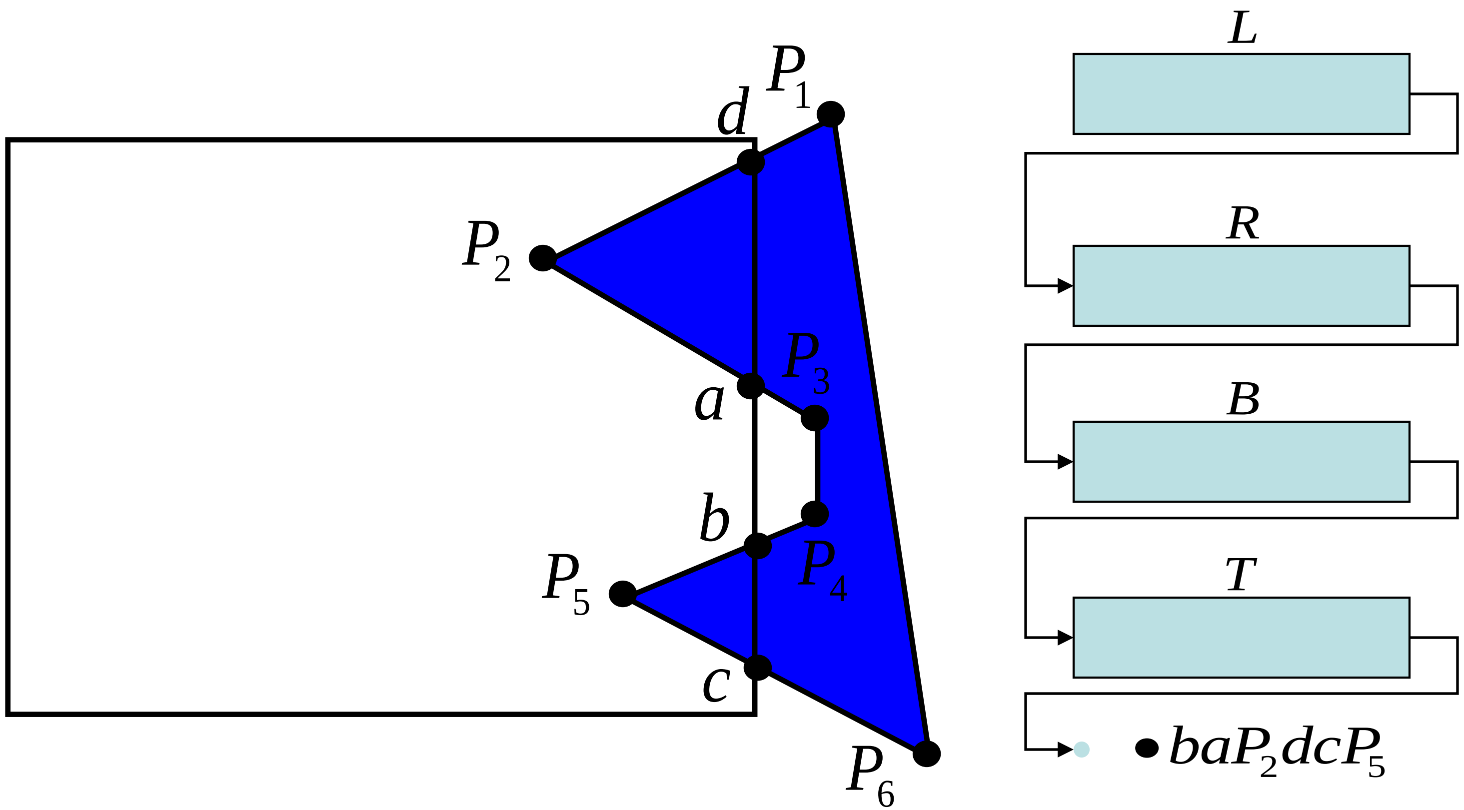
Sutherland-Hodgman Algorithm



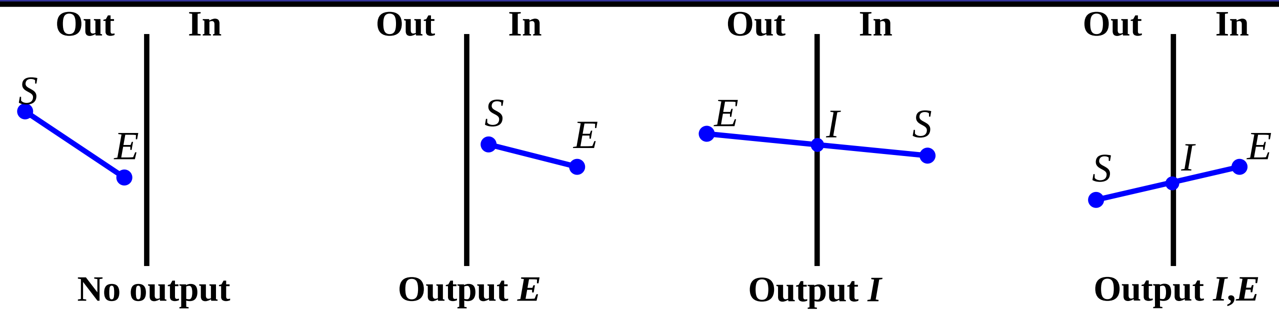


Sutherland-Hodgman Algorithm

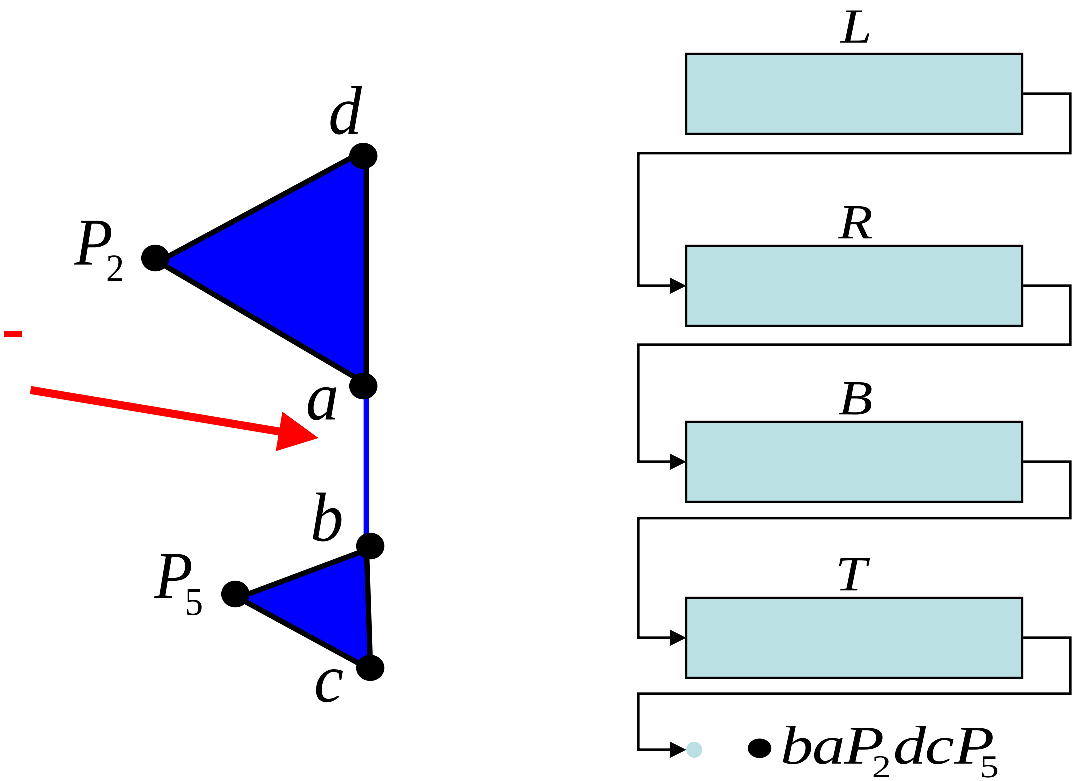




Sutherland-Hodgman Algorithm



Possible error for non-convex polygons



Sutherland-Hodgman Algorithm

Easy to pipeline for parallel processing

Polygon from one boundary does not have to be complete before next boundary starts

Leads to substantial performance gains

Summary

- Objects within a scene must be clipped to display the scene in a window
- Because there are can be so many objects clipping must be extremely efficient
- The Cohen-Sutherland algorithm can be used for line clipping
- The Sutherland-Hodgman algorithm can be used for area clipping