Architectural Home Plan Designing System with Knowledge Base Inspection

Knowledge Base Implementation

- Geometrical Rules
- Architectural Rules
- Legal Rules

Knowledge Base Implementation First Phase

Geometrical Rules

- Most difficult implementation
- It is necessary to define the basic shape of a house

Basic Geometrical Rules

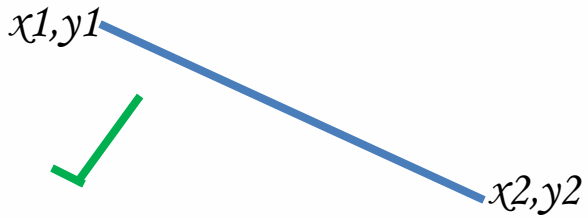
- Define a Line
- Define a Polygon
- Find adjacency of two polygons
- Find intersection of two polygons
- Find overlapping of two polygons

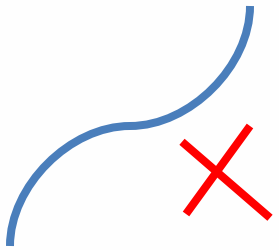
Define A Line

Line

```
% Definition to line
line([[X1,Y1],[X2,Y2]]) :- between(1,10,X1), between(1,10,X2),between(1,10,Y1),between(1,10,Y2).
```

Earlier the coordinates are defined as integers but it ends up the coordinate generation in an infinite loop. Therefore the between values are given.



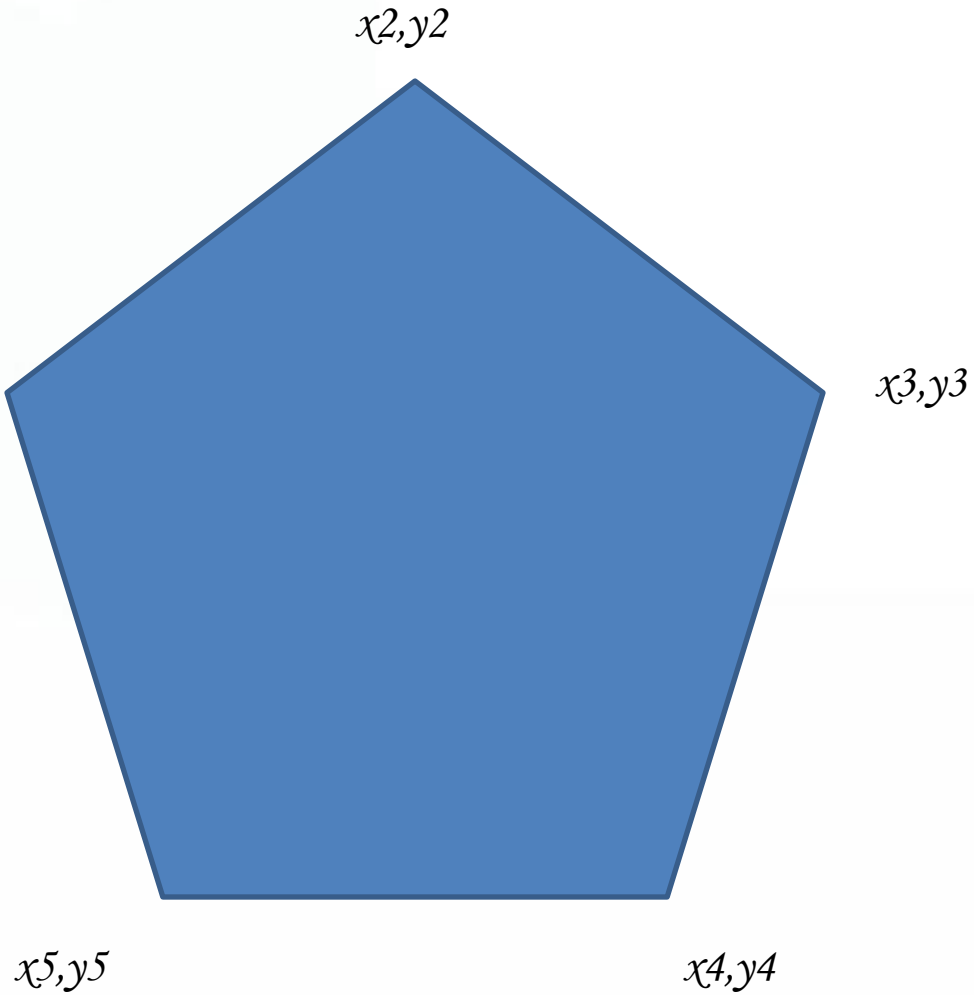


Define A Polygon

Polygon

1. Should have more than three lines.
2. First coordinates of a line should be the last coordinates of the previous line.
3. First coordinates of the first line should be equal to last coordinates of the last line.

Define a Polygon

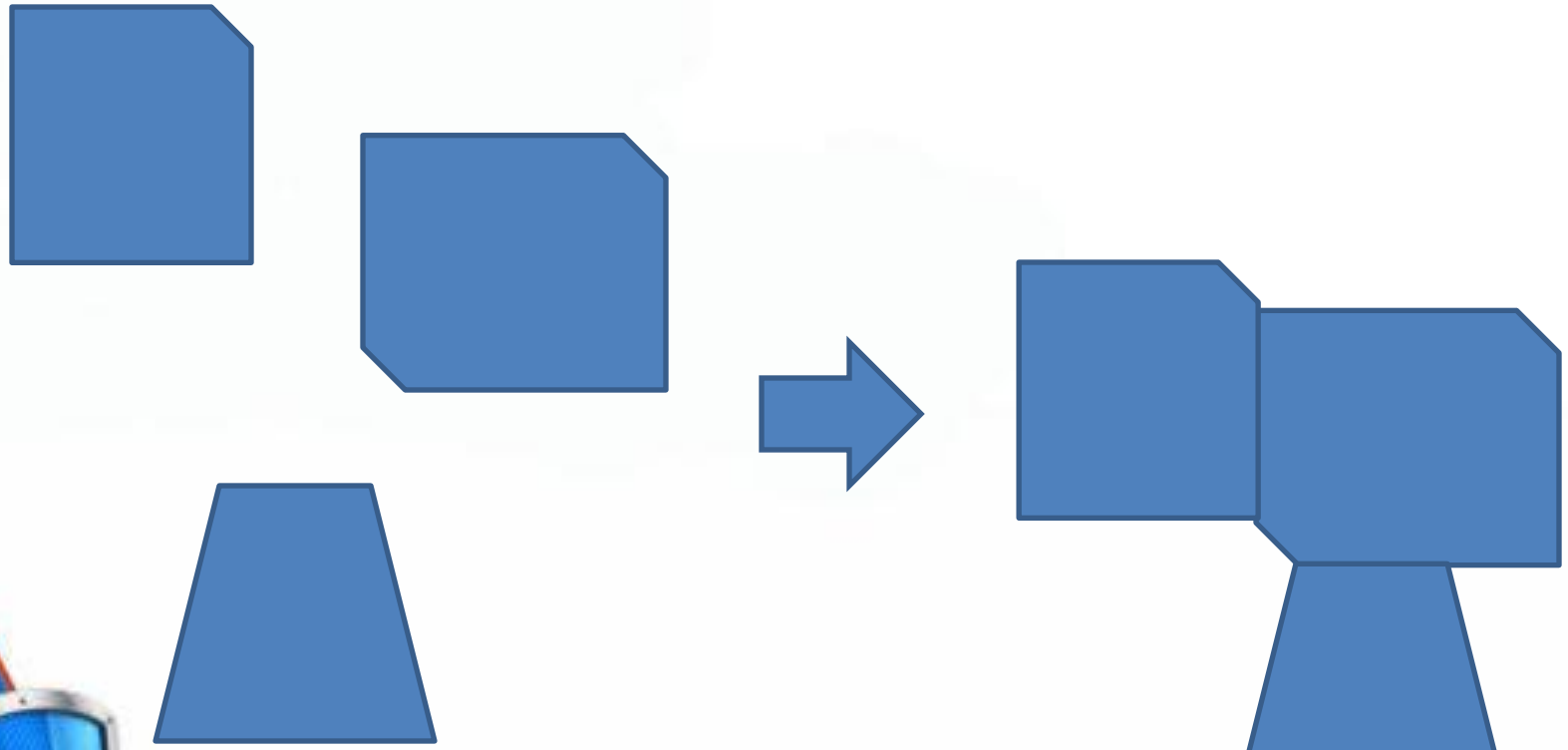


Define A Polygon

```
% Definition to polygon

polygon(X) :- isClosed(X), isConnected(X).

        isClosed([[X1,Y1]|L]) :- last(L, [X2,X1]).

        isConnected([[X1,Y1],[Y1,Z1],[Z1,U1]]):- line([X1,Y1]),line([Y1,Z1]),line([Z1,U1]).

        isConnected([[X1,Y1]|[[Y1,Z1]|L]]):- isConnected([[Y1,Z1]|L]), line([X1,Y1]),line([Y1,Z1]).
```

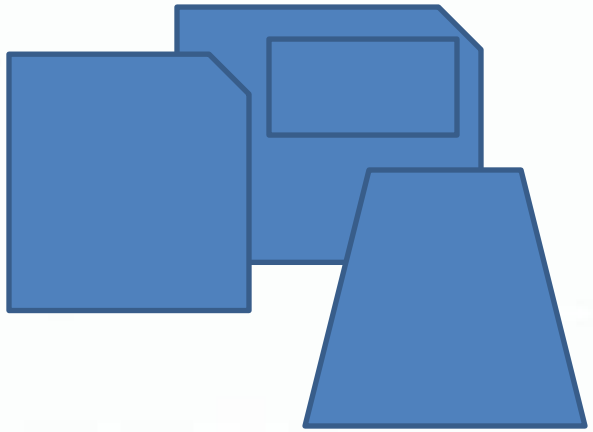
Just a set of polygon does not represent a house

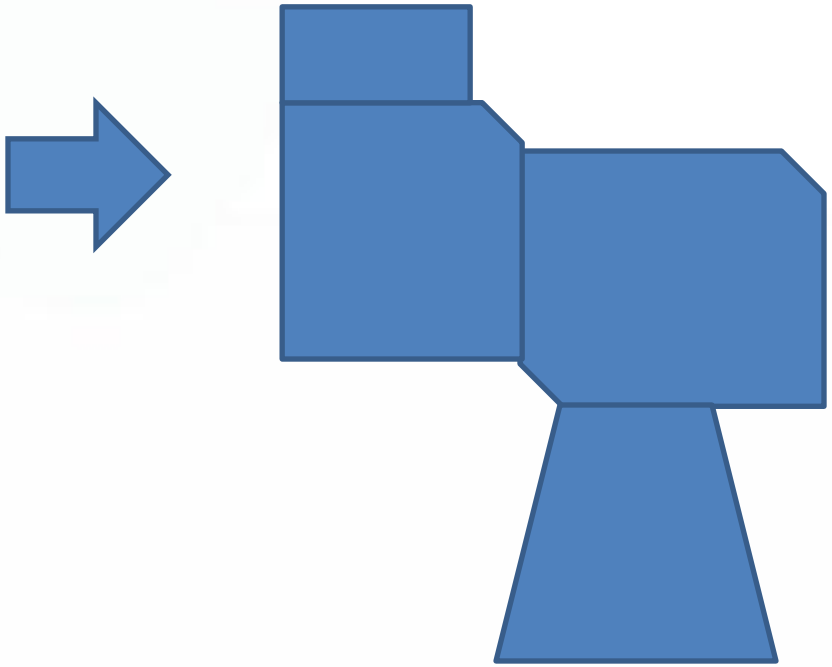
They should be interconnected

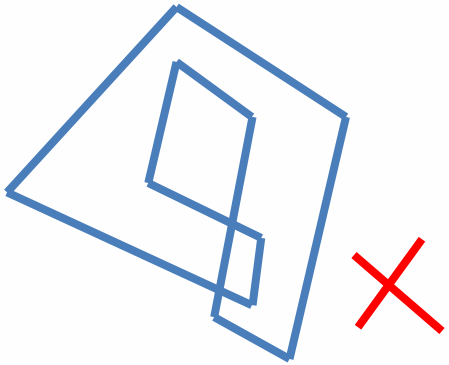


Just a set of polygon does not represent a house

They should be non overlapping







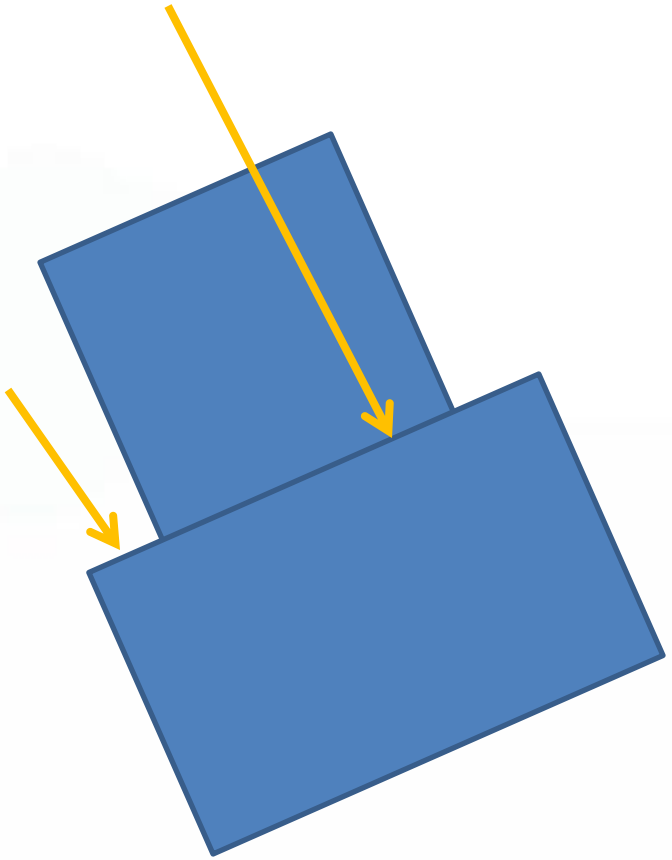
Find adjacency of two polygons

Find a line common to two polygons

1. Find the slope of line1 in polygon1-> m1

2. Find the slope of line2 In polygon2 -> m2

3. Validate m1 = m2



Find adjacency of two polygons

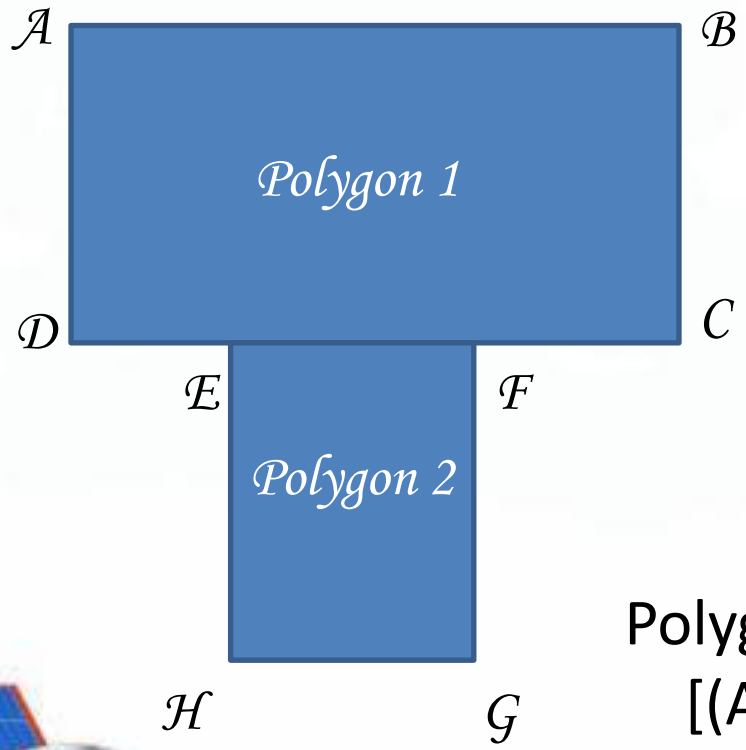
Find a line common to two polygons

- Method
  - ✓Find the slope of lines
- Issues
  - ✓Finding slope needs mathematical calculations like division
  - ✓Prolog does not support reverse calculations for division
- Prolog Code

```
adjacentLine([[X1,Y1],[X2,Y2]],[[X3,Y3],[X4,Y4]], T1, T2) :- T1 is (Y2-Y1)/(X2-X1), T2 is (Y4-Y3)/(X4-X3), T1 is T2, (Y1-Y3)/(X1-X3) = T1, Sqrt(2,(((X1-X2)*(X1-X2))+((Y1-Y2)*(Y1-Y2)))) = (Sqrt(2, (((X1-X3)*(X1-X3))+((Y1-Y3)*(Y1-Y3)))) + Sqrt(2, (((X3-X2)*(X3-X2) + ((Y3-Y2)*(Y3-Y2))))))).
```

Find adjacency of two polygons

Solution



Polygon 1
[(A,B),(B,C),(C,D),(D,A)]

Polygon 2
[(E,F),(F,G),(G,H),(H,E)]

Polygon 1
[(A,B),(B,C),(C,F),(F,E),(E,D),(D,A)]

Polygon 2
[(E,F),(F,G),(G,H),(H,E)]

Find adjacency of two polygons

Prolog Code

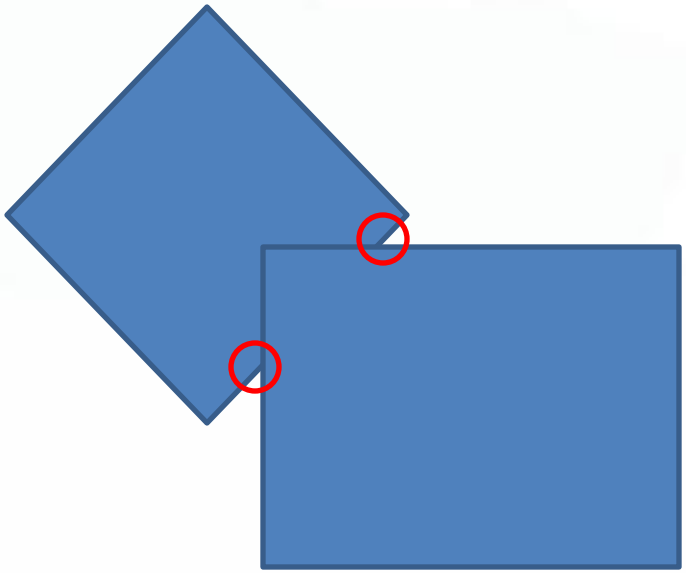
- Solution
  - ✓Break a polygon line to several lines at the common region of a line

```
% Find the adjacency of two lines

adjacent(X,Y) :-( findNth(X,[A,B]), findNth(Y,[A,B])) ; ( findNth(X,[A,B]), findNth(Y,[B,A]) ).

        % To find a member variable
        findNth([X|T],X) .
        findNth([H|T],X) :- findNth(T,X).
```

Find intersection of two polygons

Find a common point shared by two polygons

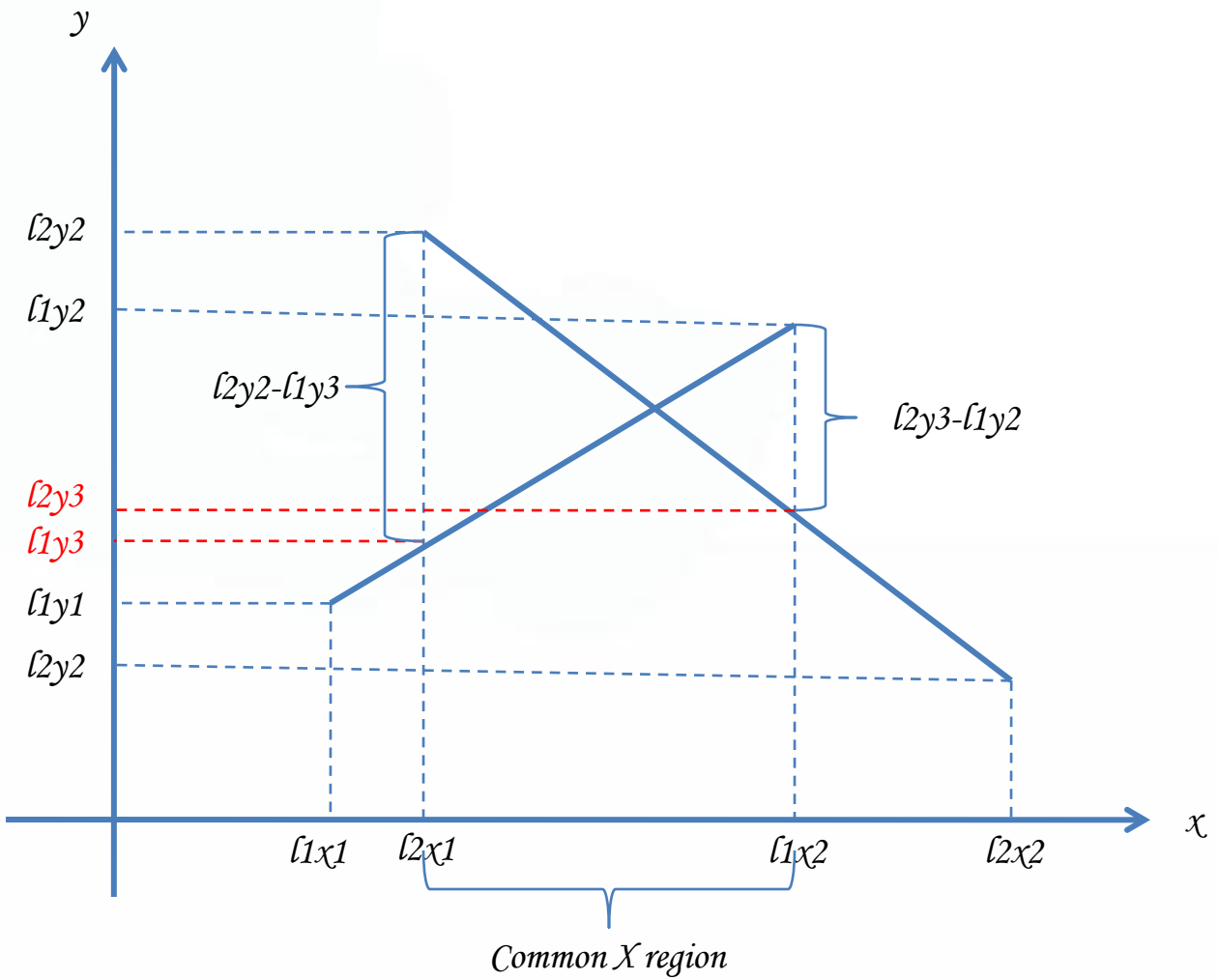
- Method
  - ✓Find intersection of two lines



Find intersection of two polygons (cont)

- Solution 1
  - ✓Get the common X regions of two lines and get the difference of Y coordinates of two lines at the common X points and the difference should be greater than zero if lines are not intersecting.
- Issues
  - ✓When prolog generates the coordinates of the lines, it generates the coordinates without following a particular order. To apply coordinates to above function coordinates should be in a particular order. Sorting the coordinates of a line in prolog is really hard since a coordinate is coupled with it's X and Y values.

Find intersection of two polygons Solution 1(cont)



Find intersection of two polygons Solution 1(cont)

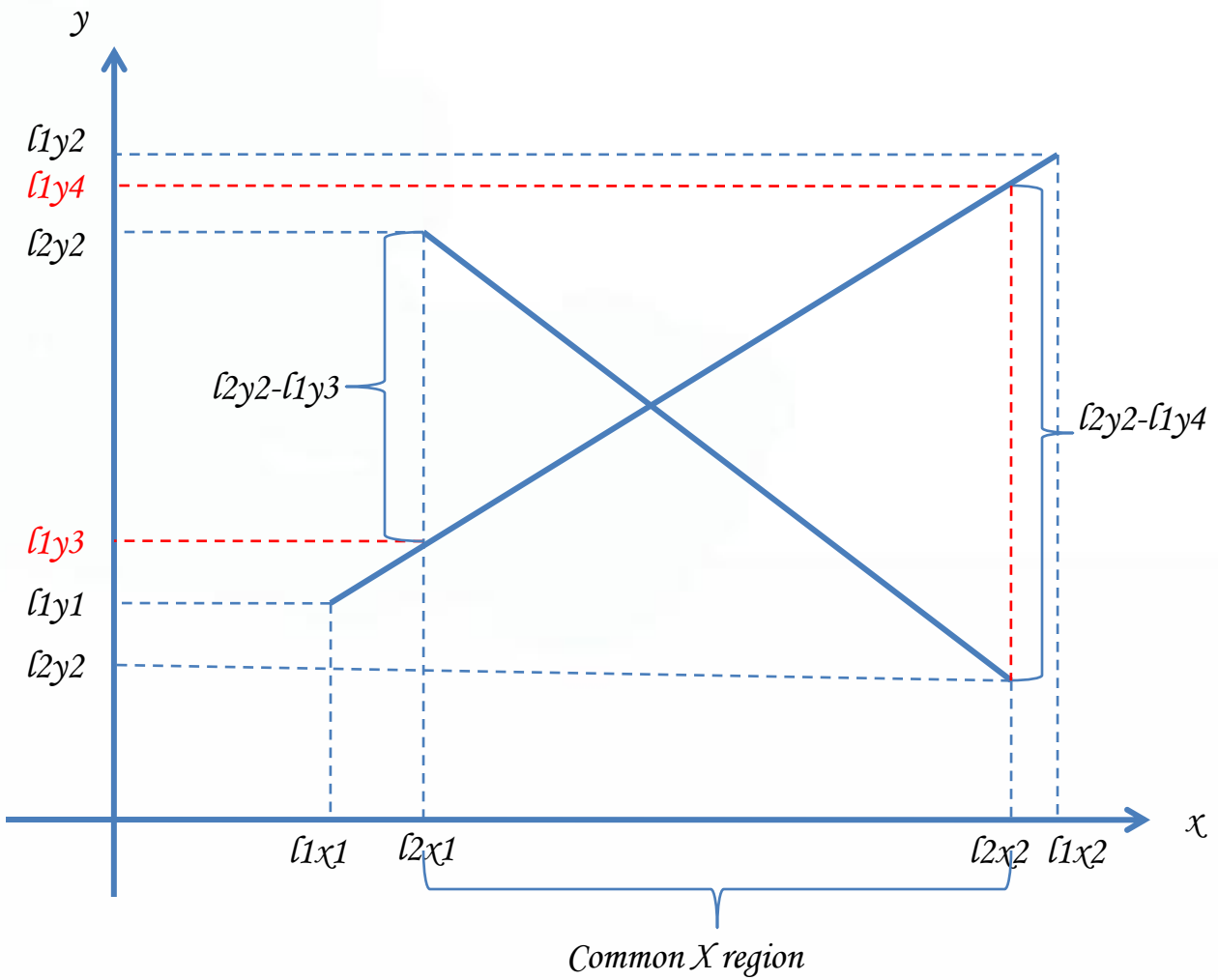
Equation

$$(l2y2-l1y3) * (l2y3-l1y2) > 0$$

$$\frac{(l2y1-l1y2)(l1x2-l1x1) + (l1y2-l1y1)(l1x2-l2x1)}{(l1x2-l1x1)} *$$

$$\frac{(l2y1-l1y2)(l1x2-l1x1) + (l2y1-l2y2)(l1x2-l2x1)}{(l2x2-l2x1)} > 0$$

Find intersection of two polygons Solution 1(cont)



Find intersection of two polygons Solution 1(cont)

Equation

$$(l2y2-l1y3) * (l2y3-l1y2) > 0$$

$$\frac{(l2y1-l1y2)(l1x2-l1x1) + (l1y2-l1y1)(l1x2-l2x1)}{(l1x2-l1x1)} *$$

$$\frac{(l2y1-l1y2)(l1x2-l1x1) + (l2y1-l2y2)(l1x2-l2x1)}{(l2x2-l2x1)} > 0$$

Find intersection of two polygons Solution 1(cont)

- Prolog Code

```
lineNotIntersect([[L1X1,L1Y1],[L1X2,L1Y2]],[[L2X1,L2Y1],[L2X2,L2Y2]]) :-
    L1X1<L1X2, L2X1<L2X2, (L1X1<L2X1 ; L1X1 is L2X1), (L1X2<L2X2 ; L1X2
    is L2X2), (((L2Y1-L1Y2)*(L1X2-L1X1)) - ((L2Y1-L2Y2)*(L1X2-L2X1))) > 0.

lineNotIntersect([[L1X1,L1Y1],[L1X2,L1Y2]],[[L2X1,L2Y1],[L2X2,L2Y2]]) :-
    L1X1<L1X2, L2X1<L2X2, L1X1<L2X1, L1X2>L2X2, (((L2Y2-L1Y1)*(L1X2-
    L1X1)) - ((L1Y1-L1Y1)*(L2X2-L1X1))) > 0.

checkPolygonInterSection(X,Y) :- append(X,Y,R),checkIntersection(R).

checkIntersection([H,T]) :- lineNotIntersect(H,T).
checkIntersection([H|[H1|T]]) :- lineNotIntersect(H,H1),
    checkIntersection([H|T]), checkIntersection([H1|T]).
```

Find intersection of two polygons (cont)

- Solution 2
  - ✓Find the common point of two lines using Bresenham's line drawing algorithm.
- Issues
  - ✓If the common point lies in a fraction value it will not be identified by the Bresenham's algorithm.
  - ✓If a line contains millions of pixel points to calculate all the pixel points using this algorithm takes much time and increases the complexity.

Find intersection of two polygons Solution 2(cont)

Bresenham's Line-Drawing Algorithm for $|m| < 1$

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

$$p_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$$p_{k+1} = p_k + 2\Delta y$$

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repeat step 4 Δx times.

Find intersection of two polygons Solution 2(cont)

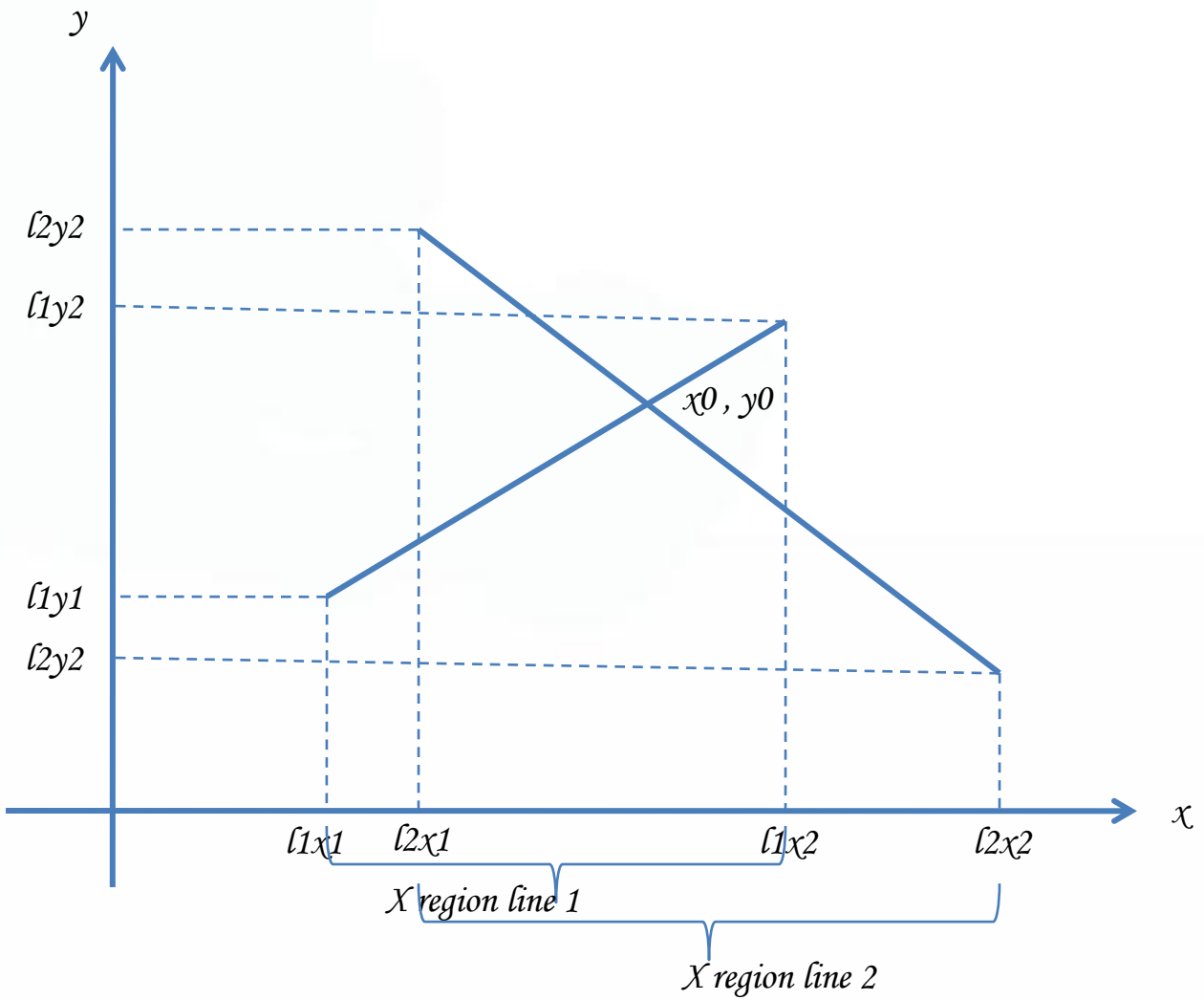
- Prolog Code

```
addToList(H,[H|T],T).

addToList(X,[H|T],[H|R]) :- addToList(X,T,R).

checkIntersection([P1,P2], [P3,P4]) :- lineDraw([P1,P2], L1), lineDraw([P3,P4],
    L2), findNth(L1,X), findNth(L2,X).

lineDraw([[X1,Y1],[X2,Y2]],R) :- X1<X2, getPo([[X1,Y1],[X2,Y2]], Po),
    getLine([[X1,Y1],[X2,Y2]],R , Po, DeltaY, DeltaX), DeltaY is Y2-Y1, DeltaX is
    X2-X1.

getPo([[X1,Y1],[X2,Y2]], Po) :- Po is 2*(Y2-Y1) - (X2-X1).

getLine([[X2,Y2],[X2,Y2]],R, Po,  DeltaY, DeltaX).

getLine([[X1,Y1],[X2,Y2]],[H|T], Po,  DeltaY, DeltaX) :- ( Po<0,
    addToList(N,R,[H|T]), N is (X1+1,Y1), Pk1 is Po + (2*DeltaY),
    getLine(N,[X2,Y2],R ,Pk1 ,DeltaY, DeltaX) ) ; ( Po>0,  addToList(N,R,[H|T]), N
    is (X1+1,Y1+1), Pk1 is Po + (2*DeltaY-2*DeltaX), getLine(N,[X2,Y2],R, Pk1,
    DeltaY, DeltaX) ).
```

Find intersection of two polygons (cont)

- Solution 3
  - ✓Define the intersection point of two lines using coordinates of the two lines and then the common X point should be in the both X coordinates range of lines.
- Challenges
  - ✓To implement above algorithm in prolog I should have to write specific prolog code for
    - Addition
    - Multiplication
    - Comparison

Find intersection of two polygons Solution 3(cont)



Find intersection of two polygons Solution 3(cont)

- Equation

$$x0 = \frac{(x1*x3*y2 - x2*x3*y1 - x1*x4*y2 + x2*x4*y1 - x1*x3*y4 + x1*x4*y3 + x2*x3*y4 - x2*x4*y3)}{(x1*y3 - x3*y1 - x1*y4 - x2*y3 + x3*y2 + x4*y1 + x2*y4 - x4*y2)}$$

$$l1x1 < x0 < l1x2 \quad AND \quad l2x1 < x0 < l2x2$$

Find intersection of two polygons Solution 3(cont)

Prolog Implementation for Arithmetic Addition

- Reason
  - ✓ Add two numbers using “+” operator is not supporting reverse functions like subtraction
- Issues Identified in implementation
  - ✓ Implemented prolog code just satisfies adding numbers in same amount of number bits
  - ✓ When going to solve above prolog adds unnecessary zero values in front of the numbers
  - ✓ Depth should be limited since when we call reverse function it ends up in an infinite loop

Prolog Implementation for Arithmetic Addition

```
% Addition

% Addition main function
plus1(A,B,R):-  plusMy(A,B,[0|R]).

        % Ommit unncessary zeros and prepare two lists into same length and call addition predicate
        plusMy(A,B,[0|R]) :- alignTwoLists(A,B,[0|X],[0|Y]), plusMy(X,Y,R).
        plusMy(A,B,[D|C]) :- alignTwoLists(A,B,X,Y), myPlusNew(X,Y,C,D).

                % Prepare two lists in same length by adding zeros infront of the small list, & manage the lists in same length also
                alignTwoLists(X,Y,X,B) :- (longOrEqual(X,Y,R1),addZeros(R1,Z1), append(Z1,Y,B)).
                alignTwoLists(X,Y,A,Y) :-  (longOrEqual(Y,X,R2),  addZerosEqual(R2,Z2), append(Z2,X,A)).

                        % Check whether the lists are in same length and if not return the exceeding elements of long list
                        longOrEqual(X,[],X).
                        longOrEqual([H1|P],[H2|Q],X) :- longOrEqual(P,Q,X).

                        % Provide a list with zeros similar to the number of elements given by above method
                        addZeros([],[]).
                        addZeros([H|X],[0|Y]) :- addZeros(X,Y).

                        % Provide an empty list if the exceeding list is empty
                        addZerosEqual([X],[0]).
                        addZerosEqual([H|X],[0|Y]) :- addZerosEqual(X,Y).

                % Add two numbers similar in length and provide the output
                myPlusNew([A], [B], [C], D) :- da(A,B,C,D).
                myPlusNew([X|A], [Y|B], [Z|R], Q) :- myPlusNew(A,B,R,Q1), da(X,Q1,R2,Q2), da(R2,Y,Z,Q3), da(Q3,Q2,Q,0).
```

Find intersection of two polygons Solution 3(cont)

Prolog Implementation for Arithmetic Multiplication

- Reason
  - ✓ Multiply two numbers using "*" operator is not supporting reverse functions.
- Issues identified in implementation
  - ✓ Adding zero elements to front and end of the number lists.
- Solution
  - ✓ Adding zeros to front of the lists has been already solved by prolog addition implementation
  - ✓ Solved adding zeros to end of the result of row multiplication

Find intersection of two polygons Solution 3(cont)

Prolog Implementation for Arithmetic Multiplication

```
% Multiplication

% Multiply two lists of numbers
multiplicatn([X|A],[B],W) :- multiplyRow([X|A],B,W,0).
multiplicatn([X|A],[Y|B],W) :- multiplyRow([X|A],Y,R1,0),addZerosMul(B,Z),append(R1,Z,R3),plus1(R2,R3,W),multiplication([X|A],B,R2).

        % Calculate appropriate number of zeros to add at the end of result of multiplication of a row
        addZerosMul([],[]).
        addZerosMul([H|X],[0|Y]) :- addZeros(X,Y).

        % To multiply a row by a particular number
        multiplyRow([X],M,[R],Q) :- mul(X,M,R,Q).
        multiplyRow([X|A],M,[Y|R],Q) :- multiplyRow(A,M,R,Q2), mul(X,M,N,L), da(N,Q2,Y,F), da(F,L,Q,0).
```

Find intersection of two polygons Solution 3(cont)

Prolog Implementation for Arithmetic Comparison

- Reason
  - ✓ Comparison of two numbers using “< and >” operators are not supporting reverse functions.
- Issues identified in implementation
  - ✓ Adding zero elements to front and end of the number lists.
- Solution
  - ✓ ssd

Find overlapping of two polygons

Count intersecting points

- Method
  - ✓Draw a line from the middle point of a line to other polygon and count the points that intersect other polygon.
  - ✓If count gives an odd value polygons are not intersecting.
  - ✓If count gives even value polygons are intersecting.

Find intersection of two polygons Solution 3(cont)

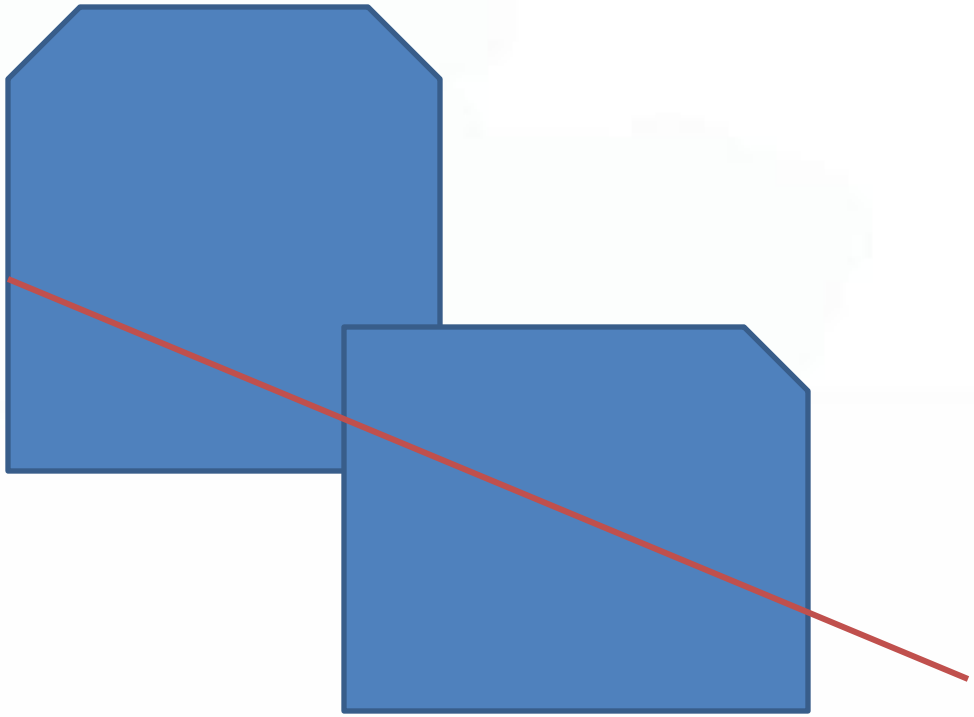
Prolog Implementation for Arithmetic Addition

- Solution
  - ✓ If two lists are in same length I should have to define a specific method to skip adding zeros to that list.

  - ✓ Depth limit
  - ✓ Visual studio – graphical display
  - ✓ Write to text
  - ✓ Read from text

Find overlapping of two polygons

Count intersecting points



Thank You