

ME 290-R:  
General Purpose Computation  
(CAD/CAM/CAE) on the GPU  
(a.k.a. Topics in Manufacturing)

Sara McMains  
Spring 2009  
Lecture 10

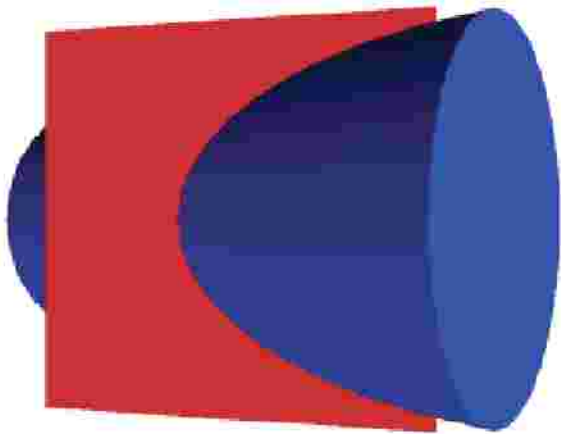
# Today's lecture

- Last time: Visibility
  - Depth peeling
  - Projective texturing
  - Shadow mapping
- Today: Blister
  - "Blister: GPU-based rendering of Boolean combinations of free-form triangulated shapes"
  - John Hable, Jarek Rossignac

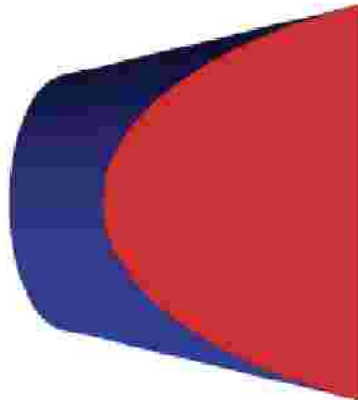
# Blister

- Blist Expression Renderer
  - Blist
    - Boolean list
    - linearized formulation of a Boolean expression
- Renders arbitrary CSG model
  - Combining depth peeling with a Blist

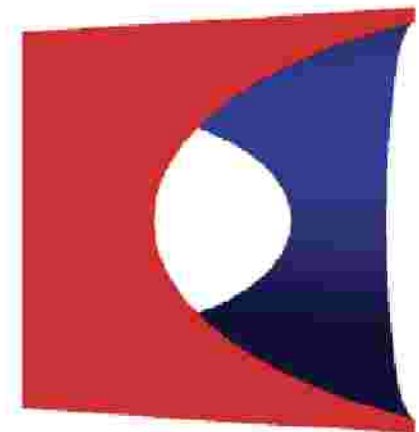
# Boolean Combinations



$A \cup B$

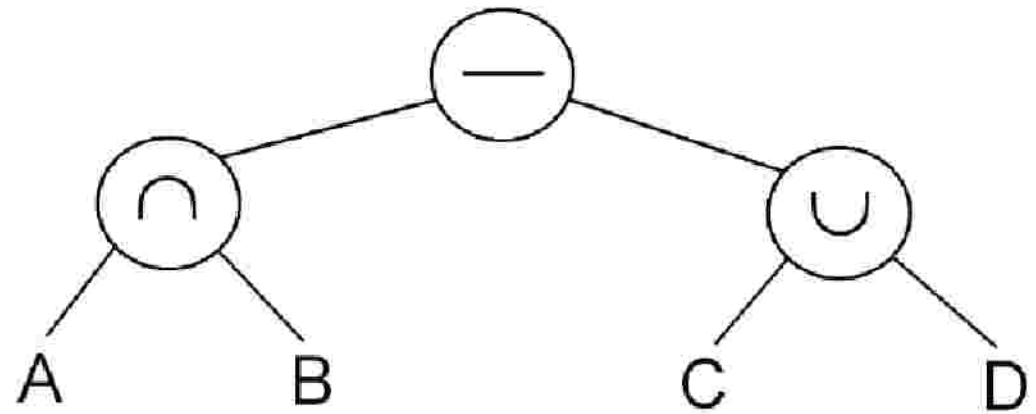
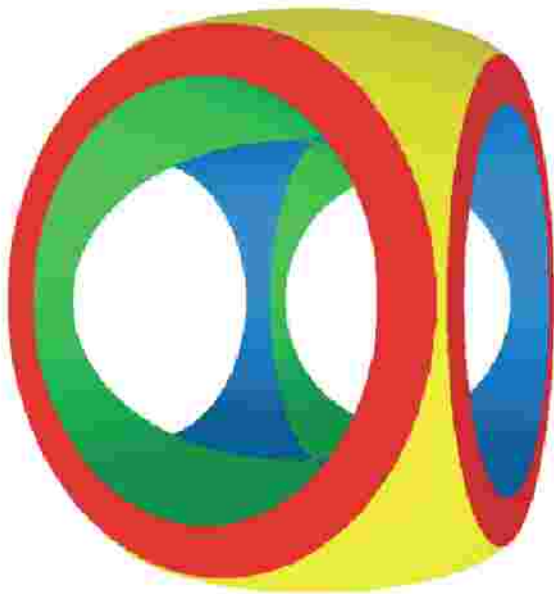


$A \cap B$



$A - B$

# CSG Solid and Tree



# View-dependent depth complexity

- $R_i$ 
  - ray from viewpoint through pixel  $i$
- $k_{ij}$ 
  - number intersections between  $R_i$  and primitive  $P_j$
- $k_i$ 
  - $\max_{\text{all } j} k_{ij}$
- $k$ , depth complexity
  - $\text{Max}_{\{\text{all } i\}} k_i$

# Previous Work

- Typical approach
  - 1) generate candidate surfels
    - $R_i$  intersects some primitive's boundary
  - 2) classify candidate surfels against CSG
    - is this surfel part of the CSG solid?
  - 3) select closest surfel for each pixel

# Previous Work

- (1.candidate, 2.classify, 3.closest)
  - typically done by rasterizer
  - multiple passes if primitives' depth > 1
    - counting approach
      - use stencil bits for counting # intersections for pixel
      - lock surfel produced during ith hit of ith pass
      - Pixel planes (UNC) did this
    - peeling approach
      - (covered last time)


# Previous Work

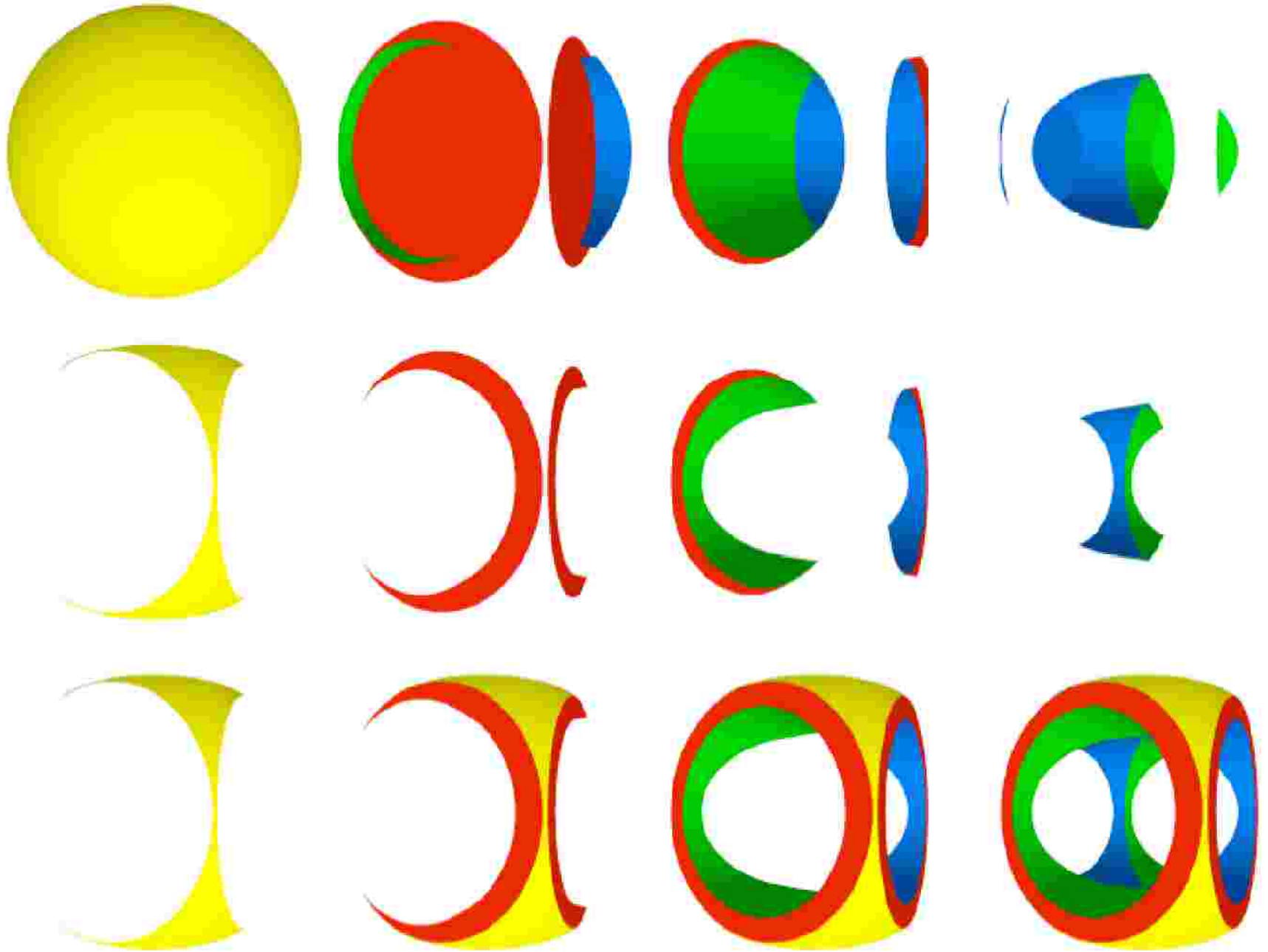
- (1.candidate, 2.classify, 3.closest)
- A previously popular approach: DNF
  - convert CSG tree to disjunctive normal form
    - union of products
    - each product the intersection of positive and negative (complemented) primitives
  - test surfels of each primitive
    - in/out vs. each other primitive in product
  - evaluate for each product, then combine (3.)
  - issue of combinatorial explosion

# Previous Work

- (1.candidate, 2.classify, 3.closest)
- Z-buffer to combine either
  - visible part of each product in DNF,
  - surfels as classified in software,or
  - contribution of each layer

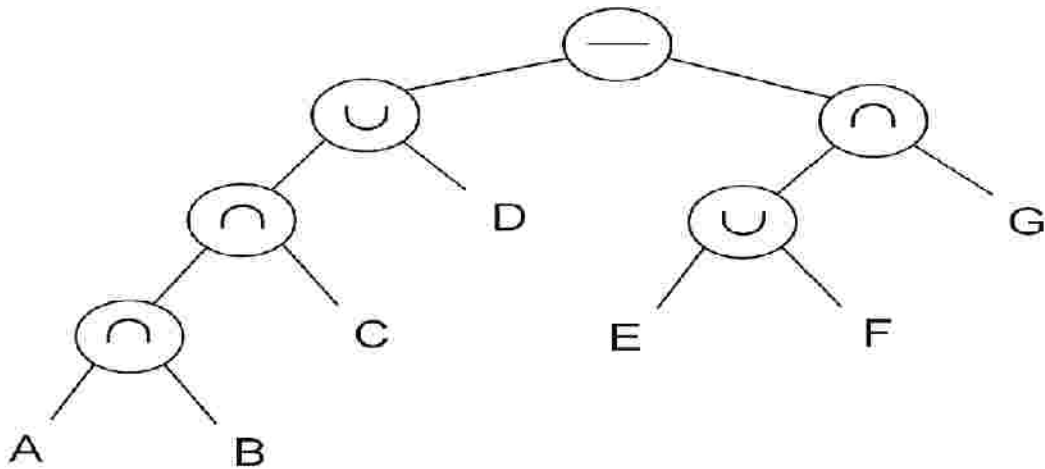
# Blister

- Basic outline
  - depth peel union of all primitives
    - process depth layers in order
  - classify each fragment (surfel) 
  - stop peeling when all pixels w/in bounds hold an opaque surfel
- k depth layers, n primitives ->  $O(nk)$  time



# Positive Form

- Convert differences  $(A - B \Rightarrow A \mid B')$

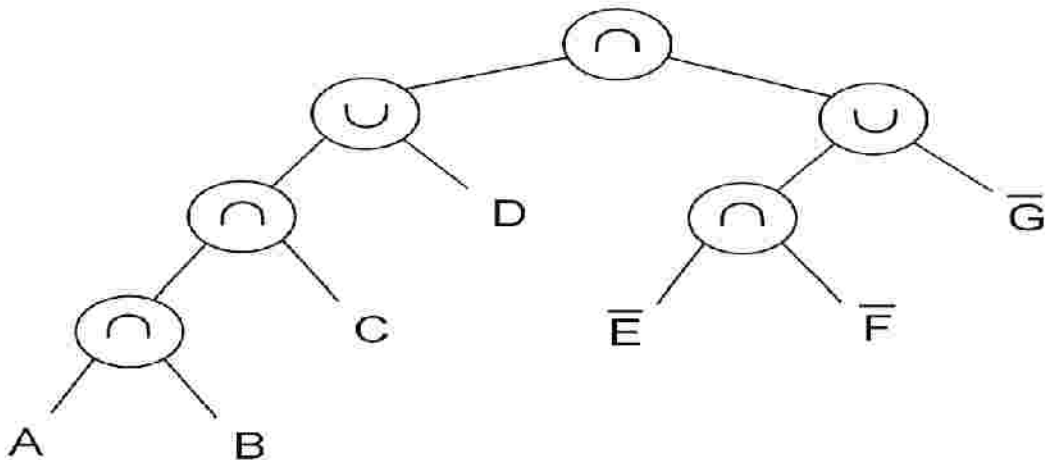


*DeMorgan's laws*

$$(A \cup B)' \Rightarrow (A' \mid B')$$

$$(A \mid B)' \Rightarrow (A' \cup B')$$

$$(A'' \Rightarrow A)$$



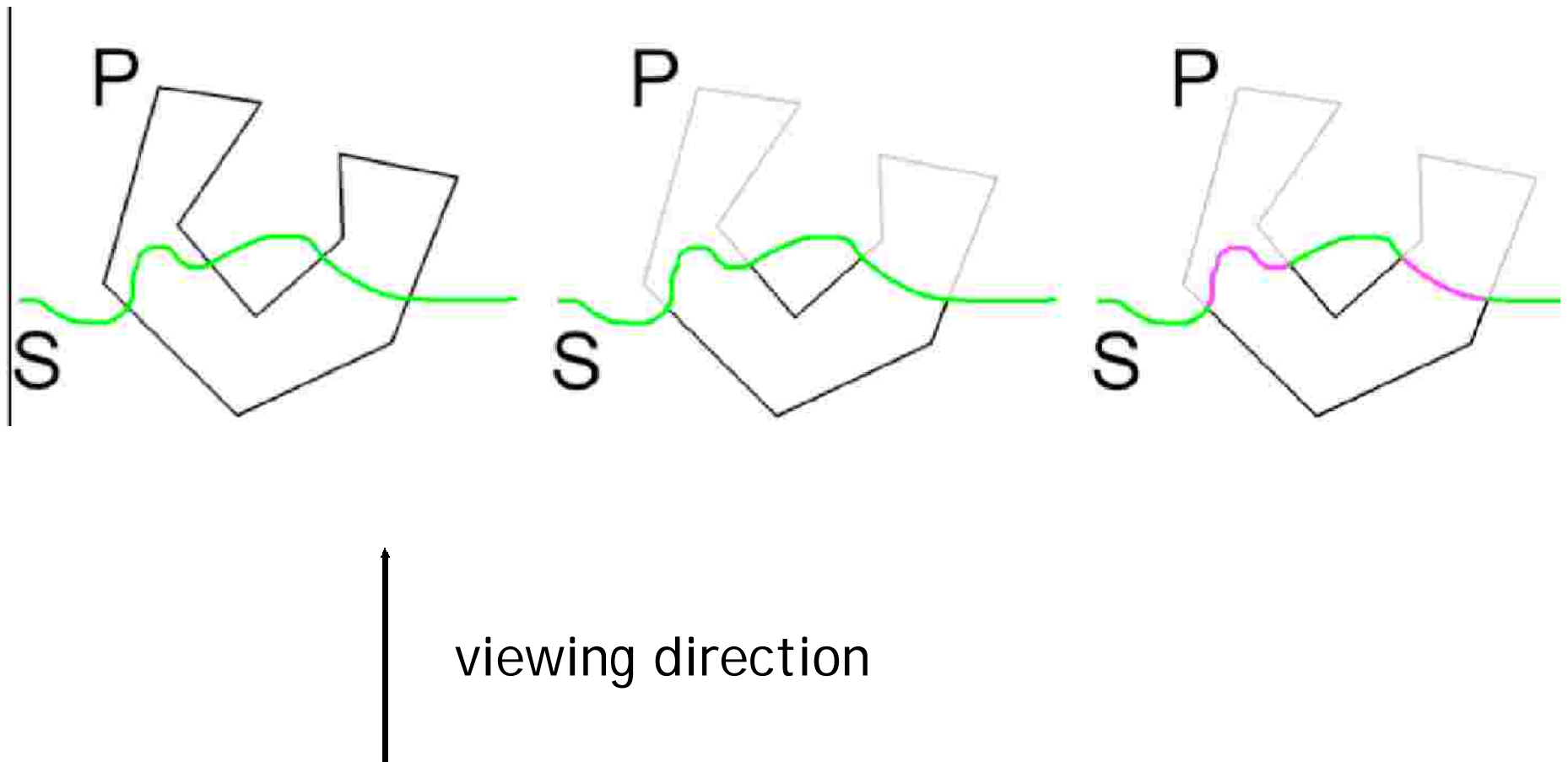
$$(ABCUB) - (EUF)G \Rightarrow$$

$$(ABCUB)((EUF)G)' \Rightarrow$$

$$(ABCUB)((EUF)' \cup G')$$

$$(ABCUB)(E'F'UG')$$

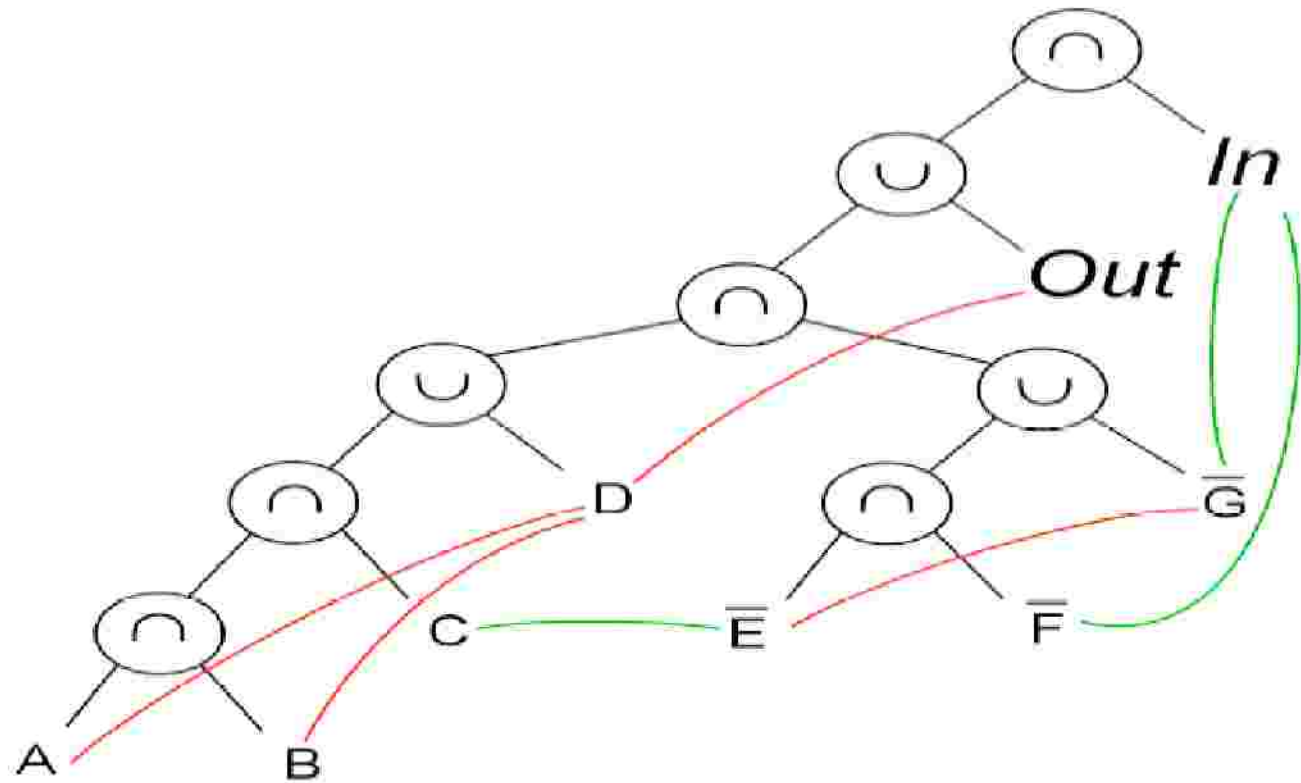
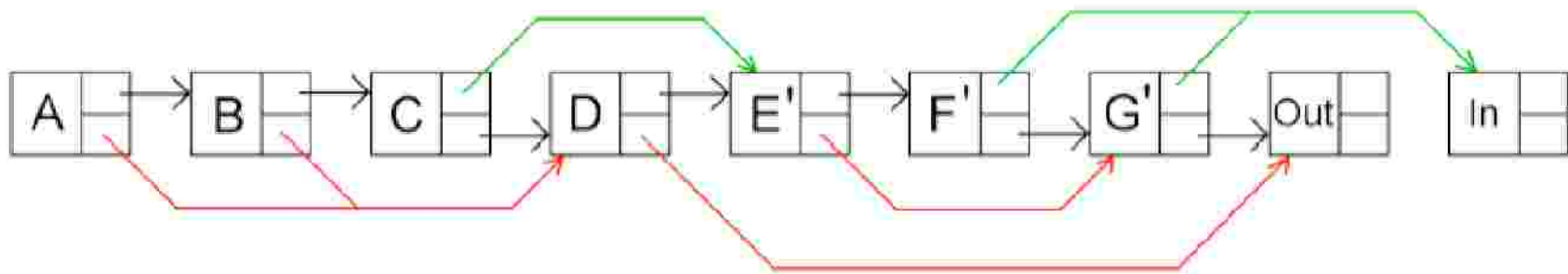
# Parity Test



P	PET(P)	if P=True	if P=False
A	(ABCUD)(E'F'UG')	(BCUD)(E'F'UG')	<b>D</b> (E'F'UG')
B	(BCUD)(E'F'UG')	(CUD)(E'F'UG')	<b>D</b> (E'F'UG')
C	(CUD)(E'F'UG')	<b>E'</b> F'UG'	D(E'F'UG')
D	D(E'F'UG')	E'F'UG'	<b>Out</b>
E'	E'F'UG'	F'UG'	<b>G'</b>
F'	F'UG'	<b>In</b>	G'
G'	G'	<b>In</b>	<b>Out</b>

Table 1: Column 2 shows the PET(P) of each primitive P. Column 3 shows what this PET would simplify to if s were in P. A bold symbol identifies match(P).

P	PET(P)	match(P)	flip(P)
A	(ABCUD)(E'F'UG')	<b>D</b>	<i>false</i>
B	(BCUD)(E'F'UG')	<b>D</b>	<i>false</i>
C	(CUD)(E'F'UG')	<b>E'</b>	<i>true</i>
D	D(E'F'UG')	<b>Out</b>	<i>false</i>
E'	E'F'UG'	<b>G'</b>	<i>false</i>
F'	F'UG'	<b>In</b>	<i>true</i>
G'	G'	<b>In</b>	<i>true</i>



# Results

	n	p	k	k <sub>f</sub>	T	T <sub>f</sub>	T <sub>f</sub> /nk <sub>f</sub>
Widget	4	1	4.00	3.85	17	18	1.17
Bunny	4	3	9.13	6.81	405	325	11.9
Bunny2	4	3	9.09	6.77	1.5s	1.2s	44.3
Complex	20	48	14.1	9.17	139	93	0.51
Gear	48	31104	36.4	25.6	636	460	0.37

n : number of primitives

p : number of products after pruning

k : number of peels (average over all viewpoints)

k<sub>f</sub> : average number of peels after pruning unnecessary pixel

T : rendering time using k-peeling

T<sub>f</sub> : rendering time using k<sub>f</sub>-peeling

# Acknowledgements

- Seung Ho Lee