CATALAN AND BOUNDARY LOGIC LANGUAGES

William Bricken
May 2001

In its simplest form, the domain addressed by Boundary Logic (BL) is the set of possible ways to nest and share containers. In typographical format, this is the set of well-balanced parens (WFP). In logical format, it is the set of possible inferences about propositions bound to a truth value {T.F}. In the language of circuitry, the domain addressed by BL is the set of all possible branching circuits with active inputs. As a data-structure, it is the set of trees. In decision theory, it is all possible sequences of yes/no decisions. Mathematically, it is the Catalan numbers.

DISCIPLINE	*DOMAIN*
boundary math	ways to nest and share
typographical	well-formed delimiter strings
logic	implications over bound propositions
circuitry	branching circuits with active input
computer science	set of trees
decision theory	possible sequences of binary decisions
mathematics	Catalan numbers

Catalan numbers are well-studied, mathematicians know of many abstract applications and visualizations of the fundamental concept of containment. These tools assist the conceptualization and design of new software and hardware architectures based in BL.

We will use the model of circuitry to describe to choices provides by mathematical models of Catalan numbers. Typographically, we will illustrate with WFPs.

Treating inputs abstractly is to provide variable labels which may be interspersed through a WFP. Each variable stands in place of a final branch, the evaluation of the variable as 0 or 1

```
( (( a)) (b ()()) )

( ((  )) (  ()()) )          a=<void>   b=<void>
( ((())) (  ()()) )          a=()       b=<void>
( ((  )) (()()()) )          a=<void>   b=()
( ((())) (()()()) )          a=()       b=()
```

Visualizing the mark () as an atomic unit, as in

```
( ((  )) (  ()()) )

((*)(**))
```

provides a representation of a particular circuit with all inputs positive. This is the set set as all possible circuits. Again, by turning on and off these stars, we can simulate a circuit with each star is a variable input. In the above example

```
((a)(bc))
```

By starring a WFP with variables (VWFP), we convert the set of algebraic circuits (in particular, those with one output and without internal reentry) into a set of functioning circuits with all inputs bound to 1.

The effect of this manipulation is to identify as set of directed acyclic graphs (DAG) with one source and one sink. When variables are used, we have multiple bottom nodes; when star is used there is one bottom. As a single output circuits, there is also one top.

We have converted the set of trees into a subset of DAGs in the process of binding variables.

CATALAN NUMBERS

Consider the generalized binomial series,

```
Bt(z) = SUM[k >=0]   (tk)^(k-1) * (z^k)/k!

                    (2k)
B2(z) = SUM[k]      ( k) * (z^k)/(1+k)
```

Catalan numbers are B2 coefficients

```
(2n)
( n) * 1/(n+1)
```

n	0	1	2	3	4	5	6	7	8	9	10
Cn	1	1	2	5	14	42	132	429	1430	4862	16796

Catalan numbers are defined by a convolution:

C[n] = C[0]*C[n-1] + C[1]*C[n-2] + ... + C[n-1]*C[0]

This can be converted into a generator function:

C[z+1] = C[z]*zC[z] + 1

PARENS

<void>

()

()()
(())

()()()
()(())   (())()
(()())   ((()))

()()()()
()()(())   ()(())()   (())()()   (()()())   (()())()   ()(()())
()((()))   ((()))()   (())(())   (()(()))   ((())())   ((()()))
(((())))

BINARY SEQUENCES

<void>

10

1010   1100

101010   101100   110010   110100   111000

10101010
10101100   10110010   11001010   11010100   11010010   10110100
10111000   11100010   11001100   11011000   11100100   11101000
11110000

PARENS WITH STARS

<void>

*

**   (*)

***   *(*)   (*)*   (**)

((*))
****
**(*)
(*)
(*)**
(***)
(**)*
*(**)

(*)(*)
*((*))
((*))*
(*(*))
((*)*)
((**))

(((*)))

```
1
      *****     => []
10
      ***(*)    => []
      **(*)*    => []
      *(*)**    => []
      (*)***    => []
      **(**)    => []
      *(**)*    => []
      (**)**    => []
      *(***)    => []
      (***)*    => []
      (****)    => <>
20
      *(*)(*)   => []
      (*)*(*)   => []
      (*)(*)*   => []
      **((*))   => []
      *((*))*   => []
      ((*))**   => []
      *(*(*))   => []
      *((*)*)   => []
      (*(*))*   => []
      ((*)*)*   => []
      *((**))   => []
      ((**))*   => []
      (*)(**)   => <><>
      (**)(*)   => <><>
      (**(*))   => <>
      (*(*)*)   => <>
      ((*)**)   => <>
      (*(**))   => <>
      ((**)*)   => <>
      ((***))   => ()
10
      *(((*)))  => []
      (((*)))*  => []
      (*)((*))  => <>()
      ((*))(*)  => ()<>
      (*((*)))  => <>
      (((*))*)  => <>
      ((*)(*))  => ()
      ((*(*)))  => ((()    )) => ()
      (((*)*))  => ((    ())) => ()
      (((**)))  => ((      )) => <>
1
      ((((*)))) => (((    ))) => ()
```

MOUNTAIN RANGES

a1 + a2 + ... + a2n = 0   a = {-1,1}

such that all partial sums are nonnegative

a1
a1 + a2
...
a1 + a2 + ... + a2n

When a=1, draw /   When a=-1, draw \

Depth of nesting = height of mountain

```
  /\  /\/\
 /  \/    \
/          \
((())(()()))
```

```
.

/\

          /\
/\/\     /  \


               /\
/\/\/\      /\/  \


/\/\/\/\


     /\          /\          /\           /\/\/\       /\/\           /\/\
/\/\/  \    /\/  \/\    /  \/\/\    /      \    /    \/\    /\/    \


    /\          /\            /\          /\          /\/\
   /  \       /  \        /\/  \       /  \/\       /    \       /\  /\
/\/    \    /    \/\    /      \    /      \    /      \    /  \/  \


   /\
  /  \
 /    \
/      \
```

THE COUNTING LOGIC

Sequences of +1s and -1s whose partial sums are always positive. For computational convenience initial all sequence with 1.

$\binom{2n+1}{n}$ sequences of n occurrences of -1 and $n+1$ occurrences of +1

exactly 1/(2n+1) has positive partial sums (Raney)

$$C[n] = \binom{2n+1}{n} * 1/(2n+1) = \binom{2n}{n} * 1/(n+1)$$

DISECTING POLYGONS

Given an (n+2)-sided polygon

n	ways	name
0	1	line
1	1	triangle
2	2	square
3	5	pentagon

BIFURCATING TREES

```
    /


    /
   /\


    /      /
   /\     /\
  /\       /\


    /        /        /        /        /
   /\       /\       /\       /\       /\
  /\       /\       /\/\       /\       /\
 /\          /\               /\          /\
```

ROOTED PLANAR BUSHES

```
            o


            |
            o


                     |
           \ /       |
            o        o


                                                |
                   \            /     \ /       |
           \|/      \ /      \ /       |        |
            o        o        o        o        o
```

BINOMIAL COEFFICIENTS

```
                  1
                1
              1   2
            1   3
          1   4   6
        1   5  10
      1   6  15  20
    1   7  21  35
  1   8  28  56  70
```

The middle sequence of binomial coefficients, divided by the place

	1	2	6	20	70	252	924	3432	12870	48620
	1	2	3	4	5	6	7	8	9	10
=C	1	1	2	5	14	42	132	429	1430	4862

OCCLUSION

Assume an outer container

grounds: <void>   ()

elementary   ()()   (())

[()()] => []
(())   => <>

()()() => []
()(()) => []
(())() => []
(()()) => <>
((())) => ()

()()()() => []

()()(()) => []
()(())() => []
(())()() => []
(()()()) => <>
(()())() => []
()(()()) => []

()((())) => []
((()))() => []
(())(()) => <><>
(()(())) => <>
((())()) => <>
((()())) => ()

(((()))) => ((   )) => <>

()()()()() => []

()()()(()) => []
()()(())() => []
()(())()() => []
(())()()() => []
()()(()()) => []
()(()())() => []

```
(()())()() => []
()(()()()) => []
(()()())() => []
(()()()()) => <>

()()((())) => []
()((()))() => []
((()))()() => []
()(())(()) => []
(())()(()) => []
(())(())() => []
()(()(())) => []
()((())()) => []
(()(()))() => []
((())())() => []
()((()())) => []
((()()))() => []
(())(()()) => <><>
(()())(()) => <><>
(()()(())) => <>
(()(())()) => <>
((())()()) => <>
(()(()())) => <>
((()())()) => <>
((()()())) => (        )

()(((()))) => []
((((()))() => []
(())((())) => <>()
((()))(()) => ()<>
(()((()))) => <>
(((()))()) => <>
((())(())) => ()
((()(()))) => ((()    )) => ()
(((())()))  => ((    ())) => ()
(((()()))) => ((      )) => <>

((((())))) => (((    ))) => ()
```

ORDER INDEPENDENT

<void>

*

**
(*)

***
(*)   (*)
(**)
((*))

****
(*)   *(*)*   (*)
(***)
(**)*   *(**)
(*)(*)
((*))   ((*))
(*(*))   ((*)*)
((**))
(((*)))

*****

(*)   **(*)*   *(*)**   (*)
(**)   *(**)*   (**)
(***)   (***)
(****)

(*)(*)   (*)(*)   (*)(*)*
((*))   *((*))*   ((*))
((*))   *((*)*)   (*(*))*   ((*)*)*
((**))   ((**))
(*)(**)   (**)(*)
(**(*))   (*(*)*)   ((*)**)
(*(**))   ((**)*)
((***))

(((*)))   (((*)))
(*)((*))   ((*))(*)
(*((*)))   (((*))*)
((*)(*))
((*(*)))   (((*)*))
(((**)))

((((*))))

REDUCED OCCLUSIONS

parens	[]	()	<>	steps	sum	no-order
1	1	0	0	0	1	1
2	1	0	1	1	2	2
3	3	1	1	1	5	4
4	8	1	5	2(1)	14	9
5	24	7	11	2(4)	42	20
6					132	

VARIETIES

parens	stars						no-order = depth					
	1	2	3	4	5	6	1	2	3	4	5	6
1	1						1					
2	1	1					1	1				
3	1	3	1				1	2	1			
4	1	6	6	1			1	4	3	1		
5	1	10	20	10	1		1	6	8	4	1	
6	1					1	1					1