Non-Halters in the Busy Beaver Problem

presented by Owen Kellett
4 April 2003

Bram van Heuveln
Boleslaw Szymanski
Selmer Bringsjord
Carlos Varela
Owen Kellett
Shailesh Kelkar
Kyle Ross

The Busy Beaver Problem

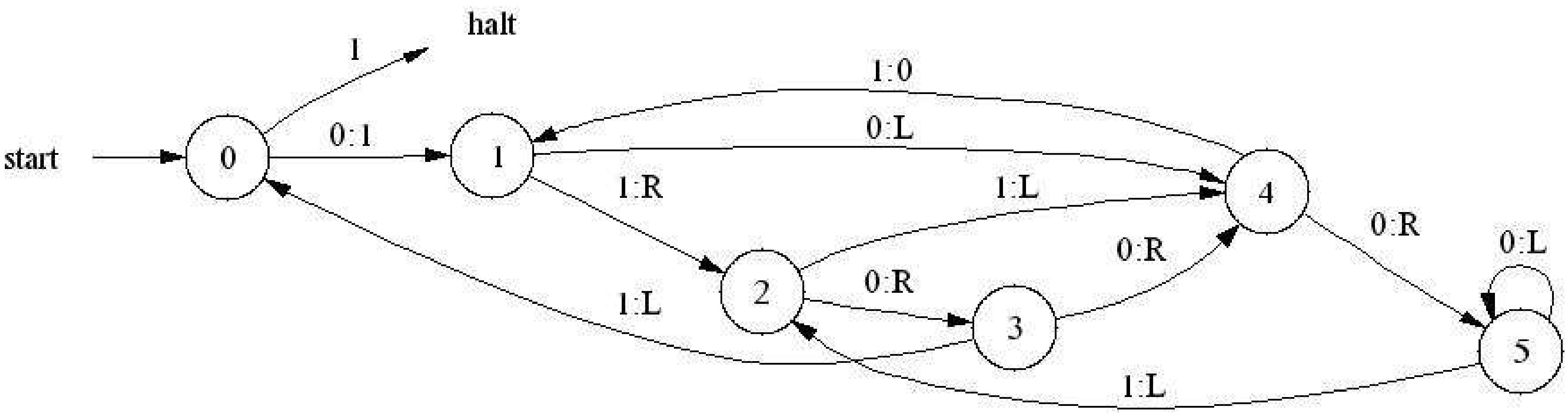
"Consider, for a fixed positive integer *n, the class K_n of all the n-card [state] binary Turing machines ... Let M be a Turing machine in this class K_n. Start M, with its card 1, on an all-0 tape. If M stops after a while, then M is termed a valid entry in the BB-n contest ... and its score σ(M) is the number of 1's remaining on the tape at the time it stops ... [the set of σ-values] has a (unique) largest element which we denote by Σ(n) ... It is practically trivial that this function Σ(n) is not general recursive ... [but] it may be possible to determine the value of Σ(n) for particular values of n.*"

-Lin & Rado "Computer Studies of Turing Machine Problems" Journal of the Association for Computing Machinery, Vol. 12, No. 2 (April, 1964), pp. 196-212

Problem: How do we know when it stops?

- Turing Machine halting problem:
  - Turing Machine *M*
  - Input tape *w*
  - Function : given any TM *M* and any Input tape *w*, return whether or not *M* halts on *w.*
  - This function does not exist
- However!
- Certain routines can be defined which identify whether or not a particular machine exhibits a specific non halting behavior
- The Busy Beaver problem exhibits several recognizable behaviors

Backtracking



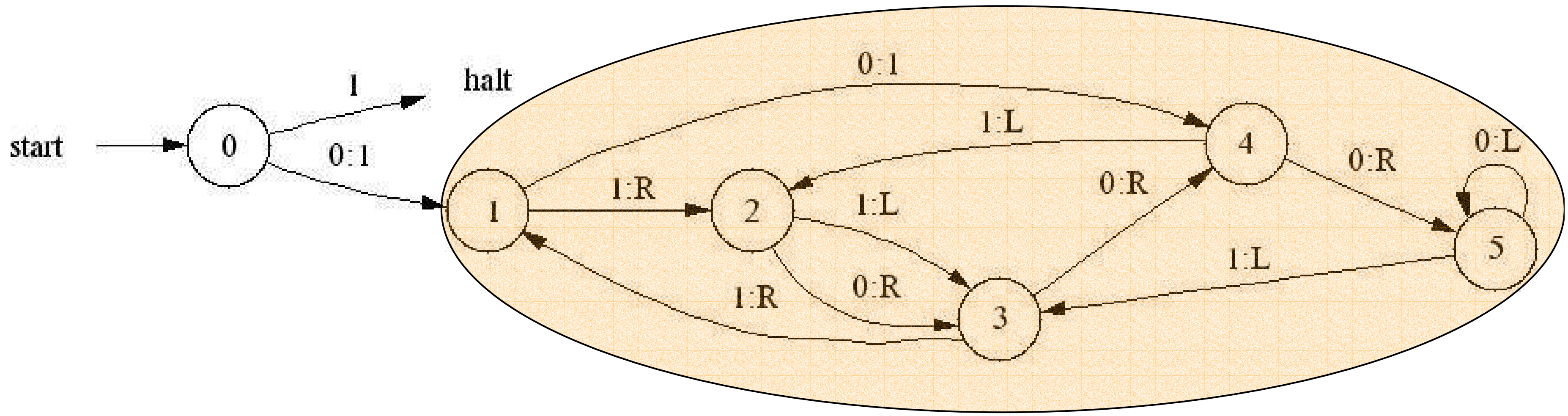
q[2]x0 → q[3]xR ⟶ q[3]x1 → q[0]xL ⟶ q[0]x1 → halt

$0_2 1$ ↓ 01_3 ⟶ 11_3 ↓ $1_0 1$ ⟶ 1_0

Does not match!
Non-halter!

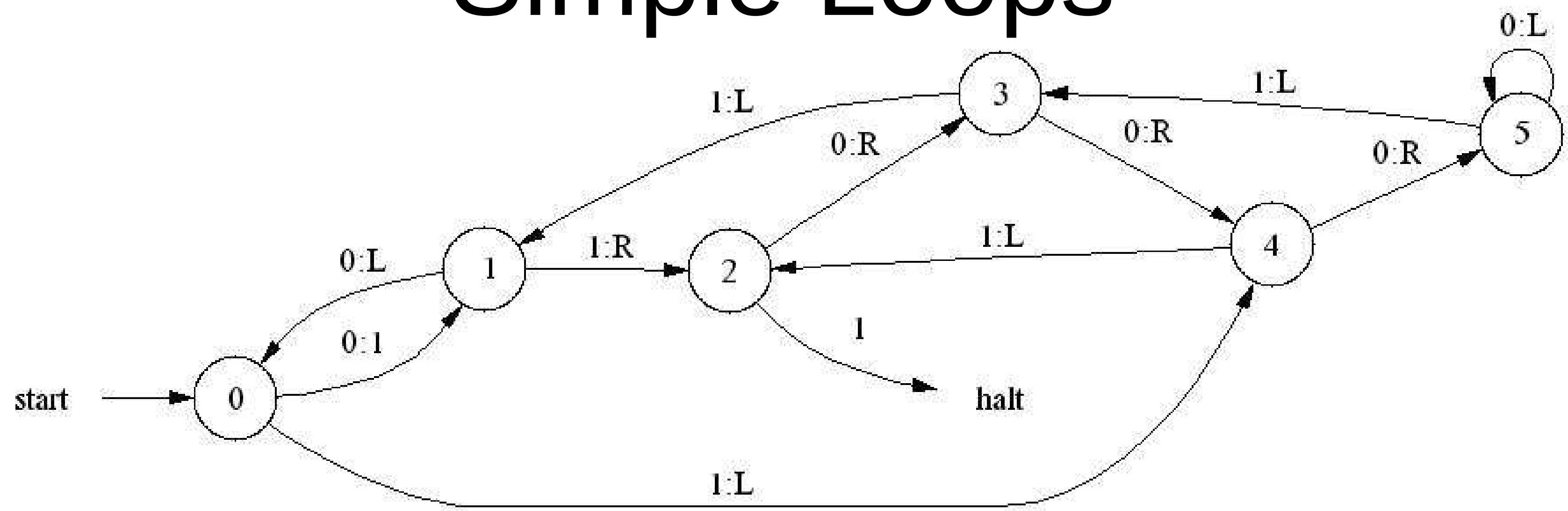
Local tapes match, continue

Subset Loops



- A Turing Machine M is classified as a subset loop if
  - There is a set of states S such that every possible transition from each state in S is defined
  - Every transition defined from a state in S is a transition to another state in S
  - During execution, at some point the machine enters one of the states in S

Simple Loops



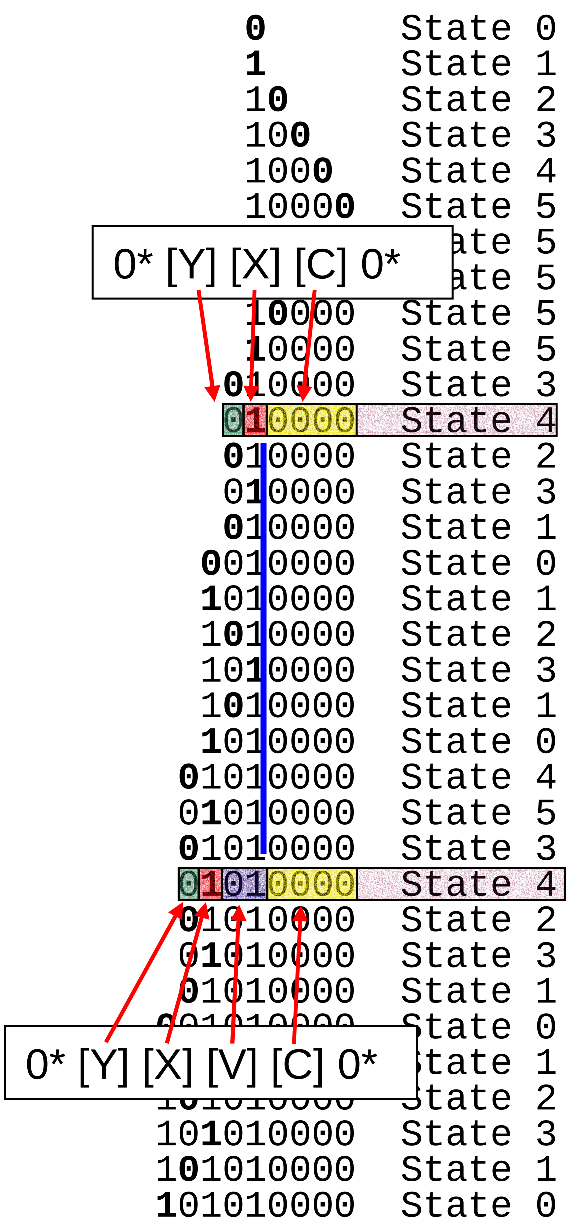
- A machine is classified as a simple loop if (given words of arbitrary length X, Y, V, and C and state *s*):
  - The following tape configuration is reached: $0^*[C][X_s][Y]0^*$

    -and one of the following-
  - The same tape configuration is reached at a later point

    -or-
  - The following tape configuration is reached at a later point: $0^*[C][V][X_s][Y]0^*$
  - Between these points, the read head never moves past the left edge of the initial X
- The corresponding mirror of the above specification also identifies a simple loop

-Machlin and Stout. "The Complex Behavior of Simple Machines" Physica D 42 (1990). pp. 85-98

Example: Simple Loop



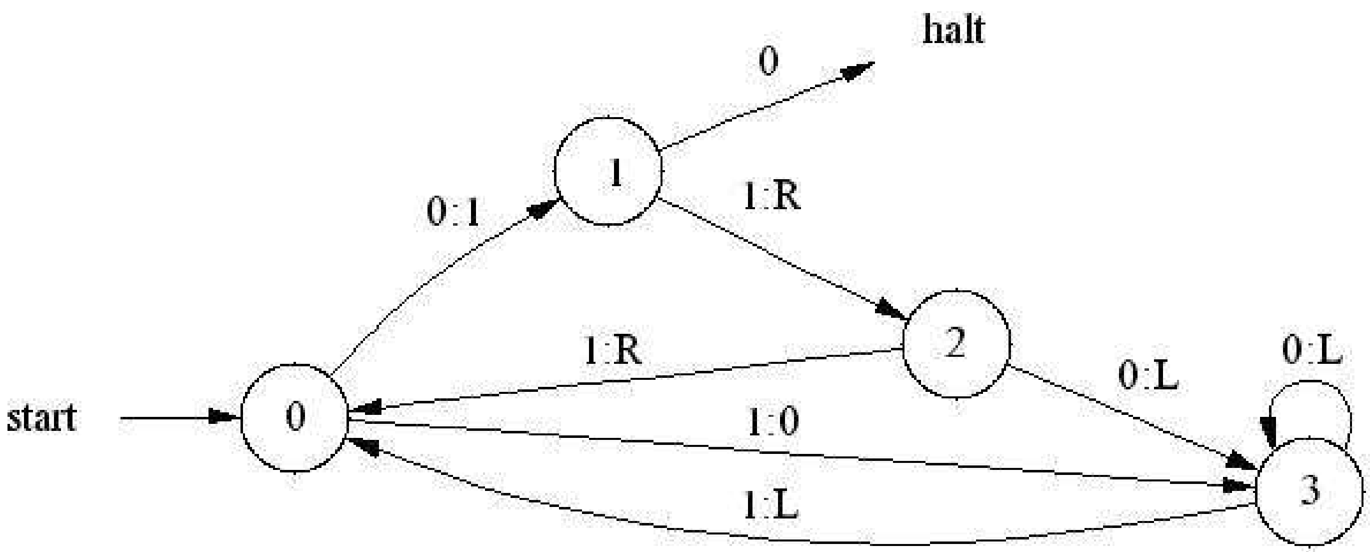
- The tape configuration of the most recent occurrence of each <state, symbol> pair is saved
- This particular instance is the pair <4,1>
- Example machine is a mirror version of the simple loop specification
- The next tape configuration of <4,1> is compared to the previous
- In this case, all components are identifiable and match
- Location of read head is in same relative location
- The read head never moves to the right of the original X (remember this is a mirror simple loop)
- All conditions are satisfied, machine is a simple loop non-halter

Christmas Trees

- In the general sense, a christmas tree non-halter sweeps back and forth across the tape in a repeatable manner:

0	State 0
1	State 1
1**0**	State 2

1101010	State 1
1**1**01010	State 2
11**0**1010	State 0
111**1**010	State 1
111**1**010	State 2



111**0**10	State 3
11**1**010	State 3
1**1**1010	State 0
1**0**1010	State 3
101010	State 3
0101010	State 0

1111**1**1010	State 1
11111**1**010	State 2
111111**0**10	State 0
1111111**1**10	State 1
11111111**1**0	State 2
11111111**0**	State 0
11111111**1**	State 1
111111111**0**	State 2

Christmas Tree Detection: Step 1

```
0* [U] [V] 0*        s=2

 1110        State 2
 1110        State 3
 1110        State 0
 1010        State 3
 1010        State 3
01010        State 0
11010        State 1
11010        State 2
11010        State 0
11110        State 1
11110        State 2
11110        State 0
11111        State 1
111110       State 2

0* [U] [X] [V] 0*    s=2
```

- The tape exhibits a back and forth sweeping motion
- After one sweep, the tape has the following configuration:
  - $0^*[U][V_s]0^*$
- After the next sweep, a new middle part, and the same end parts are seen:
  - $0^*[U][X][V_s]0^*$

Christmas Tree Detection: Step 2

Tape	State
111110	State 2
111110	State 3
111110	State 0
111010	State 3
111010	State 3
111010	State 0
101010	State 3
101010	State 3
0101010	State 0
1101010	State 1
1101010	State 2
1101010	State 0
1111010	State 1
1111010	State 2
1111010	State 0
1111110	State 1
1111110	State 2
1111110	State 0
1111111	State 1
11111110	State 2

- The machine alters the tape according to the following grammar
  - $[X][V_s] \rightarrow {}_q[X'][V']0^*$
  - $[X_q][X'] \rightarrow {}_q[X'][Y]$
  - $0^*[U_q][X'] \rightarrow 0^*[U'][Y']_r$
  - $[Y'][{}_rY] \rightarrow [Z][Y']_r$
  - $[Y'][{}_rV'] \rightarrow [Z][V''_s]$

$0^*[U][X][V_s]0^*$

$0^*[U_q][X'][V']0^*$

$0^*[U'][Y'][{}_rV']0^*$

$0^*[U'][Z][V''_s]0^*$

- U = 11
- V = 10
- X = 11
- X' = 10
- V' = 10
- U' = 111
- Y' = 11
- Z = 11
- V'' = 110

- s = 2
- q = 0
- r = 2

- The following holds:
  - 0*[U'][Z][V''s]0* = 0*[U][X][X][Vs]0*

    [11111110] [11111110]

Christmas Tree Detection: Step 3

```
 11111110     State 2
 11111110     State 3
 11111110     State 0
 11111010     State 3
 11111010     State 3
 11111010     State 0
 11101010     State 3
 11101010     State 3
 11101010     State 0
 10101010     State 3
 10101010     State 3
010101010     State 0
110101010     State 1
110101010     State 2
110101010     State 0
111101010     State 1
111101010     State 2
111101010     State 0
111111010     State 1
111111010     State 2
111111010     State 0
111111110     State 1
111111110     State 2
111111110     State 0
111111111     State 1
111111110     State 2
```

- The machine alters the tape according to the following grammar
  - $[X][V_s] \rightarrow {}_q[X'][V']0^*$
  - $[X_q][X'] \rightarrow {}_q[X'][Y]$
  - $0^*[U_q][X'] \rightarrow 0^*[U'][Y']_r$
  - $[Y'][_rY] \rightarrow [Z][Y']_r$
  - $[Y'][_rV'] \rightarrow [Z][V''_s]$

$0^*[U][X][X][V_s]0^*$

$0^*[U][X_q][X'][V']0^*$

$0^*[U_q][X'][Y][V']0^*$

$0^*[U'][Y'][_rY][V']0^*$

$0^*[U'][Z][Y'][_rV']0^*$

$0^*[U'][Z][Z][V''_s]0^*$

- U = 11
- V = 10
- X = 11
- X’ = 10
- V’ = 10
- U’ = 111
- Y’ = 11
- Z = 11
- V’’ = 110
- Y = 10

- s = 2
- q = 0
- r = 2

- The following holds:
  - 0*[U’][Z][Z][V’’s]0* = 0*[U][X][X][X][Vs]0*
    [1111111110]   [1111111110]

Alternating Christmas Trees

- The machine transforms the tape much like a normal Christmas tree, however, it takes two sweeps across the tape rather than one to complete one cycle

Christmas Trees

0*[U][X][X][X][V]0*

←

0*[U’][Y][Y][Y][V’]0*

→

0*[U’][Z][Z][Z][V’’]0*

=

0*[U][X][X][X][X][V]0*

Alternating Christmas Trees

0*[U][X][X][X][V]0*

←

0*[U’][Y][Y][Y][V’]0*

→

0*[U’][Z][Z][Z][V’’]0*

←

0*[U’’][M][M][M][V’’’]0*

→

0*[U’’][N][N][N][V’’’’]0*

=

0*[U][X][X][X][X][V]0*

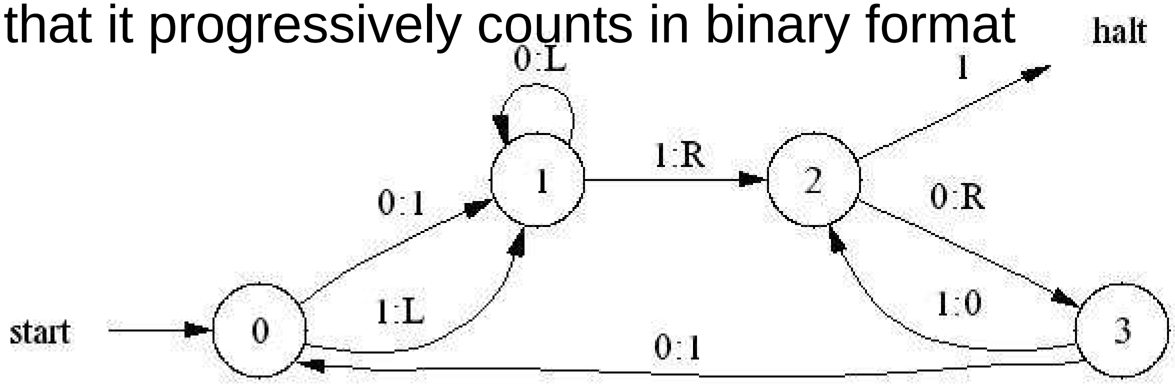
Busy Beaver Non-Halters

	n = 3		n = 4		n = 5	
backTrack	817	82.1106%	47102	85.1538%	3842187	87.1350%
subsetLoop	17	1.7085%	749	1.3541%	38761	0.8790%
simpleLoop	159	15.9799%	7243	13.0943%	508156	11.5242%
christmasTree	2	0.2010%	198	0.3580%	18012	0.4085%
alternateChristmasTree	0	0.0000%	23	0.0416%	2818	0.0639%
holdout	0	0.0000%	5	0.0090%	2623	0.0595%
total	995		55314		4409466	

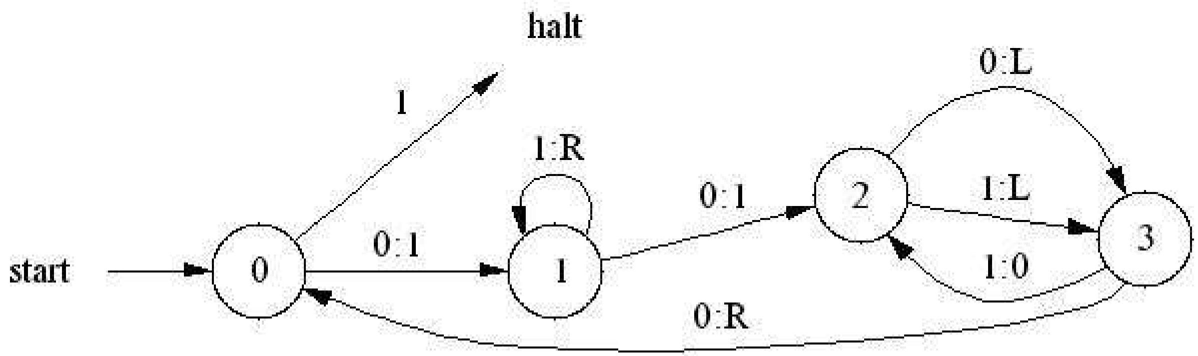
*Note: Machines are classified according to the first routine which tests positive. The detection routines are applied in succession from top to bottom for each individual machine.

B4 Holdouts

- Two of the holdouts of the B4 exhibited the one other behavior specified by Brady but not yet implemented as a detection routine for this project
- These machines mimic binary counters by altering the tape in such a way that it progressively counts in binary format



B4-counter1



B4-counter2

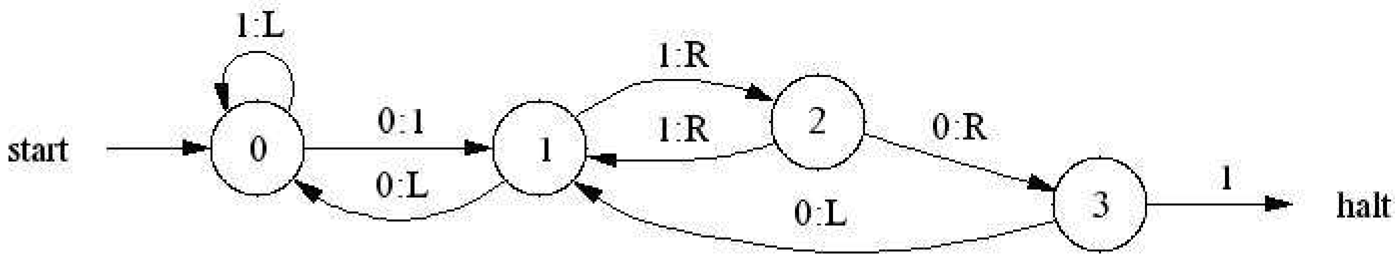
B4-Counter1 Execution

0	State 0	1000**1**	State 0
1	State 1	100**0**1	State 1
1**0**	State 2	10**0**01	State 1
10**0**	State 3	1**0**001	State 1
10**1**	State 0	**1**0001	State 1
1**0**1	State 1	1**0**001	State 2
101	State 1	10**0**01	State 3
1**0**1	State 2	10**1**01	State 0
10**1**	State 3	1**0**101	State 1
10**0**	State 2	**1**0101	State 1
100**0**	State 3	1**0**101	State 2
100**1**	State 0	10**1**01	State 3
10**0**1	State 1	10**0**01	State 2
1**0**01	State 1	100**0**1	State 3
1001	State 1	100**1**1	State 0
1**0**01	State 2	10**0**11	State 1
10**0**1	State 3	1**0**011	State 1
10**1**1	State 0	**1**0011	State 1
1**0**11	State 1	1**0**011	State 2
1011	State 1	10**0**11	State 3
1**0**11	State 2	10**1**11	State 0
10**1**1	State 3	1**0**111	State 1
10**0**1	State 2	**1**0111	State 1
100**1**	State 3	1**0**111	State 2
100**0**	State 2	10**1**11	State 3
1000**0**	State 3	10**0**11	State 2

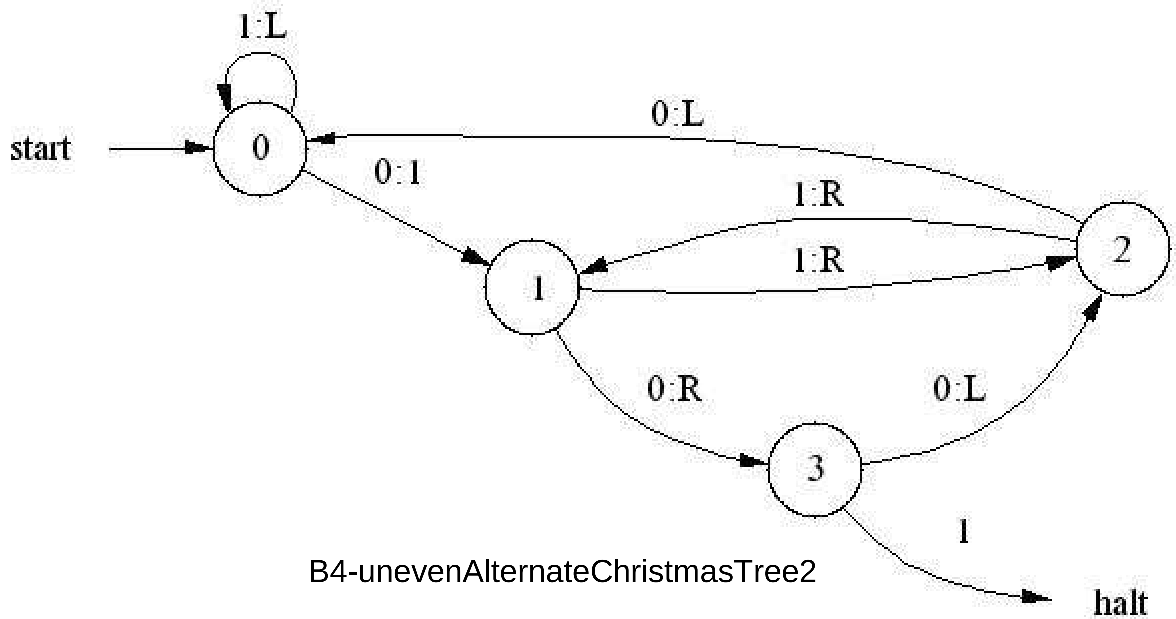
*Note: this machine generates binary numbers that read from right to left rather than the conventional left to right

B4 Holdouts

- Two of the holdouts are very similar to alternating christmas trees



B4-unevenAlternateChristmasTree1



B4-unevenAlternateChristmasTree2

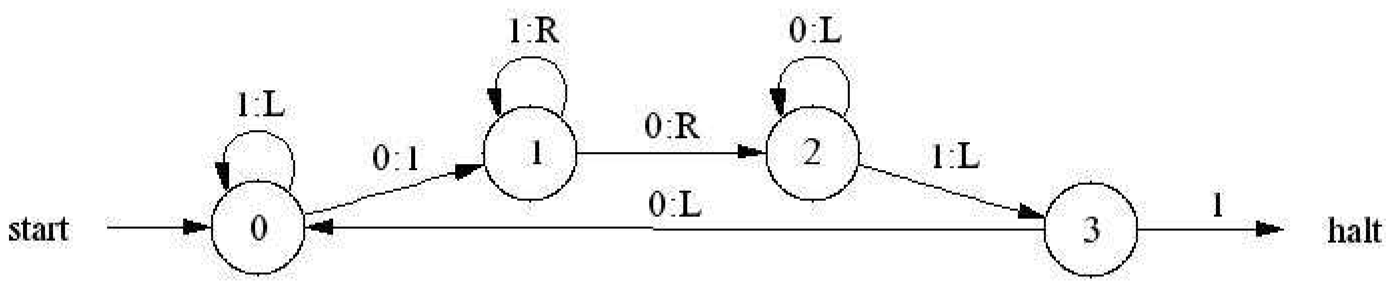
B4-uneven Alternate Christmas Tree1 Execution

0	State 0
1	State 1
1**0**	State 2
10**0**	State 3
1**0**0	State 1
100	State 0
0100	State 0
1100	State 1
1**1**00	State 2
11**0**0	State 1
1**1**00	State 0
1100	State 0
01100	State 0
11100	State 1
1**1**100	State 2
11**1**00	State 1
111**0**0	State 2
1110**0**	State 3
111**0**0	State 1
11**1**00	State 0
1**1**100	State 0
11100	State 0
011100	State 0
111100	State 1
1**1**1100	State 2
11**1**100	State 1
111**1**00	State 2
1111**0**0	State 1

- Recognizable alternating sweeping motion as seen in alternating Christmas trees
- Right boundary of intermediate sweep does not at least reach the right boundary of the previous major sweep
- Current implementation assumes that each sweep spans at least as far as the previous sweep
- Only minor modifications to the alternating Christmas tree routine should be necessary to account for this behavior

B4 Holdouts

- The final holdout escapes the Christmas tree detection routine because of unusual startup effects



B4-startupEffectsChristmasTree1

B4-startup Effects
Christmas Tree1 Execution

0	State 0
1	State 1
1**0**	State 1
10**0**	State 2
1**0**0	State 2
100	State 2
0100	State 3
00100	State 0
10100	State 1
1**0**100	State 1
10**1**00	State 2
1**0**100	State 3
10100	State 0
010100	State 0
110100	State 1
1**1**0100	State 1
11**0**100	State 1
110**1**00	State 2
11**0**100	State 3
1**1**0100	State 0
110100	State 0
0110100	State 0
1110100	State 1
1**1**10100	State 1
11**1**0100	State 1
111**0**100	State 1
1110**1**00	State 2

- The Christmas tree detection routine runs the machine for a hundred transitions or so before looking for Christmas tree behavior to account for startup effects
- These transitions, however, are still observed to establish left and right boundaries for each sweep of the tape
- This machine creates a false right boundary during the startup phase
- Again, only minor modifications to the Christmas tree routine should be necessary to account for this behavior

Future Work

- Counter detection routines
  - Brady goes into more detail regarding the behavior of Counters, specifying a grammar in the same format as the Christmas tree grammar
  - Also mentions Counter variations (unary, binary, base-3, etc.)
- Christmas tree variations
  - Account for startup effects shown in B4 holdout
  - Uneven alternating Christmas trees
  - Multi-sweep (3,4,5… sweeps) alternating Christmas trees (seen in several of the random B5 holdouts that I've looked at)
  - Several more