7 Bouncers

Acknowledgement. Sincere thanks to Tony Guilfoyle who initially implemented a decider for bouncers[15]. Others have contributed to this method by producing alternative implementations (see Section 7.3) or discussing and writing the formal proof presented here: savask, Iijil, mei, Tristan Stérin (cosmo).

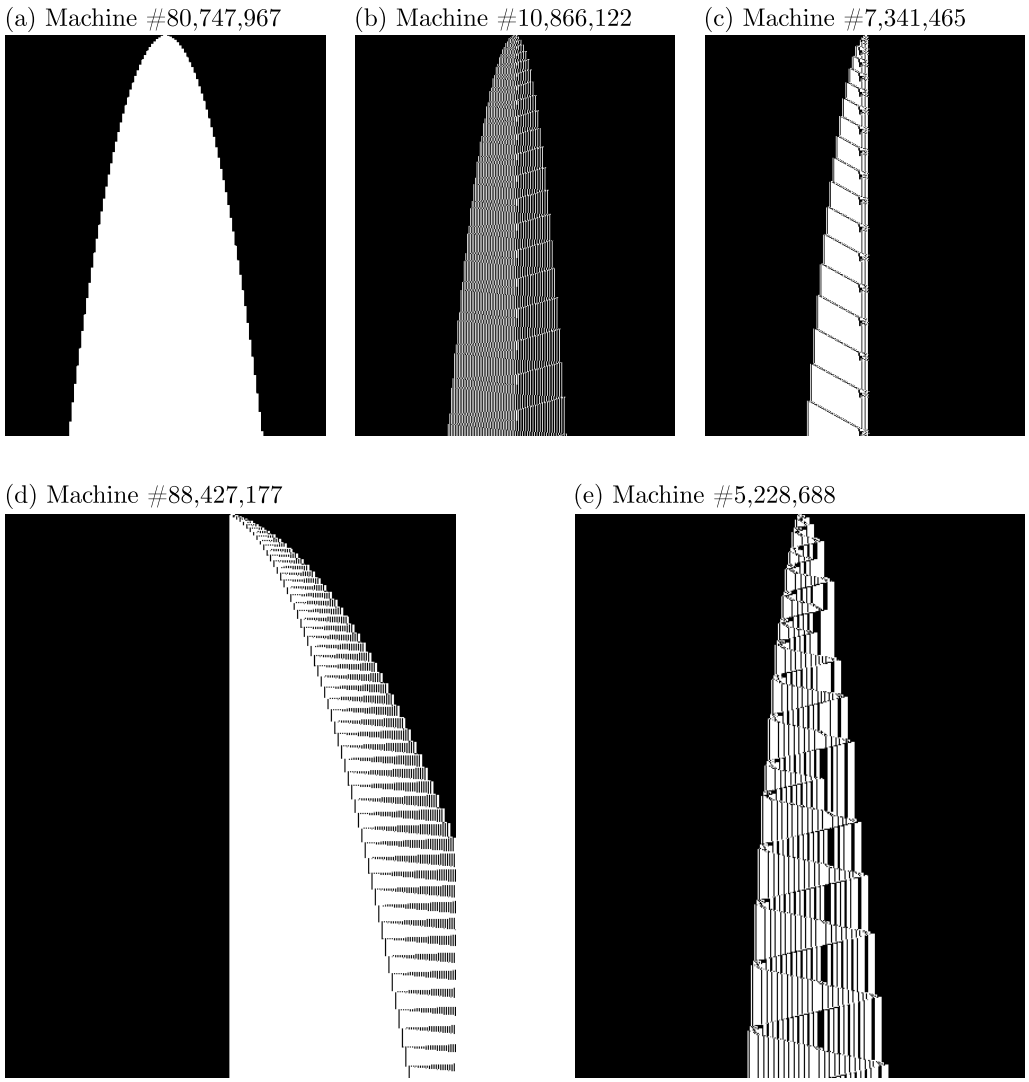

Figure 8: Space-time diagrams (10,000 steps) of several `bbchallenge` *bouncers*: (a) simple bouncer bouncing back and forth between expanding tape extremeties while writing 1s (b) bouncer with more complex alternating *repeater* patterns left and right of the origin (c) unilateral bouncer with a complex *wall* pattern at the origin (d) unilateral bouncer, main example used throughout this section (e) bouncer entering a repetitive bouncing pattern after ∼6,000 steps (bottom half of the image).

7.1 Characterising bouncers

Intuitively, a *bouncer* is a Turing machine that populates a tape with linearly-expanding patterns, called *repeaters*, possibly separated or enclosed by fixed patterns called *walls*. This intuitive definition corresponds to a wide range of behaviors, from simply bouncing back and forth between the tape's expanding extremities, Figure 8 (a), to complex traversal of *repeater* and *wall* patterns, Figure 8 (b)-(d), and possibly a delayed onset of the *bouncing* pattern, Figure 8 (e). What we call bouncers is a generalisation of the various classes of "Christmas trees" used to solve $BB(4)$ [2].

[15]See: `https://github.com/TonyGuil/bbchallenge/tree/main/Bouncers`.

The goal of this section is to formally characterise bouncers and show that they do not halt, see Theorem 7.9.

7.1.1 Directional Turing machines

We build on the concept of directional Turing machine introduced in Section 1. Directional Turing machines are an equivalent formulation of Turing machine where the machine head lives in between the tape cells and can point to the left or to the right. We choose here to treat 0^∞ as a unique symbol (instead of an infinite collection of 0s) and write $\overline{\Sigma} = \{0^\infty\} \cup \Sigma$, with $\Sigma = \{0, 1\}$. For each Turing machine with set of state S we introduce $2|S|$ new configuration symbols (i.e. symbols used to describe machine configurations and not read/write symbols) denoting the machine head in two possible orientations, namely $\Delta = \{\overset{s}{\triangleleft} \mid s \in S\} \cup \{\overset{s}{\triangleright} \mid s \in S\}$.

We define a *tape*[16] to be a finite word of the form uhv, where $u, v \in \overline{\Sigma}^*$ and $h \in \Delta$, moreover, u and v must have at most one occurrence of 0^∞ each, respectively as first symbol for u or last symbol for v. We choose the initial tape to be $0^\infty \overset{A}{\triangleright} 0^\infty$. Now, we define tape rewrite rules which will be used to simulate a directional Turing machine which we fix from now on. Given two tapes t and t', $t \vdash t'$ will denote that a rewrite rule transforms t into t'. Applying $\vdash$ successively $n \in \mathbb{N}$ times is written $\vdash^n$. The transitive closure of $\vdash$ (i.e. applying $\vdash$ *some* number of times) is written $\vdash^*$.

Suppose that for $s \in S, x \in \Sigma$ we have $\delta(s, x) = (s', d, x')$ where $s' \in S, d \in \{\mathrm{L}, \mathrm{R}\}, x' \in \Sigma$ and δ the transition function of the machine, then we define the following tape rewrite rules:

If $d = \mathrm{L}$	If $d = \mathrm{R}$
$x \overset{s}{\triangleleft} \to \overset{s'}{\triangleleft} x'$	$x \overset{s}{\triangleleft} \to x' \overset{s'}{\triangleright}$
$\overset{s}{\triangleright} x \to \overset{s'}{\triangleleft} x'$	$\overset{s}{\triangleright} x \to x' \overset{s'}{\triangleright}$

If $x = 0$, also define:

$0^\infty \overset{s}{\triangleleft} \to 0^\infty \overset{s'}{\triangleleft} x'$	$0^\infty \overset{s}{\triangleleft} \to 0^\infty x' \overset{s'}{\triangleright}$
$\overset{s}{\triangleright} 0^\infty \to \overset{s'}{\triangleleft} x' 0^\infty$	$\overset{s}{\triangleright} 0^\infty \to x' \overset{s'}{\triangleright} 0^\infty$

Given a tape $t = uvw$ and a word $t' = uv'w$, with $u, w \in \overline{\Sigma}^*$, $v, v' \in (\overline{\Sigma} \cup \Delta)^*$, suppose that there is a rewrite rule $v \to v'$. Then t' is also a tape and in this situation we write $t \vdash t'$ meaning that t' is obtained from t by one simulation step (i.e. $\vdash$ has the same meaning as for standard, non-directional, Turing machines, as introduced in Section 1); note that the definition is sound because, to any given tape, at most one rewrite rule applies, hence $v \to v'$ is defined uniquely and so is $t \vdash t'$. Note that the only case where $\vdash$ is not defined on a tape is if the machine halts in this configuration, i.e. an undefined transition is reached.

Example 7.1. Consider `bbchallenge` machine[17] #88,427,177 (Figure 8 (d)), that has the following transition table:

	0	1
A	1RB	1LE
B	1LC	1RD
C	1LB	1RC
D	1LA	0RD
E	—	$0LA$

[16] Note that in the terminology of Section 1, the definition of a tape that we use here, where the head in part of the tape, corresponds to a (partial) TM configuration.

[17] Accessible at `https://bbchallenge.org/88427177`

Given the above definitions, we will have the following rewrite rules with state A on the left-hand side:

$$\begin{aligned} 0 \overset{A}{\triangleleft} &\to 1 \overset{B}{\triangleright} \\ \overset{A}{\triangleright} 0 &\to 1 \overset{B}{\triangleright} \\ 1 \overset{A}{\triangleleft} &\to \overset{E}{\triangleleft} 1 \\ \overset{A}{\triangleright} 1 &\to \overset{E}{\triangleleft} 1 \\ 0^\infty \overset{A}{\triangleleft} &\to 0^\infty 1 \overset{B}{\triangleright} \\ \overset{A}{\triangleright} 0^\infty &\to 1 \overset{B}{\triangleright} 0^\infty \end{aligned}$$

The other rewrite rules are those with left-hand side states B, C, D and E. Simulating the machine for 4 steps, starting from initial tape yields: $0^\infty \overset{A}{\triangleright} 0^\infty \vdash 0^\infty 1 \overset{B}{\triangleright} 0^\infty \vdash 0^\infty 1 \overset{C}{\triangleleft} 1 0^\infty \vdash 0^\infty 1 \overset{C}{\triangleright} 1 0^\infty \vdash 0^\infty 11 \overset{C}{\triangleright} 0^\infty$. Hence, $0^\infty \overset{A}{\triangleright} 0^\infty \vdash^* 0^\infty 11 \overset{C}{\triangleright} 0^\infty$.

7.1.2 Wall-repeater formula tapes

We want to formalise the above-stated intuition that bouncers expand a tape by repeating *repeater* patterns between fixed *walls*, and then show that bouncers do not halt, Theorem 7.9. For this, we are going to express bouncers' tapes using regular expressions over the alphabet $\Delta \cup \overline{\Sigma}$. Given $u \in \Sigma^*$, we abbreviate the regular expression $(u)^*$, which represents zero or more repetitions of the word u, as (u). Also, we write $\Sigma^+ = \Sigma^* \setminus \{\varnothing\}$. We define a *wall-repeater formula tape* to be a regular expression of the form:

$$w_1(r_1)w_2(r_2)\dots w_n(r_n)w_{n+1}hw'_1(r'_1)w'_2(r'_2)\dots w'_m(r'_m)w'_{m+1} \tag{7.1}$$

if the following conditions are met:

$$\begin{aligned} &n, m \geq 0, \\ &h \in \Delta, \\ &r_1, \dots, r_n, r'_1, \dots, r'_m \in \Sigma^+, \\ &w_2, \dots, w_{n+1}, w'_1, \dots, w'_m \in \Sigma^*, \\ &w_1, w'_{m+1} \in \overline{\Sigma}^* \end{aligned}$$

and, as in the definition of a tape, w_1 (resp. w'_{m+1}) either does not contain the symbol 0^∞ or it starts with it (resp. ends with it). Patterns w_i, w'_j are called *walls* and can be empty, while the *repeaters*, r_i, r'_j must be nonempty. Note that we allow $n = m = 0$, hence a usual tape is also a wall-repeater formula tape. For the rest of this section, we abbreviate wall-repeater formula tapes as *formula tapes*.

Given a formula tape f let $\mathcal{L}(f)$ denote the language described by f, i.e. the set of tapes that match it.

Example 7.2. Consider the formula tape $f = 0 \overset{D}{\triangleright} (01)$. We have $\mathcal{L}(f) = \{0 \overset{D}{\triangleright}, 0 \overset{D}{\triangleright} 01, 0 \overset{D}{\triangleright} 0101, \dots\}$. Consider the formula tape $f' = 0^\infty(111)1110 \overset{A}{\triangleleft} 010101(01)10^\infty$. Then: $0^\infty 1110 \overset{A}{\triangleleft} 01010110^\infty$, $0^\infty 1111110 \overset{A}{\triangleleft} 01010110^\infty$, and $0^\infty 1110 \overset{A}{\triangleleft} 0101010110^\infty$ are three elements of $\mathcal{L}(f')$.

Extending $\vdash$ to formula tapes: *shift rules*. We wish to extend the Turing machine step relation $\vdash$ (see Section 7.1.1) to formula tapes. This is quite straightforward when the formula tape's head is pointing at a symbol of a wall (one of the w_i, w'_j in (7.1)): we simply apply a standard Turing machine step, leaving the definition of $\vdash$ unchanged.

However, we need to handle the case where the head is pointing at a repeater (one of the r_i, r'_j in (7.1)). Suppose that for some $u \in \Sigma^*$ and $r, \tilde{r} \in \Sigma^+$ and some state $s \in S$ we have $u \overset{s}{\triangleright} r \vdash^* \tilde{r}u \overset{s}{\triangleright}$. Then, for any $n \geq 0$, we have $u \overset{s}{\triangleright} r^n \vdash^* \tilde{r}^n u \overset{s}{\triangleright}$. This motivates the definition of (right) *shift rules*, rewrite rules for formula tapes, which rewrite a subword $u \overset{s}{\triangleright} (r)$ into $(\tilde{r})u \overset{s}{\triangleright}$, denoted $u \overset{s}{\triangleright} (r) \to (\tilde{r})u \overset{s}{\triangleright}$. Similarly, left shift rules are of the form $(r) \overset{s}{\triangleleft} u \to \overset{s}{\triangleleft} u(\tilde{r})$ given that $r \overset{s}{\triangleleft} u \vdash^* \overset{s}{\triangleleft} u\tilde{r}$. Note that repeaters r and $\tilde{r}$ of a shift rule necessarily have the same size since $\vdash$ preserves tapes' size.

Hence, we have two cases to consider for defining $\vdash$ on the following formula tape f (as defined in (7.1)):

$$f = w_1(r_1)w_2(r_2)\dots w_n(r_n)w_{n+1}hw'_1(r'_1)w'_2(r'_2)\dots w'_m(r'_m)w'_{m+1}$$

1. (Usual step) If h points to nonempty w_{n+1} or w'_1, then $f \vdash \alpha\tilde{w}_{n+1}\tilde{h}\tilde{w}'_1\beta$ if $w_{n+1}hw'_1 \vdash \tilde{w}_{n+1}\tilde{h}w'_1$, $\tilde{w}_{n+1}, \tilde{w} \in \Sigma^*$, $\tilde{h} \in \Delta$, and α, β the above uniquely-defined beginning and ending of f. As for tapes, $\vdash$ is undefined if $w_{n+1}hw'_1$ corresponds to a halting configuration (i.e. undefined transition).

2. (Shift rule) Two cases:

   (a) Right shift rule. If $h = \overset{s\in S}{\triangleright}$ and w'_1 is empty, consider the set of shift rules $\mathcal{R} = \{u \overset{s}{\triangleright} (r'_1) \to (\tilde{r})u \overset{s}{\triangleright} \mid \tilde{r} \in \Sigma^+,\ u \in \Sigma^* \text{ is a suffix of } w_{n+1}\}$. If $\mathcal{R}$ is not empty then apply the right shift rule of $\mathcal{R}$ with smallest u (possibly empty), call the new formula f', and we have $f \vdash f'$. If $\mathcal{R}$ is empty, $\vdash$ cannot be applied to f.

   (b) Left shift rule. If $h = \overset{s\in S}{\triangleleft}$ and w_{n+1} is empty, consider the set of shift rules $\mathcal{R} = \{(r_n) \overset{s}{\triangleleft} u \to \overset{s}{\triangleleft} u(\tilde{r}) \mid \tilde{r} \in \Sigma^+,\ u \in \Sigma^* \text{ is a prefix of } w'_1\}$. If $\mathcal{R}$ is not empty then apply the left shift rule of $\mathcal{R}$ with smallest u (possibly empty), call the new formula f', and we have $f \vdash f'$. If $\mathcal{R}$ is empty, $\vdash$ cannot be applied to f.

From the above definition of $\vdash$ on a formula tape f, it is clear that (i) for usual tapes our new definition of $\vdash$ coincides with the old one, (ii) there is at most one formula tape f' such that $f \vdash f'$, and (iii) the only cases where $\vdash$ is not defined on a formula tape are when the machine halts (usual step case) or no shift rule applies (shift rule case). Moreover, we get the following result:

Lemma 7.3. Let f and f' be formula tapes with $f \vdash f'$. Then, for all $t \in \mathcal{L}(f)$, there exists some $t' \in \mathcal{L}(f')$ such that $t \vdash^* t'$.

Proof. If f' follows from f by a usual step, then applying one usual step to t yields $t' \in \mathcal{L}(f')$. If f' follows from f by a shift rule, we apply several steps to t (as many as there in the shift rule) to obtain $t' \in \mathcal{L}(f')$. □

Example 7.4. Taking the machine of Example 7.1, we have the right shift rule $0 \overset{D}{\triangleright} (01) \to (11)0 \overset{D}{\triangleright}$. Indeed, this is because: $0 \overset{D}{\triangleright} 01 \vdash 0 \overset{A}{\triangleleft} 11 \vdash 1 \overset{B}{\triangleright} 11 \vdash 11 \overset{D}{\triangleright} 1 \vdash 110 \overset{D}{\triangleright}$, hence $0 \overset{D}{\triangleright} 01 \vdash^* 110 \overset{D}{\triangleright}$, giving the shift rule. Consider the tape formula of previous Example 7.2: $f' = 0^\infty(111)1110 \overset{A}{\triangleleft} 010101(01)10^\infty$. We have $f' \vdash^{13} 0^\infty(111)1111110110 \overset{D}{\triangleright} (01)10^\infty$, at this point the head points to a repeater and the set of applicable right shift rules is $\mathcal{R} = \{0 \overset{D}{\triangleright} (01) \to (11)0 \overset{D}{\triangleright},\ 10 \overset{D}{\triangleright} (01) \to (11)10 \overset{D}{\triangleright},\ 110 \overset{D}{\triangleright} (01) \to (11)110 \overset{D}{\triangleright}\}$, and following the definition of $\vdash$, we apply $0 \overset{D}{\triangleright} (01) \to (11)0 \overset{D}{\triangleright}$ as it has the smallest left-hand side, giving: $0^\infty(111)111111011(11)0 \overset{D}{\triangleright} 10^\infty$.

Aligning formula tapes. One last tool that we need before characterising bouncers and proving that they do not halt (Theorem 7.9) is formula tape *alignment* (Definition 7.5): sometimes it is necessary to rewrite a formula tape in an equivalent, *aligned* form in order for any shift rules to apply.

Definition 7.5 (Alignment operator)**.** Take a formula tape, as given in (7.1):

$$f = w_1(r_1)w_2(r_2)\dots w_n(r_n)w_{n+1}hw'_1(r'_1)w'_2(r'_2)\dots w'_m(r'_m)w'_{m+1}$$

The alignment operator $f \mapsto \mathcal{A}(f)$ moves repeaters away from the head h by repeatedly applying any of the following rules until none apply anymore:

1. Replace $(r'_j)v$ with $v(r)$ in f, if $r'_j v = vr$ with v a nonempty prefix of w'_{j+1}, $r \in \Sigma^+$, and $1 \le j \le m$.

2. Replace $v(r_i)$ with $(r)v$ in f, if $vr_i = rv$ with v a nonempty suffix of w_i, $r \in \Sigma^+$, and $1 \le i \le n$.

Clearly, the order application of these rules does not matter, i.e. $\mathcal{A}$ is well-defined, and $\mathcal{A}(\mathcal{A}(f)) = \mathcal{A}(f)$.

Lemma 7.6. For any formula tape f, $\mathcal{L}(\mathcal{A}(f)) = \mathcal{L}(f)$, i.e. both f and $\mathcal{A}(f)$ represent the same set of tapes.

Proof. Consider an alignment rule as in case (1) of Definition 7.5: replacing $(r'_j)v$ with $v(r)$ if $r'_j v = vr$. Then, for all $n \in \mathbb{N}$, $r'^n_j v = vr^n$, hence $(r'_j)v$ and $v(r)$ describe the same language: $\mathcal{L}((r'_j)v) = \mathcal{L}(v(r))$. Same for case (2) and for multiple applications of (1) and (2) in any order, hence we have $\mathcal{L}(\mathcal{A}(f)) = \mathcal{L}(f)$. □

Example 7.7. Take $f = 0^\infty(111)1111 \overset{B}{\triangleright} (01)01010110^\infty$ for the machine given in Example 7.1. One can verify that no right shift rule applies to f hence $\vdash$ does not apply to f. However, we have $\mathcal{A}(f) = 0^\infty(111)1111 \overset{B}{\triangleright} 010101(01)10^\infty$, $\mathcal{L}(\mathcal{A}(f)) = \mathcal{L}(f)$, and we can apply $\vdash$ to $\mathcal{A}(f)$ by performing usual steps: $\mathcal{A}(f) \vdash^{12} 0^\infty(111)1111110110 \overset{D}{\triangleright} (01)10^\infty$ and now the shift rule of Example 7.4 can apply, giving $0^\infty(111)111111011(11)0 \overset{D}{\triangleright} 10^\infty$. Another alignment example is $f = 0^\infty 101(11)0 \overset{D}{\triangleright} 10(100)110^\infty$ for which $\mathcal{A}(f) = 0^\infty 10(11)10 \overset{D}{\triangleright} 101(001)10^\infty$.

Remark 7.8 (Alignment never hurts)**.** Example 7.7 shows that alignment (which preserves the set of recognised tapes) can allow to run a shift rule on f' with $\mathcal{A}(f) \vdash^* f'$ when no shift rule was applicable directly to f. In fact, one can show that the alignment operator can only *increase* the number of applicable shift rules: if a shift rule is applicable to f then a shift rule applicable to f' with $\mathcal{A}(f) \vdash^* f'$ can always be constructed.

Given two formula tapes f and f' we will say that f' is a *special case* of f, if $\mathcal{A}(f')$ can be obtained from $\mathcal{A}(f)$ by replacying subwords of the form (r) by $r^n(r)r^m$ for some $n, m \geq 0$ and $r \in \Sigma^+$. Note that because of alignment, $n = 0$ (resp. $m = 0$) if (r) is to the left (resp. to the right) of the head. If f is a special case of f' then $\mathcal{L}(f) \subseteq \mathcal{L}(f)$, the converse is open but conjectured true[18] (under additional assumptions).

We finally get to the main result of this section, which characterises bouncers formally – a bouncer is any machine to which Theorem 7.9 applies:

Theorem 7.9 (Bouncers)**.** Assume we are given some Turing machine and suppose there exists a tape t and some wall-repeater formula tapes $f_0, \ldots, f_n, n \geq 1$, such that $0^\infty \overset{A}{\triangleright} 0^\infty \vdash^* t$ where $t \in \mathcal{L}(f_0)$, and $\mathcal{A}(f_i) \vdash f_{i+1}$ for $i = 0, \ldots, n-1$. If f_n is a special case of f_0, then the Turing machine does not halt.

Proof. It follows from Lemma 7.3 and Lemma 7.6 that there exists tapes $t_0, \ldots, t_n$, such that $t_0 = t$ and $t_i \in \mathcal{L}(f_i)$ for $0 \leq i \leq n$, and $t_i \vdash^* t_{i+1}$ for $0 \leq i < n$, giving $t_0 \vdash^* t_n$. Since f_n is a special case of f_0, we have $\mathcal{L}(\mathcal{A}(f_n)) \subseteq \mathcal{L}(\mathcal{A}(f_0))$, and using Lemma 7.6, we have $\mathcal{L}(f_n) \subseteq \mathcal{L}(f_0)$ and thus, $t_n \in \mathcal{L}(f_0)$, we can repeat this construction indefinitely and yield an infinite sequence of tapes $(t_n)_{n\in\mathbb{N}}$ such that $t \vdash^* t_i$ for all $i \in \mathbb{N}$: the machine does not halt. □

Example 7.10. Theorem 7.9 applies to the machine of Example 7.1, illustrated in Figure 8 (d), used in our series of examples for this section. Indeed, we have $0^\infty \overset{A}{\triangleright} 0^\infty \vdash^{64} 0^\infty 11111101100 \overset{D}{\triangleright} 0^\infty$, this tape, $t = 0^\infty \vdash^{64} 0^\infty 11111101100 \overset{D}{\triangleright} 0^\infty$ is in the language the following formula tape:

$$f_0 = 0^\infty(111)1110(11)00 \overset{D}{\triangleright} 0^\infty$$

At this point, and for the next 25 usual steps alignment does not affect the formulas, and we get:

$$f_{25} = 0^\infty(111)1110(11) \overset{A}{\triangleleft} 01010110^\infty$$

One shift rule gives:

$$f_{26} = 0^\infty(111)1110 \overset{A}{\triangleleft} (01)01010110^\infty$$

After alignment:

$$\mathcal{A}(f_{26}) = 0^\infty(111)1110 \overset{A}{\triangleleft} 010101(01)10^\infty$$

From there, $\mathcal{A}(f_{26}) \vdash f_{27}$ with:

$$f_{27} = 0^\infty(111)1111 \overset{B}{\triangleright} 010101(01)10^\infty$$

[18] See `https://discuss.bbchallenge.org/t/186`.

After 12 usual steps, not affected by alignment, we arrive at:

$$f_{39} = 0^\infty(111)1111110110 \overset{\mathrm{D}}{\triangleright} (01)10^\infty$$

One shift rule gives:

$$f_{40} = 0^\infty(111)111111011(11)0 \overset{\mathrm{D}}{\triangleright} 10^\infty$$

Aligning gives:

$$\mathcal{A}(f_{40}) = 0^\infty(111)1111110(11)110 \overset{\mathrm{D}}{\triangleright} 10^\infty$$

Finally, from there $\mathcal{A}(f_{40}) \vdash f_{41}$ with one usual step:

$$f_{41} = \mathcal{A}(f_{41}) = 0^\infty(111)1111110(11)1100 \overset{\mathrm{D}}{\triangleright} 0^\infty$$

Now, f_{41} is a special case of f_0 because $\mathcal{A}(f_{41})$ differs from $f_0 = \mathcal{A}(f_0)$ only by including one repetition of repeaters (111) and (11) in the walls directly to their right. The assumptions of Theorem 7.9 hold and our Turing machine is a bouncer: it does not halt.

Remark 7.11. Both Cyclers (Section 2) and Translated Cyclers (Section 3) are special cases of bouncers, as they satisfy Theorem 7.9.

7.1.3 Linear-quadratic growth

Theorem 7.9 characterises bouncers but we lack a practical criterion for recognising a bouncer from its sequence of successive tapes starting from $0^\infty \overset{\mathrm{A}}{\triangleright} 0^\infty$. We make a step in that direction by showing that if a bouncer is not a cycler (Section 2), we can expect to see tapes of the form $w_1 r_1^n w_2 r_2^n \dots w_m r_m^n w_{m+1}$ for some $m \in \mathbb{N}$ and for all successive $n \in \mathbb{N}$ as a subsequence of the sequence of all tapes visited by the bouncer.

We define *record-breaking tapes* to be tapes where the head is pointing at 0^∞, i.e. tapes of the form $0^\infty u \overset{s}{\triangleright} 0^\infty$ or $0^\infty \overset{s}{\triangleleft} u0^\infty$ with $u \in \Sigma^*$. Give a tape t, we call $\mathcal{S}(t) \in \Sigma^*$ the *headless* version of t which is the tape without head and 0^∞ symbols. The *length* of a tape t is defined as the number of symbols in $\mathcal{S}(t)$. We also introduce headless formula tapes: $\mathcal{S}(f)$ is f without head and 0^∞ symbols. Also, we extend the alignment operator $\mathcal{A}$ (Definition 7.5) to headless formula tapes by performing right-alignment. We denote an infinite sequence $u_0, u_1, \dots u_n, \dots$ with $n \in \mathbb{N}$ by $(u_n)_{n\in\mathbb{N}}$. We define the difference operator $D : \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ via $D((u_n)_{n\in\mathbb{N}}) = (u_{n+1} - u_n)_{n\in\mathbb{N}}$.

Bouncers will grow their tape such that we can extract a subsequence of record-breaking tapes of the form $w_1 r_1^n w_2 r_2^n \dots w_m r_m^n w_{m+1}$ with length in arithmetic progression and time steps in quadratic progression:

Theorem 7.12 (*Linear-quadratic growth*)**.** Let M be a bouncer Turing machine, i.e. it satisfies Theorem 7.9 for some formula tape. Call $(t_n)_{n\in\mathbb{N}}$ the sequence of tapes that it visits starting from $t_0 = 0^\infty \overset{\mathrm{A}}{\triangleright} 0^\infty$. Call l_n the length of t_n and suppose that $\lim_{n\to\infty} l_n = +\infty$, i.e. M is not a cycler. Then, there is an integer $m \geq 1$ and $w_1, \dots, w_{m+1} \in \Sigma^*$ and $r_1, \dots, r_m \in \Sigma^+$, and $g : \mathbb{N} \to \mathbb{N}$ with $g(n) \geq n$ for all $n \in \mathbb{N}$ that extracts a subsequence of $(t_n)_{n\in\mathbb{N}}$ such that:

1. There is a formula tape f with head pointing at 0^∞ that solves the bouncer according to Theorem 7.9 and with $\mathcal{S}(f) = w_1(r_1)w_2(r_2)\dots w_m(r_m)w_{m+1}$.
2. All tapes $t_{g(n)}$ are record-breaking with the same head state and pointing-direction.
3. $\mathcal{S}(t_{g(n)}) = w_1 r_1^n w_2 r_2^n \dots w_m r_m^n w_{m+1}$ for all $n \in \mathbb{N}$.
4. $(l_{g(n)})_{n\in\mathbb{N}}$ is an arithmetic progression, i.e. $D((l_{g(n)})_{n\in\mathbb{N}}) = (K)_{n\in\mathbb{N}}$ for some constant $K \in \mathbb{N}$.
5. $(g(n))_{n\in\mathbb{N}}$ is a quadratic progression, i.e. $D(D((g(n))_{n\in\mathbb{N}})) = (K')_{n\in\mathbb{N}}$ for some constant $K' \in \mathbb{N}$.

Proof. Because M is a bouncer, there is a formula tape (See (7.1)):

$$\tilde{f} = 0^\infty \tilde{w}_1(\tilde{r}_1)\tilde{w}_2(\tilde{r}_2)\dots\tilde{w}_n(\tilde{r}_n)\tilde{w}_{n+1}h\tilde{w'}_1(r'_1)\tilde{w'}_2(\tilde{r'}_2)\dots\tilde{w'}_m(\tilde{r'}_m)\tilde{w'}_{m+1}0^\infty$$

With $h \in \Delta$, $\tilde{w}_i, \tilde{w}'_j \in \Sigma^*$, $\tilde{r}_i, \tilde{r}_j \in \sigma^+$.

There is also $\tilde{f}_0 = \tilde{f}$, $\tilde{f}_1, \dots \tilde{f}_k$ with $k \in \mathbb{N}$ such that $\mathcal{A}(\tilde{f}_i) \vdash \tilde{f}_{i+1}$ for $0 \geq i < k$ and $\tilde{f}_k$ is a special case of $\tilde{f}_0$.

Because $\lim_{n\to\infty} l_n = +\infty$, there must be $0 \leq j < k$ such that the head of $\tilde{f}_j$ is pointing to 0^∞ (otherwise the tape would never grow). Hence, without loss of generality, by rotating the sequence of formula tapes, we can assume that the head of $\tilde{f}_0 = \tilde{f}$ is pointing at 0^∞, so does the head of $\tilde{f}_k$, by special case. Furthermore, we can assume that each wall preceding a repeater in $\tilde{f}_k$ is bigger than its corresponding wall in $\tilde{f}_0$, otherwise the repeater is useless (it never grows) and can be merged into its preceding wall in both formulas.

Hence, from the above and because $\tilde{f}_k$ is a special case of $\tilde{f}_0$, renaming the walls' and repeaters' symbols we get:

$$\mathcal{A}(\mathcal{S}(\tilde{f})) = \tilde{w}_1(\tilde{r}_1)\tilde{w}_2(\tilde{r}_2)\dots\tilde{w}_n(\tilde{r}_n)\tilde{w}_{n+1}$$
$$\mathcal{A}(\mathcal{S}(\tilde{f}_k)) = \tilde{w}_1\tilde{r}_1^{m_1}(\tilde{r}_1)\tilde{w}_2\tilde{r}_2^{m_2}(\tilde{r}_2)\dots\tilde{w}_n\tilde{r}_2^{m_n}(\tilde{r}_n)\tilde{w}_{n+1}$$

With $m_1, \dots, m_n \geq 1$.

From Theorem 7.9, we know that the machine reaches a tape $t \in \mathcal{L}(\tilde{f})$. We can furthermore suppose that the machine will reach the tape $\tilde{w}_1\tilde{w}_2\dots\tilde{w}_n\tilde{w}_{n+1}$ as we simply have to redefine our walls if it reaches some tape with more initial repetitions of the repeaters instead. Hence, by successive simulation and alignment of the formula tape f with same head as $\tilde{f}_0$ such that $\mathcal{A}(\mathcal{S}(f)) = \tilde{w}_1(\tilde{r}_1^{m_1})\tilde{w}_2(\tilde{r}_2^{m_2})\dots\tilde{w}_n(\tilde{r}_2^{m_n})\tilde{w}_{n+1}$, renamed as $\mathcal{A}(\mathcal{S}(f)) = w_1(r_1)w_2(r_2)\dots w_n(r_2)w_{n+1}$, we are assured to solve the bouncer (i.e. reach a special case) and, by continuing simulation, to generate an infinite subsequence of reached tapes of the form $\mathcal{S}(t_{g(n)}) = w_1r_1^nw_2r_2^n\dots w_mr_m^nw_{m+1}$ for all $n \in \mathbb{N}$ and for $g : \mathbb{N} \to \mathbb{N}$ some extraction function with $g(n) \geq n$ for all $n \in \mathbb{N}$.

All such tapes are record-breaking with same head, by construction. Hence, we proved points 1-3. Point 4 comes easily because from the expression we have for $t_{g(n)}$, we get $l_{g(n+1)} - l_{g(n)} = |r_1| + \dots |r_m| = K$, which is constant. Concerning point 5, we have $g(n+1) - g(n) = M + ns_1 + ns_2 + \dots ns_m$, with $M \in \mathbb{N}$ a constant related to processing walls and $s_1, \dots s_m \in \mathbb{N}$ the constant time needed to run the shift rule associated to each repeater $r_1 \dots r_m$. Hence we get $D(D((g(n))_{n\in\mathbb{N}})) = s_1 + \dots + s_n = K'$, which is constant.

□

Example 7.13. Using the machine of Example 7.1 used in the previous section, we have:

$$\begin{aligned}
0^\infty \overset{A}{\triangleright} 0^\infty &\vdash^{37} 0^\infty 111\ 01111 \overset{C}{\triangleright} 0^\infty && g(0) = 37,\ l_{37} = 11\\
&\vdash^{39} 0^\infty 111\ 111\ 01111\ 11 \overset{C}{\triangleright} 0^\infty && g(1) = 76,\ l_{76} = 16\\
&\vdash^{45} 0^\infty 111\ 111\ 111\ 01111\ 11\ 11 \overset{C}{\triangleright} 0^\infty && g(2) = 121,\ l_{121} = 21\\
&\vdash^{51} 0^\infty 111\ 111\ 111\ 111\ 01111\ 11\ 11\ 11 \overset{C}{\triangleright} 0^\infty && g(3) = 172,\ l_{172} = 26
\end{aligned}$$

Each of these tapes are record-breaking using the same head $\overset{C}{\triangleright}$. Then, in this finite sample, $l_{g(0)}, ..., l_{g(3)}$ is in arithmetic progression with first difference $K = 5$ and $g(0), ..., g(3)$ is in quadratic progression with second difference $K' = 6$.

In general, we get:

$$0^\infty \overset{A}{\triangleright} 0^\infty \vdash^{g(n)} 0^\infty 111\ (111)^n\ 01111(11)^n \overset{C}{\triangleright} 0^\infty \quad g(n) = 3n^2 + 36n + 37,\ l_{g(n)} = 5n + 11$$

This general expression can be verified by simulating (and aligning) the formula tape $f = 111(111)01111(11) \overset{C}{\triangleright} 0^\infty$ until it reaches a special case, which solves the bouncer (Theorem 7.9).

7.2 Deciding bouncers in practice

In this section, we use the theory presented above in order to give a practical and efficient algorithm for deciding bouncers.

7.2.1 Formula tape fitting

Theorem 7.12 assures us that, in the sequence of successive tapes of a bouncer, we'll always find a subsequence where all repeaters' repetitions successively increase by one. Given such a subsequence, we are left with the task of fitting an appropriate formula tape. It turns out that only 3 successive tapes (t_0, t_1, t_2) in such subsequence, with at least one repetition of each repeater in the first tape, are enough to fit a formula tape efficiently using a greedy algorithm which recursively constructs the left-most wall using the biggest common prefix of t_0, t_1 and t_2 and then fits the left-most repeater using the longest prefix r of t_1 such that rr is a prefix of t_2:

Algorithm 7 Greedy formula tape fitting algorithm FIT-FORMULATAPE

```
1:  procedure FormulaTape FIT-FORMULATAPE(Word t0, Word t1, Word t2)
2:      if t0.empty() && t1.empty() && t2.empty() then
3:          return FormulaTape::empty()
4:
5:      if t0.len() > 0 && t1.len() > 0 && t2.len() > 0 && t0[0] == t1[0] && t1[0] == t2[0] then
6:          return FormulaTape::symbol(t0[0]).concat(FIT-FORMULATAPE(t0[1:], t1[1:], t2[1:]))
7:
8:      uint longest_prefix_size = get_longest_prefix_size(t1, t2)
9:
10:     for l = longest_prefix_size; k ≥ 1; k -= 1  do
11:         if 2*l < t2.len() && t2[:l] == t2[l:2*l]  then
12:             return FormulaTape::repeater(t2[:l]).concat(FIT-FORMULATAPE(t0, t1[l:], t2[2*l:]))
13:
14:     raise Failure
```

Remark 7.14. In Algorithm 7, we assume that we are given a **FormulaTape** construct (see Equation (7.1)) that we use for headless formula tapes and a function **get_longest_match_size** which returns the size of the longest prefix of two words. Furthermore, for an array `a`, we use the notation `a[b:e]` to mean the slice of this array between indices `b` and `e`, excluding `e`.

Theorem 7.15 (Greedy formula tape fitting)**.** Let there be 3 words $t_0, t_1, t_2 \in \Sigma^*$, an integer $m > 0$ and nonempty intermediary walls $w_2, \dots w_m \in \Sigma^+$ and possibly-empty initial and final walls $\alpha, \beta \in \Sigma^*$ and nonempty repeaters $r_1, \dots, r_m \in \Sigma^+$ so that they are the longest repeaters for t_0, t_1, t_2 to be written as:

$$\begin{aligned} t_0 &= \alpha\ w_2 \dots w_m\ \beta \\ t_1 &= \alpha\ r_1\ w_2\ \dots w_m\ r_m\ \beta \\ t_2 &= \alpha\ r_1 r_1\ w_2\ \dots w_m\ r_m r_m\ \beta \end{aligned}$$

Then Algorithm 7 ran on FIT-FORMULATAPE(t_0, t_1, t_2) does not raise failure and returns the right-aligned headless formula tape $\mathcal{A}(\alpha(r_1)w_2(r_2)\dots w_m(r_m)\beta)$.

Proof. Let's first prove the case $m = 1$ and write $r = r_1$. Notation: if a word $w \in \Sigma^*$ is not empty we use the notation $w[0]$ to mean the first symbol of w. Using the case of Algorithm 7, line 10, we get:

$$\begin{array}{lll} t_0 = \alpha\ \beta & & t_0 = \beta \\ t_1 = \alpha\ r\ \beta & \rightarrow_{\text{Algorithm 7 goes to}} & t_1 = r\ \beta \\ t_2 = \alpha\ rr\ \beta & & t_2 = rr\ \beta \\ f = \varnothing & & f = \alpha \end{array}$$

From there, we have three cases: (1) if $\beta = \varnothing$ then the greedy algorithm returns $f = \alpha(r)$, right-aligned, as needed, (2) if $\beta[0] \neq r[0]$ (r is not empty by hypothesis), then the greedy algorithm returns $f = \alpha(r)\beta$ which is right-aligned (because $\beta[0] \neq r[0]$), as needed and (3) if $\beta[0] = r[0]$ then, call $\beta = u\beta'$ and $r = ur'$ with $|u| > 0$ and $\beta'[0] \neq r'[0]$ if they are both nonempty, the system rewrites as:

$$\begin{aligned} t_0 &= u\beta' & & t_0 = \beta' \\ t_1 &= ur'\ u\beta' & \rightarrow_{\text{Algorithm 7 goes to}} & t_1 = r'\ u\beta' \\ t_2 &= ur'ur'\ u\beta' & & t_2 = r'ur'\ u\beta' \\ f &= \alpha & & f = \alpha u \end{aligned}$$

Two cases:

- If r' is nonempty, we are in the previously seen case $(\beta, r\beta, rr\beta)$ by assimilating β' to β and ur' to r, except that case (3) cannot happen since we've chosen β' and r' such that $\beta'[0] \neq r'[0]$ if they are both nonempty. Hence, the outputted formula is $f = \alpha u(r'u)\beta'$ which is the right-aligned version of $\alpha(r)\beta$ (because $\beta'[0] \neq r'[0]$, if both nonempty), as needed.
- If r' is empty, we are also in the previously seen case $(\beta, r\beta, rr\beta)$ by assimilating β' to β and u to r, however $|\beta'| < |\beta|$, hence, by induction, using the same set of 3 cases, this system will eventually reach case (1) $\beta' = \varnothing$ or case (2) $\beta'[0] = u[0]$ where we can conclude.

In the general case, using the case $m = 1$ we have:

$$\begin{aligned} t_0 &= w_1\ r_1\ w_2r_2\ \ldots w_m\ r_m\ w_{m+1} & & t_0 = r_2\ \ldots w_m\ r_m\ w_{m+1} \\ t_1 &= w_1\ r_1r_1\ w_2r_2r_2\ \ldots w_m\ r_mr_m\ w_{m+1} & \rightarrow_{\text{Algorithm 7 goes to}} & t_1 = r_2r_2\ \ldots w_m\ r_mr_m\ w_{m+1} \\ t_2 &= w_1\ r_1r_1r_1\ w_2r_2r_2r_2\ \ldots w_m\ r_mr_mr_m\ w_{m+1} & & t_2 = r_2r_2r_2\ \ldots w_m\ r_mr_mr_m\ w_{m+1} \end{aligned}$$

From there, we get the theorem by induction.

□

Remark 7.16 (Minimal formula tapes)**.** While Algorithm 7 is assured, by Theorem 7.15, to fit a formula tape (if it exists) starting from t_0 containing at least one occurrence of each repeater, the algorithm can fail when t_0 has none. Here is an example trying to parse 1011(01011)(1)0 starting from $t_0 = 1011\ 0$:

$$\begin{aligned} t_0 &= 1011\ 0 & t_0 &= \varnothing & & t_0 = \varnothing \\ t_1 &= 1011\ 01011\ 1\ 0 \quad \rightarrow_{\text{Algorithm 7 goes to}} & t_1 &= 1011\ 1\ 0 & \rightarrow_{\text{Algorithm 7 goes to}} & t_1 = 1\ 1\ 0 \\ t_2 &= 1011\ 01011\ 01011\ 1\ 1\ 0 & t_2 &= 1011\ 01011\ 1\ 1\ 0 & & t_2 = 011\ 1\ 1\ 0 \\ f &= \varnothing & f &= 10110 & & f = 10110(101) \end{aligned}$$

At this point, the greedy algorithm raises failure since there are no more wall symbols in t_0 and no prefix r of t_1 such that rr is a prefix of t_2 since $t_1[0] \neq t_2[0]$. However, starting from $t_0 = 1011\ 01011\ 1\ 0$ it will successfully fit 101101011(01011)1(1)0 which is a special case of 1011(01011)(1)0.

Hence, the greedy algorithm does not always fit *minimal* formula tapes. By adding backtracking and memoization, it can easily be transformed into a Dynamic Programming algorithm that will succeed starting from smaller tapes, i.e. t_0 with no repeater occurrence, and hence fit minimal formula tapes, at the cost of being less efficient (quadratic vs. linear time complexity). Note that another approach to formula tape minimisation is to deduce a minimal formula tape from the solution outputted by the greedy algorithm, essentially by removing occurrences of repeaters appearing in walls as much as possible, in linear time.

7.2.2 Bouncers decider

Using Theorem 7.9, Theorem 7.12 and Algorithm 7 together gives a straightforward way to decide bouncers:

1. Track record-breaking tapes with same head state and pointing-direction that grow linearly in quadratic time.
2. Run Algorithm 7 on triples of headless tapes extracted from these record-breaking sequences, until a formula tape satisfying Theorem 7.9 is found or some limits are met.

If the machine is a bouncer, by Theorem 7.12, we are assured, at some point, of being able to extract a suite of three tapes compliant with the hypotheses of Theorem 7.15 (at least one repetition of each repeater), and hence, to decide the bouncer.

There are multiple strategies one can implement for enumerating record-breaking tape triples that are *plausible*, i.e. that grow linearly in quadratic time. Here we propose a simple approach consisting of enumerating record-breaking tapes with same head in increasing length and for each, looking in the previous record-breaking tapes for 3 other tapes (so, 4 tapes total) in linear length progression and quadratic time progression. We then run Algorithm 7 on the last 3 tapes of these 4 tapes and, that way, the decider should find relatively early a triple where the first tape contains at least one of each repeater, as required by Theorem 7.15.

This strategy is implemented in Algorithm 8, where we assume that we are given: (1) a **get_record breaking_tapes** routine which returns the record-breaking tapes for each tape head $h \in \Delta$ (i.e. tape head state and pointing-direction), in increasing length (2) a **binary_search_len** routine which finds by binary search a tape of a given length, or returns **None** if it does not exist and (3) a **is_quadratic** which tests that a sequence of integers is in quadratic progression (for instance by computing second differences and testing they are constant), (4) routines **headless** and **attach_head** to manipulate tapes/formula tapes with and without head, and (5) a **reaches_special_case** routine on formula tapes which returns **true** if successive simulation and alignment of the formula tape, as described in Theorem 7.9, reaches a special case in a given amount of macro steps, where a macro step consists in either performing a usual step or a shift rule step.

Theorem 7.17 (Deciding bouncers)**.** Let M be a bouncer Turing machine in the sense of Theorem 7.9. Then, there exists a step limit $s \in \mathbb{N}$, a macro step limit $m \in \mathbb{N}$ and a formula tape testing limit l such that Algorithm 8, DECIDER-BOUNCERS(M,s,m,l) outputs **true**.

Proof. Algorithm 8 enumerates all record-breaking tapes triple that grow linearly in quadratic time. If s is large enough, by Theorem 7.12, it will, given a large enough formula tape testing limit l, eventually meet a triple of the form:

$$
\begin{aligned}
t_0 &= w_1\ r_1\ \ldots w_m\ r_m\ w_{m+1} \\
t_1 &= w_1\ r_1 r_1\ \ldots w_m\ r_m r_m\ w_{m+1} \\
t_2 &= w_1\ r_1 r_1 r_1 \ldots w_m\ r_m r_m r_m\ w_{m+1}
\end{aligned}
$$

Such that tapes of this triple are instances of a formula tape f with headless form $\mathcal{S}(f) = w_1(r_1)w_2(r_2)\ldots w_m(r_m)w_{m+1}$ that solves the bouncer according to Theorem 7.9. Algorithm 8, line 21, will call Algorithm 7 on this triple which will, by Theorem 7.15, return $\mathcal{S}(\mathcal{A}(\tilde{f}))$ with $\tilde{f} = w_1r_1(r_1)w_2r_2(r_2)\ldots w_mr_m(r_m)w_{m+1}$, which is a special case of f, hence that will also solve the bouncer when calling **reaches_special_case** in Algorithm 8, line 25, given that we give a large enough macro step limit m. □

7.3 Implementations and results

Here are the implementations of the decider that were realised:

1. Tony Guilfoyle's C++ initial implementation (does not use the theory presented in this section): `https://github.com/TonyGuil/bbchallenge/tree/main/Bouncers`
2. Iijil's Go implementation (does not use the theory presented in this section): `https://github.com/Iijil1/Bouncers`
3. savask's Haskell implementation (basis of the theory presented in this section): `https://gist.github.com/savask/888aa5e058559c972413790c29d7ad72`
4. mei's optimised Rust implementation, reproducing savask's: `https://github.com/meithecatte/busycoq/`. This implementation outputs certificates that are verified using Coq, more details about this approach will be given in future versions of this text.
5. Tristan Stérin (cosmo)'s Rust implementation, reproducing savask's and mei's: `https://github.com/bbchallenge/bbchallenge-deciders/tree/main/decider-bouncers-reproduction`. This implementation follows this text to the letter, reproducing each concept and algorithm as presented here. It is less efficient than mei's.

Algorithm 8 DECIDER-BOUNCERS

```
1: procedure bool DECIDER-BOUNCERS(TM machine, uint step_limit, uint macro_step_limit, uint
   max_formula_tapes)
2:     Map[TMHead,Vec[Tape]] record_breaking_tapes = get_record_breaking_tapes(machine,
   step_limit)
3:     uint num_tested_formula = 0
4:     for TMHead head in record_breaking_tapes do
5:         for uint i, Tape tape4 in record_breaking_tapes[head].enumerate() do
6:             if i < 3 then continue
7:             for uint j, Tape tape3 in record_breaking_tapes[head][:i].enumerate() do
8:                 if j < 2 then continue
9:                 uint len_diff = tape4.len() - tape3.len()
10:                uint tape2_len = tape3.len() - len_diff
11:                Tape or None tape_2 = record_breaking_tapes[head][:i].binary_search_len(tape2_len)
12:                if tape_2 is None then
13:                    continue
14:
15:                uint tape1_len = tape2.len() - len_diff
16:                Tape or None tape_1 = record_breaking_tapes[head][:i].binary_search_len(tape1_len)
17:                if tape_1 is None then
18:                    continue
19:
20:                if not is_quadratic([tape_1.step,tape_2.step,tape_3.step,tape_4.step]) then
21:                    continue
22:
23:                try FormulaTape f = FIT-FORMULATAPE(tape_2, tape_3, tape_4).headless())
24:                if Failure was raised then
25:                    continue
26:
27:                f.attach_head(tape_1.head)
28:                if f.reaches_special_case(macro_step_limit) then
29:                    return true
30:
31:                num_tested_formula += 1
32:
33:                if num_tested_formula == max_formula_tapes then
34:                    return false
35:    return false
```

Verifiers. Verifiers for Theorem 7.9 – i.e. programs that check that a given formula tape gives a valid nonhalting proof for a given bouncer – have also been given as part of the above implementations. Mei's implementation provides a Coq implementation of the verifier. There is an ongoing effort for standardising bouncers certificates (and certificates in general) but the following are enough for practical verifiation: (1) formula tape (2) time step at which a tape of the formula tape is reached (2) number of macro steps needed to reach special case. Other useful information can be added such as the set of shift rules used by the formula tape simulation.

Results. Using limits of 250,000 steps and 50,000 macro steps and 10 formula tapes, the method decides 29,799 machines out of the 32,632 remaining machines (91%) after FAR (Section 6), in less than 3 seconds using mei's implementation[19] and in approximatively 1 minute using cosmo's on a standard laptop. Hence, after bouncers, we have 29,799 machines left to be decided.

Remarkable bouncers. Out of the 29,799 bouncers that were decided using the method presented in this section, here are some remarkable facts:

1. Only two bouncers have 3 repeaters and that's the maximum: `https://bbchallenge.org/347505`

[19] Only step limit is used in this implementation, macro step limit is deduced from step limit and there is no formula tapes limit.

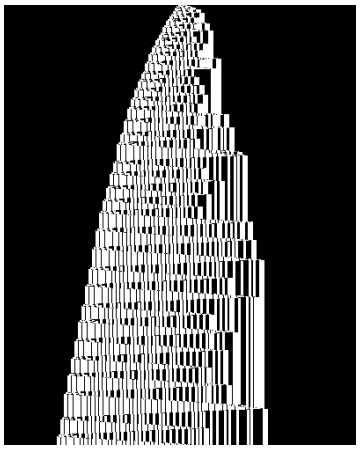

Figure 9: 50,000-step space-time diagram of `https://bbchallenge.org/5608043`. Out of the 29,799 decided bouncers, this bouncer takes the most steps (141,509) to be detected and fits the biggest formula tape (378 0/1 symbols), using the method presented in this section.

and `https://bbchallenge.org/8131743`. Otherwise, 2132 bouncers have 2 repeaters and the rest has only 1.

2. The biggest fitted formula tapes by the algorithm (but note that formulas are not necessarily minimal, see Remark 7.16) have 378 symbols (summing walls and repeater symbols), there are two of them, such as for machine `https://bbchallenge.org/5608043`, see Figure 9:

   $0^\infty \overset{A}{\triangleleft} 1000001110001110000001110000111011100111011100111000011100001110111001110000111$
   $0011100111011100111000011100001110111001110000111000011100001110111001110111001110$
   $1110111001110111001110111001110111001110111001110111001110111001110111001110111001$
   $1011100111011100111011100111011100111011100111011100111011100111011100(111011100111011$
   $1001110111001110111001)0001110000111000000111001111111111111111000011111111111111111(11111$
   $1111111)0011100000001110000000111111001111110^\infty$

3. The above machine of Point 2 is also the machine that is detected after the most steps: 141,509. Over this dataset, it took 202 steps on average.

4. The most macro steps (i.e. number of usual or shift rule steps in formula tape simulation) needed to conclude using Theorem 7.9 was 44,898 for `https://bbchallenge.org/347505`. Otherwise, it took 107 macro steps on average.

References

[1] S. Aaronson. The Busy Beaver Frontier. *SIGACT News*, 51(3):32–54, Sept. 2020. `https://www.scottaaronson.com/papers/bb.pdf`.

[2] A. H. Brady. The determination of the value of rado's noncomputable function $|sum(k)$ for four-state turing machines. *Mathematics of Computation*, 40(162):647–665, 1983.

[3] R. Cuninghame-Green. Minimax algebra and applications. *Fuzzy Sets and Systems*, 41(3):251–267, 1991.

[4] Iijil. Bruteforce-ctl. `https://github.com/Iijil1/Bruteforce-CTL`, 2022.

[5] S. Ligocki. CTL Filter. Blog: `https://www.sligocki.com/2022/06/10/ctl.html`. Accessed: 2023-03-20.

[6] H. Marxen and J. Buntrock. Attacking the Busy Beaver 5. *Bull. EATCS*, 40:247–251, 1990.

[7] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.