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), Justin Blanchard, Pascal Michel.

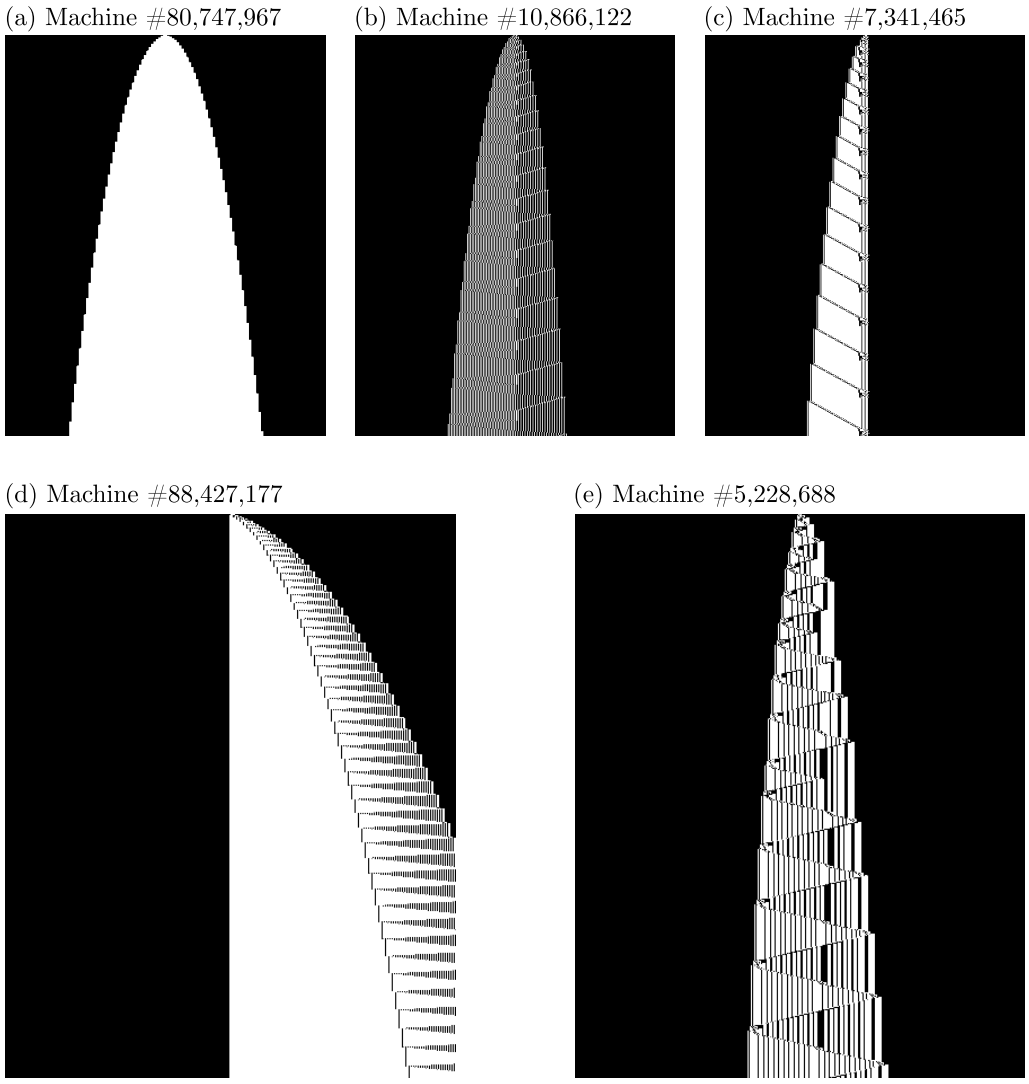

Figure 8: Space-time diagrams (10,000 steps) of several `bbchallenge` *bouncers*: (a) simple bouncer bouncing back and forth between expanding tape extremeties while writing 1s (b) bouncer with more complex alternating *repeater* patterns left and right of the origin (c) unilateral bouncer with a complex *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)-

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

(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].

The goal of this section is to formally characterise bouncers and show that they do not halt, see Theorem 7.9. Then, in Section 7.3, we show how do detect bouncers algorithmically in practice, see Algorithm 8.

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 machines 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 Σ the tape alphabet of the machine – in the context of BB(5), we have $\Sigma = \{0, 1\}$. For each Turing machine with set of states 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 $\Delta = \{\overset{s}{\triangleleft} | s \in S\} \cup \{\overset{s}{\triangleright} | s \in S\}$.

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. 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; 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. Applying $\vdash$ successively $n \in \mathbb{N}$ times is written $\vdash^n$. The transitive closure of $\vdash$ (i.e. applying $\vdash$ one or more times) is written $\vdash^+$; applying $\vdash$ zero or more times is written $\vdash^*$.

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:

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

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

In this section, we formalise the above-stated intuition that bouncers expand a tape by repeating *repeater* patterns between fixed *walls*. Then, we formally define bouncers in Definition 7.8 and show that they do not halt in Theorem 7.9. For this, we are going to express bouncers' tapes using abbreviated regular expressions over the alphabet $\Delta \cup \overline{\Sigma}$. Given $u \in \Sigma^*$, we represent the regular language $\{u\}^*$ (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 an 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). Words 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 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 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*, that is, rewrite rules for formula tapes, which rewrite a subword $u \overset{s}{\triangleright} (r)$ into $(\tilde{r})u \overset{s}{\triangleright}$, denoted by $u \overset{s}{\triangleright} (r) \rightarrow (\tilde{r})u \overset{s}{\triangleright}$. Similarly, left shift rules are of the form $(r) \overset{s}{\triangleleft} u \rightarrow \overset{s}{\triangleleft} u(\tilde{r})$ 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.

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 at nonempty w_{n+1} or w'_1, and $w_{n+1}hw'_1 \vdash \tilde{w}_{n+1}\tilde{h}\tilde{w}'_1$ as tapes, replace the terms $w_{n+1}hw'_1(r'_1)$ of f with $\tilde{w}_{n+1}\tilde{h}\tilde{w}'_1$. Call the new formula f', and define $f \vdash f'$. 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 $h = \overset{s}{\triangleright}$ with $s \in S$ 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 define $f \vdash f'$. If $\mathcal{R}$ is empty, $\vdash$ cannot be applied to f.

   (b) Left shift rule. If $h = \overset{s}{\triangleleft}$ with $s \in S$ 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 $t \vdash^* t'$ and, in case $f \vdash f'$ via usual step, $t \vdash^+ t'$.

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 (of the form $u \overset{s}{\triangleright} (r) \vdash^k (\tilde{r})u \overset{s}{\triangleright}$ or $(r) \overset{s}{\triangleleft} u \vdash^k \overset{s}{\triangleleft} u(\tilde{r})$), and the corresponding repeater (r) is used n times in the match $t\ \in \mathcal{L}(f)$, we apply nk steps to t to obtain $t' \in \mathcal{L}(f')$. This amounts to a positive number of steps (justifying $\vdash^+$) except in case of a shift rule applied $k = 0$ times. □

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 at 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 $n \in \mathbb{N}$, ${r'_j}^n v = vr^n$, 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 $f = 0^\infty(111)1111 \overset{\mathrm{B}}{\triangleright} (01)01010110^\infty$ 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 $0^\infty(111)111111011(11)0 \overset{\mathrm{D}}{\triangleright} 10^\infty$. Another alignment example is $f = 0^\infty101(11)0 \overset{\mathrm{D}}{\triangleright} 10(100)110^\infty$ for which $\mathcal{A}(f) = 0^\infty10(11)10 \overset{\mathrm{D}}{\triangleright} 101(001)10^\infty$.

The above example 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 found. This motivates the introduction of the notation $f \vdash_{\mathcal{A}} f'$ which means that we have $\mathcal{A}(f) \vdash f'$.

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 replacing subwords of the form (r) in $\mathcal{A}(f)$ 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) in $\mathcal{A}(f)$ is to the left (resp. to the right) of the head. If f' is a special case of f then $\mathcal{L}(f') \subseteq \mathcal{L}(f)$, and the authors conjecture that the converse is true under some mild additional assumptions[18].

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:

Definition 7.8 (Bouncers)**.** A bouncer is a Turing machine such that there exists a wall-repeater formula tape f which satisfies:

1. There is a reachable tape $t \in \mathcal{L}(f)$, i.e. $0^\infty \overset{\mathrm{A}}{\triangleright} 0^\infty \vdash^* t$
2. There is a formula tape f' such that $f \vdash_{\mathcal{A}}^+ f'$ and f' is a special case of f

We say that formula tape f *solves* the bouncer.

Theorem 7.9. A bouncer does not halt.

Proof. Unraveling $f \vdash_{\mathcal{A}}^+ f'$ gives $n \in \mathbb{N}$ and $f_1, \ldots, f_n$ with $f_n = f'$ and $\mathcal{A}(f_i) \vdash f_{i+1}$ for $0 \leq i < n$. It follows from Lemma 7.3 and Lemma 7.6 that there exist 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 $t_0 \vdash^* t_n$. Moreover, $t_0 \vdash^+ t_n$ if the any of the $\mathcal{A}(f_i) \vdash f_{i+1}$ relations are via usual steps.

Indeed, this must be the case. Suppose instead we had a sequence of shift rules. Each one would preserve the direction of the head, and increment the number of repeaters behind the head in the formula tape. However, these properties are unchanged by passing from a formula tape f to $\mathcal{A}(f)$ or a special case of f. It would follow that f_1 has strictly more repeaters behind the head than f_1, a contradiction.

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 $i \in \mathbb{N}$ hence the machine does not halt. □

Definition 7.10 (Bouncer certificate)**.** For a given bouncer, a bouncer certificate consists at least of the following:

1. The formula tape f of Definition 7.8 that solves the bouncer.
2. The time step at which tape t such that $t \in \mathcal{L}(f_0)$ is reached from $0^\infty \overset{\mathrm{A}}{\triangleright} 0^\infty$.
3. The number of *macro steps* n such that $f \vdash_{\mathcal{A}}^n f'$ with f' special case of f, where a macro step on a formula tape consists in applying $\vdash_{\mathcal{A}}$, i.e. alignment followed by $\vdash$ (one usual step or one shift rule step).

Given such certificate, one can verify that the machine is a bouncer by applying Theorem 7.9 or that the certificate is erroneous. Additional information, such has the list of used shift rules can be added to the certificate for convenience.

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

Example 7.11. The machine of Example 7.1 (Figure 8 (d)) that is used in our series of examples for this section, is a bouncer. Indeed, we have $0^\infty \overset{\mathrm{A}}{\triangleright} 0^\infty \vdash^{64} 0^\infty 11111101100 \overset{\mathrm{D}}{\triangleright} 0^\infty$. The tape $t = 0^\infty \vdash^{64} 0^\infty 11111101100 \overset{\mathrm{D}}{\triangleright} 0^\infty$ is in the language the following formula tape:

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

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

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

One shift rule gives:

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

After alignment:

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

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

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

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.

A bouncer certificate (Definition 7.10) for this machine is $f_0 = 0^\infty(111)1110(11)00 \overset{\mathrm{D}}{\triangleright} 0^\infty$, time step 64 at which $0^\infty 11111101100 \overset{\mathrm{D}}{\triangleright} 0^\infty \in \mathcal{L}(f_0)$ is reached and 41 macro steps which transform f_0 into special case f_{41} under successive $\vdash$ and alignement applications.

Note that 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

For a given formula tape, using the notation of Equation 7.1:

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

call $C_f(k) \in \mathcal{L}(f)$ the tape where each repeater is used exactly $k \in \mathbb{N}$ times:

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

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 $D : \mathbb{Z}^\mathbb{N} \to \mathbb{Z}^\mathbb{N}$ via $D((u_n)_{n\in\mathbb{N}}) = (u_{n+1} - u_n)_{n\in\mathbb{N}}$. The *length* of a tape t is the number of symbols of Σ in t, i.e. ignoring head and 0^∞.

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 solved by formula tape f that is not a cycler (Section 2),then, we can construct from f a new formula tape $\mathrm{sync}(f)$, that also solves the bouncer but such that tapes $C_{\mathrm{sync}(f)}(k)$, where all repeaters are synchronously repeated k times, are reached by the machine for all successive $k \in \mathbb{N}$. Importantly for being detected in practice, the length of tapes $C_{\mathrm{sync}(f)}(k)$ grows linearly in quadratic time:

Theorem 7.12 (Linear-quadratic growth)**.** Let M be a bouncer that is not a cycler, solved by some formula tape f. 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. Then, there is a formula tape called $\mathrm{sync}(f)$, obtained from f, that also solves the bouncer such that there is an extraction function $g : \mathbb{N} \to \mathbb{N}$ with $g(n) \geq n$ for all $n \in \mathbb{N}$, satisfying:

1. For all $n \in \mathbb{N}$, $t_{g(n)} = C_{\mathrm{sync}(f)}(n)$.
2. $(l_{g(n)})_{n\in\mathbb{N}}$ is an arithmetic progression, i.e. $D((l_{g(n)})_{n\in\mathbb{N}}) = (K)_{n\in\mathbb{N}}$ for some constant $K \in \mathbb{N}$.
3. $(g(n))_{n\in\mathbb{N}}$ 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. For simplicity of notations, we suppose that there are no repeaters after the tape head in f, as we would apply exactly the same transformations after the head. Using Equation (7.1), we write $f = w_1(r_1)w_2(r_2)\dots w_n(r_n)w_{n+1}hw$ with $n \geq 1$, $w_i \in \Sigma^*$ for $2 \leq i \leq n+1$, $w_1, w \in \overline{\Sigma}^*$ and $r_i \in \Sigma^+$ for $1 \leq i \leq n$ and $h \in \Delta$. Without loss of generality we suppose that f is aligned (otherwise we just apply $\mathcal{A}$ to it).

Because f solves the bouncer, by Definition 7.8, the machine reaches tape t of the form $t = w_1 r_1^{k_1} w_2 r_2^{k_2} \dots r_n^{k_n} w_{n+1} hw$ with $k_1, \dots, k_n \in \mathbb{N}$. By Theorem 7.9, we have $t \vdash^+ t'$ with $t' \in \mathcal{L}(f')$ and $t' = w_1 r_1^{k_1+p_1} w_2 r_2^{k_2+p_2} \dots r_n^{k_n+p_n} w_{n+1} hw$ with $p_1, \dots, p_n \in \mathbb{N}$ and same w_i and r_i as f because f is aligned. Note that p_i is positive because shift rules preserve the size of repeaters. Because $t' \in \mathcal{L}(f') \subseteq \mathcal{L}(f)$ we can repeat this indefinitely and get $t \vdash^+ w_1 r_1^{k_1+Np_1} w_2 r_2^{k_2+Np_2} \dots r_n^{k_n+Np_n} w_{n+1} hw$ for all $N \in \mathbb{N}$.

Hence, define $\mathrm{sync}(f) = w_1 r_1^{k_1}(r^{p_1}) w_2 r_2^{k_2}(r^{p_2}) \dots w_n r_n^{k_n}(r^{p_n}) w_{n+1} hw$. When $p_i = 0$, then $r^{p_i} = \varnothing$ and we discard it. All p_i cannot be 0, otherwise the machine is a cycler which is excluded. Altogether, we can write: $\mathrm{sync}(f) = \tilde{w}_1(\tilde{r}_1)\tilde{w}_2(\tilde{r}_2)\dots\tilde{w}_m(\tilde{r}_m)\tilde{w}_{m+1}hw$ with $2 \leq m \leq n$ and nonempty $\tilde{r}_i$, which is a valid formula tape which also solves the bouncer. By construction, the machine will reach tapes of the form $C_{\tilde{f}}(n)$ for all $n \in \mathbb{N}$ at increasing time steps. We immediately get Point 2 because $C_{\tilde{f}}(n+1)$ contains $K = \sum_{i=1}^{n} |\tilde{r}_i|$ more symbols than $C_{\tilde{f}}(n)$, which is a constant.

Concerning Point 3, call $g(n)$ the time step at which $C_{\tilde{f}}(n)$ is reached. By construction, $g(n) \geq n$. The number of macro steps $M \geq 1$ such that we get $\mathrm{sync}(f) \vdash_{\mathcal{A}}^{M} f''$ with f'' special case of $\mathrm{sync}(f)$ is a constant. We write $M = M_u + M_s$ with M_u the number of usual steps and M_s the number of shift rule applications. Call M'_s the number of Turing machine steps taken in total to perform the base case (i.e. the steps needed to prove that a shift rule is correct) of each of the M_s shift rules. When processing $C_{\tilde{f}}(n)$, each shift rule is applied once each to the n repetitions of each repeater (because shift rules preserve the number of repetitions), hence we get: $g(n+1) - g(n) = M_u + nM'_s$, which means $D(D((g(n))_{n\in\mathbb{N}})) = (M'_s)_{n\in\mathbb{N}}$ which is constant, as needed.

□

Example 7.13. Using Example 7.11, we get that $f = 0^\infty(111)1110(11)00 \overset{\mathrm{D}}{\triangleright} 0^\infty$ solves the bouncer given in Example 7.1. Using Theorem 7.12, we have $\mathrm{sync}(f) = 0^\infty(111)1111110(11)1100 \overset{\mathrm{D}}{\triangleright} 0^\infty$. We have $\mathrm{sync}(f) \vdash_{\mathcal{A}}^{M} f'$ with $M = 47$ and f' special case of $\mathrm{sync}(f)$. Note that $47 > 41$ found in Example 7.11 because $\mathrm{sync}(f)$ is a bit bigger than f.

We can decompose $M = M_u + M_s$ with $M_u = 45$ usual steps and $M_s = 2$ shift rule applications. The shift rules that are used are $0 \overset{\mathrm{D}}{\triangleright} (01) \to (11)0 \overset{\mathrm{D}}{\triangleright}$ which takes 4 Turing machine steps to execute (Example 7.4) and $(11) \overset{\mathrm{A}}{\triangleleft} \to \overset{\mathrm{A}}{\triangleleft} (01)$ which takes 2, hence $M'_s = 6$.

Calling $g(n)$ the time step at which $C_{\mathrm{sync}(f)}(n)$ is reached, we have $g(0) = 64$. Using the proof of Theorem 7.12, $g(n+1) - g(n) = M_u + nM'_s = 45 + 6n$, hence, $g(n) = 3n^2 + 42n + 64$. We also have $l_{g(n)} = 11 + 5n$. One can check by simulation that tapes $C_{\mathrm{sync}(f)}(n)$ are reached at time steps $g(n)$.

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 in practice.

7.2.1 Formula tape fitting

We introduce $\mathcal{A}_r$, the *right-alignment* operator on formula tapes which only applies rule of type 1 in Definition 7.5 to all repeaters, both before and after the formula's head, bubbling them to the right.

In this section, we give a greedy algorithm (Algorithm 7) that fits a formula tape f only given three tapes: $C_f(0)$, $C_f(1)$ and $C_f(2)$. The algorithm is guaranteed to work on right-aligned formula tapes with nonempty *intermediary walls*, which are all walls but the first and last one, see Theorem 7.18. First, we must show that the precondition of having nonempty intermediary walls is without loss of generality, i.e. that all bouncers can be solved using such formulas. We prove this result in Lemma 7.16, using two technical lemmas, Lemmas 7.14 and 7.15.

Lemma 7.14. Assume that M is a bouncer solved by some formula tape f, then:

1. $\mathcal{A}_r(f)$ also solves the bouncer.
2. Any special case f' of f also solves the bouncer.

Proof. 1. We have $\mathcal{A}(\mathcal{A}_r(f)) = \mathcal{A}(f)$ hence, Theorem 7.9 will reach $\mathcal{A}(f)$ after one application of $\mathcal{A}$ in any case and the bouncer will get solved in the same way.

2. By definition of special case f' can be constructed from f by replacing any (r) by $r^n(r)r^m$ for some $n, m \in \mathbb{N}$. Aligning f' will have the same effect as aligning f with the addition that repeaters before the head will look like $(r)r^{n+m}$ and after the head $r^{n+m}(r)$. Then, shift rule will apply similarly to f' and f, with the addition in f' that segments of the form r^{n+m} will be handled by simulating the same shift rule as for (r) through usual steps. Hence, f' solves the bouncer.

□

Lemma 7.15. Let $a, b \in \Sigma^+$ be two nonempty words such that we have $a[i \bmod |a|] = b^\infty[i]$ for all $i \in \mathbb{N}$, with $b^\infty[i]$ being the character at position i in the infinite concatenation of word b with itself. Call $k = \gcd(|a|, |b|)$. Then, a and b are powers of the same word c such that $a = c^m$ and $b = c^n$, with c the first k characters of a and b, and $m = |a|/k \geq 1$ and $n = |b|/k \geq 1$.

Proof. Note that $|a| \neq 0$ and $|b| \neq 0$ by hypothesis. Call $f(i) = b^\infty[i]$. It is immediate that $|b|$ is a period of f. By hypothesis, we also have that $|a|$ is a period of f. Hence, any linear combination that has positive value of $|a|$ and $|b|$ is also a period of f. Hence, by Euclid's algorithm, $k = \gcd(|a|, |b|)$ is a period of f, from which we get $b = c^{|b|/k}$ with c the first k characters of b since $k \leq |b|$. By hypothesis, we know that $a^\infty = b^\infty$, hence, we have $a = c^{|a|/k}$, as needed.

□

Lemma 7.16. Let a machine M be a bouncer solved by formula tape f, then, f can be transformed into a formula tape f' such that $\mathcal{A}_r(f')$ has nonempty intermediary walls that solves the bouncer.

Proof. By Lemma 7.14, $\mathcal{A}_r(f)$ also solves the bouncer. For each case $\dots(r_1)(r_2)\dots$ where two repeaters in $\mathcal{A}_r(f)$ are separated by an empty wall, apply the following transformation to $\mathcal{A}_r(f)$, according to two cases:

1. There is $k \geq 1$ such that right-aligning $(r_1)r_2^k(r_2)$ yields a non-empty intermediary wall, replace $(r_1)(r_2)$ by $\mathcal{A}_r((r_1)r_2^k(r_2))$, yielding a new formula tape f'. Noticing that $(r_1)r_2^k(r_2)$ is a special case of $(r_1)(r_2)$ and combining both points of Lemma 7.14 we get that f' still solves the bouncer.
2. For all $k \geq 1$ there's $r'_k \in \Sigma^+$ such that we have $\mathcal{A}_r((r_1)r_2^k(r_2)) = r_2^k(r'_k)(r_2)$. We deduce that $r_1[i \bmod |r_1|] = r_2^\infty[i]$ for all $i \in \mathbb{N}$. By Lemma 7.15 we get that there is c such that $r_1 = c^m$ and $r_2 = c^n$. Hence, we can replace $(r_1)(r_2) = (c^m)(c^n)$ by $(c^m c^n) = (r_1 r_2)$ in the formula tape and still solve the bouncer with this new formula tape. Indeed, because f solves the bouncer we have $u \in \Sigma^*$, $s \in S$, $r', r'' \in \Sigma^+$ and two shift rules, for instance, right shift rules: $u \overset{s}{\triangleright} (c^m) \to (r')u \overset{s}{\triangleright}$ and $u \overset{s}{\triangleright} (c^n) \to (r'')u \overset{s}{\triangleright}$. Hence, considering $u \overset{s}{\triangleright} c^m c^n$, we get $u \overset{s}{\triangleright} c^m c^n \vdash^* r'u \overset{s}{\triangleright} c^n \vdash^+ r'r''u \overset{s}{\triangleright}$, meaning that we have a right shift rule $u \overset{s}{\triangleright} (c^m c^n) \to (r'r'')u \overset{s}{\triangleright}$, as needed.

By applying case (1) or (2) to all repeaters of $\mathcal{A}_r(f)$ that are separated by an empty wall, we get f', a transformation of f with nonempty intermediary walls that solves the bouncer. Because f' is right-aligned by construction, $\mathcal{A}_r(f') = f'$ and we get the result.

□

We are now ready to infer such a formula tape f from three examples $t_i = C_f(i)$. To simplify the procedure, we wish to drop the head and 0^∞ symbols from the tapes and re-attach them to the matching formula.

Given a tape t, we call $\mathcal{S}(t) \in \Sigma^*$ the *headless* version of t which is the tape without head and 0^∞ symbols. We also introduce headless formula tapes: $\mathcal{S}(f)$ is f without head and 0^∞ symbols.

Algorithm 7 is a greedy algorithm which fits a headless formula tape given three headless tapes t_0, t_1, t_2. It proceeds by recursively constructing the left-most wall using the longest 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, we have:

Algorithm 7 Greedy formula tape fitting algorithm FIT-FORMULATAPE

```
1:  procedure HeadlessFormulaTape FIT-FORMULATAPE(Word t0, Word t1, Word t2)
2:      if t0.empty() && t1.empty() && t2.empty() then
3:          return HeadlessFormulaTape::empty()
4:
5:      if t0.len() > 0 && t1.len() > 0 && t2.len() > 0 && t0[0] == t1[0] && t1[0] == t2[0] then
6:          return HeadlessFormulaTape::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 HeadlessFormulaTape::repeater(t2[:l]).concat(FIT-FORMULATAPE(t0, t1[l:],
    t2[2*l:]))
13:
14:     raise Failure
```

Remark 7.17 (Implementation details of Algorithm 7)**.** We assume that we are given a **HeadlessFormulaTape** construct, 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 7.18 (Greedy formula tape fitting)**.** Let f be a headless formula tape with nonempty intermediary walls such that $\mathcal{A}_r(f) = w_1(r_1)w_2(r_2)\dots w_m(r_m)w$ with $m \in \mathbb{N}$, $r_i, w_i \in \Sigma^+$ with $1 \leq i \leq m$, and $w_1, w \in \overline{\Sigma}^*$. Take $t_0, t_1, t_2 \in \Sigma^+$ satisfying:

$$\begin{aligned} t_0 &= \mathcal{S}(C_f(0)) = w_1\ w_2 \dots w_m\ w \\ t_1 &= \mathcal{S}(C_f(1)) = w_1\ r_1\ w_2\ \dots w_m\ r_m\ w \\ t_2 &= \mathcal{S}(C_f(2)) = w_1\ r_1 r_1\ w_2\ \dots w_m\ r_m r_m\ w \end{aligned}$$

Then Algorithm 7 ran on FIT-FORMULATAPE(t_0, t_1, t_2) does not raise failure and returns $\mathcal{A}_r(f)$.

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

$$\begin{aligned} t_0 &= w_1\ w \\ t_1 &= w_1\ r\ w \\ t_2 &= w_1\ rr\ w \\ f_{\text{out}} &= \varnothing \end{aligned} \quad \rightarrow_{\text{Algorithm 7 goes to}} \quad \begin{aligned} t_0 &= w \\ t_1 &= r\ w \\ t_2 &= rr\ w \\ f_{\text{out}} &= w_1 \end{aligned}$$

From there, we have two cases: (1) if $w = \varnothing$ then the greedy algorithm returns $f_{\text{out}} = w_1(r) = \mathcal{A}_r(f)$, as needed, (2) if $w \neq \varnothing$, because $\mathcal{A}_r(f)$ is right-aligned, we must have $w[0] \neq r[0]$ (r is not empty by hypothesis), and then the greedy algorithm returns $f_{\text{out}} = w_1(r)w = \mathcal{A}_r(f)$, as needed.

Assuming $m \geq 2$, we have:

$$\begin{aligned} t_0 &= w_1\ w_2\ \ldots w_m\ w \\ t_1 &= w_1\ r_1\ w_2\ r_2\ \ldots w_m\ r_m\ w \\ t_2 &= w_1\ r_1 r_1\ w_2\ r_2 r_2\ \ldots w_m\ r_m r_m\ w \\ f_{\text{out}} &= \varnothing \end{aligned} \quad \rightarrow_{\text{Algorithm 7 goes to}} \quad \begin{aligned} t_0 &= w_2\ \ldots w_m\ w \\ t_1 &= r_1\ w_2\ r_2\ \ldots w_m\ r_m\ w \\ t_2 &= r_1 r_1\ w_2\ r_2 r_2\ \ldots w_m\ r_m r_m\ w \\ f_{\text{out}} &= w_1 \end{aligned}$$

By hypothesis, w_2 and r_1 are not empty and, because $\mathcal{A}_r(f)$ is right-aligned, we get $w_2[0] \neq r_1[0]$. Hence, Algorithm 7 enters the repeater-fitting case on line 8 and goes to:

$$\begin{aligned} t_0 &= w_2\ \ldots w_m\ w \\ t_1 &= w_2\ r_2\ \ldots w_m\ r_m\ w \\ t_2 &= w_2\ r_2 r_2\ \ldots w_m\ r_m r_m\ w \\ f_{\text{out}} &= w_1(r_1) \end{aligned}$$

From here we can inductively conclude that Algorithm 7 will reach:

$$\begin{aligned} t_0 &= w_m\ w \\ t_1 &= w_m\ r_m\ w \\ t_2 &= w_m\ r_m r_m\ w \\ f_{\text{out}} &= w_1(r_1) \ldots w_{m-1}(r_{m-1}) \end{aligned}$$

Using the same argument as for $m = 1$, we get that Algorithm 7 returns $f_{\text{out}} = w_1(r_1) \ldots w_m(r_m)w = \mathcal{A}_r(f)$, as needed.

□

Example 7.19. Consider the headless right-aligned formula tape with nonempty intermediary walls $f = 100(100)0000(0)$. Then the Algorithm 7 proceeds as follows:

$$\begin{aligned} t_0 &= 100\ 0000 \\ t_1 &= 100\ 100\ 0000\ 0 \\ t_2 &= 100\ 100\ 100\ 0000\ 0\ 0 \\ f_{\text{out}} &= \varnothing \end{aligned} \quad \rightarrow_{\text{Algorithm 7 goes to}} \quad \begin{aligned} t_0 &= 0000 \\ t_1 &= 100\ 0000\ 0 \\ t_2 &= 100\ 100\ 0000\ 0\ 0 \\ f_{\text{out}} &= 100 \end{aligned} \quad \rightarrow_{\text{Algorithm 7 goes to}} \quad \begin{aligned} t_0 &= 0000 \\ t_1 &= 0000\ 0 \\ t_2 &= 0000\ 0\ 0 \\ f_{\text{out}} &= 100(100) \end{aligned}$$

$$\begin{aligned} t_0 &= 0000 \\ t_1 &= 0000\ 0 \\ t_2 &= 0000\ 0\ 0 \\ f &= 100(100) \end{aligned} \quad \rightarrow_{\text{Algorithm 7 goes to}} \quad \begin{aligned} t_0 &= \varnothing \\ t_1 &= 0 \\ t_2 &= 0\ 0 \\ f_{\text{out}} &= 100(100)0000 \end{aligned} \quad \rightarrow_{\text{Algorithm 7 goes to}} \quad \begin{aligned} t_0 &= \varnothing \\ t_1 &= \varnothing \\ t_2 &= \varnothing \\ f_{\text{out}} &= 100(100)0000(0) \end{aligned}$$

Hence, the output is f, as claimed in Theorem 7.18.

Note that in some case, Algorithm 7 is able to fit formula tapes that have some empty intermediary walls, such as $f = 10110(110)(101)$. But in other cases, it cannot, such as with $f = 01(1)(0111)1$, which raises failure.

7.2.2 Bouncers decider

We finally piece all the elements of this text together and describe a decider for bouncers (excluding cyclers, decided in Section 2), Algorithm 8 which is proven correct in Theorem 7.22.

In order to drastically limit the amount of plausible subsequences to check, we limit our interest to *record-breaking* formula tapes, which are formula tapes where the head is pointing at 0^∞. In order to fit them, we can simply track record-breaking tapes that share the same head, i.e. tapes where the head is pointing at 0^∞. We first show that this is without loss of generality:

Lemma 7.20. If M is a bouncer that is not a cycler, solved by a formula tape f, then there is a record-breaking formula tape that also solves it.

Proof. Let M be a bouncer that is not a cycler and f a formula tape that solves it. Then we have $f \vdash_{\mathcal{A}} f_1 \vdash_{\mathcal{A}} f_2 \cdots \vdash_{\mathcal{A}} f_n$ with $n \geq 1$ and f_n is a special case of f. There is $1 \leq i \leq n$ such that f_i is record-breaking otherwise, the formula tapes do not grow and M is a cycler, which is excluded. Because f_n is a special case of f, it will eventually reach, under applications of $\vdash_{\mathcal{A}}$, f' that is special case of f_i and we have the result. □

Lemma 7.21. If f is a formula tape with nonempty intermediary walls, then sync(f) built in Theorem 7.12 has nonempty intermediary walls.

Proof. The construction of sync(f) only involves (1) removing some repeaters (2) replacing some repeaters by some powers of themselves (3) increasing the size of some walls. Hence, sync(f) has nonempty intermediary walls. □

Algorithm 8 proceeds by (1) tracking record-breaking tapes with same head (i.e. same state and pointing-direction) that grow linearly in quadratic time, (2) fitting formula tapes from triples from these plausible subsequences until a formula tape solves the bouncer is found or some limits are met.

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_1, tape_2, tape_3).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
```

Theorem 7.22 (Deciding bouncers)**.** Let M be a bouncer (Definition 7.8). 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. Because M is a bouncer, using Lemma 7.20 we know that there is a record-breaking formula tape $\tilde{f}$ that solves the bouncer. By Lemma 7.16 we know that we can transform $\tilde{f}$ into f with nonempty intermediary walls and that also solves the bouncer.

Algorithm 8 enumerates all record-breaking tapes triple that grow linearly in quadratic time. If s and l are large enough, by Theorem 7.12, we know that it will, eventually meet a triple of the form:

$$t_0 = C_{\text{sync}(f)}(0)$$
$$t_1 = C_{\text{sync}(f)}f(1)$$
$$t_2 = C_{\text{sync}(f)}f(2)$$

By Lemma 7.21, $f' = \text{sync}(f)$ is with nonempty intermediary walls and, at line 23, Algorithm 8 will feed $(\mathcal{S}(C_{f'}(0)), \mathcal{S}(C_{f'}(1)), \mathcal{S}(C_{f'}(2)))$ to Algorithm 7, which, by Theorem 7.18, will output $\mathcal{S}(\mathcal{A}_r(\text{sync}(f)))$ which be reconstructed into $\mathcal{A}_r(\text{sync}(f))$ at line 27. By Theorem 7.12, we know that $\text{sync}(f)$ also solves the bouncer, and by Lemma 7.14, $\mathcal{A}_r(\text{sync}(f))$ also solves the bouncer. Theorem 7.9 will be verified on $\mathcal{A}_r(\text{sync}(f))$ using **reaches_special_case** at line 28, which, by hypothesis will output **true**, and we have the result. □

Remark 7.23. While Theorem 7.22 assures us that the decider will be able to detect a bouncer in bounded time, the formula tapes found in practice by the algorithm can be smaller than the $\text{sync}(f)$ construction we used to prove the upper bound.

Remark 7.24 (Implementation details of Algorithm 8)**.** In Algorithm 8, 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 performing alignment followed by either performing a usual step or a shift rule step as. We can note that when detecting shift rules (see Section 7.1.2), it is important to implement cycler detection which as it is possible for a machine to cycle indefinitely on a finite tape.

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's (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.

Verifiers. Verifiers for Theorem 7.9, i.e. programs that verify bouncer certificates (Definition 7.10) 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 the format of bouncers certificates (and certificates in general).

Results. Using limits of 250,000 steps and 50,000 macro steps and 20 formula tapes tested per head machine, 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.

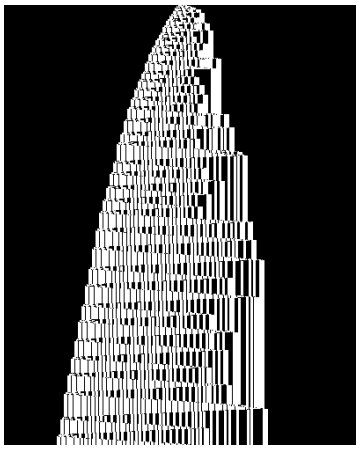

Figure 9: 50,000-step space-time diagram of `https://bbchallenge.org/5608043`. Out of the 29,799 decided bouncers, this bouncer takes the most steps (141,509) to be detected and fits the biggest formula tape, using Algorithm 8 presented in this section.

Remarkable bouncers. Out of the 29,799 bouncers that were decided using Algorithm 8, here are some remarkable facts:

1. Using Algorithm 8, only two bouncers with 3 repeaters were found (and that's the maximum): `https://bbchallenge.org/347505` 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 have 328 symbols (summing walls and repeater symbols, not counting head and 0^∞), there are two of them, such as for machine `https://bbchallenge.org/5608043`, see Figure 9:

$$0^\infty \overset{A}{\triangleleft} 1000001110001110000001110000111011100111011100111000111$$
$$0000111011100111000011101110011101110011100001110000111011100$$
$$1110000111000011100001110000111011100111011100111011100111011$$
$$1001110111001110111001(1110111001110111001110111001110111001)000$$
$$1110000111000000111001111111111111110000111(111111111111)00111$$
$$0000000111000000111111001111111(0)^\infty$$

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 207 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 41,628 for `https://bbchallenge.org/347505`. Otherwise, it took 66 macro steps on average.

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