Baba Is Universal

Zachary Abel
Massachusetts Institute of Technology, Cambridge, MA, USA

Della Hendrickson
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

We consider the computational complexity of constant-area levels of games which allow an unlimited number of objects in a fixed region. We discuss how to prove that such games are RE-hard (and in particular undecidable) and capable of universal computation, even on constant-area levels. We use the puzzle game Baba is You as a case study, showing that 8×17 levels are capable of universal computation by constructing a particular small universal counter machine within Baba is You.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Undecidability, Baba is You, RE-hardness, counter machines, universal computation

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.1

Acknowledgements This work was initiated at the 34th Bellairs Winter Workshop on Computational Geometry in March 2019 in Holetown, Barbados, which was organized by Erik Demaine and Godfried Toussaint. We thank the other participants for helpful discussion and contributing to a collaborative and productive research environment.

1 Introduction

There has recently been a surge of computational complexity results for games, especially video games [18, 1, 10, 9, 6]. Most commonly, games (technically, decision problems of the form "given a level of this game, is it possible to win?") are shown to be NP- or PSPACE-complete, and frameworks have been developed to simplify such proofs [14, 18, 1, 7, 5]. There is not as much infrastructure for other complexity classes, so proofs of e.g. EXP-hardness are more bespoke, from more "primitive" problems like formula games [17].

In this paper, we work towards resolving this infrastructure gap for RE-hardness. We are aware of only a few RE-hardness (which implies undecidability) results for single-player video games: Braid [13], Recursed [8] New Super Mario Bros. [2], and very recent work extending the latter to other Mario games [12]. The result for Recursed uses a fundamentally different approach, but the others use counter machines, inspiring this work on Baba is You. Ani [2] begins to develop a framework for RE-hardness proofs, by extending the motion-planning gadget framework [7] to allow gadgets with infinitely many states, which can act as registers for a counter machine.

In this paper, we focus on *constant-area levels*, meaning instances of a game that have a fixed size in the game world, but may contain any number of objects. To obtain any kind of hardness in this setting, we need a reduction that encodes information in the number of objects present in a small area – for us, in a single cell. In particular, we explain how – and demonstrate with Baba is You – to prove RE-hardness for two decision problems:

1. Given a constant-area level, is it possible to win?
2. For a specific constant level, given a sequence of inputs, is it possible to win after performing those inputs?

These decision problems are always in RE (the class of recognizable problems), for reasonable single-player games (including Baba is You): we can enumerate over all possible input sequences, test whether each results in victory, and accept if so.

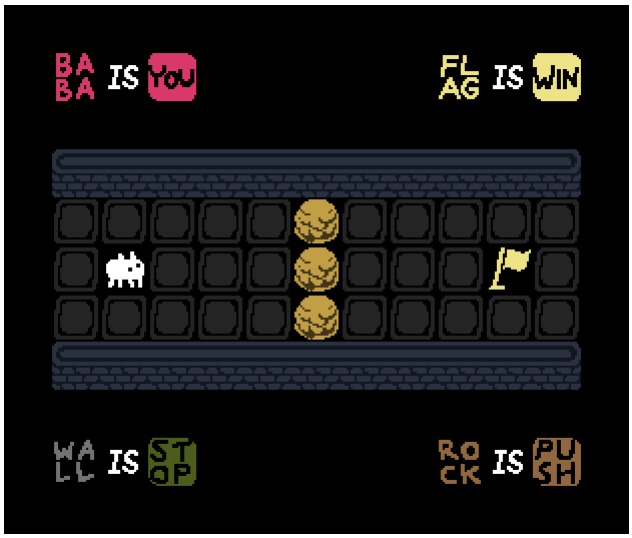

Figure 1 An introductory level from Baba is You. How many different solutions can you find?

Prior undecidability results use levels that scale with the size of the counter machine being constructed, and don't consider hardness on constant-area levels. However, the techniques we use can easily be applied to any similar proof (including Braid [13] and New Super Mario Bros. [2]), provided (for the first decision problem) that we allow levels to start with arbitrarily many objects in the same place – indeed, this has very recently been done for several Mario games [12].

The key idea to making counter-machine proofs work on constant-area levels is to reduce from a specific universal counter machine instead of a family of counter machines. The inputs are encoded in the initial state of the counter machine, which translate to starting the level with large piles of objects (which take constant space) or an input sequence that initializes the counters correctly.

1.1 Baba is You

Baba is You is a 2019 puzzle game by Hempuli in the genre of Sokoban. The player (typically) controls a white creature called `BABA` in a square grid, and can interact with the world by pushing objects. The main gimmick in Baba is You is that almost all of the rules are represented inside the game world, and can be pushed to change the game mechanics. A typical level, such as the one in Figure 1 will have rules like the following.

- `BABA IS YOU`: the player controls `BABA` with the arrow keys.
- `FLAG IS WIN`: the player wins if an object they control (e.g. `BABA`) touches a `FLAG`.
- `WALL IS STOP`: nothing can move into a `WALL`.
- `ROCK IS PUSH`: an object moving into a `ROCK` will push it.
- `DOOR IS SHUT, KEY IS OPEN`: if a `KEY` touches (or is pushed into) a `DOOR`, both are destroyed.

However, there is no fundamental difference between (non-text) objects; any "noun" can be paired with any "adjective", and the challenge in a typical level is to construct useful rules by rearranging the available words.

Baba is You has a feature for sharing custom levels: each uploaded level is given a unique 8-character identifier, and can then be downloaded and played by anyone with the game. We have uploaded all levels in this paper; level codes are given with each level.

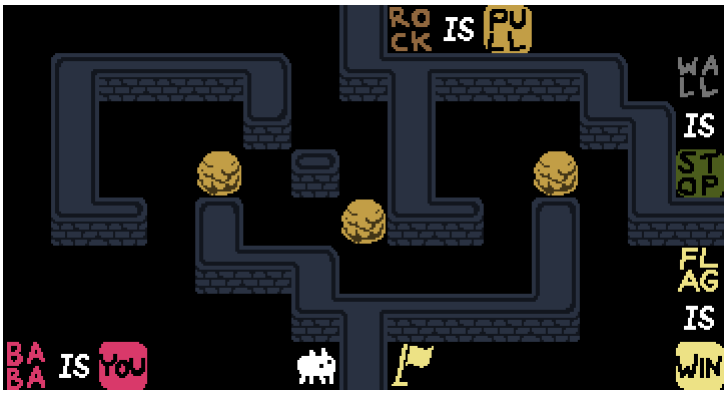

Figure 2 The rules needed to reduce Pull!-*F [3] to Baba is You, and a level based on a gadget that proves PSPACE-hardness [4]. Level code: `VMN9-1J93`.

1.2 Prior work on Baba

In addition to being a video game, Baba is You is a programming language: with the right set of rules, you can embed all sorts computational system in it. Within weeks of the game's release in March 2019, players had constructed Conway's game of life,[1] the Rule 110 cellular automaton,[2] and a Turing machine.[3]. Often, these constructions are accompanied by a claim that Baba is You is "Turing-complete", a term that doesn't have a precise definition in this context.

From a complexity theory perspective, these constructions are reductions to (decision problems about) Baba is You. They show that Baba is You *with infinite space* is capable of arbitrary computation. In particular, provided they trigger victory under some appropriate condition, they show that the decision problem "can you win this level of infinite-space Baba is You" – or even "do you eventually win this fully deterministic level of infinite-space Baba is You" – is RE-hard.

However, levels in Baba is You are finite, and in complexity theory we typically generalize to large but still finite instances. This is especially problematic when extending the construction to an infinite level would require (without other changes) the level starting with infinitely many objects, as is the case for some of the above constructions.

For finite Baba is You, constructing one of these "Turing-complete" systems typically proves PSPACE-hardness – but Rule 110, a perennial favorite due to its simplicity, is not known to be PSPACE-hard, so constructing it doesn't prove much in terms of complexity.

If the goal is PSPACE-hardness, it is simpler to reduce from a problem more similar to Baba is You: Figure 2 demonstrates an easy reduction to Baba is You from Pull!-*F, a PSPACE-complete block-pulling game [3]. But formally proving hardness generally isn't the goal of constructions like those above – rather, they showcase the expressive power of Baba is You by pushing limits on what can be done from within the game.

Geller [11] explicitly considers Baba is You on an infinite board, and proves RE-hardness (though they don't use that word) of determining whether a level is winnable using a reduction from the Post correspondence problem. The proof requires a finite region plus a two-cell-high empty infinite strip.

[1] `https://www.youtube.com/watch?v=YH0NR1kmMUo`
[2] `https://twitter.com/mattar0d/status/1109987662608384000`
[3] `https://www.youtube.com/watch?v=hsXpLx4soQY`

Figure 3 A screenshot of Baba is You, after creating more than 2000 objects.

1.3 Our contributions

We consider Baba is You on bounded levels, and even on constant-area levels.

Unfortunately, the actual game has a practical limitation that puts Baba is You in PSPACE:[4] there can only be six instances of an object in a single cell, and any additional copies will be destroyed. This means the total number of objects is bounded by a polynomial, so the game can be simulated by a polynomial-space nondeterministic machine. Thus Baba is You is in NPSPACE=PSPACE [16].

The this constraint seems to be present for practical reasons, e.g. to prevent memory leaks or lag. But in an idealized theoretical version of Baba is You, there wouldn't be any limit on the number of copies of each object – this limit doesn't fit naturally with the other rules of the game, and it's never relevant to the levels in the game or taught to the player.[5]

We think that the more natural game to consider from a theoretical perspective is Baba is You without this limit. In particular, we believe this is a more natural generalization than infinite boards, so our undecidability result sticks to the spirit of the game better than the prior results which require unlimited space.

There is an additional limitation on constructing levels in practice: levels are described by three layers, and each layer can have only one object in each cell. So we are unable to make a level that starts with more than three objects in the same place. This is not a problem for constructing our counter machines, but it means a constant-area level can't encode the input to a counter machine as a large stack. This restriction seems to exist to make the interface for editing levels more convenient, but again we think the most natural generalization of Baba is You to study allows any pile of objects, even at the start of the level.

Finally, there is another rare practical failsafe in the game. If there are more than 2000 objects (or any of a few similar conditions are met), the entire level is replaced by the screen in Figure 3. While it demonstrates a surprising amount of self-awareness, this feature is inelegant for the same reasons as the limit to 6 copies.

For the remainder of this paper, we consider the version of Baba is You without any of these artificial limitations, which we call *Unlimited Baba is You* – however, with the exception of initialized registers, our levels use at most three objects per cell and thus can be built in the actual game. Our main result is Theorem 1: that Unlimited Baba is You is capable of universal computation, even on 8×17 levels. We prove this by constructing a specific universal counter machine called U_{22} [15] in Baba is You.

[4] Baba is You has randomness, which complicates this – it's not even obvious what the decision problem should be in the presence of randomness. Here we consider the deterministic fragment of Baba is You.

[5] We only learned of its existence when we started building counter machines, and they failed in mysterious ways.

As a consequence of building U_{22}, we obtain Corollaries 2 and 4, that the two decision problems mentioned above are RE-complete for Baba is You.

Beyond Baba, the ideas we use can likely be applied to many games that allow arbitrarily many objects in a bounded region, and we hope to see many similar results for other games in the future.

1.4 Baba is You mechanics

We now describe the specific mechanics of Baba is You that our reductions use. We do not attempt to cover all of the mechanics in the game, or even to define the ones we need in full detail. The game contains *text* objects, which have a single word. These words can be read left-to-right or top-to-bottom to form *rules*, which affect the game. The words in Baba is You can (roughly) be partitioned into *nouns*, *verbs*, *adjectives*, and *prepositions*, which approximately correspond to the usual meanings of those terms.

Nouns refer to kinds of objects, most of which are interchangeable. Verbs are the cores of rules; by far the most common is `IS`, which usually assigns an adjective to a noun. Adjectives represent mechanics, which affect the behavior of nouns the apply to. Prepositions are used to make *noun phrases* like "`BABA ON FRUIT`", which define a new set of objects and are used in mostly the same way as nouns.

We use n, a, and N as variables for nouns, adjectives, and noun phrases (including nouns), respectively. In addition to nouns (which are listed in Figure 6), our constructions use the words in Table 1.

In Section 2, we discuss counter machines and how they can be used to prove RE-hardness, even for constant-area instances of a problem like Unlimited Baba is You. In Section 3, we construct counter machines in Unlimited Baba is You, and obtain RE-hardness for 8×17 levels.

2 Count is Universal

A counter machine is a finite state machine with access to a constant number of *registers* which each store an arbitrary nonnegative integer. A counter machine interacts with its registers via *instructions* such as "add 1" or "transition to a different state depending on whether the register is 0".

For instance, Figure 4 shows a simple counter machine with four registers which multiplies two numbers. The four registers are called 0 through 3. It uses two kinds of instructions which act on a particular register:

- Increment (inc): increase the register by 1.
- Jump if zero and decrement (dec): go to one state if the register is positive and a different state if it's 0, and decrease it by 1 in the former case.

Korec [15] calls these RiP and RiZM, respectively.

We draw increment instructions as rectangles and jump-and-decrement instructions as diamonds. The outgoing path from a diamond marked with a 0 is the one taken when the register is 0. To make them convenient to build in Baba is You, we draw our counter machines so that paths go straight through increment instructions, go straight through jump-and-decrement instructions to take the zero branch, and turn to take the nonzero branch. We write R_i for the value of register i.

To understand the counter machine in Figure 4, start the registers with $R_1 = m$, $R_2 = n$, and $C = D = 0$. The machine runs the main loop n times, decrementing R_2 on each loop and stopping when $R_2 = 0$. Each time, it moves R_1 to both R_0 and R_3, then moves R_3 back to R_1, for a net effect of increasing R_0 by m. When it stops, $R_0 = mn$.

Table 1 The words in Baba is You we use other than specific nouns.

Word	Part of speech	Behavior
IS	verb	The "rule N IS a" applies the mechanic corresponding to a to all instances of N. The rule "N IS n" replaces each instance of N with an n.
MAKE	verb	Each time step, the rule "N MAKE n" creates an n in each cell with an N that doesn't already have an n.
YOU	adjective	This object is controlled by the player's inputs. There can be zero or multiple such objects. The player can also choose to *pass*, which lets other rules process one time step.
WIN	adjective	If this object is in the same cell as an object that's YOU, the player wins.
MOVE	adjective	Each time step, this object moves one unit in the direction it's facing. If it can't move forward (e.g. because it's the edge of the level), it bounces off (the details won't matter for our reduction).
SHIFT	adjective	Each time step, move all other objects in the same cell one unit in the direction this object is facing, and set their facing directions to match.
FLOAT	adjective	This object is on a second "layer", which is independent of the default layer for some adjectives including WIN and SHIFT.
STOP	adjective	Other objects can't move into the cell containing this object.
PUSH	adjective	If another object attempts to move into the cell containing this object, this object is pushed in the same direction.
PULL	adjective	This object is STOP, and if an object adjacent to it moves away, this object follows.
OPEN SHUT	adjectives	If an OPEN object is in the same cell as a SHUT object, destroy both. This always destroys objects in pairs – it will destroy the same number of OPEN and SHUT objects, up to whichever there are fewer of.
ON	preposition	The noun phrase "n ON n'" refers to each n which is in the same cell as an n'.
AND	preposition	The noun phrase "n AND n'" refers to both each n and each n', and similarly for conjunctions of more than two nouns. AND can also be used to combine adjectives, e.g. "BABA IS YOU AND FLOAT".
NOT	preposition	The noun phrase "NOT n" refers to object that aren't n. A rule of the form "n IS NOT a" overrules the rule "n IS a".
TEXT	noun	Refers to all text objects.

There are many possible instructions we can allow counter machines to use. From the perspective of using counter machines to prove undecidability, there is a tradeoff between simplicity and efficiency: more powerful instruction sets require more work to implement, but may allow us to use a counter machine that has fewer registers or states, or that simulates a Turing machine more efficiently.

Korec [15] constructs explicit universal counter machines for a variety of instruction sets at different points on this tradeoff. For Baba is You, we will use counter machines with the two instructions described above.

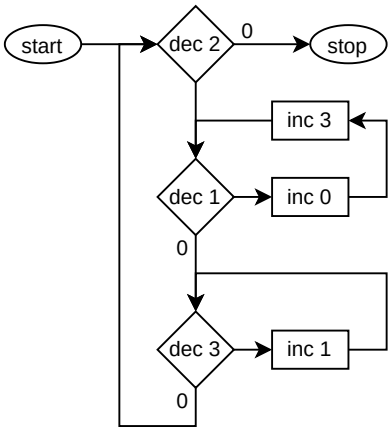

Figure 4 A counter machine which inputs two numbers A and B and outputs their product on register D. Rectangles are increment instructions; diamonds are jump if zero and decrement instructions, with 0 indicating the branch to take when the register is 0.

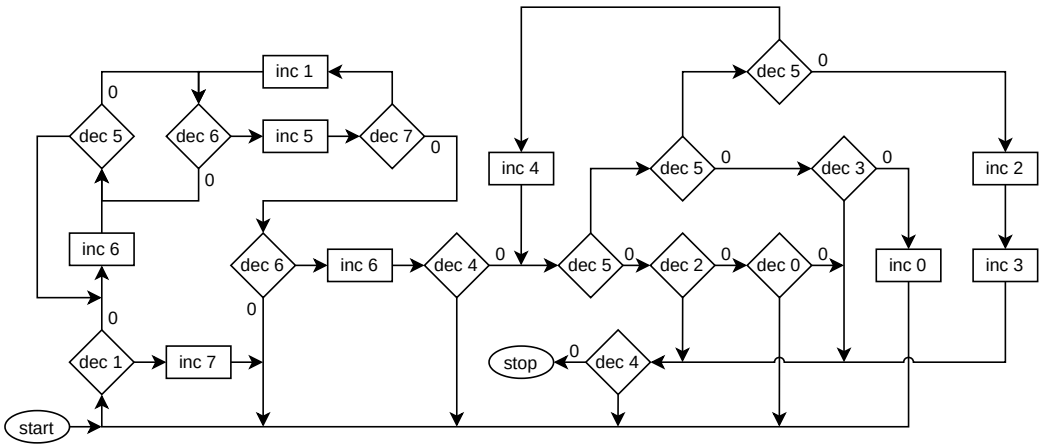

Figure 5 Korec's strongly universal counter machine U_{22} [15].

The counter machine we will use is the 22-state machine U_{22} [15], which is shown in Figure 5 laid out in a way that suits our purposes. This machine is *strongly universal*, meaning the following. Let Φ_i be an enumeration of partial computable functions $\mathbb{N} \to \mathbb{N}$, e.g. as Turing machines. There is a computable function g such that, if we run U_{22} starting from $R_1 = g(i)$, $R_2 = n$, and all other registers 0, it halts if and only if $\Phi_i(n)$ is defined, and if so when it halts $R_0 = \Phi_i(n)$. That is, if we input $g(i)$ and n, this machine computes $\Phi_i(n)$.

We now attempt to summarize how U_{22} works. The input on register 1 is actually a description of a particular kind of counter machine with three very specific instructions, which U_{22} simulates – the first step of computing g is to convert your Turing machine into such a counter machine. Registers 0, 2, and 3 are for the simulated machine. The description and current state of the simulated machine are kept in registers 1 and 4, respectively. The left half of U_{22} parses these two registers, unpacking the encoding of the counter machine at the current state. The result is that R_5 encodes both the next instruction to execute (which determines where the machine exits the loop with three dec 5 instructions) and the next state to switch to (which becomes R_4). The bottom right section carries out the instruction, possibly decrementing R_4 to allow the simulated machine to branch, before moving to the next step or halting.

2.1 Game Make Count

Consider a game capable of building counters in bounded space, typically by stacking arbitrarily many objects in the same place. Suppose that we can also implement the instructions used by U_{22}, or some other sufficiently powerful instruction set. Games that fit these criteria include Braid [13], several Mario games [10, 12], and Baba is You (as we will show).

After designing these components, it is straightforward to prove RE-hardness for such a game: convert any Turing machine to an appropriate counter machine, and then construct the counter machine in the game. But we can do better, by using a universal counter machine to obtain results for constant-area levels.

Specifically, construct a level which simulates U_{22} (or another universal counter machine). Then that level is capable of universal computation: if it is initialized with the appropriate set of objects to represent $R_1 = g(i)$ and $R_2 = n$, when it halts it will encode $R_0 = \Phi_i(n)$.

To prove RE-hardness for constant-area levels, we need two additional conditions: the counter machine terminating should trigger victory, and the initial state of a level must be allowed to have arbitrarily many objects to encode an arbitrary initial register value. Then we can reduce from the halting problem. Given a number encoding a Turing machine i and an input n, build the level that simulates U_{22} initialized as above, so the player wins if and only if $\Phi_i(n)$ is defined.

For RE-hardness of a constant level after a given sequence of inputs, we must build something that allows the player to choose the initial values of the input registers. Then we can again reduce from the halting problem: start by making a sequence of moves that initializes the counter machine as described above and then transitions to deterministically simulating the machine. After this sequence, the player will win if and only if $\Phi_i(n)$ is defined.

In the next section, we will do all of this for Unlimited Baba is You. We will build counter machines by representing each register as the number of copies of an object in a single cell. Then we will describe a level that runs U_{22} with no player choices, and a version of it that allows the player to set input values during the initialization phase.

3 Baba is Count

We now start building counter machines in Unlimited Baba is You. There are many possible approaches to doing this; we choose one which is space-efficient and will result in levels visually matching diagrams like Figure 4.

In our construction, the player will never be able to make choices – nothing will be `YOU` until the frame the player wins, so all the player can do is wait for the counter machine. We will never move text or have rules change. Each register will be represented by a particular objects, with the number of copies matching the current value. All of these registers will be in the same cell, and will move around the counter machine together. When the pile encounters certain other objects, it will gain or lose register objects and be redirected to simulate the counter machine.

The moving pile will be "carried" by a `CART`, with the rule "`CART IS MOVE`". It will be routed using `BELT`s, with "`BELT IS SHIFT`". We will decrement registers by using a `GATE` to consume an item, with '`GATE IS SHUT`'.

Each register will use three types of object, which we call `REG`, `INC`, and `DEC` to explain a generic register. `REG` is the object that forms a pile to keep track of the value of the register. We have the rule "`REG ON CART IS MOVE`" so that it follows the cart, but other instances of `REG` waiting to be picked up stay where they are.

To increment, we have "INC MAKE REG". Each copy of INC will have one REG sitting on it, and if the CART drives over that cell the REG will be added to the pile. The facing direction of the waiting REG matches that of the INC, so we need place the INC facing the direction the CART will cross it.

To decrement, we have "DEC MAKE GATE" and "REG ON DEC IS OPEN". When the pile of registers crosses DEC, the GATE on it will consume exactly one REG. Having REG only be OPEN when on DEC ensures that we won't consume an item from the wrong register.

To branch, we again use DEC. We have the rule "DEC ON REG IS SHIFT", with the DEC facing a direction orthogonal to the direction the CART is moving. If the register is zero, the DEC is not on a REG, so the CART continues straight. Otherwise, the DEC deflects it.

There is a potential conflict between decrementing and branching: if the pile MOVEs onto the DEC and is immediately SHIFTed, it fails to consume one of the REGs–SHIFT is applied before OPEN and SHUT. We can avoid this by having a BELT directly into every DEC, so that the pile MOVEs onto the BELT, is SHIFTed onto the DEC, and then the GATE can consume a REG (but the pile's direction will change to match the DEC if the register is nonzero).

When the counter machine halts, the CART arrives at a FLAG. So that this results in victory, we have the rules "CART ON FLAG IS YOU" and "FLAG IS WIN".

To summarize: the objects we use are CART, BELT, GATE, FLAG, and three objects (REG, INC, DEC) for each register. The rules are

- CART IS MOVE
- BELT IS SHIFT
- GATE IS SHUT
- CART ON FLAG IS YOU
- FLAG IS WIN
- For each register:
  - REG ON CART IS MOVE
  - INC MAKE REG
  - DEC MAKE GATE
  - REG ON DEC IS OPEN
  - DEC ON REG IS SHIFT

Figure 6 shows these objects, including the objects we'll use for the eight registers of U_{22}.

Figure 7 shows an Unlimited Baba is You level implementing the simple counter machine from Figure 4. The actual counter machine in is the top left, below it are the "generic" rules, and to the right is a square of rules for each of the four registers. The CART is facing up, and start by driving into the top belt and then the SWORD (it needs a bit of space at the start so the SWORD has time to MAKE a GATE).

To supply inputs, we start the level with a pile of m ROBOTs and n LEVERs in the same cell as the CART – for readability, the figure shows $m = n = 0$. The CART will eventually reach the FLAG carrying mn PAPERs (and still m ROBOTs), at which point the player wins.

It's somewhat inelegant that we have to encode the input in the level: if a player simply wants to know what the product of two numbers is, they have to load up the level editor and stick in some ROBOTs and LEVERs (which in practice requires adding more space). We modify the level as shown in Figure 8 to allow the player to choose the inputs to the counter machine.

We have changed the rules a bit, added a TILE above the CART and created the cage on the left. Now the counter machine doesn't start running immediately, and the player doesn't win when it halts. Instead, the player controls BABA, who is stuck in the cage.

Figure 6 The Baba is You objects used in our counter machines. The objects at the top are not connected to a specific register, and each register 0 through 7 uses the items in one of the rows, listed in the order `REG`, `INC`, and `DEC`. The `DEC`s are chosen so that the direction they're facing is visible – all objects have a facing direction, but some don't have directional sprites. We have attempted to use semantically-related items to make the registers somewhat intuitive, but the limited inventory of objects makes this challenging.

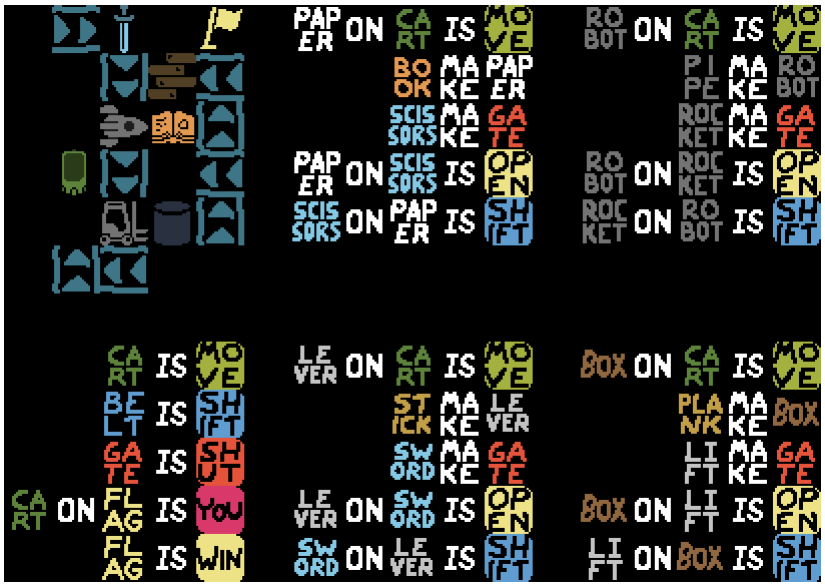

Figure 7 A counter machine that multiplies two numbers (Figure 4) built in Unlimited Baba is You. Level code: `EUQT-TCZJ`.

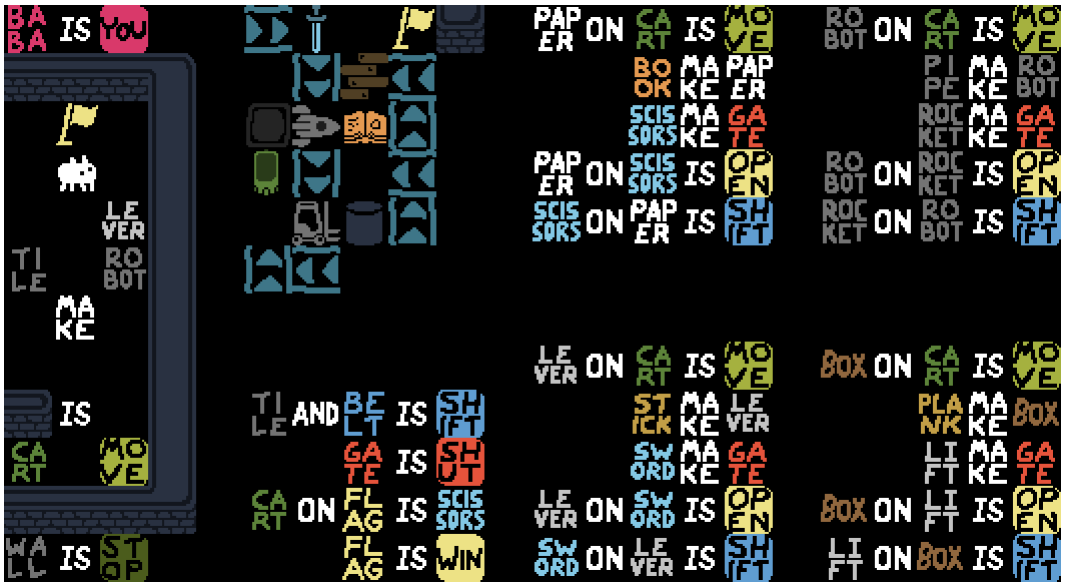

Figure 8 The same Unlimited Baba is You counter machine as in Figure 7, but with infrastructure that lets the player choose inputs. Level code: `Q1V8-PRIE`.

The player will eventually create `CART IS MOVE`, which starts the counter machine and requires putting `MAKE` somewhere it can't be used. Before this, the player chooses the inputs to R_1 and R_2, which are the initial numbers of `ROBOT`s and `LEVER`s, respectively. They do so by creating `TILE MAKE ROBOT` and `TILE MAKE LEVER` for any desired number of time steps: push `TILE` down, and then push the nouns on the right down three times, waiting in between for the appropriate number of steps while the created objects `SHIFT` off the `TILE`. The player can use a slightly different sequence to make either or both registers start at 0 – the cage gives just enough freedom to allow this without letting the player create any unintended rules or `MAKE` any more objects after starting the counter machine.

To read out the product the machine computes, the player must be attentive when the `CART` reaches the `FLAG`: it turns into `SCISSORS`, which create `GATE`s to consume one `PAPER` each time step – they decrement register 0. The `WALL` prevents the `SCISSORS` from `SHIFT`ing the pile of counters away. The player receives the desired product by counting how many times they hear the distinctive noise of mutual annihilation of `OPEN` and `SHUT`.

Finally, the player moves to the `FLAG` at the top of their cage. This allows them to complete the level after it multiplies the numbers of their choosing, or to abandon it early should they decide to use a more efficient calculator.

3.1 U_{22} in Baba

Since we've described how to construct general counter machines in Unlimited Baba is You, we can build Korec's U_{22} [15]. This is shown in Figure 9. The actual counter machine is about a third of the space of the level, with the rest being taken up by rules.

As before, the input is given as a stack of `ROBOT`s and `LEVER`s on the cart, representing the initial values of R_1 and R_2, respectively. If we wish to compute $\Phi_i(n)$, we should set up the level with $g(i)$ `ROBOT`s and n `LEVER`s. Then the player wins exactly when $\Phi_i(n)$ exists, and if so the number of `PAPER`s on the `CART` just before victory is $\Phi_i(n)$. We now apply the construction we used to create Figure 8, to allow us to perform computation in a level that is not only constant-area, but truly constant. This level is shown in Figure 10.

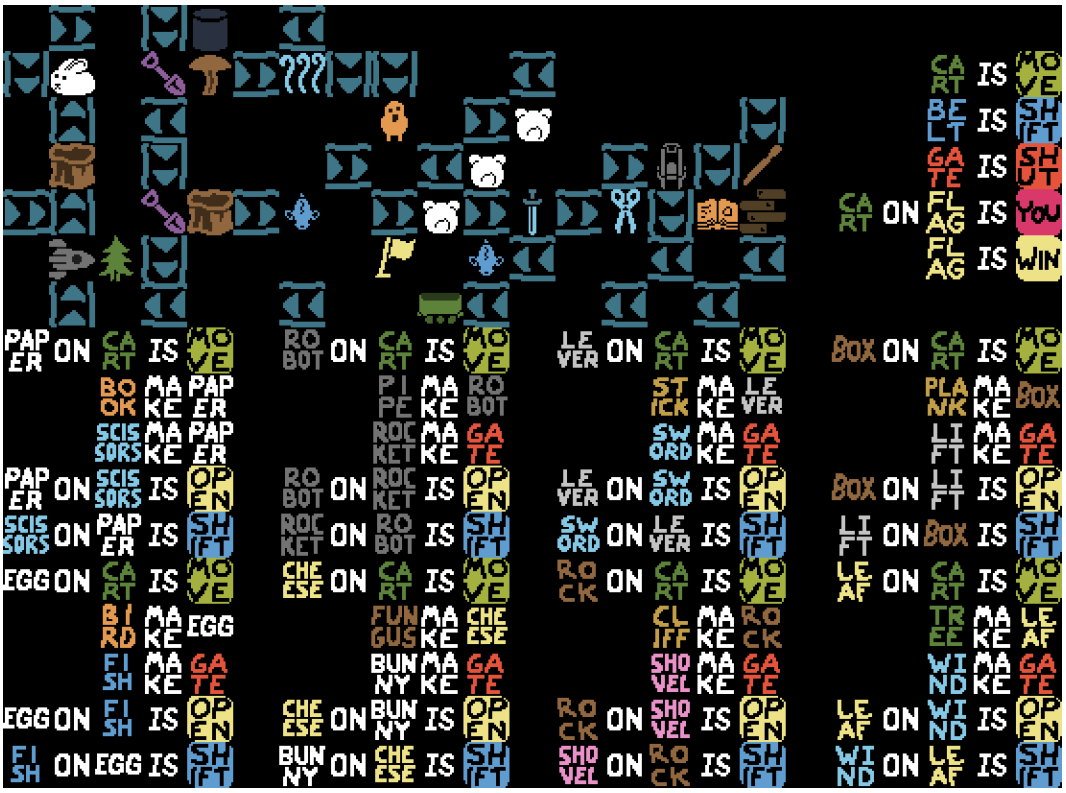

Figure 9 U_{22} (Figure 5) in Unlimited Baba is You. Level code: `7LKG-WKBJ` (initialized with one `ROBOT` so that it terminates).

The cage is now in the top right, and we have moved some rules to make space. There are two differences from the cage in Figure 8: there isn't an accessible `FLAG`, and the `CART` becomes `BABA` (instead of `SCISSORS`) when it reaches the `FLAG`, triggering victory.

As before, the player can choose how many `ROBOT`s and `LEVER`s to start the counter machine with, setting the initial values of R_1 and R_2. Once the player makes `CART IS WIN`, there is nothing of use they can do but wait, and will eventually win if the counter machine halts on the provided input.

3.2 Smaller levels

We now compact our counter machine construction to show RE-hardness for smaller constant-area levels. The actual counter machine in Figure 9 is already pretty compact; our goal here is to make the rules take less space. In addition to leaving less empty space, there are a few tricks we employ:

- We place rules in the same region as the counter machine. We need `TEXT IS FLOAT` so that text isn't affected by `BELT`s, and `TEXT IS NOT PUSH` so that it isn't affected by the moving `CART`.
- Multiple words can go in the same cell: to write `ROCKET MAKE GATE` and `SWORD MAKE GATE`, we have cell with both words `ROCKET` and `SWORD` in front of `MAKE GATE`.
- Another way to make that pair of rules is with `AND`: the rule `ROCKET AND SWORD MAKE GATE` is more efficient than having both rules separately.

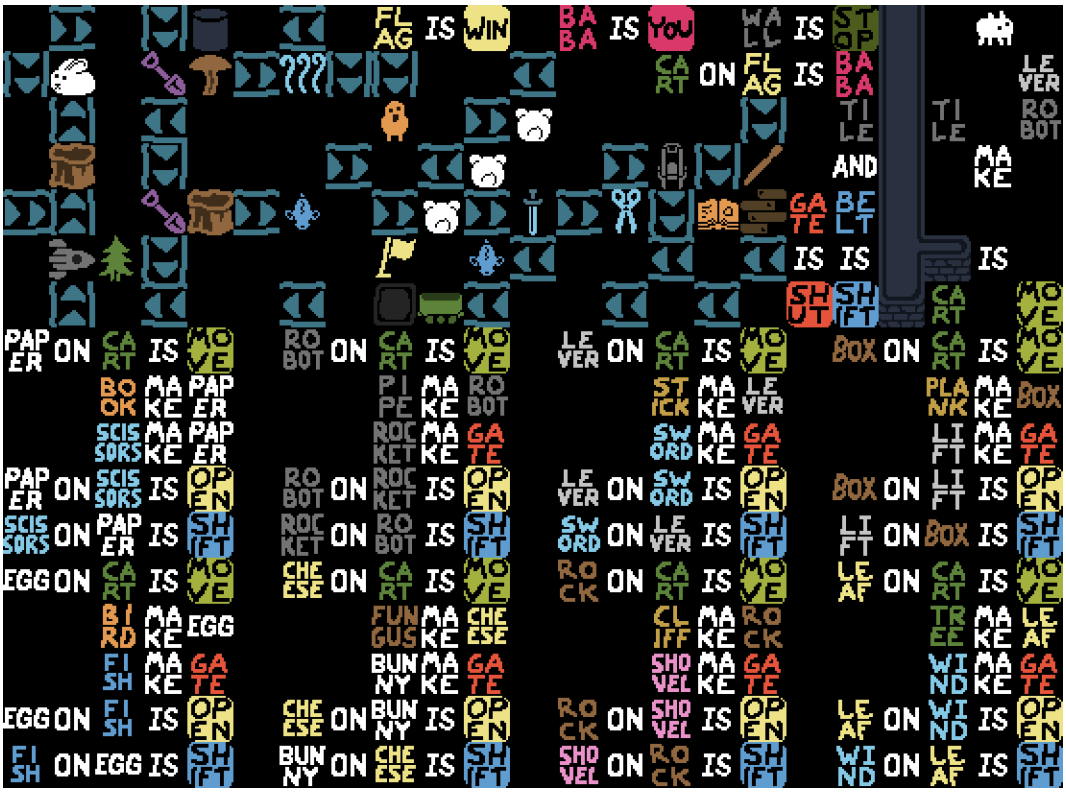

Figure 10 The same construction of U_{22} as in Figure 9, but now the player can control `BABA` to specify inputs to the counter machine. Level code: `ZFGG-WH6Z`.

- Combining the two previous points, we make rules like `SWORD/ROCKET AND FISH/BUNNY MAKE GATE`, which is parsed as four separate rules of the form `X AND Y MAKE GATE`. We end up with multiple copies of `SWORD MAKE GATE` – this is fine for `MAKE`, but is an issue for other verbs like `MOVE`, where two copies of `CART IS MOVE` means the `CART` moves two cells per time step.

The level editor allows us to place multiple objects in the same cell using layers, but there is a limit of 3 objects per cell.

The result is the unreadable mess in Figure 11, which is an 8×17 level. It behaves the same as in Figure 9. In particular, inputs are given as piles of left-facing `ROBOT`s and `LEVER`s in the seventh row, and the output is the number of `PAPER`s when the player wins.

▶ **Theorem 1.** *An 8×17 level of Unlimited Baba is You is capable of universal computation, where inputs and outputs are encoding as the number of identical objects in a single cell.*

By *universal computation*, we mean that with the inputs $g(i)$ and n in the format specified for some computable function g, the output is $\Phi_i(n)$, again in the format specified.

▶ **Corollary 2.** *Deciding whether an 8×17 level of Unlimited Baba is You is solvable is RE-complete (and in particular undecidable).*

We can also compactify the level in Figure 10, which lets the player specify inputs to the counter machine. The only additional complication is that we need `TEXT IS NOT PUSH` to not exist until after the player is done initializing. We also place the `TILE` under the `CART` to avoid needing `TILE IS SHIFT`.

Figure 11 U_{22} in Baba is You (Figure 9), after compactifying at the cost of clarity. Level code: `4ZGV-BPQ3` (initialized with one `ROBOT` so that it terminates).

Figure 12 U_{22} in Baba is You with inputs given by the player (Figure 10), after compactifying in the same way as Figure 11. Level code: `7GG1-1PTH`.

▶ **Corollary 3.** *There is a specific* 8×21 *level of Unlimited Baba is You which is capable of universal computation, where inputs are given by a sequence of moves the player makes and the output is the number of identical objects in a single cell when the player wins.*

To compute $\Phi_i(n)$, the player performs a fixed sequence of moves to make `TILE MAKE ROBOT`, waits $g(i) - 1$ times, performs another fixed sequence of moves to destroy that rule and create `TILE MAKE LEVEL`, waits $n - 1$ times, and finally destroys that rule as well while creating `CART IS MOVE`. If $g(i)$ or n is zero, the player must use a slightly different initialization sequence.

▶ **Corollary 4.** *There is a specific* 8×21 *level of Unlimited Baba is You for which the decision problem "if the player starts by performing a given sequence of inputs, can they go on to win (without undoing)?" is RE-complete.*

Results about constant-area levels have another small advantage: our levels are buildable in the Baba is You level editor, which has a maximum size of 33×18 (and three objects per cell) – and you can play them at the level codes given. If you enter initial values for the levels where that's possible (Figures 8, 10 and 12), you can then run the counter machine the level simulates, with only the caveat that the values of registers are capped at 6.

References

1. Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic Nintendo games are (computationally) hard. *Theoretical Computer Science*, 586:135–160, 2015.
2. Hayashi Ani. *Unsimulability, Universality, and Undecidability in the Gizmo Framework*. PhD thesis, Massachusetts Institute of Technology, 2023.
3. Hayashi Ani, Sualeh Asif, Erik D. Demaine, Jenny Diomidov, Della Hendrickson, Jayson Lynch, Sarah Scheffler, and Adam Suhl. PSPACE-completeness of pulling blocks to reach a goal. *Journal of Information Processing*, 28:929–941, 2020.
4. Hayashi Ani, Jeffrey Bosboom, Erik D. Demaine, Jenny Diomidov, Della Hendrickson, and Jayson Lynch. Walking through doors is hard, even without staircases: Proving PSPACE-hardness via planar assemblies of door gadgets. In *10th International Conference on Fun with Algorithms (FUN 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
5. Hayashi Ani, Lily Chung, Erik D. Demaine, Jenny Diomidov, Della Hendrickson, and Jayson Lynch. Pushing blocks via checkable gadgets: PSPACE-completeness of Push-1F and Block-/Box Dude. In *11th International Conference on Fun with Algorithms (FUN 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
6. Jeffrey Bosboom, Josh Brunner, Michael Coulombe, Erik D. Demaine, Della Hendrickson, Jayson Lynch, and Elle Najt. The Legend of Zelda: The complexity of mechanics: Discrete and computational geometry, graphs, and games. *Thai Journal of Mathematics*, 21(4):687–716, 2023.
7. Erik D. Demaine, Della Hendrickson, and Jayson Lynch. Toward a general complexity theory of motion planning: Characterizing which gadgets make games hard. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
8. Erik D. Demaine, Justin Kopinsky, and Jayson Lynch. Recursed is not recursive: A. jarring result. In *31st International Symposium on Algorithms and Computation (ISAAC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
9. Erik D. Demaine, Joshua Lockhart, and Jayson Lynch. The computational complexity of Portal and other 3D video games. In *9th International Conference on Fun with Algorithms (FUN 2018)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2018.
10. Erik D. Demaine, Giovanni Viglietta, and Aaron Williams. Super Mario Bros. is harder/easier than we thought. In *8th International Conference on Fun with Algorithms (FUN 2016)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.
11. Jonathan Geller. Baba is You is undecidable. *arXiv preprint arXiv:2205.00127*, 2022.
12. MIT Hardness Group, Hayashi Ani, Erik D. Demaine, Holden Hall, Ricardo Ruiz, and Naveen Venkat. You can't solve these Super Mario Bros. levels: undecidable Mario games. In *12th International Conference on Fun with Algorithms (FUN 2024)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
13. Linus Hamilton. Braid is undecidable. *arXiv preprint arXiv:1412.0784*, 2014.
14. Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. CRC Press, 2009.
15. Ivan Korec. Small universal register machines. *Theoretical Computer Science*, 168(2):267–301, November 1996. doi:10.1016/S0304-3975(96)00080-1.
16. Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.
17. Matthew Stephenson, Jochen Renz, and Xiaoyu Ge. The computational complexity of Angry Birds. *Artificial Intelligence*, 280:103232, 2020.
18. Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54:595–621, 2014.