IJHSR
ijhighschoolresearch.org
■ RESEARCH ARTICLE

A new approach to Turing completeness in *Baba is You*

Caleb J. Su

Mercer Island High School, 8219 SE 28th Street, Mercer Island, Washington, 98040, USA; calebjzsu@gmail.com
Mentors: Dr. Jim Purbrick, Dr. Chaitanya Mishra, Dr. Dan Grossman, Alvin Shi

ABSTRACT: *Baba is You* is a 2019 indie puzzle game whose levels are two-dimensional grids of objects. This paper demonstrates how a universal Turing machine could be created as a custom level in the game if a specific in-game limit were to be removed. A player could perform any computation by playing this level, allowing infinitely many new and meaningful puzzles to be created. Furthermore, the construction is such that the level is won if and only if the Turing machine halts, proving perfect play of the game to be unattainable by any computer. The machine is found to be exponentially time-complex, and the paper concludes with a discussion of the practical feasibility of in-game computation.

KEYWORDS: Systems Software; Other; Logic in Computer Science; Computational Complexity; Turing Completeness.

■ Introduction

Baba is You is a puzzle game released in 2017. Like many puzzle games, it is turn-based, and playable characters in its levels can push objects.[1] However, *Baba is You* contains a type of object that can alter the properties of the level and change, for instance, which characters are playable and which objects are solid. Furthermore, if a level meets certain conditions, its objects can move autonomously without player input. The emergent dynamics of such a system have led to some consideration among players of the computational complexity of *Baba is You* levels.

Previous Work:

In 2019, the game was considered Turing complete on an infinite grid. The cellular automaton, Rule 110, has been proven Turing complete, and Matthew Rodriguez demonstrated how it could be simulated in a custom level.[2] Rodriguez's design required the *Baba is You* grid to be extended to an infinite size, though the game's maximum level size is 33 by 18 squares. Thus, his proof presented an application of general *Baba is You* rules rather than a level usable for computation in the actual game.

New Contributions:

This paper demonstrates for the first time how Turing completeness can be achieved in standard *Baba is You* by describing the first universal Turing machine that fits within the game's native level size. The effects of this proof are that one can calculate anything computable simply by playing vanilla *Baba is You* and that future expansions or online content for the vanilla game could contain levels provably unsolvable by any computer. Unfortunately, the design cannot store large Turing machine configurations because of a limitation on in-game object stacking. However, after learning of this design, the developer of Baba is You considered removing this limitation in a future port of the game precisely because of the potential to create such levels.[3]

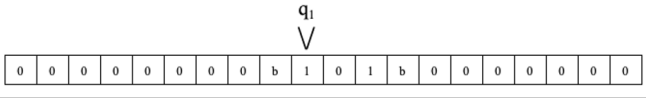

Figure 1: A representation of a finite portion of a Turing machine. The tape head is represented by a V and q_1 refers to its state. It is currently above a square with the symbol "1".

Preliminaries:

A Turing machine is a theoretical model of computing first proposed in 1936. It can be described as a tape and a tape head.

The tape is divided into squares and extends infinitely in both directions. Each square is marked with a symbol. Before starting the machine, the user may write their own symbols on a finite portion of the tape beyond which zeroes are assumed to extend infinitely in both directions.

At all times, the tape head is occupying one (and only one) square of the tape. The tape head also keeps track of a state: at all times, the tape head is in one (and only one) state. A construction is shown in Figure 1.

The tape head is capable of "writing" symbols that replace symbols on the squares it occupies, as well as moving one square to the left or to the right, before altering its own state. It can also halt the Turing machine permanently. The tape head determines which of these actions to take based on its current state and the symbol it is on. It will keep reading and editing the tape in a deterministic, repeating, and sometimes infinitely long process.

Meanwhile, *Baba is You* is a turn-based puzzle game consisting of objects on a rectangular grid. It is a Nomic game, meaning that almost all of its "rules" can be changed, from the playable character to the win condition.[4] *Baba is You* implements its system of "rules" using "text" objects. At the start of each turn, the game examines all text objects to find which ones form what it defines as rules. Rules declare a type of object to have a specific behavior. Rules can be read left-to-right or top-to-bottom.

DOI: 10.36838/v5i6.23

Figure 2 contains three text objects that create a rule, "BABA IS YOU," which declares baba objects in the level to be playable. Since the rule exists on the level, the two babas will move with directional input from the player.

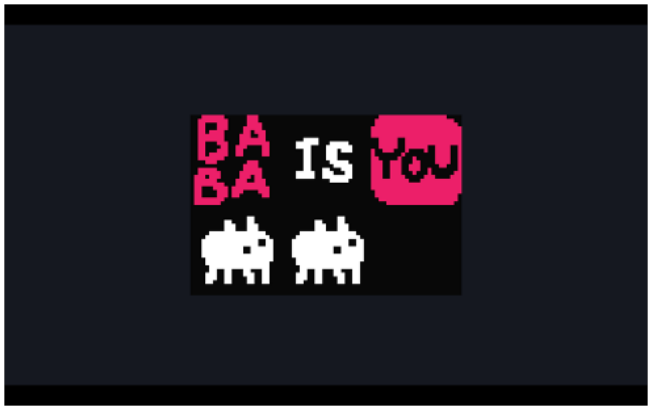

Figure 2: A level containing two "baba" objects ( ), as well as three text objects (which spell "BABA IS YOU").

Figure 3: A level containing two baba objects and five text objects.

If the player somehow breaks the sentence (e.g., by separating or destroying a word), the rule that babas are playable will no longer apply, and the babas will no longer respond to directional input from the player.

Rules can be conditional, only applying to objects when they are "on" or "near" an object of a given type. For instance, the rule "ROCK NEAR BABA IS PUSH" means that all "rock" objects adjacent to baba objects are pushable.

A mechanic essential to the design of this universal Turing machine is the property "WORD." Objects defined as "WORD" can be used in sentences as if they were their text counterparts.

The rule at the top of Figure 3 declares all babas to have the property "WORD." Thus, both babas in the level can be used in sentences as if they were text. This allows the second line to be interpreted as "baba IS YOU".[a] Thus, both babas become playable as in the first example.

The Design:

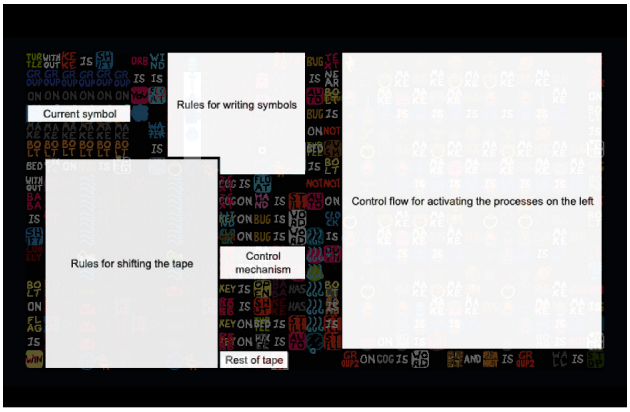

Figure 4: A schematic of the design overlaid with an image of the in-game level, meant to highlight the machine's basic structure.

Size constraints of the level prevent an arbitrarily large physical tape from being laid out across the level with a physical tape head moving along it. Instead, the design uses overlapping objects so it can store both the tape to the left of the tape head and the tape to the right of the tape head on single squares.[b] (The symbol directly under the tape head is stored separately.) This storage method can hold arbitrarily large amounts of tape in fixed amounts of space, at the cost of turning simple operations like shifting the tape into complex conversion processes. Indeed, far more of the level is devoted to implementing operations than to storing the tape itself, as seen in Figure 4.

The Turing machine the design simulates is Rogozhin's four-state, six-symbol universal Turing machine from his 1996 paper, *Small universal Turing machines*. It comes from a set of universal Turing machines that simulate string-modification algorithms called "tag systems" that have been proven to be universal.[5] The process requires more setup for initializing the tape but contains fewer states and symbols than a direct emulation of other Turing machines.[6] (Rogozhin's six-symbol machine contains the fewest total instructions and thus takes up the least in-game space in *Baba is You*.)

To closer match the Baba is You objects chosen to represent them, the states q_1, q_2, q_3, and q_4 in Rogozhin's machine have been renamed to q_{seed}, q_{sprout}, q_{tree}, and q_{trees} respectively. Furthermore, to better fit the arithmetic method used to express the symbols, Rogozhin's symbols 1, b, b [right], b [left], c, and 0 have been renamed to 1, 2, 3, 4, 5, and 0 respectively.

Storing the Current Symbol:

Figure 5: Part of the level featuring a cog ( ) on a track of six wind objects. The cog is on the fifth tile, so the Turing machine's current symbol is a 5.

The symbol the tape head is on is stored using a cog's position on one of the six tiles in Figure 5. A cog on the first tile means that the current symbol is a 1, a cog on the second tile represents the symbol 2, and so on through the symbols 3, 4, 5, and 0.

DOI: 10.36838/v5i6.23

[a] For clarity, text objects are named in capital letters, but all other objects are named in lowercase, even when they function as WORDs.

[b] This section explains how the design would work if the limit on stacking copies of a single object were arbitrarily high. Currently, the limit is 6.

Unique identifying objects on each square help the game determine the placement of the cog. The first tile (for the symbol 1) contains a dust object ( ), the second tile (for the symbol 2) contains a brick object ( ), and so on through rubble ( ), rock ( ), cliff ( ), and planet ( ).

Storing the Rest of the Tape:

This design stores the tape to the left of the tape head using the number of babas on the cart seen in Figure 6. The number of babas is set to the base 6 number created by reading each

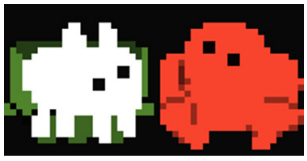

Figure 6: Part of the level featuring several babas ( ) on a cart and several kekes ( ) on a car[c].

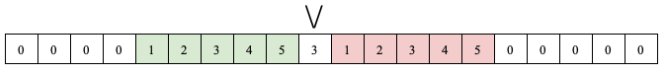

Figure 7: A sample Turing machine tape. V represents the position of the tape head, and the area from the leftmost nonzero symbol to the symbol directly to the left of the tape head is highlighted in green. The area from the rightmost nonzero symbol to the symbol directly to the right of the tape head is highlighted in pink.

symbol from the leftmost nonzero symbol to the symbol directly to the left of the tape head. The tape to the left of the tape head in Figure 7 can be read as the number 12,345 in base 6 or 1,865 in base 10. Thus, it is encoded as a stack of 1,865 babas on the cart.

Similarly, the design stores the tape to the right of the tape head using the number of kekes on the car seen in Figure 6. The number of kekes is set to the base 6 number created by reading the symbols from the rightmost nonzero symbol to the symbol directly to the right of the tape head. The tape to the right of the tape head in Figure 7 can be read (right to left) as 54,321 in base 6 or 7,465 in base 10. Thus, it is encoded as a stack of 7,465 kekes on the car.

Each finite configuration of the left or right side of the tape can thus be read as a unique number and represented with a unique number of objects.

Performing Operations:

Instead of writing a symbol, shifting the tape, and then reading the new symbol, this design executes instructions by shifting the tape, reading the new symbol, and then writing a symbol on the space it just left.

The "shift left" operation, for instance, works by dividing the

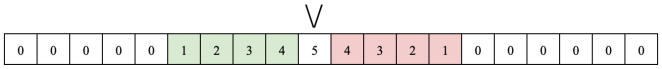

Figure 8: Another sample Turing machine tape.

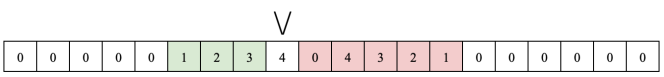

Figure 9: The Turing machine tape from Figure 8 after a "shift left" operation as this design implements it.

number of babas by six (rounding down), setting the current symbol to the remainder, and multiplying the number of kekes by six. Because of the base six arithmetic, this serves to remove the last digit from the number of babas and set the current symbol to said digit. It also adds a 0 to the end of the number of kekes. The result is shown in Figure 9: it almost completes the shift operation but puts a 0 in the square the tape head leaves.

Writing a symbol in the square the tape head leaves is now a matter of changing the last digit of the number of kekes. It is a 0, so the design can change it to a number from 1 to 5 simply by adding that number of kekes to the car.

The right-shifting functions work equivalently to the left-shifting functions except that they divide the kekes by six, multiply the babas by six, and add up to five babas to the cart.

Splitting objects into copies is a relatively common gimmick in *Baba is You* levels. This design uses the most common technique of abusing how the game resolves conflicting rules about baba or keke transformation when such rules are created on the same turn.

Dividing stacks of objects, however, is featured in almost no *Baba is You* level solutions, because losing objects is usually counterproductive. This design accomplishes division using the "OPEN" mechanic: when an object is "opened" in *Baba is You*, it is destroyed, unless a rule specifies that the opened object "HAS" another object (in which case the opened object instead becomes the object that it "HAS"). In this design, such a rule is active only once every six turns, allowing only one in six babas or kekes to survive being opened.

Control Flow:

Typical *Baba is You* puzzles expect the rules of the level to be modified primarily by the playable character pushing text objects around the level, moving them into and out of sentences. While some in-game objects can be programmed to move autonomously, the level of control over them is insufficient to facilitate the sort of strategic text placement and displacement expected of a user.

Thus, instead of featuring movable text objects, this design lays out rows of incomplete rules that substitute one piece of text with the object the text describes, as in Figure 10. The design goes on to define as "WORD" any objects overlapping a bug object ( ) placed in the level. Thus, while the bug is on one of these objects, the rule applies because the object

Figure 10: An almost complete rule from the design. It spells "wind MAKE HAND", where the word "WIND" is replaced with a wind object.

can be used interchangeably with text to complete the rule (in Figure 10's case, as "wind MAKE HAND"). However, the rule deactivates as soon as the bug moves off of the object (the object will no longer be "WORD" and the game will go back to interpreting the rule as incomplete).

DOI: 10.36838/v5i6.23

[c] It is worth clarifying that in *Baba is You*, a stack of objects looks the same as a single object. The game knows how many objects are on each tile, but it is impossible to tell visually.

The result of this is that the bug can execute entire algorithms simply by moving along a straight path through preset arrangements of incomplete rules, as can be seen in Figure 11.

The bug is controlled by a bolt object ( ) which follows a similar process of completing rules using the WORD mechanic. The bolt can follow one of 24 tracks in the level – one for each of the six symbols Rogozhin's tape head can read in each of its four states.

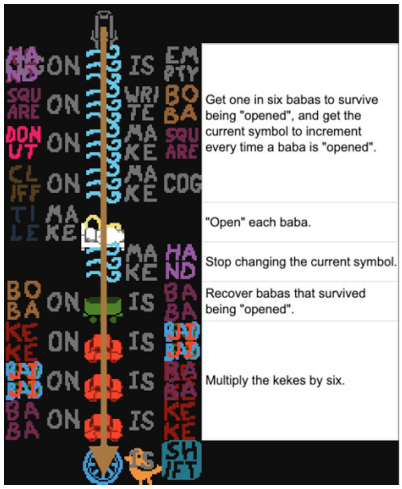

Figure 11: A bug's path through a set of incomplete rules (left) and a rough translation of what the rules would do when the bug activates them (right). A bug summoned to move through these carefully designed rules will induce specific object interactions that simulate shifting the tape to the left.

Changing State:

At the end of its current instruction, the bolt coordinates with the cog that stores the current symbol in order to create a specific rule that forms one bolt in the square at the beginning of the Turing machine's next instruction.

When the bolt gets to the end of its instruction, it will define objects representing the Turing machine's next state as part of a custom "GROUP". For instance, to switch into the q_{seed} state, it will activate the rule "seed IS GROUP".

Meanwhile, the cog that stores the tape head's current symbol (as shown in Figure 4) activates a rule that directs objects of said GROUP to spawn bolts if they overlap with objects that represent the tape head's current symbol. For instance, if the current symbol is 1 (represented by dust objects) the cog would form "GROUP ON dust MAKE BOLT"[d]. Continuing the previous example, this would mean that SEED ON DUST MAKE BOLT.

The result of such a rule is that the next bolt is created in squares containing both an object representing the Turing machine's next state and an object representing the Turing machine's current symbol. In this design, there will only ever be one such square in the level, and it will always be directly above the rules for the appropriate next instruction.

Halting:

A level is won in *Baba is You* when an object defined as YOU and an object defined as WIN both occupy the same tile. The design contains the rules BOLT ON FLAG IS YOU and BOLT ON FLAG IS WIN, so the level is won instantly (and the Turing machine stops) when the bolt moves onto a flag. The design contains these flags at the end of instructions in which the Turing machine is to halt.

Setup:

When the Turing machine is first loaded, there are neither babas on the cart nor kekes on the car, and the bolt is in a paused state. There is a playable "orb" that the user can control (i.e., "ORB IS YOU.") By moving it, the user can add any number of babas to the cart and any number of kekes to the car. The starting current symbol, however, is always assumed to be 1, as Rogozhin's machine expects to only start on a 1.

By controlling the orb, the user may also unpause the bolt and start the machine. Doing so destroys the playable orb so the user cannot tamper with the tape while the machine runs. This sets the machine into a completely deterministic state – the only legal move on all subsequent turns is to wait (and to let the autonomous parts of the level run their courses).

Reading the Tape:

The user may count how many babas are opened and how many kekes are opened and thus keep track of the entirety of the tape. If the user ignores the tape for a while and then starts reading at an arbitrary point in time, they can figure out the contents of the tape as soon as they watch long enough to see one left shift and one right shift occur.

The Full Construction:

Figure 12: The entire Turing machine level when it is first loaded.

Results and Discussion

Time-complexity of the Machine:

As explained in a previous section, writing symbols requires adding between zero and five babas or kekes to the stacks of objects. Adding zero babas or kekes is equivalent to leaving the stack unchanged, and will require zero turns, while adding between one and five babas will take between three and seven turns.

However, the slowest part of the machine's execution is performing the shift operations. Multiplying the babas or kekes by six will take a fixed twelve turns.

But dividing the babas or kekes by six will take one turn per baba or keke. In the best case, the symbols on the tape that the tape head moves towards read "1000…", which, using the base 6 conversion explained in a previous section, will be stored as 6^{n-1} babas or kekes, where n is the number of symbols (so opening all of them would take 6^{n-1} turns)[e].

In the worst case, the symbols on the tape that the tape head moves towards read "5555…", which will be stored as

DOI: 10.36838/v5i6.23

[d] The cog would accomplish this using a similar process to that of the bug and bolt: by letting the dust object in the middle of the sentence be used as a WORD so it can finish an otherwise incomplete rule.

[e] Of course, when the tape the tape head moves towards is completely blank (and thus stored as zero babas or kekes), opening them will take zero turns (and not the "one sixth of a turn" that the best-case model suggests).

$6^n - 1$ babas or kekes, where n is the number of symbols (so opening all of them would take $6^n - 1$ turns).

Thus, the number of turns to execute an instruction, where n is the number of symbols the tape head moves towards, can be expressed as:

(number from 0 to 7) + 12 + (number from $(6^n - 1)$ to $(6^n - 1)$)

When the input repeat delay is set to a minimum, holding the spacebar will allow turns to run about nine times per second, about as fast as a human tapping the spacebar repeatedly. As shown in Table 1, this combined with the exponential time-complexity of the shift operation has alarming implications for the practical efficiency of the machine.

On Baba is You Limits:

Table 1: The time-complexity of the machine.

Length of nonzero tape towards which tape head moves	Turns to execute instruction	Approximate time to execute instruction
0	12 to 19	1.33 to 2.11 seconds
2	18 to 54	2.00 to 6.00 seconds
4	228 to 1314	25.33 seconds to 2 minutes, 26 seconds
6	7,788 to 46,674	14 minutes, 25 seconds to 1 hour, 26 minutes
8	279,948 to 1,679,634	8 hours, 38 minutes to 2 days, 4 hours
10	1,007,708 to 60,466,194	12 days, 23 hours to 2 months, 17 days

The limit on stacking copies of the same object seems unusually low. Many limitations in *Baba is You* are very large and seem to be determined directly by memory and performance (meaning that they might increase if *Baba is You* was ported to more robust systems). Levels will end with a "Too Complex!" message if one of the following conditions occurs:

- There are more than 5,000 first words of rules in the level,
- There are more than 2,000 objects in the level,
- A single rule has more than 3,000 ways it could be read (because of combinatorics with overlapping text), or
- A "level" object from a custom world is transformed into more than 50 objects simultaneously.

Furthermore, there is no limit on overlapping objects of different types.

These generous restrictions allow things like Five Nights at Freddy's, Pac-Man, Tetris, Bloons Tower Defense, Donkey Kong, sliding puzzles, Bomberman, and even a variant of Super Mario Brothers to be created in *Baba is You*.[7-14] The advanced projects that push Baba is You to its literal limits show that the game was indeed willing to sacrifice performance to allow for creativity. These projects lag the game significantly but are still allowed by the constraints of *Baba is You*.

However, levels in Baba is You have a set maximum size of 33 spaces wide by 18 spaces tall. This limit is more restrictive – complex custom levels (including this one) almost always use maximum size levels and fill their entireties. Despite this, there are multiple reasons why the developer, Arvi Teikari, would enforce this limit. Perhaps they believed that vast levels (or even the very concept of scrolling) departed from the puzzle-based essence of *Baba is You*. Or maybe they did not want hubs containing many smaller areas to fit in single levels, instead wanting players to utilize the built-in "custom world" mechanic to hold small areas.

The limit on stacks of identical objects (currently 6) was likely set merely because Teikari did not see the potential of utilizing large stacks of identical objects. This may have been because stacks of the same object in *Baba is You* function identically to single objects in almost all cases except OPEN/SHUT. For instance, a stack of multiple babas will move as if it were one baba, push things as if it were one baba, and can be destroyed by a single object (e.g., one skull that IS DEFEAT) as if it were one baba. Moreover, in most *Baba is You* puzzles, working with separated objects is strictly advantageous over working with overlapping objects, especially if they are YOU.

However, allowing arbitrarily large stacks of identical objects (because it enables the full functionality of this design) allows *Baba is You* to support new logical and mathematical puzzles that would not otherwise work in a block-pushing game. For example, consider a custom world where each level asks the player to compute a number, say, the sum of the first 20 primes. The player may input a guess at the number into the level by creating that number of objects. The level is won if and only if the number inputted is equal to the number that the level wants (in this case, 77). The level checks each attempt by repeatedly dividing it by ten and comparing the remainders with hardcoded ones for the answer (in this case, seven and seven). This comparison process is hidden from the player's view with the HIDE property to prevent cheating. The player is instead supposed to solve the number puzzle by exiting the level, traversing the custom world to enter the universal Turing machine level, and calculating the answer there. After finding the answer to the computation, the player may go back to the number puzzle level and input the answer.

Indeed, removing the stacking limit allows for infinitely many significant new puzzles. The puzzles would have to already be solved beforehand, but of course, this could also be done using this universal Turing machine level. If the stacking limit were removed, any question solvable by computation whose answer is somehow expressible as a number could be turned into a playable, winnable *Baba is You* level that can be beaten without leaving the game. These puzzles could include graphing the Mandelbrot set to a certain precision, running von Neumann's universal constructor for a set number of turns, or even (once this is solved) attempting to find the winning first move in chess.

Admittedly, even calculating addition in this design would require excruciatingly tedious conversions – after a Turing machine is created, it would have to be converted into a 2-tag system, which would need to be formatted as an input for Rogozhin's machine. Any computation a user insisted on performing in *Baba is You* would require laying on the virtual starting tape an unreadable symbol monstrosity and then waiting cosmic amounts of time for single instructions to complete.

However, it is worth noting that Turing machines are themselves extremely inefficient compared to modern forms of computation. Contributions of this paper's nature are valued, if at all, because they demonstrate the expressive potential of

the Turing-complete systems and not because the operation of the simulated machines will cause paradigm shifts in how the systems are actually used.

Teikari learned of this design in August of 2022 and said it inspired them to consider removing the stacking limit if the *Baba is You* were ported to new platforms.[3]

The Unsolvability of Baba is You:

If the stacking limitation were removed, *Baba is You* would be computationally unsolvable. Any Turing machine could be simulated in this level, and a move (if it starts the Turing machine and destroys the playable orb character) could put the level into a state where it will be won if and only if the Turing machine halts. This proves that *Baba is You* without the stacking limitation is unsolvable: a computer that could decide with certainty whether a *Baba is You* move is winning would be able to solve the halting problem.

Comparison with Rodriguez's machine:

Rodriguez's design simulates the cellular automaton Rule 110, which was proven Turing complete in 2004 by Matthew Cook. Cook created a way for Rule 110 to simulate any "cyclic tag system," which in turn simulates a regular tag system.[15] Regular tag systems were proven Turing complete in 1961 by Marvin Minsky.[16]

Minsky's tag systems are linearly larger than the Turing machines they simulate, and Cook's cyclic tag systems are linearly larger than the regular tag systems they simulate. However, Cook's starting configurations of Rule 110 must contain two infinitely long patterns no matter which cyclic tag systems they simulate. If the patterns were to be abbreviated, the cyclic tag systems might stop before the Turing machines they are simulating have halted.

This creates a practical problem. Rodriguez proposes that the starting Rule 110 configurations for his level be created one object at a time, either in the level editor or by playing the level itself. The user must, therefore, manually create an infinite number of objects in the level before the Rule 110 simulation can even be started. Thus, even if *Baba is You* were, as Rodriguez proposes, extended to an infinite grid, his design as it is formulated is not only impractical but entirely impossible to run.

This design improves on Rodriguez's in one more way. In Rodriguez's design, the Turing machine has halted when a particular pattern appears in the level. The user may find it tedious to check large swaths of the level every turn for a fourteen-object-long pattern. In this design, on the other hand, it is readily apparent when the Turing machine has halted, because this wins the level (which makes the game play a sound and display "CONGRATULATIONS!" before returning the player to the custom world).

Conclusion

This paper has demonstrated that if a hardcoded stacking limit is removed from the game, a Turing complete custom level can be created in *Baba is You*, meaning that the game is capable of universal computation. It has found the proposed custom level vastly superior to a previous attempt at Turing completeness in the game. Furthermore, it has proposed theories about possibilities for unprecedented computation-based puzzle levels that the design unlocks. Moreover, it has proven that a game long known to be challenging for humans would be ultimately unsolvable by computers.

Acknowledgments

Many thanks to Alvin Shi for his guidance on the paper's structure, and to Dr. Jim Purbrick, Dr. Chaitanya Mishra, and Dr. Dan Grossman for reviewing this paper and providing key guidance on what to simplify and what to clarify. Thanks to Arvi "Hempuli" Teikari for reading a first draft of this paper and somewhat popularizing the machine.

References

1. Game Maker's Toolkit. How Baba Is You Makes Brain Busting P uzzles. https://www.youtube.com/watch?v=7zLwa4bztWs (access ed 2023-05-25).
2. Rodriguez, M. BABA IS TURING COMPLETE: A Sketch of a Proof (v2) https://www.twitlonger.com/show/n_1sqrh1m (access ed February 1, 2023).
3. You, B. I. And re: The stacking limit; ideally in a future reimpleme ntation of the engine stacked objects would be combined into a si ngle object with knowledge of the stack size instead of being delet ed, allowing for arbitrarily-sized stacks. it'd be a big under taking but it's in my mind [sic]
4. Suber, P. In The Paradox of Self-Amendment: *A Study of Law, Lo gic, Omnipotence, and Change*; Peter Lang Publishing, 1990; 362.
5. Rogozhin, Y. Small Universal Turing Machines. *Theoretical Comp uter Science*. [Online] **1996**, *168*, 215-240. Science Direct.https:// www.sciencedirect.com/science/article/pii/S0304397596000771 (accessed February 5, 2023).
6. Turing, A. On Computable Numbers, with an Application to the Entscheidungsproblem (1936). *The Essential Turing* **2004**, *244*. D OI:10.1093/oso/9780198250791.003.0005. Turing lays out his o wn universal Turing machine in the same paper where he introduc es regular Turing machines. It is very difficult to read, primarily be cause of his (by modern standards) unusual language and syntax u sed to describe them (e.g., he refers to as "m-configurations" what nearly all computer scientists now call "states").
7. Su, C. J.-Z. I Created the Entirety of Five Nights at Freddy's in Ba ba Is You. https://steamcommunity.com/app/736260/discussions /0/3372657079037273155/ (accessed 2023-05-25). Although the mechanics are ported faithfully (Bonnie and Chica move along th e correct paths, Freddy can be camera stunned, and blocking Foxy with the door drains power), the port of a point-and-click game i nto a tile-based puzzle game renders it very confusing, even to seri es veterans.
8. Germaphobe2. Pac-Man in Baba Is You (Custom Level). https:// www.youtube.com/watch?v=97BhtVA0E_0 (accessed 2023-05-2 5).
9. CavoX5. Baba Is You Improved Tetris. https://www.youtube.com/ watch?v=FEKD3MM5rsA (accessed 2023-05-25).
10. TeslaPenguin1. Baba Is Bloon. https://www.youtube.com/watch? v=g1reHDw_8rY (accessed 2023-05-25).
11. Semicolon, G. Donkey Kong Recreated in Baba Is You (Custom Level). https://www.youtube.com/watch?v=v1m26xUYeRY (acce ssed 2023-05-25).
12. Mu, M. [Baba Is You] New Adventures - Arcade A - Baba hour (449b). https://www.youtube.com/watch?v=YVIBMS0pgdk (acc essed 2023-05-25).
13. GincentVeez. Bomberman in Baba Is You. https://www.youtube. com/watch?v=IHOTyF7S_vg (accessed 2023-05-25).
14. Mu, M. [Baba Is You] New Adventures - Arcade G - Jump up, B abastar (Orb) (449b) https://www.youtube.com/watch?v=rbuJFr DS1kI (accessed 2023-05-25).

DOI: 10.36838/v5i6.23

15. Cook, M. A concrete view of Rule 110 computation. https://arxiv.org/abs/0906.3248 (accessed Feb 5, 2023).
16. Minsky, M. Recursive Unsolvability of Post's Problem of "Tag" and Other Topics in Theory of Turing Machines. The Annals of Mathematics 1961, 74 (3), 437–455. Minsky published a simpler second proof in 1964.

Author

Caleb Su is a senior at Mercer Island High School in Washington State. He is currently creating a working Python interpreter in Scratch and hopes to study computer science in college.