Baba Is Axioms: A Challenging Benchmark for Classical Planning with Axioms

Ronny Rochwerg, Allix Fletcher, Samantha Papais, Cynthia Wang, Connor Little, Ting Hu, Christian Muise

Queen's University
ronny.rochwerg@queensu.ca, 20ampf@queensu.ca, 21smp6@queensu.ca, cynthia.w@queensu.ca, connor.little@queensu.ca, ting.hu@queensu.ca, christian.muise@queensu.ca

Abstract

We introduce a new benchmark planning domain based on the puzzle game *Baba Is You*. As in Sokoban, a popular classical planning domain, the agent operates in a fully observable, deterministic grid world and can move in the cardinal directions while pushing objects. However, unlike Sokoban, the properties governing objects in a level are determined by sentences inside the level that the player can manipulate: *the state of the world impacts the fundamental dynamics of the environment*. This introduces a meta-planning aspect in which the planner must reason not only about object states but also about dynamically reconfigurable rules that result from object locations. Aside from two primitive actions, the behaviour of objects, rules, and movement is modelled using a complex web of derived predicates. Finally, we evaluate the Fast Downward domain independent classical planning system with multiple search configurations on a suite of hand-crafted instances that vary in level size and the number of objects. Our results show that while current planners can solve tiny or sparsely populated levels, their performance degrades sharply as the number of objects or the level size increases. The *Baba Is You* setting requires thinking outside the box and serves as a highly compelling benchmark for classical planning with axioms. Our results highlight the difficulty of solving classical domains in the presence of complex interacting axioms and the need for improved planners that support them.

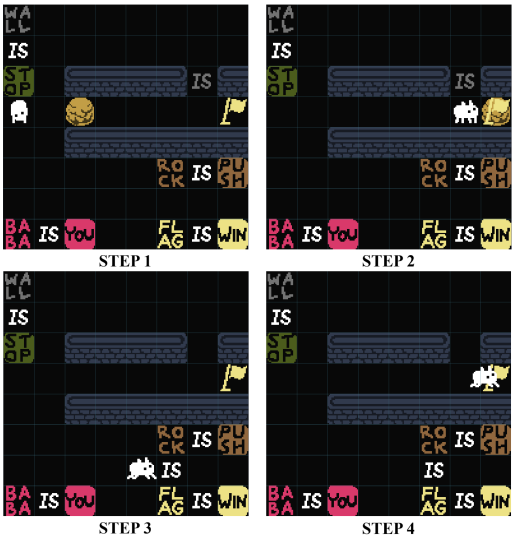

Figure 1: The key steps in solving one of the custom benchmark levels (small05). Step 1 shows the initial level setup. Step 2 shows pushing the rock to the right to gain access to the `IS` text block. Step 3 shows how to create the sentence `ROCK IS FLAG`. Step 4 shows solving the level.

1 Introduction

Planning can be used to model problems in many sectors, ranging from agriculture and aeronautics to manufacturing and design (Ghallab, Nau, and Traverso 2004). In classical planning, automated planners search for a valid series of deterministic actions to achieve the problem's goal state. There are several extensions of classical planning to strengthen its modelling capabilities, such as non-determinism, temporal actions, and partial observability. In this paper, we focus on derived predicates, or axioms, an extension of classical planning in which predicates are deduced from the world state rather than set as effects in action definitions. They allow for a more compact definition of complex behaviour. This facilitates a broader range of automated planning applications by enabling the modelling of more sophisticated problem dynamics in a more manageable form. In practice, axioms allow recursive definitions which are significantly more difficult, if not impossible, to do with regular actions (Thiébaux, Hoffmann, and Nebel 2005). However, automated planners have varying degrees of success in handling and solving problems with axioms.

To assess planner capability, researchers turn to common benchmark sets. Game settings are ideal benchmarks for automated planning due to their structured world states and well-defined goals. A relevant game setting that has been adapted into a planning problem is *Sokoban*, which involves an agent pushing boxes to their goal positions in a grid world. *Sokoban* was shown to be PSPACE-complete (Culberson 1997) and has been used extensively as a baseline for classical planning (Demaret, Van Lishout, and Gribomont 2008).

In this paper, we present a model of the game *Baba Is You* in the Planning Domain Definition Language (PDDL). *Baba Is You* is a game similar to *Sokoban* as it has the same small action space. A modified version of *Baba Is You* has been shown to be undecidable (Geller 2023). While the exact

computational complexity of our simplified version of the game is unknown, we conjecture that it is PSPACE.

Although the game has a basic action space, it features significant modelling complexity as the rules governing a level can change dynamically. Figure 1 shows an example level being solved by changing the properties of the objects in the level. These dynamic changes cause axioms to be a fundamental part of modelling the rules of a level and give us a compelling benchmark for classical planning with axioms.

2 Related Works

Several artificial intelligence-based strategies have been employed to solve *Baba Is You* levels. van Wetten et al. found that state-of-the-art large language models (LLMs) are not able to reason about the dynamic rule changes within the setting of the game and thus perform poorly when tasked with producing solutions (van Wetten, Plaat, and van Duijn 2025). Jens created a formalism to mathematically represent generalization through systematic compositionality and used *Baba Is You* as a benchmark to test its effectiveness (Jens 2023). Systematic compositionality is when two separate ideas are combined to make one complex expression—for example, combining the words "red" and "apple" to make "red apple." This is an important aspect of rule creation in *Baba Is You*; however, it is difficult to create AI agents that can use this principle to reason about complex notions effectively. Jens used reinforcement learning and supervised learning techniques to test their formalism and found that they did not perform well on the *Baba Is You* benchmark. Additionally, various works have explored improving the performance of domain-specific agents: Olson et al. used evolutionary optimization techniques to create better-guided heuristics for *Baba Is You*-specific agents to use when solving levels (Olson, Wagner, and Dockhorn 2023); and Charity and Togelius evaluated four *Baba Is You*-specific agents—a breadth-first search agent, a depth-first search agent, a random sequence agent, and the default Baba is Y'all agent (Charity, Khalifa, and Togelius 2020)—against levels of increasing complexity to varying degrees of success (Charity and Togelius 2022).

While no prior work applies automated planning directly to *Baba Is You*, related puzzle games have been explored. The classic benchmark *Sokoban*, in which an agent pushes boxes to goal locations on a grid, can be viewed as a simplified reduction of *Baba Is You*. Despite its PSPACE-complete complexity (Culberson 1997), planners have become increasingly effective at solving *Sokoban* over the years. For example, Botea et al. present an automated planning approach *Sokoban* that abstracts the game into rooms and tunnels, making the domain less cumbersome to explore than with previously employed heuristic search methods (Botea, Müller, and Schaeffer 2002). Espasa et al. used PDDL to create benchmarks of the games *Plotting* (Espasa et al. 2023) and *Puzznic* (Espasa et al. 2024), both of which involve reasoning about moving blocks around on a grid. Bofill et al. presented their work on modelling the puzzle game *A Good Snowman is Hard to Build* in both PDDL and as a SAT problem (Bofill et al. 2023). Similar to *Baba Is You* and *Sokoban*, the game involves moving objects around a discrete grid in order to meet some criteria; in this case, the player moves snowballs around to make snowmen. The author created three PDDL models — one with axioms and two without — and found that using axioms did not significantly improve solve time, and that the SAT encoding outperformed both PDDL models. However, Ivankovic and Haslum applied axioms to the *Sokoban* domain and found that this improved the planners' search efficiency (Ivankovic and Haslum 2015).

3 Background

3.1 Classical Planning

Classical planning concerns the problem of generating a sequence of actions that transforms a given initial state into a desired goal state. This formulation assumes a deterministic, fully observable, and static environment, in which the effects of each action are known and unambiguous. Under these assumptions, a planner searches over the space of possible action sequences to identify one that achieves the goal.

We model our problem using PDDL (Haslum et al. 2019), a standard formalism for expressing planning domains and problem instances. PDDL supports the specification of predicates, actions, and states. In addition to standard predicates, our domain makes use of derived predicates (also called axioms) (Thiébaux, Hoffmann, and Nebel 2005). Derived predicates are logical conditions whose truth values are inferred from other predicates according to a set of rules, rather than being modified directly by actions. While they may appear in action preconditions, they cannot be used as action effects. Their capacity for recursive definition makes them particularly useful for modelling the mechanics of *Baba Is You*.

Given a domain and problem specification, a planner reasons over the grounded representation to produce a plan that transforms the initial state into one satisfying the goal conditions. This framework enables us to model puzzle configurations in *Baba Is You* as classical planning problems and to investigate how well planners can solve such levels.

3.2 Baba Is You

Baba Is You is a rule-manipulation puzzle game in which the player moves one or more objects around a grid containing both object sprites and textual rule blocks. Text blocks represent nouns (e.g., `BABA`, `FLAG`, `WALL`, `ROCK`), operators (e.g., `IS`), and properties (e.g., `YOU`, `WIN`, `PUSH`, `STOP`). When arranged into valid sequences, these blocks form sentences that define the rules governing the level. Sentences are parsed either left-to-right or top-to-bottom, and if a sentence is disrupted by moving one of its constituent blocks, the corresponding rule ceases to apply. Text blocks are always pushable, allowing the player to alter the ruleset dynamically during play.

Rules in *Baba Is You* take two primary forms. A **noun-operator-noun** sentence (e.g., `ROCK IS WALL`) transforms all sprites associated with the first nouns into sprites of the second. A **noun-operator-property** sentence (e.g.,

ROCK IS PUSH) assigns the specified property to all instances of that noun. Objects with PUSH can be pushed by the player, objects with STOP block movement, and objects with YOU become controllable agents. The player's actions consist of moving up, down, left, or right. Each action simultaneously moves all sprites possessing the YOU property one grid cell in the chosen direction, if possible, and applies the PUSH and STOP mechanics accordingly.

A level is won when an object with the YOU property also has the WIN property, or when a YOU object occupies the same cell as an object with the WIN property.

4 Baba is PDDL

We crafted the *Baba Is You* domain in PDDL 2.2 using: typing, derived-predicates, negative-preconditions, disjunctive-preconditions, conditional-effects, and equality. For the typing, at the top level we distinguish `location`, `direction`, and `thing`. The type `thing` represents objects found inside a level and is refined into `text`, for tiles that represent words, and `sprite` for any objects that are not text (e.g., instances of Baba, walls, rocks, and flags). Text tiles are further partitioned into `noun`, `property`, and `operator`, mirroring the categories of the original game's rule sentences. The domain introduces constants for the four cardinal directions (`north`, `south`, `east`, `west`), and for all the text objects: `BABA`, `WALL`, `FLAG`, `ROCK`, `PUSH`, `STOP`, `YOU`, `WIN`, and the binary operator `IS`. These constants allow us to reference attributes directly in the domain, resulting in more concise modelling.

The rest of the domain is organized around a small set of basic predicates (predicates that are not the head of an axiom) and a set of derived predicates. The basic predicates `at(?t, ?l)`, `is-type(?t, ?n)`, and `connected(?from, ?to, ?dir)` describe the object positions, type tags, and underlying grid, respectively. All of the game mechanics are then computed from these using derived predicates: `blocked(?l, ?dir)`, `open(?l, ?dir)`, `will-move(?t, ?from, ?to, ?dir)`, `has-prop(?n, ?p)`, `conversion(?n1, ?n2)`, `check-conversion()`, and the goal predicate `won()`. In the predicates, the parameters: `?t` represents `thing`, `?l`, `?to`, `?from` all represent `location`, `?dir` represents `direction`, `?n` represents `noun`, and `?p` represents `property`.

A central component of the movement semantics is the pair of derived predicates `blocked(?l, ?dir)` and `open(?l, ?dir)`. Intuitively, `blocked` captures when a location `?l` cannot be entered from a given direction `?dir`. This could be due to an object with the property `STOP` being in the location, in which case it cannot be entered from any direction; or by an object with `PUSH` preventing entering from a specific direction because it cannot be pushed into the following grid location. Two axioms derive the `blocked` predicate, one covers the base cases, and the other is recursive. The recursive derivation allows for chains of pushable objects (an object with the property `PUSH`) to easily indicate that they are all blocked from a direction due to the final object not being able to enter the next location. The companion predicate `open(?l, ?dir)` is defined simply as the negation of `blocked(?l, ?dir)`, and it identifies locations into which a sprite can safely move or be pushed in direction `?dir`. This is used for readability and could be replaced with `(not (blocked(?l, ?dir)))`

Listing 1: The axioms for the `will-move` predicate

```
(:derived (will-move ?t - thing ?from ?to - location ?dir - direction)
    (exists (?n - noun)
        (and
            (at ?t ?from)
            (is-type ?t ?n)
            (has-prop ?n you)
            (connected ?from ?to ?dir)
            (open ?to ?dir)
        )
    )
)

(:derived (will-move ?t - thing ?from ?to - location ?dir - direction)
    (exists (?n - noun ?l3 - location ?t2 - thing)
        (and
            (at ?t ?from)
            (is-type ?t ?n)
            (has-prop ?n push)
            (connected ?from ?to ?dir)
            (connected ?l3 ?from ?dir)
            (will-move ?t2 ?l3 ?from ?dir)
        )
    )
)
```

The derived predicate `will-move(?t, ?from, ?to, ?dir)` determines which objects will actually move on a given step when the player chooses a direction. Similar to the `blocked` predicate, there are two axioms that derive `will-move`, they are shown in Listing 1. The first covers any `thing` whose noun currently has the `YOU` property, if the grid cell in the given direction is `open` then it `will-move` in that direction. A second, recursive axiom, propagates movement through chains of pushable objects: if a pushable object is in a grid cell which another object `will-move` into, it too is given the `will-move` predicate. Together with `blocked` and `open`, this definition ensures that entire chains of pushable sprites move coherently in a single action step only when there is sufficient space in front of them, and remain stationary otherwise.

The link between the rule sentences on the board and the rest of the mechanics is captured by two families of derived predicates: `has-prop(?n, ?p)` and `conversion(?n1, ?n2)`. The predicate `has-prop` encapsulates all currently active **noun-operator-property** rules. It is derived by three axioms, one encodes a static rule that all text blocks are always pushable since this is an underlying rule that is always true. The remaining axioms detect rule sentences laid out in the grid either vertically or horizon-

tally as text tiles. The derived `conversion(?n1, ?n2)` predicate summarizes all active **noun-operator-noun** rules in the same way. Because derived predicates are recomputed after every action, changing the arrangement of text through movement directly changes which nouns have which properties and which conversions are active.

Lastly, the domain has two actions: `move ?dir` and `convert`. The `move ?dir` action corresponds to the player choosing a direction `?dir` to move in. Its effect moves every `thing` that satisfies the `will-move` predicate in that direction, and it also sets the bookkeeping predicate `check-conversion` to true so that potential noun conversions are evaluated after each movement. The `convert` action applies any active conversions and must be executed if a `conversion(?n1, ?n2)` predicate exists and there is a sprite of type `?n1`. The conversion action is needed since if the conversion effect was placed in the movement action, it would not convert nouns on the same movement that creates a `conversion(?n1, ?n2)` sentence.

It should be noted that while axioms are not necessary for creating a model of *Baba Is You*, they are necessary for creating an effective model. Without axioms, forced actions using predicate gates would have to be used to check for rule changes after text is moved. For reference, the full domain PDDL file is available at https://github.com/ronrochwerg/Baba-Is-Axioms.

5 Experiments and Results

To evaluate our domain, we constructed a benchmark suite of 23 custom levels. This set consists of 10 *small* problems on an 8×8 grid, 10 *large* problems on a 10×10 grid, and 3 *hard* problems on an 8×8 grid. These problem files and images of their starting configurations are available at https://github.com/ronrochwerg/Baba-Is-Axioms. We also include level codes for each problem, which can be used in the game to play the custom levels.

All of the solutions found were manually verified with the game, and hand-crafted solutions to every one of the 23 problems were validated with the PDDL. Interestingly, VAL (Howey, Long, and Fox 2004) was unable to process the axioms within a reasonable amount of time, but a configuration of INVAL[1] was up to the task.

The small problems are ordered by the authors' subjective difficulty, with each level introducing a new aspect of the game (movement, sentence creation, noun changing, multiple controlled objects, etc.). Each large problem is a scaled-up version of the corresponding small problem, preserving the general solution strategy while increasing spatial complexity. The hard problems are three levels that were judged to require substantially more reasoning than the others.

We evaluated three search algorithms from Fast Downward[2], A* (eager), eager greedy, and lazy greedy search, in combination with four standard heuristics: *blind*, *hmax*, *add*, and *ff*. The handling of axioms for each heuristic was set to $approximate\_negative$, as $approximate\_negative\_cycles$ failed on all problems. Each planner configuration was run on all 23 problems with a computer running Ubuntu 24.04 with an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz CPU and 128 GB of RAM. Each run had a limit of 30 minutes for translation and 30 minutes for searching, plus 8GB of memory. Table 1 shows the coverage for each combination of search and heuristic. All problems that were not solved were due to a timeout during search time. A more detailed table of results for each individual level is available at https://github.com/ronrochwerg/Baba-Is-Axioms.

Heuristic	A*	Eager Greedy	Lazy Greedy
blind	13	13	13
hmax	15	15	14
add	**16**	13	13
ff	13	13	14

Table 1: Coverage and failure statistics for each search algorithm and heuristic over 23 problems.

Overall, A* with the *add* heuristic achieves the best coverage, solving 16 out of the 23 problems, followed closely by A* with *hmax*. Even the best-performing configuration times out on just under a third of instances, indicating that the combination of dynamic rules and axiom-heavy semantics creates challenging search spaces. It is somewhat surprising that the best configuration was an inadmissible heuristic used within A*. We conjecture that this is due to the $g-$value providing a meaningful anchor to the partial plans, with the heuristics being largely ineffective in this domain.

We also attempted to run several other state-of-the-art planners and search configurations, including LAMA (Richter and Westphal 2010), AxSAT (Behnke, Speck, and Gnad 2025), SymK (Speck, Seipp, and Torralba 2025), and BFWS (Lipovetzky and Geffner 2017), but none of these were able to solve even the simplest test problems due to timing out or running out of memory during translation. This suggests that the axiomatic *Baba Is You* domain is a demanding benchmark for general-purpose planners.

6 Conclusion

Baba Is You is a puzzle game that contains over 200 levels and more than 250 distinct text objects. In this paper, we present a new benchmark for classical planning with axioms by modelling the mechanics of the game and the objects in the first level of the game. We evaluated the model with two search algorithms and various heuristics. Even with the limited scope of our model, the planners struggled to provide solutions for some of the more difficult levels, timing out before a plan was found. Overall, our results indicate that planners are not currently able to handle the complexities of advanced axiom use, but can indeed start to solve one of the most complex block-pushing puzzles in circulation.

[1] https://github.com/patrikhaslum/INVAL

[2] https://www.fast-downward.org

Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). This research is also supported by the Vector Scholarship in Artificial Intelligence, provided through the Vector Institute, as well as the Ontario Graduate Scholarship (OGS).

References

Behnke, G.; Speck, D.; and Gnad, D. 2025. AxSAT - Bringing Axioms to SAT Planning. In Casini, G.; Dundua, B.; and Kutsia, T., eds., *Logics in Artificial Intelligence - 19th European Conference, JELIA 2025, Kutaisi, Georgia, September 1-4, 2025, Proceedings, Part II*, volume 16094 of *Lecture Notes in Computer Science*, 77–93. Springer.

Bofill, M.; Borralleras, C.; Espasa, J.; Martín, G.; Patow, G.; and Villaret, M. 2023. A Good Snowman is Hard to Plan. *CoRR*, abs/2310.01471.

Botea, A.; Müller, M.; and Schaeffer, J. 2002. Using Abstraction for Planning in Sokoban. In Schaeffer, J.; Müller, M.; and Björnsson, Y., eds., *Computers and Games, Third International Conference, CG 2002, Edmonton, Canada, July 25-27, 2002, Revised Papers*, volume 2883 of *Lecture Notes in Computer Science*, 360–375. Springer.

Charity, M.; Khalifa, A.; and Togelius, J. 2020. Baba is Y'all: Collaborative Mixed-Initiative Level Design. In *IEEE Conference on Games, CoG 2020, Osaka, Japan, August 24-27, 2020*, 542–549. IEEE.

Charity, M.; and Togelius, J. 2022. Keke AI Competition: Solving puzzle levels in a dynamically changing mechanic space. In *IEEE Conference on Games, CoG 2022, Beijing, China, August 21-24, 2022*, 570–575. IEEE.

Culberson, J. C. 1997. Sokoban is PSPACE-complete.

Demaret, J.-N.; Van Lishout, F.; and Gribomont, P. 2008. Hierarchical planning and learning for automatic solving of sokoban problems. *Belgian/Netherlands Artificial Intelligence Conference*, 57–64.

Espasa, J.; Gent, I. P.; Miguel, I.; Nightingale, P.; Salamon, A. Z.; and Villaret, M. 2024. Cross-Paradigm Modelling: A Study of Puzznic. In *36th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2024, Herndon, VA, USA, October 28-30, 2024*, 89–95. IEEE.

Espasa, J.; Miguel, I.; Nightingale, P.; Salamon, A. Z.; and Villaret, M. 2023. Challenges in Modelling and Solving Plotting with PDDL. *CoRR*, abs/2310.01470.

Geller, J. 2023. Baba is You is Undecidable. In *IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023*, 1–8. IEEE.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier. ISBN 978-1-55860-856-6.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. ISBN 978-3-031-00456-8.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15-17 November 2004, Boca Raton, FL, USA*, 294–301. IEEE Computer Society.

Ivankovic, F.; and Haslum, P. 2015. Optimal Planning with Axioms. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 1580–1586. AAAI Press.

Jens, M. 2023. *Baba is AI: A Grounded Benchmark for Compositional Generalization in Dynamic Rule Systems*. Ph.D. thesis, Massachusetts Institute of Technology.

Lipovetzky, N.; and Geffner, H. 2017. A Polynomial Planning Algorithm that Beats LAMA and FF. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, 195–199. AAAI Press.

Olson, C.; Wagner, L.; and Dockhorn, A. 2023. Evolutionary Optimization of Baba Is You Agents. In *IEEE Congress on Evolutionary Computation, CEC 2023, Chicago, IL, USA, July 1-5, 2023*, 1–8. IEEE.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.*, 39: 127–177.

Speck, D.; Seipp, J.; and Torralba, Á. 2025. Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions. *J. Artif. Intell. Res.*, 82.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *Artif. Intell.*, 168(1-2): 38–69.

van Wetten, F.; Plaat, A.; and van Duijn, M. J. 2025. Baba Is LLM: Reasoning in a Game with Dynamic Rules. In Marcelloni, F.; Madani, K.; van Stein, N.; and Filipe, J., eds., *Computational Intelligence - 17th International Joint Conference, IJCCI 2025, Marbella, Spain, October 22-24, 2025, Proceedings, Part I*, volume 2827 of *Communications in Computer and Information Science*, 60–77. Springer.