Designing Zero-Knowledge Proofs for Agent Motion Planning

Agent Motion Planning

The agent interacts with adjacent cells according to a set of "replacement rules." For example, in the game Push-1F (Figure 1), the agent (gray) can push movable (blue) blocks but not fixed (brown) blocks. The goal of agent motion planning games is to get the agent to a particular location.

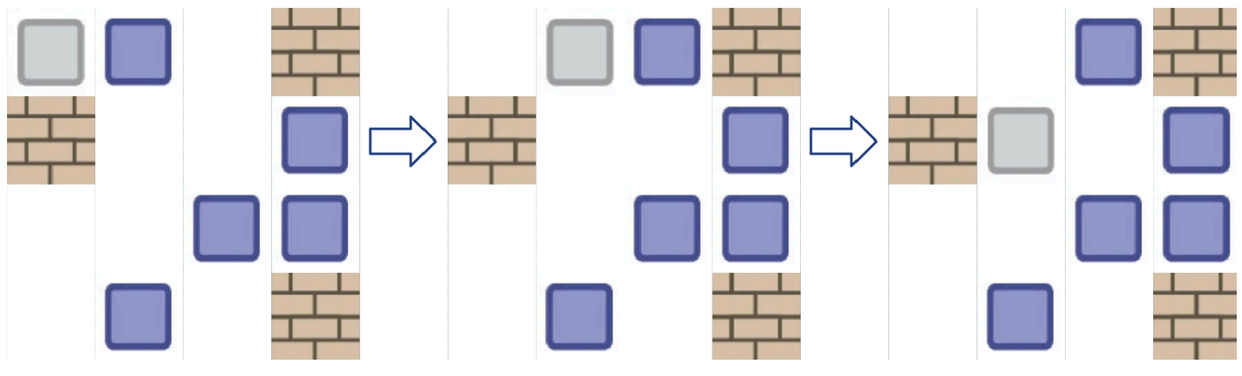


Figure 1: Push-1F.
Original images from "Agent Motion Planning as Block Asynchronous Cellular Automata: Pushing, Pulling, Suplexing, and More," by MIT Hardness Group, 2024, Unconventional Computation and Natural Computation, 2024, p. 219-236.
Adapted by the finalist using SVG Tiler, 2025.

Zero-Knowledge Proofs

A zero-knowledge proof is a cryptographic protocol between two entities, a prover and a verifier, in which the prover can prove to the verifier that she knows a solution to a problem, without revealing what that solution is. For example, the red card proof (Figure 3) allows one to show that the card on the right is red, without revealing which red card it is.

Figure 3: The red card proof.
Image taken of a deck of Bicycle playing cards by finalist, 2025.

Adjacency Check

To ensure that the agent interacts with grid cells adjacent to itself, we take from the card grid the cells up, left, down, and right from the agent (Figure 4 left) and put them in a separate grid (Figure 4 right). Then, we secretly remove the column associated with the desired direction of movement and place it separately to be used in the Permutation Check. Since these cards were taken from the grid of Figure 4 right, we know that the positioning of the move is legal, but we don't know which direction the move is made in.

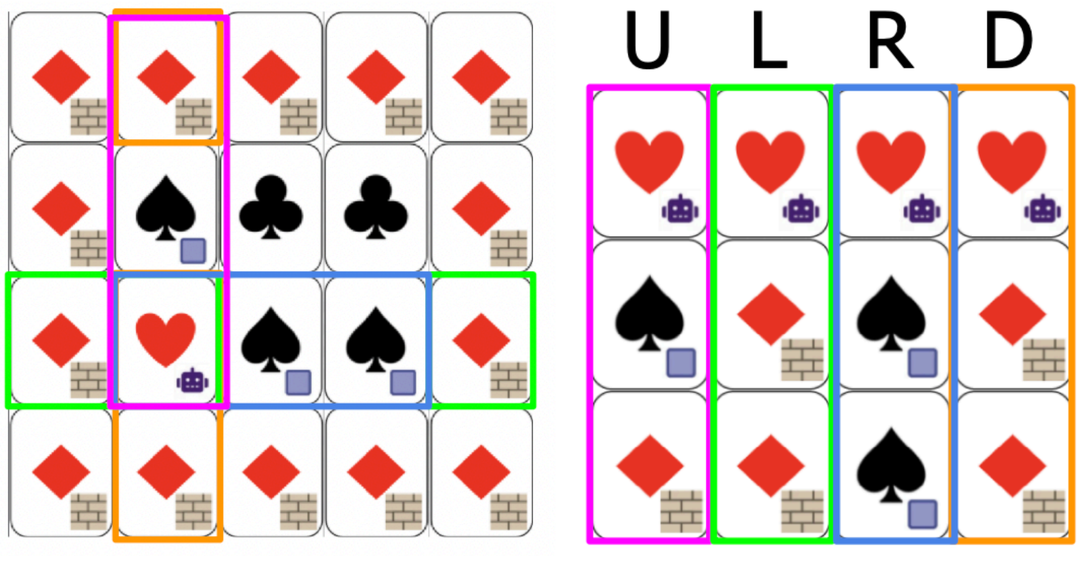


Figure 4: Adjacency Check.
Original images from "Agent Motion Planning as Block Asynchronous Cellular Automata: Pushing, Pulling, Suplexing, and More," by MIT Hardness Group, 2024, Unconventional Computation and Natural Computation, 2024, p. 219-236.
Adapted by the finalist using SVG Tiler and Google Slides, 2025-2026.

Permutation Check

The column removed from the Adjacency Check matrix is the initial state of the group of cells involved in the move (Figure 7 left). In Push Merge, moves involve replacing an initial state with a final state. In the Permutation Check, we check that the initial state outputted by the Adjacency Check is of a legal form. In Push Merge, there are four legal initial states. If the output of the Adjacency Check is a legal initial state, the prover can make three other columns comprising the three other legal initial states (Figure 7 right). Then, when this matrix is shuffled, it can be turned face-up to reveal a scrambled set of all four legal forms. So, we know that the move made is legal, but we don't know which type of move was made.

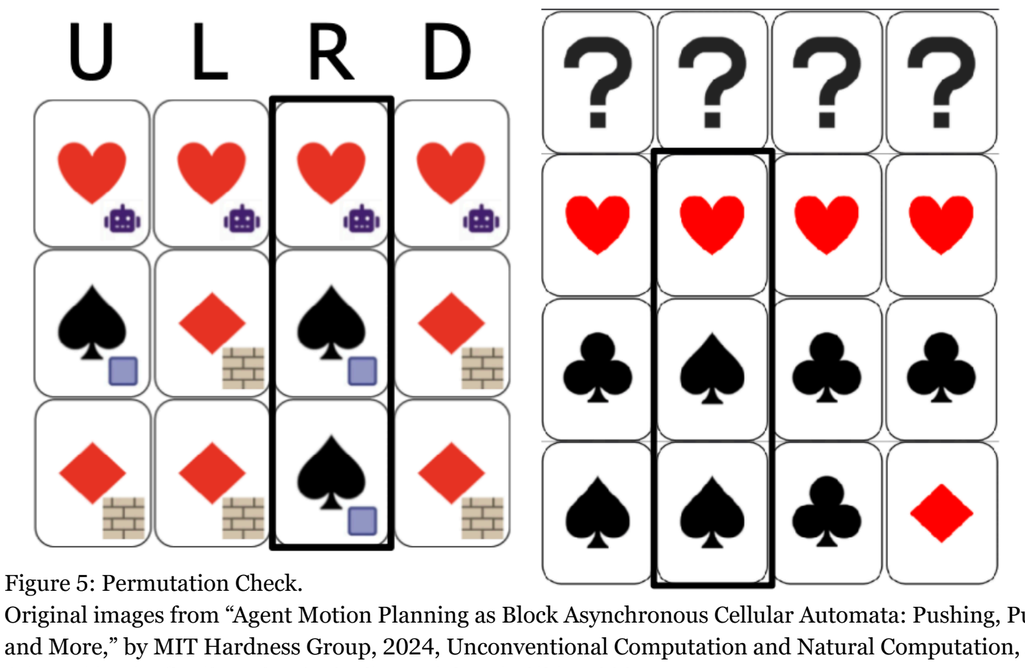
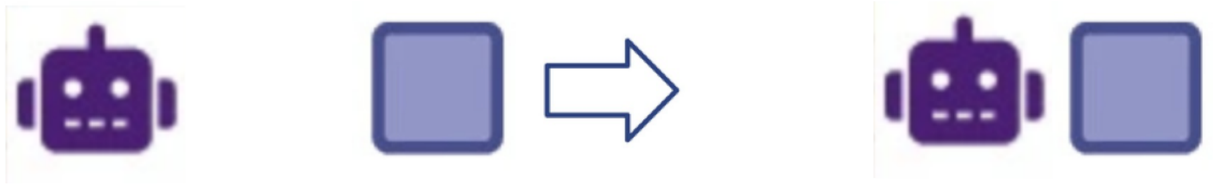


Figure 5: Permutation Check.
Original images from "Agent Motion Planning as Block Asynchronous Cellular Automata: Pushing, Pulling, Suplexing, and More," by MIT Hardness Group, 2024, Unconventional Computation and Natural Computation, 2024, p. 219-236.
Adapted by the finalist using SVG Tiler and Google Slides, 2025.

Push Merge is a specific agent motion planning game that we will use as a running example. Push Merge allows two types of moves, which can be made in four directions (up, down, left, right):

(1) Moving through empty space



(2) Merging two adjacent blocks

Figure 2: Push Merge.
Original images from "Agent Motion Planning as Block Asynchronous Cellular Automata: Pushing, Pulling, Suplexing, and More," by MIT Hardness Group, 2024, Unconventional Computation and Natural Computation, 2024, p. 219-236.
Adapted by the finalist using SVG Tiler, 2025.

A Zero-Knowledge Proof for Push Merge

To prove that we know a solution sequence of moves to a Push Merge game, for each individual move, we prove that it is legal with respect to the previous Push Merge grid state, while revealing nothing about the move beyond its legality.

(1) **Chosen Pile Cut** ([1]): allows the prover to isolate the cells relevant to the move without revealing the positions of those cells.

(2) **Adjacency Check** (original): allows the prover to isolate the relevant direction of the move, without revealing which direction that is.

(3) **Permutation Check** (original): allows the prover to prove that the move is of a legal form, without revealing which form.

References

- Goldreich, O., Micali, S., and Widgerson, A. (1991). Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. Journal of the ACM (JACM), 38(3), 690-728.
- Koch, A., Walzer, S. (2021). Foundations for Actively Secure Card-Based Cryptography. 10th International Conference on Fun with Algorithms, FUN 2021. https://doi.org/10.5445/IR/1000125391
- MIT Hardness Group (2024). Agent Motion Planning as Block Asynchronous Cellular Automata: Pushing, Pulling, Suplexing, and More. Unconventional Computation and Natural Computation, 2024, 219-236. https://doi.org/10.1007/978-3-031-63742-1_16
- Tamura, Y., Suzuki, A., Mizuki, T. (2024). Card-Based Zero-Knowledge Proof Protocols for the 15-puzzle and the Token Swapping Problem. APKC '24: Proceedings of the 11th ACM Asia Public-Key Cryptography Workshop, 11-22. https://doi.org/10.1145/3659467.3659905
- Poster created by the finalist using Canva, 2026.