Unsimulability, Universality, and Undecidability in the Gizmo Framework

by

Joshua Ani

S.B., Computer Science and Engineering
Masssachusetts Institute of Technology, 2021

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

© Massachusetts Institute of Technology 2023. All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
January 20, 2023

Certified by . . . .
Erik D. Demaine
Professor
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Unsimulability, Universality, and Undecidability in the Gizmo Framework

by

Joshua Ani

Submitted to the Department of Electrical Engineering and Computer Science on January 20, 2023, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

The gizmo framework is a recent development of the gadget framework used for proving computational complexity results of videogames and other motion planning problems. This thesis explores three aspects of the gizmo framework: unsimulability (the inability of one gizmo to simulate another gizmo), universality (the ability of a gizmo to simulate all gizmos in its simulability class), and undecidability (the inability to decide whether a maze made of a gizmo is solvable). We give a proof that the 1-toggle cannot simulate the 2-toggle, as it contains important techniques. We explore a class of gizmos called dicrumbler variants, and give partial results for which ones simulate which others. We give universal gizmos for simulability classes **Reg** and **DAG**, and explore the concept of finding all the gizmos that simulate a particular gizmo, with partial results given for the dicrumbler. We show that reachability for a gizmo representing a counter in a counter machine is undecidable, and show several gizmo simulations. We give a proof that generalized New Super Mario Bros. is undecidable using one of the undecidable gizmos.

Thesis Supervisor: Erik D. Demaine
Title: Professor

Acknowledgments

I would like to thank Erik Demaine for the Fun with Hardness Proofs class he taught in Spring 2019, which inspired me to work on the gadget framework. Much of the work on the framework was done with Dylan Hendrickson, Jayson Lynch, and Yevhenii Diomidov, with Dylan in particular introducing the gizmo framework.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Gadgets

The gadget framework for motion planning was first introduced in [3] as a way to simplify hardness proofs for videogames. In this framework, a gadget is a set of locations and states along with traversals between locations that are allowed on specific states. These gadgets can be copied and have their locations connected together to form a maze, where the goal is to get from some start location to some end location, and for certain gadgets, this problem is PSPACE-hard.

More formally, a *gadget* is a tuple (L, S, T) where L is a set of locations, S is a set of states, and T is a set of tuples $(s_0, \ell_0, s_1, \ell_1) \in S \times L \times S \times L$, notated as $(s_0, \ell_0) \to (s_1, \ell_1)$ with the intention that if the gadget is in state s_0, an agent can traverse from ℓ_0 to ℓ_1 and set the state to s_1. An example of a gadget is shown in Figure 1-1, and another example is shown in Figure 1-2.

These two particular examples were shown in [3], and reachability in a maze with them was shown to be PSPACE-hard. Reducing from TQBF for each gadget is tedious, so instead Demaine et al. used gadget simulations. If a gadget G can

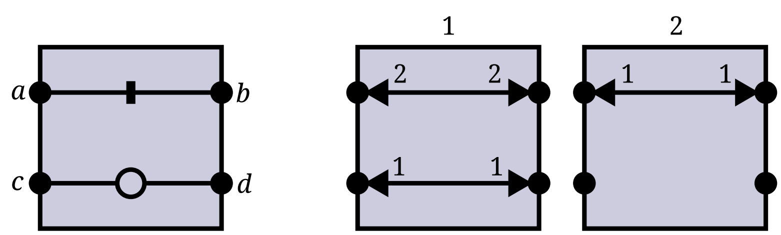


Figure 1-1: The tripwire lock gadget from [3]. The notation for state 1 is shown on the left, and the state diagram on the right. The tripwire (tunnel connecting a and b) switches whether the lock (tunnel connecting c and d) is open whenever it is crossed. The lock does not change the gadget's state. $L = \{a, b, c, d\}$, $S = \{1, 2\}$, and $T = \{(1, a, 2, b), (1, b, 2, a), (2, a, 1, b), (2, b, 1, a), (1, c, 1, d), (1, d, 1, c)\}$

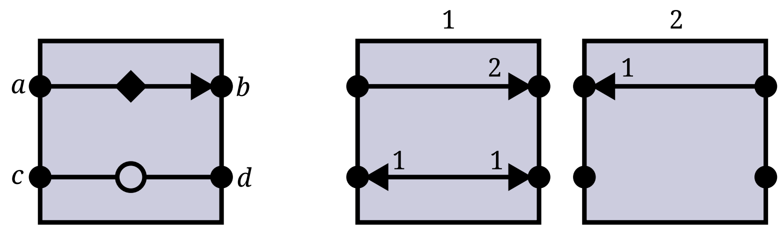


Figure 1-2: The toggle lock gadget from [3]. Similar to the tripwire lock, but the tunnel connecting a and b is a toggle and is only traversable in one direction, switching direction whenever it is traversed. $L = \{a, b, c, d\}$, $S = \{1, 2\}$, and $T = \{(1, a, 2, b), (2, b, 1, a), (1, c, 1, d), (1, d, 1, c)\}$

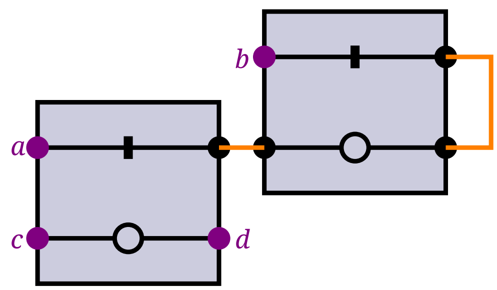


Figure 1-3: A simulation of the toggle lock with the tripwire lock. Orange lines connect locations between gadgets, and purple locations are locations of the simulated toggle lock. An agent can traverse $a \to b$, closing the locks of both gadgets. Afterward, an agent can traverse $b \to a$, opening the locks of both gadgets. The lock of the gadget on the right prevents $b \to a$ initially and prevents $a \to b$ after $a \to b$.

simulate a gadget H, then reachability with H can be reduced to reachability with G by replacing each copy of H with its simulation with G. For example, the tripwire lock can simulate the toggle lock, as shown in Figure 1-3.

Since the introductory paper, many other results have been shown in the gadget framework. Demaine et al. showed that all 2-state 2-tunnel reversible deterministic gadgets simulate each other and reachability with each of them is PSPACE-complete, even if the gadgets are restricted to a plane and lines connecting gadget locations cannot cross. In [4], Demaine et al. introduced new gadgets such as the locking 2-toggle, and showed that all reversible deterministic gadgets with interacting tunnels can simulate the locking 2-toggle, and that reachability with the locking 2-toggle is PSPACE-complete, even in the planar model. In fact, reachability with any reversible deterministic gadget without interacting tunnels is in NL.

In [1], we explored nonreversible gadgets. In particular, we explored doors, which have three tunnels: one that opens the third tunnel, one the closes the third tunnel, and the third tunnel itself, which does not change the state of the gadget. Each tunnel can be directed or undirected, and the open tunnel can be a single location. We showed in [1] that reachability with any variant is PSPACE-complete and that all variants can simulate each other, except one special case in the planar model. We

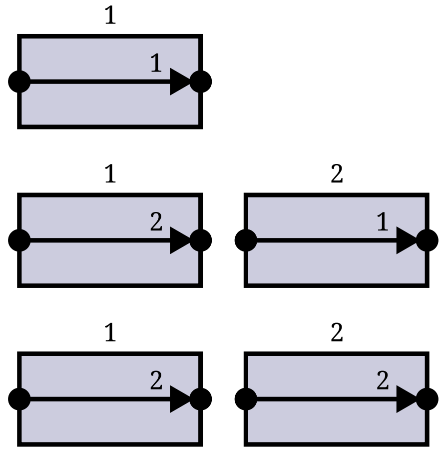


Figure 1-4: Three gadgets that are all one-way paths. In the middle and bottom gadgets, state 2 is equivalent to state 1 since it allows the same set of sequences of traversals.

also explored a related gadget family called the self-closing doors, and showed that all those variants simulate each other and are PSPACE-complete, even in the planar model.

In [2], we explored input/output gadgets, where each location is either an entrance or an exit, with applications in games with automation. The main focus was 0-player, where exits cannot be connected to multiple entrances and no choice is allowed inside gadgets either. For gadgets that contain certain components, we showed that reachability is PSPACE-complete, while for the toggle switch, which alternates which exit it leads to each time it is taken, we showed that reachability is NP-hard.

1.2 Gizmos

Gadgets, as defined above, have some problems that make them suboptimal for studying simulation. For example, there are some gadgets that have different definitions but are effectively the same in behavior. This can be caused by the presence of equivalent states (Figure 1-4), by sequences of traversals implying the existence of other traversals (Figure 1-5), or by nondeterminism (Figure 1-6). These equivalences mean that a specific intended behavior has many different representations, and computing whether gadgets are equivalent (especially regarding equivalent states) can be trickier than necessary.

These problems also make defining certain simulability classes (sets of gadgets where no gadget inside can simulate a gadget outside) tricky. For example, **LDAG** was defined in [6] as the set of gadgets whose state graphs (vertices are states, and an edge exists between state s_0 and s_1 if there is a traversal in s_0 that sets the gadget's state to s_1) are directed acyclic graphs with self-loops allowed. But because of equivalent states, they can simulate gadgets whose state diagrams have loops more

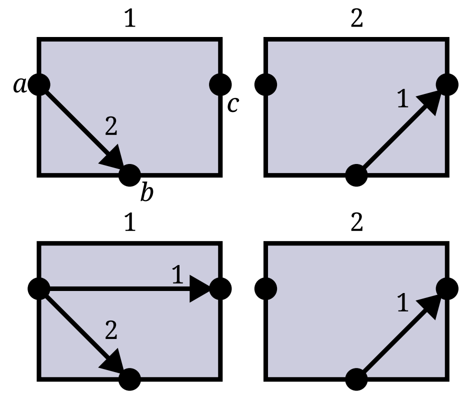


Figure 1-5: In the top gadget, in state 1, the traversal $a \to b$ followed by $b \to c$ is allowed, setting the gadget back to state 1. The bottom gadget has this sequence of traversals explicitly marked as a single traversal, which does not change the behavior.

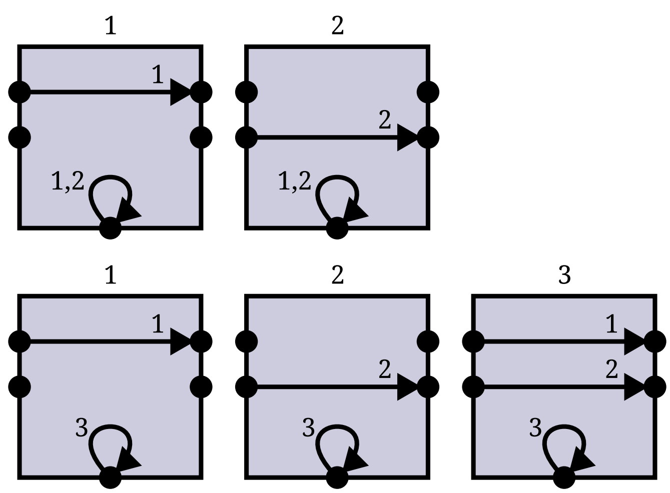


Figure 1-6: In the top gadget, the same location traversal gives you the option to choose between state 1 and state 2. The bottom gadget merely delays this choice until it is actually used, keeping the behavior the same.

than 1 edge long. **1st** can be defined as the set of gadgets with 1 state. But because of equivalent states and traversals implying other traversals, they can simulate gadgets with more than 1 state.

Due to these problems, a new model was developed: the *gizmo*, first introduced in [5]. This thesis will focus on gizmos instead of gadgets.

As a side note these equivalences are only equivalences in the 1-player reachability model, where a single agent traverses a maze of gadgets and tries to reach a particular location. If the goal is to set some gizmos to particular states (reconfiguration), then 'equivalent' states and nondeterminism do not necessarily make gadgets equivalent. Multiplayer would require a very different model due to timing.

1.3 Outline

Section 2 introduces gizmos and talks about properties they have. It also introduces and defines simulation between gizmos.

Section 3 proves results relating to gizmos not simulating other gizmos. First, we show that a gizmo called the 1-toggle cannot simulate the 2-toggle, introducing important terms and techniques along the way. Then Section 3.2 talks about simulability classes, which group gizmos based on which set of gizmos they can simulate. Section 3.2.1 describes a specific set of simulability classes, ones generated by invariant properties of gizmos about how traversals affect other traversals. Section 3.2.2 introduces other simulability classes, including ones based on a limit of how many times an agent can traverse a gizmo, and ones based on automata. Section 3.3 explores a specific family of gizmos that are variants of the single-use one-way gizmo (dicrumbler) and provides partial results of which ones can simulate which ones.

Section 4 proves results relating to a gizmo simulating all gizmos in its simulability class. First, we show universal gizmos for simulability classes **Reg** and **DAG**. Then Section 4.3 flips the problem around and proves results concerning the set of gizmos that can simulate a particular gizmo. This problem proves to be much harder, so only partial results are given. We show that gizmos with certain properties can simulate a dicrumbler.

Section 5 shows results concerning which gizmos reachability in a maze is undecidable for. We show a reduction from the counter machine halting problem to a particular gizmo, then several gizmo simulations, culminating in a proof that generalied New Super Mario Bros. is undecidable.

Chapter 2

Gizmos

This chapter is joint work with Dylan Hendrickson, Yevhenii Diomidov, and Jayson Lynch.

A gizmo [5] is similar to a gadget, but instead of encoding its traversal info into multiple states, it encodes it as a set of allowed traversal sequences in a single state.

Let L be a set of locations. A *traversal* on L is a tuple $(a, b) \in L \times L$ and is notated as $[a \to b]$. The set of all possible traversals on L (all pairs of locations) is $\mathcal{T}(L)$. A *traversal sequence* on L is a finite sequence of elements of $\mathcal{T}(L)$, notated, for example, as $[a \to b][c \to d][e \to f]$. The set of all possible traversal sequences on L is $\mathcal{T}(L)^*$. In this thesis, every set of locations is assumed to be finite.

A *gizmo* on L is a set of traversal sequences on L which says which traversal sequences in $\mathcal{T}(L)^*$ are allowed in the gizmo. It must satisfy the following properties, where a, b, c are arbitrary locations in L and X, Y are arbitrary traversal sequences on L:

- **Reflexivity.** If $XY \in G$, then $X[a \to a]Y \in G$. Same-location traversals are always allowed.
- **Transitivity.** If $X[a \to b][b \to c]Y \in G$, then $X[a \to c]Y \in G$.
- **Prefix closure.** If $XY \in G$, then $X \in G$.

Unlike in [5], gizmos in this thesis require prefix closure. A simple example of a gizmo is the *diode*, shown in Figure 2-1. The diode has a tunnel that can be traversed in one direction but not the other.

Some useful notation is defined below:

- $\text{locs}(G)$ is the set of locations of gizmo, traversal, or traversal sequence G.
- If S is a sequence, S_i is term i in the sequence, 0-indexed. $S_{i:j}$ is the substring starting at index i and ending at, but not including index j. $S_{i:} := S_{i:|S|}$, and $S_{:i} := S_{0:i}$.

Figure 2-1: The diode gizmo G. $L = \{a, b\}$, and G is the set generated by the regular expression $([a \to a] \mid [a \to b] \mid [b \to b])^*$.

- $\text{start}(T)$ is the first location in traversal or nonempty traversal sequence T. If $T_0 = [a \to b]$, where U is a traversal sequence, then $\text{start}(T) = a$.

- $\text{end}(T)$ is the last location in traversal or nonempty traversal sequence T. If $T_{|T|-1} = [a \to b]$, where U is a traversal sequence, then $\text{end}(T) = b$.

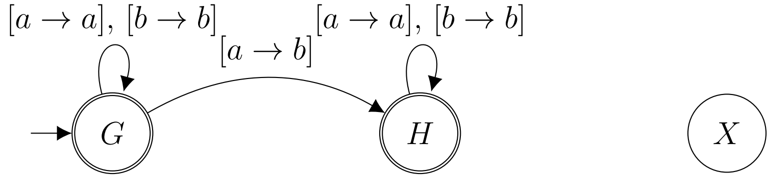
- $\mathcal{E} = \{\text{start}, \text{end}\}$ is the set of endpoint functions on a traversal, and for ordering purposes, start $<$ end.

- $\text{rtp}(R)$ is the set of traversal sequences generated by R after applying reflexivity, transitivity, and prefix closure, where R is a regular expression.

It's useful to talk about what happens after a sequence of traversals in a gizmo. The notation $G[X]$, where G is a gizmo and X is a traversal sequence in that gizmo, indicates the gizmo that results from traversing X in G. Importantly, $XY \in G \Leftrightarrow Y \in G[X]$. Another example, the *directed crumbler* or *dicrumbler*, shown in Figure 2-2, illustrates this. The dicrumbler is like a diode, but can only be crossed once. If G is a dicrumbler with locations a and b, then G allows $[a \to b]$, but $G[[a \to b]]$ does not allow $[a \to b]$.

A gizmo can become another gizmo after some traversals are taken. A *reachable state* of a gizmo G is a gizmo H such that there exists a traversal sequence T where $G[T] = H$. Each gizmo G that has a finite number of reachable states can be recognized by a DFA whose alphabet is $\mathcal{T}(\text{locs}(G))$, whose states are the reachable states of G (which are all accepting) and one non-accepting state X, with G as the starting state, and where there's a transition from state A to state B labelled T if $A[T] = B$, and a transition from A to X labelled T if $T \notin A$. We will use $\text{states}(G)$ to represent the set of reachable states of G

The DFA for the dicrumbler is shown below, with missing transitions leading to X:



Other examples of gizmos are shown in Figure 2-3, Figure 2-4, and Figure 2-5.

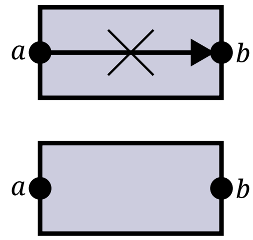


Figure 2-2: The directed crumbler gizmo G. $\mathrm{locs}(G) = \{a, b\}$, and $G = \mathrm{rtp}([a \to b]?)$. The top picture is the notation that we will use, and the bottom picture is $G[[a \to b]]$.

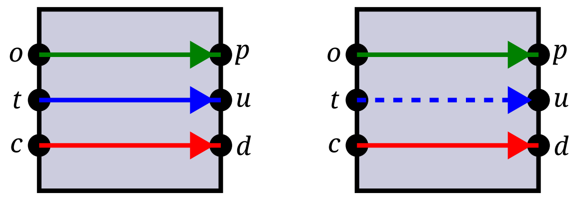


Figure 2-3: The *open door* (O) and *closed door* (C) gizmos. The green traversal, $[o \to p]$, opens the blue traversal, $[t \to u]$. The red traversal, $[c \to d]$, closes $[t \to u]$. $O = \mathrm{rtp}((([o \to p] \mid [t \to u])^*[c \to d]^*[o \to p])^*)$. $C = \mathrm{rtp}(([c \to d]^*[o \to p]([o \to p] \mid [t \to u])^*)^*)$. The green tunnel is called the *opening* tunnel and crossing it is *opening the door*, the blue tunnel is called the *traverse* tunnel and crossing it is *traversing the door*, and the red tunnel is called the *closing* tunnel and crossing it is *closing the door*.

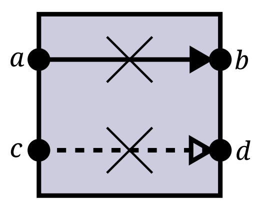


Figure 2-4: The *symmetric self-closing door* or *mismatched dicrumblers* gizmo G. The top traversal closes itself and opens the bottom traversal, which when traversed, closes itself and reopens the top traversal. $G = \mathrm{rtp}(([a \to b][c \to d])^*)$.

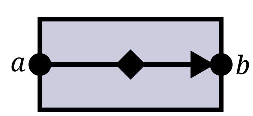


Figure 2-5: The *1-toggle* gizmo G. It's like a diode, except that it reverses direction every time it's crossed. $G = \mathrm{rtp}(([a \to b][b \to a])^*)$.

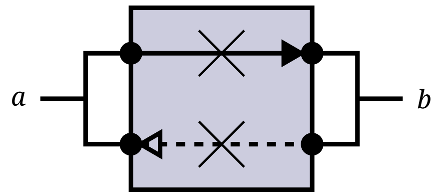


Figure 2-6: A symmetric self-closing door simulating a 1-toggle. An agent can move from a to b, and then move from b back to a. But they cannot move from b to a initially, or move from a to b twice in a row.

2.1 Simulation

Gizmos can simulate other gizmos, allowing the simulated gizmo to be replaced by the simulation in a network of gizmos while preserving the computational complexity of reachability in the network. Informally, to simulate a gizmo G with some gizmos H, you connect some locations between gizmos in H so the result behaves like G. An example is shown in Figure 2-6. A set of gizmos H can simulate G if there is a way to simulate G with a finite number of copies of elements of H. To formally define simulation, some definitions will first be introduced.

If L and L' are sets of locations and $f : L \to L'$ is a function between them, then $f_{\mathcal{T}} : \mathcal{T}(L) \to \mathcal{T}(L')$ is a function on traversals that takes $[a \to b]$ to $[f(a) \to f(b)]$, and $f^*_{\mathcal{T}} : \mathcal{T}(L)^* \to \mathcal{T}(L')^*$ is a function on traversal sequences that replaces each traversal t with $f_{\mathcal{T}}(t)$.

Let G and H be gizmos. G and H are *isomorphic* if there exists a bijection $f : \mathrm{locs}(G) \to \mathrm{locs}(H)$ where $H = \{f^*_{\mathcal{T}}(t) \mid t \in G\}$.

Let T be a set of traversal sequences. An *interleaving* of T is a traversal sequence U where the multiset of traversals in U is equal to the multiset combining the multiset of traversals in t for all $t \in T$. The set of all possible interleavings of T is $\mathcal{I}(T)$.

The first step of simulation is making copies of gizmos from the source set. Let $\mathcal{G}$ be a length-n sequence of gizmos, with G_i being the ith gizmo in the sequence. Let $f_i(a) = (a, i)$, notated as a_i for convenience. Then $\bigotimes \mathcal{G}$ is a gizmo on $\{f_i(a) \mid a \in \mathrm{locs}(G_i)\}$, and $\bigotimes \mathcal{G} := \bigotimes_{0 \le i < n} G_i := \mathcal{I}(\{f_i{}^*_{\mathcal{T}}(t) \mid t \in G_i\})$. $\bigotimes \mathcal{G}$ is called the *product* of $\mathcal{G}$, and intuitively, it is the gizmo that consists of elements of $\mathcal{G}$ with no interaction between them. G_i are called the *factors*. Note that unlike in [5], $\mathcal{G}$ is a finite sequence.

The second step of simulation is connecting locations together. Let G be a gizmo on L and $\sim$ be an equivalence relation on L. Then $G/{\sim}$ is a gizmo on $L/{\sim}$, and $G/{\sim} := \{(\pi_\sim)^*_{\mathcal{T}}(t) \mid t \in G_i\}$ after taking a transitive closure, where $\pi_\sim(a)$ is the equivalence class of a under $\sim$. $G/{\sim}$ is the *quotient* of G by $\sim$. From now on, given $\sim$, we will use $\pi_\sim$ to represent the function to equivalence classes under $\sim$. Figure 2-7 shows an example of why the transitive closure is necessary.

The final step of simulation is choosing which locations represent locations in the

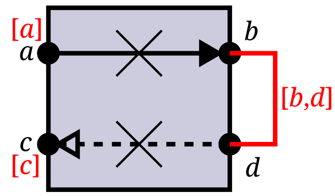


Figure 2-7: The traversal $[a \to c]$ is normally not allowed. However, if $b \sim d$, then $[a \to b][d \to c]$ becomes $[[a] \to [b,d]][[b,d] \to [c]]$, transitively allowing $[[a] \to [c]]$.

simulated gizmo. Let G be a gizmo on L, and L' be a partial injection $L \to \mathrm{locs}(H)$, where H is some gizmo. Let $\ell \in L'$ mean that L' is defined on ℓ. Then $G|_{L'}$ is a gizmo on L', and $G|_{L'} := \{{L'_{\mathcal{T}}}^*(t) \mid t \in G \text{ and } t \text{ contains only locations in } L'\}$. This is called the *subgizmo* of G on L'. There will be several cases where just the domain of L' is important, in which case we will describe just the domain.

Putting it all together, a set G of gizmos *simulates* a gizmo H if there exists a sequence $\mathcal{G} \in G^*$, an equivalence relation $\sim$, and a partial injection $L : \mathrm{locs}(G) \to \mathrm{locs}(H)$ where $H = \bigotimes \mathcal{G}/{\sim}\,|_L$. A *simulation* of H with G is the tuple $(\mathcal{G}, \sim, L)$.

Consider, for example, the open door O and closed door C shown in Figure 2-3. We show that they simulate the symmetric self-closing door S shown in Figure 2-4. Refer to Figure 2-8 for the simulation. A single copy of O and a single copy of C are needed, so set $\mathcal{G} = (O, C)$. Then $\bigotimes \mathcal{G}$ is generated by interleavings of $\mathrm{rtp}((([o_0 \to p_0] \mid [t_0 \to u_0])^*[c_0 \to d_0]^*[o_0 \to p_0])^*)$ and $\mathrm{rtp}(([c_1 \to d_1]^*[o_1 \to p_1]([o_1 \to p_1] \mid [t_1 \to u_1])^*)^*)$, after applying reflexivity and prefix closure. Then have $\sim$ produce the following equivalence classes: $[t_0], [u_0, c_0], [d_0, o_1], [p_1], [t_1], [u_1, c_1], [d_1, o_0], [p_0]$. Call them $x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3$ respectively. Then $\bigotimes \mathcal{G}/{\sim}$ contains $\mathrm{rtp}(([x_0 \to x_3][y_0 \to y_3])^*)$, along with some traversal sequences that start or end at x_1, x_2, y_1, or y_2. Lastly, set L to $\{x_1, x_2, y_1, y_2\}$. Then $\bigotimes \mathcal{G}/{\sim}\,|_L = \mathrm{rtp}(([x_0 \to x_3][y_0 \to y_3])^*)$. The result is isomorphic to the referenced symmetric self-closing door by mapping x_1, x_2, y_1, y_2 to a, b, c, d, respectively.

2.2 Reachability

We will show a few results in this thesis about the computational complexity of reachability in a maze of gizmos. It is thus useful to say what a maze is:

Definition 1. A *maze* of gizmos is a tuple $(H, \sim, s, t)$ where H is a gizmo (typically a product of gizmos), $\sim$ is an equivalence relation on $\mathrm{locs}(H)$, and $s, t \in \mathrm{locs}(H/{\sim})$ are start and target locations. *Reachability* is the problem of deciding whether $[s \to t] \in H/{\sim}$.

As mentioned before, simulation is useful for preserving computational complexity of reachability in a maze. If a gizmo G simulates a gizmo H, then reachability with H can be reduced to reachability with G by replacing every instance of H with its simulation with G, and the behavior will be the same.

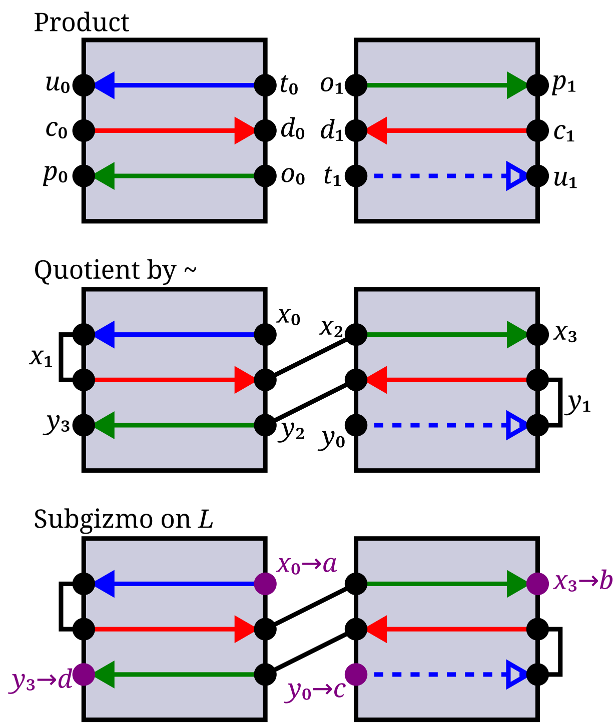


Figure 2-8: Constructing a simulation of the symmetric self-closing door with the open and closed doors.
Top: $\bigotimes \mathcal{G}$
Middle: $\bigotimes \mathcal{G}/{\sim}$. Connected locations are equivalent, and all locations can still be entered and exited. $\sim$ is the minimal equivalence relation where $u_0 \sim c_0$, $d_0 \sim o_1$, $u_1 \sim c_1$, and $d_1 \sim o_0$, and the equivalence classes are given new names.
Bottom: $\bigotimes \mathcal{G}/{\sim}\,|_L$. Locations that can be entered/exited (the ones in L) are purple, along with what they map to. $L = \{x_0 \mapsto a, x_3 \mapsto b, y_0 \mapsto c, y_3 \mapsto d\}$

Chapter 3

Unsimulability

3.1 1-Toggles

Not all gizmos can simulate each other. Reachability in a maze with 1-toggles is in NL, but reachability in a maze with 2-toggles (Figure 3-1) is PSPACE-hard [3]. Since $\mathsf{NL} \neq \mathsf{PSPACE}$, the 1-toggle cannot simulate the 2-toggle. We will soon prove this without using computational complexity, but some terms and techniques are needed. This proof will be more formal than other proofs in this thesis, as it introduces important techniques used to prove unsimulability results.

Definition 2. A *gizmo on tunnels* is a gizmo G where $\mathrm{locs}(G)$ can be partitioned into pairs of locations where no traversal in G uses locations from different pairs. A *k-tunnel gizmo* is a gizmo on tunnels where the number of pairs is k.

For example, the 1-toggle is a 1-tunnel gizmo, the symmetric self-closing door is a 2-tunnel gizmo, and the open door is a 3-tunnel gizmo.

Lemma 1. Let $H = \bigotimes \mathcal{G}$ be a product of 1-tunnel gizmos. Let $a, b \in \mathrm{locs}(H)$, Let $X, Y \in H$. If $X[a \to b]Y \in H$ and Y does not contain a or b in its locations used, then $XY \in H$.

Proof. Assume $a \neq b$. $X[a \to b]Y$ is an interleaving of traversals in $\mathcal{G}$ after labelling locations. Since H consists of 1-tunnel gizmos and since Y does not contain a or b,

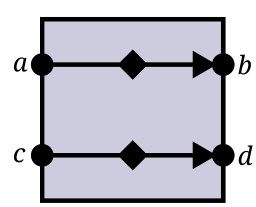


Figure 3-1: The *2-toggle* gizmo G. It consists of 2 entangled 1-toggles, so that crossing one toggles both. $G = \mathrm{rtp}((([a \to b] \mid [c \to d])([b \to a] \mid [d \to c]))^*)$.

every traversal in Y must be in a different factor of H than the one containing a and b. Thus, $[a \to b]$ can be reordered after Y. H is prefix closed because it is a gizmo, so $XY \in H$.

Now assume $a = b$. Then every traversal in Y that is in the factor of H containing a must be $[b \to b]$. Since gizmos in $\mathcal{G}$ have reflexivity, $[a \to a]$ is not necessary for $[b \to b]$ to be allowed, so $[a \to a]$ can be removed, and $XY \in H$. $\square$

It is useful to talk about paths in simulations before taking a transitive closure after connecting gizmo locations with an equivalence relation.

Definition 3. Let G be a gizmo and $\sim$ be an equivalence relation in $\text{locs}(G)$.

A $\sim$*-path* in G is a traversal sequence T in G consisting of traversals T_i where $\text{end}(T_i) \sim \text{start}(T_{i+1})$ for all i where $0 \le i < |T| - 1$.

Let $a, b \in \text{locs}(G)$. A $\sim$*-path from* a *to* b is a $\sim$-path P where $\text{start}(P) \sim a$ and $\text{end}(P) \sim b$, or if $a \sim b$, the empty traversal sequence.

A *path* in G is an $=$-path.

A *simple* $\sim$*-path* in G is a $\sim$-path P where for valid indices i, j into P and endpoint functions $e, f \in \mathcal{E}$ where $(i, e) < (j, f)$, if $e(P_i) \sim f(P_j)$, then $e = \text{end}$, $f = \text{start}$, and $j = i + 1$.

In other words, when a simple $\sim$-path exits an equivalence class of locations, it cannot reenter it.

Note that the empty traversal sequence is always a $\sim$-path, and substrings of $\sim$-paths are also $\sim$-paths.

It is also useful to talk about connectedness of traversal sequences in a way that doesn't special case the empty traversal sequence.

Definition 4. Let X, Y be traversal sequences. X is $\sim$*-connected* to Y if $\text{end}(X) \sim \text{start}(Y)$ or $X = []$ or $Y = []$. This is notated as $X \overset{\sim}{\hookrightarrow} Y$. In the case where $\sim$ is $=$, X is *connected* to Y, notated as $X \hookrightarrow Y$.

The following lemmas make the connection abstraction useful.

Lemma 2. Let X and Y be traversal sequences in the same gizmo G and let $\sim$ be an equivalence relation on $\text{locs}(G)$. XY is a $\sim$-path in G if and only if X and Y are both $\sim$-paths and $X \overset{\sim}{\hookrightarrow} Y$.

Proof. $\Rightarrow$: If X or Y is empty, then X and Y are both $\sim$-paths (the nonempty one is equal to XY), and $X \overset{\sim}{\hookrightarrow} Y$ by Definition 4. Otherwise, by Definition 3, $\text{end}(X) \sim \text{start}(Y)$, so $X \overset{\sim}{\hookrightarrow} Y$. Since X and Y are substrings of XY, they are both $\sim$-paths.

$\Leftarrow$: If X or Y is empty, then XY is a $\sim$-path. Otherwise, by Definition 4, $\text{end}(X) = \text{start}(Y)$. So the condition for being a path applies for all traversal indices into XY, so XY is a $\sim$-path. $\square$

The following lemma proves that the last loop can be taken out of paths through mazes with only 1-tunnel gizmos.

Lemma 3. Let $H = \bigotimes \mathcal{G}$ be a product of 1-tunnel gizmos. Let $\sim$ be an equivalence relation in H. If XYZ is a $\sim$-path in H, and $Y \overset{\sim}{\hookrightarrow} Y$, and Z does not contain locations in X or Y, then XZ is a $\sim$-path in H.

Proof. Since Z does not contain locations in Y, repeatedly apply Lemma 1 to the last traversal of Y until it becomes the empty traversal sequence. Thus, $XZ \in H$. Since XYZ is a $\sim$-path, $X \overset{\sim}{\hookrightarrow} Y$ and $Y \overset{\sim}{\hookrightarrow} Z$. Note that $Y \overset{\sim}{\hookrightarrow} Y$. If X or Z is empty, then $X \overset{\sim}{\hookrightarrow} Z$. Otherwise if Y is empty, then XZ is a $\sim$-path. Otherwise, $\text{end}(X) \sim \text{start}(Y)$ and $\text{end}(Y) \sim \text{start}(Y)$ and $\text{end}(Y) \sim \text{start}(Z)$, so $X \overset{\sim}{\hookrightarrow} Z$. By Lemma 2, XZ is a $\sim$-path. $\square$

The following lemma proves that all loops can be taken out of paths through mazes with only 1-tunnel gizmos.

Lemma 4. Let $H = \bigotimes \mathcal{G}$ be a product of 1-tunnel gizmos. Let $\sim$ be an equivalence relation in H. Let $a, b \in \text{locs}(H)$. If there is a $\sim$-path P' in H from a to b, then there exists a simple $\sim$-path in H from a to b.

Proof. Let P be a shortest $\sim$-path in H from a to b. If P is a simple $\sim$-path, then the statement is proven. Otherwise let $i, j \in [0..|\mathcal{G}|)$ and $e, f \in \mathcal{E}$ be counterexamples of the condition of the definition of a simple $\sim$-path, with (j, f) as late as possible in P. Then $(i, e) < (j, f)$ and $e(P_i) \sim f(P_j)$ but $e \neq \text{end}$, $f \neq \text{start}$, or $j \neq i + 1$. Consider $i', j' \in [0..|\mathcal{G}|)$ and $e', f' \in \mathcal{E}$ where $(i', e') = \begin{cases} (i, \text{start}) & e = \text{start} \\ (i+1, \text{start}) & e = \text{end} \end{cases}$ and $(j', f') = \begin{cases} (j, \text{end}) & f = \text{end} \\ (j-1, \text{end}) & f = \text{start} \end{cases}$. Note that $i' \leq j'$. No matter what, $i < j$. Either $e = \text{start}$, in which case $i' = i \leq j'$, or $f = \text{end}$, in which case $i' \leq j' = j$, or $j > i + 1$, in which case $i' \leq j'$ since the gap can shrink by only 2. Note that $P_{:j'+1}$ has no locations equivalent by $\sim$ to ones in $P_{j'+1:}$ except $\text{end}(P_{j'})$ (if $P_{j'+1:}$ is nonempty) since (j, f) is as late as possible while still being a counterexample. In addition, $P_{j'+1:}$ is a simple $\sim$-path. Also note that $\text{start}(P_{i'}) \sim \text{end}(P_{j'})$, so setting $P_{i':j'+1} \overset{\sim}{\hookrightarrow} P_{i':j'+1}$. Let $X = P_{:i'}$, $Y = P_{i':j'+1}$, and $Z = P_{j'+1:}$. Lemma 3 can almost be used, but it is necessary for Z to not share locations with Y, including $\text{end}(P_{j'})$. If Z is empty, this is true. Otherwise, it is sufficient to show that $P_{j'}$ and $P_{j'+1}$ occur in different factors of H. Assume they happen in the same factor G. Let $c = \text{end}(P_{j'})$ and label the other location in the factor d. If $\text{start}(P_{j'+1}) = c$, then $P_{j'}P_{j'+1}$ transitively reduces to $[\text{start}(P_{j'}) \to \text{end}(P_{j'+1})]$ shortening the $\sim$-path, a contradiction. Otherwise, $c \sim d$, so all traversals in G are unnecessary since all its locations are equivalent, and in particular, $P_{j'}$ and $P_{j'+1}$ can be removed, shortening the $\sim$-path, a contradiction. So Lemma 3 can be applied, shortening the $\sim$-path, a contradiction. So P must be a simple $\sim$-path. $\square$

The following lemma proves that with 1-tunnel gizmos, if two paths intersect, then you can switch paths at the intersection.

Lemma 5. Let $H = \bigotimes \mathcal{G}$ be a product of 1-tunnel gizmos. Let $\sim$ be an equivalence relation in H. Let $a, b, c, d, e \in \text{locs}(H)$. Let P be a simple $\sim$-path in H from a to b, and Q be defined similarly but from c to d. If P and Q both contain locations that $\sim e$, then there is a $\sim$-path in H from a to d.

Proof. Let $(i, f) \in (\mathbb{N}, \mathcal{E})$ be the biggest such tuple where $f(Q_i)$ is equivalent by $\sim$ to some location in P. This must exist because P and Q contain locations that $\sim e$. Let $(j, g) \in (\mathbb{N}, \mathcal{E})$ be the smallest such tuple where $g(P_j) \sim f(Q_i)$. If $f = \text{end}$, then since Q is a path and (i, f) is the biggest such tuple that meets its condition, $f(Q_i) = \text{end}(Q) \sim d$. Then P contains as a prefix a $\sim$-path from a to d, proving the statement. Otherwise, if $g = \text{start}$, then $g(P_j) = \text{start}(P)$, and Q crosses a location that $\sim a$ en route to d. Since H is a product of 1-tunnel gizmos and Q is a simple $\sim$-path, Q traverses a different factor of H on each traversal, and so every suffix of Q is in H, including the suffix that is a $\sim$-path from a to d. Otherwise, $f = \text{start}$ and $g = \text{end}$. An example of the situation is shown in Figure 3-2. Consider $P_{:j+1}$ and $Q_{i:}$. Note that $P_{:j+1} \in H$ due to prefix closure. By similar logic as used above, $Q_{i:} \in H$ since it is a suffix of Q. If these two subpaths traversed the same factor of H, then they would both contain locations equivalent by $\sim$ at a point in Q later than (i, f) since the traversals would have to, without loss of generality, both be $[x \to y]$ or one be $[x \to y]$ and one be $[y \to x]$, a contradiction. So $P_{:j+1}$ and $Q_{i:}$ traverse different factors of H and can be taken in any order. Thus, $P_{:j+1}Q_{i:}$, which is a $\sim$-path from a to d, is in H. $\square$

To finish the proof, it is necessary to first connect the above lemmas to gizmo simulations.

Lemma 6. Let $H = \bigotimes \mathcal{G}$ be a product of gizmos. Let $\sim$ be an equivalence relation in H. Let L be a partial injection from $\text{locs}(H/\sim)$ and let $a, b \in L$. If $[L(a) \to L(b)] \in H/\sim|_L$, then there is a $\sim$-path from some location $a' \in H$ to some location $b' \in H$ where $\pi_\sim(a') = a$ and $\pi_\sim(b') = b$.

Proof. $[a \to b]$ in $H/\sim$ must have been constructed somehow. Either there is a traversal $[a' \to b']$ in H where $\pi_\sim(a') = a$ and $\pi_\sim(b') = b$ (which would already be a $\sim$-path), or it was constructed by transitive closure during the construction of $H/\sim$, in which case, there must be a $\sim$-path from some location $a' \in H$ to some location $b' \in H$ where $\pi_\sim(a') = a$ and $\pi_\sim(b') = b$ to construct $[a \to b]$ with the transitive closure of after taking the quotient of H by $\sim$. $\square$

Now it can be proven that 1-tunnel gizmos cannot simulate the 2-toggle.

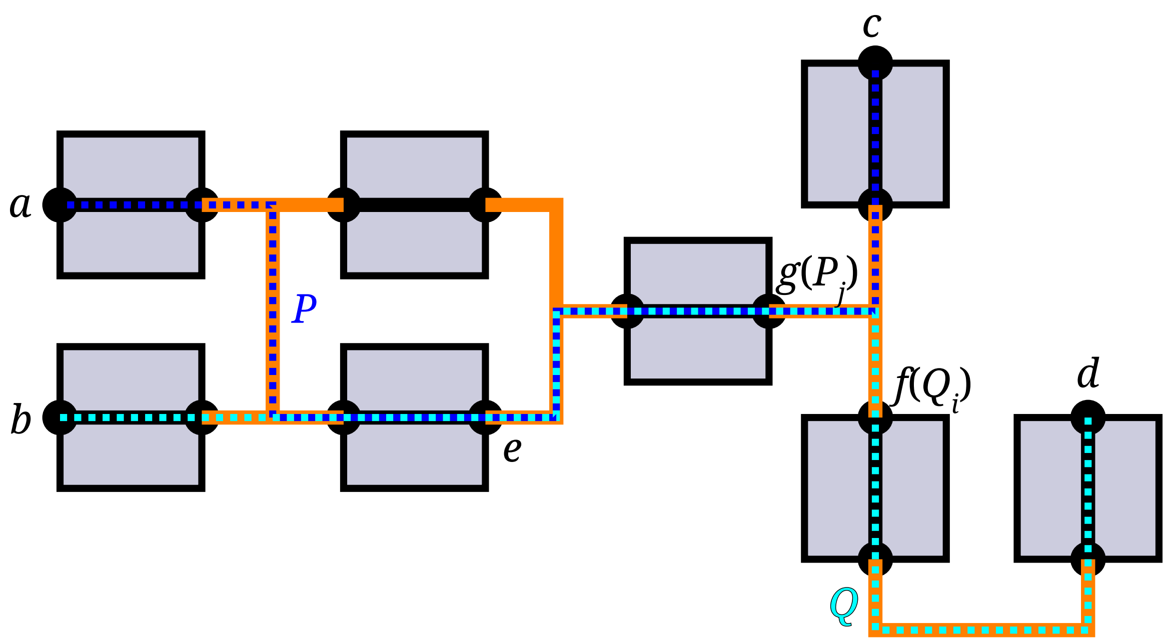


Figure 3-2: An illustration of the proof of Lemma 5. Orange lines connect locations equivalent by $\sim$. In this case, $i = 3$ and $j = 2$. $P_{:j+1}$ and $Q_{i:}$ combine to form a $\sim$-path from a to d.

Theorem 1. *Let G be a set of 1-tunnel gizmos. G cannot simulate the 2-toggle W shown in Figure 3-1.*

Proof. Let $(\mathcal{G}, \sim, L)$ be a simulation of W with G. Note that $[a \to b] \in W$ and $[c \to d] \in W$, but $[a \to b][c \to d] \notin W$ and $[a \to d] \notin W$. Consider $H = \bigotimes \mathcal{G}$. By Lemma 6 and Lemma 4, there exists $a', b', c', d' \in H$ where $L(\pi_\sim(a')) = a$ and similarly for the rest, and a simple $\sim$-path P in H from a' to b', and a simple $\sim$-path Q in H from c' to d'. If P and Q do not intersect, then by definition of $\bigotimes \mathcal{G}$ as consisting of all interleavings of traversal sequences in $\mathcal{G}$, $PQ \in H$. Then $[a \to b][c \to d] \in W$ by transitive closure, a contradiction. So P and Q have to intersect at some location e. By Lemma 5, there is a simple $\sim$-path from a' to d' in H. By transitive closure, $[a \to d] \in W$, a contradiction. So $(\mathcal{G}, \sim, L)$ is not a simulation of W after all. $\square$

3.2 Simulability Classes

This section is joint work with Dylan Hendrickson, Yevhenii Diomidov, and Jayson Lynch.

Sets of gizmos can be grouped into simulability classes based on which gizmos they can simulate and which ones they cannot. A *simulability class* is a set A of gizmos where no gizmo in A can simulate any gizmo outside A. For example, consider the simulability class **1st**, containing only gizmos that satisfy $\forall X \, . \, X \in G \implies G[X] = G$. (**1st** stands for 1-state). The diode, for example, is in **1st**. (We will reference a proof that **1st** is indeed a simulability class later.)

3.2.1 Implication Properties

One way to define simulability classes is with *implication properties*, first defined in [5]. First, some preliminary definitions are needed.

Definition 5. Let $T = [a \to b]$ be a traversal. Then the *inverse* of T, notated as T^{-1}, is $[b \to a]$.

Let T be a traversal sequence. Then the *inverse* of T, notated as T^{-1}, is the sequence of inverses of T_i in reverse order.

A *traversal formula* is a function $f : (\mathcal{T}(L)^*)^n \to \mathcal{T}(L)^*$, for some n and no matter what location set L is, that takes a sequence $\mathcal{X}$ of traversal sequences and outputs a traversal sequence choosing elements of $\mathcal{X}$ and/or their inverses and concatenating them.

For example, if X, Y are traversal sequences, f could take (X, Y) and return $YXXX^{-1}$.

Definition 6. Given n, a set of traversal formulas $F \in \mathcal{P}((\mathcal{T}(L)^*)^n \to \mathcal{T}(L)^*)$, no matter what L is, is *simple* if given a sequence $\mathcal{X} : . (\mathcal{T}(L)^*)^n$, for some L, of traversal sequences, $\mathcal{X}_i$ appears exactly once in the outputs of elements of F on $\mathcal{X}$ for all valid indices i, and $\mathcal{X}_i^{-1}$ doesn't appear for any i.

For example, the set $\{f, g\}$ where $f(X, Y, Z) = X$ and $g(X, Y, Z) = ZY$ is simple. If $f(X) = X$ and $g(X) = X$, then $\{f, g\}$ is not simple because X appears multiple times in the outputs. If $f(X) = X^{-1}$, then $\{f\}$ is not simple because an inverse appears in the output.

Implication properties can now be defined:

Definition 7. An *implication property* is a tuple consisting of a simple set F of traversal formulas and a traversal formula g. The notation $X, YZ \implies XZX^{-1}$ corresponds to $F = \{f_0, f_1\}$ where $f_0(X, Y, Z) = X$ and $f_1(X, Y, Z) = YZ$, and $g(X, Y, Z) = XZX^{-1}$. Notation for other implication properties is defined similarly.

An important theorem is that every implication property forms a simulability class.

Theorem 2. *Let (F, g) be an implication property. Let A be the set of all gizmos G that each have the property: "Let $\mathcal{X}$ be a sequence of traversal sequences in G. Then $(\forall f \in F \,.\, f(\mathcal{X}) \in G) \implies g(\mathcal{X}) \in G$" Then A is a simulability class.*

This theorem is proven in [5], and will not be repeated here.

It is now provable that **1st** is a simulability class. Note that the property $\forall X \,.\, X \in G \implies G[X] = G$ is equivalent to $\forall X, Y \,.\, X \in G \implies ((Y \in G[X] \implies Y \in G) \wedge (Y \in G \implies Y \in G[X]))$. This can be written as the implication properties $XY \implies Y$ and $X, Y \implies XY$. (Note that the first property doesn't

Name	Description	Implication property	Example
OI	Order-independent	$XY \implies YX$	2-use matched dicrumblers (Figure 3-3)
DirBlind	Direction-blind	$X \implies X^{-1}$	Crumbler (Figure 3-4)
Reuse	Reusable	$X \implies XX$	Tunnel chooser (Figure 3-5)
Undo	Undoable	$X \implies XX^{-1}$	1-toggle (Figure 2-5)
Close	Closing	$XY \implies Y$	ZXY enforcer with cutting (Figure 3-6)
Open	Opening	$X, Y \implies XY$	Open-only door (Figure 3-7)
Close*	Closing forever	$ZXY \implies ZY$	Tunnel chooser (Figure 3-5)

Table 3.1: Simulability classes by implication property

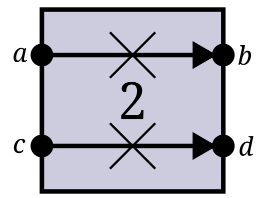


Figure 3-3: The *2-use matched dicrumblers* G. $[a \to b]$ and $[c \to d]$ can be traversed a total of 2 times before the gizmo closes. $G = \mathrm{rtp}(([a \to b] \mid [c \to d])\{2\})$

say $X, XY \implies Y$ because that would not be an implication property, and by prefix closure, $XY \in G$ implies $X \in G$ anyway.)

Other simulability classes defined by implication properties are shown in Table 3.1.

An interesting fact is that **Close** and **Close*** are different simulability classes. The ZXY enforcer with cutting G is in **Close**, but it is not in **Close*** because $[z_0 \to z_1][x_0 \to x_1][y_0 \to y_1] \in G$ but $[z_0 \to z_1][y_0 \to y_1] \notin G$. So sometimes requiring an implication property to be satisfied even after taking an arbitrary traversal sequence in a gizmo makes a new simulability class. This is not always true, however. For gizmos in **Undo**, $X \implies XX^{-1}$. Then $ZX \implies ZXX^{-1}Z^{-1}$, and by prefix closure, $ZX \implies ZXX^{-1}$, showing that **Undo*** is not a new simulability class. In addition, attempting to construct **Open*** gives $ZX, ZY \implies ZXY$, which is not an implication property due to the repeated Z.

From now on, we will prove results less formally than the unsimulability between the 1-toggle and 2-toggle because the formal proofs get complicated.

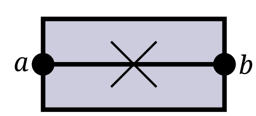


Figure 3-4: The *crumbler* G. Like the dicrumbler, but both directions can be taken. $G = \mathrm{rtp}([a \to b] \mid [b \to a])$

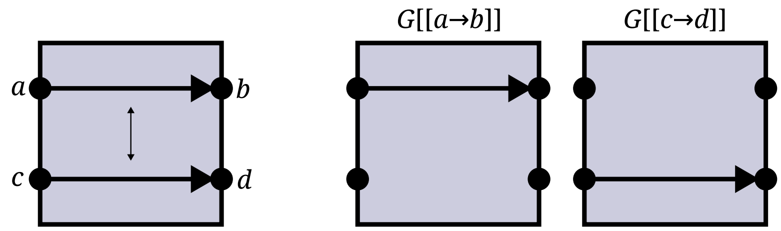


Figure 3-5: The *tunnel chooser* G. Either tunnel can be traversed, but then it becomes the only traversable tunnel. $G = \mathrm{rtp}([a \to b]^* \mid [c \to d]^*)$

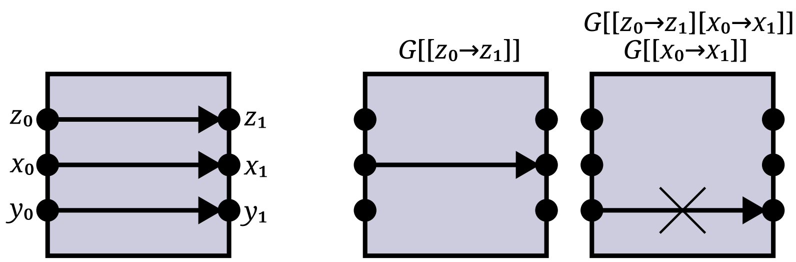


Figure 3-6: The *ZXY enforcer with cutting* G. Traversals must be made in the order $[z_0 \to z_1]$, $[x_0 \to x_1]$, $[y_0 \to y_1]$, but the sequence can be started anywhere in the order. $G = \mathrm{rtp}([z_0 \to z_1][x_0 \to x_1][y_0 \to y_1] \mid [x_0 \to x_1]?[y_0 \to y_1])$

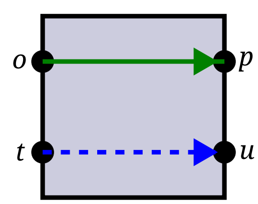


Figure 3-7: The *open-only door* G. Similar to the closed door, but there is no tunnel to close it back. $G = \mathrm{rtp}([o_0 \to o_1]([t_0 \to t_1] \mid [o_0 \to o_1])^*)$

3.2.2 Other Simulability Classes

Not all simulability classes are generated from implication properties. For example, as mentioned and proven before, the 1-toggle cannot simulate the 2-toggle. However, as we will now show, the 2-toggle satisfies every implication property that the 1-toggle satisfies. Therefore, there exists a simulability class that the 1-toggle is in but the 2-toggle is not in, and that class is not generated from an implication property.

Theorem 3. *The 2-toggle satisfies every implication property that the 1-toggle satisfies.*

Proof. Let the 1-toggle's locations be labelled as in Figure 2-5 and the 2-toggle's locations be labelled as in Figure 3-1. Let (F, g) be an implication property that the 1-toggle satisfies. Let $\mathcal{S}$ be a sequence of traversal sequences in the 2-toggle that g and elements of F can take as input. Replace every instance of c in $\mathcal{S}$ with a and every instance of d with b to get $\mathcal{S}'$. Since (F, g) is satisfied by the 1-toggle, $(\forall f \in F \ . \ f(\mathcal{S}') \in \text{1-toggle}) \implies g(\mathcal{S}') \in \text{1-toggle}$. But note that in the 2-toggle, $[c \to c]$, $[c \to d]$, $[d \to c]$, and $[d \to d]$ are allowed whenever $[a \to a]$, $[a \to b]$, $[b \to a]$, and $[b \to b]$ are allowed, respectively, and traversals from one of $\{\{a, b\}, \{c, d\}\}$ to the other are never allowed. So $(\forall f \in F \ . \ f(\mathcal{S}) \in \text{2-toggle}) \implies g(\mathcal{S}) \in \text{2-toggle}$. So the 2-toggle satisfies (F, g). $\square$

It is an open question whether there is a simple way to check membership in the set of gizmos that the 1-toggle can simulate.

A concrete example of a simulability class not generated by an implication property is **DAG** [4]. **DAG**, which stands for "directed acyclic graph", is the class of gizmos G that satisfy: "There exists an integer n where every traversal sequence in G has at most n different-location traversals." and have a finite number of reachable states, which in effect says that you can *cross* the gizmo only a bounded number of times before it closes. These are so called because the DFA state transition diagram, excluding self-loops made by same-location traversals, is a directed acyclic graph. The 2-use matched dicrumblers, for example, is in **DAG** with $n = 2$, and the ZXY enforcer with cutting (Figure 3-6) is in **DAG** with $n = 3$.

Theorem 4. **DAG** *is a simulability class.*

Proof. Let $\mathcal{G}$ be a sequence of gizmos in **DAG**, and let $n(G)$ be the lowest bound for the number of different-location traversals for arbitrary **DAG** gizmo G. Since $\bigotimes \mathcal{G}$ contains only interleavings of allowed traversal sequences in $\mathcal{G}$, $\bigotimes \mathcal{G} \in \mathbf{DAG}$ with $n = \sum_{G \in \mathcal{G}} n(G)$. Quotienting by an arbitrary equivalence relation $\sim$ can turn traversals into same-location traversals, and can collapse $\sim$-paths into single traversals, but cannot add a new different-location traversal to an allowed traversal sequence without removing a different-location traversal. Subgizmoing by a location set L can only remove traversal sequences. So $\bigotimes \mathcal{G}/\!\sim|_L \in \mathbf{DAG}$ with $n \leq \sum_{G \in \mathcal{G}} n(G)$. So every gizmo that a gizmo in **DAG** can simulate is also in **DAG**. $\square$

We will now prove that gizmos in **DAG** has the DAG like property mentioned above.

Theorem 5. *Let $G \in$ **DAG**. Then the DFA state transition diagram, excluding self-loops made by same-location traversals, is a directed acyclic graph.*

Proof. It suffices to show that the reachable states of G have a partial order when ordered by reachability.

A different-location traversal decreases the lowest bound on the number of different-location traversals allowed by 1, and a same-location traversal that changes the state changes it to one with strictly more traversability, since $G[[a \to a]] \supseteq G$ by reflexive closure, without increasing the lowest bound. So the states are partially ordered by decreasing lowest bound, then by increasing traversability in the case of ties, when ordered by reachability. □

A similar simulability class is **LDAG** ("loops directed acyclic graph") [6], which contains only gizmos G that have a finite number of reachable states and satisfy: "There exists an integer n and an NFA N that recognizes G where every traversal sequence in G has at most n state transitions in N." This is similar to **DAG**, but with an NFA instead of a DFA and with self-loops in the NFA made by different-location traversals can happen an unbounded number of times. The tunnel chooser, for example, is in **LDAG** with $n = 1$.

Theorem 6. LDAG *is a simulability class.*

Proof. Let $\mathcal{G}$ be a sequence of gizmos in **LDAG**, let N_i be an NFA that meets the condition for G_i to be in **LDAG**, and let $n(G)$ be the lowest bound for the number of state-transitioning traversals under N for arbitrary **LDAG** gizmo G. Since $\bigotimes \mathcal{G}$ contains only interleavings of allowed traversal sequences in $\mathcal{G}$, and not changing the state in any $\mathcal{G}_i$ means not changing the state of $\bigotimes \mathcal{G}$, then $\bigotimes \mathcal{G} \in$ **LDAG** with N', the product of all N_i, being the NFA, and $n = \sum_{G \in \mathcal{G}} n(G)$ being the bound. Quotienting by an arbitrary equivalence relation $\sim$ can add traversals implied by $\sim$-paths, which turns N' into an NFA where all paths still have at most n state transitions. Subgizmoing by a location set L can only remove traversals from the NFA and does not affect the bound. So $\bigotimes \mathcal{G}/\sim|_L \in$ **LDAG**. □

Another simulability class is **Reg** ("regular"), which contains only gizmos that have a finite number of reachable states. Every gizmo mentioned so far is in **Reg**, but in the undecidability section, we will discuss some exotic gizmos that have an infinite number of reachable states.

Theorem 7. Reg *is a simulability class.*

Proof. Let $\mathcal{G}$ be a sequence of gizmos in **Reg**, and let $n(G)$ be the number of reachable states in G. The number of reachable states in $\bigotimes \mathcal{G}$ is $N := \prod_{G \in \mathcal{G}} n(G)$ because a reachable state in $\bigotimes \mathcal{G}$ is a combination of reachable states in its factors. Given

$(\mathbf{m}, \mathbf{n})$	Simulates
(m, n)	(m, n') where $n' \leq n$
(m, n)	(m', n) where m divides m'
(m, n) where $n > m$	(m', n')

Table 3.2: Positive dicrumbler variant simulation results

$(\mathbf{m}, \mathbf{n})$	Does not simulate
$(m, 1)$	$(m', 1)$ where m does not divide m'
(m, n) where $n \leq m$	(m, n) where $n > m$
(2, 2)	(3, 2)

Table 3.3: Negative dicrumbler variant simulation results

an equivalence relation $\sim$, and a set of locations $L \subseteq \text{locs}(\bigotimes \mathcal{G}/\sim)$, $\bigotimes \mathcal{G}/\sim |_L$ is recognized by an NFA with N states: the NFA that is the result of taking the DFA for $\bigotimes \mathcal{G}$, replacing every location with its equivalence class, adding new transitions for transitive closure, and removing transitions that contain locations not in L. It is then recognized by a (not necessarily minimal) DFA with 2^N states, which means that it has a finite number of reachable states. $\square$

3.3 Dicrumbler Variants

A later section of this thesis discusses the notion of bottom universality, which is a universality result in reverse: every gizmo with some property can simulate a specific gizmo. A simple gizmo to try this with is the dicrumbler. Even with the dicrumbler, though, coming up with the right property is tricky. This motivates a dicussion on why gizmos can or can't simulate the dicrumbler. This section in particular goes into some unsimulability results regarding variants of the dicrumbler, varying the number of uses and the number of tunnels.

The (m, n)-*dicrumbler* or *m-use n-tunnel dicrumbler* is a gizmo G consisting of $2n$ locations (which will be labelled s_0 through s_{n-1} and t_0 through t_{n-1}), where

$$G = \text{rtp}\left(\left(\bigcup_{i=0}^{n-1} [s_i \to t_i]\right)^m\right).$$

. In other words, the traversals $[s_i \to t_i]$ can be taken a total of up to m times before the gizmo closes. For example, a 2-use dicrumbler would be a (2, 1)-dicrumbler, and the dicrumbler is a (1, 1)-dicrumbler. Positive results are summarized in Table 3.2, and negative results are summarized in Table 3.3.

First, we give a complete characterization of which dicrumbler variants can simulate which ones when $n = 1$:

Theorem 8. *Given m and p, the m-use dicrumbler can simulate the p-use dicrumbler*

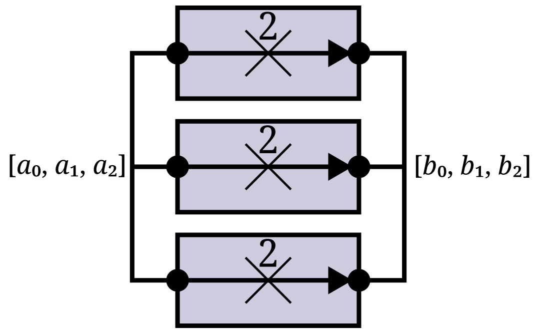


Figure 3-8: A 2-use dicrumbler simulating a 6-use dicrumbler. This is possible because 2 divides 6.

if and only if m divides p.

Proof. $\Rightarrow$: Assume that m does not divide p. Let $(\mathcal{G}, \sim, L)$ be a simulation of the p-use dicrumbler with the m-use dicrumbler. Let a be the entrance of the p-use dicrumbler and b be its exit. The simulation can be modelled as a flow network whose vertices are locations in $\text{locs}(\bigotimes \mathcal{G}/\sim)$, whose edges are the multiset $\{(\pi_\sim(u), \pi_\sim(v))$ with capacity $m \mid u, v \in \text{locs}(\bigotimes \mathcal{G}), [u' \to v'] \in \bigotimes \mathcal{G}\}$ after combining equal edges and their capacities, whose source is $L^{-1}(a)$, and whose target is $L^{-1}(b)$. Since all edges have capacities that are multiples of m, the flow must be a multiple of m, which p is not. So the simulation allows too many traversals.

$\Leftarrow$: The simulation takes $\frac{p}{m}$ m-use dicrumblers, combines all their start locations and all their end locations, and results in a dicrumbler that can be used $\frac{p}{m} \times m = p$ times. Let $\mathcal{G}$ be a sequence of $\frac{p}{m}$ m-use dicrumblers with the m-use dicrumbler having its locations labelled a and b. Then $\bigotimes \mathcal{G}$ has locations labelled a_i and b_i for all i where $0 \leq i < \frac{p}{m}$. Let $\sim$ be the minimal equivalence relation where $a_i \sim a_j$ and $b_i \sim b_j$ for all i, j where $0 \leq i, j < \frac{p}{m}$. Then $\bigotimes \mathcal{G}/\sim$ is a p-use dicrumbler. An example is shown in Figure 3-8. □

Then, a universality result concerning the (1, 2)-dicrumbler:

Theorem 9. *Given m and n, the (1, 2)-dicrumbler can simulate the (m, n)-dicrumbler.*

Proof. Label the locations of the (1, 2)-dicrumbler as in Figure 3-3, except that it is 1-use.

First, we show that the (1, 2)-dicrumbler simulates the $(1, n)$-dicrumbler. Basically, the simulation takes n tunnels and puts a (1, 2)-dicrumbler between each pair of tunnels, so that if any tunnel is crossed, no tunnels can then be crossed.

If $n = 1$, then restrict the (1, 2)-dicrumbler to the locations $\{a, b\}$, simulating a (1, 1)-dicrumbler. Otherwise, for all integers i, j where $0 \leq i < j < n$, let there be a (1, 2)-dicrumbler labelled i, j. For simplicity, for all integers j where $0 \leq j < n$, let there be a (1, 1)-dicrumbler (which can be simulated, as mentioned)

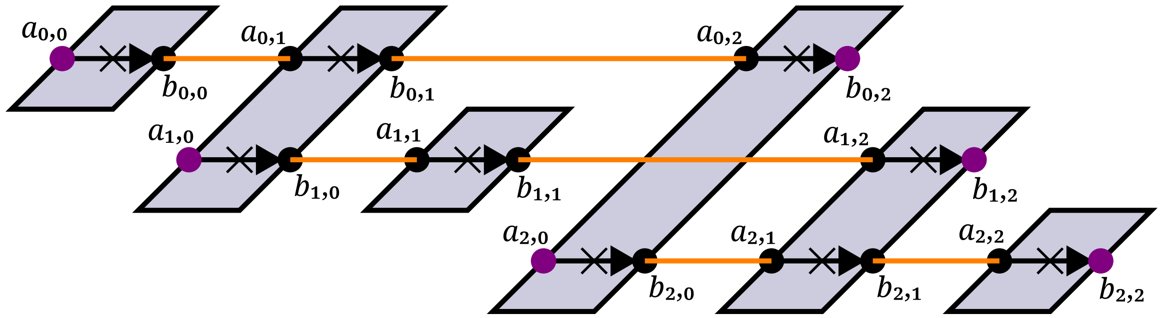


Figure 3-9: A (1, 2)-dicrumbler simulating a (1, 3)-dicrumbler. Orange lines connect equivalent locations, and purple locations are in L. The (1, 1)-dicrumblers are technically not necessary, but they simplify the proof.

labelled j, j with its locations labelled a (entrance) and b (exit). Combine these into a sequence $\mathcal{G}$. Then $H = \bigotimes \mathcal{G}$ has locations $a_{i,j}$, $b_{i,j}$, $c_{i,j}$ and $d_{i,j}$ for all i, j satisfying $0 \leq i < j < n$, as well as $a_{j,j}$ and $b_{j,j}$ satisfying $0 \leq j < n$. Do the relabellings $a_{j,i} := c_{i,j}$ and $b_{j,i} := d_{i,j}$. Let $\sim$ be the minimal equivalence relation where $b_{i,j} \sim a_{i+1,j}$ for all i, j where $0 \leq i < n-1$ and $0 \leq j < n$. Let L map the locations $bigcup\{\{\pi_\sim(a_{0,j}), \pi_\sim(b_{n-1,j})\} \mid 0 \leq j < n\}$. The only minimal (no unnecessary same-location traverals) $\sim$-paths in H between locations that map using $\pi_\sim$ to different locations in L are $X_j := [a_{0,j} \to b_{0,j}] \cdots [a_{n-1,j} \to b_{n-1,j}]$. There are n of them, and they do not share locations equivalent by $\sim$. If X_j is taken for some j, then X_j cannot be taken again. In addition, for each i where $0 \leq i < n$ and $i \neq j$, X_j contains $[a_{i,j} \to b_{i,j}]$, so $[a_{j,i} \to b_{j,i}]$ (which is part of the same factor of H) closes, so X_i cannot be taken. Thus, $H/\sim|_L$ is a (1, n)-dicrumbler. An example is shown in Figure 3-9.

To show that the $(1, n)$-dicrumbler can simulate the (m, n)-dicrumbler, use a construction similar to the proof of Theorem 8, taking m $(1, n)$-dicrumblers and combining their start locations for each tunnel, and combining their end locations for each tunnel. Each $(1, n)$-dicrumbler can be crossed only once, giving an m-use n-tunnel dicrumbler. □

It is useful to prove a version of Lemma 5 for gizmos in **Close***. Since dicrumbler variants are in **Close***, this allows for more unsimulability results.

Lemma 7. Let $H = \bigotimes \mathcal{G}$ be a product of gizmos in **Close*** that also satisfy "for each $G \in \mathcal{G}$, $\forall X, Y \in G$. XY does not repeat locations at all $\implies XY \in G$". Let $\sim$ be an equivalence relation in H. Let $a, b, c, d, e \in \text{locs}(H)$. Let P' be a $\sim$-path in H from a to b, and Q be defined similarly but from c to d. If P and Q both contain locations that $\sim e$, then there is a $\sim$-path in H from a to d.

Proof. This proof is similar to the one for Lemma 5, but adapted for gizmos in **Close***.

Let P be a simple $\sim$-path in H from a to b. It must exist because $H \in$ **Close*** and loops can be removed. Let $(i, f) \in (\mathbb{N}, \mathcal{E})$ be the biggest such tuple where

$f(Q_i) \sim e$. Let $(j, g) \in (\mathbb{N}, \mathcal{E})$ be the smallest such tuple where $g(P_j) \sim e$. If $f = \text{end}$, then since Q is a path and (i, f) is the biggest such tuple that meets its condition, $f(Q_i) = \text{end}(Q) \sim d$. Then P contains as a prefix a $\sim$-path from a to d, proving the statement. Otherwise, if $g = \text{start}$, then $g(P_j) = \text{start}(P)$, and Q crosses a location that $\sim a$ en route to d. Since H is a product of gizmos in simulability class $\mathbf{Close}^*$, $H \in \mathbf{Close}^*$ so every suffix of Q is in H, including the suffix that is a $\sim$-path from a to d. Otherwise, $f = \text{start}$ and $g = \text{end}$. Consider $P_{:j+1}$ and $Q_{i:}$. Since $H \in \mathbf{Close}^*$, $P_{:j+1}$, $Q_{i:}$, and all subsequences of both are in H. $P_{:j+1}Q_{i:}$ is a $\sim$-path in $\mathcal{T}(\text{locs}(H))^*$ from a to d. If it is not a simple $\sim$-path in $\mathcal{T}(\text{locs}(H))^*$, then it contains loops, which can be removed because $H \in \mathbf{Close}^*$. Then there exists P', Q' where P' is a subsequence of $P_{:j+1}$, Q' is a subsequence of $Q_{i:}$, and $P'Q'$ is a simple $\sim$-path in $\mathcal{T}(\text{locs}(H))^*$ from a to d. Let R be a shortest simple $\sim$-path in $\mathcal{T}(\text{locs}(H))^*$ from a to d. It is sufficient to show that R is a $\sim$-path in H from a to d. Consider a factor J that R crosses, and consider the subsequence R' of R that R traverses J with. Each traversal in R is in J since $J \in \mathbf{Close}^*$. Since R is a simple $\sim$-path, the only location repeats allowed in R' look like $[x \to y][y \to z]$. This must be consecutive in R since loops were taken out, so it can be reduced to $[x \to z]$, shortening R, a contradiction. So R' does not repeat locations. Thus, by the gizmo requirement in the statement, $R' \in J$. So $R \in H$. □

Then, we give a complete characterization of which (m, n)-dicrumblers can simulate all (m, n)-dicrumblers:

Theorem 10. *Given m and n, the (m, n)-dicrumbler can simulate all (p, q)-dicrumblers for integers p, q greater than 0 if and only if $n > m$.*

Proof. $\Leftarrow$: Let G be a (m, n)-dicrumbler, with the entrance of tunnel i labelled a_i and the exit of tunnel i labelled b_i for all integers i where $0 \le i < n$. Let $\mathcal{G}$ be a sequence of 2 copies of G and let $H = \bigotimes \mathcal{G}$. Let $\sim$ be the minimal equivalence relation where $(b_i)_j \sim (a_{i+1})_j$ for all valid i, j where $0 \le i < n-1$, and $(b_{m-1})_j \sim (a_m)_{1-j}$ for all valid j. Finally, let $L = \{(a_0)_0, (b_m)_1, (a_0)_1, (b_m)_0\}$. Note that m is a valid index into the tunnels since $n > m$. The only minimal $\sim$-paths in H between locations that map using $\pi_\sim$ to different locations in L are $X_j := [(a_0)_j \to (b_0)_j] \cdots [(a_{m-1})_j \to (b_{m-1})_j][(a_m)_{1-j} \to (b_m)_{1-j}]$ for $j \in \{0, 1\}$, and they do not share locations equivalent by $\sim$. If X_j is taken, then factor j of H is crossed m times, closing it, and making X_{i-j} untraversable due to the last traversal. Then $H/{\sim}\,|_L$ is a 1-use 2-tunnel dicrumbler. An example is shown in Figure 3-10.

$\Rightarrow$: Assume $n \le m$. Label the locations of the (1, 2)-dicrumbler as in Figure 3-3, except that it is 1-use.

Let $(\mathcal{G}, \sim, L)$ be a simulation of the (1, 2)-dicrumbler with the (m, n)-dicrumbler. Let $H = \bigotimes \mathcal{G}$ and let a', b', c', d' be locations in H where $L(\pi_\sim(a')) = a$ and similarly for the others. There must be a simple $\sim$-path P in H from a' to b' and a simple $\sim$-path Q in H from c' to d', simple because dicrumbler variants are in $\mathbf{Close}^*$. If they intersect, then by Lemma 7 there is a $\sim$-path in H from a' to d', which would

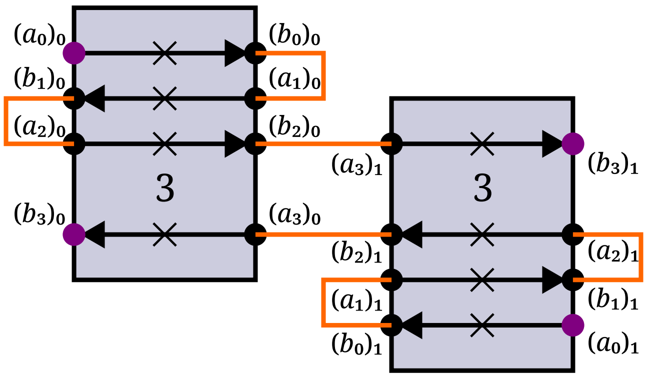


Figure 3-10: A (3, 4)-dicrumbler simulating a (1, 2)-dicrumbler. Orange lines connect equivalent locations, and purple locations are in L.

induce $[a \to d]$, a contradiction, so they cannot intersect. The lemma can be applied because there are at least as many uses as tunnels, meaning that each combination of traversals in the (m, n)-dicrumbler is in the (m, n)-dicrumbler. So P and Q must not intersect. Each gizmo in the simulation can be crossed at most n times combined by P and Q since they are simple and do not intersect. But $m \geq n$, so $PQ \in H$, which induces $[a \to b][c \to d]$, a contradiction. So no such simulation exists. $\square$

In the case where $m \geq n$, simulability gets more complicated. A variant of the proof for Theorem 8 can be used to show that the (m, n)-dicrumbler can simulate the (p, n)-dicrumbler if (but not necessarily only if) m divides p (Combine start locations per tunnel index, and do the same for end locations). The (m, n)-dicrumbler can also simulate the (m, n') dicrumbler for $n' < n$ by simply ignoring tunnels. Adding uses in general, however, is tricky and at least sometimes impossible. In particular, the (2, 2)-dicrumbler cannot simulate the (3, 2)-dicrumbler.

Theorem 11. *The (2, 2)-dicrumbler cannot simulate the (3, 2)-dicrumbler.*

Proof. Let $(\mathcal{G}, \sim, L)$ be a simulation of the (3, 2)-dicrumbler with the (2, 2)-dicrumbler. Let $H = \bigotimes \mathcal{G}$. Label the locations of $H/\!\sim|_L$ according to Figure 3-3 except that it has 3 uses instead of 2. Let $a_0, a_1, b_0, b_1 \in H$ such that $L(\pi_\sim(a_0)) = a$, $L(\pi_\sim(a_1)) = b$, $L(\pi_\sim(b_0)) = c$, and $L(\pi_\sim(b_1)) = d$. Since $H/\!\sim_L$ has 3 uses and dicrumbler variants are in **Close***, there must be 3 simple $\sim$-paths A_1, A_2, A_3 in H from a_0 to a_1 such that $A_1A_2A_3 \in H$ and $A_1A_2A_3$ is as short as possible.

Define an *interacting gizmo* to be a factor of H where some $\sim$-path from a_0 to a_1 and some $\sim$-path from b_0 to b_1 both cross it. By Lemma 7, every $\sim$-path from a_0 to a_1 that crosses some interacting gizmo must cross the same tunnel of that gizmo. If they crossed different ones, then a $\sim$-path from a_0 to a_1 would intersect a $\sim$-path from b_0 to b_1, leading to a supposedly disallowed path being allowed. Using interacting gizmos, we will show that there are 2 simple $\sim$-paths in H from a_0 to a_1 that do not cross the same interacting gizmo. Then doing something similar for b_0 and b_1 will allow traversing the supposed 3-use gizmo $H/\!\sim|_L$ 4 times.

Note that the (2, 2)-dicrumbler is in a simulability class called **OI*** (order-independent forever), which is generated by $ZXY \implies ZYX$. So traversal sequences in H can be arbitrarily reordered. In particular, if $A_1 = X_0G_0X_1G_1X_2$ and $A_2 = X_3G_1X_4G_2X_5$, then they can be reordered into $A_1' = X_0G_0X_1G_1X_4G_2X_5$ and $A_2' = X_3G_1X_2$, and $A_1'A_2'$ would still be in H. This is the kind of traversal reordering that we will do later.

Define a *2-interacting gizmo* to be an interacting gizmo that at least two of A_1, A_2, and A_3 cross. The situation is illustrated in Figure 3-11. Assume the 2-interacting gizmo traversals are transitive, i.e. if J is traversed before K in any of A_1, A_2, A_3, then J is traversed before K in all of those traversal sequences where they both appear. Each pair of 2-interacting gizmos appears in at least one of the traversal sequences (since each relevant gizmo is traversed in at least 2 of the traversal sequences), so there is a total ordering on the relevant gizmos. Let $G_0, G_1, ..., G_{k-1}$ be traversals through 2-interacting gizmos in order. A relevant traversal sequence must contain G_0 and G_1 and some traversal sequence must contain G_1 and G_2. Do a traversal reordering to place G_0, G_1, and G_2 in the same traversal sequence. Continue this until some traversal sequence contains every interacting gizmo. Since each interacting gizmo appears twice, the other 2 traversal sequences (now called A_4 and A_5) combined contain each 2-interacting gizmo once and thus do not share 2-interacting gizmos. They in fact do not share any interacting gizmos, because interacting gizmos that are not 2-interacting gizmos are crossed only once by $A_1A_2A_3$. They are also $\sim$-paths in H from a_0 to a_1, since the traversal reorderings done preserved that property for the 3 individual traversal sequences.

If the 2-interacting gizmo traversals are not transitive, there exists traverals J and K through separate 2-interacting gizmos and traversal sequences $X, Y \in \{A_1, A_2, A_3\}$ where JK is a subsequence of X and KJ is a subsequence of Y. Then $X = Z_0JZ_1KZ_2$ and $Y = Z_3KZ_4JZ_5$ for some 6-traversal sequence Z. A traversal reordering creates $X' = Z_0JZ_1KZ_4JZ_5$ and $Y' = Z_3KZ_2$. But then X' can be shortened to Z_0JZ_5, a contradiction since $A_1A_2A_3$ was as short as possible. So the 2-interacting gizmo traversals must be transitive.

A similar process can be performed to construct B_4 and B_5, two $\sim$-paths in H from b_0 to b_1 that do not share interacting gizmos. Then in $H[A_4A_5]$, every interacting gizmo has at least one use left, and since B_4B_5 crosses every interacting gizmo at most once, then $A_4A_5B_4B_5 \in H$, extracting 4 uses out of a 3-use dicrumbler variant, a contradiction. So no such simulation exists. $\square$

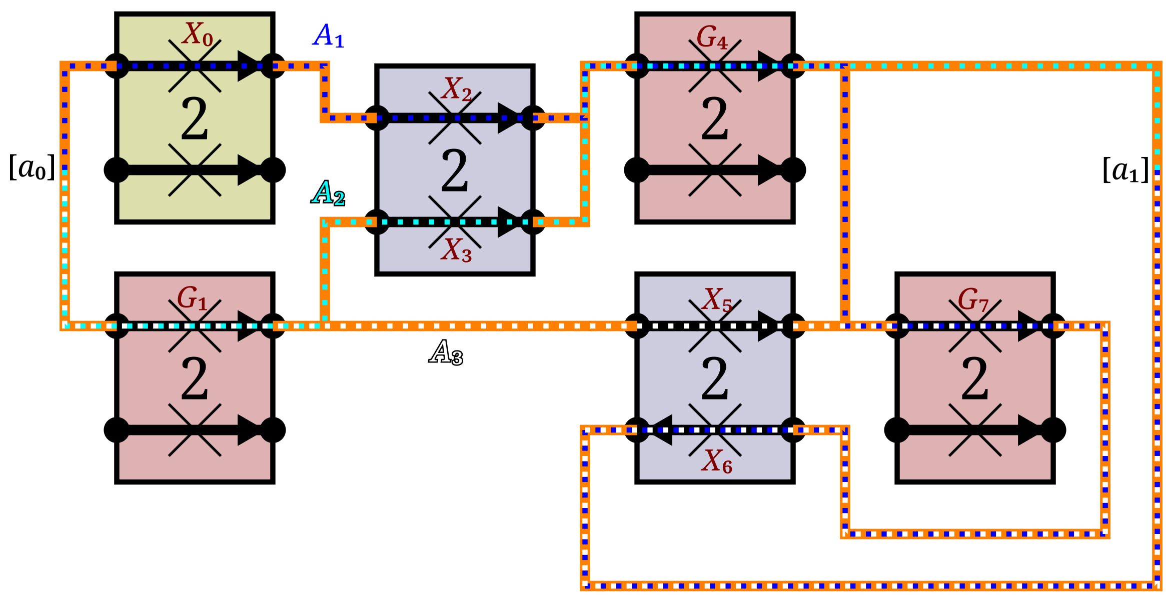


Figure 3-11: An example situation from the proof of Theorem 11. Orange lines connect locations equivalent by $\sim$. $[a_0] = \pi_\sim(a_0)$ and $[a_1] = \pi_\sim(a_1)$. 2-interacting gizmos are tinted red, and other interacting gizmos are tinted yellow. Traversals are labelled dark red, with ones for 2-interacting gizmos being called G_i instead of X_i. Note that $A_1 = X_0 Z_2 G_4 G_7 X_6$, $A_2 = G_1 X_3 G_4$, and $A_3 = G_1 X_5 G_7 X_6$. With the traversal reorderings mentioned in the proof, a $\sim$-path $G_1 X_3 G_4 G_7 X_6$ goes through all 2-interacting gizmos, leaving $A_4 = X_0 Z_2 G_4$ and $A_5 = G_1 X_5 G_7 X_6$.

Chapter 4

Universality

In the previous section, we show that some gizmos cannot simulate some other gizmos. Here, we will show that some gizmos can simulate a lot of other gizmos. In fact, they can simulate every gizmo that has a certain property. The motivation for this is that simulability classes can be equated using universal gizmos. To show that two simulability classes A and B equal, it is sufficient to show that some gizmo that simulates every gizmo in A is in B, and some gizmo that simulates every gizmo in B is in A.

First, note that there is no gizmo that can simulate every gizmo, even the ones outside **Reg**.

Theorem 12. *There is no gizmo that can simulate every gizmo.*

Proof. The cardinality of the number of gizmos is $2^{\aleph_0}$, because the cardinality of the number of traversal sequences constructible from a (finite) set of locations is $\aleph_0$, and a gizmo is a set of traversal sequences. Given a gizmo G, the cardinality of the number of simulations that can be constructed using G is $\aleph_0$, because the cardinality of the numbers of copies of G that can be made is $\aleph_0$, the cardinality of the number of equivalence relations (partitions of locations) is finite, since the number of locations is finite, and the cardinality of the number of subgizmos is also finite for the same reason. Since the number of gizmos has a higher cardinality than the number of simulations constructible from G, there must be some gizmo that G cannot simulate. $\square$

4.1 Reg

For the simulability class **Reg**, which, as mentioned before, contains only gizmos that have a finite number of reachable states, there is in fact a gizmo in **Reg** that can simulate all members of **Reg**: the product of the open door and the closed door. The proof will be very similar to the one in [1], but adapted to gizmos.

Theorem 13. *The product G of the open door and the closed door can simulate G' for all $G' \in \mathbf{Reg}$.*

Proof. The product can easily simulate the open door O and the closed door C shown in Figure 2-3, so we will use them for the proof.

As an overview, the simulation

- constructs a copy of G for each location in G' (the location doors), a copy for each combination of location and reachable state in G' (the location-state doors), and a copy for each transition in the DFA of G' that doesn't lead to the non-accepting state (the transition doors),
- connects locations to force certain paths through the simulation, so that an agent must
  - traverse the location-state door corresponding to the current state A of G' and some location a,
  - open the transition door corresponding to a chosen transition from A labelled $[a \to b]$,
  - open the location door corresponding to location b,
  - close all location-state doors,
  - traverse and close the transition door that was opened,
  - open the location-state doors corresponding to state $A[[a \to b]]$, and
  - traverse and close the location door that was opened, ending at simulated location b.

Create a sequence $\mathcal{G}$ of gizmos: a gizmo for each location a in G' labelled a, a gizmo for each location a and reachable state A in G' labelled (a, A), and a gizmo for each transition $A \to B$ labelled $[a \to b]$ of the DFA of G', which said gizmo being labelled $([a \to b], A \to B)$, but only for those transitions that do not lead to the non-accepting state. Let $H := \bigotimes \mathcal{G}$. Each gizmo labelled (a, G') for some location $a \in G'$ is a copy of O, and all others are copies of C.

Let there be an ordering $\prec$ of the combinations of locations and states in G', which $\mathfrak{S}$ being the first combination and $\mathfrak{T}$ being the last one. Also let there be an ordering also called $\prec$ of the locations of G', with s being the first location and t being the last one.

Let $\sim$ be the minimal equivalence relation of $\mathrm{locs}(H)$ where:

- For reachable states A, B in G', for locations a, b in G': $u_{a,A} \sim o_{[a\to b],A\to B}$ if $A[[a \to b]] = B$
- For reachable states A, B in G', for locations a, b in G': $p_{[a\to b],A\to B} \sim o_b$ if $A[[a \to b]] = B$

- For location a in G': $p_a \sim c_{\mathfrak{S}}$
- For reachable states A, B in G', for locations a, b in G': $d_{a,A} \sim c_{b,B}$ if (a, A) immediately precedes (b, B) under $\prec$
- For reachable states A, B in G', for locations a, b in G': $d_{\mathfrak{T}} \sim t_{[a\to b],A\to B}$ and $u_{[a\to b],A\to B} \sim c_{[a\to b],A\to B}$ if $A[[a \to b]] = B$
- For reachable states A, B in G', for locations a, b in G': $d_{[a\to b],A\to B} \sim o_{s,B}$ if $A[[a \to b]] = B$
- For reachable state A in G', for locations a, b in G': $p_{a,A} \sim o_{b,A}$ if a immediately precedes b under $\prec$
- For reachable state A in G', for location a in G': $p_{t,A} \sim t_a$ and $u_a \sim c_a$
- For reachable state A in G', for reachable location a in G': $d_a \sim t_{a,A}$

An example is shown in Figure 4-1.

Let $L = \{\pi_\sim(d_a) \mapsto a \mid a \in \text{locs}(G')\}$.

Same-location traversals in O and C do not do anything. Because of how the locations are connected by $\sim$, and because closed doors cannot be traversed, a nonempty minimal $\sim$-path in H between locations that are equivalent by $\sim$ to locations in L must look like the following: $X_{[a\to b],A\to B} := [t_{a,A} \to u_{a,A}][o_{[a\to b],A\to B} \to p_{[a\to b],A\to B}][o_b \to p_b][c_{\mathfrak{S}} \to d_{\mathfrak{S}}]\cdots[c_{\mathfrak{T}} \to d_{\mathfrak{T}}][t_{[a\to b],A\to B} \to u_{[a\to b],A\to B}][c_{[a\to b],A\to B} \to d_{[a\to b],A\to B}][o_{s,B} \to p_{s,B}]\cdots[o_{t,B} \to p_{t,B}][t_b \to u_b][c_b \to d_b]$, where $a, b \in \text{locs}(G')$ and $A, B \in \text{states}(G')$ such that $A[[a \to b]] = B$, with possible additional copies for different values of a, b, A, and B that still satisfy the condition. Note that after $X_{[a\to b],A\to B}$, the gizmos labelled (c, B) for each location $c \in G'$ are open and the rest are closed, so the next traversal sequence must start $[t_{c,B} \to u_{c,B}]$.

If a traversal sequence $[a_0 \to b_0]\cdots[a_{n-1} \to b_{n-1}]$ is allowed in G' for some sequences a, b of locations in G', then an isomorphic traversal sequence induced by the sequence of $\sim$-paths $X_{[a_0\to b_0],G'\to G'[[a_0\to b_0]]}\cdots$ $X_{[a_{n-1}\to b_{n-1}],G'[[a_0\to b_0]\cdots[a_{n-2}\to b_{n-2}]]\to G'[[a_0\to b_0]\cdots[a_{n-1}\to b_{n-1}]]}$ is allowed in $H/\sim|_L$. If a traversal sequence is allowed in $H/\sim|_L$, it must be induced by $X_{[a_0\to b_0],A_0\to A_1}\cdots$ $X_{[a_{n-1}\to b_{n-1}],A_{n-1}\to A_n}$ for sequences a, b of locations in G' and a sequence A of states in G', where $A_i[[a_i \to b_i]] = A_{i+1}$ for each valid index i and $A_0 = G'$. Then $[a_0 \to b_0]\cdots[a_{n-1} \to b_{n-1}] \in G'$. So $H/\sim|_L = G'$, completing the simulation. $\square$

Another simulability class with a universal member is **DAG**. Specifically, we will show that the 2-use mismatched dicrumblers (Figure 4-2), which is in **DAG**, can simulate every gizmo in **DAG**. First, it simulates the matched dicrumblers (the (1, 2)-dicrumbler), which then simulates the n-tunnel matched dicrumblers as shown in Theorem 9. This will be an important helper gizmo.

Lemma 8. The 2-use mismatched dicrumblers can simulate the matched dicrumblers.

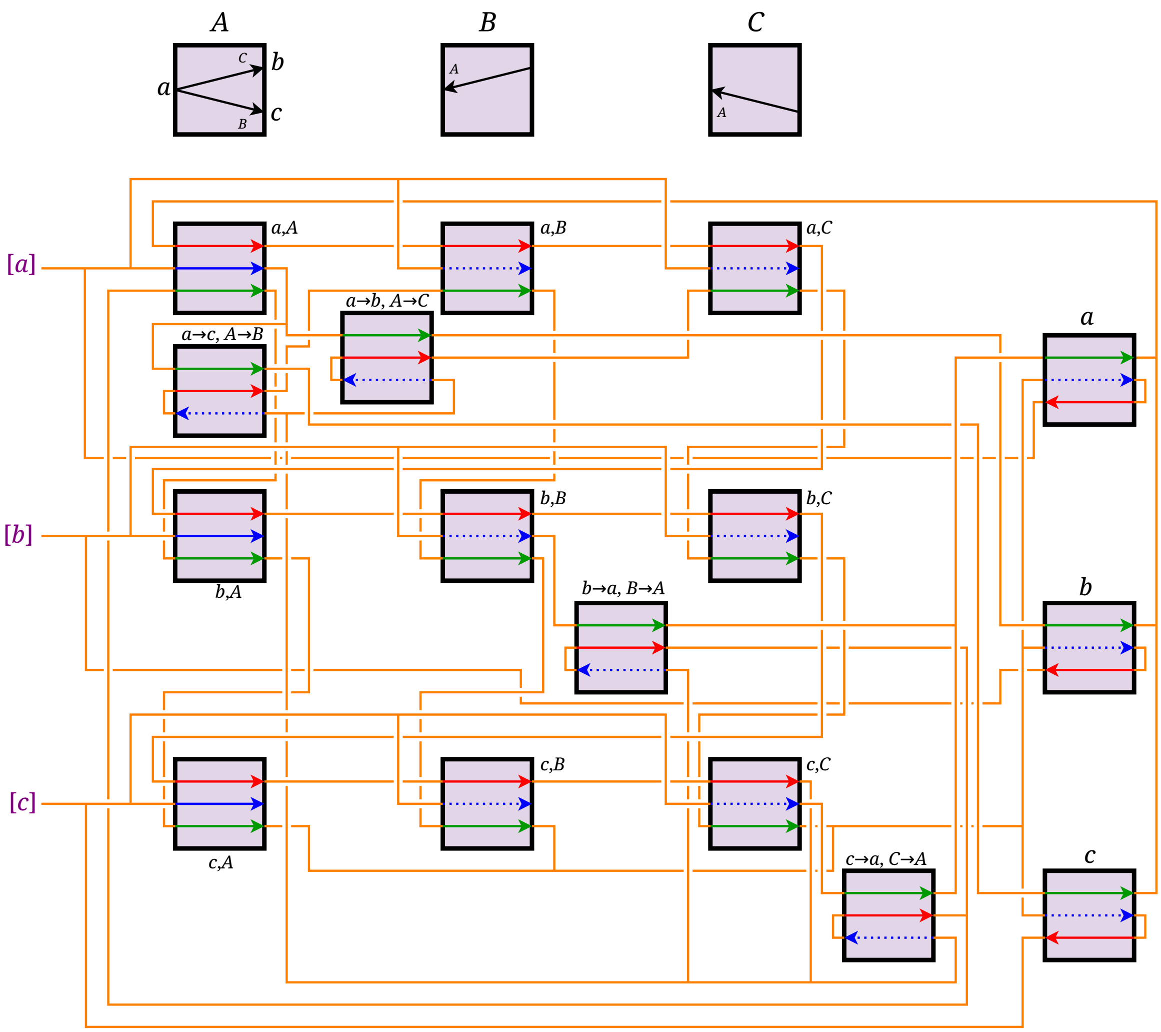


Figure 4-1: The open and closed doors simulating an example gizmo shown at the top, with its resulting states labelled on each traversal. The locations $[a]$, $[b]$, and $[c]$ are $\pi_\sim(d_a)$, $\pi_\sim(d_b)$, and $\pi_\sim(d_c)$, respectively. The gizmos are labelled with the labels used in the proof. This is based on Figure 4 of [1].

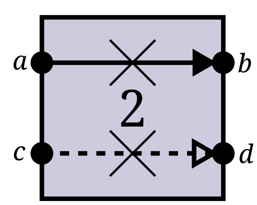


Figure 4-2: The *2-use mismatched dicrumblers* G. Similar to the mismatched dicrumblers, but can be crossed only twice before it closes. $G = \mathrm{rtp}([a \to b][c \to d])$

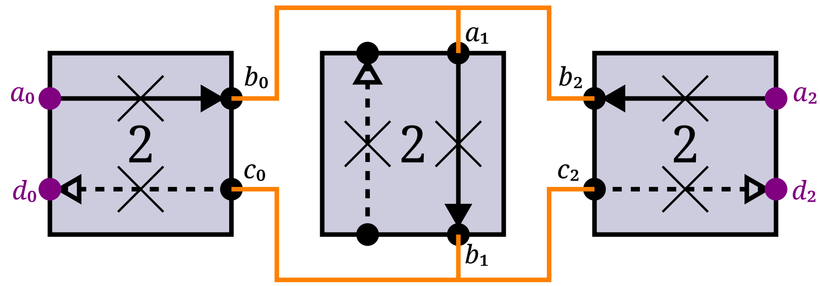


Figure 4-3: The 2-use mismatched dicrumblers simulating the matched dicrumblers. Equivalent locations are connected with orange lines, and locations whose equivalent class is in L are purple.

Proof. Let $\mathcal{G}$ be a sequence of three 2-use mismatched dicrumblers. Let $\sim$ be the minimal equivalence relation in $\mathrm{locs}(\bigotimes \mathcal{G})$ where $b_0 \sim a_1 \sim b_2$ and $c_0 \sim b_1 \sim c_2$. Let L map $\{\pi_\sim(a_0), \pi_\sim(d_0), \pi_\sim(a_2), \pi_\sim(d_2)\}$. The only minimal $\sim$-paths in $\bigotimes \mathcal{G}$ between locations that are equivalent to ones in L are $X_i := [a_i \to b_i][a_1 \to b_1][c_i \to d_i]$ for $i \in \{0, 2\}$. When X_i is taken, $[a_1 \to b_1]$ closes, closing both X_0 and X_2. No new minimal $\sim$-paths between locations equivalent to ones in L appear. Therefore, $\bigotimes \mathcal{G}/\sim|_L$ is a matched dicrumblers. The simulation is shown in Figure 4-3. □

4.2 DAG

Theorem 14. *The 2-use mismatched dicrumblers G can simulate $G' \in$ **DAG**.*

Proof. The simulation will use n-tunnel matched dicrumblers for arbitrary n, since the 2-use mismatched dicrumblers can simulate them (Lemma 8, Theorem 9).

As an overview, the simulation

- constructs a copy of G for each combination of location and reachable state in G' (the location-state gizmos), a copy for each transition in the DFA of G' that doesn't lead to the non-accepting state (the transition gizmos), and a $|\mathrm{locs}(G')|$-tunnel matched dicrumblers for each reachable state in G' (the state dicrumblers).
- connects locations to force certain paths through the simulation, so that an agent must
  - cross the second tunnel of the location-state gizmo corresponding to the current state A of G' and some location a,
  - cross the state dicrumblers corresponding to state A,
  - cross the first tunnel of the transition gizmo corresponding to a chosen transition from A labelled $[a \to b]$,

- cross the first tunnels of all the location-state gizmos corresponding to state $A[[a \to b]]$, and
- cross the second tunnel of the transition gizmo whose first tunnel got crossed.

- enforces state transitions using the state dicrumblers.

An example is shown in Figure 4-4.

Create a sequence $\mathcal{G}$ of gizmos: a copy of G for each location p and reachable state P in G' labelled (p, P), a copy of G for each transition $P \to Q$ labelled $[p \to q]$ of the DFA of G', which said gizmo being labelled $([p \to q], P \to Q)$, but only for those transitions that do not lead to the non-accepting state, and a $|\text{locs}(G')|$-tunnel matched dicrumblers for each reachable state P in G' labelled P. The entrance locations of the multi-tunnel matched dicrumblers will be labelled a_p and the respective exits labelled b_p for $p \in \text{locs}(G')$.

Let $H := \bigotimes \mathcal{G}$. Each gizmo labelled (p, G') for some location $p \in G'$ is set to state $G[[p \to q]]$, since that state is just a dicrumbler and can be easily simulated with the 2-use mismatched dicrumblers.

Let there be an ordering called $\prec$ of the locations of G', with s being the first location and t being the last one.

Let $\sim$ be the minimal equivalence relation of $\text{locs}(H)$ where:

- For reachable state P in G', for location p in G': $d_{p,P} \sim (a_p)_P$.
- For reachable states P, Q in G', for locations p, q in G': $(b_p)_P \sim a_{[p\to q],P\to Q}$ if $P[p \to q] = Q$.
- For reachable states P, Q in G', for locations p, q in G': $b_{[p\to q],P\to Q} \sim a_{s,Q}$ if $P[p \to q] = Q$.
- For reachable state P in G', for locations p, q in G': $b_{p,P} \sim a_{q,P}$ if p immediately precedes q according to $\prec$.
- For reachable states P, Q, R in G', for locations p, q in G': $b_{t,R} \sim c_{[p\to q],P\to Q}$ if $P[p \to q] = Q$.
- For reachable states P, Q, R in G', for locations p, q in G': $d_{[p\to q],P\to Q} \sim c_{q,R}$ if $P[p \to q] = Q$.

Let $L = \{\pi_\sim(c_{p,G'}) \mapsto p \mid p \in \text{locs}(G')\}$.

Because of how the locations are connected by $\sim$, a nonempty minimal $\sim$-path in H between locations that are equivalent by $\sim$ to locations in L without same-location traversals must start with the following: $X_{[p\to q],P\to Q} := [c_{p,P} \to d_{p,P}][(a_p)_P \to (b_p)_P][a_{[p\to q],P\to Q} \to b_{[p\to q],P\to Q}][a_{s,Q} \to b_{s,Q}] \cdots [a_{t,Q} \to b_{t,Q}][c_{[p\to q],P\to Q} \to d_{[p\to q],P\to Q}]$ where $b, q \in \text{locs}(G')$ and $P, Q \in \text{states}(G')$ such that $P[[p \to q]] = Q$. In the starting state, $P = G'$, since the $[c \to d]$ tunnels of only the p, G' gizmos are traversable.

After traversing $X_{[p\to q],P\to Q}$, the multi-tunnel matched dicrumblers corresponding to state P closes, blocking any traversal sequences of the form $X_{[i\to j],P\to K}$ for $i, j \in \text{locs}(G')$ and $K \in \text{states}(G')$. In addition, the $[c \to d]$ tunnels of the i, Q gizmos are traversable for $i \in \text{locs}(G')$, allowing $X_{[i\to j],Q\to R}$ for $i, j \in \text{locs}(G')$ and $R \in \text{states}(G')$ if the multi-tunnel matched dicrumblers corresponding to R is still open. Thus, minimal $\sim$-paths between locations equivalent to ones in L must look like $X_{[p_0\to q_0],P_0\to P_1} \cdots X_{[p_{n-1}\to q_{n-1}],P_{n-1}\to P_n}$, where p, q are sequences of locations in G' and P is a sequence of *distinct* reachable states in G' such that $P_i[[p_i \to q_i]] = P_{i+1}$.

If a traversal sequence $S = [p_0 \to q_0] \cdots [p_{n-1} \to q_{n-1}]$ is allowed in G' for some sequences p, q of locations in G', then an isomorphic traversal sequence induced by the sequence of $\sim$-paths $X_{[p_0\to q_0],G'\to G'[[p_0\to q_0]]} \cdots$
$X_{[p_{n-1}\to q_{n-1}],G'[[p_0\to q_0]\cdots[p_{n-2}\to q_{n-2}]]\to G'[[p_0\to q_0]\cdots[p_{n-1}\to q_{n-1}]]}$ is allowed in $H/\sim|_L$, when traversals $[p_i \to q_i]$ where $p_i = q_i$ and that are unnecessary are skipped. This is because according to Theorem 5, traversals in S either change the state of G' to one not seen before, or are same-location traversals that do not change the state, in which case they are unnecessary. If a traversal sequence is allowed in $H/\sim|_L$, it must be induced by $X_{[p_0\to q_0],p_0\to p_1} \cdots X_{[p_{n-1}\to q_{n-1}],p_{n-1}\to p_n}$ for sequences p, q of locations in G' and a sequence A of states in G', where $p_i[[p_i \to q_i]] = p_{i+1}$ for each valid index i and $p_0 = G'$. Then $[p_0 \to q_0] \cdots [p_{n-1} \to q_{n-1}] \in G'$. So $H/\sim|_L$ is isomorphic to G', completing the simulation. $\square$

4.3 Bottom Universality

Universality is a gizmo simulating a whole class of gizmos. The concept of a class of gizmos simulating a specific gizmo is also interesting, but harder to find results for. This concept is called *bottom universality*, as it can be thought of as a class of gizmos simulating one below them, rather than a gizmo simulating a class of gizmos below it. It has been explored before, for example in [4], where Demaine et al. that all reversible deterministic gadgets with interacting tunnels simulates the locking 2-toggle. (This uses the old model of "gadget", defined by states and transitions instead of traversal sequences, and the concept of "deterministic" doesn't translate well.) This section lists some partial results for bottom universality.

First, we give a partial characterization of which gizmos can simulate the directed crumbler.

There are 3 important implication properties that the dicrumbler doesn't satisfy:

- **DirBlind** ($X \implies X^{-1}$)
- **Reuse** ($X \implies XX$)
- **Undo** ($X \implies XX^{-1}$)

Since the implication properties induce simulability classes, no gizmo with any of these properties can simulate the dicrumbler.

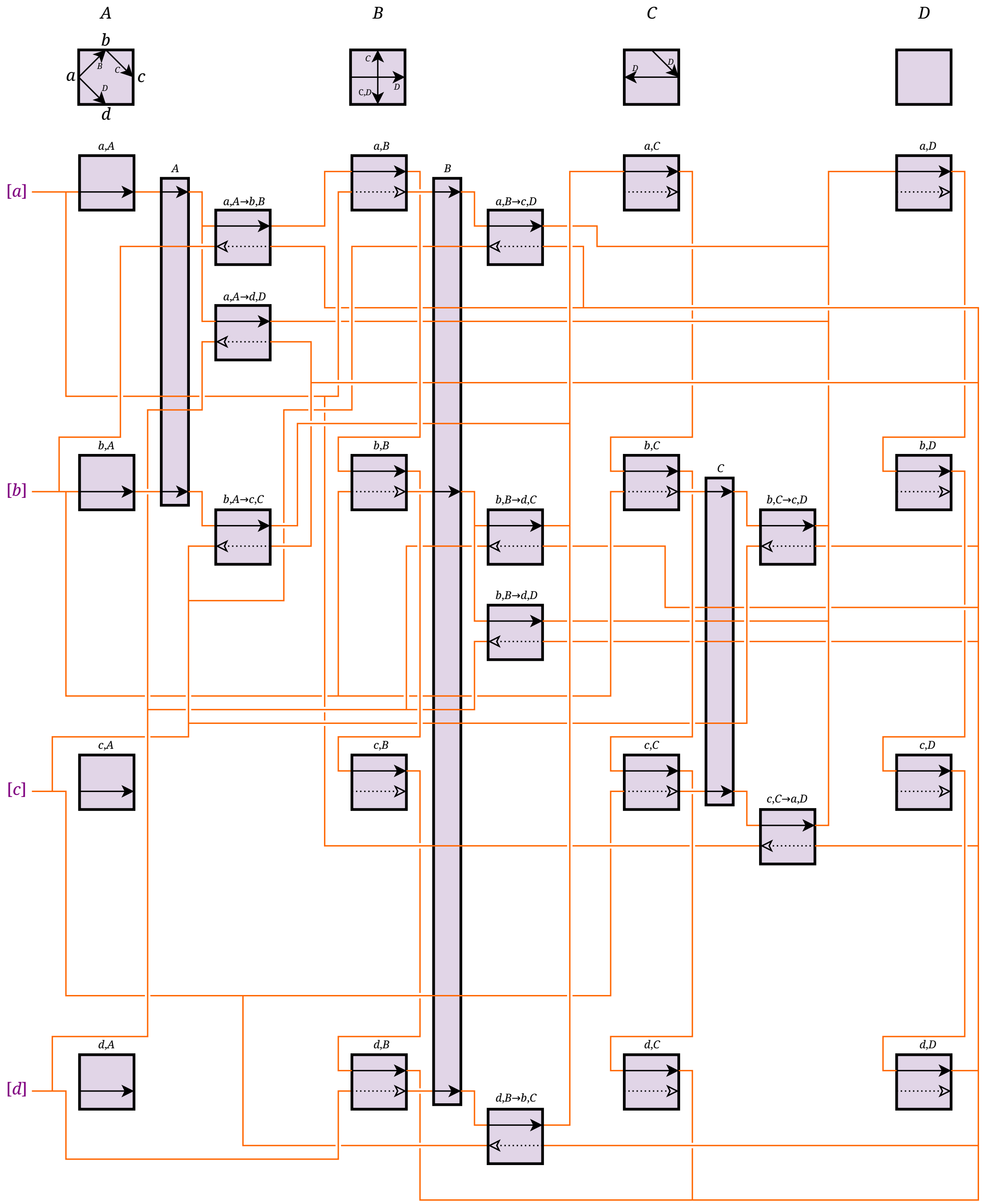


Figure 4-4: The 2-use mismatched dicrumblers simulating an example gizmo G shown at the top, with its resulting states labelled on each traversal. The locations $[a]$, $[b]$, $[c]$, and $[d]$ are $\pi_{\sim}(c_{a,G})$, $\pi_{\sim}(c_{b,G})$, $\pi_{\sim}(c_{c,G})$, and $\pi_{\sim}(c_{d,G})$, respectively. The gizmos are labelled with the labels used in the proof.

Definition 8. Let (F, g) be an implication property where F_i and g take 1 traversal sequence as input. For a gizmo G, a traversal sequence X *breaks* (F, g) if $F_i(X) \in G$ for all valid indices i, but $g(X) \notin G$. If A is a simulability class induced by (F, g), then X *breaks* A if X breaks (F, g).

A gizmo G is *buttonless* if, for all $X \in G$ and for all $a \in \text{locs}(G)$, $G[X] = G[X[a \to a]]$.

Lemma 9. Let G be a buttonless gizmo on tunnels. Let X be a traversal sequence that breaks $X \implies X^{-1}$, breaks $X \implies XX$, or breaks $X \implies XX^{-1}$ and does not repeat locations. Then G simulates a gizmo H with a traversal $[a \to b]$ that breaks breaks $X \implies X^{-1}$, breaks $X \implies XX$, or breaks $X \implies XX^{-1}$, respectively.

Proof. This is proven by induction on the number of traversals in X

If X is a single traversal, then let $H = G$ and $[a \to b] = X$, and we are done.

Otherwise, let $X = [c_0 \to d_0] \cdots [c_{n-1} \to d_{n-1}]$ for location sequences c and d. Let $\sim$ be the minimal equivalence relation where $d_i \sim c_{i+1}$ for all valid indices i. Let $\tilde{G} = G/\sim$, let $\tilde{c_i} = \pi_\sim(c_i)$ where $0 \le i < n$, and let $\tilde{c_n} = \pi_\sim(d_{n-1})$. Since G is on tunnels, for traversal sequences $Y, Z \in \tilde{G}$:

- $Y[\tilde{c_0} \to x]Z \in \tilde{G} \implies x \in \{\tilde{c_0}, \tilde{d_0}\}$
- $Y[\tilde{c_i} \to x]Z \in \tilde{G} \implies x \in \{\widetilde{c_{i-1}}, \tilde{c_i}, \widetilde{c_{i+1}}\}$, where $0 < i < n$
- $Y[\tilde{c_n} \to x]Z \in \tilde{G} \implies x \in \{\widetilde{c_{n-1}}, \tilde{c_n}\}$

For this proof, if A is a traversal sequence in G, then $\tilde{A} = (\pi_\sim)^*_{\mathcal{T}}(A)$.

We will show that the traversal that $\tilde{X}$ transitively closes to $([\tilde{c_0} \to \tilde{c_n}])$ breaks the same implication property in $\tilde{G}$ that X breaks in G.

If X breaks $X \implies X^{-1}$, then $[d_{n-1} \to c_{n-1}] \cdots [d_0 \to c_0] \notin G$, and it must be proven that $\tilde{Z} = [\tilde{c_n} \to \tilde{c_0}] \notin \tilde{G}$. Assume this is false. Then $\tilde{Z}$ must be derivable from a $\sim$-path Z from d_{n-1} to c_0 in G by transitive closure. Since G is buttonless, same-location traversals can be removed from Z. Since G is on tunnels and X does not repeat locations, Z must be $[d_{n-1} \to c_{n-1}] \cdots [d_0 \to c_0]$ with possible traversals from $[c_i \to d_i]$ in the middle. If there is such a "backwards" traversal, there must be a first traversal $[c_i \to d_i]$ in Z. But then $[d_i \to c_i][c_i \to d_i]$ is a substring of Z. By transitive closure, turn this into $[d_i \to d_i]$. Then remove it since G is buttonless, shrinking Z, and apply this process until it cannot be applied anymore.

If X breaks $X \implies XX$, then $[c_0 \to d_0] \cdots [c_{n-1} \to d_{n-1}][c_0 \to d_0] \cdots [c_{n-1} \to d_{n-1}] \notin G$, and it must be proven that $\tilde{Z} = [\tilde{c_0} \to \tilde{c_n}][\tilde{c_0} \to \tilde{c_n}] \notin \tilde{G}$. The same proof as above can be applied here, with the caveat that Z is a concatenation of 2 $\sim$-paths from c_0 to d_{n-1}.

If X breaks $X \implies XX^{-1}$, then $[c_0 \to d_0] \cdots [c_{n-1} \to d_{n-1}][d_{n-1} \to c_{n-1}] \cdots [d_0 \to c_0] \notin G$, and it must be proven that $\tilde{Z} = [\tilde{c_0} \to \tilde{c_n}][\tilde{c_n} \to \tilde{c_0}] \notin \tilde{G}$. The same proof

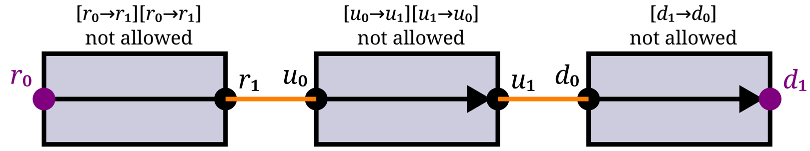


Figure 4-5: Simulation of a dicrumbler with traversals that break **DirBlind**, **Reuse**, and **Undo**. Orange lines connect equivalent locations, and purple locations are in L.

as the one for $X \implies X^{-1}$ can be applied here, with the caveat that Z is a $\sim$-path from c_0 to c_0 that touches d_{n-1}.

Thus, $[\tilde{c_0} \to \tilde{c_n}]$ is a traversal in $\tilde{G}$ that breaks $X \implies X^{-1}$, breaks $X \implies XX$, or breaks $X \implies XX^{-1}$. □

Theorem 15. *Let G be a buttonless gizmo on tunnels. Let D, R, and U be a traversal sequences in G that break* **DirBlind**, **Reuse**, *and* **Undo**, *respectively, and do not repeat locations. Then G simulates a dicrumbler.*

Proof. Use the lemma above to construct gizmos G_d, G_r, and G_u which have traversals $[d_0 \to d_1]$, $[r_0 \to r_1]$, and $[u_0 \to u_1]$ that break **DirBlind**, **Reuse**, and **Undo**, respectively. Construct gizmo $C_\to = \bigotimes(G_d, G_r, G_u)/\sim|_L$, where $\sim$ is the minimal equivalence relation where $r_1 \sim u_0$ and $u_1 \sim d_0$, and L maps $\{\pi_\sim(r_0), \pi_\sim(d_1)\}$. It is enough to show that the only minimal $\sim$-path in $H := \bigotimes(G_d, G_r, G_u)$ between two different locations equivalent to locations in L is $X := [r_0 \to r_1][u_0 \to u_1][d_0 \to d_1]$, and after X is traversed, no other such $\sim$-paths can be taken.

- $[d_1 \to d_0] \notin H$ because $[d_0 \to d_1]$ breaks **DirBlind**.
- $[r_0 \to r_1] \notin H[X]$ because $[r_0 \to r_1]$ breaks **Reuse**.
- $[u_1 \to u_0] \notin H[X]$ because $[u_0 \to u_1]$ breaks **Undo**.

So H has only X, and $H[X]$ has no $\sim$-paths between r_0 and d_1. So $C_\to$ is a dicrumbler.

The simulation is shown in Figure 4-5. □

Unfortunately, if almost any restriction (on tunnels, no repeating locations) is removed, this is not true. The 2-use dicrumbler with entrance a and exit b has a traversal sequence $[a \to b][a \to b]$ that breaks **Reuse**, and a traversal sequence $[a \to b]$ that breaks **DirBlind** and **Undo**. However, the 2-use dicrumbler cannot simulate the dicrumbler (Theorem 8). Without the tunnels restriction, the other restriction is meaningless because two ports can just always be connected by a traversal.

Chapter 5

Undecidability

The gizmos described so far were all in **Reg** and had a finite number of states. Reachability in mazes made of gizmos in **Reg** must be in PSPACE. This is because the amount of space required to store the state of the maze (agent location combined with a combination of the states of all the gizmos) is polynomial in the number of locations in the maze, the number of gizmos, and the number of states per gizmo, proving NPSPACE membership, and NPSPACE = PSPACE [9]. However, gizmos outside of **Reg** have no such restriction, and reachability with mazes made with them can be undecidable.

An example of a gizmo outside of **Reg** is the *inc-dec-jz* gizmo, shown in Figure 5-1. This gizmo acts like a counter. By taking different traversals, it can be incremented or decremented (but not below 0). The counter can also be checked with a branching traversal. Thus, it can be shown that reachability in a maze full of value-0 inc-dec-jz gizmos is undecidable by reducing from a counter machine. It is known that the halting problem with a counter machine with 3 counters that start at 0 and can each be incremented, decremented, and checked for equality to 0 with a branch, is undecidable [7].

Before proving undecidability with the inc-dec-jz gizmo, it is necessary to construct some helper gizmos, allowing the arbitrary duplication of the increment tunnel, the decrement tunnel, and the branch. In particular, the *value-n inc^i-dec^d-jz^j* gizmo is like a value-n inc-dec-jz gizmo except that it has i increment tunnels, d decrement tunnels, and j branches.

Lemma 10. Given a, b, c, the value-0 inc-dec-jz gizmo can simulate the value-0 inc^a-dec^b-jz^c gizmo.

Proof. First, we will show that the increment and decrement tunnels can be arbitrarily duplicated. This uses a method similar to Lemma 8, where a gizmo is crossed and then the gizmo that was crossed must be the one that is crossed to leave. The increment tunnel will be duplicated, and the proof that the decrement tunnel can be

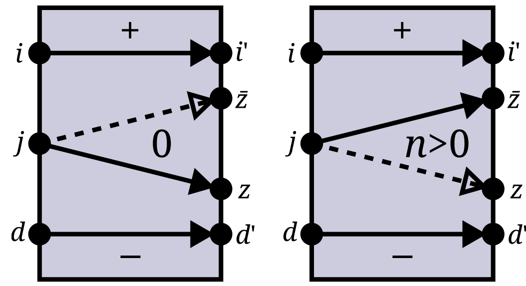


Figure 5-1: The *value-0 inc-dec-jz* gizmo G_0 (left), and the *value-n inc-dec-jz* gizmo G_n (right), where $n > 0$. These gizmos store a value-n. Traversing $[i \to i']$ adds 1 to n, and traversing $[d \to d']$ subtracts 1 if $n > 0$. $[j \to z] \in G_0$, and $[j \to \overline{z}] \in G_n$ for $n > 0$. The $[i \to i']$ traversal is called the *increment* tunnel and traversing it is called *incrementing* the gizmo. The $[d \to d']$ traversal is called the *decrement* tunnel and traversing it is called *decrementing* the gizmo. The $[j \to z]$ and $[j \to \overline{z}]$ traversals combined are called the *branch*.

duplicated is similar. The proof that both can be duplicated is just a combination of the individual proofs.

Construct a sequence $\mathcal{G}$ of $n+1$ value-0 inc-dec-jz gizmos. Gizmos 0 to $n-1$ will power the tunnel duplicated, and gizmo n will contain the tunnel to be duplicated. Let $H = \bigotimes \mathcal{G}$ and let $\sim$ be the minimal equivalence relation in locs(H) where for k where $0 \leq k < n$:

- $i'_k \sim i_n$
- $i'_n \sim j_k$
- $\overline{z}_k \sim d_k$

and let $L = \{\pi_\sim(j_n), \pi_\sim(z_n), \pi_\sim(\overline{z}_n), \pi_\sim(d_n), \pi_\sim(d'_n)\} \ \cup \ \bigcup_{k=0}^{n-1}\{\pi_\sim(i_k), \pi_\sim(d'_k)\}$.

A nonempty minimal $\sim$-path in H between locations that are equivalent to ones in L must start with $[d_n \to d'_n]$ or $[j_n \to z_n]$ or $X_k = [i_k \to i'_k][i_n \to i'_n][j_n \to \overline{z}_n][d_n \to d'_n]$. After taking X_k, gizmo n will be incremented, gizmo k will be returned to its initial state, all other gizmos will be untouched, and X_k will still be traversable. So $H/\!\sim|_L$ is a value-0 inc^n-dec^1-jz^1 gizmo. An example is found in Figure 5-2.

Now we will show that the branch can be arbitrary duplicated, specifically that the value-0 inc^a-dec^b-jz^1 gizmo can simulate the value-0 inc^a-dec^b-jz^c gizmo.

Label the entrances of the increment tunnels of an inc^a-dec^b-jz^1 gizmo i_0 through i_{a-1}, the exits i'_0 through i'_{a-1}, the entrances of the decrement tunnels d_0 through d_{b-1}, and the exits d'_0 through d'_{b-1}. Construct a sequence $\mathcal{G}$ of c value-0 inc^a-dec^b-jz^1 gizmos. They will be wired so that increment and decrement paths cross all the gizmos so they stay in sync, but branches read from only one gizmo. Let $H = \bigotimes \mathcal{G}$ and let $\sim$ be the minimal equivalence relation in locs(H) where for k where $0 \leq k < c-1$:

- For m where $0 \leq m < a$: $(i'_m)_k \sim i_{k+1}$.

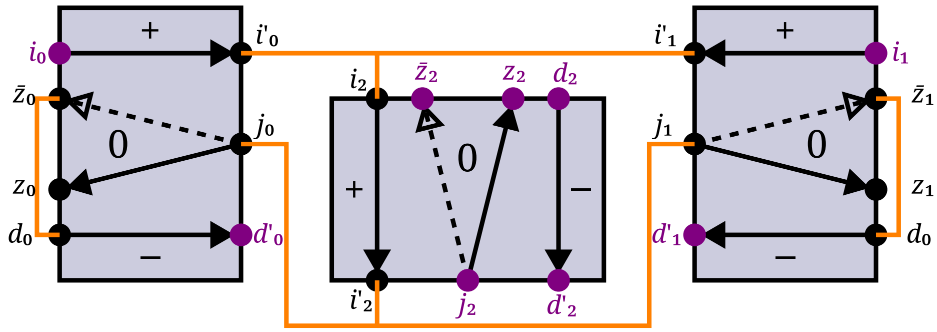


Figure 5-2: The value-0 inc-dec-jz gizmo simulating the value-0 inc^2-dec^1-jz^1 gizmo. Orange lines connect equivalent locations, and purple locations are in L.

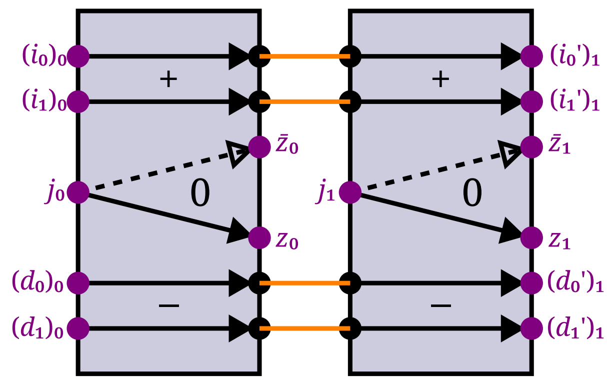


Figure 5-3: The value-0 inc^2-dec^2-jz^1 gizmo simulating the value-0 inc^2-dec^2-jz^2 gizmo. Orange lines connect equivalent locations, and purple locations are in L.

- For m where $0 \le m < b$: $(d'_m)_k \sim d_{k+1}$.

and let L map $\bigcup_{m=0}^{a-1}\{\pi_\sim((i_m)_0), \pi_\sim((i'_m)_{c-1})\} \cup \bigcup_{m=0}^{b-1}\{\pi_\sim((d_m)_0), \pi_\sim((d'_m)_{c-1})\} \cup \bigcup_{m=0}^{c-1}\{\pi_\sim(j_m), \pi_\sim(z_m), \pi$

A nonempty minimal $\sim$-path in H between locations that are equivalent to ones in L must start with $I_m = [(i_m)_0 \to (i'_m)_0] \cdots [(i_m)_{c-1} \to (i'_m)_{c-1}]$, $0 \le m < a$ or $D_m = [(d_m)_0 \to (d'_m)_0] \cdots [(d_m)_{c-1} \to (d'_m)_{c-1}]$, $0 \le m < b$ or $[j_m \to z_m]$, $0 \le m < c$. After traversing I_m or D_m, all the gizmos are incremented or decremented, respectively. So when $[j_m \to z_m]$ or $[j_m \to \overline{z}_m]$ is traversed, which one is actually traversed depends only on the number of times I_m has been traversed over all m and the number of times D_m has been traversed over all m. So $H/\!\sim|_L$ is a value-0 inc^a-dec^b-jz^c gizmo. An example is found in Figure 5-3. □

Theorem 16. *Reachability in a maze with the value-0 inc-dec-jz gizmo is undecidable.*

Proof. Let P be a program for a 3-counter counter machine. It is a sequence of instructions, containing instructions like `inc`(i), which increments counter i by 1, `dec`(i), which decrements counter i by 1, `jz`(i, z), which jumps to instruction P_z if counter i is 0 and continues otherwise, and `halt`, which ends the program.

First, simulate a value-0 inc$^{|P|}$-dec$^{|P|}$-jz$^{|P|}$ gizmo using the previous lemma. Then make a sequence $\mathcal{G}$ of 3 copies of said gizmo. These gizmos make up the counters, and they will be wired according to the program. Let $H = \bigotimes \mathcal{G}$, and let $\sim$ be the minimal equivalence relation where for all k where $0 \leq k < |P|$:

- $a \sim b$ if $k < |P| - 1$, where:
  - $a = (i'_k)_c$ if $P_k = \texttt{inc}(c)$ for some c
  - $a = (d'_k)_c$ if $P_k = \texttt{dec}(c)$ for some c
  - $a = (\overline{z}_k)_c$ if $P_k = \texttt{jz}(c, p)$ for some c and p
  - $b = (i_{k+1})_c$ if $P_{k+1} = \texttt{inc}(c)$ for some c
  - $b = (d_{k+1})_c$ if $P_{k+1} = \texttt{dec}(c)$ for some c
  - $b = (j_{k+1})_c$ if $P_{k+1} = \texttt{jz}(c, p)$ for some c and p
  - $b = (i_{k+1})_0$ if $P_{k+1} = \texttt{halt}$
- $(z_k)_e \sim b$ if $k < |P| - 1$ and $P_k = \texttt{jz}(e, q)$ for some e and q, where:
  - $b = (i_q)_c$ if $P_q = \texttt{inc}(c)$ for some c
  - $b = (d_q)_c$ if $P_q = \texttt{dec}(c)$ for some c
  - $b = (j_q)_c$ if $P_q = \texttt{jz}(c, p)$ for some c and p
  - $b = (i_q)_0$ if $P_q = \texttt{halt}$
- For all p, q where $P_p = P_q = \texttt{halt}$: $(i_p)_0 \sim (i_q)_0$.

Let s be:

- $\pi_\sim((i_0)_c)$ if $P_0 = \texttt{inc}(c)$ for some c
- $\pi_\sim((d_0)_c)$ if $P_0 = \texttt{dec}(c)$ for some c
- $\pi_\sim((j_0)_c)$ if $P_0 = \texttt{jz}(c, p)$ for some c and p
- $\pi_\sim((i_0)_0)$ if $P_0 = \texttt{halt}$

and let t be $\pi_\sim((i_p)_0)$ for some p where $P_p = \texttt{halt}$. If P does not contain a `halt` instruction, the halting problem is easy, so assume it does.

Starting from a location in H that maps to s, a minimal $\sim$-path X has to follow the program. If $X_m \in \{[(i_k)_c \to (i'_k)_c], [(d_k)_c \to (d'_k)_c], [(j_k)_c \to (\overline{z}_k)_c]\}$ for some k and c, then X_{m+1}, if it exists, must correspond to the next instruction P_{k+1} since the corresponding location is the only one that is reachable. If $X_m = [(j_k)_c \to (z_k)_c]$ for some k and c, then $P_k = \texttt{jz}(c, p)$ for some p, and X_{m+1} must correspond to P_p. X_0 must correspond to P_0 according to the definition of s. X thus reaches a location equivalent to t if and only if P halts. An example is shown in Figure 5-4. $\square$

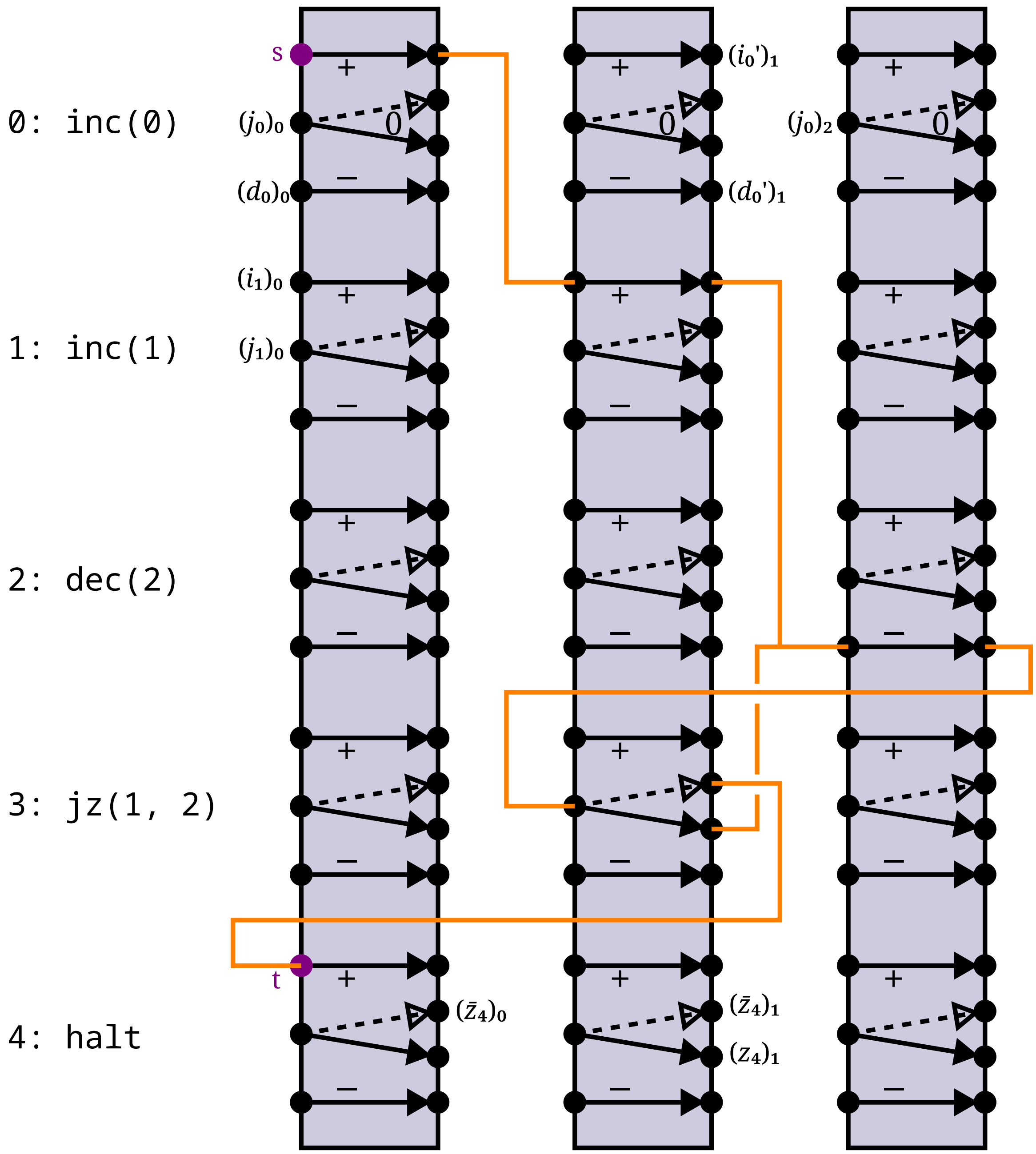


Figure 5-4: A maze made of value-0 inc^5-dec^5-jz^5 gizmos, which the value-0 inc-dec-jz gizmo can simulate. The program the maze is simulating is shown on the left. Only some locations are labelled to avoid clutter. Purple locations are s and t, and orange lines connect locations equivalent under $\sim$.

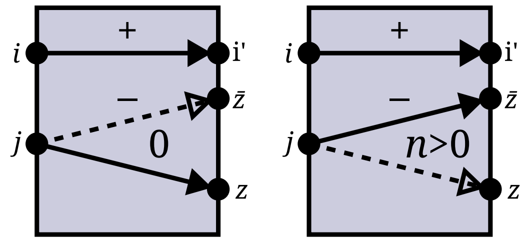


Figure 5-5: The *value-0 inc-jzdec* gizmo G_0 (left), and the *value-n inc-jzdec* gizmo G_n (right), where $n > 0$. These gizmos store a value-n. Traversing $[i \to i']$ adds 1 to n. $[j \to z] \in G_0$. $[j \to \overline{z}] \in G_n$ when $n > 0$, and subtracts 1 from n. The $[i \to i']$ traversal is called the *increment* tunnel and traversing it is called *incrementing* the gizmo. The $[j \to z]$ and $[j \to \overline{z}]$ traversals combined are called the *branch*.

The inc-dec-jz gizmo is perhaps a bit complicated with its seven locations. Another example of a gizmo with an infinite number of unreachables states is the inc-jzdec gizmo, shown in Figure 5-5. It is similar to the inc-dec-jz gizmo, but the decrement tunnel is mixed with the branch, such that if the branch is traversed while the counter is positive, the counter decrements. We will show that the value-0 inc-jzdec gizmo can simulate the inc-dec-jz gizmo, and thus reachability in a maze consisting of it is undecidable.

Theorem 17. *The value-0 inc-jzdec gizmo can simulate the value-0 inc-dec-jz gizmo.*

Proof. Let $\mathcal{G}$ be a sequence of 5 value-0 inc-jzdec gizmos. Gizmos 0 and 1 keep track of the counter, gizmos 2 and 3 are tunnel duplication scaffolding, and gizmo 4 is used as a diode. Let $H = \bigotimes \mathcal{G}$ and let $\sim$ be the minimal equivalence relation in $\mathrm{locs}(H)$ where:

- $i'_0 \sim i'_2 \sim i_1$
- $i'_1 \sim j_2$
- $i'_3 \sim i_2$
- $\overline{z}_0 \sim i_3$
- $i'_4 \sim i'_3 \sim j_1$
- $\overline{z}_1 \sim j_3$
- $z_0 \sim \overline{z}_3$

Label some equivalence classes according to the locations they intend to simulate: $[i] = \pi_\sim(i_0)$, $[d] = \pi_\sim(j_0)$, $[j] = \pi_\sim(i_4)$, $[i'] = \pi_\sim(z_2)$, $[d'] = \pi_\sim(\overline{z}_3)$, $[z] = \pi_\sim(z_1)$, and $[\overline{z}] = \pi_\sim(\overline{z}_2)$. Let L map $\{[i], [d], [j], [i'], [d'], [z], [\overline{z}]\}$.

Initially, the only minimal nonempty $\sim$-paths between locations equivalent to ones in L are $X_i = [i_0 \to i'_0][i_1 \to i'_1][j_2 \to z_2]$, $X_{dz} = [j_0 \to z_0]$, and $X_{jz} = [i_4 \to i'_4][j_1 \to z_1]$. Both X_{dz} and X_{jz} do nothing to the state of H (except incrementing gizmo 4,

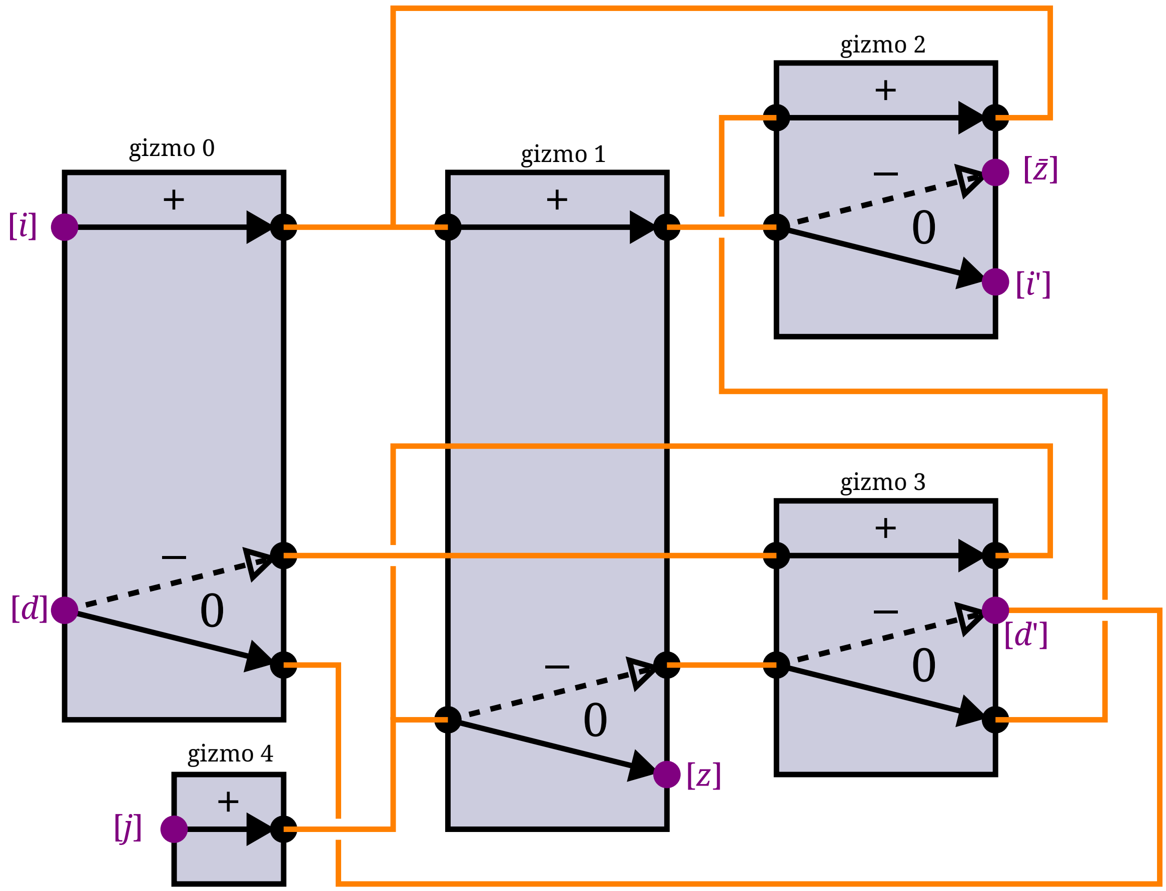


Figure 5-6: The value-0 inc-jzdec gizmo simulating the value-0 inc-dec-jz gizmo. Orange lines connect equivalent locations, and purple locations are in L.

but since gizmo 4's branch is unused, this does not matter), as intended. Whenever X_i is taken, gizmos 0 and 1 increment and all other gizmos do not change state. If gizmos 0 and 1 have a counter above 0, then X_{dz} and X_{jz} close, and new $\sim$-paths open up: $X_{d\bar{z}} = [j_0 \to \bar{z}_0][i_3 \to i'_3][j_1 \to \bar{z}_1][j_3 \to \bar{z}_3]$ and $X_{j\bar{z}} = [i_4 \to i'_4][j_1 \to \bar{z}_1][j_3 \to z_3][i_2 \to i'_2][i_1 \to i'_1][j_2 \to \bar{z}_2]$. $X_{j\bar{z}}$ does not change the state of H as intended, and $X_{d\bar{z}}$ decrements both gizmos 0 and 1 and leaves all others unchanged. Therefore, $H/\sim|_L$ is a value-0 inc-dec-jz gizmo. The simulation is shown in Figure 5-6. $\square$

Next, we will show that beating a level in generalized New Super Mario Bros. is undecidable. New Super Mario Bros. [8] is a 2-dimensional platforming game made for the Nintendo DS where Mario travels through many words and does various platforming challenges. Each level has its own platforming challenges, a timer, and a flagpole that must be reached in time to beat the level. In some levels, there is a pipe that spawns Goombas. Normally, leaving an enemy behind offscreen resets its state, and Goomba-spawning pipes can have only a bounded number of Goombas spawned at a time. But in generalized New Super Mario Bros., enemies never reset their state, and spawning pipes do not stop spawning.

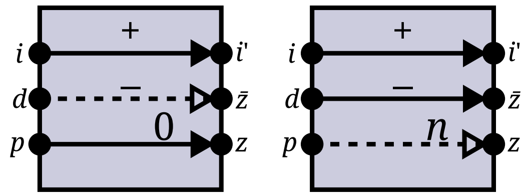


Figure 5-7: The *value-0 inc-decnz-pz* gizmo G_0 (left), and the *value-n inc-decnz-pz* gizmo G_n (right), where $n > 0$. These gizmos store a value-n. Traversing $[i \to i']$ adds 1 to n. $[p \to z] \in G_0$. $[d \to \overline{z}] \in G_n$ when $n > 0$, and subtracts 1 from n.

We will need a helper gizmo called the *inc-decnz-pz* gizmo, shown in Figure 5-7. This gizmo easily simulates the inc-jzdec gizmo by connecting locations d and p.

Theorem 18. *Beating a level in generalized New Super Mario Bros. is undecidable.*

Proof. We will show this by a reduction from reachability in a maze of value-0 inc-decnz-pz gizmos.

For the reduction, a value-0 inc-decnz-pz gizmo must be built in New Super Mario Bros., and since that game is 2-dimensional, a crossover must also be built.

The inc-decnz-pz gizmo is built in Figure 5-8. When Mario enters via `INC in`, he presses the left switch, the topmost 2 brown blocks, the leftmost 2 brown blocks A, and the brown block B in the long stretch of purple blocks disappear temporarily. This allows a Goomba to spawn from the pipe. A rising platform appears in place of A, forcing Mario up and away from the switch. A slowly lowering platform appears in place of B. The Goomba goes in the hole without being turned around by Goombas already in the hole. The blocks reappear and the lowering platform disappears, popping the Goombas in the hole back up to their normal position. The switch also reappears. Mario is forced to exit via `INC out`. The end result is an extra Goomba in the hole.

When Mario enters via `DECNZ in`, he presses the right switch. The nearest 2 brown blocks A to the switch and the rightmost brown block B disappear. A rising platform appears in place of A, forcing Mario up and away from the switch. In addition, a Goomba leaves the hole. B reappears before another Goomba can leave. The Goomba that left goes into the spike hole. Mario can exit via `DECNZ out` if and only if there is a Goomba in the spike hole, since the spikes are impossible to clear otherwise. The end result is one less Goomba in the hole, or a dead Mario if there were no Goombas in the hole in the first place.

When Mario enters via `PZ in`, he has to pass over the Goomba hole to get to `PZ out`. If there are any Goombas inside, he automatically jumps on them and hits the spikes. Otherwise, he safely runs over the hole.

The crossover is built in Figure 5-9. When Mario enters the crossover, he presses the switch he can access, which allows him to cross to the other side, but not turn. □

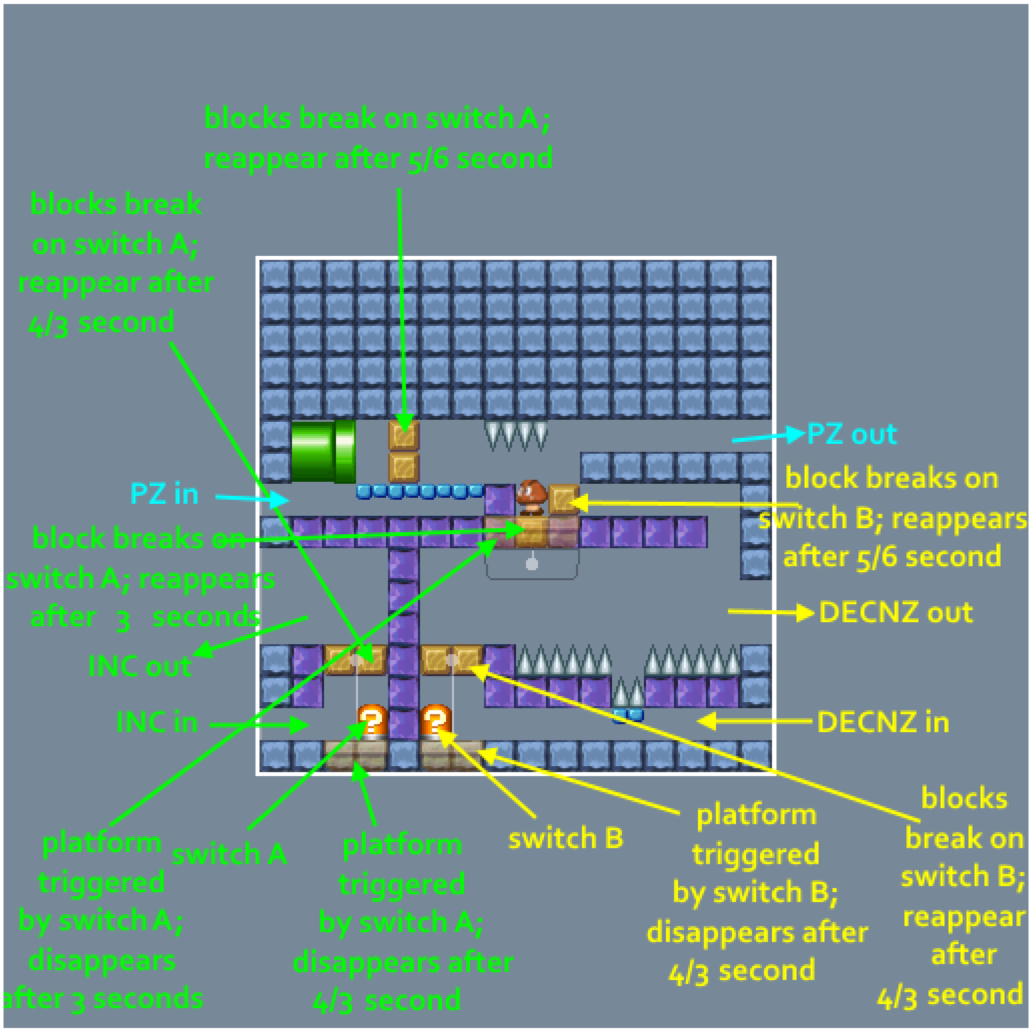


Figure 5-8: A value-n inc-decnz-pz gizmo built in New Super Mario Bros, where n is the number of Goombas in the hole.

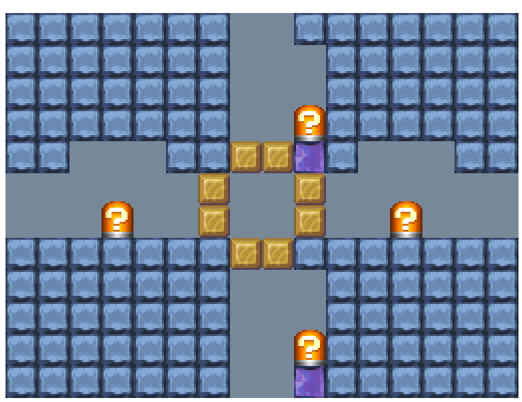


Figure 5-9: A crossover built in New Super Mario Bros. The switches on purple platforms make the horizontal stacks of brown blocks temporarily disappear, and the switches on blue platforms make the vertical stacks of brown blocks temporarily disappear.

Bibliography

[1] Joshua Ani, Jeffrey Bosboom, Erik D. Demaine, Yevhenii Diomidov, Dylan Hendrickson, and Jayson Lynch. Walking through doors is hard, even without staircases: Proving PSPACE-hardness via planar assemblies of door gadgets. In *Proceedings of the 10th International Conference on Fun with Algorithms (FUN 2020)*, pages 3:1–3:23, La Maddalena, Italy, September 28–30 2020.

[2] Joshua Ani, Erik D. Demaine, Dylan Hendrickson, and Jayson Lynch. Trains, games, and complexity: 0/1/2-player motion planning through input/output gadgets. In Petra Mutzel, Md. Saidur Rahman, and Slamin, editors, *Proceedings of the 16th International Conference and Workshops on Algorithms and Computation (WALCOM 2022)*, volume 13174 of *Lecture Notes in Computer Science*, pages 187–198, Jember, Indonesia, March 24–26 2022.

[3] Erik D. Demaine, Isaac Grosof, Jayson Lynch, and Mikhail Rudoy. Computational complexity of motion planning of a robot through simple gadgets. In *Proceedings of the 9th International Conference on Fun with Algorithms (FUN 2018)*, pages 18:1–18:21, La Maddalena, Italy, June 13–15 2018.

[4] Erik D. Demaine, Dylan Hendrickson, and Jayson Lynch. Toward a general theory of motion planning complexity: Characterizing which gadgets make games hard. In *Proceedings of the 11th Conference on Innovations in Theoretical Computer Science (ITCS 2020)*, pages 62:1–62:42, Seattle, Washington, January 12–14 2020.

[5] Dylan Hendrickson. Gadgets and gizmos: A formal model of simulation in the gadget framework for motion planning. Master's thesis, Massachusetts Institute of Technology, 2019.

[6] Jayson Lynch. *A Framework for Proving the Computational Intractability of Motion Planning Problems*. PhD thesis, Massachusetts Institute of Technology, 2020.

[7] Marvin L. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

[8] Nintendo. New super mario bros. Nintendo DS, 2006.

[9] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.