6.5440 | Lecture 11 | Sept. 1, 2023

Motion planning through gadgets

general theory of "gadgets" that can be traversed by an agent (player, robot, etc.)

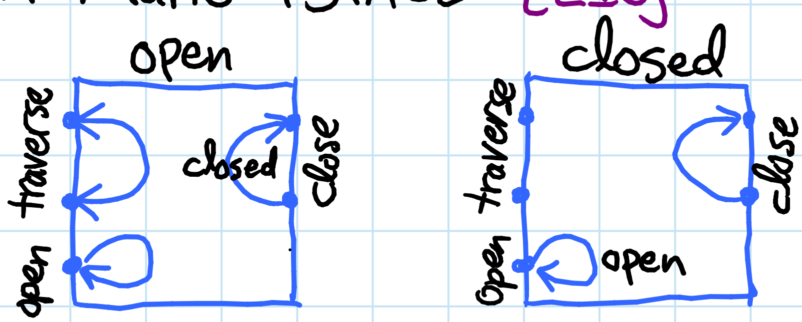
- introduced by Demaine, Grosof, Lynch, Rudoy [FUN 2018] & Demaine, Hendrickson, Lynch [ITCS 2020]
- L6 called this "[planar] motion planning"

What is a gadget?

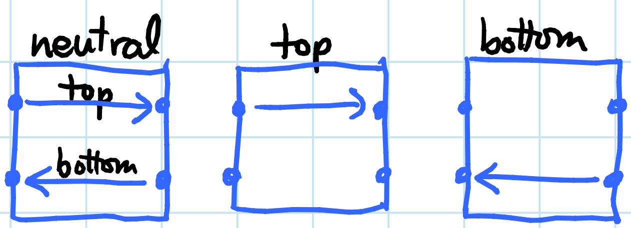
- simple: $O(1)$ size & states
- local: all the action & state is within the gadget
- traversable: agent traverses between some locations (entrances/exits), possibly changing state

Example: door from Mario PSPACE [L10]

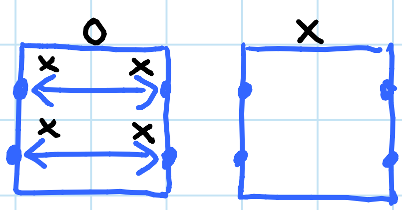
- 2 states
- 5 locations



Example: directed antiparallel NAND [L6]



Example: undir. noncrossing matched crumblers [L6]



Gadget $G = (Q, L, T)$ consists of:

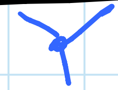
- Q = finite set of **states**
- L = finite set of **locations**
  - cyclicly ordered for planar gadgets
- T = set of valid **transitions** of the form $(q,a) \to (r,b)$ where $q, r \in Q$ & $a, b \in L$
  $\subseteq (Q \times T)^2$ essentially

AGENT → **Traversal** $\overset{\in L}{a} \to \overset{\in L}{b}$ is **legal** in state $q \in Q$ if
GADGET → transition $(q,a) \to (r,b) \in T$ for some $r \in Q$

- r is the new gadget state after
- might be multiple possible → **nondeterministic**
  - e.g. optional open→open in door
- gadget **deterministic** if state q & location a determine ≤ 1 transition $(q,a) \to (r,b)$
  - e.g. NAND & matched crumblers
- **tunnel** $a, b \in L$, $a \neq b$, if all legal traversals involving a or b are $a \to b$ or $b \to a$
  - **tunnel gadget** if every location in a tunnel
  - e.g. NAND & matched crumblers, not door:
- **button** $a \in L$ if $a \to a$ only legal traversal involving a
  - e.g. door's open

Automaton view: gadget = finite automaton with

- alphabet = $\{\text{traversals } a \to b \mid a, b \in L\}$
- states = Q
- $\delta(q, a \to b) = \{r \mid (q,a) \to (r,b) \in T\}$

<u>System</u> S of gadgets consists of
- finite set of gadgets ← often, many copies of 1 or a few gadgets
- initial state for each
- <u>connection graph</u> on gadgets' locations
  - free agent traversals with no state change
  - connected components = <u>system locations</u>
  - ⇒ generally assume <u>branching hallway</u>  (cf. L6)
- <u>planar</u> if gadgets + conn. graph has no crossings

<u>Reachability</u>: (a.k.a. 1-player motion planning)
given system S and start & goal locations s, t, can agent get from s to t by traversals + connections?
- also work on <u>reconfiguration</u>: reaching target state for each gadget [Ani, Demaine, Diomidov, Hendrickson, Lynch – WALCOM 2022]

<u>Hardness</u>: complexity with various gadgets
- reachability with doors is PSPACE-complete [L10]
- planar reachability with antiparallel NAND or matched crumblers is NP-complete [L6]

⇒ to prove your motion-planning problem/game/puzzle hard, suffices to build that gadget (+ connections)

Simulation of gadget G in state q
= system S of gadgets $\in \mathcal{H}$
+ mapping from G's locations
to distinct system locations
such that legal system traversals correspond
to legal G traversals
- saw several examples in L6 ("makes")
  - e.g.: undir. noncrossing NAND simulates
    undir. crossing NAND

$\Rightarrow$ reduction from reachability with G
to reachability with $\mathcal{H} = \{$gadgets in S$\}$
$\Rightarrow \mathcal{H}$ as hard as G

Door framework: 3 families of 2-state gadgets

open / closed

[Ani, Bosboom, Demaine, Diomidov, Hendrickson, Lynch — FUN 2020]

Door: 3 possible traversals
- open: [optionally] switch to open state
- close: switch to closed state
- traverse: possible only in open state
- each can be directed or undirected

Self-closing door: 2 possible traversals
- open: [optionally] switch to open state (as before)
- self-close: possible only in open state & switches to closed state (traverse + close)

Symmetric self-closing door:
- self-open: possible only in closed state & switches to open
- self-close: possible only in open state & switches to closed state (traverse + close)

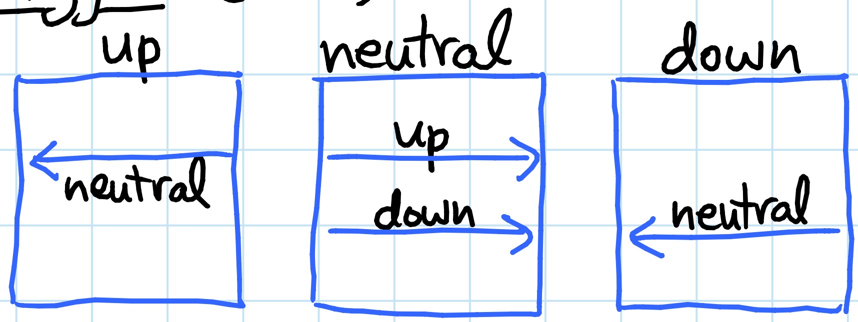
PSPACE-hard planar reachability with any door

Applications: many

Universal: simulate ALL gadgets! even planarly

<u>Reversible</u>: every transition $(q, a) \to (r, b) \in T$
has reverse $(r, b) \to (q, a) \in T$

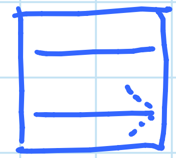
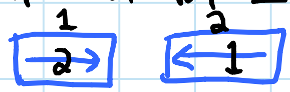
<u>Example</u>: <u>locking 2-toggle</u> (L2T)
- 3 states
- 4 locations
- 2 tunnels
- deterministic



<u>Hardness characterization</u>: [Demaine, Hendrickson, Lynch 2020]
set of reversible deterministic tunnel gadgets have PSPACE-complete reachability problem
$\Leftrightarrow$ planar reachability is PSPACE-complete
$\Leftrightarrow$ some gadget has <u>interacting tunnels</u>
traversing some tunnel in some state affects (adds or removes) traversability of some other tunnel (in next state)
$\Leftrightarrow$ gadgets can simulate locking 2-toggle
- call {L2T} <u>basis</u> for these gadgets
door simulates all simulates L2T

- otherwise, in P (in fact, NL = NLOGSPACE)

Proof:

- no interacting tunnels
  - ⇒ can split each tunnel into own gadget
  - ⇒ shortest path visits each tunnel $\leq$ once so only initial state matters
- interacting tunnels
  - ⇒ say top tunnel affects bottom tunnel's right traverse 
  - — reversibility ⇒ exists in state 2, not in 1
  - — then build L2T via 1-toggle via 1-direction edge 

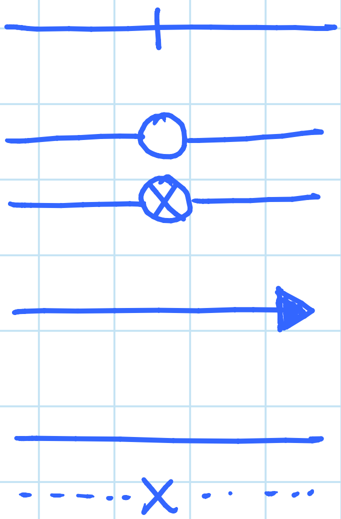
- planar L2T PSPACE-hard by reduction from "planar nondeterministic constraint logic" [L12]

Applications: many

2-state 2-tunnel reversible deterministic gadgets:

[Demaine, Grosof, Lynch, Rudoy – FUN 2018]

Tunnel types:



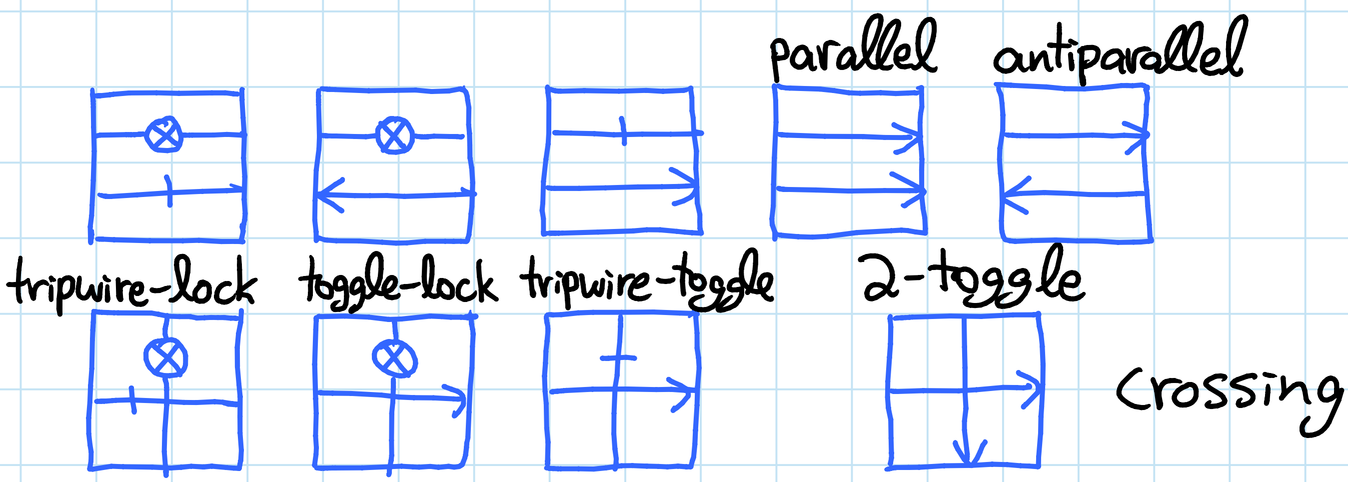
tripwire: always bitraversable & traversal flips state

lock: bitraversable in one state not in other

toggle: traversal in one direction & toggles state + direction

trivial: always/never bitraversable

Characterization: reachability PSPACE-complete
$\Leftrightarrow$ planar reachability PSPACE-complete
$\Leftrightarrow$ both tunnels nontrivial



– each can simulate all others
$\Rightarrow$ each forms a basis / is universal

– otherwise polynomial (effectively 1 tunnel)

Sokoban: PSPACE-complete [Culberson 1997]
=Push-1FS via self-closing door [subset]
↳ storage
— can we adapt to Push-1F?

Checkable gadgets: [Ani, Chung, Demaine, Diomidov, Hendrickson, Lynch — FUN 2022]

Checkable G = gadget G′
+ sequence of check traversals

"post-selection"

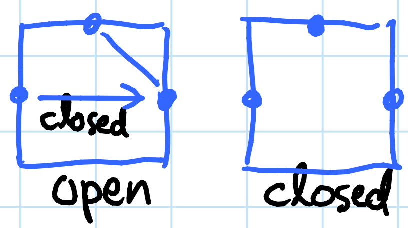
where G = G′ restricted to good states
from which checks can be traversed

- require broken (not good) states to be closed under traversals

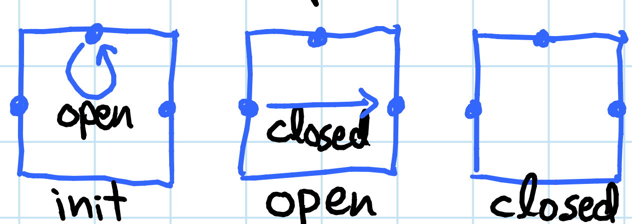
Nonlocal simulation: planar reachability with G reduces to planar reachability with {G′, MSC, SO}

⇒ can pretend we built G / no broken states

- merged single-use closing (MSC):



- single-use opening (SO):



Proof idea: after old goal, lock old connections & force checking traversals

Application: Push-1F

__I/O gadget__: can partition locations into __inputs__ (entrances) & __outputs__ (exits)

- all traversals from input to output
- require ≤1 input in each system location

⇒ __0-player__ motion planning! [Ani, Demaine, Hendrickson, Lynch – WALCOM 2022]

__Output disjoint__: no 2 inputs traverse to same output

⇒ 2-state subunits: $Q = \{\text{up}, \text{down}\}$

unbounded with both set-up & -down

unbounded

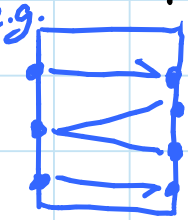
- __switch__: output depends on state
- __set-up__: sets state to up (sym. → down)
- __toggle__: flips state
- __set-up switch__: switch + set-up
- __toggle switch__: switch + toggle

__Hardness characterization__: 2-state deterministic output-disjoint I/O gadgets PSPACE-complete

⇔ unbounded & >1 nontrivial input & traversals depend on state (≥1 switch)

assuming NP≠PSPACE

e.g.



set-up
switch
set-down

- every such gadget can simulate every deterministic I/O gadget
  ⇒ both basis & universal
- 1-player simulates all gadgets (like door)

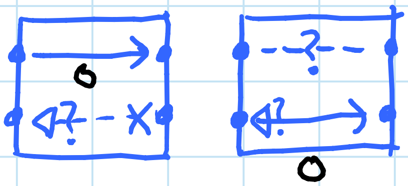
OPEN: complexity of 1 toggle switch "ARRIVAL"

Also 2-player & team variants [Demaine, Hendrickson, Lynch 2020]

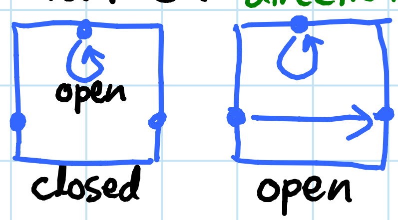
DAG gadget: state-transition graph is acyclic
↳ possible transitions on states (merging all locations)
⇒ |transition sequence| < #states

Hardness characterization: [Demaine, Hendrickson, Lynch 2020]
set of DAG tunnel gadgets have NP-complete reachability problem
↳ not planar!

⇔ some gadget has distant opening: traversal that opens another tunnel in some direction

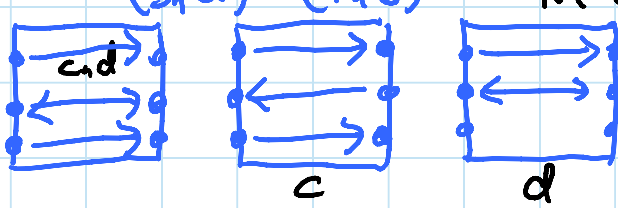


cf. opening door with 1 button [LG]

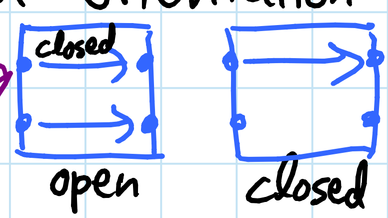


OR some gadget has forced distant closing: orientation of the tunnels + traversal (s, a→b) that always closes ≥1 other tunnel in direction of orientation
(s, a)→(*, b)

e.g.



cf. closing door [LG]



loops ← LDAG/eventually static, not DAG

– otherwise, in P (in fact, NL)