# Control Improvisation

**Sanjit A. Seshia**

Professor

EECS, UC Berkeley

Joint work with:

Daniel Fremont, Tommaso Dreossi,
Rafael Valle, Sophie Libkind, Alexandre Donze,
David Wessel

December 11, 2017

# Diversity and Resilience in Control

- Diversity: How to *engineer* diversity of behaviors?
  - For system robustness, resilience, privacy, etc.

- Resilience: How to *adapt* to changing environment conditions?
  - Learn about environment via controlled random exploration

➢ Underlying computational problem: Control Improvisation

TerraSwarm

# What does it mean to "improvise"?

David Wessel
(1942-2014)

"To perform without preparation"

– Merriam-Webster Dictionary

"Learn the rules and then break them in such a way as to exercise good taste."

– Sir George Shearing (Jazz musician)

"Only he who is well prepared has any opportunity to improvise."

– Ingmar Bergman (movie director, writer and producer)

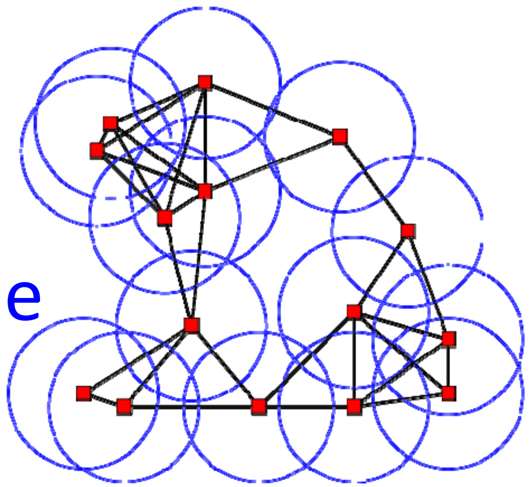
# Music Improvisation

- **Generate a variant of existing tune**
  - "improvise on this jazz melody"
- Follow conventions of the genre — **Hard Constraint**
- Don't do the same thing too often — **Randomness Constraint**
- Usually, be similar to the original — **Soft Constraint**

(examples at http://www.eecs.berkeley.edu/~donze/impro_page.html)

## Robotic Patrolling

“Patrol an area in a way unpredictable to an adversary”



- Visit each location sufficiently often — **Hard Constraint**
- Don’t always take the same route — **Randomness Constraint**
- Usually, take a route close to the best (e.g., shortest) one — **Soft Constraint**

# Commonalities between these Applications

Generate (event/action) sequences subject to three kinds of constraints:

- Hard constraint
  - *Every* sequence satisfies some property P1
- Randomness requirement
  - No sequence is generated too frequently
- Soft constraint
  - *"Most"* sequences satisfy some other property P2

*Control Improvisation* is a precisely-defined theoretical problem capturing these requirements

[Fremont et al., FSTTCS 2015; Donze et al., ICMC'14; Valle et al., IoTDI'16]

# Deterministic Planning from High-Level Task Specification

Generate sequences subject to:

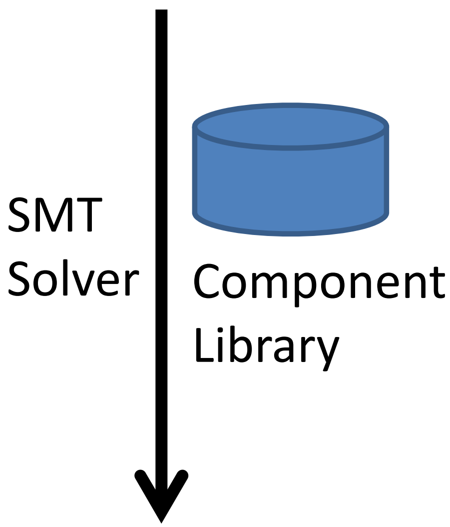
- Hard constraint
  - *Every* sequence satisfies some property P1
- ~~Randomness requirement~~
  - ~~No sequence is generated too frequently~~
- ~~Soft constraint~~
  - ~~*Most* sequences satisfy some other property P2~~

- Encodeable as a *Satisfiability Solving* problem (without adversarial environment)
  - E.g. [Saha et al., IROS'14; Shoukry et al., CDC'16, HSCC'17, CDC'17]
- Reactive Synthesis (with adversarial environment)

# Multi-Robot Motion Planning from Temporal Logic using Satisfiability Modulo Theories (SMT)

[IROS'14]

Declarative Task Specification (Temporal Logic)




Executable Software

Video of Demonstration on Quadrotors

MORE SCALABLE APPROACHES: ICCPS'16, CDC'16, HSCC'17, CDC'17

# Probabilistic Planning from High-Level Task Specification

Generate ONE sequence subject to:

- Hard constraint
  - Sequence satisfies some property ("reach a goal state")
- Randomness requirement
  - Probability of success above some threshold
- ~~Soft constraint~~
  - ~~*Most* sequences satisfy some other property P2~~

- Randomness is an input to the problem
- Proved to be undecidable in the '90s

# Control Improvisation

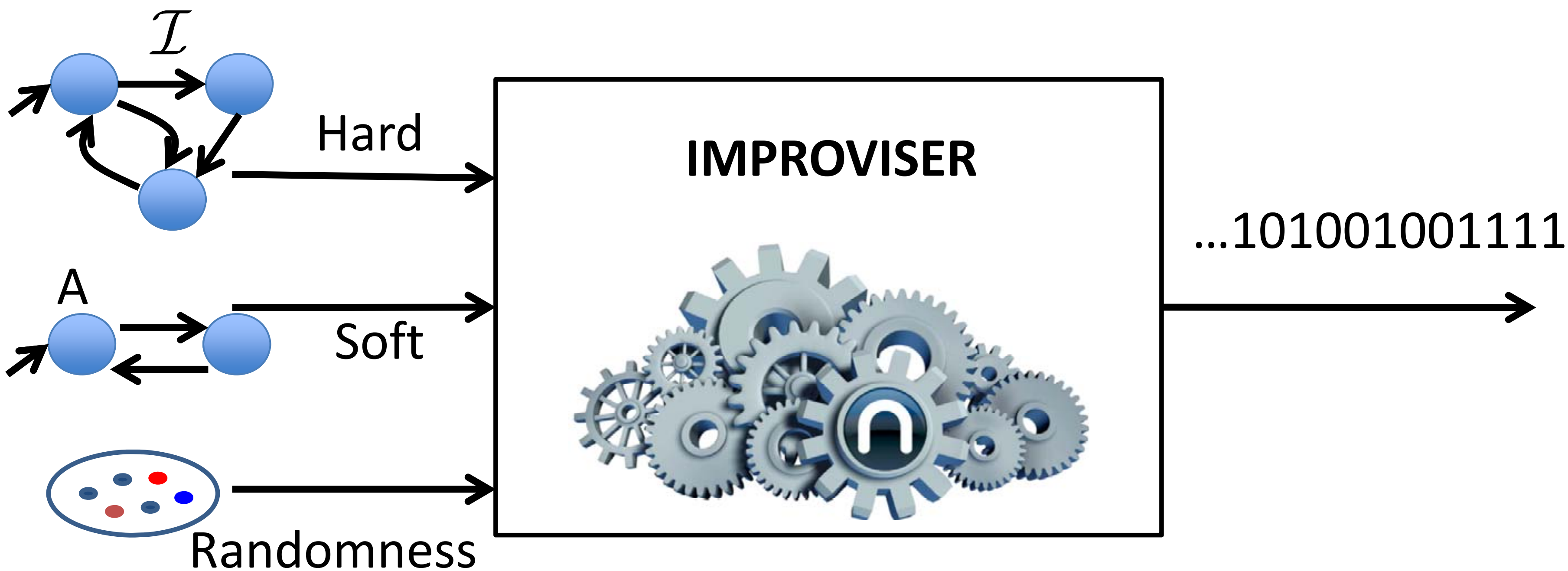
Generate (MANY) sequences subject to three kinds of constraints:

- Hard constraint
  - *Every* sequence satisfies some property P1
- Randomness requirement
  - No sequence is generated too frequently
- Soft constraint
  - *"Most"* sequences satisfy some other property P2

[Fremont et al., FSTTCS 2015; Donze et al., ICMC'14; Valle et al., IoTDI'16]

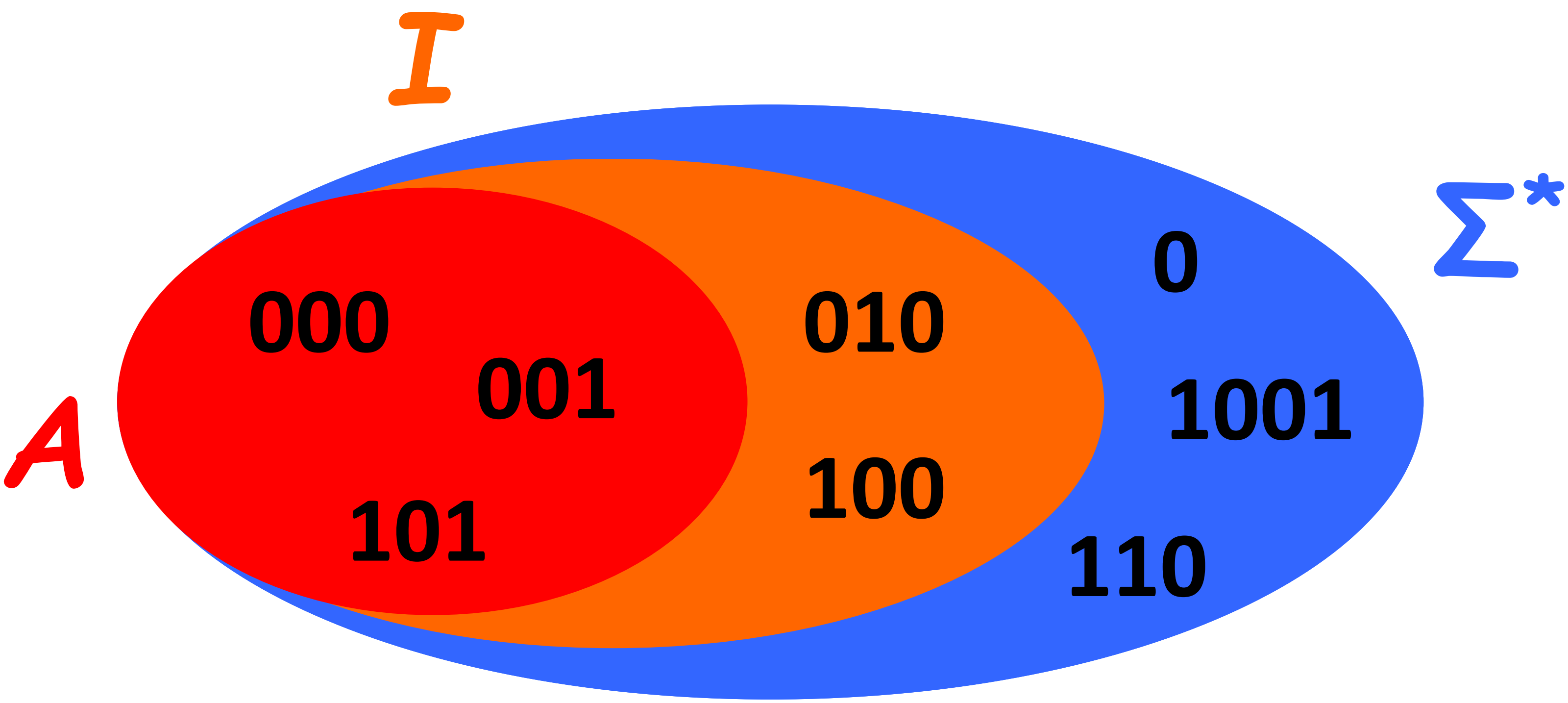
# Structure of an Improviser

A generator of strings from a finite alphabet $\Sigma$ (say {0,1})



# Example

- Variations of the string *s* = '001'
  - Hard constraint: length 3, and no consecutive '1's (easily encoded as a DFA $\mathcal{I}$)
  - Soft constraint: Hamming distance to *s* is at most 1



# Definition of Control Improviser (initial)

[FSTTCS'15]

Given $\mathcal{I}$ and *A*, for $0 \leq \varepsilon \leq 1$ and $0 < \rho \leq 1$
a distribution $D : \Sigma^* \to [0,1]$ with support *S* is an
*(ε, ρ)-improvising distribution* if:

- $S \subseteq I$ — Hard constraint
- $\forall w \in S,\ D(w) \leq \rho$ — Randomness requirement
- $\Pr[w \in A \mid w \leftarrow D] \geq 1 - \varepsilon$ — Soft Constraint

The CI instance C = ($\mathcal{I}$, *A*, ε, ρ) is *feasible* if such a distribution exists. An *improviser* is a probabilistic algorithm generating strings whose output distribution is an *(ε, ρ)-improvising distribution*.

# Existence of Improvisers

- Feasibility just requires *I* and *A* to be large enough
- All feasible instances have improvisers

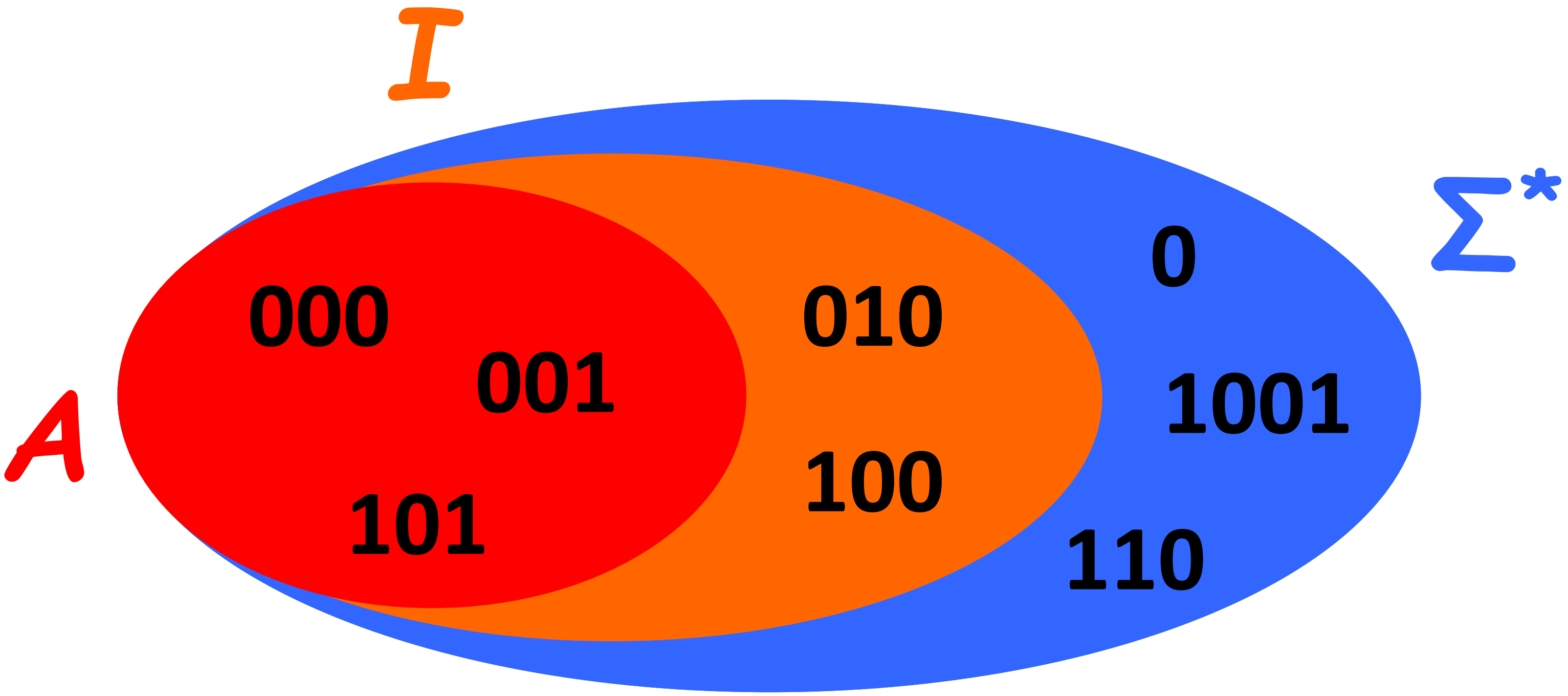
**Theorem.** *For any* $\mathcal{C} = (\mathcal{I}, \alpha, \epsilon, \rho)$, *the following are equivalent:*

*(a)* $\mathcal{C}$ *is feasible.*

*(b)* $|I| \geq 1/\rho$ *and* $|A| \geq (1 - \epsilon)/\rho$.

*(c) There is an improviser for* $\mathcal{C}$.

# Example




- With $\varepsilon = \rho = 1/4$:
  - Uniformly sample from *A* with probability ¾
  - Uniformly sample from *I* \ *A* with probability ¼

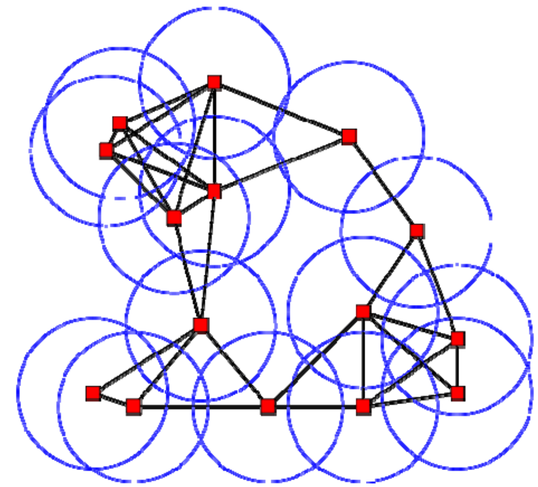
# Complexity Results

- Complexity grows as *I* and *A* are given by more complex automata

<table>
<tr><th colspan="2">$\mathcal{I}$ \ $A$</th><th rowspan="2">DFA</th><th colspan="2">NFA</th><th rowspan="2">PFA</th></tr>
<tr><th colspan="2"></th><th>$L(\mathcal{A}) = \infty$</th><th>$L(\mathcal{A}) < \infty$</th></tr>
<tr><td colspan="2">DFA</td><td colspan="2" rowspan="2">poly-time</td><td rowspan="3">#P-hard</td><td rowspan="4">undecidable</td></tr>
<tr><td rowspan="2">NFA</td><td>$L(\mathcal{I}) = \infty$</td></tr>
<tr><td>$L(\mathcal{I}) < \infty$</td><td>#P-hard</td><td>-</td></tr>
<tr><td colspan="2">PFA</td><td colspan="3">undecidable</td></tr>
</table>

- Symbolic approach: encode constraints in (propositional) logic, leverage progress in SAT solving, model counting

# Recall: Robotic Patrolling



"Patrol an area in a way unpredictable to an adversary"

- Visit each location infinitely often — **Hard Constraint**
- Every sub-sequence of size s of the plan is repeated with low probability — **Randomness Constraint**
- Minimize mean time between visits to a location (over all locations) — **Soft Constraint**

# Recent Control Improvisation Demo (Berkeley + Penn)

[joint with Vijay Kumar and George Pappas]

(see https://math.berkeley.edu/~dfremont/impro.html)

**Hard constraints:**

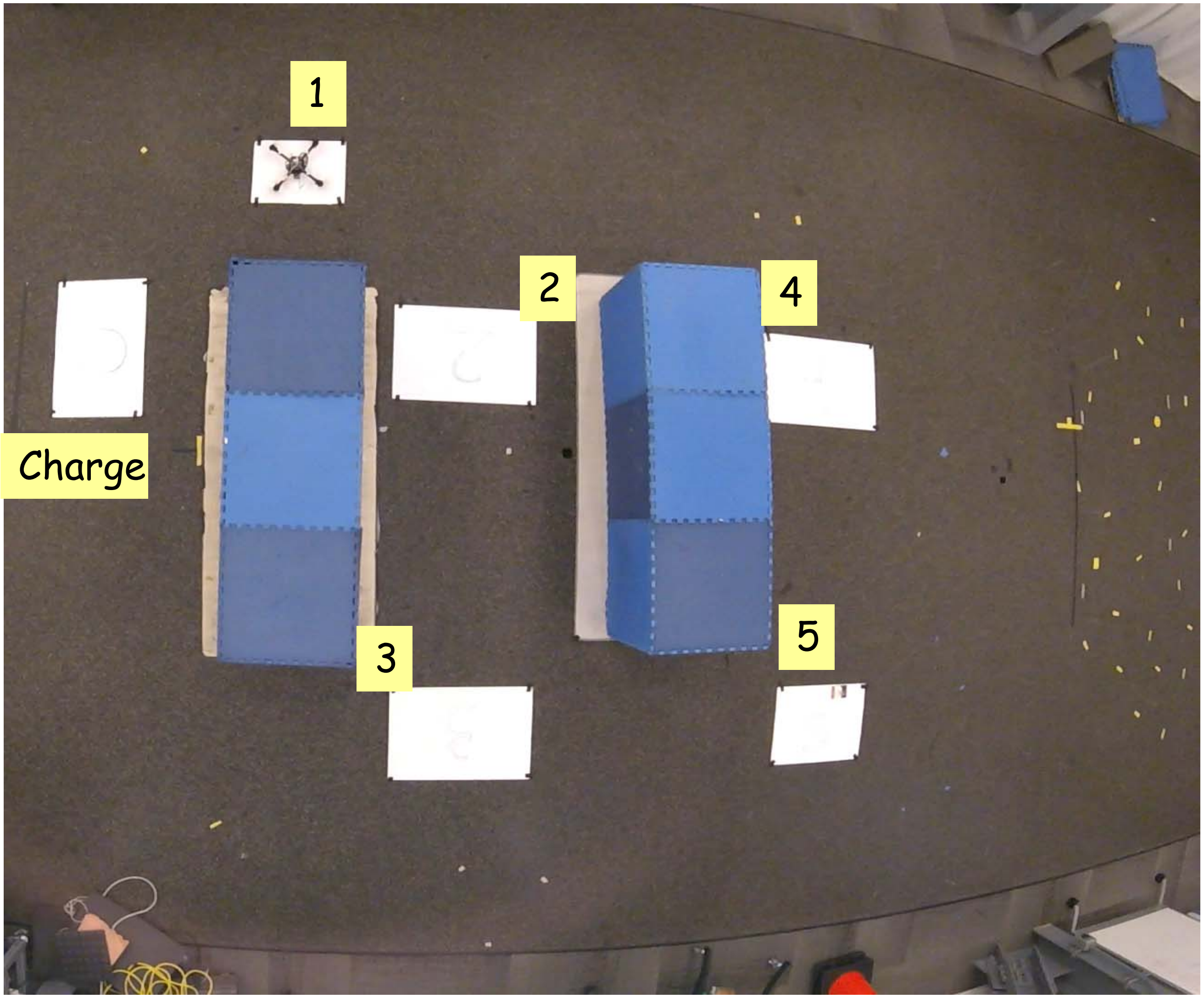
- Visit each location at least once
- Do not visit any location twice in a row
- No more than three locations without recharging

**Soft constraints:**

Visit each location exactly once with prob. > 0.8

**Randomness:**

No trajectory generated with prob. > 0.052



## Summary and Future Work

- Control improvisation is a new class of problems involving random generation of event/action sequences with hard & soft constraints
- Many Ongoing/Future Directions:
  - Reactive setting: adversarial environments
  - Soft constraints encoding general quantitative requirements
  - Different types of randomness constraints encoding specific distributional properties
  - More applications

https://math.berkeley.edu/~dfremont/impro.html