Liveness of Randomised Parameterised Systems under Arbitrary Schedulers

Anthony W. Lin and Philipp Ruemmer

Yale NUS College

UPPSALA UNIVERSITET

Summary of results

- Automatic method for proving liveness for randomised parameterised systems, e.g.,
    - Randomised Self-Stabilising (Israeli-Jalfon/Herman)
    - Randomised Dining Philosopher (Lehmann-Rabin)
- Regular model checking as symbolic framework
- CEGAR/Learning to synthesise “regular proofs”

Background

Parameterised Systems

Definition: An infinite family of finite-state systems

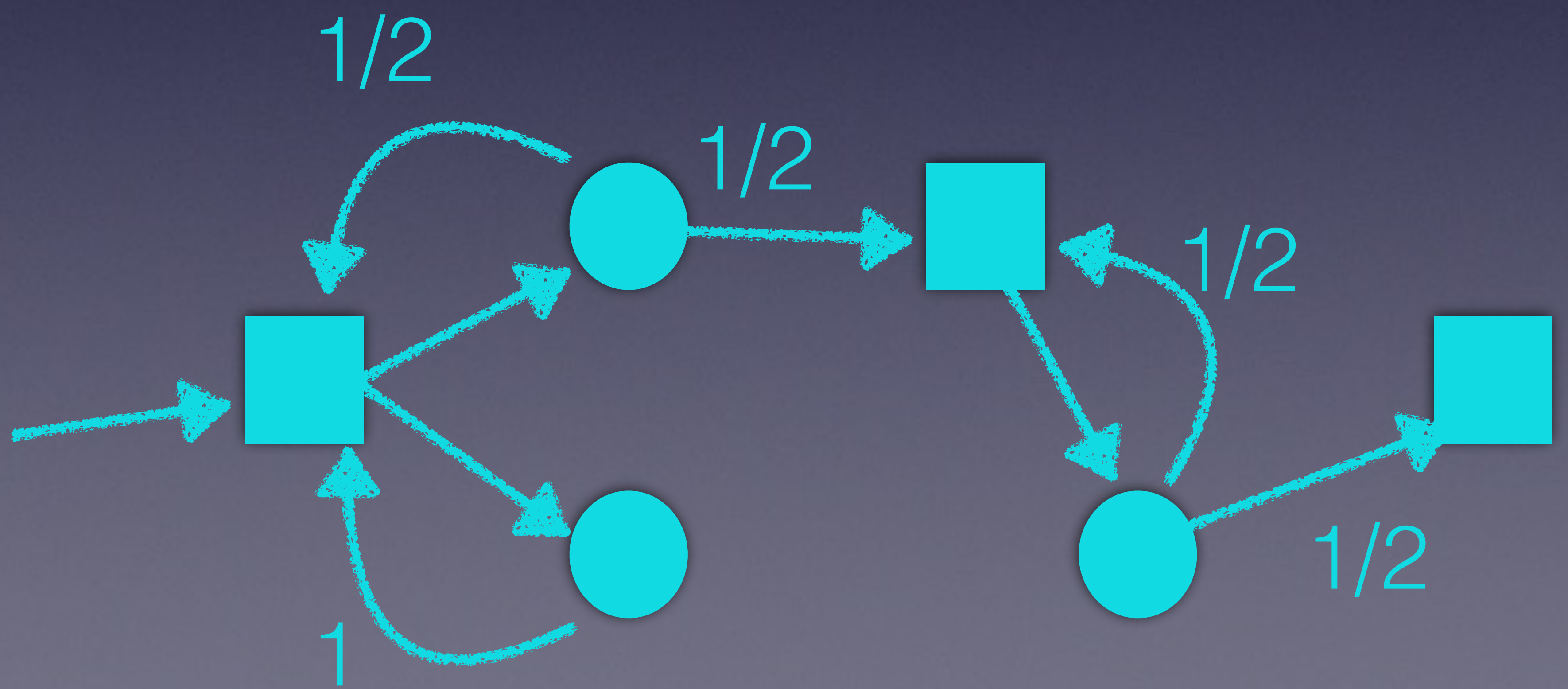
Example: most distributed protocols in the verification literature, e.g., for the Dining Philosopher problem

$$\mathcal{F} = \{\text{Protocol with } n \text{ processes} : n \in \mathbb{N}\}$$

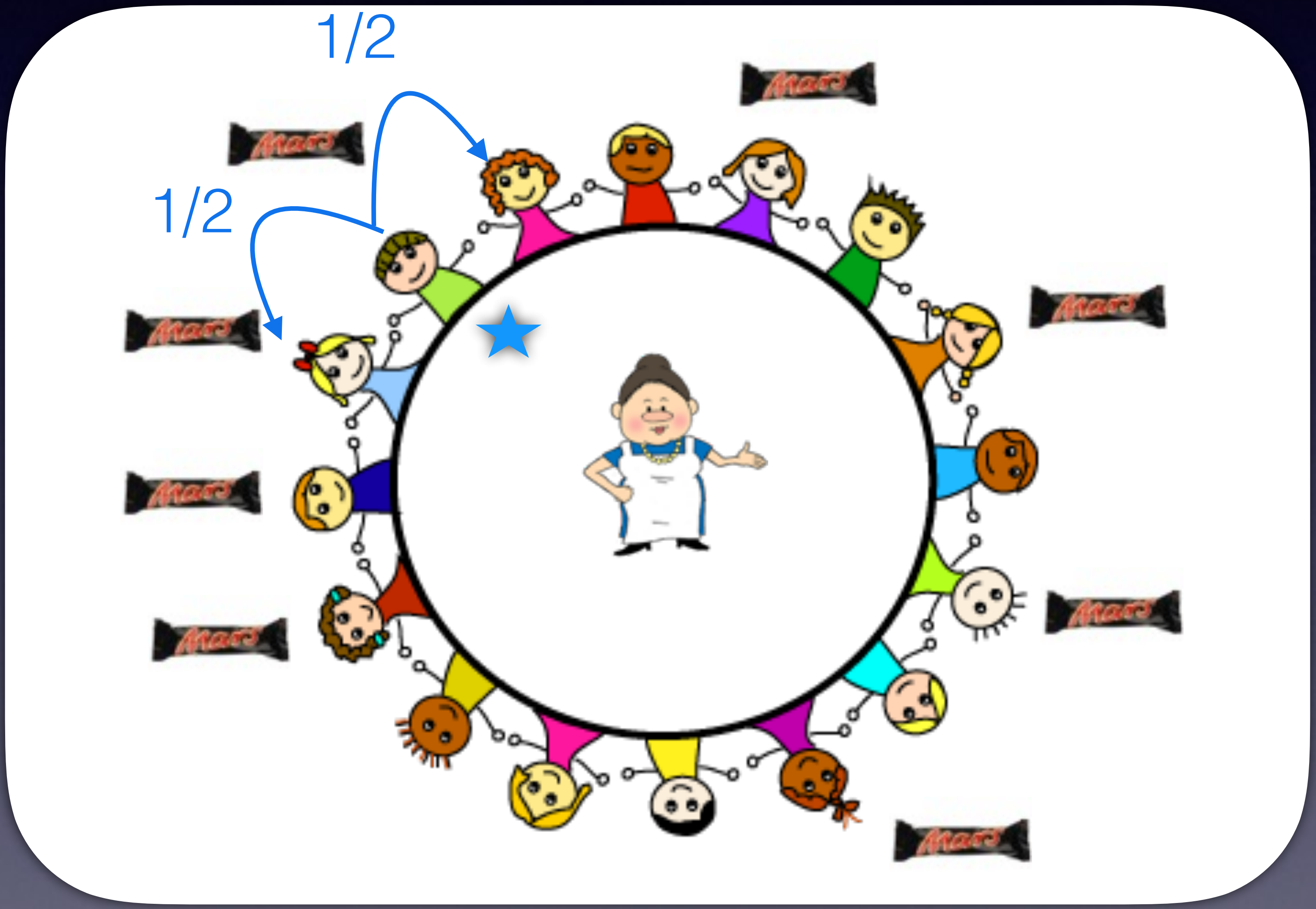
Randomised Parameterised Systems

Definition: An infinite family of **randomised** finite-state systems

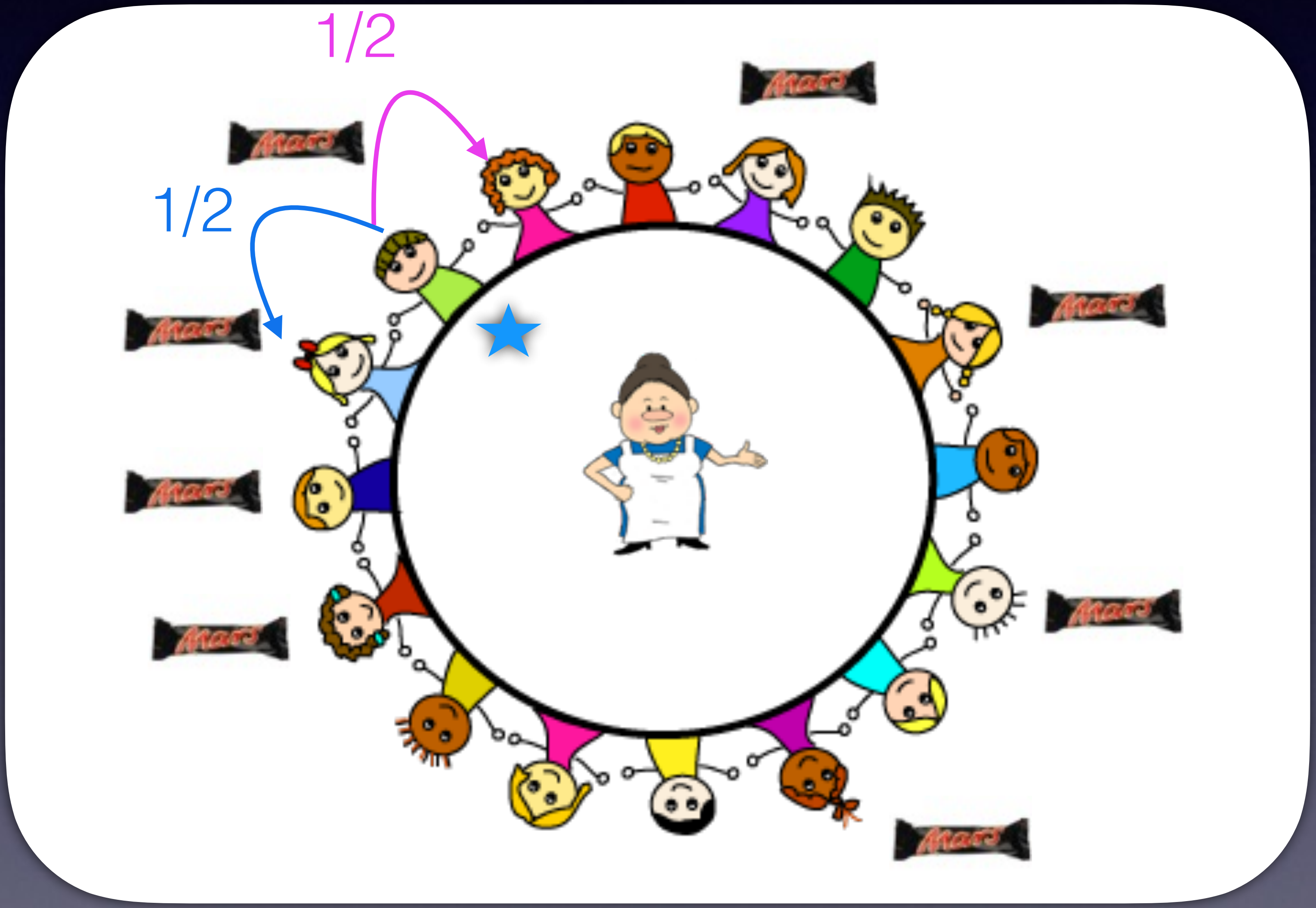
Markov Decision Processes



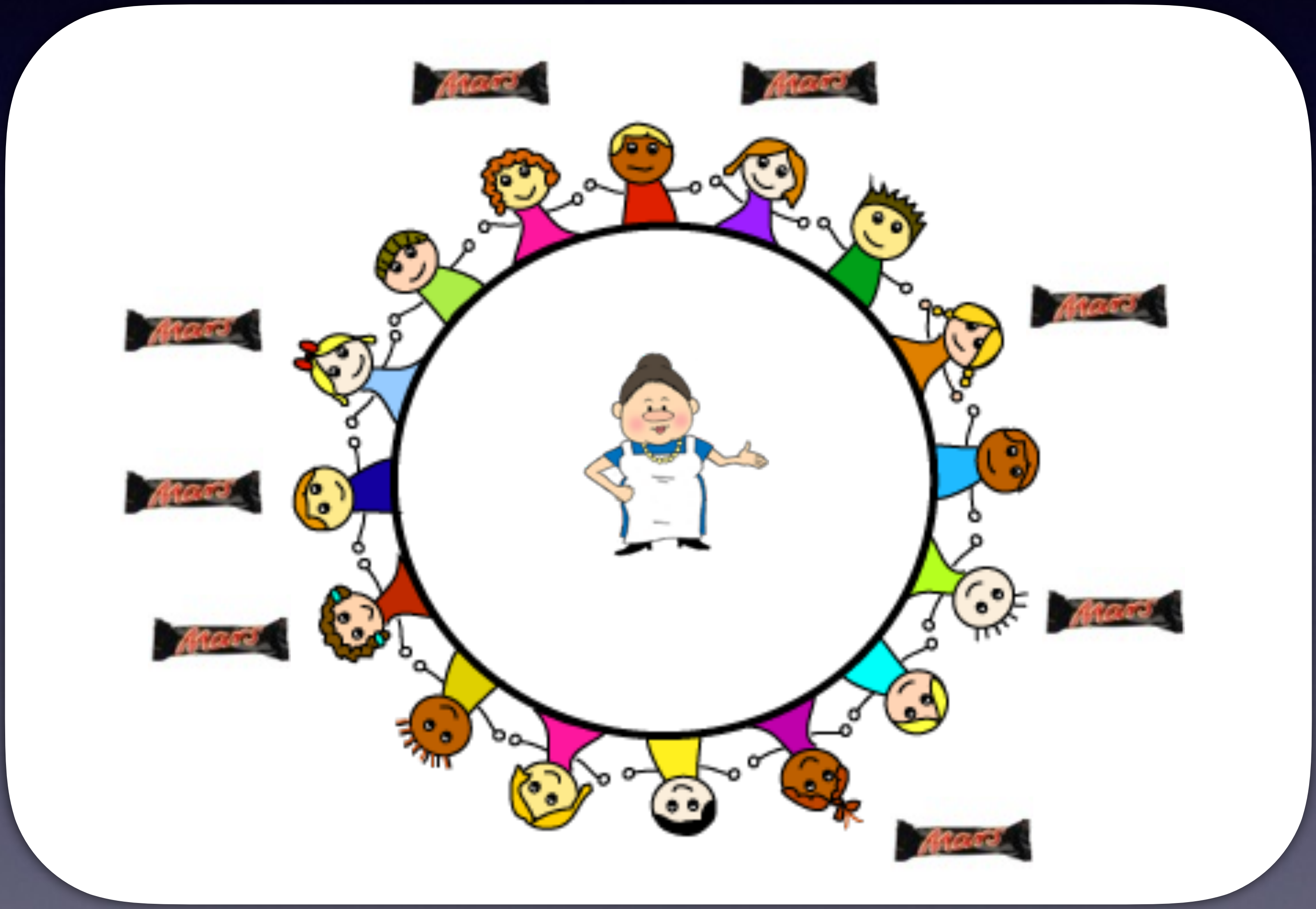
Israeli-Jalfon Randomised Self-Stabilising Protocol



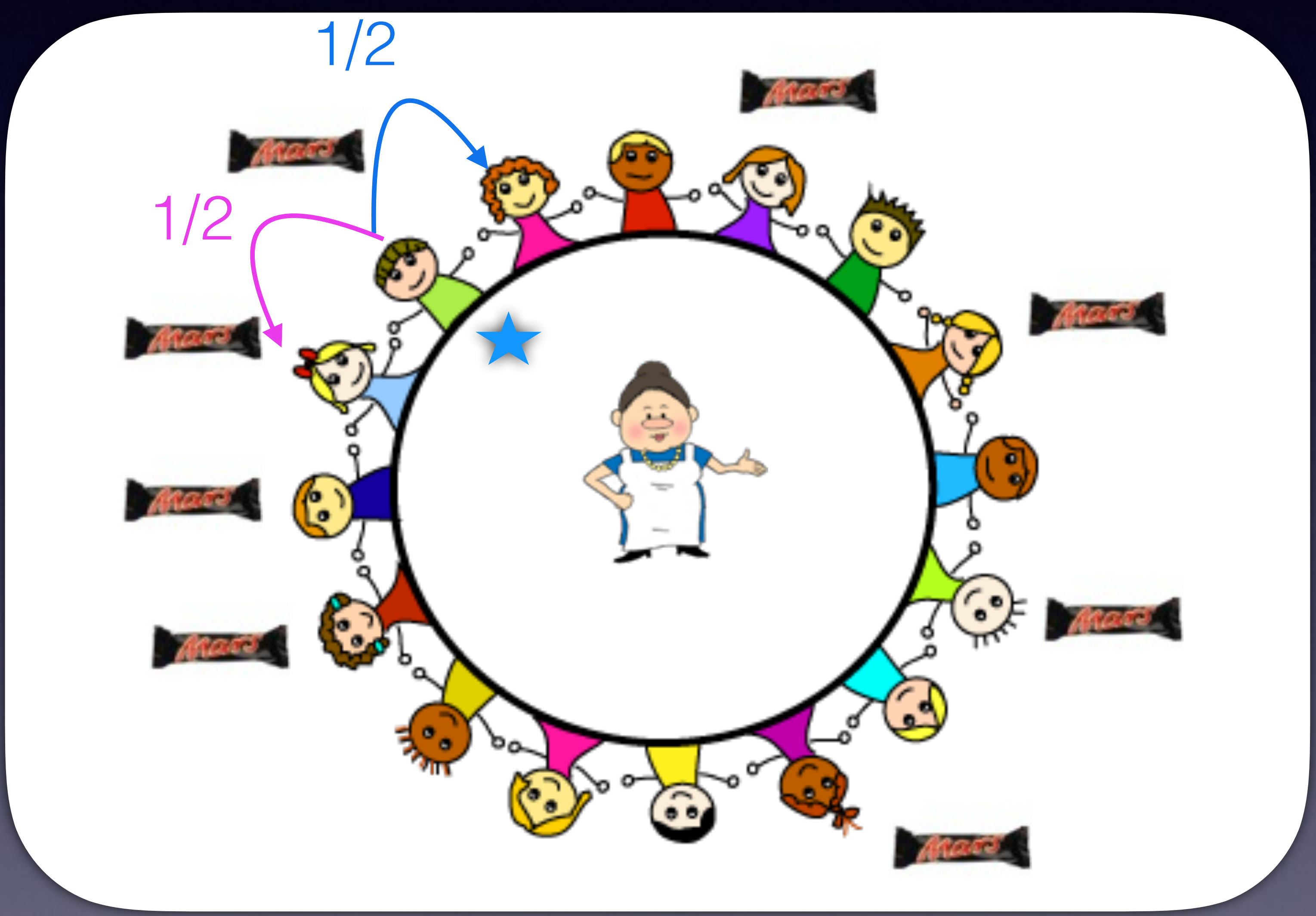
Israeli-Jalfon Randomised Self-Stabilising Protocol



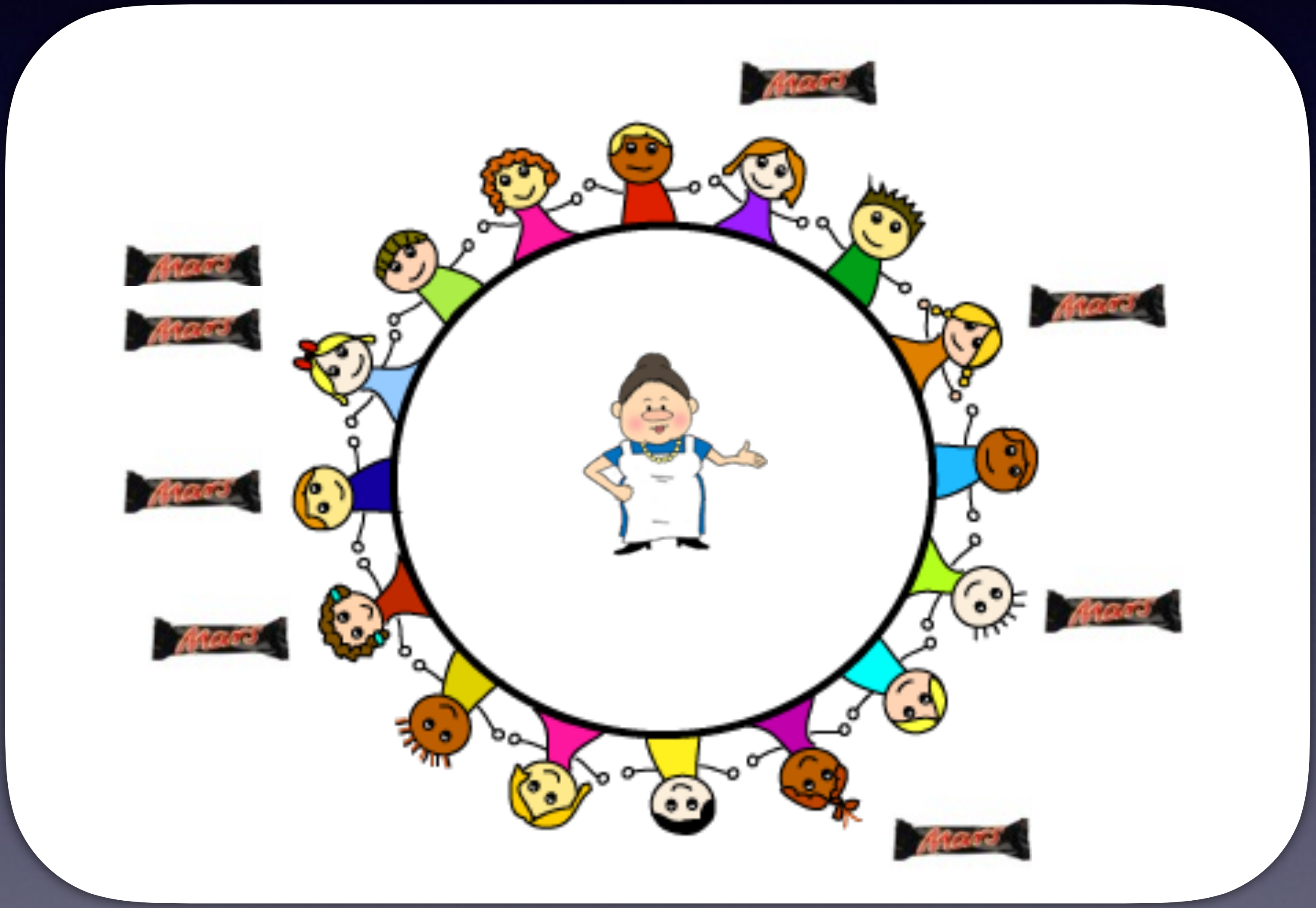
Israeli-Jalfon Randomised Self-Stabilising Protocol



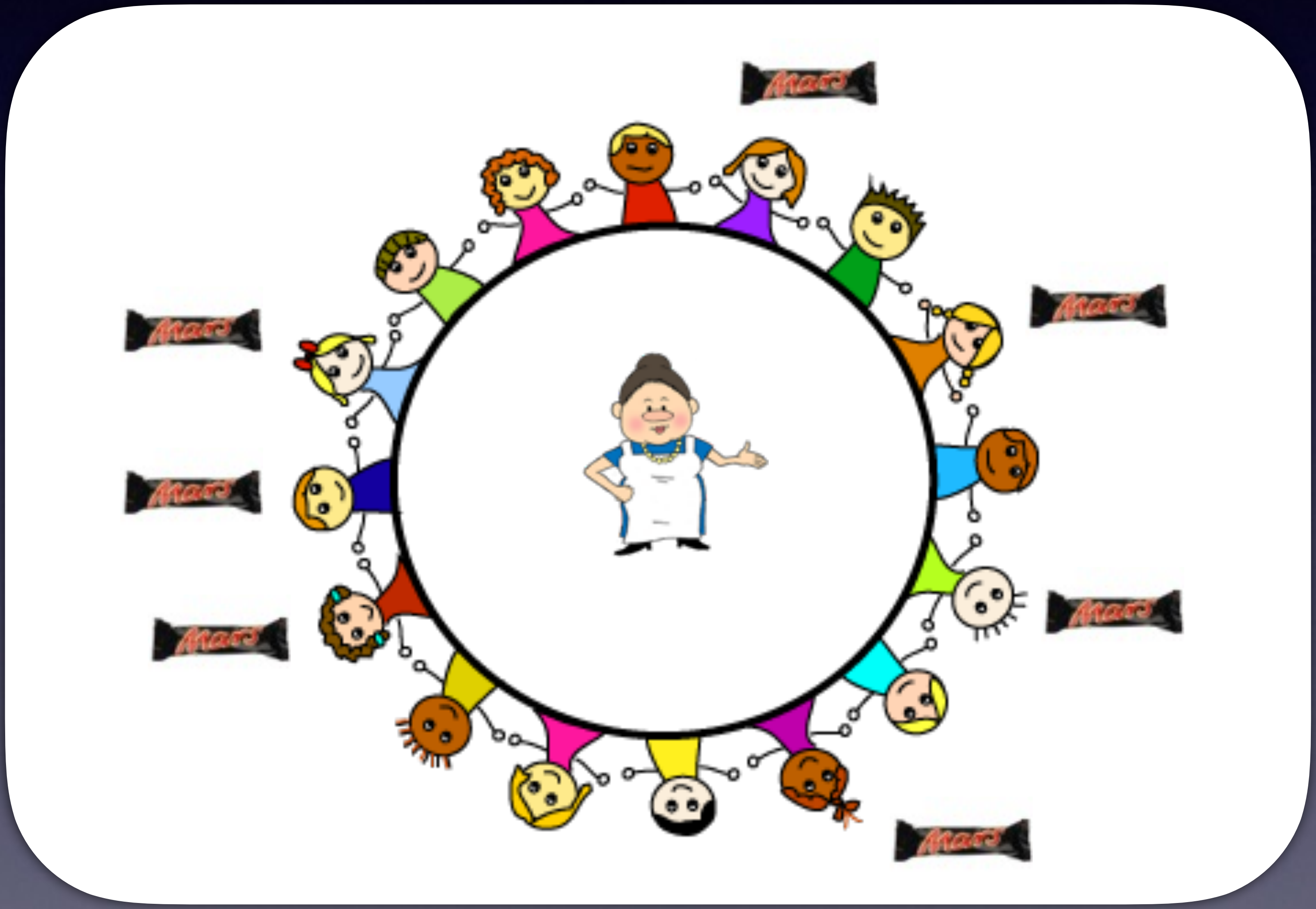
Israeli-Jalfon Randomised Self-Stabilising Protocol



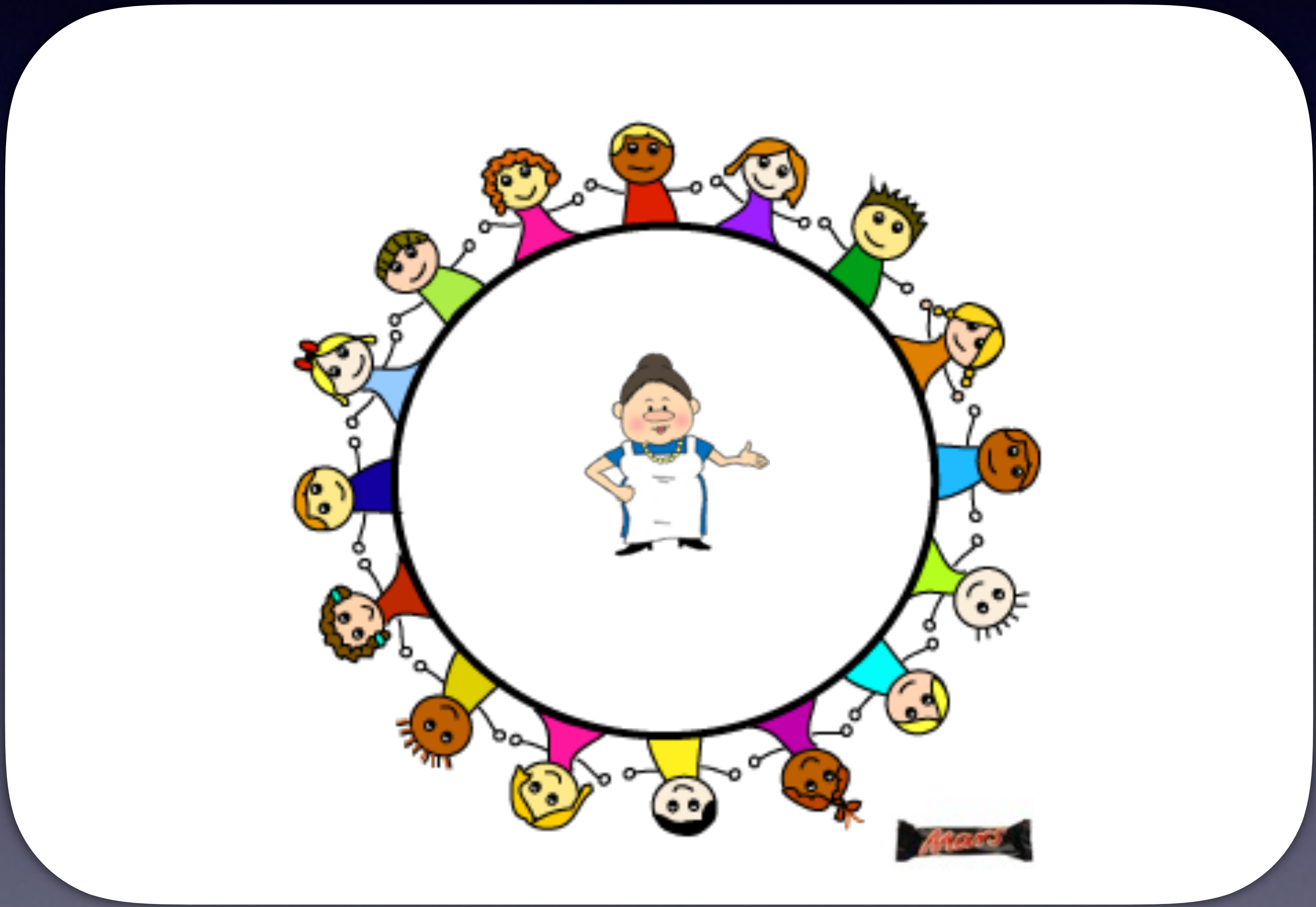
Israeli-Jalfon Randomised Self-Stabilising Protocol



Israeli-Jalfon Randomised Self-Stabilising Protocol



Israeli-Jalfon Randomised Self-Stabilising Protocol



Probability of reaching a stable configuration from any faulty configuration under arbitrary schedulers is 1

Liveness (a.k.a. almost-sure termination)

Probability of reaching a target set of states from any initial state for the system under arbitrary schedulers is 1

(1) Can be unfair

(2) Desirable property in self-stabilising protocol literature

Liveness for Parameterised Systems

- Infinite-state verification (verify for **each** instance)
- Challenging esp. for probabilitistic systems, e.g.,
  - Randomised Self-Stabilising (Israeli-Jalfon/Herman)
  - Randomised Dining Philosopher (Lehmann-Rabin)

reachability games on infinite graphs

Regular Model Checking: Symbolic Framework

Regular Specification

“Rich language for specifying parameterised systems using automata”

Pioneered by:

* Kesten, Maler, Marcus, Pnueli, and Shahar (1997)
* Wolper and Boigelot (1998)
* Jonsson and Nilsson (2000)
* Bouajjani, Jonsson, Nilsson, and Touili (2000)

Premier of regular specifications

<u>Configuration</u>: represented as a word

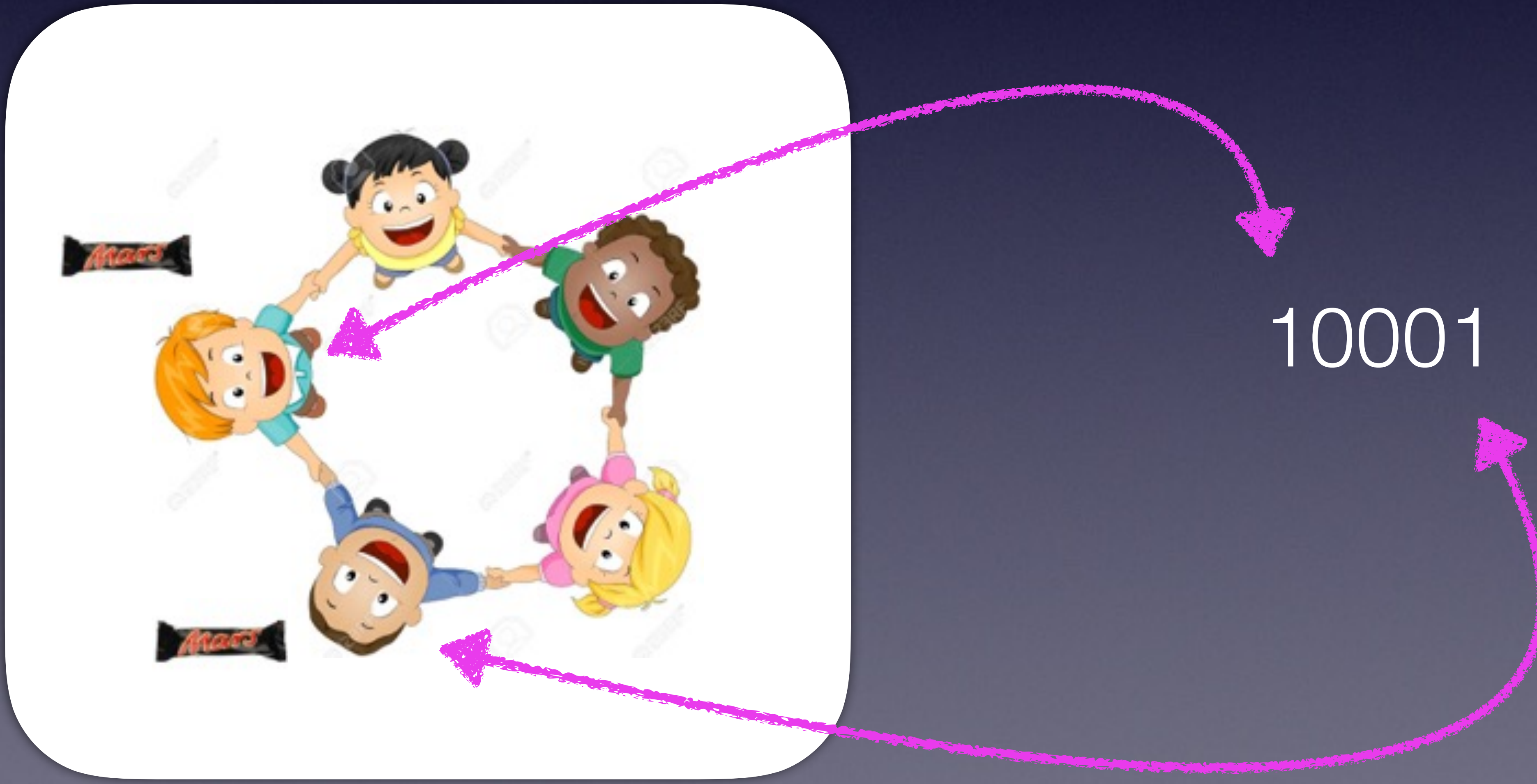
<u>Set of configurations</u>: represented as a regular automaton

<u>Transition relation</u>: represented as a transducer

Length-preserving

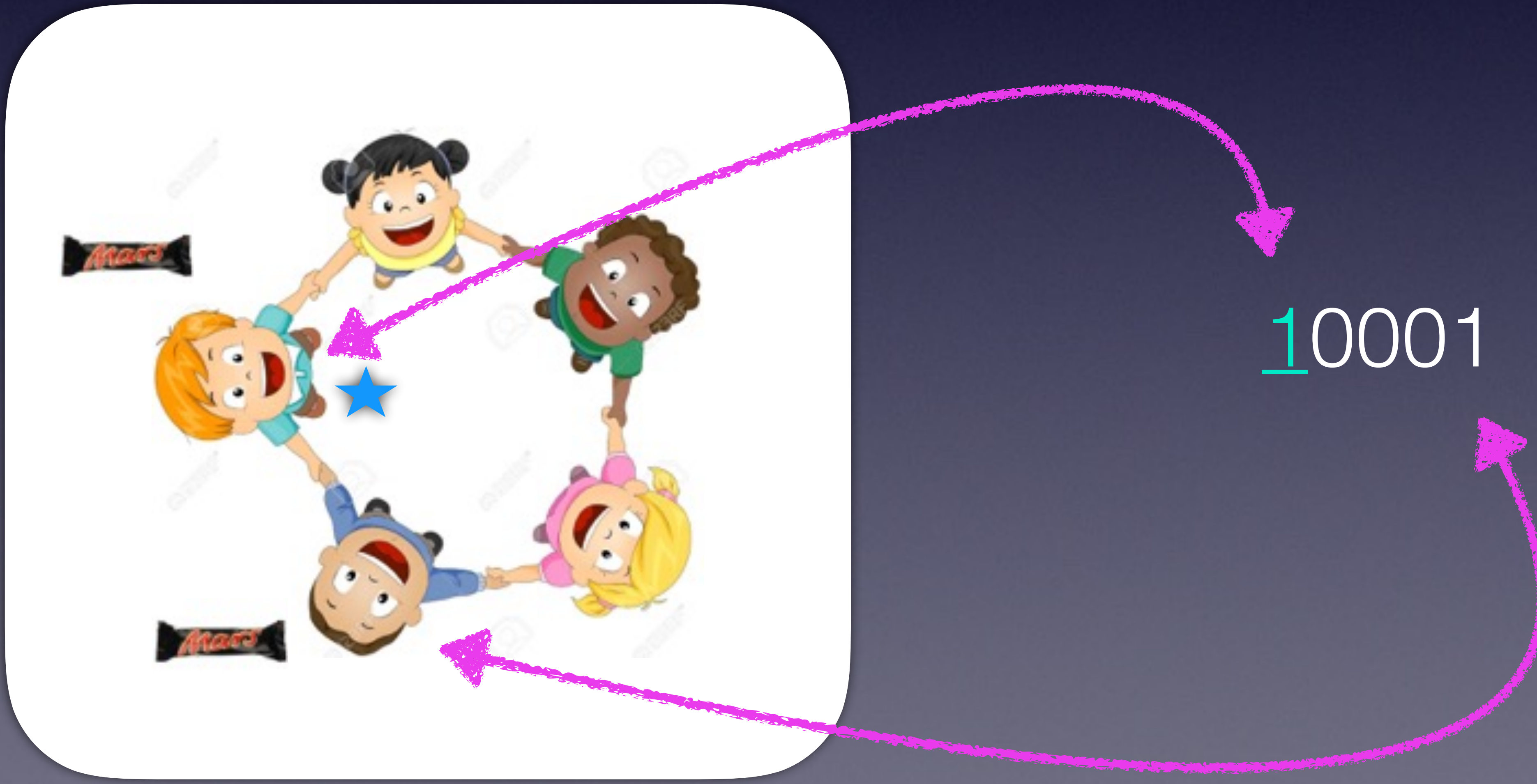
Israeli-Jalfon as a regular specification

Configuration: a word over the alphabet $\{0,1,\underline{1}\}$



Israeli-Jalfon as a regular specification

Configuration: a word over the alphabet {0,1,$\underline{1}$}



Israeli-Jalfon as a regular specification

Set of configurations: a regular language over $\{0,1,\underline{1}\}$

All stable configurations

0*10*

All initial configurations

1+

Israeli-Jalfon as a regular specification

Nondeterministic transition relation: a regular language over $\{0,1\} \times \{0,1,\underline{1}\}$

10001

↓

$\underline{1}$0001

Israeli-Jalfon as a regular specification

Nondeterministic transition relation: a regular language over {0,1} x {0,1,$\underline{1}$}

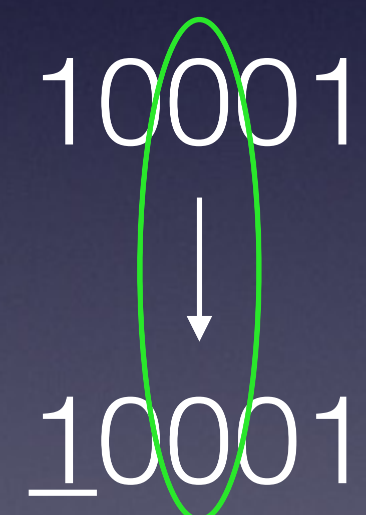
10001

↓

$\underline{1}$0001

Israeli-Jalfon as a regular specification

<u>Nondeterministic transition relation</u>: a regular language over $\{0,1\} \times \{0,1,\underline{1}\}$



Israeli-Jalfon as a regular specification

Nondeterministic transition relation: a regular language over $\{0,1\} \times \{0,1,\underline{1}\}$

10001

↓

$\underline{1}$0001

Israeli-Jalfon as a regular specification

<u>Nondeterministic transition relation</u>: a regular language over {0,1} x {0,1,$\underline{1}$}

10001

↓

$\underline{1}$0001

$$L = \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^* \begin{bmatrix} 1 \\ \underline{1} \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^*$$

Israeli-Jalfon as a regular specification

Problem: How do you represent probabilistic transitions as transducers?

Answer: almost sure liveness for finite MDPs, need only distinguish zero or non-zero probabilities

Proposition (Hart et al.'83): almost sure liveness = 2-player non-stochastic reachability games

Generalises to infinite family of finite MDPs (why?)

Israeli-Jalfon as a regular specification

<u>Probabilistic transition relation</u>: a regular language over $\{0,1,\underline{1}\} \times \{0,1\}$

$$\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^* \begin{bmatrix} \underline{1} \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^*$$

Pass to right (w/o Mars bar)

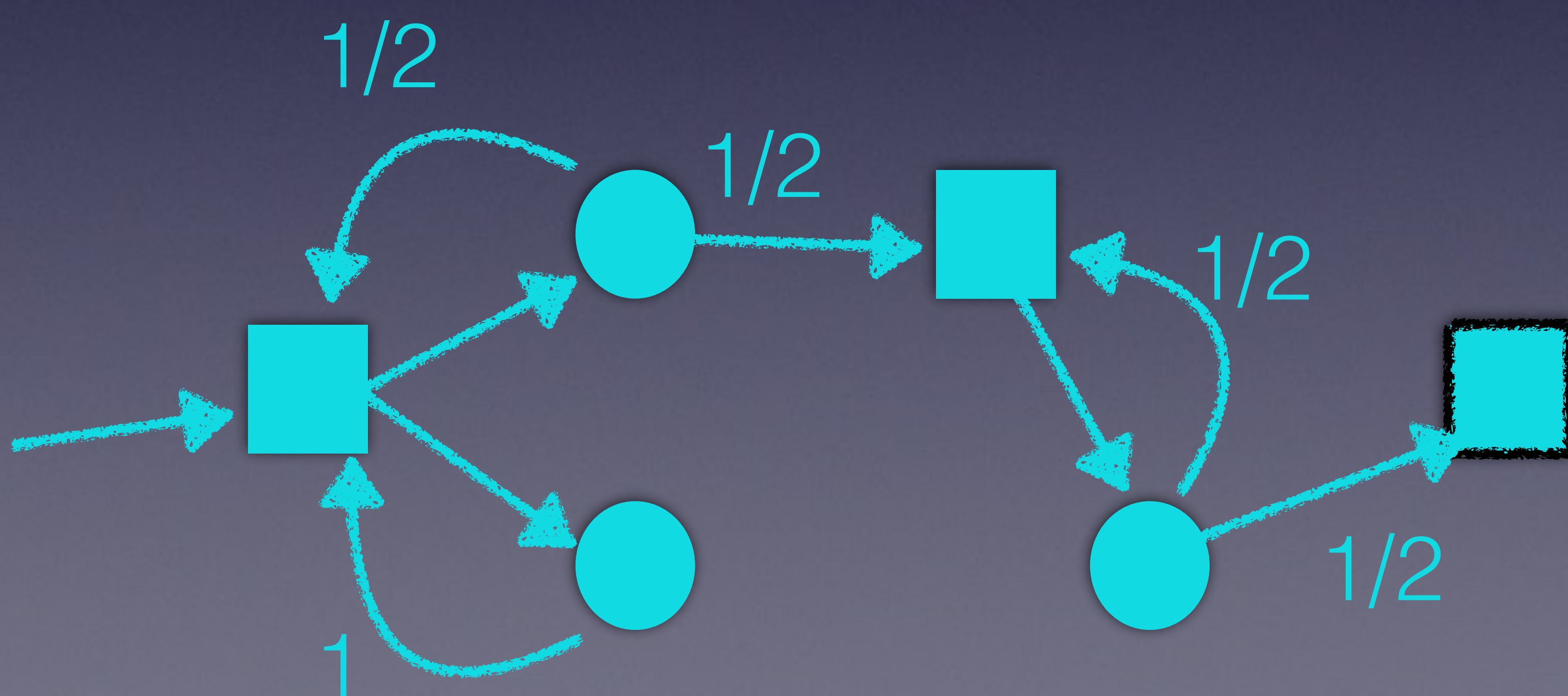
$$\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^* \begin{bmatrix} \underline{1} \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^*$$

Pass to right (with Mars bar)

………. (~10 more cases)

Semi-decision procedure

Proposition (Hart et al.'83): almost sure liveness = wins non-stochastic reachability games from <u>each</u> reachable state.



Semi-decision procedure

Prop (LR'16): 's winning strategies can be represented as "advice bits"

$$\langle A, \prec \rangle$$

$$A \subseteq S \quad , \quad \prec \subseteq S \times S$$

Inductive invariant

Well-founded relation that guides to win

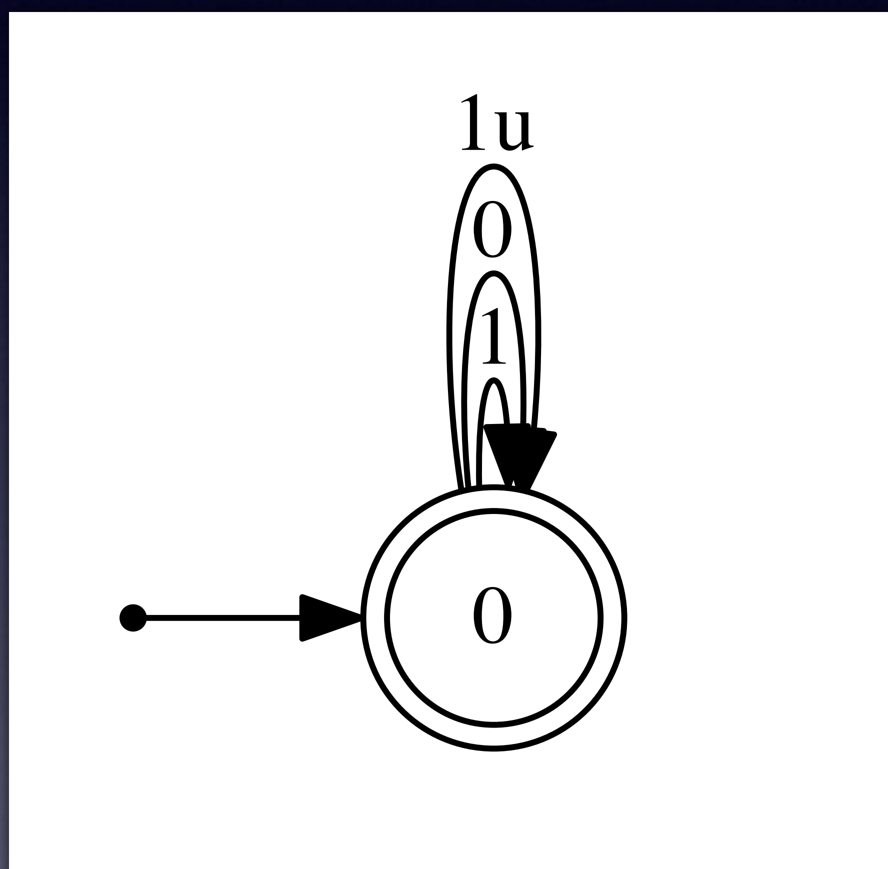
Semi-decision procedure

- Advice bits $\langle A, \prec \rangle$ are infinite objects
- **Solution**: represent A by an automaton and $\prec$ by a transducer ("regular advice bits")

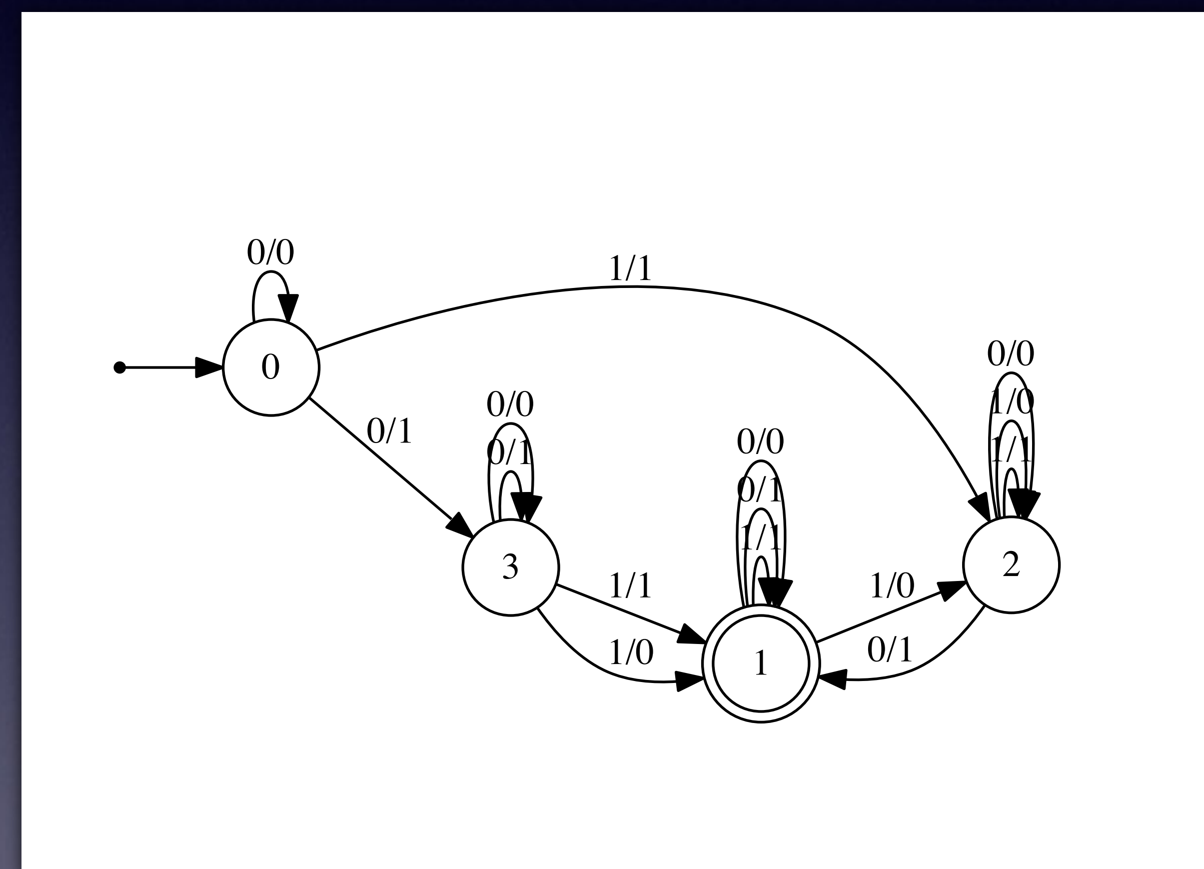
Prop: There exists a complete algorithm for verifying regular advice bits

Regular advice bits often exist in practice

Regular advice bits for Israeli-Jalfon



A



$\prec$

Learning Regular Advice Bits

Problem

Although regular advice bits exist, a **naive enumeration** might take a long time to find them

Our monolithic learning procedure

Learner

Teacher

Regular
advice bits?

NO

(cex)

YES

DONE

Inside the learner

SAT-solving to guess smallest DFAs

Boolean formulas constraining candidate regular advice bits

Inside the teacher

Automata-based algorithm

If incorrect advice bits,
return cex
(as a boolean formula)

The learner then …

Add the counterexample constraint from Teacher to further restrict

And make another guess, etc.

The main bottleneck

The number of iterations

~

The number of candidate regular advice bits considered

Each iteration is quite cheap

Further optimisations

Problem: When no “small” regular proof exists, monolithic procedure becomes very slow

- **Incremental learning algorithm**: use “disjunctive” advice bits
- Precomputation of inductive invariant with **Angluin’s L* algorithm**
- **Symmetries** (e.g. rotations for rings)

Experiments

(https://github.com/uuverifiers/ autosat/tree/master/
LivenessProver)

Experimental results

	Mono	Incr	Incr+Inv	Incr+Symm	Incr+Inv+Symm
Randomised parameterised systems					
Lehmann-Rabin (DP) [34]	T/O	T/O	T/O	48min	10min
Israeli-Jalfon [47]	4.6s	22.7s	21.4s	9.9s	9.7s
Herman [46]	1.5s	1.6s	2.4s	—	—
Firewire [35, 60]	1.3s	1.3s	2.0s	—	—
Deterministic parameterised systems					
Szymanski [4, 65]	5.7s	27min	10min	—	—
DP, left-right strategy	1.9s	6.4s	3.4s	—	—
Bakery [4, 65]	1.6s	2.7s	1.9s	—	—
Resource allocator [32]	2.2s	2.2s	2.0s	—	—
Games on infinite graphs					
Take-away [38]	2.8s	—	—	—	—
Nim [38]	5.3s	—	—	—	—

Experimental results

	Mono	Incr	Incr+Inv	Incr+Symm	Incr+Inv+Symm
Randomised parameterised systems					
Lehmann-Rabin (DP) [34]	T/O	T/O	T/O	48min	10min
Israeli-Jalfon [47]	4.6s	22.7s	21.4s	9.9s	9.7s
Herman [46]	1.5s	1.6s	2.4s	—	—
Firewire [35, 60]	1.3s	1.3s	2.0s	—	—
Deterministic parameterised systems					
Szymanski [4, 65]	5.7s	27min	10min	—	—
DP, left-right strategy	1.9s	6.4s	3.4s	—	—
Bakery [4, 65]	1.6s	2.7s	1.9s	—	—
Resource allocator [32]	2.2s	2.2s	2.0s	—	—
Games on infinite graphs					
Take-away [38]	2.8s	—	—	—	—
Nim [38]	5.3s	—	—	—	—

Conclusion

Summary of results

- Automatic method for proving liveness for randomised parameterised systems, e.g.,
  - Randomised Self-Stabilising (Israeli-Jalfon/Herman)
  - Randomised Dining Philosopher (Lehmann-Rabin)
- Regular model checking as symbolic framework
- CEGAR/Learning to synthesise “regular proofs”

Future Work

- Embedding fairness in RMC
  - *New result (joint with O. Lengal, R. Majumdar)*
- Extend the framework to encode process IDs