Synthesising Strategy Improvement and Recursive Algorithms for Solving 2.5 Player Parity Games

Sven Schewe

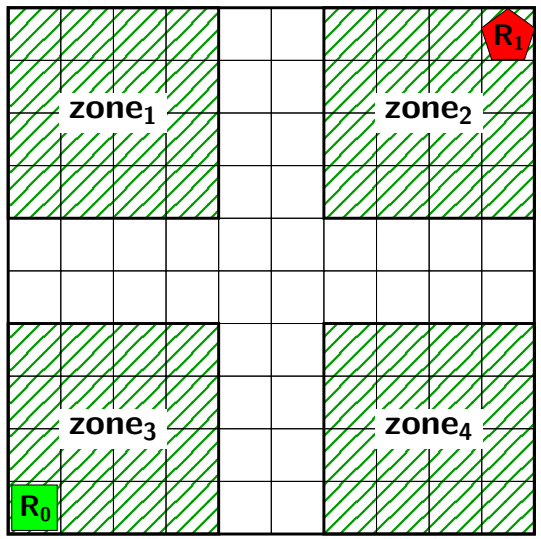
University of Liverpool

joint work with Ernst Moritz Hahn, Andrea Turrini & Lijun Zhang

State Key Lab. of Computer Science, Institute of Software, CAS, Beijing, China

Qualitative Probabilistic Parity Games

- Good abstraction for systems with
  - choices under our control
  - choices under environment control
  - randomised choices
- Suitable for linear time properties

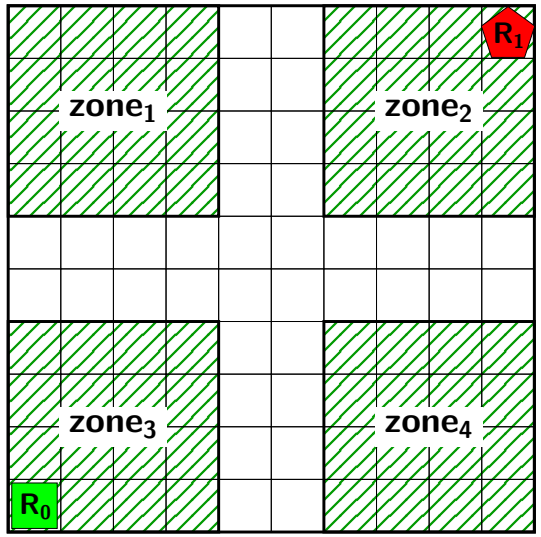


$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\geq 1}[\mathbf{GF}(\mathrm{zone}_1 \wedge \mathbf{F}\mathrm{zone}_2)]$

- Our goal is to win the game with probability 1.

Quantitative Probabilistic Parity Games

- Good abstraction for systems with
  - choices under our control
  - choices under environment control
  - randomised choices
- Suitable for linear time properties



$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\max=?}[\mathbf{GF}(\mathrm{zone}_1 \wedge \mathbf{F}\mathrm{zone}_2)]$

- Our goal is to maximise the probability to win the game.

Solving Quantitative Parity Games

Solving Qualitative Parity Games

- invites similar techniques as non-stochastic parity games

$\Rightarrow$ McNaughton style algorithms stile algorithms

CAV 2016

or: reduction to non-stochastic parity games

Solving Quantitative Reachability Games

- strategy improvement

Part

2 Player Games

An Abundance of Algorithms

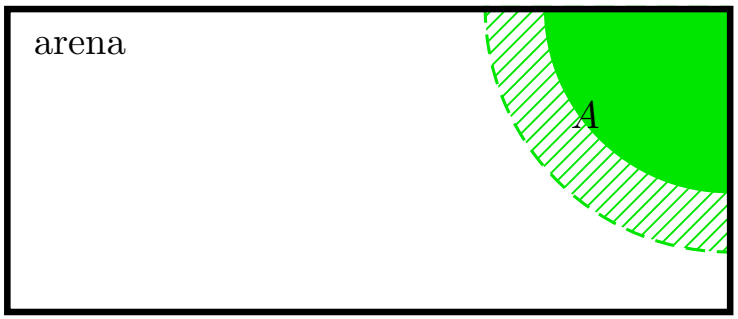
- McNaughton-style algorithms are the oldest and fastest $\mathcal{O}(m\, n^{c-1})$
- many theoretical improvements
  - counting algorithms, roughly $\mathcal{O}(m\, n^{c/2})$
  - Big Step algorithms, roughly $\mathcal{O}(m\, n^{c/3})$
  - quasi polynomial, roughly $\mathcal{O}(n^{\log c})$
- various strategy improvement techniques

McNaughton's Algorithm

McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the minimal priority, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$(\alpha^{-1}(c))$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

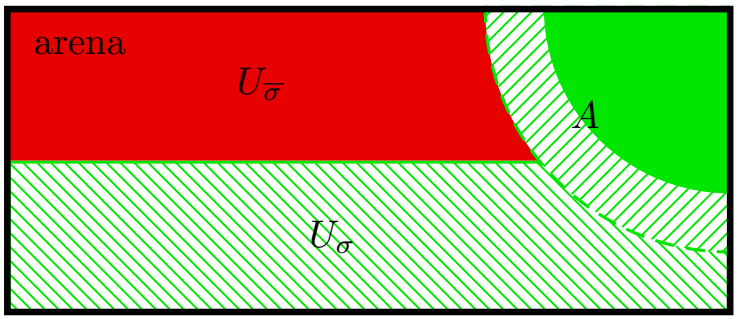
McNaughton's Algorithm



McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the minimal priority, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

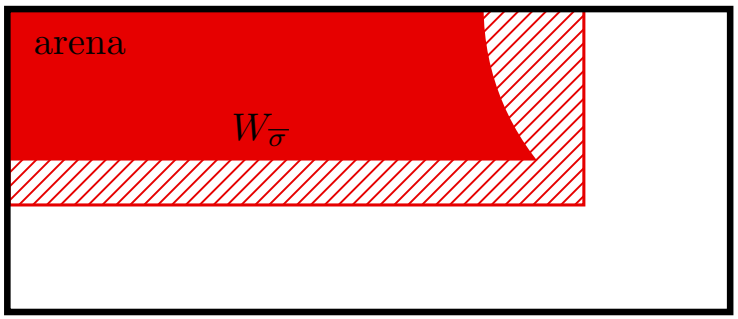
McNaughton's Algorithm



McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the minimal priority, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$(\alpha^{-1}(c))$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

McNaughton's Algorithm



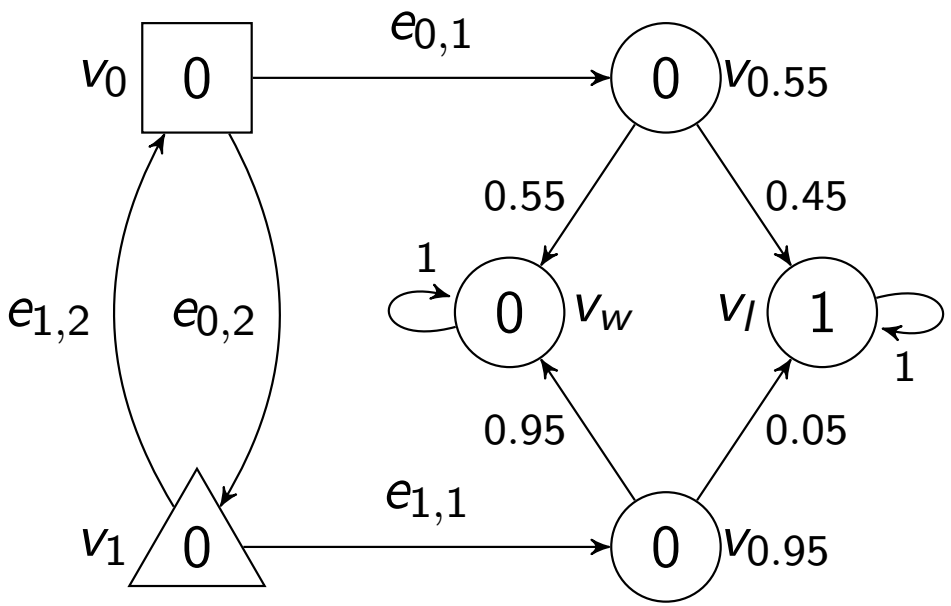
McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the minimal priority, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

Part I

2.5 Player Games

A Toy Example



Overview

Qualitative Solution

- native – adjust McNaughton
  CAV 2016 — fast
- through gadgets – reduction to 2 player
  classic — complexity
- experimental results: native faster then building reduction

Quantitative Solution

- integrated strategy improvement

$\rightsquigarrow$ entangled improvement of

  - quantitative solution — like reachability
  - qualitative solution — **on** plateaux

- quantitative solution (SI) — like reachability
- qualitative solution — **for** plateaux
  – fed back as strategy

Part II

Qualitative Solution

Different Attractors

Weak attractors

- reach goal almost surely

Strong attractors

- reach goal with positive probability

$\Rightarrow$ treat probabilistic nodes as "yours"

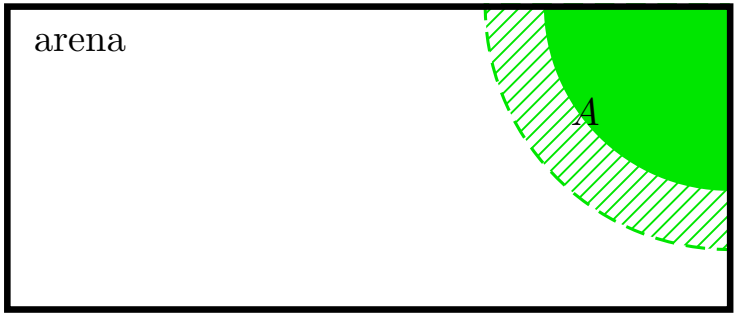
Note: weak attractor(G) $\subseteq$ strong attractor(G)

Probabilistic McNaughton

Prob-McNaughton's Algorithm – for $P = \langle V_0, V_1, V_R, E, \alpha \rangle$

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$(\alpha^{-1}(c))$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

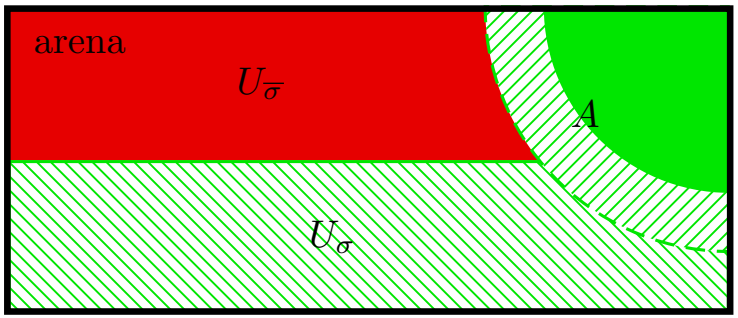
Probabilistic McNaughton



Prob-McNaughton's Algorithm – for $P = \langle V_0, V_1, V_R, E, \alpha \rangle$

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

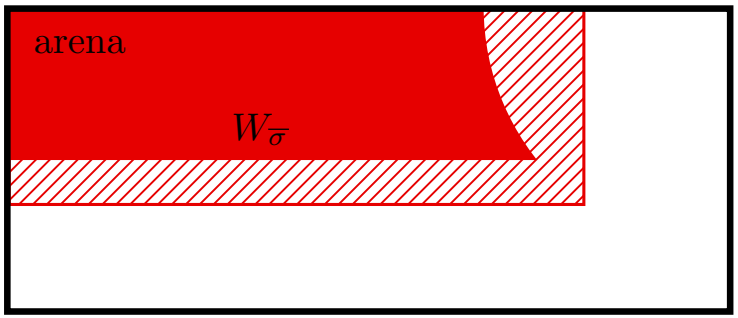
Probabilistic McNaughton



Prob-McNaughton's Algorithm – for $P = \langle V_0, V_1, V_R, E, \alpha \rangle$

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

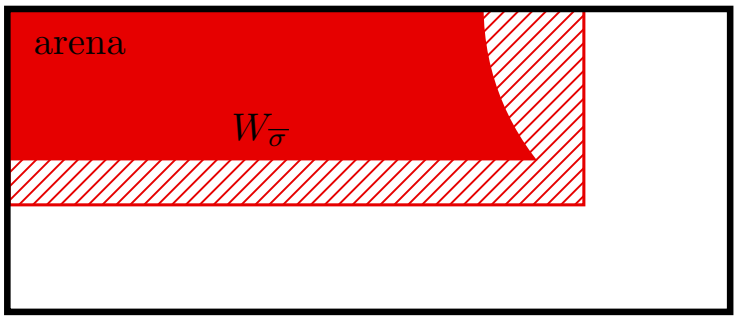
Probabilistic McNaughton – $\sigma = 0, \overline{\sigma} = 1$



Prob-McNaughton's Algorithm – for $P = \langle V_0, V_1, V_R, E, \alpha \rangle$

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

Probabilistic McNaughton – $\sigma = 1, \overline{\sigma} = 0$



Prob-McNaughton's Algorithm – for $P = \langle V_0, V_1, V_R, E, \alpha \rangle$

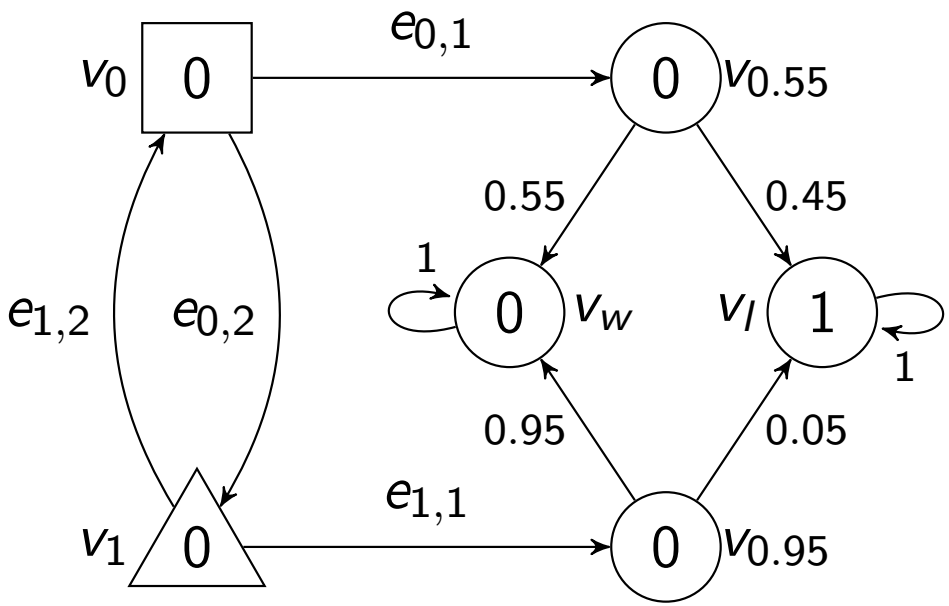
- use weak attractor
  (instead of the strong attractor)
- co-game of the weak attractor might have sub-stochastic vertices
- they are turned into "good" vertices, by
  - giving them priority 0 or
  - adding an accepting sink as successor

$\Rightarrow$ visiting them infinitely often wins

Part III

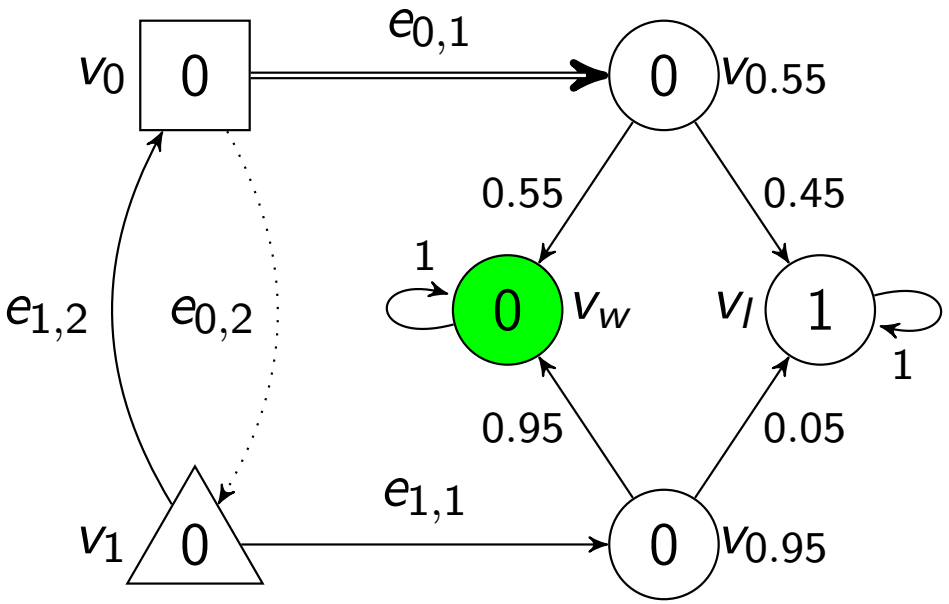
Quantitative Solution

Quantitative Solution



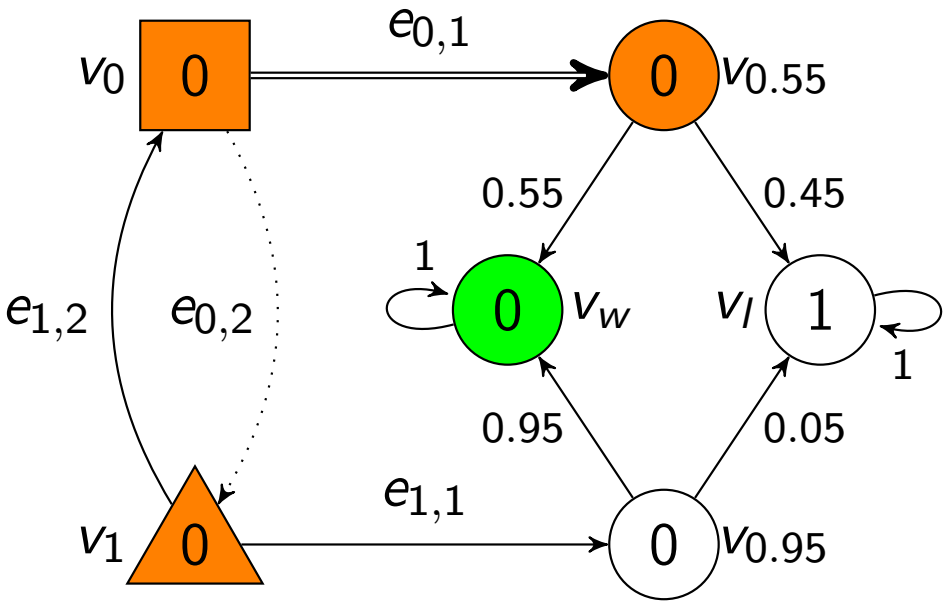
1. Find Winning Region

& fix winning strategy

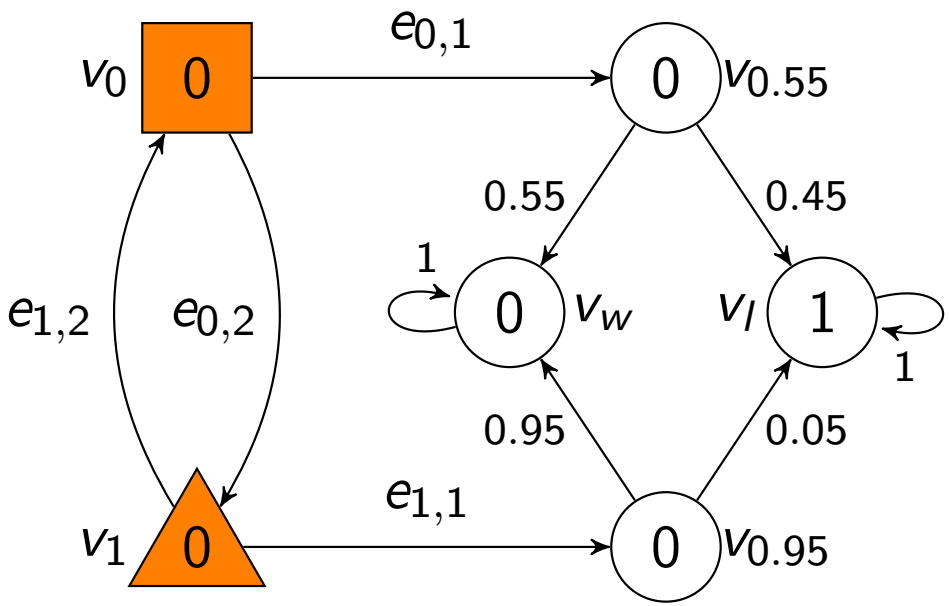


2. Solve MDP

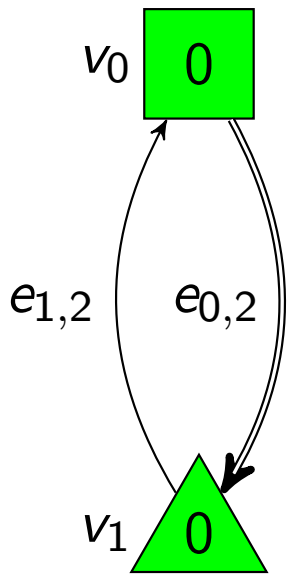
& identify area/s with equal value



3. Check where you can stay . . .

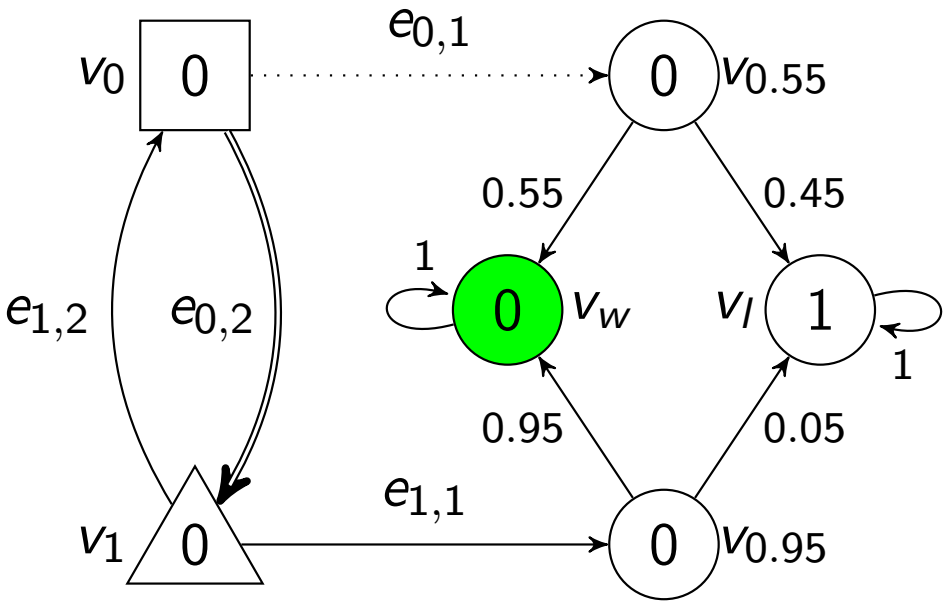


. . . & Find Winning Region



Go to 2. (Transfer Back)

& evaluate parity MDP



Part IV

Evaluation

Robots analysis: different reachability properties

property	n	b	MPG vertices	MPG colours	$p_{destr} = 0.1$ $p_{\max}$	$p_{destr} = 0.1$ t_{sol}	$p_{destr} = 0.3$ $p_{\max}$	$p_{destr} = 0.3$ t_{sol}	$p_{destr} = 0.5$ $p_{\max}$	$p_{destr} = 0.5$ t_{sol}	$p_{destr} = 0.7$ $p_{\max}$	$p_{destr} = 0.7$ t_{sol}	$p_{destr} = 0.9$ $p_{\max}$	$p_{destr} = 0.9$ t_{sol}
Reachability $\langle\langle R_0 \rangle\rangle \mathcal{P}_{\max=?}$ [$\mathbf{F}\text{zone}_1 \wedge \mathbf{F}\text{zone}_2 \wedge \mathbf{F}\text{zone}_3 \wedge \mathbf{F}\text{zone}_4$]	7	1	663 409	2	0.9614711	33	0.8178044	22	0.6247858	22	0.3961410	21	0.1384328	23
	7	2	1 090 537	2	0.9244309	56	0.6742610	66	0.4017138	57	0.1708971	58	0.0230085	52
	7	3	1 517 665	2	0.8926820	89	0.5793073	87	0.2995397	77	0.0953904	86	0.0060025	68
	7	4	1 944 793	2	0.8667039	112	0.5385632	109	0.2409219	96	0.0649772	107	0.0026513	85
	7	5	2 371 921	2	0.8571299	147	0.5062357	144	0.2167625	127	0.0506530	140	0.0019157	112
Ordered Reachability $\langle\langle R_0 \rangle\rangle \mathcal{P}_{\max=?}$ [$\mathbf{F}(\text{zone}_1 \wedge \mathbf{F}\,\text{zone}_2)$]	8	1	528 168	2	0.9613511	23	0.8176058	19	0.6246643	21	0.3962011	20	0.1384974	19
	8	2	868 986	2	0.9243652	35	0.6999023	44	0.4522051	35	0.2083732	42	0.0320509	40
	8	3	1 209 804	2	0.9091132	62	0.6538475	71	0.3643938	56	0.1352710	60	0.0131408	58
	8	4	1 550 622	2	0.9013742	91	0.6200998	91	0.3316778	72	0.1168758	74	0.0097312	71
	8	5	1 891 440	2	0.8977303	113	0.6031945	108	0.3207408	90	0.1138603	88	0.0093679	83
Reach-Avoid $\langle\langle R_0 \rangle\rangle \mathcal{P}_{\max=?}$ [$\neg\text{zone}_1\ \mathbf{U}\ \text{zone}_2 \wedge \neg\text{zone}_4\ \mathbf{U}\ \text{zone}_2 \wedge \mathbf{F}\text{zone}_3$]	9	1	833 245	4	0.9447793	46	0.8005413	31	0.6125397	35	0.3914531	25	0.1372075	24
	9	2	1 370 827	4	0.9095579	81	0.6824329	52	0.4411181	61	0.2089446	49	0.0302023	45
	9	3	1 908 409	4	0.8972146	108	0.6375883	68	0.3792906	84	0.1444959	71	0.0106721	66
	9	4	2 445 991	4	0.8936231	148	0.6221536	93	0.3478172	117	0.1158094	103	0.0051508	89
	9	5	2 983 573	4	0.8918034	172	0.6162166	109	0.3366050	136	0.1010400	120	0.0035468	105
Reachability $\langle\langle R_0 \rangle\rangle \mathcal{P}_{\max=?}$ [$\mathbf{F}\text{zone}_1 \wedge \mathbf{F}\text{zone}_2 \wedge \mathbf{F}\text{zone}_3 \wedge \mathbf{F}\text{zone}_4$]	10	1	3 307 249	2	0.9614711	186	0.8178044	141	0.6247858	142	0.3961410	142	0.1384328	141
	10	2	5 440 429	2	0.9244267	296	0.6755372	414	0.4017718	374	0.1665626	732	0.0207851	615
	10	3	7 573 609	2	0.8931881	570	0.5742127	572	0.2864117	509	0.0847474	1019	0.0043153	861
	10	4	9 706 789	2	0.8676441	530	0.5239018	794	0.2248369	735	0.0479367	1396	0.0009959	1610
	10	5	11 839 969	2	0.8503684	968	0.4885654	980	0.1866995	971	0.0305890	1708	—TO—	

Summary

Solving probabilistic parity games:

- fast
- simple
- beautiful

. . . and the same class of algorithms works best