Synthesis Algorithms (continued)

Rayna Dimitrova
University of Leicester

Midlands Graduate School 2019

Acknowledgement: Many slides courtesy of Bernd Finkbeiner.

Attractor construction

$$
\begin{aligned}
Attr_i^0(R) &= R \\
Attr_i^{n+1}(R) &= Attr_i^n(R) \cup CPre_i(Attr_i^n(R))
\end{aligned}
$$

$$
Attr_i(R) = \bigcup_{n \in \mathbb{N}} Attr_i^n(R)
$$

where

$$
\begin{aligned}
CPre_i(R) = &\{ v \in V_i \mid \exists v' \in V. (v, v') \in E \wedge v' \in R \} \\
&\cup \{ v \in V_{1-i} \mid \forall v' \in V. (v, v') \in E \Rightarrow v' \in R \}
\end{aligned}
$$

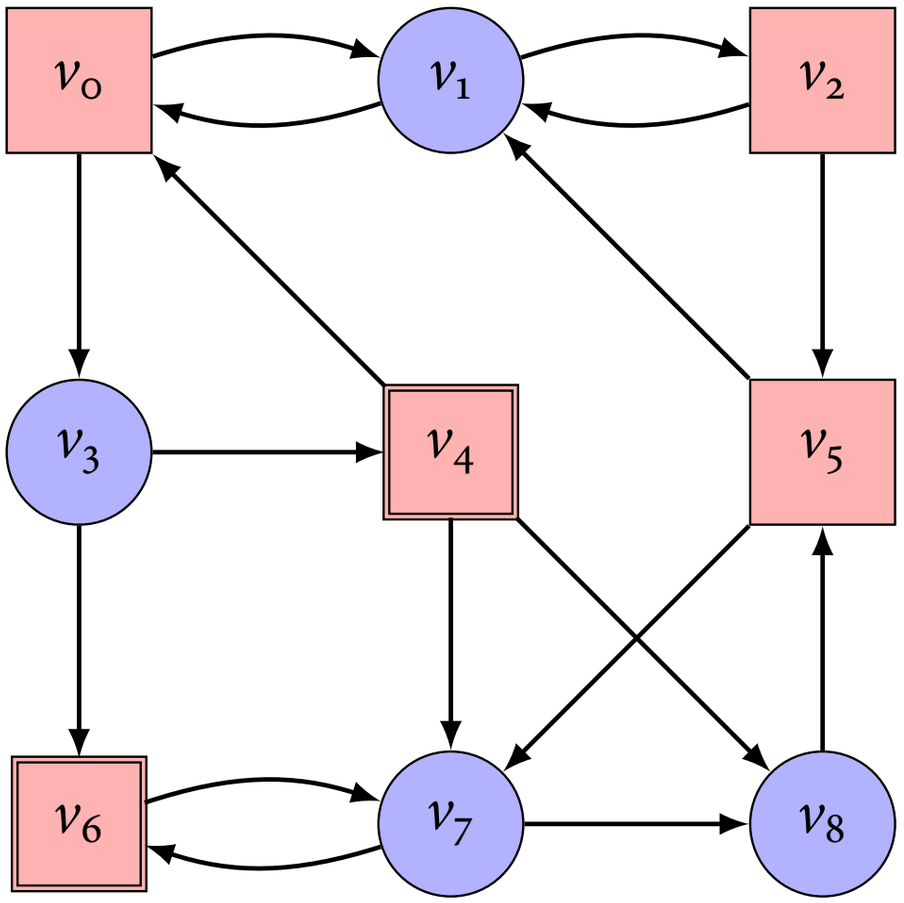
Winning regions of a reachability game

Winning region of Player 0: $W_0(\mathcal{G}) = Attr_0(R)$

Winning region of Player 1: $W_1(\mathcal{G}) = V \setminus Attr_0(R)$

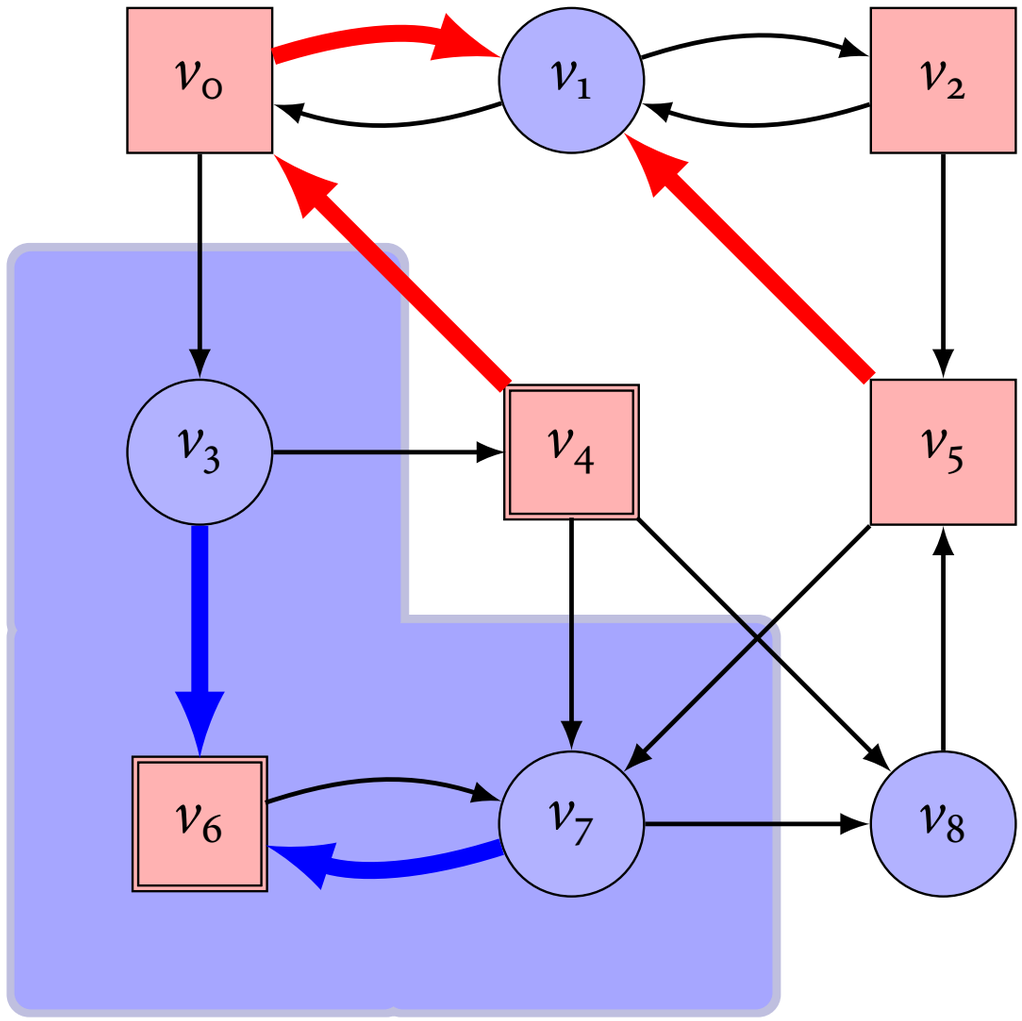
Büchi games

Büchi game: Player 0 wins a play π if π visits F infinitely often, otherwise Player 1 wins.



Büchi games

Büchi game: Player 0 wins a play π if π visits F infinitely often, otherwise Player 1 wins.



Recurrence construction

$$
\begin{aligned}
Recur^{0}(F) &= F \\
W_1^n(F) &= V \setminus Attr_0(Recur^n(F)) \\
Recur^{n+1}(F) &= Recur^n(F) \setminus CPre_1(W_1^n(F))
\end{aligned}
$$

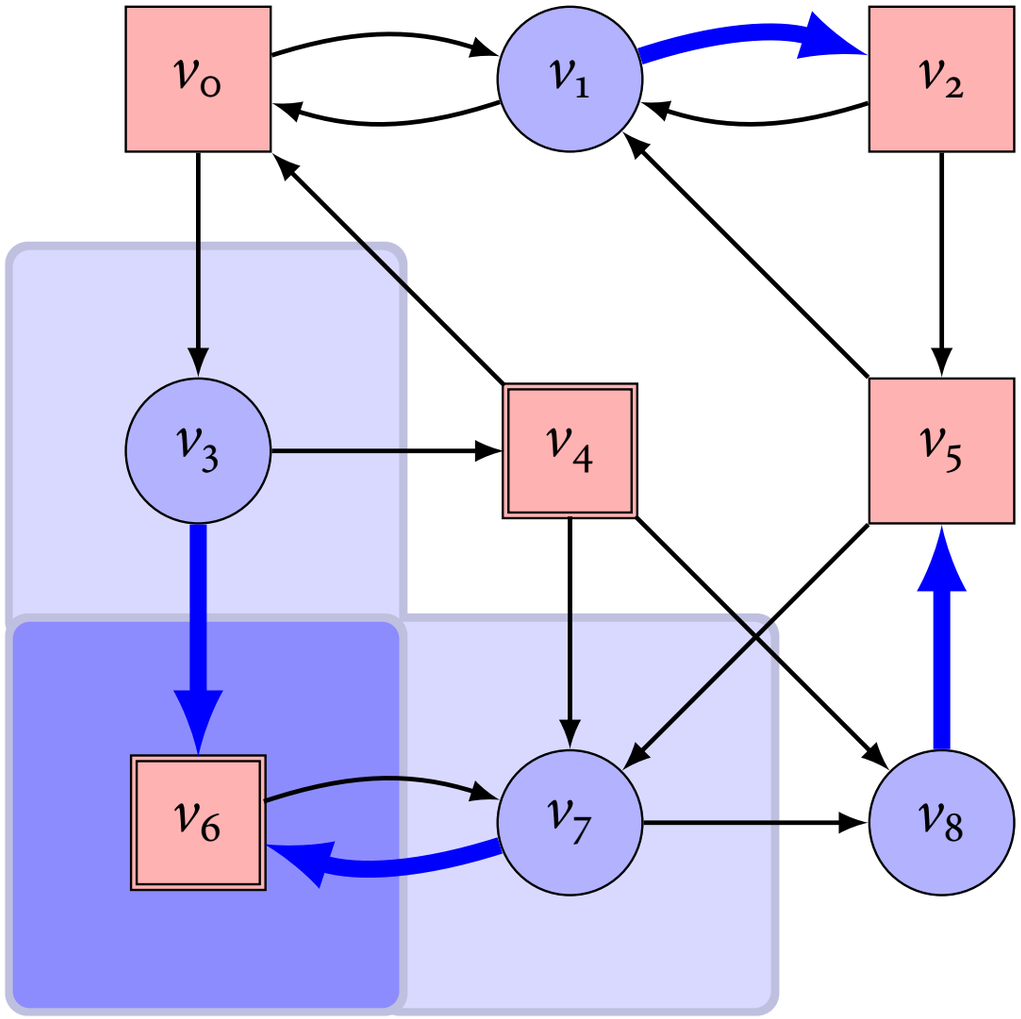
$$
Recur(F) = \bigcap_{n \in \mathbb{N}} Recur^n(F)
$$

Winning regions of a Büchi game

Winning region of Player 0: $W_0(\mathcal{G}) = Attr_0(Recur(F))$

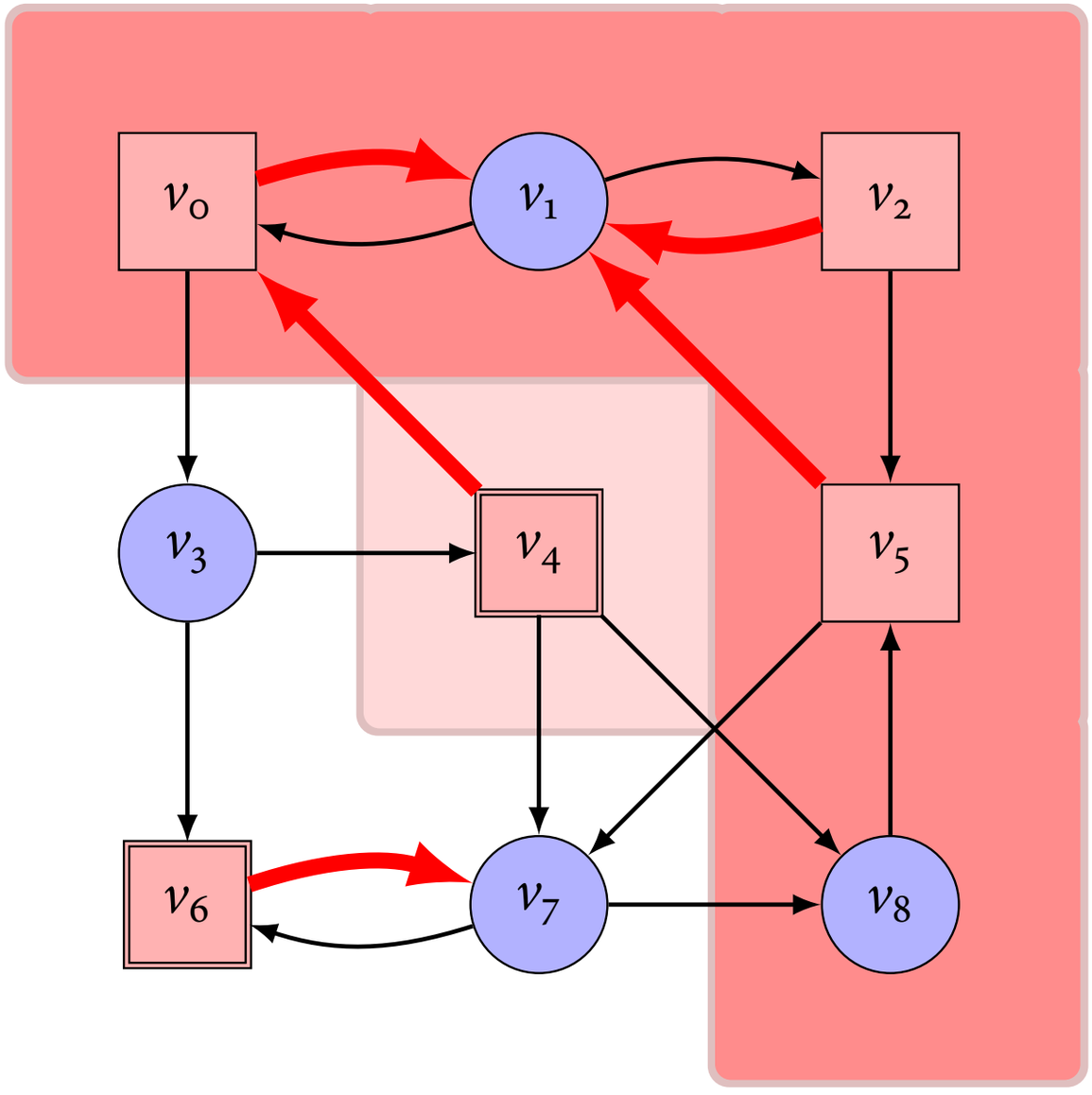
Winning region of Player 1: $W_1(\mathcal{G}) = V \setminus Attr_0(Recur(F))$

Winning strategy of Player o (Büchi Strategy)



The winning strategy for Player o always moves to $Attr_{\mathrm{o}}^{n}(Recur(F))$ for the smallest possible n.

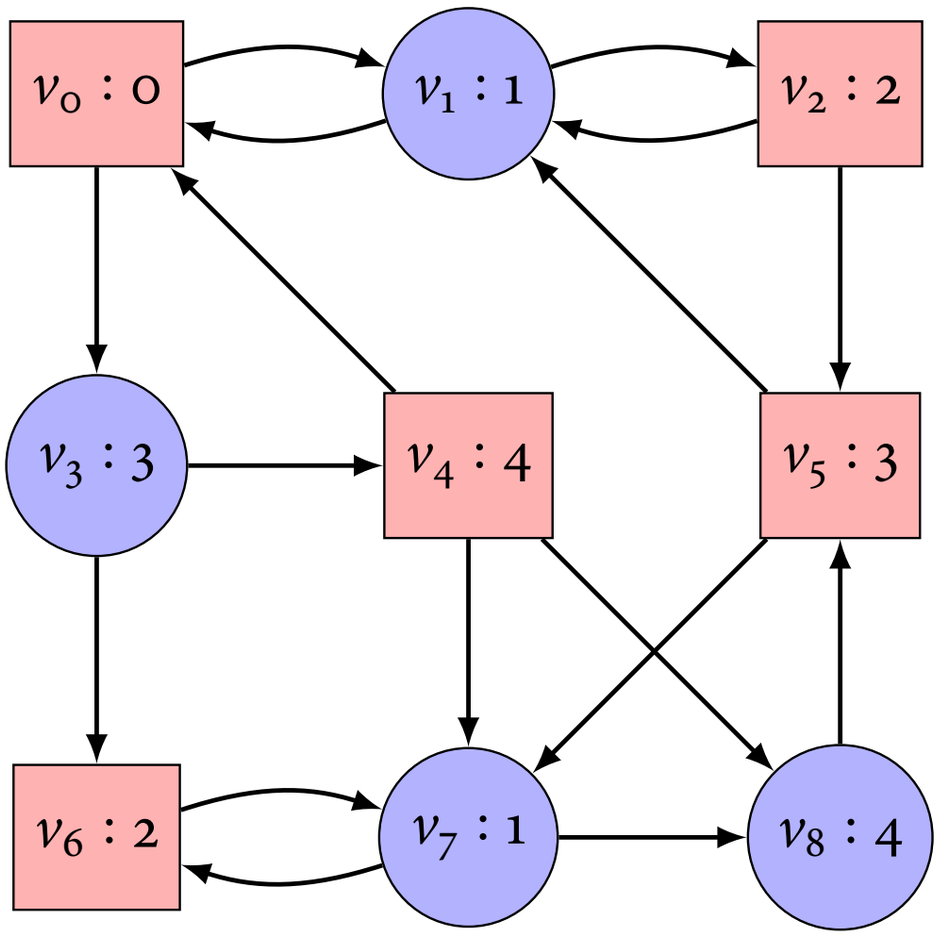
Winning strategy of Player 1 (co-Büchi Strategy)



The winning strategy for Player 1 moves from a state in W_1^n to W_1^{n-1} whenever possible, and stays in W_1^n otherwise.

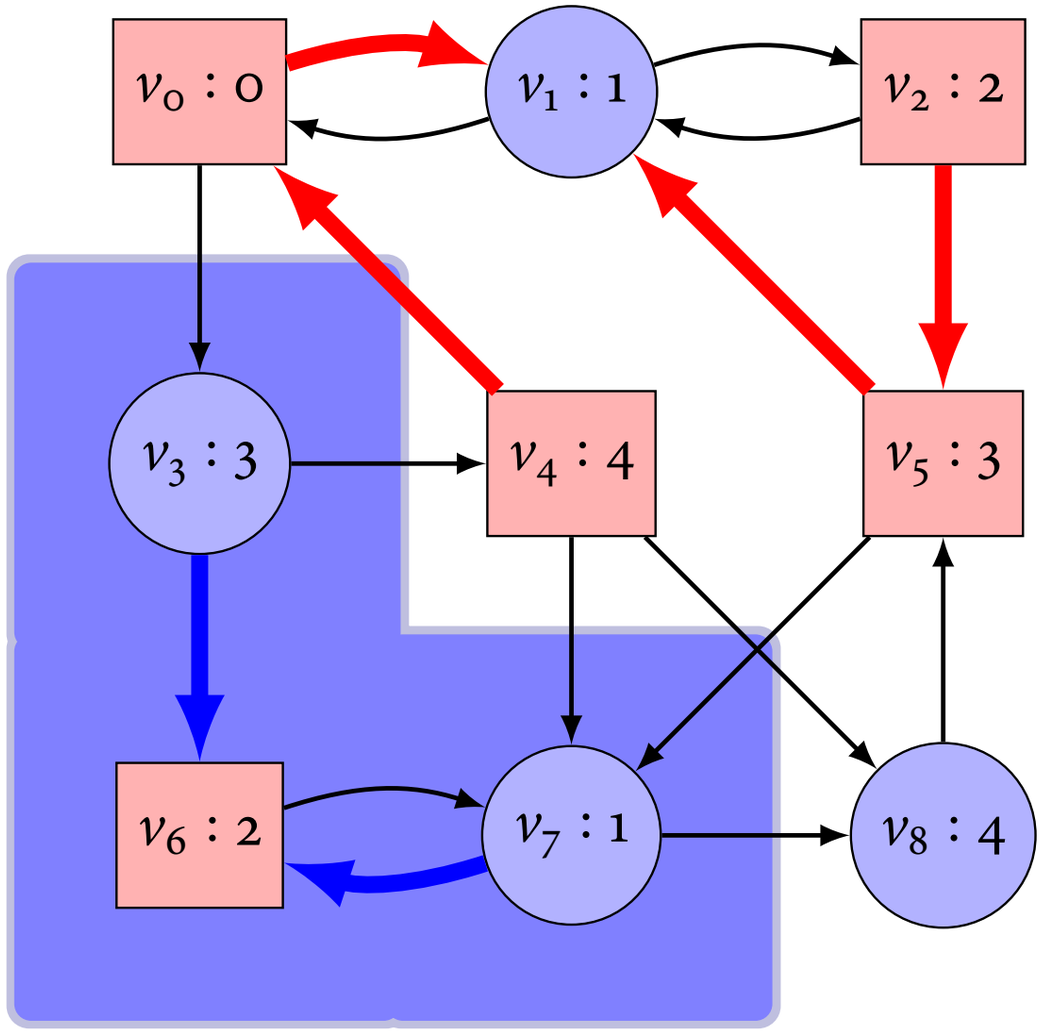
Parity games

Parity game: Player o wins a play π if the highest color that is seen infinitely often is even.



Parity games

Parity game: Player o wins a play π if the highest color that is seen infinitely often is even.



McNaughton's Algorithm: Solving parity games

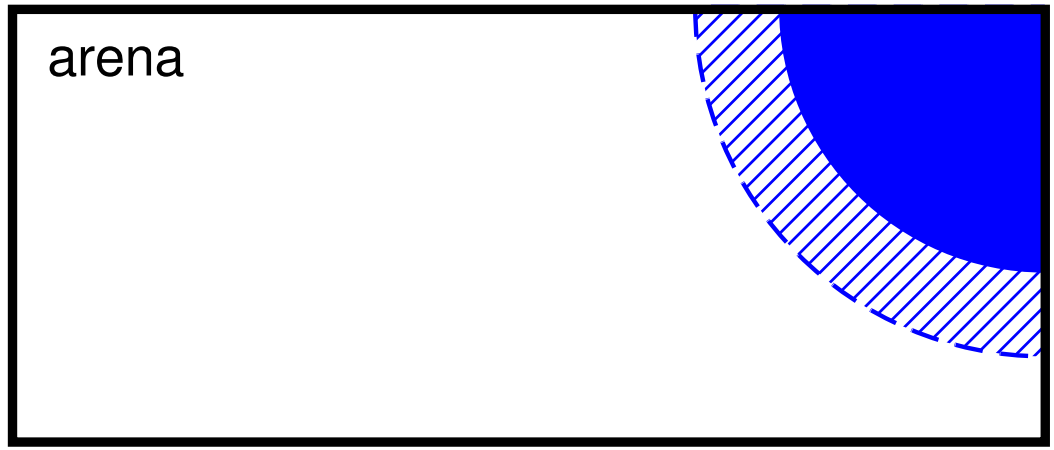
$McNaughton(\mathcal{G})$

1. $c :=$ highest color in $\mathcal{G}$
2. if $c = 0$ or $V = \emptyset$ then return $(V, \emptyset)$
3. set i to $c \mod 2$
4. set W_{1-i} to $\emptyset$
5. repeat
   - 5.1 $\mathcal{G}' := \mathcal{G} \setminus Attr_i(\alpha^{-1}(c), \mathcal{G})$
   - 5.2 $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   - 5.3 if $(W'_{1-i} = \emptyset)$ then
     - 5.3.1 $W_i := V \setminus W_{1-i}$
     - 5.3.2 return (W_0, W_1)
   - 5.4 $W_{1-i} := W_{1-i} \cup Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$
   - 5.5 $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$

McNaughton's Algorithm: Solving parity games

$McNaughton(\mathcal{G})$

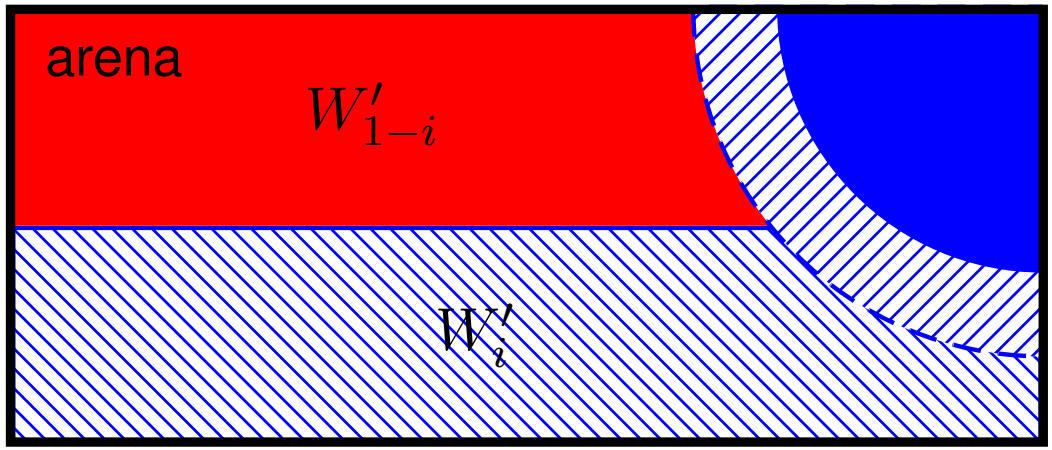
1. $c :=$ highest color in $\mathcal{G}$
2. if $c = 0$ or $V = \emptyset$ then return $(V, \emptyset)$
3. set i to $c \mod 2$
4. set W_{1-i} to $\emptyset$
5. repeat
   - 5.1 $\mathcal{G}' := \mathcal{G} \setminus Attr_i(\alpha^{-1}(c), \mathcal{G})$
   - 5.2 $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   - 5.3 if $(W'_{1-i} = \emptyset)$ then
     - 5.3.1 $W_i := V \setminus W_{1-i}$
     - 5.3.2 return (W_0, W_1)
   - 5.4 $W_{1-i} := W_{1-i} \cup Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$
   - 5.5 $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$



McNaughton's Algorithm: Solving parity games

$McNaughton(\mathcal{G})$

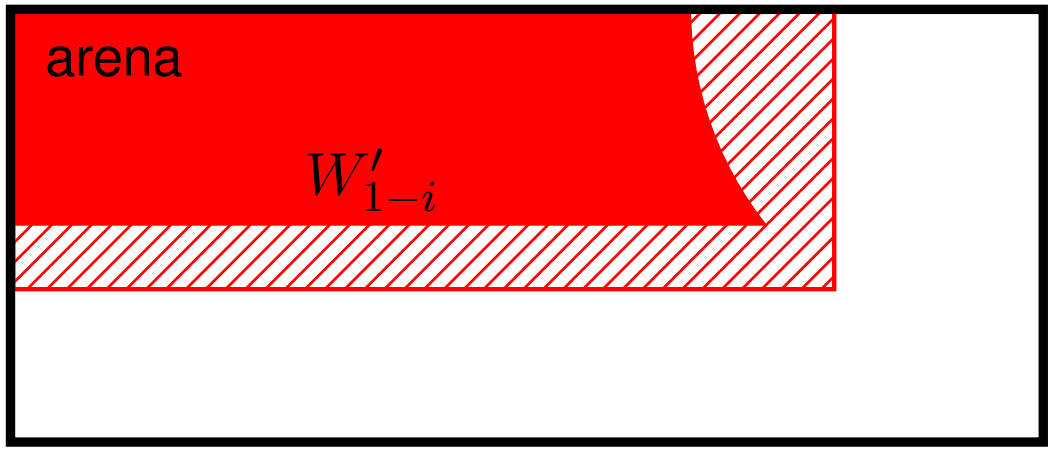
1. $c :=$ highest color in $\mathcal{G}$
2. if $c = 0$ or $V = \emptyset$ then return $(V, \emptyset)$
3. set i to $c \mod 2$
4. set W_{1-i} to $\emptyset$
5. repeat
   - 5.1 $\mathcal{G}' := \mathcal{G} \setminus Attr_i(\alpha^{-1}(c), \mathcal{G})$
   - 5.2 $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   - 5.3 if $(W'_{1-i} = \emptyset)$ then
     - 5.3.1 $W_i := V \setminus W_{1-i}$
     - 5.3.2 return (W_0, W_1)
   - 5.4 $W_{1-i} := W_{1-i} \cup Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$
   - 5.5 $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$



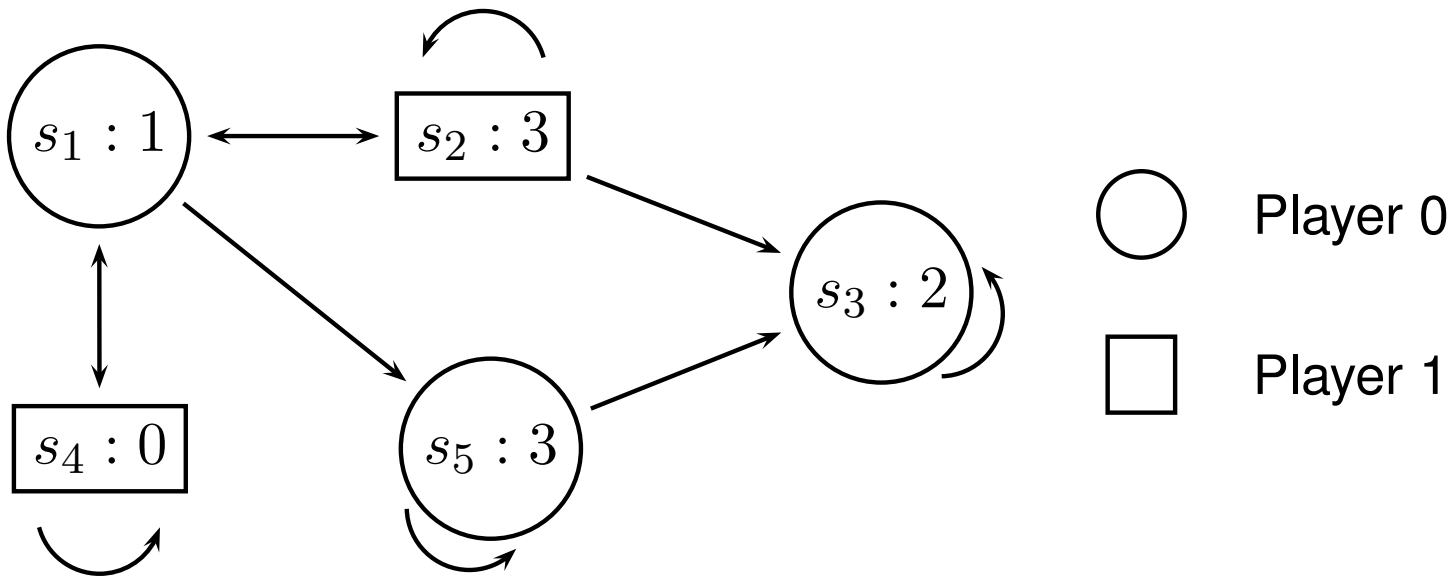
McNaughton's Algorithm: Solving parity games

$McNaughton(\mathcal{G})$

1. $c :=$ highest color in $\mathcal{G}$
2. if $c = 0$ or $V = \emptyset$ then return $(V, \emptyset)$
3. set i to $c \mod 2$
4. set W_{1-i} to $\emptyset$
5. repeat
   - 5.1 $\mathcal{G}' := \mathcal{G} \setminus Attr_i(\alpha^{-1}(c), \mathcal{G})$
   - 5.2 $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   - 5.3 if $(W'_{1-i} = \emptyset)$ then
     - 5.3.1 $W_i := V \setminus W_{1-i}$
     - 5.3.2 return (W_0, W_1)
   - 5.4 $W_{1-i} := W_{1-i} \cup Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$
   - 5.5 $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$



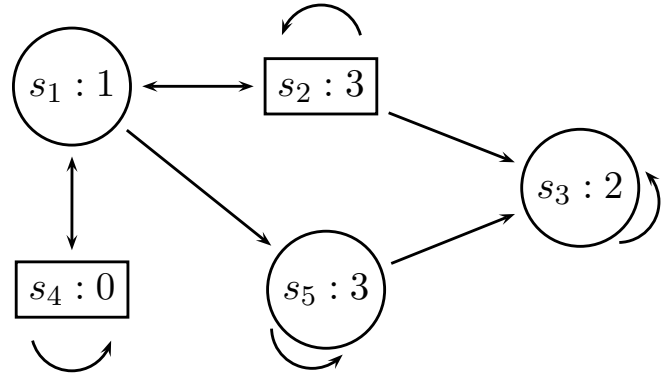
Example



Example

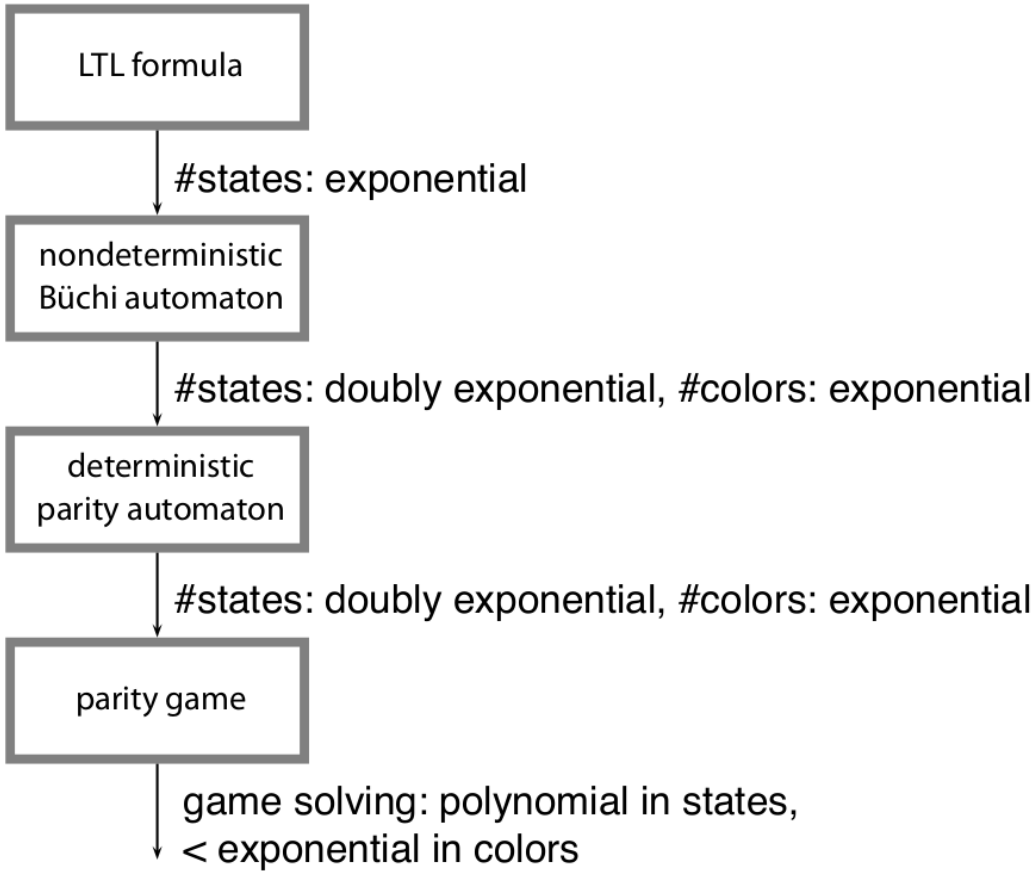
$McNaughton(\mathcal{G})$

1. $c :=$ highest color in $\mathcal{G}$
2. if $c = 0$ or $V = \emptyset$
   then return $(V, \emptyset)$
3. set i to $c \mod 2$
4. set W_{1-i} to $\emptyset$
5. repeat
   - 5.1 $\mathcal{G}' := \mathcal{G} \setminus Attr_i(\alpha^{-1}(c), \mathcal{G})$
   - 5.2 $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   - 5.3 if $(W'_{1-i} = \emptyset)$ then
     - 5.3.1 $W_i := V \setminus W_{1-i}$
     - 5.3.2 return (W_0, W_1)
   - 5.4 $W_{1-i} := W_{1-i} \cup Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$
   - 5.5 $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-i)}(W'_{1-i}, \mathcal{G})$



- $k = 3, c^{-1}(3) = \{s_2, s_5\}, i = 1$
- $\mathcal{G}' := \mathcal{G} \setminus Attr_1(\{s_2, s_5\}, \mathcal{G}) = \{s_1, s_4, s_3\}$
- $(\{s_3\}, \{s_1, s_4\}) = McNaughton(\mathcal{G}')$
- $W_0 := \emptyset \cup Attr_0(\{s_3\}, \mathcal{G}) = \{s_1, s_3, s_5\}$
- $\mathcal{G} := \mathcal{G} \setminus Attr_0(\{s_3\}, \mathcal{G}) = \{s_2, s_4\}$
- $\mathcal{G}' := \mathcal{G} \setminus Attr_1(\{s_2\}, \mathcal{G}) = \{s_4\}$
- $(\{s_4\}, \emptyset) = McNaughton(\mathcal{G}')$
- $W_0 = \{s_1, s_3, s_5\} \cup \{s_4\}$
- $\mathcal{G} := \mathcal{G} \setminus \{s_4\} = \{s_2\}$
- $\mathcal{G}' = \mathcal{G} \setminus Attr_1(\{s_2\}, \mathcal{G}) = \emptyset$
- $(\emptyset, \emptyset) = McNaughton(\mathcal{G}')$
- $W_1 = V \setminus \{s_1, s_3, s_4, s_5\} = \{s_2\}$
- return $(\{s_1, s_3, s_4, s_5\}, \{s_2\})$

LTL synthesis



LTL synthesis

LTL synthesis is 2EXPTIME-complete.

Complexity and practical issues

- The problem is 2EXPTIME-complete
- Determinization is difficult to implement

Alternative approaches bypassing determinization

- Consider efficiently decidable fragments of LTL
  - enforce "deterministic" specification
- Replace determinization by counting
  - search for bounded strategies

GR(1): an efficient fragment of LTL

Generalized Reactivity(1) restricts specifications to specific form

[Piterman et al, 2006]

$$A_1 \wedge A_2 \wedge \ldots \wedge A_m \rightarrow G_1 \wedge G_2 \wedge \ldots \wedge G_n$$

Assumptions $A_1, A_2, \ldots, A_m$, **Guarantees** $G_1, G_2, \ldots, G_n$

Assumptions and guarantees are restricted to formulas of the form:

- **initialization properties:** Boolean combinations of propositions
- **safety properties** of the form $\mathsf{G}(\phi \rightarrow \mathsf{X}\,\psi)$
- **liveness properties** of the form $\mathsf{G}\,\mathsf{F}\,\phi$

Example: $(\mathsf{G}\,\mathsf{F}\,r) \rightarrow (\mathsf{G}\,\mathsf{F}\,g)$

GR(1): an efficient fragment of LTL

GR(1) games have comparatively smaller size:

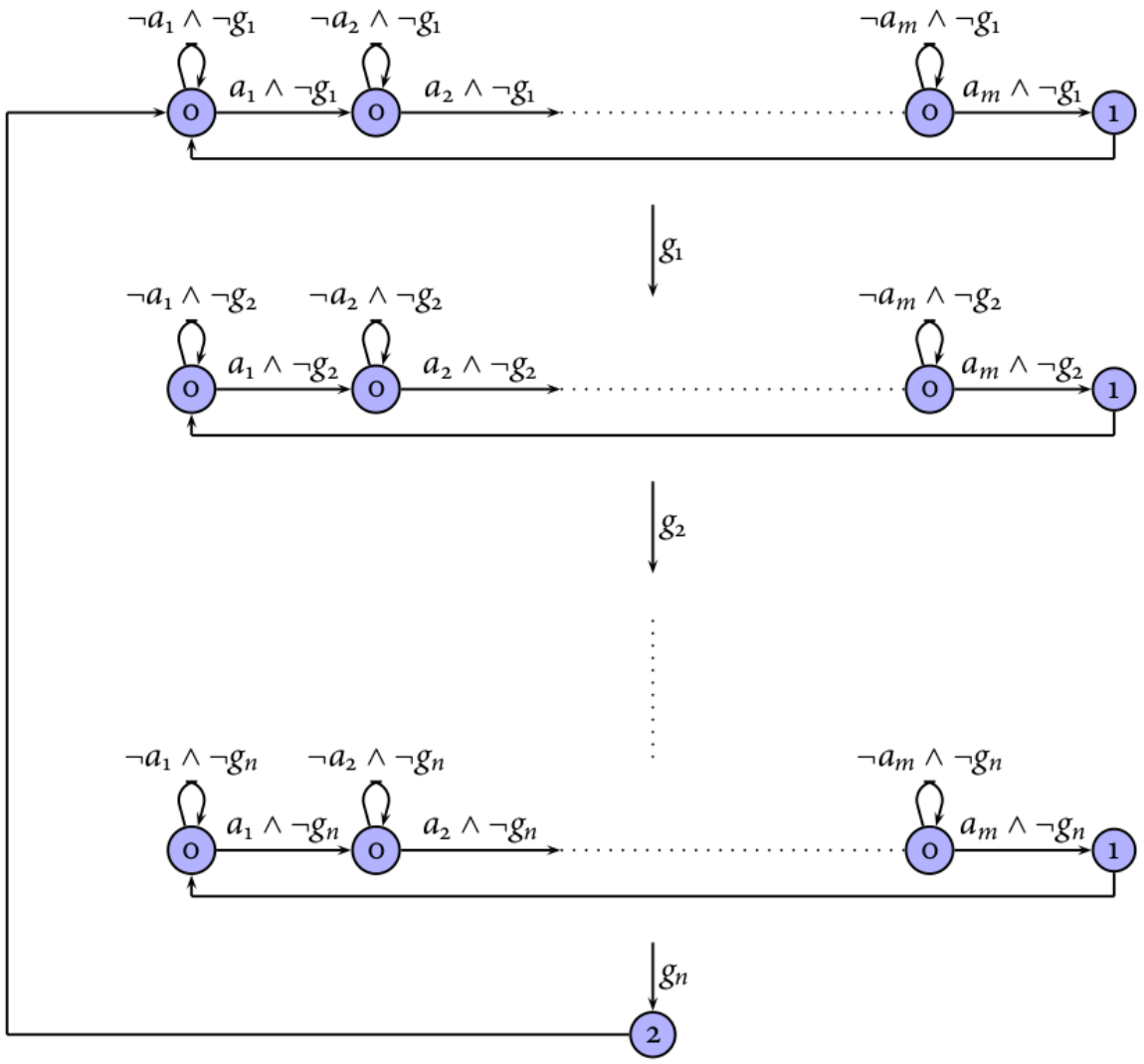
- safety properties $\mathsf{G}(\varphi \mathsf{X} \psi)$ lead to exponential state space
  - keep track of active X formulas
- liveness properties $\mathsf{G}\,\mathsf{F}\,\varphi$ lead to parity games with 3 colors
  - see next slide

GR(1) synthesis [Piterman et al, 2006]

GR(1) synthesis can be done in exponential time.

GR(1): an efficient fragment of LTL

$$\mathsf{G}\,\mathsf{F}\,a_1 \wedge \mathsf{G}\,\mathsf{F}\,a_2 \wedge \ldots \wedge \mathsf{G}\,\mathsf{F}\,a_m \rightarrow \mathsf{G}\,\mathsf{F}\,g_1 \wedge \mathsf{G}\,\mathsf{F}\,g_2 \wedge \ldots \wedge \mathsf{G}\,\mathsf{F}\,g_n$$



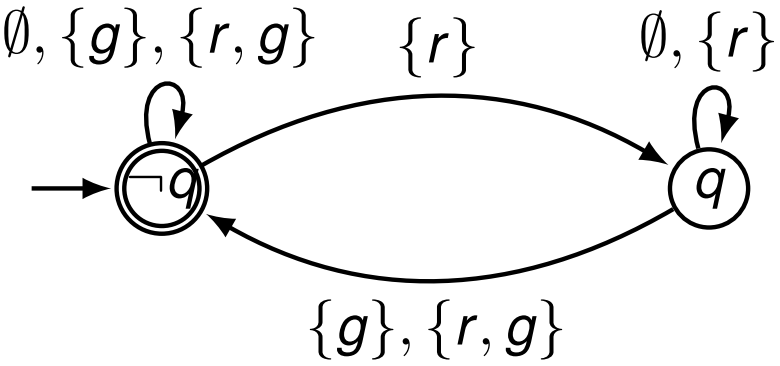
GR(1) synthesis

Specifications often cannot be directly expressed in GR(1).

Pre-synthesis step: synthesize a DBA monitor for (sub-)property and express the property in terms of the states of that monitor. The monitor

- is expressed in GR(1)
- can be doubly exponential

Example: $\mathsf{G}\,(r \rightarrow \mathsf{F}\,g)$ [Recall: not equivalent to $(\mathsf{G}\,\mathsf{F}\,r) \rightarrow (\mathsf{G}\,\mathsf{F}\,g)$]



Introduce s as variable for state

Encoding:

$$
\begin{aligned}
&\neg q \wedge \\
&\mathsf{G}(\neg q \wedge (\neg r \vee g) \rightarrow X \neg q) \wedge \\
&\mathsf{G}(\neg q \wedge (r \vee \neg g) \rightarrow X q) \wedge \\
&\mathsf{G}(q \wedge \neg g \rightarrow X q) \wedge \\
&\mathsf{G}(q \wedge g \rightarrow X \neg q) \wedge \\
&\mathsf{GF}\,\neg q
\end{aligned}
$$

Not all LTL properties can be expressed in GR(1).

Nevertheless, very useful in practical applications.

Bounded Synthesis

Synthesis

Is there an implementation that satisfies the specification?

Bounded synthesis

Is there an implementation with **at most N states**?

Bounded Synthesis

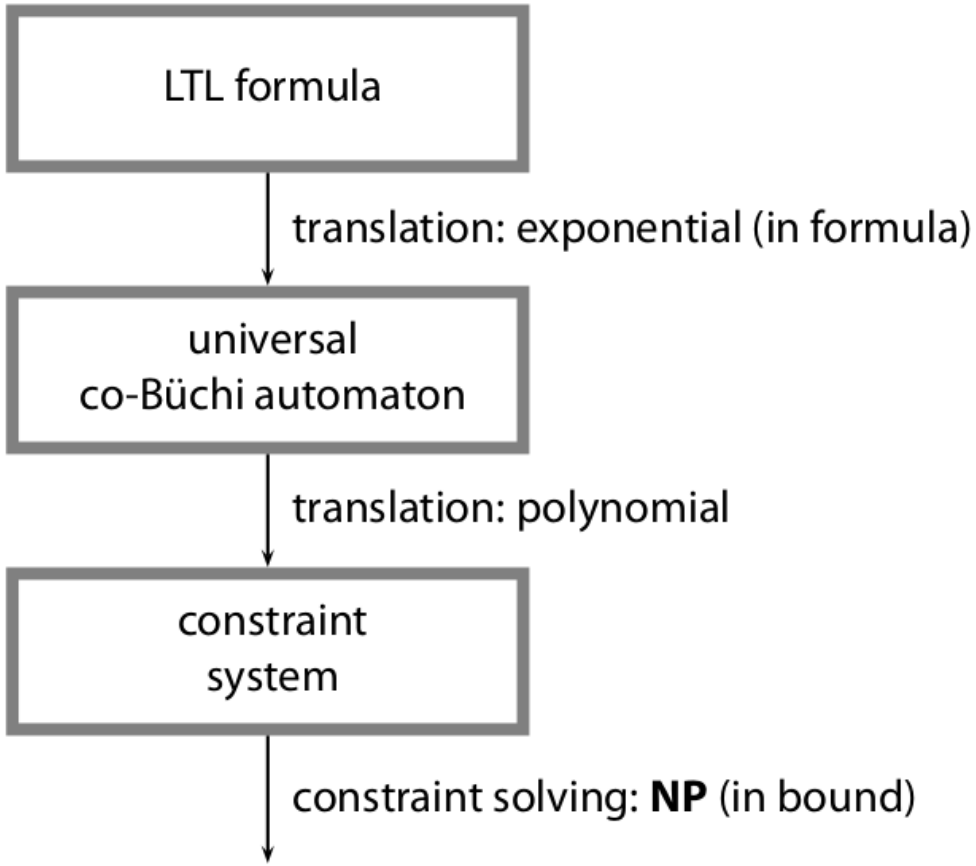
Synthesis

2EXPTIME-complete in length of LTL formula (input).

Bounded synthesis [F.,Schewe 2007]

NP-complete in size of implementation (output).

Bounded Synthesis



Output complexity

LTL synthesis is NP-complete in the size of the implementation.

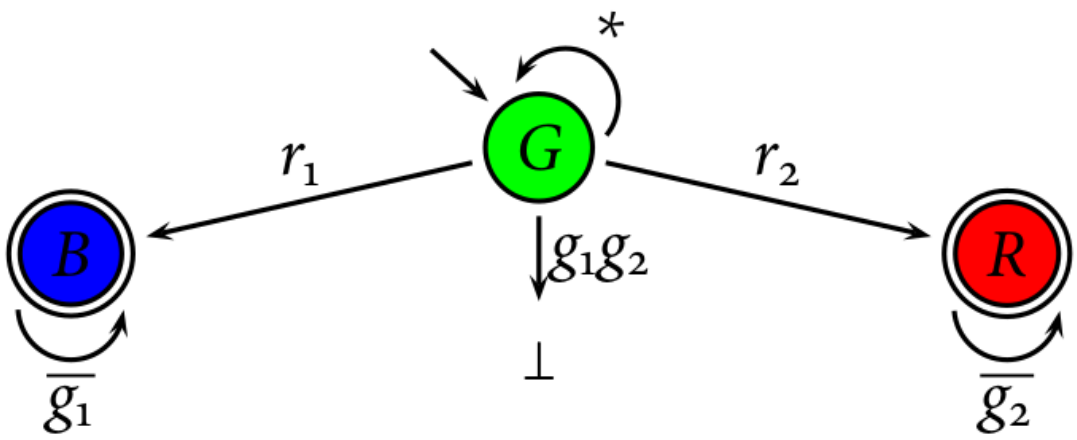
Universal Co-Büchi Automata

co-Büchi condition: a run $q_0 q_1, \ldots$ is accepting if it visits given set F of **rejecting states** only finitely many times ($q_i \in F$ for finitely many i).

A universal co-Büchi automaton accepts a word iff **all runs on the word** satisfy the co-Büchi condition.

Example: Arbiter

$$\varphi = \mathsf{G}(r_1 \rightarrow \mathsf{XF}g_1) \wedge \mathsf{G}(r_2 \rightarrow \mathsf{XF}g_2) \wedge \mathsf{G}\neg(g_1 \wedge g_2)$$



Acceptance of a finite state machine

A finite state machine $\mathcal{M}$ satisfies an LTL formula φ iff every trace is accepted by the universal co-Büchi automaton for φ.

This is the case iff every path in the product satisfies the co-Büchi condition. This holds iff the number of visits to F on each path does not exceed $|F| \cdot |\mathcal{M}|$ (where $|\mathcal{M}|$ is the number of states in $\mathcal{M}$).

The existence of a state machine with N states that is accepted by a universal co-Büchi automaton can be **encoded as a SAT/SMT problem**.

Bounded synthesis as decision procedure

- Increase bound until an implementation or a counterexample strategy (strategy for the environment) is found.
- There exists an implementation or a counterexample strategy of size doubly-exponential in the size of the formula.

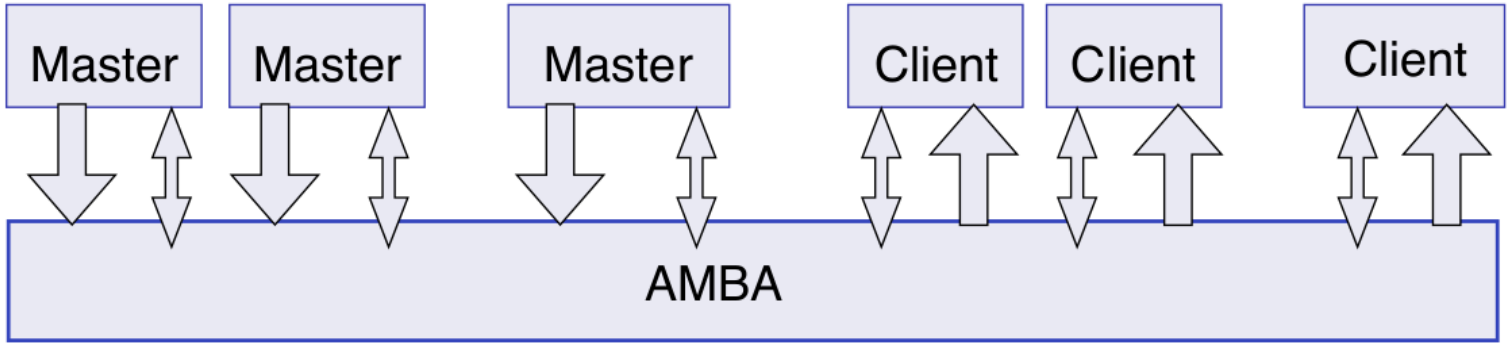
Advantage: Find small implementations fast!

Summary

- The synthesis of reactive systems is a hard problem!
- An active area of research, significant progress in recent years
- Success stories!

Success stories

First industrial-strength application [Bloem et al, 2007]

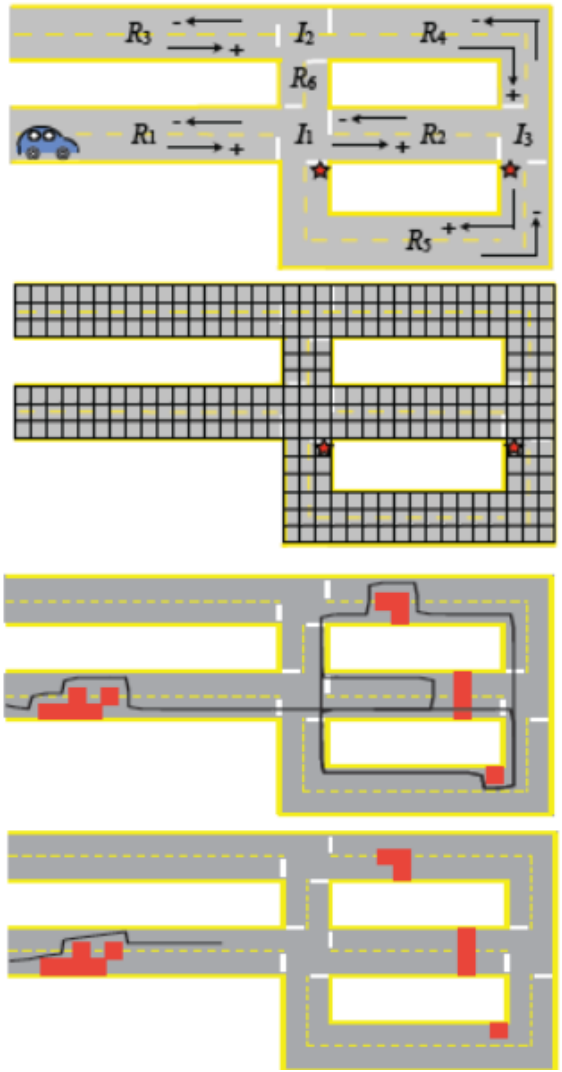


Success stories (continued)

Automatic Synthesis of Robust Embedded Control Software

Tichakorn Wongpiromsarn, Ufuk Topcu and **Richard M. Murray**
California Institute of Technology
Pasadena, California 91125

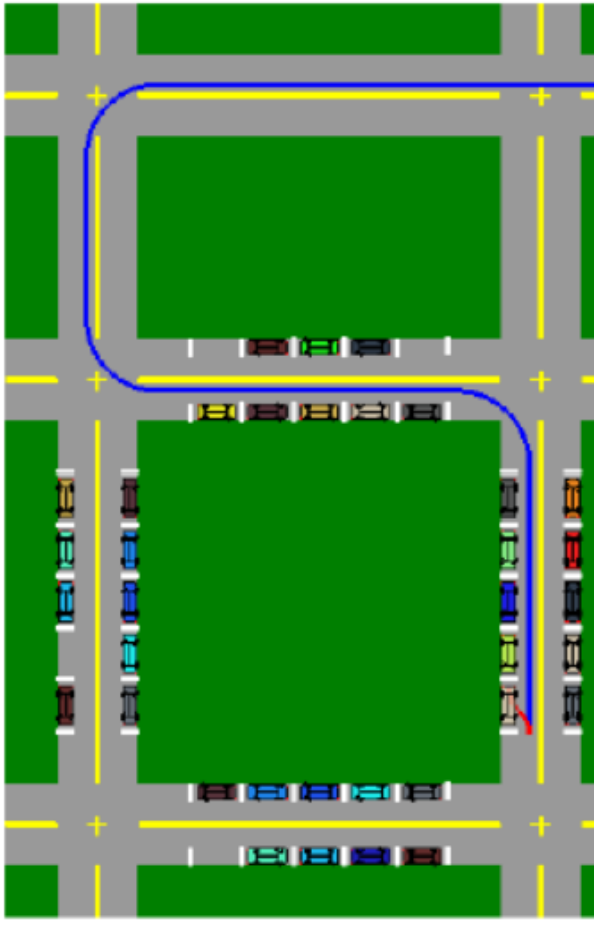
Fig. 1. *Left.* Alice, Team Caltech's entry in the 2007 DARPA Urban Challenge. *Right.* Alice's planner-controller subsystem.



Success stories (continued)

Valet Parking Without a Valet

David C. Conner†, Hadas Kress-Gazit‡, Howie Choset†, Alfred A. Rizzi†, and George J. Pappas‡



Where's Waldo? Sensor-Based Temporal Logic Motion Planning

Hadas Kress-Gazit, Georgios E. Fainekos and George J. Pappas