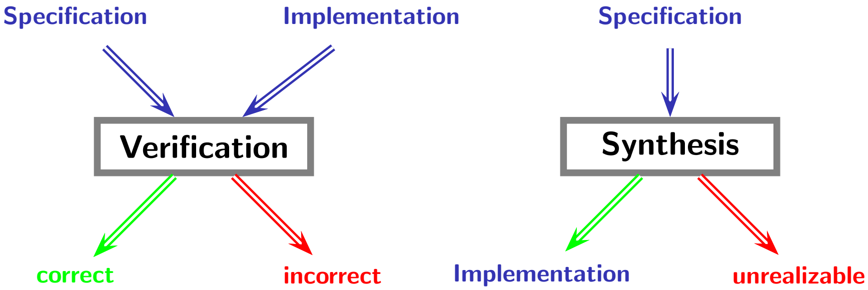
Tutorial: Synthesis

Seminar “Games, Synthesis, and Robotics”

Bernd Finkbeiner
Universität des Saarlandes

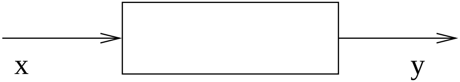
From Verification to Synthesis



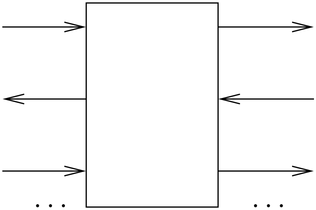
Realizability: Does there exist an implementation?
Synthesis: Construct an implementation (if there is one).

Reactive Systems

- **Transformational Systems**



- **Reactive Systems**



  - **nonterminating**
  - **interaktive** (system vs. environment)

Unrealizable Specifications

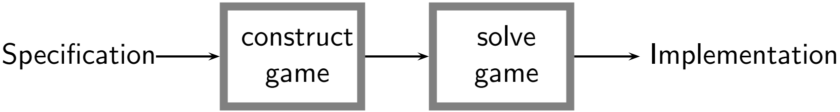
- "If the *start* button is pressed, then the system will immediately start brewing for the next two cycles and, after that, coffee will be produced."
- "If the *power off* button is pressed, brewing stops immediately and permanently."

The specification is unrealizable, because *the environment can produce input* that makes it *impossible* to satisfy both requirements at the same time.

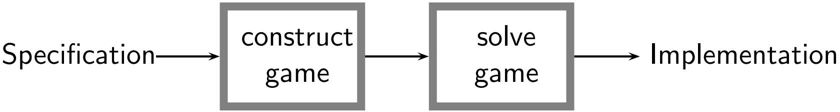
Synthesis as Games

- **Two Players**
  - System vs. Environment
  - Environment chooses inputs
  - System chooses outputs
- **Competing Objectives**
  - System attempts to satisfy specification
  - Environment attempts to violate specification

Synthesis workflow



Synthesis workflow

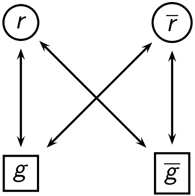


Infinite games
over finite graphs

Infinite games over finite graphs

A **game arena** is a triple $\mathcal{A} = (V_0, V_1, E)$, where

- V_0 and V_1 are disjoint sets of positions, called the positions of player 0 and 1,
- $E \subseteq V \times V$ for set $V = V_0 \uplus V_1$ of game positions,
- every position $p \in V$ has at least one outgoing edge $(p, p') \in E$.



Example: Resource administrator, Player 1 (environment) chooses value of r (request), Player 0 (system) chooses value of g (grant)

Plays and strategies

A **play** is an infinite sequence $\pi = p_0 p_1 p_2 \ldots \in V^\omega$ such that $\forall i \in \omega \,.\, (p_i, p_{i+1}) \in E$.

A **strategy** for player σ is a function $f_\sigma : V^* \cdot V_\sigma \to V$ s.t. $(p, p') \in E$ whenever $f(u \cdot p) = p'$.

A play $\pi = p_0, p_1, \ldots$ **conforms to** strategy f_σ of player σ if $\forall i \in \omega \,.$ if $p_i \in V_\sigma$ then $p_{i+1} = f_\sigma(p_0, \ldots, p_i)$.

Winning conditions

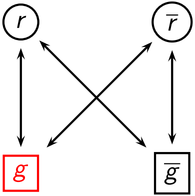
- A **safety/reachability game** $\mathcal{G} = (\mathcal{A}, S)$ consists of a game arena and a safe set of positions $S \subseteq V$. Player 0 wins a play $\pi = p_0 p_1 \ldots$ if $p_i \in S$ for all $i \in \mathbb{N}$, otherwise Player 1 wins.
- A **Büchi/co-Büchi game** $\mathcal{G} = (\mathcal{A}, F)$ consists of an arena $\mathcal{A}$ and a set $F \subseteq V$. Player 0 wins a play π if $In(\pi) \cap F \neq \emptyset$, otherwise Player 1 wins.
- A **parity game** $\mathcal{G} = (\mathcal{A}, \alpha)$ consists of an arena $\mathcal{A}$ and a coloring function $\alpha : V \to \mathbb{N}$. Player 0 wins play π if $\max\{c(q) \mid q \in In(\pi)\}$ is even, otherwise Player 1 wins.

$In(\pi)$: set of positions that occur infinitely often in π.

Winning conditions

- A **safety/reachability game** $\mathcal{G} = (\mathcal{A}, S)$ consists of a game arena and a safe set of positions $S \subseteq V$. Player 0 wins a play $\pi = p_0 p_1 \ldots$ if $p_i \in S$ for all $i \in \mathbb{N}$, otherwise Player 1 wins.

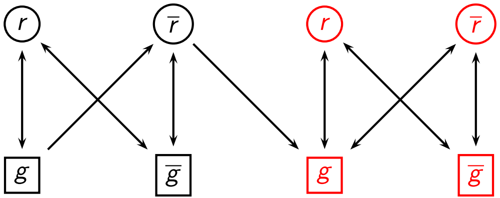
Example: "Never issue a grant."



Winning conditions

- A **safety/reachability game** $\mathcal{G} = (\mathcal{A}, S)$ consists of a game arena and a safe set of positions $S \subseteq V$. Player 0 wins a play $\pi = p_0 p_1 \dots$ if $p_i \in S$ for all $i \in \mathbb{N}$, otherwise Player 1 wins.

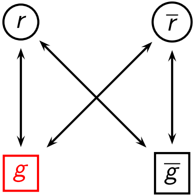
Example: "Only issue a grant when there is a request."



Winning conditions

- A **Büchi/co-Büchi game** $\mathcal{G} = (\mathcal{A}, F)$ consists of an arena $\mathcal{A}$ and a set $F \subseteq V$. Player 0 wins a play π if $In(\pi) \cap F \neq \emptyset$, otherwise Player 1 wins.

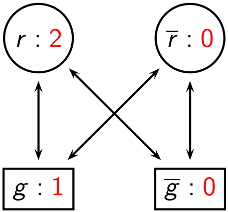
Example: "Issue infinitely many grants."



Winning conditions

- A **parity game** $\mathcal{G} = (\mathcal{A}, \alpha)$ consists of an arena $\mathcal{A}$ and a coloring function $\alpha : S \to \mathbb{N}$. Player 0 wins play π if $\max\{c(q) \mid q \in In(\pi)\}$ is even, otherwise Player 1 wins.

Example: "If there are only finitely many requests, issue only finitely many grants."



Determinacy

A strategy f_σ is ***p*-winning** for player σ and position p if all plays that conform to f_σ and that start in p are won by Player σ.

The **winning region** for player σ is the set of positions

$$W_\sigma = \{p \in V \mid \text{there is a strategy } f_\sigma \text{ s.t. } f_\sigma \text{ is } p\text{-winning}\}.$$

A game is **determined** if $V = W_0 \cup W_1$.

A **memoryless** strategy for player σ is a function $f_\sigma : V_\sigma \to V$ which defines a strategy $f'_\sigma(u \cdot v) = f(v)$.

A game is **memoryless determined** if for every position some player wins the game with memoryless strategy.

Solving Games

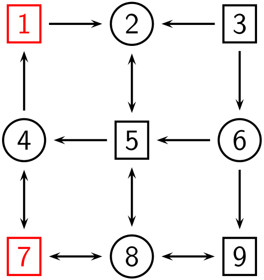
Theorem safety/reachability, Büchi/co-Büchi, and parity games are memoryless determined.

Proof: By fixpoint constructions:
Safety games: $W_1 = Attr_1(V \setminus S)$

Attractor Construction

$$
\begin{aligned}
&Attr^0_\sigma(X, \mathcal{G}) = \emptyset;\\
&Attr^{i+1}_\sigma(X, \mathcal{G}) = Attr^i_\sigma(X)\\
&\qquad \cup\ \{p \in V_\sigma \mid \exists p' \,.\, (p, p') \in E \wedge p' \in Attr^i_\sigma(X, \mathcal{G}) \cup X\}\\
&\qquad \cup\ \{p \in V_{1-\sigma} \mid \forall p' \,.\, (p, p') \in E \Rightarrow p' \in Attr^i_\sigma(X, \mathcal{G}) \cup X\};\\
&Attr^+_\sigma(X, \mathcal{G}) = \bigcup_{i \in \omega} Attr^i_\sigma(X, \mathcal{G}).\\
&Attr_\sigma(X, \mathcal{G}) = Attr^+_\sigma(X, \mathcal{G}) \cup X
\end{aligned}
$$

Example



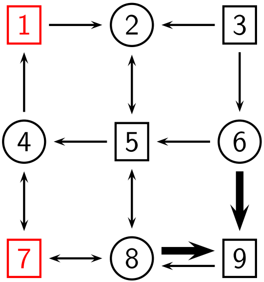
○ = Player 0
□ = Player 1

$S = \{2, 3, 4, 5, 6, 8, 9\}$

$$
\begin{aligned}
Attr_1^0(\{1,7\}, \mathcal{G}) &= \emptyset \\
Attr_1^1(\{1,7\}, \mathcal{G}) &= \{4\} \\
Attr_1^2(\{1,7\}, \mathcal{G}) &= \{4,5,7\} \\
Attr_1^3(\{1,7\}, \mathcal{G}) &= \{2,4,5,7\} \\
Attr_1^4(\{1,7\}, \mathcal{G}) &= \{1,2,3,4,5,7\} \\
Attr_1^+(\{1,7\}, \mathcal{G}) &= \{1,2,3,4,5,7\} \\
Attr_1(\{1,7\}, \mathcal{G}) &= \{1,2,3,4,5,7\}
\end{aligned}
$$

$$
\begin{aligned}
W_1 &= \{1,2,3,4,5,7\} \\
W_0 &= \{6,8,9\}
\end{aligned}
$$

Example



$\bigcirc$ = Player 0
$\square$ = Player 1

$S = \{2, 3, 4, 5, 6, 8, 9\}$

$$
\begin{aligned}
Attr_1^0(\{1,7\}, \mathcal{G}) &= \emptyset \\
Attr_1^1(\{1,7\}, \mathcal{G}) &= \{4\} \\
Attr_1^2(\{1,7\}, \mathcal{G}) &= \{4,5,7\} \\
Attr_1^3(\{1,7\}, \mathcal{G}) &= \{2,4,5,7\} \\
Attr_1^4(\{1,7\}, \mathcal{G}) &= \{1,2,3,4,5,7\} \\
Attr_1^+(\{1,7\}, \mathcal{G}) &= \{1,2,3,4,5,7\} \\
Attr_1(\{1,7\}, \mathcal{G}) &= \{1,2,3,4,5,7\}
\end{aligned}
$$

$$
\begin{aligned}
W_1 &= \{1,2,3,4,5,7\} \\
W_0 &= \{6,8,9\}
\end{aligned}
$$

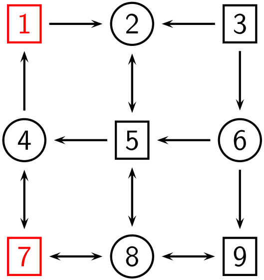
Solving Büchi games

$$W_0 = Attr_0(Recur_0(\mathcal{G}), \mathcal{G})$$

Recurrence Construction:

$$Recur_\sigma^0(\mathcal{G}) = F;$$
$$Recur_\sigma^{i+1}(\mathcal{G}) = F \cap Attr_\sigma^+(Recur_\sigma^i, \mathcal{G});$$
$$Recur_\sigma(\mathcal{G}) = \bigcap_{i \in \mathbb{N}} Recur_\sigma^i(\mathcal{G}).$$

Example



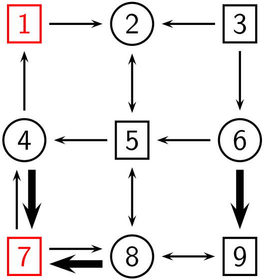
$\bigcirc$ = Player 0
$\square$ = Player 1

$F = \{1, 7\}$

$$Recur_0^0(\mathcal{G}) = \{1, 7\}$$
$$Attr_0^+(\{1, 7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\}$$
$$Recur_0^1(\mathcal{G}) = \{7\}$$
$$Attr_0^+(\{7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\}$$
$$Recur_0(\mathcal{G}) = \{7\}$$
$$Attr_0(\{7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\}$$

$$W_0 = \{4, 6, 7, 8, 9\}$$
$$W_1 = \{1, 2, 3, 5\}$$

Example



$\bigcirc$ = Player 0
$\square$ = Player 1

$F = \{1, 7\}$

$Recur_0^0(\mathcal{G}) = \{1, 7\}$
$Attr_0^+(\{1, 7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\}$
$Recur_0^1(\mathcal{G}) = \{7\}$
$Attr_0^+(\{7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\}$
$Recur_0(\mathcal{G}) = \{7\}$
$Attr_0(\{7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\}$

$W_0 = \{4, 6, 7, 8, 9\}$
$W_1 = \{1, 2, 3, 5\}$

McNaughton's Algorithm: Solving parity games

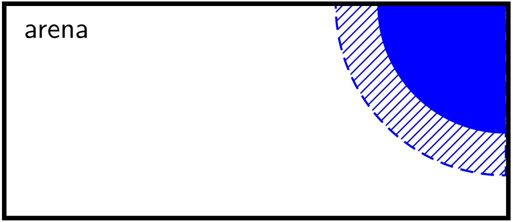
McNaughton($\mathcal{G}$)

1. $c :=$ highest color in $\mathcal{G}$
2. if $c = 0$ or $V = \emptyset$ then return $(V, \emptyset)$
3. set σ to $c \mod 2$
4. set $W_{1-\sigma}$ to $\emptyset$
5. repeat
    1. $\mathcal{G}' := \mathcal{G} \setminus Attr_\sigma(\alpha^{-1}(c), \mathcal{G})$
    2. $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
    3. if $(W'_{1-\sigma} = \emptyset)$ then
        1. $W_\sigma := V \setminus W_{1-\sigma}$
        2. return (W_0, W_1)
    4. $W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$
    5. $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$

McNaughton's Algorithm: Solving parity games

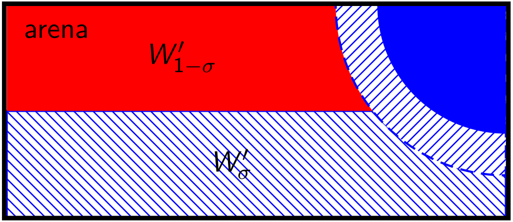
McNaughton$(\mathcal{G})$

1. $c :=$ highest color in $\mathcal{G}$
2. <u>if</u> $c = 0$ or $V = \emptyset$
   <u>then</u> <u>return</u> $(V, \emptyset)$
3. set σ to $c \mod 2$
4. set $W_{1-\sigma}$ to $\emptyset$
5. <u>repeat</u>
   1. $\mathcal{G}' := \mathcal{G} \setminus Attr_\sigma(\alpha^{-1}(c), \mathcal{G})$
   2. $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   3. <u>if</u> $(W'_{1-\sigma} = \emptyset)$ <u>then</u>
      1. $W_\sigma := V \setminus W_{1-\sigma}$
      2. <u>return</u> (W_0, W_1)
   4. $W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$
   5. $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$



McNaughton's Algorithm: Solving parity games

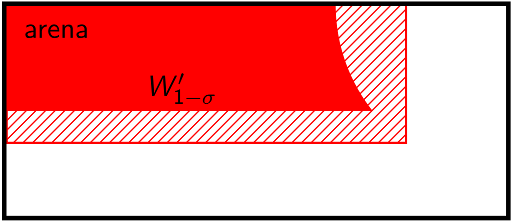
McNaughton$(\mathcal{G})$



1. $c :=$ highest color in $\mathcal{G}$
2. <u>if</u> $c = 0$ or $V = \emptyset$
   <u>then</u> <u>return</u> $(V, \emptyset)$
3. set σ to $c \mod 2$
4. set $W_{1-\sigma}$ to $\emptyset$
5. <u>repeat</u>
   1. $\mathcal{G}' := \mathcal{G} \setminus Attr_\sigma(\alpha^{-1}(c), \mathcal{G})$
   2. $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   3. <u>if</u> $(W'_{1-\sigma} = \emptyset)$ <u>then</u>
      1. $W_\sigma := V \setminus W_{1-\sigma}$
      2. <u>return</u> (W_0, W_1)
   4. $W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$
   5. $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$

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 σ to $c \mod 2$
4. set $W_{1-\sigma}$ to $\emptyset$
5. repeat
   1. $\mathcal{G}' := \mathcal{G} \setminus Attr_\sigma(\alpha^{-1}(c), \mathcal{G})$
   2. $(W'_0, W'_1) := McNaughton(\mathcal{G}')$
   3. if $(W'_{1-\sigma} = \emptyset)$ then
      1. $W_\sigma := V \setminus W_{1-\sigma}$
      2. return (W_0, W_1)
   4. $W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$
   5. $\mathcal{G} := \mathcal{G} \setminus Attr_{(1-\sigma)}(W'_{1-\sigma}, \mathcal{G})$

Synthesis workflow

Specification $\longrightarrow$ construct game $\longrightarrow$ solve game $\longrightarrow$ Implementation

$\uparrow$

LTL

in the seminar also:

GR(1), CTL,
or game directly given

Linear-Time Temporal Logic (LTL)

Syntax:

- Let AP be a set of atomic propositions.
- Every atomic proposition $p \in AP$ is an LTL formula
- If φ and ψ are LTL formulas, then so are
  - $\neg\varphi,\ \varphi \wedge \phi,$
  - $\bigcirc\,\varphi,\ \varphi\,\mathcal{U}\,\psi$

Abbreviations:

$\Diamond\,\varphi \;\equiv\; \mathit{true}\,\mathcal{U}\,\varphi;$
$\Box\,\varphi \;\equiv\; \neg(\Diamond\,\neg\varphi);$
$\varphi\,\mathcal{W}\,\psi \;\equiv\; (\varphi\,\mathcal{U}\,\psi) \vee \Box\,\varphi;$

Semantics

For an infinite sequence $\alpha \in (2^{AP})^\omega$:

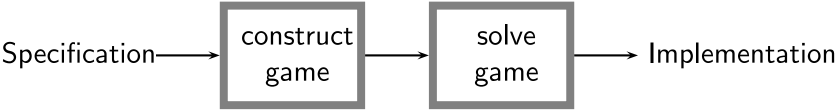
- $\alpha, i \models p$ iff $p \in \alpha(i)$;
- $\alpha, i \models \neg\varphi$ iff $\alpha, i \not\models \varphi$;
  $\alpha, i \models \varphi \wedge \psi$ iff $\alpha, i \models \varphi$ and $\alpha, i \models \psi$;
- $\alpha, i \models \bigcirc \varphi$ iff $\alpha, i+1 \models \varphi$
- $\alpha, i \models \varphi \mathcal{U} \psi$ iff there is some $j \geq i$ s.t.
  $\alpha, j \models \psi$ and for all $i \leq k < j$: $\alpha, k \models \varphi$

- $\alpha \models \varphi$ iff $\alpha, 0 \models \varphi$

Examples

- Invariant: $\Box\, p$
- Guarantee: $\Diamond\, p$
- Recurrence: $\Box \Diamond\, p$
- Request-Response: $\Box(p \rightarrow \Diamond\, q)$
- Fairness: $(\Box \Diamond\, p) \rightarrow (\Box \Diamond\, q)$

Synthesis workflow



$\uparrow$

LTL

$\downarrow$

NBA

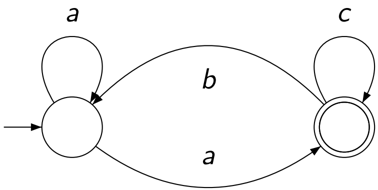
$\downarrow$

DPA

$\downarrow$

parity game

Büchi automata



A **NBA** (nondeterministic Büchi automaton) $\mathcal{A} = (\Sigma, S, I, T, F)$ consists of the following:

- Σ: alphabet
- S: finite set of states
- $I \subseteq S$: initial states
- $T \subseteq S \times \Sigma \times S$: transitions
- $F \subseteq S$: accepting states

Accepting runs

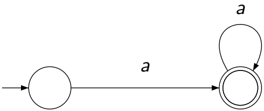
- A **run** of an NBA $\mathcal{A} = (\Sigma, S, I, T, F)$ on an **infinite** word $\sigma_0\sigma_1 \ldots \in \Sigma^\omega$ is an infinite sequence of states $q_0\, q_1 \ldots \in S^\omega$, such that the following holds:
  - $q_0 \in I$ and
  - $(q_i, \sigma_i, q_{i+1}) \in T$ for all $i \geq 0$.
- A run $q_0\, q_1\, q_2 \ldots$ is **accepting** iff $q_n \in F$ **for infinitely many** n.
- A word w is **accepted by** $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on w.
- The **language of** $\mathcal{A}$:

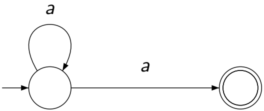
$$\mathcal{L}_\omega(\mathcal{A}) = \left\{\sigma \in \Sigma^\omega \mid \sigma \text{ is accepted by } \mathcal{A} \right\}$$

  $\mathcal{A}$ **recognizes** $\mathcal{L}_\omega(\mathcal{A})$.
- Two NBAs $\mathcal{A}$ and $\mathcal{A}'$ are **equivalent** iff $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}')$.

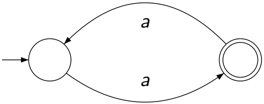
NBA vs. NFA

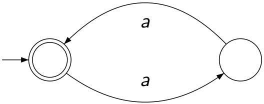
- finite equivalence $\not\Rightarrow$ ω-equivalence





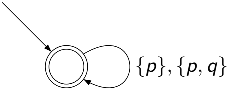
- ω-equivalence $\not\Rightarrow$ finite equivalence

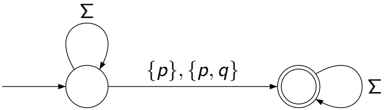




NFA: nondeterministic finite-word automaton

LTL vs. NBA

$\Box\, p$: 

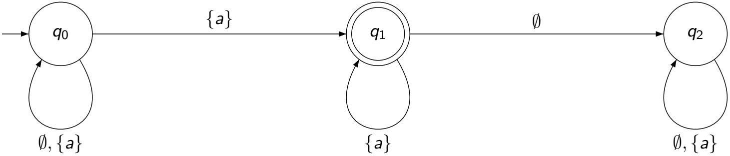
$\Diamond\, p$: 

- models(φ)=$\{\alpha \in (2^{AP})^w \mid \alpha \models \varphi\}$
- For every LTL formula φ there is an NBA $\mathcal{A}_\varphi$ over $\Sigma = 2^{AP}$ that recognizes models(φ).
- The size of $\mathcal{A}_\varphi$ is exponential in the length of φ.
- There are NBA-recognizable languages that cannot be defined as an LTL formula.
  Example: $(\emptyset\emptyset)^*\{p\}^\omega$

Deterministic Büchi automata (DBA)

- A Büchi automaton $\mathcal{A}$ is **deterministic (DBA)** iff
  $|I| \leq 1$ and
  $|\{q' \in S \mid (q, \sigma, q') \in T\}| \leq 1$ for all $q \in S$ und $\sigma \in \Sigma$

- NBAs are strictly more expressive than DBAs.
  There is no DBA for $\Diamond \Box a$



Parity automata

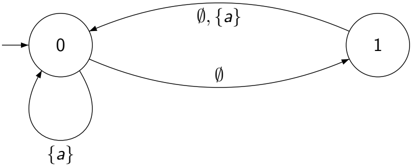
A **NPA** (nondeterministic parity automaton) $\mathcal{A} = (\Sigma, S, I, T, \alpha)$ consists of the following:

- Σ: alphabet
- S: finite set of states
- $I \subseteq S$: initial states
- $T \subseteq S \times \Sigma \times S$: transitions
- $\alpha : V \to \mathbb{N}$ coloring function

A run π of a parity automaton is **accepting** iff $\max\{c(q) \mid q \in In(\pi)\}$ is even.

From NBA to DPA

- **DPA:** Deterministic parity automaton
- For every NBA there exists an equivalent DPA
- The number of states of the DPA is exponential in the number of states of the NBA.



From LTL to DPA

- **Corollary:** For every LTL formula φ there exists a DPA $\mathcal{P}_\varphi$ such that $\mathcal{L}(\mathcal{P}_\varphi) = \text{models}(\varphi)$.
- The number of states of $\mathcal{P}_\varphi$ is doubly-exponential in the length of φ.

Example:

$\mathcal{L}_n = \{\{0,1,\#\}^* \cdot \# \cdot w \cdot \{0,1,\#\}^* \cdot \$ \cdot w \mid w \in \{0,1\}^n\}$

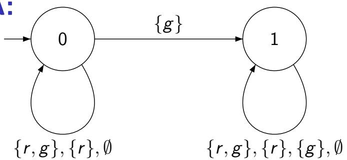
- Smallest deterministic automaton recognizing $\mathcal{L}_n$ has 2^{2^n} states.
- $\mathcal{L}_n$ can be defined with small (quadratic) LTL formula:

$$[(\neg\$ \,\mathcal{U}\, \$ \wedge \bigcirc \Box \neg\$)] \wedge$$
$$\Diamond[\# \wedge \bigwedge_{1 \le i \le n}((\bigcirc^i 0 \wedge \Box(\$ \rightarrow \bigcirc^i 0)) \vee (\bigcirc^i 1 \wedge \Box(\$ \rightarrow \bigcirc^i 1)))]$$

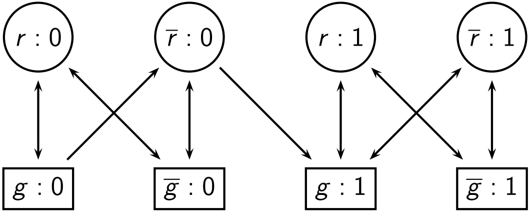
Example

"Only issue a grant when there is a request."

- **LTL:** $\square(\neg r \to \neg g)$
- **DPA:**



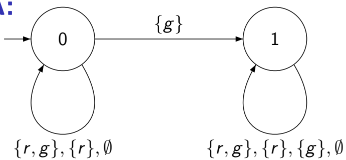
- **Parity game:**



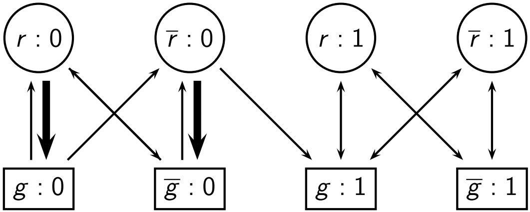
Example

"Only issue a grant when there is a request."

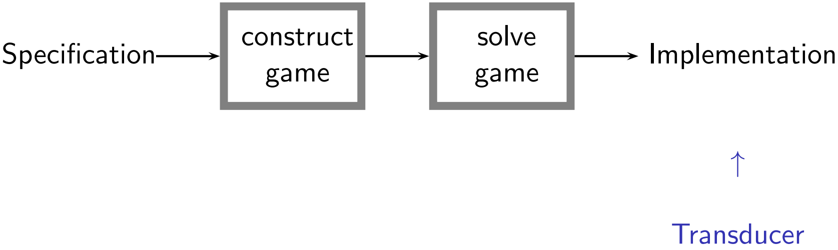
- **LTL:** $\Box(\neg r \to \neg g)$
- **DPA:**



- **Parity game:**



Synthesis workflow



Transducer

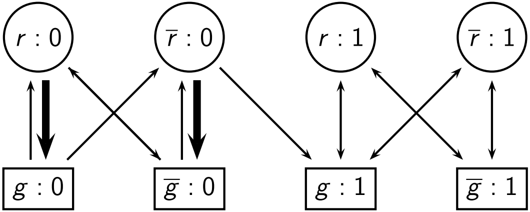
A **transducer** (Mealy machine) $\mathcal{A} = (\Sigma, \Delta, S, i, T, \delta)$ consists of the following:

- Σ: input alphabet
- Δ: output alphabet
- S: finite set of states
- $i \in S$: initial state
- $T : S \times \Sigma \to S$: transition function
- $\delta : S \times \Sigma \to \Delta$: output function

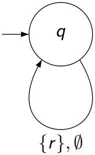
The winning strategy can be represented as a transducer.

Example

- **Parity game:**

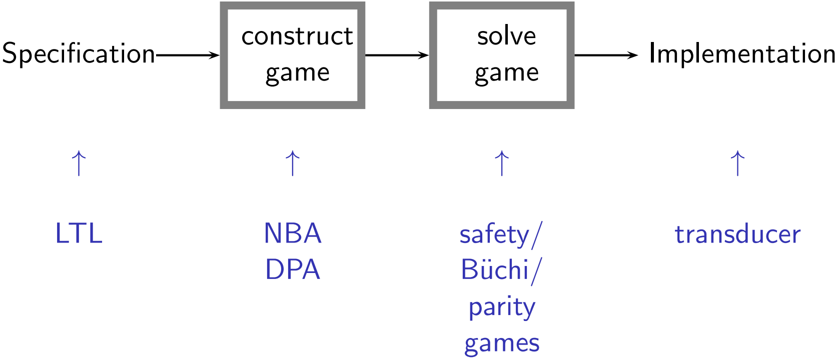


- **Transducer:**



$\delta(q,r)=g$
$\delta(q,\overline{r})=\overline{g}$

Synthesis workflow



Major extensions in the seminar

- GR(1) — an efficient fragment of LTL
- timed games — games with real time
- CTL — from linear time to branching time
- distribution — incomplete information
- robotics!