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**

  ![Diagram](image)

  \( x \rightarrow \) System \( \rightarrow \) \( y \)

- **Reactive Systems**

  ![Diagram](image)

  \( \ldots \rightarrow \) System \( \rightarrow \) \( \ldots \)

  - 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

Specification → construct game → solve game → Implementation

Infinite games over finite graphs
Synthesis workflow

Specification → construct game → solve game → Implementation

Infinite games over finite graphs
Infinite games over finite graphs

A **game arena** is a triple \( A = (V_0, V_1, E) \), 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 \( V = V_0 \cup V_1 \) 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)
A **play** is an infinite sequence \( \pi = p_0p_1p_2\ldots \in V^{\omega} \) such that \( \forall i \in \omega . \ (p_i, p_{i+1}) \in E \).

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

A play \( \pi = p_0, p_1, \ldots \) **conforms to** strategy \( f_{\sigma} \) of player \( \sigma \) 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** $G = (\mathcal{A}, S)$ consists of a game arena and a safe set of positions $S \subseteq V$. Player 0 wins a play $\pi = p_0p_1 \ldots$ if $p_i \in S$ for all $i \in \mathbb{N}$, otherwise Player 1 wins.

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

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

$\text{ln}(\pi)$: set of positions that occur infinitely often in $\pi$. 


Winning conditions

- A *safety/reachability game* $G = (A, S)$ consists of a game arena and a safe set of positions $S \subseteq V$. Player 0 wins a play $\pi = p_0p_1\ldots$ 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 $G = (A, S)$ 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: “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 $\pi$ if $\text{ln}(\pi) \cap F \neq \emptyset$, otherwise Player 1 wins.

**Example:** “Issue infinitely many grants.”
Winning conditions

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

Example: “If there are only finitely many requests, issue only finitely many grants.”
Determinacy

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

The **winning region** for player $\sigma$ 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 $\sigma$ 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.
**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**

\[
\begin{align*}
Attr^0_\sigma(X, G) &= \emptyset; \\
Attr^{i+1}_\sigma(X, G) &= Attr^i_\sigma(X) \\
&\quad \cup \left\{ p \in V_\sigma \mid \exists p' . (p, p') \in E \land p' \in Attr^i_\sigma(X, G) \cup X \right\} \\
&\quad \cup \left\{ p \in V_{1-\sigma} \mid \forall p' . (p, p') \in E \Rightarrow p' \in Attr^i_\sigma(X, G) \cup X \right\}; \\
Attr^+_\sigma(X, G) &= \bigcup_{i \in \omega} Attr^i_\sigma(X, G). \\
Attr_\sigma(X, G) &= Attr^+_\sigma(X, G) \cup X
\end{align*}
\]
Example

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

$\text{Attr}_1(\{1, 7\}, G) = \emptyset$

$\text{Attr}_1(\{1, 7\}, G) = \{4\}$

$\text{Attr}_1(\{1, 7\}, G) = \{4, 5, 7\}$

$\text{Attr}_1(\{1, 7\}, G) = \{2, 4, 5, 7\}$

$\text{Attr}_1(\{1, 7\}, G) = \{1, 2, 3, 4, 5, 7\}$

$\text{Attr}_1(\{1, 7\}, G) = \{1, 2, 3, 4, 5, 7\}$

$W_0 = \{6, 8, 9\}$

$W_1 = \{1, 2, 3, 4, 5, 7\}$
Example

\[
\begin{align*}
\text{Attr}_0^0(\{1, 7\}, G) &= \emptyset \\
\text{Attr}_1^0(\{1, 7\}, G) &= \emptyset \\
\text{Attr}_2^0(\{1, 7\}, G) &= \{4\} \\
\text{Attr}_3^0(\{1, 7\}, G) &= \{4, 5, 7\} \\
\text{Attr}_4^0(\{1, 7\}, G) &= \{2, 4, 5, 7\} \\
\text{Attr}_5^0(\{1, 7\}, G) &= \{1, 2, 3, 4, 5, 7\} \\
\text{Attr}_6^0(\{1, 7\}, G) &= \{1, 2, 3, 4, 5, 7\}
\end{align*}
\]

\[
W_1 = \{1, 2, 3, 4, 5, 7\} \\
W_0 = \{6, 8, 9\}
\]
Solving Büchi games

\[ W_0 = Attr_0(\text{Recur}_0(\mathcal{G}), \mathcal{G}) \]

**Recurrence Construction:**

\[
\begin{align*}
\text{Recur}_0^0(\mathcal{G}) &= F; \\
\text{Recur}_0^{i+1}(\mathcal{G}) &= F \cap Attr_0^+(\text{Recur}_0^i, \mathcal{G}); \\
\text{Recur}_\sigma(\mathcal{G}) &= \bigcap_{i \in \mathbb{N}} \text{Recur}_\sigma^i(\mathcal{G}).
\end{align*}
\]
Example

\[ \text{Recur}_0^0(G) = \{1, 7\} \]
\[ \text{Attr}_0^+({\{1, 7\}, G}) = \{4, 6, 7, 8, 9\} \]
\[ \text{Recur}_0^1(G) = \{7\} \]
\[ \text{Attr}_0^+({\{7\}, G}) = \{4, 6, 7, 8, 9\} \]
\[ \text{Recur}_0(G) = \{7\} \]
\[ \text{Attr}_0({\{7\}, G}) = \{4, 6, 7, 8, 9\} \]

\[ \text{W}_0 = \{4, 6, 7, 8, 9\} \]
\[ \text{W}_1 = \{1, 2, 3, 5\} \]
Example

\[ \text{Recur}_0^0(\mathcal{G}) = \{1, 7\} \]
\[ \text{Attr}_0^+(\{1, 7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\} \]
\[ \text{Recur}_0^1(\mathcal{G}) = \{7\} \]
\[ \text{Attr}_0^+(\{7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\} \]
\[ \text{Recur}_0(\mathcal{G}) = \{7\} \]
\[ \text{Attr}_0(\{7\}, \mathcal{G}) = \{4, 6, 7, 8, 9\} \]

\[ \mathcal{W}_0 = \{4, 6, 7, 8, 9\} \]
\[ \mathcal{W}_1 = \{1, 2, 3, 5\} \]
McNaughton’s Algorithm: Solving parity games

\textit{McNaughton}(\mathcal{G})

1. $c := \text{highest color in } \mathcal{G}$
2. \textbf{if} $c = 0$ or $V = \emptyset$
   \textbf{then} return $(V, \emptyset)$
3. set $\sigma$ to $c \mod 2$
4. set $W_{1-\sigma}$ to $\emptyset$
5. repeat
   1. $\mathcal{G}' := \mathcal{G} \setminus \text{Attr}_{\sigma}(\alpha^{-1}(c), \mathcal{G})$
   2. $(W_0', W_1') := \text{McNaughton}(\mathcal{G}')$
   3. \textbf{if} $(W_1'_{1-\sigma} = \emptyset)$ \textbf{then}
      1. $W_{\sigma} := V \setminus W_{1-\sigma}$
      2. return $(W_0, W_1)$
   4. $W_{1-\sigma} := W_{1-\sigma} \cup \text{Attr}_{(1-\sigma)}(W_1', \mathcal{G})$
   5. $\mathcal{G} := \mathcal{G} \setminus \text{Attr}_{(1-\sigma)}(W_1', \mathcal{G})$
McNaughton’s Algorithm: Solving parity games

McNaughton(\mathcal{G})

1. \( c := \text{highest color in } \mathcal{G} \)
2. \( \text{if } c = 0 \text{ or } V = \emptyset \)
   \hspace{1cm} then return \((V, \emptyset)\)
3. set \( \sigma \) to \( c \mod 2 \)
4. set \( W_{1-\sigma} \) to \( \emptyset \)
5. repeat

   1. \( \mathcal{G}' := \mathcal{G} \setminus \text{Attr}_\sigma(\alpha^{-1}(c), \mathcal{G}) \)
   2. \((W_0', W_1') := \text{McNaughton}(\mathcal{G}')\)
   3. \( \text{if } (W_1'_{1-\sigma} = \emptyset) \) then
      1. \( W_\sigma := V \setminus W_{1-\sigma} \)
      2. return \((W_0, W_1)\)
   4. \( W_{1-\sigma} := W_{1-\sigma} \cup \text{Attr}_{(1-\sigma)}(W_1', \mathcal{G}) \)
   5. \( \mathcal{G} := \mathcal{G} \setminus \text{Attr}_{(1-\sigma)}(W_1', \mathcal{G}) \)
McNaughton’s Algorithm: Solving parity games

**McNaughton(G)**

1. $c :=$ highest color in $G$
2. if $c = 0$ or $V = \emptyset$
   then return $(V, \emptyset)$
3. set $\sigma$ to $c \mod 2$
4. set $W_{1-\sigma}$ to $\emptyset$
5. repeat

   1. $G' := G \setminus Attr_\sigma(\alpha^{-1}(c), G)$
   2. $(W'_0, W'_1) := McNaughton(G')$
   3. if $(W'_1 = \emptyset)$ then
      1. $W_\sigma := V \setminus W_{1-\sigma}$
      2. return $(W_0, W_1)$
   4. $W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_1, G)$
   5. $G := G \setminus Attr_{(1-\sigma)}(W'_1, G)$
McNaughton’s Algorithm: Solving parity games

\( McNaughton(G) \)

1. \( c := \) highest color in \( G \)
2. if \( c = 0 \) or \( V = \emptyset \) then return \( (V, \emptyset) \)
3. set \( \sigma \) to \( c \mod 2 \)
4. set \( W_{1-\sigma} \) to \( \emptyset \)
5. repeat

1. \( G' := G \setminus Attr_{\sigma}(\alpha^{-1}(c), G) \)
2. \((W'_0, W'_1) := McNaughton(G') \)
3. if \( (W'_1 - \sigma = \emptyset) \) then
   1. \( W_{\sigma} := V \setminus W_{1-\sigma} \)
   2. return \( (W_0, W_1) \)
4. \( W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_1 - \sigma, G) \)
5. \( G := G \setminus Attr_{(1-\sigma)}(W'_1 - \sigma, G) \)
Synthesis workflow

Specification $\xrightarrow{\text{construct}} \text{game} \xrightarrow{\text{solve}} \text{game} \xrightarrow{\text{Implementation}}$

$LTL$

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 $\varphi$ and $\psi$ are LTL formulas, then so are
  - $\neg \varphi$, $\varphi \land \varphi$, $\bigcirc \varphi$, $\varphi \mathcal{U} \psi$

Abbreviations:
- $\Diamond \varphi \equiv true \mathcal{U} \varphi$;
- $\Box \varphi \equiv \neg (\Diamond \neg \varphi)$;
- $\varphi \mathcal{W} \psi \equiv (\varphi \mathcal{U} \psi) \lor \Box \varphi$;
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 $\alpha, i \not\models \varphi$;
  $\alpha, i \models \varphi \land \psi$ iff $\alpha, i \models \varphi$ and $\alpha, i \models \psi$;
- $\alpha, i \models \Diamond \varphi$ 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: $\square p$
- Guarantee: $\Diamond p$
- Recurrence: $\square \Diamond p$
- Request-Response: $\square (p \rightarrow \Diamond q)$
- Fairness: $(\square \Diamond p) \rightarrow (\square \Diamond q)$
Synthesis workflow

Specification → construct game → solve game → Implementation

↑

LTL

↓

NBA

↓

DPA

↓

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

- \( \Sigma \): 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 $A = (\Sigma, S, I, T, F)$ 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 $A$ if there exists an accepting run of $A$ on $w$.

- The language of $A$:
  \[ L_{\omega}(A) = \{ \sigma \in \Sigma^\omega \mid \sigma \text{ is accepted by } A \} \]

  $A$ recognizes $L_{\omega}(A)$.

- Two NBAs $A$ and $A'$ are equivalent iff $L_{\omega}(A) = L_{\omega}(A')$. 
NBA vs. NFA

- finite equivalence $\not\Leftrightarrow \omega$-equivalence

- $\omega$-equivalence $\not\Rightarrow$ finite equivalence

NFA: nondeterministic finite-word automaton
LTL vs. NBA

For every LTL formula \( \varphi \) there is an NBA \( A_{\varphi} \) over \( \Sigma = 2^{AP} \) that recognizes \( \text{models}(\varphi) \).

The size of \( A_{\varphi} \) is exponential in the length of \( \varphi \).

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
  \[ |I| \leq 1 \quad \text{and} \quad |\{q' \in S \mid (q, \sigma, q') \in T\}| \leq 1 \quad \text{for all } q \in S \text{ und } \sigma \in \Sigma \]

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

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

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

A run \( \pi \) of a parity automaton is **accepting** iff \( \max\{c(q) \mid q \in \text{ln}(\pi)\} \) 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.
Corollary: For every LTL formula $\varphi$ there exists a DPA $P_\varphi$ such that $L(P_\varphi) = \text{models}(\varphi)$.

The number of states of $P_\varphi$ is doubly-exponential in the length of $\varphi$.

Example:

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

Smallest deterministic automaton recognizing $L_n$ has $2^{2^n}$ states.

$L_n$ can be defined with small (quadratic) LTL formula:

$$[(\neg \$ U \$ \land \bigcirc \square \neg \$)] \land$$
$$\diamond[\# \land \land_{1 \leq i \leq n}(\bigcirc^i 0 \land \square(\$ \rightarrow \bigcirc^i 0)) \lor (\bigcirc^i 1 \land \square(\$ \rightarrow \bigcirc^i 1))]$$
"Only issue a grant when there is a request."

- **LTL:** $\Box(\neg r \rightarrow \neg g)$
- **DPA:**
  - Transition diagram:
  - Parity game:
Example

“Only issue a grant when there is a request.”

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

```
0 –> 1
\{g\}
\{r, g\}, \{r\}, \emptyset
\{r, g\}, \{r\}, \{g\}, \emptyset
```

- **Parity game:**
Synthesis workflow

Specification $\rightarrow$ construct game $\rightarrow$ solve game $\rightarrow$ Implementation

↑

Transducer
Transducer

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

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

The winning strategy can be represented as a transducer.
Example

- Parity game:

- Transducer:
Synthesis workflow

Specification → construct game → solve game → Implementation

↑↑↑↑
LTL NBA DPA safety/ Büchi/ parity games transducer
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!