A landscape with games in the background

Igor Walukiewicz
Bordeaux University

Plan

- Basic games.
- Games behind model-checking.
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.
**Propositional logic (model checking)**

- Propositions: $\neg\text{Prop} \mid \varphi \lor \psi \mid \varphi \land \psi$

- Valuation: $V : \text{Prop} \rightarrow \{0, 1\}$

- Model checking rules:
  - Eve chooses $V \models \varphi \lor \psi$:
    - $V \models \varphi$
    - $V \models \psi$
  - Adam chooses $V \models \varphi \land \psi$:
    - $V \models \varphi$
    - $V \models \psi$

- Eve wins if $V(P) = 1$.
- Eve wins if $V(P) = 0$.

- Eve has a winning strategy from $V \models \varphi$ iff $\varphi$ is true in $V$. 

---
We want to design a game for satisfiability checking.

We work with sets of formulas.

\[ \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma} \quad \frac{\varphi \lor \psi, \Gamma}{\psi, \Gamma} \]

Eve chooses

\[ \frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma} \]

Adam chooses

if \( \Gamma \)-irreducible then Eve wins if no \( P, \neg P \in \Gamma \).

Eve has a winning strategy from \( \{ \varphi \} \) iff \( \varphi \) is satisfiable.
We want to design a game for satisfiability checking.

We work with sets of formulas.

\[
\frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma} \quad \frac{\varphi \lor \psi, \Gamma}{\psi, \Gamma}
\]

Eve chooses

\[
\frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma}
\]

Adam chooses

if \(\Gamma\)-irreducible then Eve wins if no \(P, \neg P \in \Gamma\).

Eve has a winning strategy from \(\{\varphi\}\) iff \(\varphi\) is satisfiable.

Every model of \(\varphi\) can be obtained from a winning strategy in the satisfiability game for \(\{\varphi\}\).
In the satisfiability game Adam has nothing to say. (This is a peculiarity of the simple case).

Satisfiability games are related to synthesis.

MC games are related to model-checking.

In the MC game we work with formulas while in the satisfiability game we work with sets of formulas. (Boolean algebra).

Satisfiability games are constructed from MC games by a kind of power-set construction. (A position in sat game is like a set of positions in MC game)
Part Ib

Games: basic definitions.

- Games behind model-checking.
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.
\[ G = \langle V_E, V_A, R, \lambda : V \to C, \text{Acc} \subseteq C^\omega \rangle \]

Eve wins a play if the labeling of it is in Acc.

(There is an edge from every node.)
Path forming games

\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, \ Acc \subseteq C^\omega \rangle \]

Eve wins a play if the labeling of it is in Acc.

(There is an edge from every node.)
Path forming games

\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, \text{Acc} \subseteq C^\omega \rangle \]

**Eve wins** a play if the labeling of it is in \( \text{Acc} \).

(There is an edge from every node.)
Path forming games

\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, \text{Acc} \subseteq C^\omega \rangle \]

Eve wins a play if the labeling of it is in Acc.

(There is an edge from every node.)
Path forming games

\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \to C, \text{Acc} \subseteq C^\omega \rangle \]

Eve wins a play if the labeling of it is in Acc.
(There is an edge from every node.)
\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, \text{Acc} \subseteq C^\omega \rangle \]

**Path forming games**

- Eve wins a play if the labeling of it is in \text{Acc}.
- (There is an edge from every node.)
\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, \text{Acc} \subseteq C^\omega \rangle \]

- **Strategy** for Eve is \( \sigma : V^* \times V_E \rightarrow V \) such that \( \sigma(\vec{v}v_0) \in R(v_0) \)

- A strategy \( \sigma \) for Eve is **winning from** \( v \) if all plays from \( v \) respecting the strategy are winning for Eve.

- **Positional/memoryless strategy** for Eve is a function \( \sigma : V_E \rightarrow V \) such that \( \sigma(v) \in R(v) \).
Winning conditions

\[ \mathcal{G} = \langle V_E, V_A, R, \lambda : V \to C, \text{Acc} \subseteq C^\omega \rangle \]

- **Inf}_{\lambda}(\vec{v})**: the set of colours appearing infinitely often on a path \( \vec{v} \).
- **Muller condition**: given by a partition of \( \mathcal{P}(C) \) into \( (\mathcal{F}_E, \mathcal{F}_A) \).
  \[ \vec{v} \in \text{Acc} \iff \{ \vec{v} : \text{Inf}_{\lambda}(\vec{v}) \in \mathcal{F}_E \} \]

- **Parity condition** colours are numbers \( \{0, \ldots, d\} \) and:
  \[ \vec{v} \in \text{Acc} \iff \text{min}(\text{Inf}_{\lambda}(\vec{v})) \text{ is even.} \]
**Thm:** Every game with a Muller winning condition is determined, i.e., from every vertex one of the players has a winning strategy.

**Thm:** In a parity game a player has a memoryless winning strategy from each of his winning vertices.

**Def:** To **solve a game** is to determine for each position who has a winning strategy from this position.

**Thm:** There is an algorithm for solving finite Muller games.

[Martin, Emerson & Jutla, Mostowski]
Part II

- Games: basic definitions.
- **Games behind model-checking.**
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.
The mu-calculus

Syntax: \( P \mid \neg P \mid X \mid \alpha \lor \beta \mid \alpha \land \beta \mid \langle a \rangle \alpha \mid [a] \alpha \mid \mu X. \alpha \mid \nu X. \alpha \)

Semantics in a transition system \( \mathcal{M} = \langle V, \{ R_a \}_{a \in \text{Act}}, P^\mathcal{M}, \ldots \rangle \); we need \( \text{Val} : \text{Var} \to \mathcal{P}(V) \)

\[
\begin{align*}
[ P ]^\mathcal{M}_\text{Val} &= P^\mathcal{M} \\
[ X ]^\mathcal{M}_\text{Val} &= \text{Val}(X) \\
[ \langle a \rangle \alpha ]^\mathcal{M}_\text{Val} &= \{ v : \exists v'. \ R_a(v, v') \land v' \in [ \alpha ]^\mathcal{M}_\text{Val} \} \\
[ \mu X. \alpha (X) ]^\mathcal{M}_\text{Val} &= \bigcap \{ S \subseteq V : [ \alpha (S) ]^\mathcal{M}_\text{Val} \subseteq S \}
\end{align*}
\]

The model-checking problem: given a sentence \( \alpha \), a finite transition system \( \mathcal{M} \), and a state \( s_0 \), check if \( s_0 \in [ \alpha ]^\mathcal{M} \).
(Notation \( \mathcal{M}, s_0 \models \alpha \))
We are given a transition system $\mathcal{M}$.

Model checking rules

- $s \vDash \alpha \lor \beta$ if $s \vDash \alpha$ or $s \vDash \beta$
- $s \vDash \alpha \land \beta$ if $s \vDash \alpha$ and $s \vDash \beta$
- $s \vDash \langle a \rangle \alpha$ if $t \vDash \alpha$ and $(s, t) \in R_a$
- $s \vDash [a] \alpha$ if $t \vDash \alpha$

$s \vDash P$ if $s \in P^\mathcal{M}$;
$s \vDash \neg P$ if $s \not\in P^\mathcal{M}$.
We are given a transition system $\mathcal{M}$.

Model checking rules

- $s \models \alpha \lor \beta$ if $s \models \alpha$ or $s \models \beta$
- $s \models \alpha \land \beta$ if $s \models \alpha$ and $s \models \beta$
- $s \models \langle a \rangle \alpha$ if $s \models \alpha$ and $(s, t) \in R_a$
- $s \models [a] \alpha$ if $t \models \alpha$

- $s \models P$ if $s \in P^\mathcal{M}$; $s \models \neg P$ if $s \notin P^\mathcal{M}$.

- $s \models \mu X.\alpha(X)$ if $s \models \alpha(\mu X.\alpha(X))$
- $s \models \nu X.\alpha(X)$ if $s \models \alpha(\nu X.\alpha(X))$

The last two rules may be a source of infinite plays.

Wanted: Eve wins in $G(\mathcal{M}, \alpha)$ from $s_0 \models \alpha$ iff $\mathcal{M}, s_0 \models \alpha$. 

- p.13/7
 Infinite plays

\[ s \models \mu X.\langle a\rangle X \]
\[ \downarrow \]
\[ s \models \langle a\rangle \mu X.\langle a\rangle X \]
\[ \downarrow \]
\[ s \models \mu X.\langle a\rangle X \]
\[ \downarrow \]
\[ \vdots \]

\[ s \models \nu X.\langle a\rangle X \]
\[ \downarrow \]
\[ s \models \langle a\rangle (\nu X.\langle a\rangle X) \]
\[ \downarrow \]
\[ s \models \nu X.\langle a\rangle X \]
\[ \downarrow \]
\[ \vdots \]

- Eve should win in the second game but not in the first.
\[ \mu X.\beta (X) = \bigcup_{\tau \in \text{Ord}} \mu^\tau X.\beta (X) \]

\[ \left[ \mu^0 X.\beta (X) \right]_{\text{Val}} = \emptyset \]

\[ \left[ \mu^{\tau+1} X.\beta (X) \right] = \left[ \beta (X) \right]_{\text{Val}} \left[ \left[ \mu^\tau X.\beta (X) \right]_{\text{Val}} / X \right] \]

\[ \left[ \mu^\tau X.\beta (X) \right]_{\text{Val}} = \bigcup_{\tau' < \tau} \left[ \mu^{\tau'} X.\beta (X) \right]_{\text{Val}} \quad \text{if } \tau \text{ is a limit ordinal} \]

\[ \nu X.\beta (X) = \bigcap_{\tau \in \text{Ord}} \nu^\tau X.\beta (X) \]

\[ \left[ \nu^0 X.\beta (X) \right]_{\text{Val}} = V \]

\[ \left[ \nu^{\tau+1} X.\beta (X) \right] = \left[ \beta (X) \right]_{\text{Val}} \left[ \left[ \nu^\tau X.\beta (X) \right]_{\text{Val}} / X \right] \]

\[ \left[ \nu^\tau X.\beta (X) \right]_{\text{Val}} = \bigcap_{\tau' < \tau} \left[ \nu^{\tau'} X.\beta (X) \right]_{\text{Val}} \quad \text{if } \tau \text{ is a limit ordinal} \]
Eve should win in the second game but not in the first.
Eve should win in the second game but not in the first.

Assign rank 1 to \( \mu \)-regeneration and rank 2 to \( \nu \)-regeneration.
Defining winning conditions

\[ \mu X_1. \nu X_2. \mu X_3. \nu X_4 \ldots \varphi(X_1, X_2, \ldots) \]

- \( \mu \)'s have odd ranks,
- \( \nu \)'s have even ranks,
- if \( \beta \) is a subformula of \( \alpha \) then \( \beta \) has bigger rank than \( \alpha \).

With such acceptance conditions we have:

**Thm [Emerson & Jutla, Stirling]:**

\[ \mathcal{M}, s_0 \models \alpha \quad \text{iff} \quad \text{Eve wins in } G(\mathcal{M}, \alpha) \text{ from } s_0 \models \alpha. \]
\[ \nu Y. \mu X. (P \land \langle a \rangle Y) \lor \langle b \rangle X \]

\[ s \models \langle a \rangle Y \]
\[ s \models P \land \langle b \rangle Y \]
\[ s \models P \]
\[ s \models \beta^2_Y \]
\[ s \models \beta^3_X \]
\[ G(\mathcal{M}, \alpha) \]
Model checking rules

- $s \models \alpha \lor \beta$

  - $s \models \alpha$
  - $s \models \beta$

- $s \models \alpha \land \beta$

  - $s \models \alpha$
  - $s \models \beta$

- $s \models \langle a \rangle \alpha$

  - $t \models \alpha$
  - $(s, t) \in R_a$

- $s \models [a] \alpha$

  - $t \models \alpha$

- $s \models \mu X. \alpha(X)$

  - $s \models \alpha(\mu X. \alpha(X))$

- $s \models \nu X. \alpha(X)$

  - $s \models \alpha(\nu X. \alpha(X))$

$s \models P$  Eve wins if $s \in P^M$;  $s \models \neg P$  Eve wins if $s \notin P^M$. 
Tableaux rules

These rules define a tableau $T_\alpha$ for a formula $\alpha$.

Operation $\mathcal{M} \otimes T_\alpha$ of “synchronized product” of a transition system and a tableau that gives the MC game.

Obs: $\mathcal{M}, s_0 \models \alpha$ iff Eve wins from $(s_0, \alpha)$ in $\mathcal{M} \otimes T_\alpha$. 
\( \nu Y. \mu X. (P \land \langle a \rangle Y) \lor \langle b \rangle X \)
Example

\[ \nu Y. \mu X. (P \land \langle a \rangle Y) \lor \langle b \rangle X \]

\[ s \models \langle a \rangle Y \]
\[ s \models P \land \langle b \rangle Y \]
\[ s \models P \]
\[ s \models \beta^3_X \]
\[ s \models \beta^2_Y \]
\[ s \models \langle b \rangle X \]

\[ M \otimes T_{\beta_Y} \]
Given a structure $\mathcal{M}$ and a formula $\alpha$ we construct the game $G(\mathcal{M}, \alpha)$ such that:

$\mathcal{M}, s \models \alpha$ iff Eve wins from $(s \models \alpha)$ in $G(\mathcal{M}, \alpha)$

The winning condition in $G(\mathcal{M}, \alpha)$ is a parity condition which size is the depth of alternation of fixpoints in $\alpha$.

One can defined a tableau $\mathcal{T}_\alpha$ and a synchronized product $\mathcal{M} \otimes \mathcal{T}_\alpha$ so that $G(\mathcal{M}, \alpha) = \mathcal{M} \otimes \mathcal{T}_\alpha$.

In particular the size of $|\mathcal{M}| \otimes |\mathcal{T}_\alpha|$ is $|\mathcal{M}| \cdot |\alpha|$.

This works also for infinite transition systems.
A game can be represented as a transition system where

- proposition $P_E$ designates Eve's positions,
- propositions $P_0, \ldots, P_d$ define $\lambda : V \to \{0, \ldots, d\}$.

**Thm [Emerson & Jutla]:** There is a formula of the mu-calculus $\varepsilon_d$ such that

$$\mathcal{M}_G, v \models \varepsilon_d \iff \text{Eve wins from } v \text{ in } G.$$

$$\gamma(Z_0, \ldots, Z_d) =$$

$$(P_E \land \bigwedge_{i=0,\ldots,d} (P_i \Rightarrow \langle \rangle Z_i)) \lor (\neg P_E \land \bigwedge_{i=0,\ldots,d} (P_i \Rightarrow [ ] Z_i))$$

$$\varepsilon_d = \nu Z_0. \mu Z_1. \ldots \sigma Z_d. \gamma(Z_0, \ldots, Z_d)$$
In summary

- Parity games and model-checking for the mu-calculus are very close to each other (inter-reducible in linear time).

  \[ \mathcal{M}, s \models \alpha \text{ iff in } \mathcal{M} \otimes \mathcal{T}_\alpha \text{ Eve wins from } (s, \alpha) \]

- Because of this translation it is enough to consider the games solving problem instead of MC problem.

- The tableau construction gives an alternating automaton accepting models of the formula.

- The \( \mathcal{M} \otimes \mathcal{T}_\alpha \) operation defines the space of runs of the automaton \( \mathcal{T}_\alpha \) on the structure \( \mathcal{M} \).

- As \( \mathcal{T}_\alpha \) accepts all models of \( \alpha \), the satisfiability problem reduces to the emptiness test of \( \mathcal{T}_\alpha \).

- Indeed, the satisfiability game is obtained from converting \( \mathcal{T}_\alpha \) into a nondeterministic automaton.
Pushdown systems

Pushdown system: \[ P = (Q, \Gamma, \Delta) \]

Transitions rules: \[ \Delta \subseteq Q \times \Gamma \times Q \times Op \]

\[ (q, a) \mapsto (q', pop) \quad (q, a) \mapsto (q', push_b) \]

Pushdown graph: \[ G(P) \]

Vertices: \( Q \times \Gamma^* \)

Edges: \( qw \rightarrow q'w' \) according to the rules.

\( q_0 \) is always the initial state and \( \perp \) is the initial stack symbol.
Pushdown graph: an example

This is (a part of) the graph of the system:

\[(q_0, a) \mapsto (q_0, \text{push}_a)\quad (q_0, a) \mapsto (q_1, \text{pop})\]
\[(q_1, a) \mapsto (q_1, \text{pop})\]
\[(q_0, \bot) \mapsto (q_0, \text{push}_a)\quad (q_1, \bot) \mapsto (q_0, \text{push}_a)\]

The **push-down model checking problem**:

Given \(P\) and \(\alpha\) decide if \(\alpha\) holds in the initial vertex of \(G(P)\).
Given $P$ and $\alpha$ decide if $\alpha$ holds in the initial vertex of $G(P)$.

Construct $T_\alpha$ and the product $G(P) \otimes T_\alpha$.

This gives an infinite pushdown game:

$$P = \langle Q, \Gamma, \Delta, Q_E, Q_A, \Omega : Q \rightarrow \mathbb{N} \rangle$$

- pushdown system with states partitioned between Eve and Adam
- where each state is assigned a rank ($\Omega : Q \rightarrow \mathbb{N}$).

$\alpha$ holds in the initial vertex of $G(P)$ iff
Eve has a winning strategy from the initial vertex in the game.
We have that:

- $q_0$ is a vertex of Adam and $q_1$ of Eve;
- $\Omega(q_0) = 0$ and $\Omega(q_1) = 1$.

Eve has a winning strategy in this game.

The game solving problem: Given $P$ with a partition $(Q_E, Q_A)$ of states, and a function $\Omega : Q \rightarrow \mathbb{N}$ decide who has a winning strategy from the initial vertex of $G(P)$.

**Thm:** The problem of solving parity pushdown games is EXPTIME-complete. The same for MC problem.
Higher-order pushdown systems

- **1-store**: a sequence $a_l \ldots a_1$ over an alphabet $\Gamma$.
- **$n$-store**: a sequence $[s_l] \cdots [s_1]$ of $(n - 1)$-stores.
- We have standard operations $\text{push}_a^1$ and $\text{pop}_a^1$.
- Additionally we have $\text{push}_a^k$ and $\text{pop}_a^k$ operations:

\[
\text{push}_a^k([s_l] \cdots [s_1]) = \begin{cases} 
[s_l][s_l] \cdots [s_1] & \text{stack order} = k \\
\text{push}_a^k(s_l) \cdots [s_1] & \text{stack order} > k 
\end{cases}
\]

\[
\text{pop}_a^n([s_l][s_{l-1}] \cdots [s_1]) = \begin{cases} 
[s_{l-1}] \cdots [s_1] & \text{stack order} = k \\
[s_{l-1}] \cdots [s_1] & \text{stack order} > k 
\end{cases}
\]

- Pushdown system of order $n$: $P = \langle Q, \Gamma, \Delta \rangle$ where

$\Delta \subseteq Q \times \Gamma \times Q \times O_{p_n}$. 
A system where all paths are of the form \( q_1^k q_2^k q_3^k \)

\[

tabular
\hline
q_1[a] & \rightarrow & q_1[aa] & \rightarrow & \cdots & \rightarrow & q_1[a^k] \\
q_2[a^k][a^k] & \rightarrow & q_2[a^{k-1}][a^k] & \rightarrow & \cdots & \rightarrow & q_2[][a^k] \\
q_3[a^k] & \rightarrow & q_3[a^{k-1}] & \rightarrow & \cdots & \rightarrow & q_3[] \\
\hline
\]

2-store gives additional power. If considered as an accepting device 2-store automaton would recognize \( \{ a^k b^k c^k : k \in \mathbb{N} \} \).
Once again the model checking problem reduces to solving games. This time higher-order pushdown games.

Such a game is given by a higher-order pushdown automaton with states partitioned into Adam’s and Eve’s states and a function $\Omega : Q \rightarrow \mathbb{N}$.

**Thm[Engelfreit, Cachat]:** Solving $n$-order pushdown games is $n$-EXPTIME complete.

Higher-order pushdown automata “implement” higher-order (safe) program schemes.

The graphs of configurations of $n$-order pushdown automata are the graphs of $n$-th level of the Caucaul hierarchy.
Part III

- Games: basic definitions.
- Games behind model-checking.

**Games behind synthesis.**

- Extensions of the basic game model.
- Distributed synthesis.
Synthesis via satisfiability checking

Synthesis problem I: Given a specification find a system satisfying it.


$$\phi \lor \psi, \Gamma \quad \phi, \Gamma \quad \phi \lor \psi, \Gamma \quad \phi, \Gamma$$

Eve chooses

$$\phi \land \psi, \Gamma \quad \phi, \psi, \Gamma$$

Adam chooses

when $\Gamma$-irreducible then Eve wins if no $P, \neg P \in \Gamma$.

Eve has a winning strategy from $\{\phi\}$ iff $\phi$ is satisfiable.

Every model of $\phi$ can be obtained from a winning strategy in the satisfiability game for $\{\phi\}$. 
Extension to the $\mu$-calculus

Adam chooses

$$\Gamma, \{ \beta : [a]\beta \in \Gamma \} : \langle a \rangle \alpha \in \Gamma$$

$$\Gamma, \mu X.\alpha(X) \quad \Gamma, \nu X.\alpha(X)$$

There are now infinite paths and we need a rule to decide the winner there.
Path condition

\[
\begin{align*}
\mu X. \langle a \rangle X & \quad \nu X. \langle a \rangle X \\
\langle a \rangle (\mu X. \langle a \rangle X) & \quad \langle a \rangle (\nu X. \langle a \rangle X) \\
\mu X. \langle a \rangle X & \quad \nu X. \langle a \rangle X
\end{align*}
\]

- On the left Adam should win on the right it should be Eve.

\[
\begin{align*}
\mu X. [a] X, \nu X. \langle a \rangle X & \\
[a] (\mu \ldots), \langle a \rangle (\nu \ldots) & \\
\mu X. [a] X, \nu X. \langle a \rangle X
\end{align*}
\]

- The conditions should talk about traces inside the path.
Extension to the $\mu$-calculus

\[
\Gamma \vdash \{\alpha, \{\beta : [a] \beta \in \Gamma\} : \langle a \rangle \alpha \in \Gamma\}
\]

Adam chooses

\[
\frac{\Gamma, \mu X. \alpha(X)}{\Gamma, \alpha(\mu X. \alpha(X))}
\quad \frac{\Gamma, \nu X. \alpha(X)}{\Gamma, \alpha(\nu X. \alpha(X))}
\]

- The rule for infinite paths says that Eve wins if there is no bad trace inside the path. (It can be converted to a parity condition).

- Eve wins from $\{\alpha\}$ iff $\alpha$ is satisfiable.

- Every model (transition system) for $\alpha$ comes from some winning strategy in this game.

**Thm [Emerson & Jutla]:** The satisfiability problem for the $\mu$-calculus is $\text{EXPTIME}$-complete.
A plant is a deterministic transition system over $\Sigma$.

$$P = \langle S^p, \Sigma, s^p_I, e^p : S \times A \rightarrow S \rangle$$

Given $P$ and $\alpha$, find a controller $C$ (deterministic transition system) s.t. $P \times C \models \alpha$.

$$P \times C = \langle S = S^p \times S^c, \Sigma, (s^p_I, s^c_I), e : S \times \Sigma \rightarrow S \rangle$$

$$e((s^p, s^c), a) = (e^p(s^p, a), e^c(s^c, a))$$

Solution:
- Define an operation $\alpha/P$ such that:
  $$C \models \alpha/P \iff P \times C \models \alpha$$
- Find a model $C$ of $\alpha/P$. 
Example

$$\alpha \equiv \text{execute } d \text{ action}$$

Some possible controllers:

The second solution is non-blocking.

$$C \models (\alpha / P) \land \beta_{\text{nonblock}}$$

So we can require additional properties from the controller.
Uncontrollable/unobservable actions

- Divide $\Sigma$ into:
  - $\Sigma_{\text{con}}$ and $\Sigma_{\text{ucon}}$ of controllable and uncontrollable actions.
  - $\Sigma_{\text{obs}}$ and $\Sigma_{\text{uobs}}$ of observable and unobservable actions.

- Additional conditions:

\[ \theta_{\text{ucon}} : \quad C \text{ cannot forbid actions from } \Sigma_{\text{ucon}} \]
\[ \equiv \forall s \in S. \quad \forall a \in \Sigma_{\text{ucon}}. \quad e(s, a) \text{ defined} \]
\[ \equiv \nu X. \quad \left( \bigwedge_{a \in \Sigma} [a]X \right) \land \left( \bigwedge_{a \in \Sigma_{\text{ucon}}} \langle a \rangle tt \right) \]

\[ \theta_{\text{uobs}} : \quad C \text{ cannot observe actions from } \Sigma_{\text{uobs}} \]
\[ \equiv \forall s \in S. \quad \forall a \in \Sigma_{\text{uobs}}. \quad e(s, a) = s \]
\[ \equiv \nu X. \quad \left( \bigwedge_{a \in \Sigma} [a]X \right) \land \left( \bigwedge_{a \in \Sigma_{\text{uobs}}} \ominus_a \right) \]
Uncontrollable/unobservable actions

- Divide $\Sigma$ into:
  - $\Sigma_{\text{con}}$ and $\Sigma_{\text{ucon}}$ of controllable and uncontrollable actions.
  - $\Sigma_{\text{obs}}$ and $\Sigma_{\text{uobs}}$ of observable and unobservable actions.

- Additional conditions:

  $\theta_{\text{ucon}} : \ C \text{ cannot forbid actions from } \Sigma_{\text{ucon}}$

  $$\equiv \nu X. \left( \bigwedge_{a \in \Sigma} [a]X \right) \land \left( \bigwedge_{a \in \Sigma_{\text{ucon}}} \langle a \rangle tt \right)$$

  $\theta_{\text{uobs}} : \ C \text{ cannot observe actions from } \Sigma_{\text{uobs}}$

  $$\equiv \nu X. \left( \bigwedge_{a \in \Sigma} [a]X \right) \land \left( \bigwedge_{a \in \Sigma_{\text{uobs}}} \Diamond a \right)$$

- **Solution**: Find $C \models (\alpha/P) \land \theta_{\text{ucon}} \land \theta_{\text{uobs}}$
\[ \Sigma_{uobs} = \{a\}, \Sigma_{ucon} = \{e\} \] and the goal is to avoid \( e \)

- A solution:
  - \( d \)
  - \( a \)
  - \( e \)

- Wrong:
  - \( a \)
  - \( c \)
  - \( d \)
Now we have reduced various synthesis problems to games.

Games themselves can be considered as specifications and strategies as programs.

In this kind of setting we can vary only the shape of a graph and the rest of a specification is fixed.

All centralized synthesis problems are reducible to this one.
Part IV

- Games: basic definitions.
- Games behind model-checking.
- Games behind synthesis.

**Extensions.**

- Distributed synthesis.
More elaborate winning conditions

- The mu-calculus specifications translate into parity winning conditions. Similarly for other standard program logics.

- In the context of push-down games we have phenomena not expressible in these logics:
  - explosion: the height of the stack is unbounded.

Thm[Cachat & Duparc & Thomas, Bouquet & Serre & W., Gimbert]: Games with winning conditions that are boolean combinations of parity and explosion conditions can be solved in $\text{EXPTIME}$.

Thm: Winning conditions that are unions of explosion and parity conditions admit memoryless strategies. Intersection of Büchi and explosion conditions may need infinite memory.
Example

\[ q_0^0 \perp \rightarrow q_0^0 a \perp \rightarrow q_0^1 a a \perp \rightarrow q_0^1 a a a \perp \rightarrow \cdots \rightarrow q_0^1 a^k \perp \rightarrow \cdots \]

\[ q_1^1 \leftarrow q_1^1 a \leftarrow q_1^1 a a \leftarrow \cdots \leftarrow q_1^1 a^{k-1} \leftarrow \cdots \]
Memoryless strategies are interesting as the size of memory influences:
- the size of controllers,
- the size of counterexamples,
- the complexity of the algorithms.

A winning condition admits positional determinacy iff all the games with this condition are positionally determined.

**Thm [McNaughton]:** Parity condition is the only Muller condition admitting positional determinacy.

**Rem [Zielonka]:** If all nodes need to be coloured then the class is a bit bigger.
Muller conditions with infinite number of colours.

\[ G = \{ V_E, V_A, R, \lambda : V \to \omega \} \]

Infinite parity condition:

Eve wins iff \( \min(\text{Inf}(p)) \) is even or \( \text{Inf}(p) = \emptyset \).

**Thm [Graedel & W.]:** Games with infinite parity condition admit memoryless determinacy. All other conditions need infinite memory.

**Thm [Graedel & W.]:** The conditions given by \( \lambda : V \to (\omega + 1) \) admit positional determinacy over graphs of bounded out-degree.

**Thm [Colcombet & Niwiński]:** If partial colouring functions are allowed then only finite parity conditions admit positional determinacy.
Games with time

Like in timed automata, game has clocksand restrictions on when transitions can bетaken.
A taxonomy of the types of rules:

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Probabilistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn-based</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Concurrent</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
We add randomized positions: \( \triangle \).

In such a vertex we have a probability distribution on outgoing edges.

Adam wins in this game
None of the players may be sure to win.

Eve wins with the probability $\frac{2}{3}$ and Adam with the probability $\frac{1}{3}$.

Thm [de Alfaro & Majumdar, Chatterjee & Jurdziński & Henzinger, Zielonka] :
In a finite game each state has a value and each player has an positional, pure and optimal strategy.
Two players choose their moves concurrently. Their joint choice determines the successor.

None of the players may have a pure winning strategy.

There exists randomized strategies, but they may require infinite memory [de Alfaro, Henzinger].
Part V

- Games: basic definitions.
- Games behind model-checking.
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.
Given a plant $P$ and formulas $\alpha, \beta_1, \beta_2$ do there exist controllers $C_1, C_2$ such that:

$C_1 \models \beta_1, \quad C_2 \models \beta_2 \quad \text{and} \quad P \times C_1 \times C_2 \models \alpha.$

(Each controller has its own $\Sigma^i_{ucon}$ and $\Sigma^i_{uobs}$.)

Synthesis for a given architecture:
Given a plant $P$ and formulas $\alpha, \beta_1, \beta_2$ do there exist controllers $C_1, C_2$ such that:

$$C_1 \models \beta_1, \quad C_2 \models \beta_2 \quad \text{and} \quad P \times C_1 \times C_2 \models \alpha.$$  

(Each controller has its own $\Sigma^i_{ucon}$ and $\Sigma^i_{uobs}$.)

 Define new operation $\alpha/\beta$ with the property:

$$P \models \alpha/\beta \iff \text{there is } C \text{ such that } C \models \beta \text{ and } P \times C \models \alpha$$

The operation $\alpha/\beta$ works only if $\beta$ does not use $\ominus$.

We have:

$$P \models (\alpha/\beta_1)/\beta_2 \iff \text{there is } C_2 \text{ with } P \times C_2 \models \alpha/\beta_1$$

$$\text{iff there are } C_1, C_2 \text{ with } P \times C_1 \times C_2 \models \alpha.$$
Fact: The following problem is undecidable: Given $\alpha, \beta_1, \beta_2$ are there $C_1, C_2$ such that $C_1 \times C_2 \models \alpha$ and $C_1 \models \beta_1, C_2 \models \beta_2$.

Thm[PNueli & Rosner]: The problem:

For a fixed architecture, given $\alpha$ are there controllers that make the system satisfy $\alpha$.

is decidable only for pipelines.
Fact: The following problem is undecidable: Given $\alpha, \beta_1, \beta_2$ are there $C_1, C_2$ such that $C_1 \times C_2 \models \alpha$ and $C_1 \models \beta_1, C_2 \models \beta_2$.

Thm[Pnueli & Rosner]: The problem:

For a fixed architecture, given $\alpha$ are there controllers that make the system satisfy $\alpha$.

is decidable only for pipelines.
A specification is *local* if it is a conjunction of requirements on each controller.

**Thm[Madhusudan]:** The problem:

For a fixed architecture given a *local* specification, are there controllers that make the system satisfy the specification.

is decidable only for doubly flanked pipelines.
For most architectures there are specifications that make the problem undecidable.

It may be more fruitful to take a specification into account and look for which pairs (architecture, specification) the problem is decidable.

Idea: Compile (architecture, specification) pair into a game and use tools developed there.

Problem: Compiling to two player games does not make much sense.

We want a setting with a coalition of players against the environment.

Distributed games to distributed strategies as standard games to centralized strategies.
Solving distributed synthesis

\[ P(n) \rightarrow P(n - 1) \]

\[ P(2) \rightarrow P(1) \]

\[ G(2) \]

Solution
The game setting can be:

- more general,
- combinatorially easier to handle.
Take \( n \) “local” games \( G_i = \langle A_i, R_i, T_i \rangle \). (bipartite)

Distributed game \( G = \langle A, E, R, Acc \subseteq (E \cup P)^\omega \rangle \).

\[ E = E_1 \times \cdots \times E_n, \]
\[ A \subseteq (A_1 \cup E_1) \times \cdots \times (A_n \cup E_n) \setminus E. \]

Eve’s (environment) moves: \([e_1, \ldots, e_n] \rightarrow (x_1, \ldots, x_n)\) with \( x_i = e_i \) or \( e_i \rightarrow x_i \).

Some of these transitions can be suppressed.

Adam (system) moves: \((x_1, \ldots, x_n) \rightarrow [e_1, \ldots, e_n]\) with \( x_i = e_i \) or \( x_i \rightarrow e_i \).

Every such transition must be present.
Goal: Avoid blue positions.
Goal: Avoid blue positions.
Given a play $\bar{v}$ in $G$, a view of Adam $i$ is $\text{view}_i(v) \in (E_i \cdot P_i)\omega$.

$(e_1, e_2, \ldots, e_n)$  $e_1$

$(e_1, p_2, \ldots, e_n)$  $p_1$

$(e_1, e'_2, \ldots, e_n)$

$(p_1, e_2, \ldots, e_n)$

$(e'_1, e'_2, \ldots, e_n)$  $e_1$

An $i$-local strategy is a strategy in the game $G_i$.

Distributed strategy is a tuple $\langle \sigma_1, \ldots, \sigma_n \rangle$ of local strategies.

$$\sigma(\bar{v} \cdot (x_1, \ldots, x_n)) = (e_1, \ldots, e_n)$$

where $e_i = x_i$ or $e_i = \sigma_i(\text{view}_i(\bar{v} \cdot x_i))$. 
Adams may have a global strategy in a game but not a distributed one. (Distributed games are not determined).

Distributed games are like concurrent games, but the players who have partial information play with and not against each other.

It is not decidable if there is a distributed winning strategy in a given distributed game.

There may be a memoryless global strategy but all distributed strategies may require memory.
Example 3

Goal: Avoid blue positions.
Goal: Avoid blue positions.
The game is **deterministic for the environment** iff every environment position has at most one successor.

If Adams have a global strategy in such a game then they have a distributed one.

**Cor:** Environment deterministic distributed games are solvable. (Existence of distributed strategies is decidable).
A game is \( k \)-deterministic if whenever

\[
\vec{e} \quad \text{then} \quad \vec{x}[k] \neq \vec{y}[k].
\]

(Adam \( k \), can deduce the move of Eve).

If a game is 1 and \( n \)-deterministic then we can “glue together” players 1 and \( n \) (Thm 1).

We get a game with smaller number of Adams.

There is a distributed strategy in the new game iff there is one in the old game.

If a game is not \( i \)-deterministic then, under some conditions, we can apply a kind of “powerset construction” to make it \( i \)-deterministic (Thm 2).
Distributed games are in general neither determined nor algorithmically solvable.

Many known settings of distributed synthesis are representable in distributed games.
- Pipelines.
- Local specifications and double flanked pipelines.
- Madhusudan & Thiagarajan setting.
- Rudie & Wonham distributed control.

The solutions require some coding and two theorems.

Distributed games can be hopefully as useful for distributed synthesis problem as two player games are for the centralized synthesis problem.
Classes of graphs for which game solving is decidable.

Unsafe higher-order program schemes.

Good winning conditions for push-down systems.

More decidable cases for distributed synthesis.

Randomized strategies in distributed games.
Games are behind model-checking and synthesis problems.

Parity games are tied with the $\mu$-calculus model-checking (other logics also can be easily put into the game setting).

This connection is sometimes lost in more elaborate settings but sometimes stays (concurrent probabilistic games with parity conditions).

Often in these new settings games are all what is left from the classical setting.

New game models are needed to capture concurrency directly.
Conclusions

- Games are behind model-checking and synthesis problems.
- Parity games are tied with the $\mu$-calculus model-checking (other logics also can be easily put into the game setting).
- This connection is sometimes lost in more elaborate settings but sometimes stays (concurrent probabilistic games with parity conditions).
- Often in these new settings games are all what is left from the classical setting.
- New game models are needed to capture concurrency directly.

The playful universe is expanding.