Winning Strategies in Two-Player Games with Partial Information


Bernd Puchala

RWTH Aachen University
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]

- game played on a finite graph with labelled edges
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]

- game played on a finite graph with labelled edges
- by two antagonistic players 0 and 1,
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]

- game played on a finite graph with labelled edges
- by two antagonistic players 0 and 1,
- choosing deterministic actions from the set \( A \)
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]

- game played on a finite graph with labelled edges
- by two antagonistic players 0 and 1,
- choosing deterministic actions from the set \( A \)
- for \( \omega \) many rounds,
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]

- game played on a finite graph with labelled edges
- by two antagonistic players 0 and 1,
- choosing deterministic actions from the set \( A \)
- for \( \omega \) many rounds,
- with player 0 having the goal to establish a play in \( W_0 \subseteq V^\omega \).
Infinite Two-Player Win-Loss Games

\[ G = (V, V_0, (f_a)_{a \in A}, W_0) \]

- game played on a finite graph with labelled edges
- by two antagonistic players 0 and 1,
- choosing deterministic actions from the set \( A \)
- for \( \omega \) many rounds,
- with player 0 having the goal to establish a play in \( W_0 \subseteq V^\omega \).
- As usual, game graphs are non-terminating.
Strategies

- Strategy for player $i$:
Strategies

- Strategy for player $i$:
  
  Function $f : V^*V_i \rightarrow A$ with $f(\pi v_i) \in \text{act}(v_i)$,
  
  prescribing a next move for each finite play prefix where it is player $i$’s turn.
Strategies

- Strategy for player $i$:
  
  Function $f : V^*V_i \rightarrow A$ with $f(\pi v_i) \in \text{act}(v_i)$,

  prescribing a next move for each finite play prefix where it is player $i$’s turn

  and being compatible with the knowledge of player $i$. 


Strategies

- Strategy for player $i$:
  
  Function $f : V^* V_i \rightarrow A$ with $f(\pi v_i) \in \text{act}(v_i)$,
  prescribing a next move for each finite play prefix where it is
  player $i$’s turn
  
  and being compatible with the knowledge of player $i$.

- The knowledge of player $i$ in the game is modelled by an
  equivalence relation on $V^*$. 

Strategies

- Strategy for player $i$:
  Function $f : V^* V_i \rightarrow A$ with $f(\pi v_i) \in \text{act}(v_i)$,
  prescribing a next move for each finite play prefix where it is player $i$’s turn
  and being compatible with the knowledge of player $i$.

- The knowledge of player $i$ in the game is modelled by an equivalence relation on $V^*$.
  $\pi \sim_i \pi'$ means, that after $\pi$ has been played and after $\pi'$ has been played, player $i$ has exactly the same information.
Strategies

- Strategy for player $i$:
  
  Function $f : V^*V_i \rightarrow A$ with $f(\pi v_i) \in \text{act}(v_i)$, prescribing a next move for each finite play prefix where it is player $i$’s turn 
  
  and being compatible with the knowledge of player $i$.

- The knowledge of player $i$ in the game is modelled by an equivalence relation on $V^*$.
  
  $\pi \sim_i \pi'$ means, that after $\pi$ has been played and after $\pi'$ has been played, player $i$ has exactly the same information.

$$\pi \sim_i \pi' \implies f(\pi) = f(\pi')$$
In principle, any equivalence relation can be used here, but:
In principle, any equivalence relation can be used here, but:

- we would like to impose certain natural restrictions on $\sim_i$. 
In principle, any equivalence relation can be used here, but:

- we would like to impose certain natural restrictions on $\sim_i$.
- for algorithms, we need a finite representation of $\sim_i$. 

Knowledge Representation
In principle, any equivalence relation can be used here, but:

- we would like to impose certain natural restrictions on $\sim_i$.
- for algorithms, we need a finite representation of $\sim_i$.

 Finite representation of game graphs: finite graphs, pushdown graphs, graphs generated by finitary construction rules, . . .
Knowledge Representation

In principle, any equivalence relation can be used here, but:

- we would like to impose certain natural restrictions on $\sim_i$.
- for algorithms, we need a finite representation of $\sim_i$.

- Finite representation of game graphs: finite graphs, pushdown graphs, graphs generated by finitary construction rules, . . .
- Finite representation of winning conditions: LTL, S1S, parity conditions, . . .
Knowledge Representation

In principle, any equivalence relation can be used here, but:

- we would like to impose certain natural restrictions on $\sim_i$.
- for algorithms, we need a finite representation of $\sim_i$.

- Finite representation of game graphs: finite graphs, pushdown graphs, graphs generated by finitary construction rules, . . .
- Finite representation of winning conditions: LTL, S1S, parity conditions, . . .
- Finite representation of knowledge:
Define the information that a player has about the positions in the game graph:
Equivalence relation $\sim_i$ on $V$.

1st Extend $\sim_i$ to $\sim_i$.
Define the information that a player has about the positions in the game graph:
Equivalence relation $\sim_i^V$ on $V$.

Now we make some natural assumptions:
Knowledge Representation

1st Define the information that a player has about the positions in the game graph:
Equivalence relation $\sim^V_i$ on $V$.

Now we make some natural assumptions:

(1) $v \sim_i w \implies v, w \in V_i$ or $v, w \notin V_i$
1st Define the information that a player has about the positions in the game graph:
Equivalence relation $\sim_i^{V}$ on $V$.

Now we make some natural assumptions:

(1) $v \sim_i w \implies v, w \in V_i$ or $v, w \notin V_i$

(2) $v, w \in V_i$ with $v \sim_i w$, $a \neq b \implies f_a(v) \not\sim_i f_b(w)$
Define the information that a player has about the positions in the game graph:
Equivalence relation $\sim_i$ on $V$.

Now we make some natural assumptions:

1. $v \sim_i w \implies v, w \in V_i$ or $v, w \notin V_i$
2. $v, w \in V_i$ with $v \sim_i w, a \neq b \implies f_a(v) \not\sim_i f_b(w)$
3. $v, w \in V_i$ with $v \sim_i w \implies \text{act}(v) = \text{act}(w)$
Define the information that a player has about the positions in the game graph:
Equivalence relation $\sim^V_i$ on $V$.

Now we make some natural assumptions:

1. $v \sim^V_i w \implies v, w \in V_i$ or $v, w \notin V_i$
2. $v, w \in V_i$ with $v \sim^V_i w$, $a \neq b \implies f_a(v) \not\sim_i f_b(w)$
3. $v, w \in V_i$ with $v \sim^V_i w \implies \text{act}(v) = \text{act}(w)$

Extend $\sim^V_i$ to $\sim_i$. 
Knowledge Representation

~ If player $i$ does observe any move, then
Knowledge Representation

\[ \sim \text{ If player } i \text{ does observe any move, then} \]
\[ \pi \sim_i \pi' \text{ iff } |\pi| = |\pi'| \text{ and } \pi(j) \sim^V_i \pi'(j) \text{ for all } j. \]

(Synchronous case, player share a clock.)
If player $i$ does observe any move, then

$$\pi \sim_i \pi' \text{ iff } |\pi| = |\pi'| \text{ and } \pi(j) \sim_Y^i \pi'(j) \text{ for all } j.$$  
(Synchronous case, player share a clock.)

Now, hide moves from player $i$ in which he can’t observe anything that happens:
Knowledge Representation

~ If player $i$ does observe any move, then
$$\pi \sim_i \pi' \text{ iff } |\pi| = |\pi'| \text{ and } \pi(j) \sim_i^V \pi'(j) \text{ for all } j.$$ (Synchronous case, player share a clock.)

↑ Now, hide moves from player $i$ in which he can’t observe anything that happens:
$$\pi \overset{\leftarrow}{\sim}_i \pi' \text{ iff } \overset{\leftarrow}{\pi} \sim_i \overset{\leftarrow}{\pi'} \text{ where }$$
$$\overset{\leftarrow}{\pi} \text{ is obtained from } \pi \text{ by deleting all moves } u \to v \text{ from } \pi \text{ such that } u \in V_{1-i} \text{ and } u \sim_i^V v.$$ (Asynchronous case.)
The Question

Given a finite game $G = (G, (V_i)_{i=0,1})$ and a position $v$, does player 0 have a strategy for $G$ from $v$ which is winning against all strategies of player 1?
The Question

Given a finite game $\mathcal{G} = (G, (\sim^V_i)_{i=0,1})$ and a position $v$, does player 0 have a strategy for $\mathcal{G}$ from $v$ which is winning against all strategies of player 1?

However, this is the same as asking:
Given a finite game $\mathcal{G} = (G, (\sim^V_i)_{i=0,1})$ and a position $v$, does player 0 have a strategy for $\mathcal{G}$ from $v$ which is winning?
The Question

Given a finite game $G = (G, (\sim_i V)_{i=0,1})$ and a position $v$, does player 0 have a strategy for $G$ from $v$ which is winning against all strategies of player 1?

However, this is the same as asking:
Given a finite game $G = (G, (\sim_i V)_{i=0,1})$ and a position $v$, does player 0 have a strategy for $G$ from $v$ which is winning?

Thus, we can ignore the partial information of player 1 here!

$$\sim G = (G, \sim V)$$
Aim

For large classes of games, find
Aim

For large classes of games, find

- (efficient) algorithms for the strategy problem.
Aim

For large classes of games, find

- (efficient) algorithms for the strategy problem.
- (efficient) methods to implement winning strategies with (small) finite memory.
Aim

For large classes of games, find

- (efficient) algorithms for the strategy problem.
- (efficient) methods to implement winning strategies with (small) finite memory.

Idea:
Turn a game with partial information into a game with full information such that the existence of winning strategies for player 0 is preserved.
For large classes of games, find

- (efficient) algorithms for the strategy problem.
- (efficient) methods to implement winning strategies with (small) finite memory.

Idea:
Turn a game with partial information into a game with full information such that the existence of winning strategies for player 0 is preserved.

$\rightarrow$ Powerset Construction
Synchronous Case

\[ \mathcal{G} = (G, \sim^V) \leadsto \overline{G} = (V, V_0, (E)_{a \in A}, W_0) \]
Synchronous Case

\[ \mathcal{G} = (G, \sim^V) \sim \bar{G} = (\bar{V}, \bar{V}_0, (\bar{E})_{a \in A}, \bar{W}_0) \]

- Positions in $\bar{V}$ are subsets of $\sim^V$-classes.
Synchronous Case

\[ G = (G, \sim^V) \mapsto \overline{G} = (\overline{V}, \overline{V}_0, (\overline{E})_{a \in A}, \overline{W}_0) \]

- Positions in \( \overline{V} \) are subsets of \( \sim^V \)-classes.

- The set of \( \overline{E} \)-successors of \( \overline{u} \) is obtained from the set of all successors of positions in \( \overline{u} \), divided by \( \sim^V \).
Synchronous Case

\[ G = (G, \sim^V) \sim \overline{G} = (\overline{V}, \overline{V_0}, (E)_{a \in A}, \overline{W_0}) \]

- Positions in \( \overline{V} \) are subsets of \( \sim^V \)-classes.

- The set of \( \overline{E} \)-successors of \( \overline{u} \) is obtained from the set of all successors of positions in \( \overline{u} \), divided by \( \sim^V \).

- \( \overline{W_0} \) : for parity conditions with observable colors, let \( \text{col}(\overline{u}) = \text{col}(u) \) for any \( u \in \overline{u} \).
Synchronous Case

\[ G = (G, \sim^V) \rightsquigarrow \overline{G} = (\overline{V}, \overline{V}_0, (\overline{E})_{a \in A}, \overline{W}_0) \]

- Positions in \( \overline{V} \) are subsets of \( \sim^V \)-classes.

- The set of \( \overline{E} \)-successors of \( \overline{u} \) is obtained from the set of all successors of positions in \( \overline{u} \), divided by \( \sim^V \).

- \( \overline{W}_0 \): for parity conditions with observable colors, let \( \text{col}(\overline{u}) = \text{col}(u) \) for any \( u \in \overline{u} \).

- Arbitrary winning conditions?
Synchronous Case

\[ G = (G, \sim^V) \leadsto \overline{G} = (\overline{V}, \overline{V}_0, (\overline{E})_{a \in A}, \overline{W}_0) \]

- Positions in \( \overline{V} \) are subsets of \( \sim^V \)-classes.
- The set of \( \overline{E} \)-successors of \( \overline{u} \) is obtained from the set of all successors of positions in \( \overline{u} \), divided by \( \sim^V \).
- \( \overline{W}_0 \) : for parity conditions with observable colors, let \( \text{col}(\overline{u}) = \text{col}(u) \) for any \( u \in \overline{u} \).
- Arbitrary winning conditions?
- Let \( \overline{u}_1 \overline{u}_2 \ldots \in \overline{W}_0 \) \iff \ldots
Synchronous Case

\[ G = (G, \sim^V) \sim \bar{G} = (V, V_0, (E)_{a\in A}, W_0) \]

- Positions in \( \bar{V} \) are subsets of \( \sim^V \)-classes.

- The set of \( \bar{E} \)-successors of \( \bar{u} \) is obtained from the set of all successors of positions in \( \bar{u} \), divided by \( \sim^V \).

- \( \bar{W}_0 \) : for parity conditions with observable colors, let \( \text{col}(\bar{u}) = \text{col}(u) \) for any \( u \in \bar{u} \).

- Arbitrary winning conditions?

- Let \( \bar{u}_1 \bar{u}_2 \ldots \in \bar{W}_0 : \iff \forall u_1u_2\ldots \in V^\omega : [ u_i \in \bar{u}_i \ \forall i ] \implies u_1u_2\ldots \in W_0. \)
Synchronous Case

- For parity conditions with observable colors, this is equivalent to coloring the positions in $\overline{G}$. 
Synchronous Case

- For parity conditions with observable colors, this is equivalent to coloring the positions in $\overline{G}$.

- Also true for more general notions of observable winning conditions.
For parity conditions with observable colors, this is equivalent to coloring the positions in $G$.

Also true for more general notions of observable winning conditions.

For arbitrary $\omega$-regular winning conditions?
For parity conditions with observable colors, this is equivalent to coloring the positions in $G$.

Also true for more general notions of observable winning conditions.

For arbitrary $\omega$-regular winning conditions?

For arbitrary $\omega$-regular winning conditions?

$\bar{u}_1 \bar{u}_2 \ldots \not\in \bar{W}_0 \iff \bar{u}_1 \bar{u}_2 \ldots \not\in \bar{W}_0$
Synchronous Case

- For parity conditions with observable colors, this is equivalent to coloring the positions in $\overline{G}$.
- Also true for more general notions of observable winning conditions.
- For arbitrary $\omega$-regular winning conditions?

\[ \overline{u_1 u_2 \ldots} \not\in \overline{W}_0 \iff \exists u_1 u_2 \ldots \in V^\omega \setminus W_0 : u_i \in \overline{u_i} \ \forall i \]
For parity conditions with observable colors, this is equivalent to coloring the positions in $G$.

Also true for more general notions of observable winning conditions.

For arbitrary $\omega$-regular winning conditions?

$$\overline{u_1 u_2 \ldots} \notin \overline{W_0} \iff \exists \, u_1 u_2 \ldots \in V^\omega \setminus W_0 : u_i \in \overline{u_i} \, \forall i$$

Given a Büchi automaton $\mathcal{B}$ with $L(\mathcal{B}) = W_0$, one can construct a Büchi automaton $\overline{\mathcal{B}}$ with $L(\overline{\mathcal{B}}) = \overline{W_0}$.

($\omega$-regular languages are closed under complementation.)
Theorem

- The strategy problem for \( \omega \)-regular games with partial information is decidable.

- Finite memory strategies can be synthesized.
Asynchronous Case

\[ \mathcal{G} = (G, \sim^V) \leadsto \tilde{G} = (\tilde{V}, \tilde{V}_0, (\tilde{E})_{a \in A}, \tilde{W}_0) \]
Asynchronous Case

\[ \mathcal{G} = (G, \sim^V) \rightarrow \bar{G} = (\bar{V}, \bar{V}_0, (\bar{E})_{a \in A}, \bar{W}_0) \]

- Positions in \( \bar{V} \) are subsets of \( \sim^V \)-classes.
Asynchronous Case

\[ \mathcal{G} = (G, \sim^V) \mapsto \tilde{\mathcal{G}} = (\tilde{V}, \tilde{V}_0, (\tilde{E})_{a \in A}, \tilde{W}_0) \]

- Positions in \( \tilde{V} \) are subsets of \( \sim^V \)-classes.

- We call a position \( \nu \) an extended successor of a position \( \mu \), if \( \nu \) is reachable from a successor \( \mu' \) of \( \mu \) via a sequence of moves which are hidden from player 0.
Asynchronous Case

\[ G = (G, \sim^V) \sim \tilde{G} = (\tilde{V}, \tilde{V}_0, (\tilde{E})_{a \in A}, \tilde{W}_0) \]

- Positions in \( \tilde{V} \) are subsets of \( \sim^V \)-classes.
- We call a position \( v \) an extended successor of a position \( u \), if \( v \) is reachable from a successor \( u' \) of \( u \) via a sequence of moves which are hidden from player 0.
- The set of \( \tilde{E} \)-successors of \( \tilde{u} \) is obtained from the set of all extended successors of positions in \( \tilde{u} \), divided by \( \sim^V \).
Asynchronous Case

\[ G = (G, \sim^V) \leadsto \tilde{G} = (\tilde{V}, \tilde{V}_0, (\tilde{E})_{a \in A}, \tilde{W}_0) \]

- Positions in \( \tilde{V} \) are subsets of \( \sim^V \)-classes.

- We call a position \( v \) an extended successor of a position \( u \), if \( v \) is reachable from a successor \( u' \) of \( u \) via a sequence of moves which are hidden from player 0.

- The set of \( \tilde{E} \)-successors of \( \tilde{u} \) is obtained from the set of all extended successors of positions in \( \tilde{u} \), divided by \( \sim^V \).

- \( \tilde{W}_0 \) : for parity conditions with observable colors, let \( \text{col}(\tilde{u}) = \text{col}(u) \) for any \( u \in \tilde{u} \).
Asynchronous Case

\[ \mathcal{G} = (G, \sim^V) \leadsto \tilde{\mathcal{G}} = (\tilde{V}, \tilde{V}_0, (\tilde{E})_{a \in A}, \tilde{W}_0) \]

- Positions in \( \tilde{V} \) are subsets of \( \sim^V \)-classes.

- We call a position \( v \) an extended successor of a position \( u \), if \( v \) is reachable from a successor \( u' \) of \( u \) via a sequence of moves which are hidden from player 0.

- The set of \( \tilde{E} \)-successors of \( \tilde{u} \) is obtained from the set of all extended successors of positions in \( \tilde{u} \), divided by \( \sim^V \).

- \( \tilde{W}_0 \) : for parity conditions with observable colors, let \( \text{col}(\tilde{u}) = \text{col}(u) \) for any \( u \in \tilde{u} \).

- Arbitrary winning conditions?
Asynchronous Case

Let $\tilde{u}_1\tilde{u}_2\ldots \in \tilde{W}_0$:
Asynchronous Case

Let $\tilde{u}_1\tilde{u}_2\ldots \in \tilde{W}_0 : \iff$

$\forall u_1u_2\ldots \in V^\omega :$
Asynchronous Case

Let $\tilde{u}_1 \tilde{u}_2 \ldots \in \tilde{W}_0 : \iff \\
\forall u_1 u_2 \ldots \in V^\omega : \\
[ \exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_{i+1}-1} \in \tilde{u}_i \ \forall \ i \ \text{and} \ \ k_{i+1} - k_i = 1 \text{ if } \tilde{u}_i \in \tilde{V}_0 ]
Asynchronous Case

Let $\tilde{u}_1 \tilde{u}_2 \ldots \in \tilde{W}_0 : \iff 
\forall u_1 u_2 \ldots \in V^\omega :
\left[ \exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_{i+1}-1} \in \tilde{u}_i \ \forall \ i \ \text{and} \ k_{i+1} - k_i = 1 \ \text{if} \ \tilde{u}_i \in \tilde{V}_0 \right] 
\implies u_1 u_2 \ldots \in W_0
Let $\tilde{u}_1\tilde{u}_2\ldots \in \tilde{W}_0 \iff \\
\forall u_1u_2\ldots \in V^\omega :\\n[ \exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_{i+1}-1} \in \tilde{u}_i \forall i \text{ and } \\
k_{i+1} - k_i = 1 \text{ if } \tilde{u}_i \in \tilde{V}_0 ]\\n\implies u_1u_2\ldots \in W_0}$

- For parity conditions with observable colors, this is equivalent to coloring the positions in $\overline{G}$.
- Also true for more general notions of observable winning conditions.
Asynchronous Case

Let $\tilde{u}_1\tilde{u}_2\ldots \in \tilde{W}_0 \iff \
\forall \ u_1u_2\ldots \in V^\omega : \
\exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_{i+1}-1} \in \tilde{u}_i \forall i \text{ and } k_{i+1} - k_i = 1 \text{ if } \tilde{u}_i \in \tilde{V}_0 \] 
$u_1u_2\ldots \in W_0$

- For parity conditions with observable colors, this is equivalent to coloring the positions in $\overline{G}$.
- Also true for more general notions of observable winning conditions.
- For arbitrary $\omega$-regular winning conditions?
Asynchronous Case

\[ \overline{u_1 u_2 \ldots} \notin \overline{W_0} \iff \]
Asynchronous Case

\[
\bar{u}_1 \bar{u}_2 \ldots \not\in \overline{W}_0 \iff \\
\exists u_1 u_2 \ldots \in V^\omega \setminus W_0
\]
Asynchronous Case

\[ \bar{u}_1 \bar{u}_2 \ldots \notin \overline{W}_0 \iff \exists u_1 u_2 \ldots \in V^\omega \setminus W_0 \]

\[
[ \exists 0 := k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_{i+1}-1} \in \tilde{u}_i \forall i \]

and \( k_{i+1} - k_i = 1 \) if \( \tilde{u}_i \in \tilde{V}_0 \) ]
Asynchronous Case

\[ \overline{u_1 u_2 \ldots} \notin \overline{W}_0 \iff \exists u_1 u_2 \ldots \in V^\omega \setminus W_0 \]
\[ \exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_i+1-1} \in \tilde{u}_i \forall i \]
\[ \text{and } k_{i+1} - k_i = 1 \text{ if } \tilde{u}_i \in \tilde{V}_0 \]

• Given a Büchi automaton \( B \) with \( L(B) = W_0 \), one can construct a Büchi automaton \( \overline{B} \) with \( L(\overline{B}) = \overline{W}_0 \).
Asynchronous Case

- $\overline{u_1 u_2 \ldots} \notin \overline{W_0} \iff \exists u_1 u_2 \ldots \in V^\omega \setminus W_0$

  $[ \exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_i+1-1} \in \tilde{u}_i \forall i$

and $k_{i+1} - k_i = 1$ if $\tilde{u}_i \in \tilde{V}_0 ]$

- Given a Büchi automaton $B$ with $L(B) = W_0$, one can construct a Büchi automaton $\overline{B}$ with $L(\overline{B}) = \overline{W_0}$.

- In the synchronous case, from a given S1S-formula $\varphi$ with $L(\varphi) = W_0$, one can construct an S1S-formula $\overline{\varphi}$ with $L(\overline{\varphi}) = \overline{W_0}$ directly.
Asynchronous Case

- \( \overline{u_1 u_2 \ldots} \notin \overline{W_0} \iff \exists u_1 u_2 \ldots \in V^\omega \setminus W_0 \)

\[ \exists 0 =: k_0 < k_1 < k_2 < \ldots \text{ with } u_{k_i}, \ldots, u_{k_i+1-1} \in \overline{u}_i \forall i \]

and \( k_{i+1} - k_i = 1 \) if \( \overline{u}_i \in \overline{V}_0 \]

- Given a Büchi automaton \( \mathcal{B} \) with \( L(\mathcal{B}) = W_0 \), one can construct a Büchi automaton \( \overline{\mathcal{B}} \) with \( L(\overline{\mathcal{B}}) = \overline{W}_0 \).

- In the synchronous case, from a given S1S-formula \( \varphi \) with \( L(\varphi) = W_0 \), one can construct an S1S-formula \( \overline{\varphi} \) with \( L(\overline{\varphi}) = \overline{W}_0 \) directly.

- In the asynchronous case?
Asynchronous Case

Theorem

- The asynchronous strategy problem for ω-regular games with partial information is decidable.
- Finite memory strategies can be synthesized.
<table>
<thead>
<tr>
<th>The Model</th>
<th>Powerset Construction</th>
<th>Finite Memory</th>
<th>Alternating Tree Automata</th>
<th>Future Prospects</th>
</tr>
</thead>
</table>

**First Lower Bound**
First Lower Bound
First Lower Bound
First Lower Bound
First Lower Bound
First Lower Bound

$G_n$: The number of positions and the time bound are linear in $n$. 
First Lower Bound

$G_n$: 
- The number of positions and the time bound are linear in $n$.
- Player 0 has a winning strategy which uses $2^n - 1$ memory states.
First Lower Bound

$G_n$:

- The number of positions and the time bound are linear in $n$.
- Player 0 has a winning strategy which uses $2^n - 1$ memory states.
- Player 0 does not have a winning strategy which uses at most $2^n - 2$ memory states.
The number of positions and the time bound are linear in $n$.

Player 0 has a winning strategy which uses $2^n - 1$ memory states.

Player 0 does not have a winning strategy which uses at most $2^n - 2$ memory states.

Player 0 has a memoryless winning strategy for the underlying game with full information.
First Lower Bound

$G_n$:

- The number of positions and the time bound are linear in $n$.
- Player 0 has a winning strategy which uses $2^n - 1$ memory states.
- Player 0 does not have a winning strategy which uses at most $2^n - 2$ memory states.
- Player 0 has a memoryless winning strategy for the underlying game with full information.

However:

- There are $O(n!)$ many actions in the game.
First Lower Bound

$G_n$:

- The number of positions and the time bound are linear in $n$.
- Player 0 has a winning strategy which uses $2^n - 1$ memory states.
- Player 0 does not have a winning strategy which uses at most $2^n - 2$ memory states.
- Player 0 has a memoryless winning strategy for the underlying game with full information.

However:

- There are $O(n!)$ many actions in the game.
- It is not a reachability game.
First Lower Bound

\( G_n: \)

- The number of positions and the time bound are linear in \( n \).
- Player 0 has a winning strategy which uses \( 2^n - 1 \) memory states.
- Player 0 does not have a winning strategy which uses at most \( 2^n - 2 \) memory states.
- Player 0 has a memoryless winning strategy for the underlying game with full information.

However:

- There are \( O(n!) \) many actions in the game.
- It is not a reachability game.

\[ \sim 2^{\sqrt[3]{n}} \]
<table>
<thead>
<tr>
<th>The Model</th>
<th>Powerset Construction</th>
<th>Finite Memory</th>
<th>Alternating Tree Automata</th>
<th>Future Prospects</th>
</tr>
</thead>
</table>

**Second Lower Bound (Berwanger et al.)**
Second Lower Bound (Berwanger et al.)
Second Lower Bound (Berwanger et al.)
Nondeterministic Tree-Automata

Nonemptiness for nondeterministic tree automaton $A$:
Nonemptiness for nondeterministic tree automaton $\mathcal{A}$:

\[ L(\mathcal{A}) \neq \emptyset \iff \exists \text{ player } \exists \text{ wins the game.} \]
Nondeterministic Tree-Automata

Nonemptiness for nondeterministic tree automaton $\mathcal{A}$:

\[ L(\mathcal{A}) \neq \emptyset \iff \exists \text{ tree } t \exists \text{ run } \rho \text{ of } \mathcal{A} \text{ on } t : \]

all infinite paths through $\rho$ are accepting.
Nonemptiness for nondeterministic tree automaton $\mathcal{A}$:

$L(\mathcal{A}) \neq \emptyset \iff \exists \text{ tree } t \quad \exists \text{ run } \rho \text{ of } \mathcal{A} \text{ on } t$

all infinite paths through $\rho$ are accepting.

Game:
Nondeterministic Tree-Automata

Nonemptiness for nondeterministic tree automaton $A$:

$L(A) \neq \emptyset \iff \exists \text{ tree } t \exists \text{ run } \rho \text{ of } A \text{ on } t :$

all infinite paths through $\rho$ are accepting.

Game:

- Player $\exists$ : Chooses tree and run
  (by choosing transitions)
Nondeterministic Tree-Automata

Nonemptiness for nondeterministic tree automaton $\mathcal{A}$:

$L(\mathcal{A}) \neq \emptyset \iff \exists \text{ tree } t \exists \text{ run } \rho \text{ of } \mathcal{A} \text{ on } t :
\text{ all infinite paths through } \rho \text{ are accepting.}$

Game:

- Player $\exists$ : Chooses tree and run
  (by choosing transitions)
- Player $\forall$ : Chooses path
  (by choosing directions in the tree = directions in the run)
Nondeterministic Tree-Automata

Nonemptiness for nondeterministic tree automaton $A$:

$L(A) \neq \emptyset \iff \exists$ tree $t \exists$ run $\rho$ of $A$ on $t$:
all infinite paths through $\rho$ are accepting.

Game:

- Player $\exists$: Chooses tree and run
  (by choosing transitions)
- Player $\forall$: Chooses path
  (by choosing directions in the tree = directions in the run)

$L(A) \neq \emptyset \iff$ Player $\exists$ wins the game.
Alternating tree automaton:

- Directions in the input tree $\neq$ directions in the run.
  (Several directions in the run may correspond to one direction in the tree.)
Alternating tree automaton:

- Directions in the input tree ≠ directions in the run. (Several directions in the run may correspond to one direction in the tree.)

- Labelling of the input tree may depend on the directions of the input tree that ∀ chooses but it must not depend on the directions of the run that ∀ chooses.
Alternating tree automaton:
- Directions in the input tree $\neq$ directions in the run.
  (Several directions in the run may correspond to one direction in the tree.)
- Labelling of the input tree may depend on the directions of the input tree that $\forall$ chooses but it must not depend on the directions of the run that $\forall$ chooses.

Idea:
Split $\exists$ into players $T$, guessing the tree and $A$, guessing the run of the automaton.
Alternating tree automaton:

- Directions in the input tree $\neq$ directions in the run. (Several directions in the run may correspond to one direction in the tree.)

- Labelling of the input tree may depend on the directions of the input tree that $\forall$ chooses but it must not depend on the directions of the run that $\forall$ chooses.

Idea:
Split $\exists$ into players $T$, guessing the tree and $A$, guessing the run of the automaton.

$\sim$ Three player game with partial information.
Players $\forall$ and $A$ have full information
Players $\forall$ and $A$ have full information.

Player $T$ sees only the branches of the input tree which are chosen.
From Automata to Games

- Players $\forall$ and $A$ have full information
- Player $T$ sees only the branches of the input tree which are chosen

$L(A) \neq \emptyset$ if and only if $T$ and $A$ can cooperate to win.
From Automata to Games

- Players $\forall$ and $A$ have full information
- Player $T$ sees only the branches of the input tree which are chosen

$$L(A) \neq \emptyset$$ if and only if $T$ and $A$ can cooperate to win.

If $A$ is universal, then the game is a two-player game with partial information!
From Games to Automata

Problem:
Given three-player game with partial information where only player 0 has partial information, position $v$, can player 0 and 1 cooperate to win from $v$?
Problem:
Given three-player game with partial information where only player 0 has partial information, position $v$, can player 0 and 1 cooperate to win from $v$?

(1) Construct nondeterministic tree automaton such that a tree is accepted $\iff$
Problem:
Given three-player game with partial information where only player 0 has partial information, position $v$, can player 0 and 1 cooperate to win from $v$?

1. Construct nondeterministic tree automaton such that a tree is accepted $\iff$ it is the unravelling of the game graph from $v$
From Games to Automata

Problem:
Given three-player game with partial information where only player 0 has partial information, position \( v \), can player 0 and 1 cooperate to win from \( v \)?

(1) Construct nondeterministic tree automaton such that a tree is accepted \( \iff \)
- it is the unravelling of the game graph from \( v \)
- the labellings at the positions of player 0 define a full information strategy \( f \) for player 0
From Games to Automata

Problem:
Given three-player game with partial information where only player 0 has partial information, position \( v \), can player 0 and 1 cooperate to win from \( v \)?

(1) Construct nondeterministic tree automaton such that a tree is accepted \( \iff \)
- it is the unravelling of the game graph from \( v \)
- the labellings at the positions of player 0 define a full information strategy \( f \) for player 0
- there is a strategy \( g \) for player 1
From Games to Automata

Problem:
Given three-player game with partial information where only player 0 has partial information, position $v$, can player 0 and 1 cooperate to win from $v$?

(1) Construct nondeterministic tree automaton such that a tree is accepted $\iff$

- it is the unravelling of the game graph from $v$
- the labellings at the positions of player 0 define a full information strategy $f$ for player 0
- there is a strategy $g$ for player 1
- the composition of $f$ and $g$ is winning.
From Games to Automata

Problem:
Given three-player game with partial information where only player 0 has partial information, position \( v \), can player 0 and 1 cooperate to win from \( v \)?

(1) Construct nondeterministic tree automaton such that a tree is accepted if and only if it is the unravelling of the game graph from \( v \)
- the labellings at the positions of player 0 define a full information strategy \( f \) for player 0
- there is a strategy \( g \) for player 1
- the composition of \( f \) and \( g \) is winning.

(2) Restrict the strategies of player 0 to information based strategies.
From Games to Automata

Technique for (2):

“Narrowing”
(Kupferman, Vardi: “Church’s Problem Revisited”. ’99)
From Games to Automata

Technique for (2):

“Narrowing”
(Kupferman, Vardi: “Church’s Problem Revisited”. (’99))

If the game is a two-player game:
From Games to Automata

Technique for (2):

“Narrowing”
(Kupferman, Vardi: “Church’s Problem Revisited”. (’99))

If the game is a two-player game:
- The automaton from the first step is deterministic.
From Games to Automata

Technique for (2):

“Narrowing”
(Kupferman, Vardi: “Church’s Problem Revisited”. (’99))

If the game is a two-player game:

- The automaton from the first step is deterministic.
- The “narrowing” of a deterministic automaton is universal.
Future Work

- Stochastic Games
Future Work

- Stochastic Games
  - Stochastic Moves
Future Work

- **Stochastic Games**
  - Stochastic Moves
  - Randomized Strategies
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies
- Efficient Algorithms for Interesting Classes of Games
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies
- Efficient Algorithms for Interesting Classes of Games
- Generalization of $\sim_i$ and $\overline{\sim}_i$
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies
- Efficient Algorithms for Interesting Classes of Games
- Generalization of $\sim_i$ and $\preceq_i$
  - Automata over Relations
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies

- Efficient Algorithms for Interesting Classes of Games

- Generalization of $\sim_i$ and $\bar{\sim}_i$
  - Automata over Relations
  - Logical Formulas
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies
- Efficient Algorithms for Interesting Classes of Games
- Generalization of $\sim_i$ and $\leftarrow_i$
  - Automata over Relations
  - Logical Formulas
- Connection to Logic
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies

- Efficient Algorithms for Interesting Classes of Games

- Generalization of $\sim_i$ and $\overleftarrow{\sim}_i$
  - Automata over Relations
  - Logical Formulas

- Connection to Logic
  - Dynamic/Temporal Process/Epistemic Logic
Future Work

- Stochastic Games
  - Stochastic Moves
  - Randomized Strategies

- Efficient Algorithms for Interesting Classes of Games

- Generalization of $\sim_i$ and $\widetilde{\sim}_i$
  - Automata over Relations
  - Logical Formulas

- Connection to Logic
  - Dynamic/Temporal Process/Epistemic Logic
  - IF-Logic, Dependence Logic, ...