Winning Strategies in Two-Player Games with Partial Information

LINT Workshop, Amsterdam 04. Dec. - 06. Dec. 2008

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 ω 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 ω 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 ω 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 $f : V^*V_i \to A$ with $f(\pi v_i) \in \mathrm{act}(v_i)$,

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

Strategies

- Strategy for player i:

  Function $f : V^* V_i \to A$ with $f(\pi v_i) \in \mathrm{act}(v_i)$,

  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 $f : V^*V_i \to A$ with $f(\pi v_i) \in \mathrm{act}(v_i)$,

  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 $f : V^*V_i \to A$ with $f(\pi v_i) \in \mathrm{act}(v_i)$,

  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 π has been played and after π' has been played, player i has exactly the same information.

Strategies

- Strategy for player i:

  Function $f : V^*V_i \to A$ with $f(\pi v_i) \in \mathrm{act}(v_i)$,

  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 π has been played and after π' has been played, player i has exactly the same information.

$$\pi \sim_i \pi' \Longrightarrow f(\pi) = f(\pi')$$

Knowledge Representation

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

Knowledge Representation

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

- we would like to impose certain natural restrictions on $\sim_i$.

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 $\sim_i$.

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, $\mathrm{S1S}$, 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, $\mathrm{S1S}$, parity conditions, . . .
- Finite representation of knowledge:

Knowledge Representation

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

Knowledge Representation

1^{st} 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

1^{st} 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) $v \sim_i w \Longrightarrow v, w \in V_i$ or $v, w \notin V_i$

Knowledge Representation

1^{st} 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) $v \sim_i w \Longrightarrow v, w \in V_i$ or $v, w \notin V_i$

(2) $v, w \in V_i$ with $v \sim_i w$, $a \neq b \Longrightarrow f_a(v) \not\sim_i f_b(w)$

Knowledge Representation

1^{st} 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 $v \sim_i w$, $a \neq b \implies f_a(v) \not\sim_i f_b(w)$

(3) $v, w \in V_i$ with $v \sim_i w \implies \mathrm{act}(v) = \mathrm{act}(w)$

Knowledge Representation

1^{st} 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) $v \sim_i w \Longrightarrow v, w \in V_i$ or $v, w \notin V_i$

(2) $v, w \in V_i$ with $v \sim_i w$, $a \neq b \Longrightarrow f_a(v) \not\sim_i f_b(w)$

(3) $v, w \in V_i$ with $v \sim_i w \Longrightarrow \mathrm{act}(v) = \mathrm{act}(w)$

2^{nd} Extend $\sim_i^V$ to $\sim_i$.

Knowledge Representation

- $\sim$ If player i does observe any move, then

Knowledge Representation

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

Knowledge Representation

- If player i does observe any move, then
  $\pi \sim_i \pi'$ iff $|\pi| = |\pi'|$ and $\pi(j) \sim_i^V \pi'(j)$ 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

- $\sim$ If player i does observe any move, then
  $\pi \sim_i \pi'$ iff $|\pi| = |\pi'|$ and $\pi(j) \sim_i^V \pi'(j)$ for all j.
  (Synchronous case, player share a clock.)
- $\overleftarrow{\sim}$ Now, hide moves from player i in which he can't observe anything that happens:
  $\pi \overleftarrow{\sim}_i \pi'$ iff $\overleftarrow{\pi} \sim_i \overleftarrow{\pi}'$ where
  $\overleftarrow{\pi}$ is obtained from π by deleting all moves $u \to v$ from π such that $u \in V_{1-i}$ and $u \sim_i^V v$.
  (Asynchronous case.)

The Question

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

The Question

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

The Question

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

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

$$\rightsquigarrow \mathcal{G} = (G, \sim^V)$$

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.

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.

$\rightsquigarrow$ Powerset Construction

Synchronous Case

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

Synchronous Case

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

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

Synchronous Case

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

- 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 $\sim^V$.

Synchronous Case

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

- 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 $\mathrm{col}(\overline{u}) = \mathrm{col}(u)$ for any $u \in \overline{u}$.

Synchronous Case

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

- 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 $\mathrm{col}(\overline{u}) = \mathrm{col}(u)$ for any $u \in \overline{u}$.
- Arbitrary winning conditions?

Synchronous Case

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

- 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 $\mathrm{col}(\overline{u}) = \mathrm{col}(u)$ for any $u \in \overline{u}$.
- Arbitrary winning conditions?
- Let $\overline{u}_1\overline{u}_2 \ldots \in \overline{W}_0 :\Longleftrightarrow$

Synchronous Case

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

- 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 $\mathrm{col}(\overline{u}) = \mathrm{col}(u)$ for any $u \in \overline{u}$.
- Arbitrary winning conditions?
- Let $\overline{u}_1 \overline{u}_2 \ldots \in \overline{W}_0 :\Longleftrightarrow$ $\forall\ u_1 u_2 \ldots \in V^\omega \ : \ [\ u_i \in \overline{u}_i\ \forall i\ ] \Longrightarrow u_1 u_2 \ldots \in W_0$.

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.

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 ω-regular winning conditions?

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 ω-regular winning conditions?
- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

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 ω-regular winning conditions?
- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0 :\ u_i \in \overline{u}_i\ \forall i$

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 ω-regular winning conditions?
- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0 \,:\, u_i \in \overline{u}_i\ \forall i$
- 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.)

Synchronous Case

Theorem

- *The strategy problem for ω-regular games with partial information is decidable.*
- *Finite memory strategies can be synthesized.*

Asynchronous Case

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

Asynchronous Case

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

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

Asynchronous Case

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

- 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.

Asynchronous Case

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

- 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

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

- 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 $\mathrm{col}(\tilde{u}) = \mathrm{col}(u)$ for any $u \in \tilde{u}$.

Asynchronous Case

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

- 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 $\mathrm{col}(\tilde{u}) = \mathrm{col}(u)$ for any $u \in \tilde{u}$.
- Arbitrary winning conditions?

Asynchronous Case

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

Asynchronous Case

Let $\tilde{u}_1\tilde{u}_2\ldots \in \tilde{W}_0 :\Longleftrightarrow$
$\forall\ u_1u_2\ldots \in V^\omega\quad :$

Asynchronous Case

Let $\tilde{u}_1\tilde{u}_2\ldots \in \tilde{W}_0 :\Longleftrightarrow$
$\forall\ u_1u_2\ldots \in V^\omega\quad :$
$[\ \exists\ 0 =: k_0 < k_1 < k_2 < \ldots$ 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

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

Asynchronous Case

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

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

- 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 ω-regular winning conditions?

Asynchronous Case

- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

Asynchronous Case

- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0$

Asynchronous Case

- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0$
  $[\ \exists\ 0 =: k_0 < k_1 < k_2 < \ldots$ 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

- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0$

  $[\ \exists\ 0 =: k_0 < k_1 < k_2 < \ldots$ 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 $\mathcal{B}$ with $L(\mathcal{B}) = W_0$, one can construct a Büchi automaton $\overline{\mathcal{B}}$ with $L(\overline{\mathcal{B}}) = \overline{W}_0$.

Asynchronous Case

- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0$

  $[\ \exists\ 0 =: k_0 < k_1 < k_2 < \ldots$ 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 $\mathcal{B}$ 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 φ with $L(\varphi) = W_0$, one can construct an S1S-formula $\overline{\varphi}$ with $L(\overline{\varphi}) = \overline{W}_0$ directly.

Asynchronous Case

- $\overline{u}_1\overline{u}_2\ldots \notin \overline{W}_0 \iff$

  $\exists\ u_1u_2\ldots \in V^\omega \setminus W_0$

  $[\ \exists\ 0 =: k_0 < k_1 < k_2 < \ldots$ 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 $\mathcal{B}$ 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 φ 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?

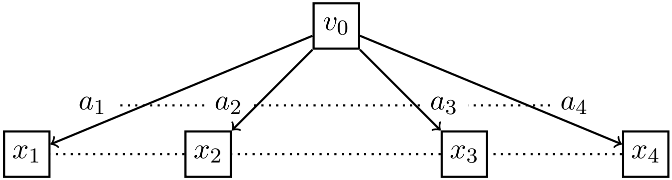
Asynchronous Case

Theorem

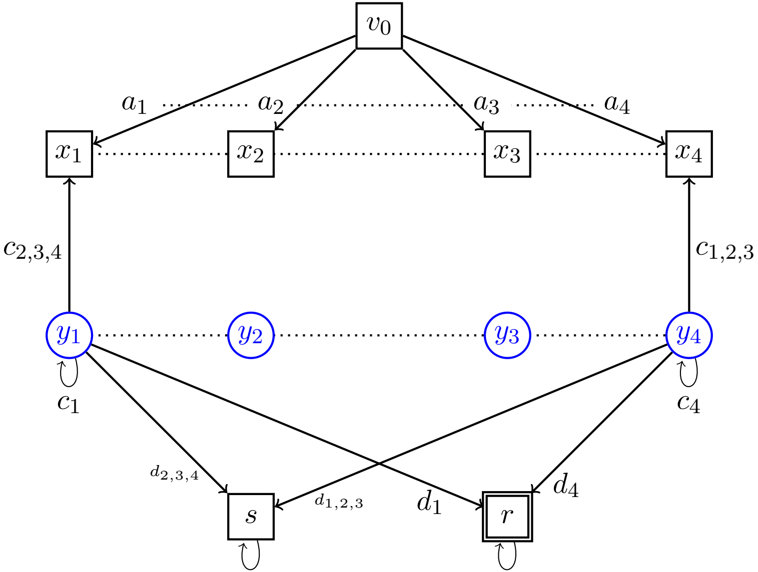
- *The asynchronous strategy problem for ω-regular games with partial information is decidable.*
- *Finite memory strategies can be synthesized.*

First Lower Bound

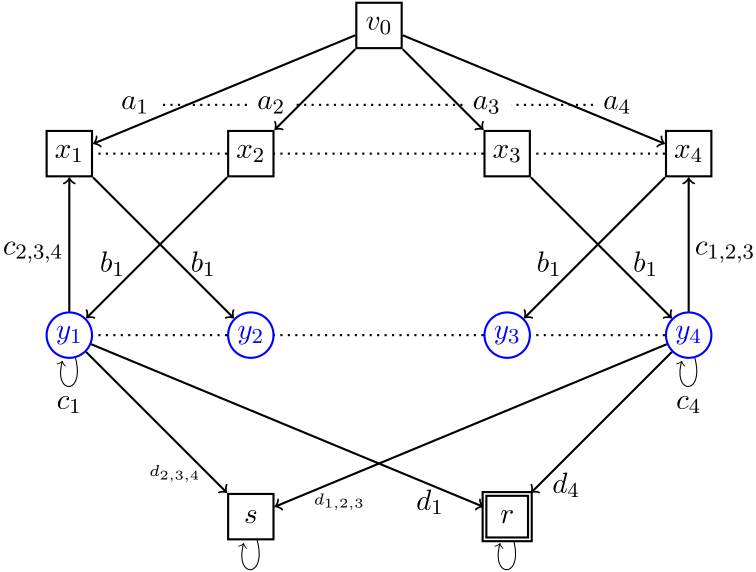
First Lower Bound



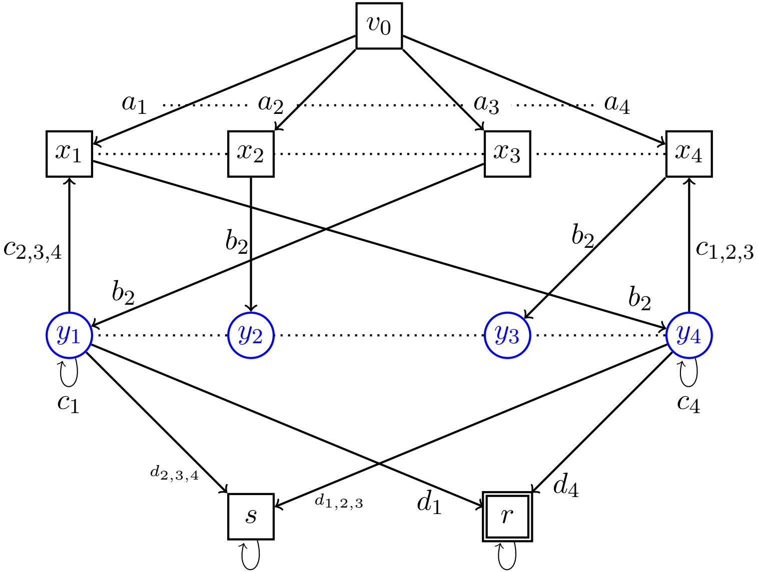
First Lower Bound



First Lower Bound



First Lower Bound



First Lower Bound

$\mathcal{G}_n$:

- The number of positions and the time bound are linear in n.

First Lower Bound

$\mathcal{G}_n$:

- 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

$\mathcal{G}_n$:

- 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.

First Lower Bound

$\mathcal{G}_n$:

- 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.

First Lower Bound

$\mathcal{G}_n$:

- 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

$\mathcal{G}_n$:

- 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

$\mathcal{G}_n$:

- 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.

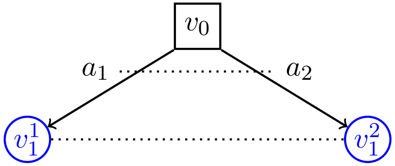
However:

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

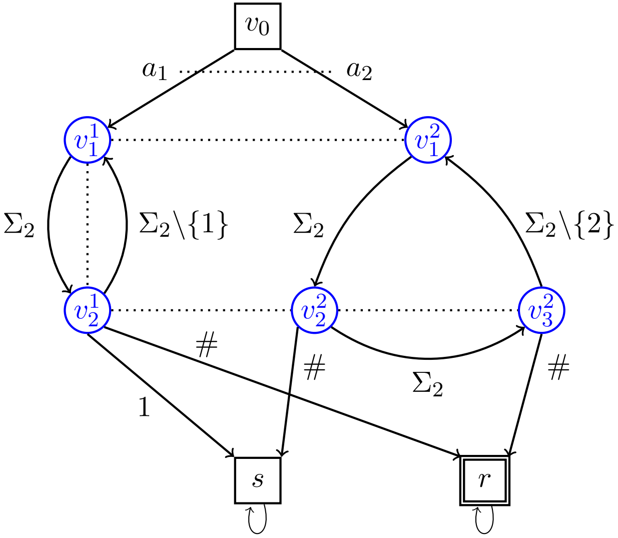
$$\rightsquigarrow 2^{\sqrt[3]{n}}$$

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 $\mathcal{A}$:

Nondeterministic Tree-Automata

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

$L(\mathcal{A}) \neq \emptyset \iff$

Nondeterministic Tree-Automata

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

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

Nondeterministic Tree-Automata

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

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

Game:

Nondeterministic Tree-Automata

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

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

Game:

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

Nondeterministic Tree-Automata

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

$L(\mathcal{A}) \neq \emptyset \iff$
$\exists$ tree t $\exists$ run ρ of $\mathcal{A}$ on t :
all infinite paths through ρ 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 $\mathcal{A}$:

$L(\mathcal{A}) \neq \emptyset \iff$
$\exists$ tree t $\exists$ run ρ of $\mathcal{A}$ on t :
all infinite paths through ρ 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)

$L(\mathcal{A}) \neq \emptyset \iff$ Player $\exists$ wins the game.

From Automata to Games

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.)

From Automata to Games

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.

From Automata to Games

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.

From Automata to Games

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.

$\leadsto$ Three player game with partial information.

From Automata to Games

- Players $\forall$ and A have full information

From Automata to Games

- 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

$L(\mathcal{A}) \neq \emptyset$ 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

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

If $\mathcal{A}$ 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?

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 $\iff$

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

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 $\Longleftrightarrow$
- 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 $\iff$
- 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.

(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 $\overleftarrow{\sim}_i$

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

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

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

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 $\overleftarrow{\sim}_i$
  - Automata over Relations
  - Logical Formulas
- Connection to Logic
  - Dynamic/Temporal Process/Epistemic Logic
  - IF-Logic, Dependence Logic, ...