Finite-state Strategies in Delay Games

Martin Zimmermann

Saarland University

September 21st, 2017

GandALF 2017, Rome, Italy

Motivation

Two goals:

1. Lift the notion of finite-state strategies to delay games.
2. Present uniform framework for solving delay games (which yields finite-state strategies whenever possible).

Motivation

Two goals:

1. Lift the notion of finite-state strategies to delay games.
2. Present uniform framework for solving delay games (which yields finite-state strategies whenever possible).

Questions:

- What are delay games?
- Why are finite-state strategies important?
- Why do we need a uniform framework?

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L \text{, if } \beta(i) = \alpha(i+2) \text{ for every } i$$

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I: b
O:

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I: b
O: a

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L \text{, if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a
O:	a	

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{rcc} I\!: & b & a \\ O\!: & a & a \end{array}$$

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{rccc} I: & b & a & b \\ O: & a & a & \end{array}$$

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$
O:	a	a	$\cdots$	

I wins

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$
O:	a	a	$\cdots$	

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$		I:	b
O:	a	a	$\cdots$			O:	

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$
O:	a	a	$\cdots$	

I:	b	a
O:		

I wins

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$		I:	b	a	b
O:	a	a	$\cdots$			O:			

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$	
O:	a	a	$\cdots$		

I:	b	a	b
O:	b		

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$		I:	b	a	b	b
O:	a	a	$\cdots$			O:	b			

I wins

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$
O:	a	a	$\cdots$	

I:	b	a	b	b
O:	b	b		

I wins

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$		I:	b	a	b	b	a
O:	a	a	$\cdots$			O:	b	b			

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$		I:	b	a	b	b	a
O:	a	a	$\cdots$			O:	b	b	a		

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$
O:	a	a	$\cdots$	

I:	b	a	b	b	a	b
O:	b	b	a			

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$
O:	a	a	$\cdots$	

I:	b	a	b	b	a	b
O:	b	b	a	b		

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)} \cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{lllll} I: & b & a & b & \cdots \\ O: & a & a & \cdots & \end{array} \qquad \begin{array}{llllllll} I: & b & a & b & b & a & b & a \\ O: & b & b & a & b & & & \end{array}$$

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$				I:	b	a	b	b	a	b	a
O:	a	a	$\cdots$					O:	b	b	a	b	a		

***I* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Delay Games

In this talk, a game is given by an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	$\cdots$			
O:	a	a	$\cdots$				

***I* wins**

I:	b	a	b	b	a	b	a	$\cdots$
O:	b	b	a	b	a	$\cdots$		

***O* wins**

In a delay game, Player O may delay her moves to gain a lookahead on Player I's moves.

Some History (1/2)

- **Hosch & Landweber ('72)**: ω-regular delay games with respect to constant delay solvable.

Some History (1/2)

- **Hosch & Landweber ('72)**: ω-regular delay games with respect to constant delay solvable.
- **Holtmann, Kaiser & Thomas ('10)**: Solving parity delay games in $2\textsc{ExpTime}$, doubly-exponential lookahead sufficient.

Some History (1/2)

- **Hosch & Landweber ('72)**: ω-regular delay games with respect to constant delay solvable.
- **Holtmann, Kaiser & Thomas ('10)**: Solving parity delay games in $2\textsc{ExpTime}$, doubly-exponential lookahead sufficient.
- **Fridman, Löding & Z. ('11)**: Nothing non-trivial is solvable for ω-contextfree delay games, unbounded lookahead necessary.

Some History (2/2)

- **Klein & Z. ('15)**: Solving parity delay games is ExpTime-complete, exponential lookahead sufficient and necessary.

Some History (2/2)

- **Klein & Z. ('15)**: Solving parity delay games is ExpTime-complete, exponential lookahead sufficient and necessary.
- **Z. ('15)**: Max-regular delay games with respect to constant delay solvable, unbounded lookahead necessary.

Some History (2/2)

- **Klein & Z. ('15)**: Solving parity delay games is ExpTime-complete, exponential lookahead sufficient and necessary.
- **Z. ('15)**: Max-regular delay games with respect to constant delay solvable, unbounded lookahead necessary.
- **Klein & Z. ('16)**: Solving LTL delay games is 3ExpTime-complete, triply-exponential lookahead sufficient and necessary.

Some History (2/2)

- **Klein & Z. ('15)**: Solving parity delay games is ExpTime-complete, exponential lookahead sufficient and necessary.
- **Z. ('15)**: Max-regular delay games with respect to constant delay solvable, unbounded lookahead necessary.
- **Klein & Z. ('16)**: Solving LTL delay games is 3ExpTime-complete, triply-exponential lookahead sufficient and necessary.
- **Z. ('17)**: Solving cost-parity delay games is ExpTime-complete, exponential lookahead sufficient and necessary.

Some History (2/2)

- **Klein & Z. ('15)**: Solving parity delay games is EXPTIME-complete, exponential lookahead sufficient and necessary.
- **Z. ('15)**: Max-regular delay games with respect to constant delay solvable, unbounded lookahead necessary.
- **Klein & Z. ('16)**: Solving LTL delay games is 3EXPTIME-complete, triply-exponential lookahead sufficient and necessary.
- **Z. ('17)**: Solving cost-parity delay games is EXPTIME-complete, exponential lookahead sufficient and necessary.

All recent (positive) results use variations of the same proof idea.

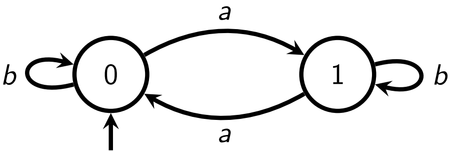
Finite-state Strategies

- A strategy in an infinite game is a map $\sigma\colon \Sigma_I^* \to \Sigma_O$, i.e., not necessarily finitely representable.

Finite-state Strategies

- A strategy in an infinite game is a map $\sigma\colon \Sigma_I^* \to \Sigma_O$, i.e., not necessarily finitely representable.
- A finite-state strategy is implemented by a finite automaton with output, and therefore finitely represented.

Example



$w \mapsto |w|_a \bmod 2$

Why Finite-state Strategies

Finite-state/positional strategies are crucial in many applications of infinite games, e.g.:

- In reactive synthesis, a finite-state winning strategy is a correct-by-construction controller.
- (Modern proofs of) Rabin's theorem rely on positional determinacy of parity games.
- In general, the existence of finite-state strategies enables the application of infinite games.
- Determining the memory requirements is one of the most fundamental tasks for a class of games.

Finite-state Strategies for Delay Games

Disclaimer: We focus here on constant delay!

- A strategy in a delay game is still a map $\sigma\colon \Sigma_I^* \to \Sigma_O$.
- So, the classical definition is still applicable.
- By "hardcoding" constant lookahead into the rules of the game, finite-state winning strategies are computable.

Finite-state Strategies for Delay Games

Disclaimer: We focus here on constant delay!

- A strategy in a delay game is still a map $\sigma \colon \Sigma_I^* \to \Sigma_O$.
- So, the classical definition is still applicable.
- By "hardcoding" constant lookahead into the rules of the game, finite-state winning strategies are computable.
- However, this notion does not distinguish "past" and "future".

A (Cautionary) Example

Example

$$L = \{ \binom{\alpha}{\alpha} \mid \alpha \in \{0,1\}^\omega \}$$

A (Cautionary) Example

Example

$$L = \{ \binom{\alpha}{\alpha} \mid \alpha \in \{0,1\}^\omega \}$$

$$
\begin{array}{lcccccccccccc}
\text{I:} & a & a & b & a & a & b & b & \overbrace{b \quad a \quad a \quad a \quad b}^{d} \\
\text{O:} & a & a & b & a & a & b & b & \\
\end{array}
$$

A (Cautionary) Example

Example

$$L = \{ \binom{\alpha}{\alpha} \mid \alpha \in \{0,1\}^\omega \}$$

$$\begin{array}{lcccccccccccc}
\text{I:} & a & a & b & a & a & b & b & \overbrace{b \quad a \quad a \quad a \quad b}^{d} \\
\text{O:} & a & a & b & a & a & b & b & b
\end{array}$$

A (Cautionary) Example

Example

$$L = \{ \binom{\alpha}{\alpha} \mid \alpha \in \{0,1\}^\omega \}$$

$$\begin{array}{lcccccccccccc} \text{I:} & a & a & b & a & a & b & b & \overbrace{b & a & a & a & b}^{d} \\ \text{O:} & a & a & b & a & a & b & b & b \end{array}$$

- Requires 2^d memory states with constant lookahead d.

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:

O:

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{rll} I\!: & b & a \\ O\!: & & \end{array}$$

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{lllll} I: & b & a & b & a \\ O: & & & & \end{array}$$

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	a
O:	b	a		

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{rcccccc} I: & b & a & b & a & b & a \\ O: & b & a & & & & \end{array}$$

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{lllllll} I\colon & b & a & b & a & b & a \\ O\colon & b & a & b & a & & \end{array}$$

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	a	b	a	b	b
O:	b	a	b	a				

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	a	b	a	b	b
O:	b	a	b	a	b	b		

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	a	b	a	b	b	a	b
O:	b	a	b	a	b	b				

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

I:	b	a	b	a	b	a	b	b	a	b
O:	b	a	b	a	b	b	a	b		

Block Games

Distinguishing between past and future: block games

- Fix a block length $d > 0$.
- Player I picks blocks $\overline{a_i} \in \Sigma_I^d$.
- Player O picks blocks $\overline{b_i} \in \Sigma_O^d$.
- Player O wins, if $\binom{\overline{a_0}\,\overline{a_1}\,\overline{a_2}\cdots}{\overline{b_0}\,\overline{b_1}\,\overline{b_2}\cdots} \in L$
- To account for (constant) lookahead, Player 1 is one move ahead.

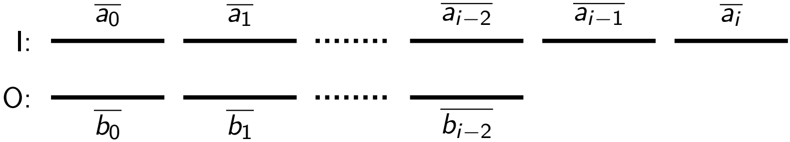
Example

$$\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\cdots \in L, \text{ if } \beta(i) = \alpha(i+2) \text{ for every } i$$

$$\begin{array}{lllllllllll}
I\colon & b & a & b & a & b & a & b & b & a & b & \cdots \\
O\colon & b & a & b & a & b & b & a & b & \cdots
\end{array}$$

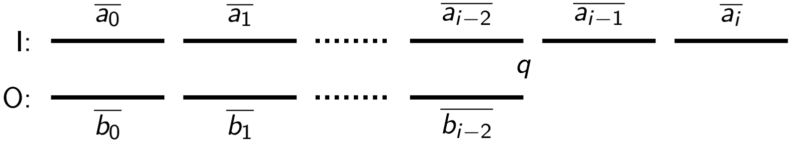
Finite-state Strategies for Block Games

- A finite-state strategy in a block game reads blocks over Σ_I and outputs blocks in Σ_O:



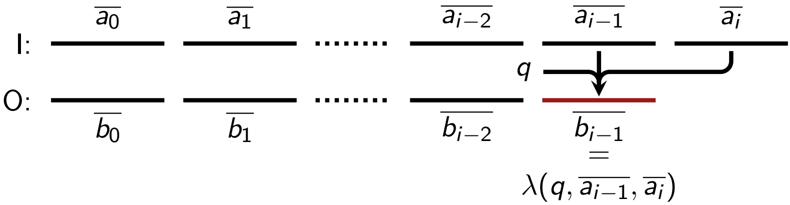
Finite-state Strategies for Block Games

- A finite-state strategy in a block game reads blocks over Σ_I and outputs blocks in Σ_O:



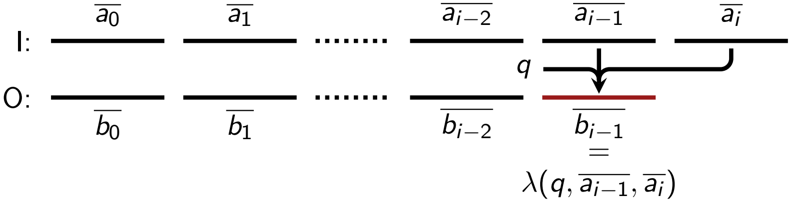
Finite-state Strategies for Block Games

- A finite-state strategy in a block game reads blocks over Σ_I and outputs blocks in Σ_O:



Finite-state Strategies for Block Games

- A finite-state strategy in a block game reads blocks over Σ_I and outputs blocks in Σ_O:



Note:

- Alphabet now exponential in block length!
- But, we distinguish past and future.
- In particular, state complexity only concerned with past.

Aggregations

- Fix ω-automaton $\mathfrak{A}$ and a finite set M.
- $s\colon Q^+ \to M$ is an aggregation for $\mathfrak{A}$, if for all runs $\rho = \pi_0\pi_1\pi_2\cdots$ and $\rho' = \pi_0'\pi_1'\pi_2'\cdots$ with $s(\pi_0)s(\pi_1)s(\pi_2)\cdots = s(\pi_0')s(\pi_1')s(\pi_2')\cdots$: ρ is accepting $\Leftrightarrow$ ρ' is accepting.

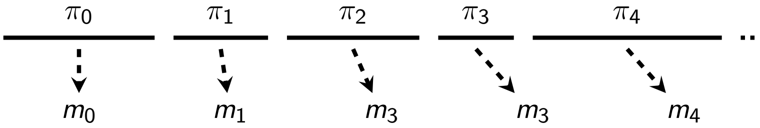
Aggregations

- Fix ω-automaton $\mathfrak{A}$ and a finite set M.
- $s\colon Q^+ \to M$ is an aggregation for $\mathfrak{A}$, if for all runs $\rho = \pi_0\pi_1\pi_2\cdots$ and $\rho' = \pi'_0\pi'_1\pi'_2\cdots$ with $s(\pi_0)s(\pi_1)s(\pi_2)\cdots = s(\pi'_0)s(\pi'_1)s(\pi'_2)\cdots$: ρ is accepting $\Leftrightarrow$ ρ' is accepting.

$$\pi_0 \qquad \pi_1 \qquad \pi_2 \qquad \pi_3 \qquad \pi_4 \qquad \cdots$$

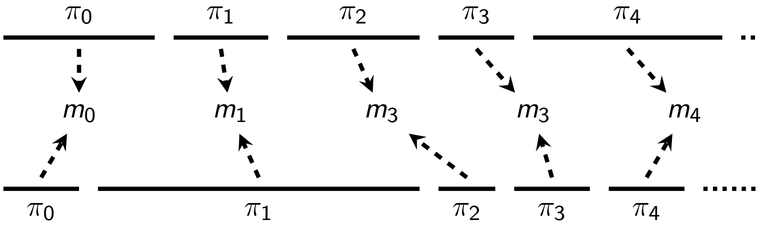
Aggregations

- Fix ω-automaton $\mathfrak{A}$ and a finite set M.
- $s\colon Q^+ \to M$ is an aggregation for $\mathfrak{A}$, if for all runs $\rho = \pi_0\pi_1\pi_2\cdots$ and $\rho' = \pi_0'\pi_1'\pi_2'\cdots$ with $s(\pi_0)s(\pi_1)s(\pi_2)\cdots = s(\pi_0')s(\pi_1')s(\pi_2')\cdots$: ρ is accepting $\Leftrightarrow$ ρ' is accepting.



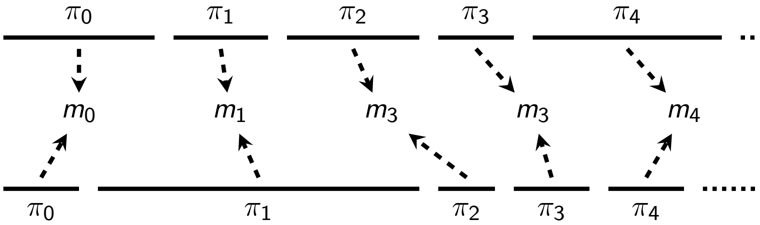
Aggregations

- Fix ω-automaton $\mathfrak{A}$ and a finite set M.
- $s\colon Q^+ \to M$ is an aggregation for $\mathfrak{A}$, if for all runs $\rho = \pi_0\pi_1\pi_2\cdots$ and $\rho' = \pi'_0\pi'_1\pi'_2\cdots$ with $s(\pi_0)s(\pi_1)s(\pi_2)\cdots = s(\pi'_0)s(\pi'_1)s(\pi'_2)\cdots$: ρ is accepting $\Leftrightarrow$ ρ' is accepting.



Aggregations

- Fix ω-automaton $\mathfrak{A}$ and a finite set M.
- $s\colon Q^+ \to M$ is an aggregation for $\mathfrak{A}$, if for all runs $\rho = \pi_0\pi_1\pi_2\cdots$ and $\rho' = \pi'_0\pi'_1\pi'_2\cdots$ with $s(\pi_0)s(\pi_1)s(\pi_2)\cdots = s(\pi'_0)s(\pi'_1)s(\pi'_2)\cdots$: ρ is accepting $\Leftrightarrow$ ρ' is accepting.



Example

$q_0 \cdots q_i \mapsto \max_{0 \le j \le i} \Omega(q_j)$ is an aggregation for a max-parity automaton with coloring Ω.

Automata Computing Aggregations

Every automaton $\mathfrak{M}$ with input alphabet Q and state set M computes an aggregation $s_{\mathfrak{M}}\colon Q^+ \to M$: $s_{\mathfrak{M}}(\pi)$ is the state reached by $\mathfrak{M}$ when processing π.

Automata Computing Aggregations

Every automaton $\mathfrak{M}$ with input alphabet Q and state set M computes an aggregation $s_{\mathfrak{M}} \colon Q^+ \to M$: $s_{\mathfrak{M}}(\pi)$ is the state reached by $\mathfrak{M}$ when processing π.

Example

$q_0 \cdots q_i \mapsto \max_{0 \le j \le i} \Omega(q_j)$ computable by automaton with state set $\Omega(Q)$.

Abstract Block Games

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.

Abstract Block Games

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.

$x \in \Sigma_I^*$:

Abstract Block Games

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.

$x \in \Sigma_I^*$:

$y \in \Sigma_O^*$:

Abstract Block Games

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.

q_0

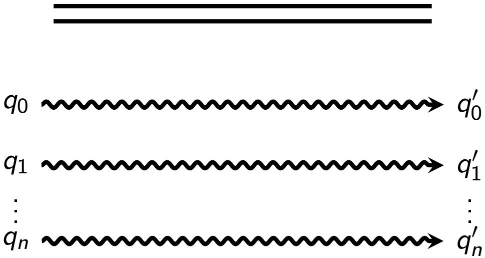
q_1

$\vdots$

q_n

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

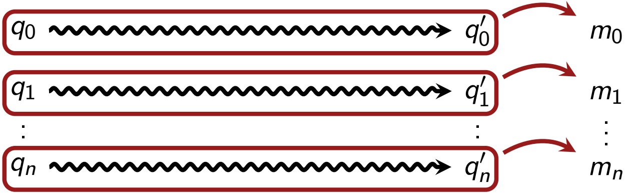
- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.



Abstract Block Games

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

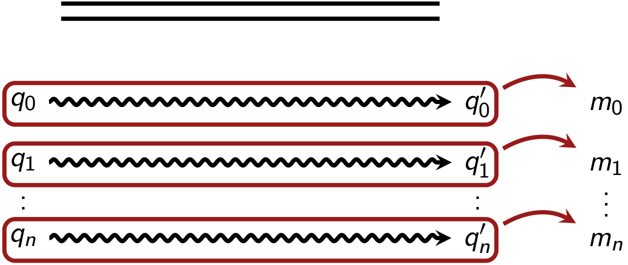
- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.



Abstract Block Games

Fix $\mathfrak{A}$ recognizing winning condition $L(\mathfrak{A}) \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and let $s_{\mathfrak{M}} \colon Q^+ \to M$ be aggregation for $\mathfrak{A}$ computed by some $\mathfrak{M}$.

- Define $x \equiv x'$ iff x and x' induce the same behavior in $\mathfrak{A}$, i.e., the same state changes and the corresponding runs have the same $s_{\mathfrak{M}}$-value.



- $\equiv$ has index at most $2^{|Q|^2|M|}$.

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.

S_0

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.

S_0

q_I

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.

S_0 S_1

q_I

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.

S_0 S_1

$\cup$

q_I

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1)\cdots$.

S_0 $\quad$ S_1

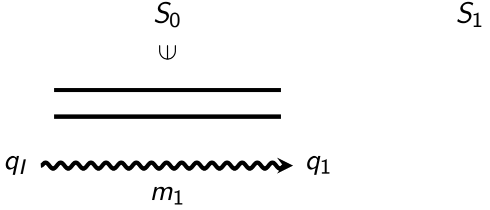
$\uplus$

q_I

Abstract Block Games

The abstract block game is played as follows:

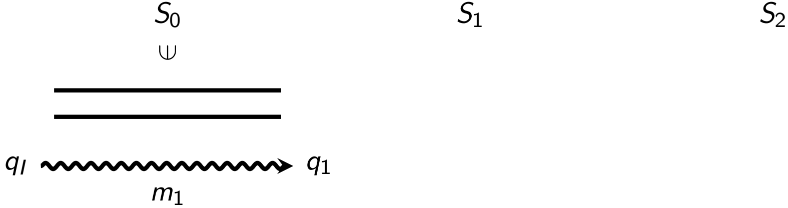
- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.



Abstract Block Games

The abstract block game is played as follows:

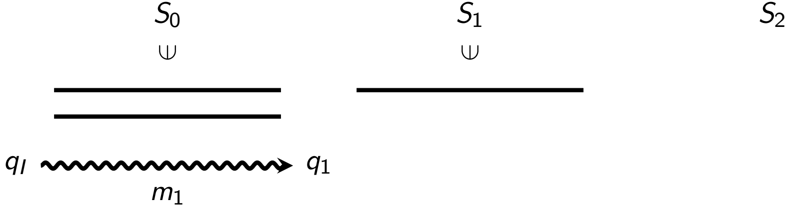
- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.



Abstract Block Games

The abstract block game is played as follows:

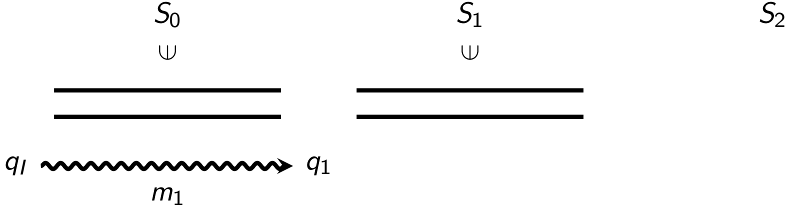
- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.



Abstract Block Games

The abstract block game is played as follows:

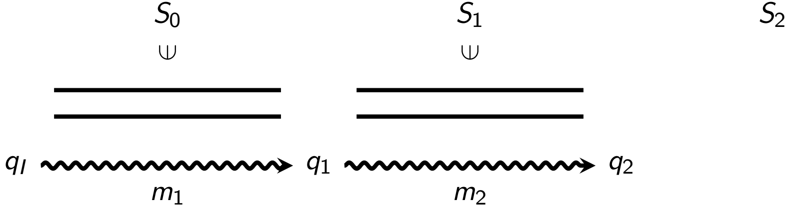
- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.



Abstract Block Games

The abstract block game is played as follows:

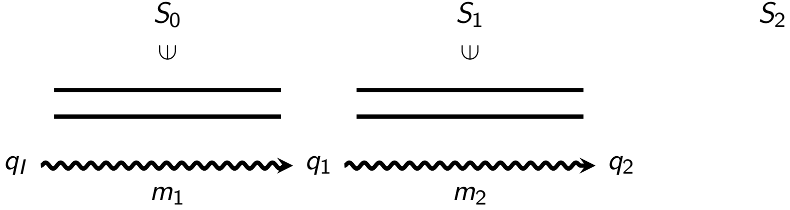
- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.



Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1) \cdots$.

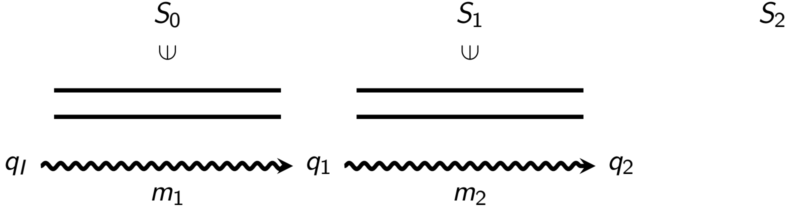


- Player O wins if $m_1 m_2 \cdots$ is aggregation of accepting run.

Abstract Block Games

The abstract block game is played as follows:

- Player I picks equivalence classes $S_0 S_1 \cdots$.
- Player O picks compatible sequence $(q_0, *)(q_1, m_1)\cdots$.



- Player O wins if $m_1 m_2 \cdots$ is aggregation of accepting run.
- This is a delay-free Gale-Stewart game!
- Automaton reconizing winning condition is (roughly) of size $\mathcal{O}(\mathrm{index}(\equiv))$.

Main Theorem

Theorem

Let $\mathfrak{A}$ be an ω-automaton, let $s_{\mathfrak{M}}$ be an aggregation for $\mathfrak{A}$, and define $d = 2^{|Q|^2 \cdot |M|}$.

1. If Player O wins the delay game with winning condition $L(\mathfrak{A})$ for any lookahead, then she also wins the corresponding abstract block game.
2. If Player O wins the abstract block game, then she also wins the block game with winning condition $L(\mathfrak{A})$ and block size d.
3. Moreover, if she has a finite-state winning strategy for the abstract game, then she has a finite-state winning strategy of the same size for the block game.

Main Theorem

Theorem

Let $\mathfrak{A}$ be an ω-automaton, let $s_{\mathfrak{M}}$ be an aggregation for $\mathfrak{A}$, and define $d = 2^{|Q|^2 \cdot |M|}$.

1. If Player O wins the delay game with winning condition $L(\mathfrak{A})$ for any lookahead, then she also wins the corresponding abstract block game.
2. If Player O wins the abstract block game, then she also wins the block game with winning condition $L(\mathfrak{A})$ and block size d.
3. Moreover, if she has a finite-state winning strategy for the abstract game, then she has a finite-state winning strategy of the same size for the block game.

Corollary

Solving delay games equivalent to solving abstract block games and constant lookahead $2d$ is sufficient.

Conclusion

Also in the Paper:

1. Another type of aggregation suitable for quantitative acceptance conditions.
2. The same framework yields decidability and finite-state strategies for quantitative delay games w.r.t. constant lookahead.

Conclusion

Also in the Paper:

1. Another type of aggregation suitable for quantitative acceptance conditions.
2. The same framework yields decidability and finite-state strategies for quantitative delay games w.r.t. constant lookahead.

Unpublished (with Sarah Winter):
Recall that automata implementing finite-state strategies in block games process blocks $\Rightarrow$ Exponentially-sized alphabets.

1. Implement transition and output function as transducers.
2. Upper and lower bounds on size in both models.
3. Tradeoffs between these models.