

Memory for Infinitary Games

Łukasz Kaiser

`kaiser@informatik.rwth-aachen.de`

Three Kinds of Infinity

Infinity in models might come from distinct sources:

- (I) **Nonterminating behaviour** arises naturally when modelling reactive systems interacting with environment, even if the system itself has only finitely many states.
- (II) **Infinite state space** appears when recursive calls and stack are modelled, but the conditions might still depend on finitely many parameters or states.
- (III) **Infinitary conditions** arise when the system has both infinite state space and the properties we are interested in depend on an unbounded number of parameters,

There are good reasons to investigate all three kinds!

Games on Infinite Graphs

Examples leading to games on infinite graphs:

- verification and synthesis of **pushdown systems**
- programs with recursion
- conditions depending on infinitely many parameters, e.g. **unbounded stack height**.

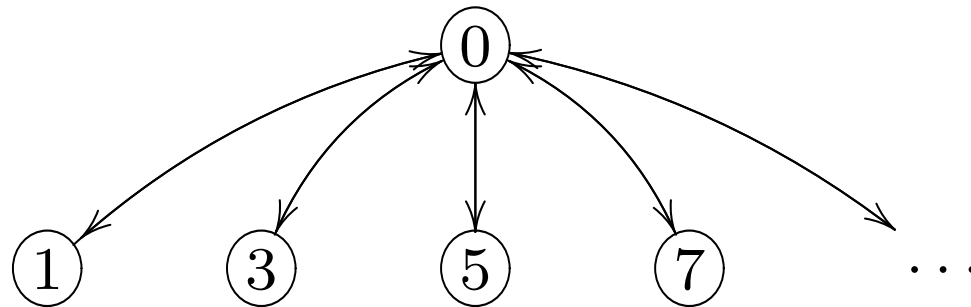
Questions:

- which infinitely coloured games are positional?
- which are determined with finite memory?
- which are determined with **simple** infinite memory?

Memory for Infinitely Coloured Parity Games

Min-parity games are positionally determined even in the case with infinitely many colours (Grädel, Walukiewicz).

This is **not the case for max-parity games**, these are not determined even with finite memory.



Memory must be infinite but can still be simple!

Memory Structures

Memory structure for a game \mathcal{G} with positions in V

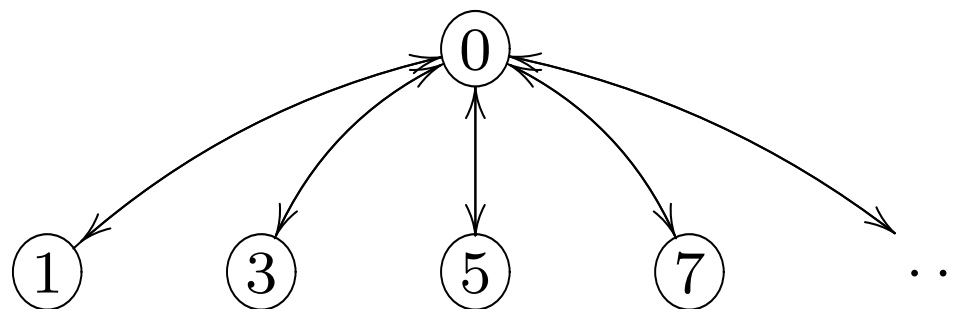
$$\mathfrak{M} = (M, \text{update}, \text{init})$$

- M is a set of **memory states** (finite or infinite)
- $\text{update} : M \times V \rightarrow M$ is a **memory update function**
- $\text{init} : V \rightarrow M$ is a **memory initialisation function**

Strategy with memory \mathfrak{M} allows the Player only to look at the current position and the memory state and update memory according to the prescribed **update** operation.

Memory Structure Example

Look at the the previously considered game:



The following memory is enough for Player 0 to win:

$$\mathfrak{M} = (\mathbb{N}, \text{update}(n, m) := \max(n, m), \text{init}(n) := 0)$$

Finite Appearance Record

Definition of Finite Appearance Record

A d -dimensional **FAR-memory** for a game \mathcal{G} with colours in C is a memory structure $(M, \text{update}, \text{init})$ with $M = (C \cup N)^d$ for some finite set N such that whenever

$$\text{update}(m_1, \dots, m_d, v) = (m'_1, \dots, m'_d)$$

then

$$m'_i \in \{m_1, \dots, m_d\} \cup N \cup \{\Omega(v)\}$$

- Intuitively captures **operating with registers**,
- Is a generalization of **latest appearance record**, which is used to reduce Muller games to parity games.

FAR for Games with Infinitely Many Colours

The class of **FAR-determined conditions** includes:

- Downward cones $\mathcal{F} = \{X : X \subseteq A\}$
- Singleton conditions $\mathcal{F} = \{A\}$
- Finite unions of upwards cones $\mathcal{F} = \{X : X \supseteq A_1 \vee \dots \vee X \supseteq A_k\}$
- Winning conditions with finitely many winning sets
 $\mathcal{F} = \{A_1, \dots, A_k\}$
- Min-parity games (positional)
- Max-parity games with bounded moves
there is a constant d so that $|\Omega(v) - \Omega(w)| \leq d$ for all $(v, w) \in E$

Game Reductions with Memory

Given a game arena $G = (V_0, V_1, E)$ and a memory structure $\mathfrak{M} = (M, \text{update}, \text{init})$ construct a new game arena $G \times \mathfrak{M} = (V_0 \times M, V_1 \times M, E_{\text{update}})$,

$$E_{\text{update}} = \{(v, m)(v', m') : (v, v') \in E \text{ and } m' = \text{update}(m, v')\}$$

with a simpler winning condition.

We say that \mathcal{G} **reduces via memory \mathfrak{M} to \mathcal{G}'** , if $G' = G \times \mathfrak{M}$ and every play in \mathcal{G}' is won by the same player as the projected play in \mathcal{G} .

Reductions are a tool for theoretical analysis as well as a way to get efficient implementations.

Memory Reduction Example: Downward Cone

Consider any **downwards cone** $\mathcal{F}_0 = \{X : X \subseteq A\}$ for a fixed set $A \subseteq \mathbb{N}$

To reduce a game with winning condition \mathcal{F} to a **min-parity** game it suffices to store **the maximal colour** m seen so far.

$$\Omega'(v, m) = \begin{cases} 2m + 2 & \text{if } \Omega(v) \in A \\ 2\Omega(v) + 1 & \text{otherwise} \end{cases}$$

\Leftarrow If $\text{Inf}(\pi) \subseteq A$ then Player 0 wins π' since no odd colour is seen infinitely often in π' .

\Rightarrow If there is some $a \in \text{Inf}(\pi) \setminus A$, then $2a + 1$ occurs infinitely often in π' , and since $a \leq m$ from some point onwards, no smaller even colour can have this property, so Player 1 wins π' .

Conclusions and Future Work

Conclusions:

- Memory analysis is of crucial importance for algorithmic synthesis
- Infinitary winning conditions arise naturally in **verification and synthesis of infinite-state systems**
- **FAR-memory** can be used for some infinitely coloured games

Open questions:

- are arbitrary **max-parity games** determined with FAR-memory?
- obtain **complete classification** of FAR-determined Muller conditions
- what other types of memory and reduction methods can be used?

Thank You