Obliging Games

Krishnendu Chatterjee[1], Florian Horn[1,2], and Christof Löding[3]

[1] IST Austria (Institute of Science and Technology Austria)
[2] CNRS, LIAFA, Université Paris 7, France
[3] RWTH Aachen, Germany
krish.chat@ist.ac.at; horn@liafa.jussieu.fr;
loeding@informatik.rwth-aachen.de

Abstract. Graph games of infinite length provide a natural model for open reactive systems: one player (Eve) represents the controller and the other player (Adam) represents the environment. The evolution of the system depends on the decisions of both players. The specification for the system is usually given as an ω-regular language L over paths and Eve's goal is to ensure that the play belongs to L irrespective of Adam's behaviour.

The classical notion of winning strategies fails to capture several interesting scenarios. For example, strong fairness (Streett) conditions are specified by a number of request-grant pairs and require every pair that is requested infinitely often to be granted infinitely often: Eve might win just by preventing Adam from making any new request, but a "better" strategy would allow Adam to make as many requests as possible and still ensure fairness.

To address such questions, we introduce the notion of *obliging* games, where Eve has to ensure a strong condition Φ, while always allowing Adam to satisfy a weak condition Ψ. We present a linear time reduction of obliging games with two Muller conditions Φ and Ψ to classical Muller games. We consider obliging Streett games and show they are co-NP complete, and show a natural quantitative optimisation problem for obliging Streett games is in FNP. We also show how obliging games can provide new and interesting semantics for multi-player games.

1 Introduction

Games played on graphs provide a natural theoretical model to study problems in verification, such as synthesis of reactive systems [PR89,RW87], synthesis of systems from specifications [BL69,Chu62], and μ-calculus model-checking [Koz83,Sti01].

The vertices of the graph represent the states of the system, the edges represent transitions, the paths represent behaviours, and the players (Eve and the opponent Adam) represent the controller for the system and its environment, respectively. The goal of the controller is to satisfy a specification (desired set of behaviours) irrespective of the way the environment behaves: the synthesis of such a controller corresponds to the construction of a winning strategy in the graph game.

The class of ω-regular objectives provide a robust specification language to express properties that arise in verification and synthesis of reactive systems [Tho97]. Muller and parity specifications are two canonical ways to specify ω-regular objectives. In the classical study of graph games with ω-regular objectives, the input is a graph game G and an ω-regular objective Φ, and the question is whether there is a winning strategy for a player (Eve) that ensures that Φ is satisfied irrespective of the strategy of the other player (Adam).

A specification Φ often consists of two parts: an assumption Φ_A and a guarantee Φ_G and the specification requires $\Phi_A \rightarrow \Phi_G$. The specification Φ_A typically represents the environment assumption under which the guarantee Φ_G needs to be ensured. A winning strategy for Φ may vacuously satisfy Φ by violating Φ_A, whereas a better strategy would ensure the "strong" specification Φ and allow the "weak" specification Φ_A. For example, consider a Streett (fairness) condition: the fairness condition consists of a set of k request-grant pairs, and requires that every request that appears infinitely often, must be granted infinitely often. A winning strategy may satisfy the fairness conditions by not allowing requests to happen, whereas a better strategy would be as follows: it ensures the strong specification that asks for the satisfaction of the strong fairness specification, and allows for the corresponding weak specification that requires that grants are allowed to happen infinitely often.

To address the question above we consider a very general framework of games with two different levels of specifications: a strong one Φ and a weak one Ψ which are, in general, independent of each other. A "gracious" strategy for Eve must *ensure* the strong specification (in the classical sense), and *allow* the weak one: Adam has the choice to satisfy Ψ. We refer to them as *obliging games*. In the important case of fairness specifications, the weak specification can be self-derived from the fairness specification, and the weak specification requires that the requests are allowed to happen infinitely often. The contribution of our work is as follows:

1. We present a linear time reduction of obliging games (with two Muller conditions) to classical games (with a single Muller condition) such that Eve has a winning strategy in the classical game if, and only if, she has a gracious strategy in the obliging game.
2. We present a detailed analysis of the reduction and memory requirement for obliging games when both specifications are given as parity conditions.
3. In the case of fairness specifications (Streett-generalized Büchi conditions), we show that the problem of the existence of a gracious strategy for Eve is co-NP complete.

   We also study a quantitative optimisation version of this problem and show that it belongs to FNP (functional NP).

4. We also show how our concepts can be extended to multi-player games, leading to new and interesting semantics in the context of verification.

Related work Our notion of "gracious strategy" can be likened to "permissive strategies", which allow as many behaviours as possible) [BJW02]. In [BJW02] it has been shown that most general strategies can be constructed only for safety conditions, and for parity objectives a strategy that captures behaviour of all memoryless winning strategies was presented. Our work is different as our objectives are more general (Muller), and the goal is to construct a strategy that allows a given objective. Our work is also related to multi-player games on graphs and Nash equilibria [CMJ04,Umm08]. However in Nash equilibria there are no two levels of specifications as considered in obliging games.

2 Definitions

Arenas. A *two-player game arena* A is a triple $(V, V_\circ, E)$ where (V, E) is a finite directed graph without deadlocks (each vertex has at least one outgoing edge) and $V_\circ$ is a subset of V called *Eve's vertices*. The vertices in $v \setminus V_\circ$ are *Adam's vertices* and are usually denoted by $V_\Box$.

Plays and Strategies. A *play* ρ on the arena A is a (possibly infinite) sequence $\rho_1\rho_2\dots$ of vertices such that for all $i < |\rho|$, we have $(\rho_i, \rho_{i+1}) \in E$. The *limit vertices of* ρ, denoted by $\mathrm{Inf}(\rho)$, are the vertices occurring infinitely often in ρ: $\mathrm{Inf}(\rho) = \{q \mid \exists^\infty i, \rho_i = q\}$.

A *strategy* of Eve on the arena A is a function σ from $V^*V_\circ$ to V such that for all $x \in V^*$ and for all $v \in V_\circ$, we have $(v, \sigma(xv)) \in E$. A play ρ is *consistent with* σ (or *a* σ*-play*) if for all $i < |\rho|,\ \rho_i \in V_\circ \Rightarrow \rho_{i+1} = \sigma(\rho_1 \dots \rho_i)$.

Strategies can also be defined as *strategies with memory*. In this case, σ is a tuple $(M, m_0, \sigma^{\mathtt{u}}, \sigma^{\mathtt{n}})$, where M is the (possibly infinite) set of *memory states*, m_0 is the initial memory content, $\sigma^{\mathtt{u}} : (E \times M) \to M$ is the *memory update* function, and $\sigma^{\mathtt{n}} : (V \times M) \to V$ is the *next-move* function. The memory-update function can naturally be extended from edges to finite sequences of vertices: $\sigma^{\mathtt{u}}_+(v_0v_1\cdots v_i, m)$ is m if $i = 0$ and $\sigma^{\mathtt{u}}((v_{i-1}, v_i), \sigma^{\mathtt{u}}_+(v_0v_1\cdots v_{i-1}, m))$ if $i \geq 1$. Using this definition, the next move determined by σ for a play $xv \in V^*V_\circ$ is $\sigma^{\mathtt{n}}(v, m)$, where $m = \sigma^{\mathtt{u}}_+(xv, m_0)$. A strategy is *finite-memory* if M is a finite set, and *memoryless* if M is a singleton. Adam's strategies are defined in a similar way.

Muller conditions. A Γ*-colouring* γ *of an arena* is a partial function of the edges of A to an arbitrary set of colours Γ. We use partial functions here because this sometimes eases the specification of the winning conditions. However, for formal reasons, we sometimes use a colour "$-$" that corresponds to the undefined value. This colour is not considered when building the limit set of a colour sequence (hence the limit set can be empty).

A *Muller condition* Φ *on* Γ is a subset of 2^{Γ}, and a play ρ of A *satisfies* Φ if, and only if, $\mathrm{Inf}(\gamma(\rho)) \in \Phi$. Here, $\gamma(\rho)$ corresponds to the sequence of colours obtained by applying γ to the edges of ρ. This is a finite or infinite sequence over Γ, or an infinite sequence over $\Gamma \cup \{-\}$ using the above convention.

We also consider the usual special cases of Muller conditions (recall that we allow partial colourings):

- the *Büchi condition* is the condition $\{\{\top\}, \{\bot, \top\}\}$ on $\{\bot, \top\}$;
- the *co-Büchi condition* is the condition $\{\emptyset, \{\top\}\}$ on $\{\bot, \top\}$;
- the k*-generalised Büchi condition* is the condition $\{\{1, \ldots, k\}\}$ on $\{1, \ldots, k\}$;
- the k*-parity condition* is the condition on $\{0, \ldots, k-1\}$ containing all and only the subsets whose minimum is even;
- a k*-Streett* condition on Γ is given by a set $\{(R_1, G_1), \ldots, (R_k, G_k))\}$ of k request-grant pairs of subsets of Γ. It contains all and only the subsets that for each i either intersect G_i or do not intersect R_i.

In the course of our proofs, it is often useful to consider boolean operations on Muller conditions, in which we interpret negation as complementation and conjunction as Cartesian product: if Φ and Ψ are conditions on Γ_Φ and Γ_Ψ, then $\Phi \wedge \Psi$ is the condition on $\Gamma_\Phi \times \Gamma_\Psi$ which contains all and only the sets whose projection on the first component belongs to Φ, and projection on the second component belongs to Ψ.

Notice that colourings are *partial* functions, so their product may return a colour for only one of the components. We then use the neutral colour "$-$" for the undefined component.

Classical and obliging games. A classical Muller game G on Γ is a triple (A, γ, Φ) where A is an arena, γ is a Γ-colouring of A, and Φ —the *winning* condition— is a Muller condition on Γ. An infinite play ρ of A is winning for Eve if it satisfies Φ. A strategy σ is uniformly winning (resp. winning from a vertex q) for Eve if any σ-play (resp. any σ-play starting in q) is winning for her. A vertex q is winning for Eve if she has a winning strategy from q. The winning region of Eve is the set of vertices winning for her. Adam's winning plays, strategies, vertices, and regions are defined likewise, except that a play is winning for Adam if it does not satisfy Φ.

An obliging game G is a tuple $(A, \gamma_\Phi, \Phi, \gamma_\Psi, \Psi)$, where A is an arena, γ_Φ is a Γ_Φ-colouring, Φ —the *strong* condition— is a Muller condition on Γ_Φ, γ_Ψ is a Γ_Ψ-colouring, and Ψ — the *weak* condition— is a Muller condition on Γ_Ψ. A *uniformly gracious strategy* σ *for Eve* is such that:

- every infinite σ-play ρ satisfies Φ;

– for any finite σ-play x, there is an infinite σ-play ρ satisfying Ψ such that x is a prefix of ρ .

So, Eve has to *allow* Adam to build a play satisfying Ψ at any position, regardless of what he previously did. However, she does not need to *ensure* Ψ if Adam is not willing to cooperate. Notice that there is no dual notion of spoiling strategy for Adam. In particular, the notion of "determinacy" does not make sense in obliging games, as Adam cannot demonstrate Eve's lack of grace with a single strategy.

We refer to obliging games by the names of the two conditions, with the strong condition first: for example, a parity/Büchi obliging game is an obliging game $G = (A, \gamma_\Phi, \Phi, \gamma_\Psi, \Psi)$, where Φ is a parity and Ψ is a Büchi condition.

Example 1. Consider the parity/parity obliging game in Figure 1. The pairs define the colours of the edge, the first component corresponding to the strong condition (Φ) and the second component to the weak condition (Ψ).

In order to satisfy Φ, a play has to either take the edge (q_4, q_6) infinitely often, or the edge (q_8, q_6) infinitely often and the edge (q_7, q_2) finitely often. To satisfy Ψ, an infinite play has to take the edge (q_7, q_2) infinitely often. In this game Eve has to behave differently depending on whether Adam moves to q_3 or q_4. If the token reaches q_6 coming from q_4, then Eve can safely move to q_7. If the game reaches q_6 coming from q_3, then she can first complete the cycle $q_6 q_5 q_8 q_6$ and then move to q_5 and then to q_0. This strategy can be implemented using memory of size 3 and it is a gracious strategy since each path satisfies Φ and Adam can produce a play satisfying Ψ by always moving to q_4.

It is not difficult to observe that there is no gracious strategy for Eve with memory of size two for this game. □

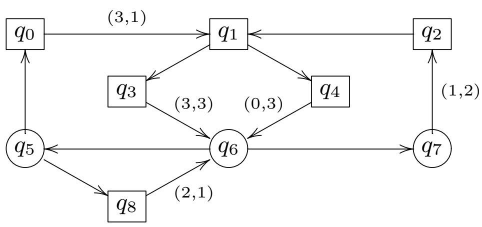

Fig. 1. A parity/parity obliging game

3 Reducing obliging games to classical games

In this section we provide a general method to reduce obliging games to classical games with a single winning condition. The underlying idea is based on the construction of merciful strategies from [BSL04]: we construct an extended game graph in which Adam decides either to choose his next move himself or to leave this choice to Eve. If he always leaves the choice to Eve from some point onwards, then Eve has to prove that Adam indeed has the possibility to satisfy the weak condition. Consequently, the winning condition for Eve in the new game is the strong condition from the obliging game in conjunction with the weak condition in the case that Adam only finitely often makes his own choice.

Note that in the case that Eve has to satisfy the weak condition, the game remains in a subarena that is completely controlled by Eve. We use this fact by allowing to simplify the weak condition by means of non-deterministic ω-automata. The required technical framework is defined below.

We use ω-automata with an acceptance condition specified on the transitions of the automaton rather than on the states. In our setting, an ω-automaton is of the form $\mathcal{M} = (Q, \Sigma, q_{in}, \Delta, \gamma_\Upsilon, \Upsilon)$, where Q is a finite set of states, Σ is the input alphabet, $q_{in} \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $\gamma_\Upsilon : \Delta \to \Gamma_\Upsilon$ is a (partial) colouring function, and Υ is an acceptance condition over Γ_Υ similar to the winning conditions defined for games. We write transitions (q, a, r) with $\gamma_\Upsilon((q, a, r)) = c$ as $q \xrightarrow{a:c} r$.

A run of $\mathcal{M}$ on an infinite word $\alpha \in \Sigma^\omega$ is an infinite sequence $\zeta = q_0 q_1 q_2 \cdots$ of states such that $q_0 = q_{in}$, and $(q_i, \alpha(i), q_{i+1}) \in \Delta$ for each $i \geq 0$. We define the infinite colour sequence induced by α and ζ as the sequence obtained by applying γ_Υ to each transition:

$$\gamma_\Upsilon(\alpha, \zeta) = \gamma_\Upsilon((q_0, \alpha(0), q_1))\gamma_\Upsilon((q_1, \alpha(1), q_2))\gamma_\Upsilon((q_2, \alpha(2), q_3)) \cdots$$

The run ζ on α is accepting if $\gamma_\Upsilon(\alpha, \zeta)$ satisfies the acceptance condition. The language $L(\mathcal{M})$ accepted by $\mathcal{M}$ is the set of all infinite words on which $\mathcal{M}$ has an accepting run.

As usual, we call an automaton deterministic if for each pair of state $q \in Q$ and each $a \in \Sigma$ there is at most one transition $(q, a, r) \in \Delta$.

We are interested in automata accepting languages that correspond to winning conditions. Given a winning condition Φ over Γ_Φ, we define the language $L_\Phi \subseteq (\Gamma_\Phi \cup \{-\})^\omega$ as the set of all infinite sequences that satisfy Φ (recall that "$-$" is a neutral colour representing the undefined value and is not considered for evaluating the winning condition).

Lemma 2. *Let $G = (A, \gamma_\Phi, \Phi, \gamma_\Psi, \Psi)$ be an obliging game with arena $A = (V, E)$, and let $\mathcal{M} = (Q, \Gamma_\Psi, q_{in}, \Delta, \gamma_\Upsilon, \Upsilon)$ be an ω-automaton accepting L_Ψ.*

There is a game $G' = (A', \gamma_\Lambda, \Lambda)$ and a mapping $\iota : V \to V'$ with the following properties: (1) $\Lambda = \Phi \wedge (\Upsilon \vee B)$ for a Büchi condition B; (2) for each vertex $v \in V$, Eve has a gracious strategy from v in G if, and only if, she has a winning strategy from the vertex $\iota(v)$ in G'; and (3) from a winning strategy for Eve in G' from $\iota(v)$ with memory of size n one can construct a gracious strategy for Eve in G from v with memory of size $2 \cdot |Q| \cdot n$.

Proof. To simplify the reduction, we assume without loss of generality that the arena is alternating, *i.e.* $E \subseteq (V_\circ \times V_\Box) \cup (V_\Box \times V_\circ)$.

We construct G' in such a way that, at any time in a play, Adam can ask Eve to show a path that satisfies Ψ. This is realised by introducing a second copy of G in which all vertices belong to Eve. In this copy we additionally keep track of the states of the automaton $\mathcal{M}$ recognising Ψ.

If Adam chooses to switch to this copy, Eve makes the choices on behalf of Adam. Consequently, if from some point onward Adam decides to always leave his choices to Eve, the resulting play has to satisfy Φ and Ψ. Otherwise, it is sufficient for Eve to satisfy Φ. The Büchi condition is used to distinguish these two cases. Whether Ψ is satisfied can be decided using the condition Υ on the state sequence of $\mathcal{M}$.

Formally, the game $G' = (A', \gamma_\Lambda, \Lambda)$ and the mapping ι are constructed as follows:

- The winning condition is $\Lambda = \Phi \wedge (\Upsilon \vee B)$ where B is a Büchi condition.
- The arena $A' = (V', V'_\circ, E')$ and the colouring γ_Λ of E' are defined as follows:
  - $V' = (V_\circ \times \{play\}) \cup (V_\Box \times \{choose\} \times Q) \cup (V \times \{show\} \times Q)$;
  - $V'_\circ = (V_\circ \times \{play\}) \cup (V \times \{show\} \times Q)$;
  - Let u and v be vertices in V; q and r be states in Q; and a, b, c be colours in Γ_Φ, Γ_Ψ, Γ_Υ such that $u \xrightarrow{(a,b)} v$ in E and $q \xrightarrow{b:c} r$ in Δ. Then the following edges belong to E':

$$\begin{aligned} u \in V_\circ :&\ (u, play) \xrightarrow{(a,-,\bot)} (v, choose, q_{in}) \\ u \in V_\Box :&\ (u, choose, q) \xrightarrow{(a,-,\top)} (v, play) \\ &\ (u, choose, q) \xrightarrow{(-,-,\bot)} (u, show, q) \\ u \in V_\circ :&\ (u, show, q) \xrightarrow{(a,c,\bot)} (v, choose, r) \\ u \in V_\Box :&\ (u, show, q) \xrightarrow{(a,c,\bot)} (v, show, r) \end{aligned}$$

- The mapping ι maps each $v \in V_\circ$ to $(v, play)$ and each $v \in V_\Box$ to $(v, choose, q_{in})$.

A schematic view of the construction is shown in Figure 2. We refer to the nodes from $V_\circ \times \{play\}$ as the play part of the game, the nodes from $V \times$

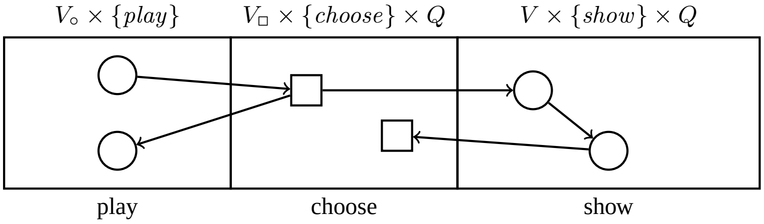

Fig. 2. Schematic view of the reduction from Lemma 2

$\{show\} \times Q$ as the show part, and the nodes from $V_\Box \times \{choose\} \times Q$ as the choice part.

We start by showing that a gracious strategy σ for Eve in the obliging game G can be used to define a winning strategy for Eve in G': Each play ρ' in G' naturally corresponds to a play ρ in G that is obtained by removing the vertices of the type $(v, show, q)$ for $v \in V_\Box$ and then projecting away the $\{play, show, choose\}$ and the Q components from the vertices. Let us denote this operation by del, i.e., $\rho = del(\rho')$.

The winning strategy of Eve in G' is defined as follows. For a finite play x' that ends in a node of the form $(u, play)$ with $u \in V_\circ$, Eve looks at the play $del(x')$ in G, checks which move (u, v) she would have made according to σ, and then moves to $(v, choose, q_{in})$ in G'.

If the play x' in G' enters the show part in a node $(u, show, q_{in})$ for the first time after having been in the play part, then Eve considers the play $x = del(x')$ in G. Since σ is a gracious strategy, there is a possible continuation ρ of x such that $x\rho$ is a σ-play satisfying Ψ. In particular, since Ψ is a Muller condition, ρ satisfies Ψ and there is an accepting run ζ of $\mathcal{M}$ on ρ. Eve stores ρ and ζ in her memory for the strategy σ' and now moves from $(u, show, q_{in})$ according to ρ for the first component, and according to ζ for the third component.

If the play x' in G' is in a node $(u, show, q)$ such that Eve has already stored some ρ and ζ in her memory as described above, then she simply moves according to ρ and ζ: she checks at which position in the play she has stored ρ and ζ, which part of ρ and ζ she has already reproduced since then, and makes the corresponding next move to reproduce one more step of ρ and ζ.

If Adam at some point decides to enter the play part, i.e., to move to a vertex from $V_\circ \times \{play\}$, then Eve erases ρ and ζ from her memory.

If π' is an infinite play according to this strategy, then it certainly satisfies Φ because $del(\pi')$ is a σ-play and the Γ_Φ sequence of π' corresponds to the one of $del(\pi')$ except for some insertions of the neutral colour $-$. Furthermore, either Adam infinitely often moves to a vertex from $V_\circ \times \{play\}$, in which case the

Büchi condition B is satisfied, or from some point onward Eve simulates ρ and ζ to infinity, yielding a play in G' that satisfies Υ because ζ satisfies Υ. This shows that π' is winning and hence we have defined a winning strategy for Eve, as desired.

For the other direction it suffices to show the third claim of the lemma since the existence of a winning strategy for Eve in G' implies the existence of a finite-memory winning strategy. Let $(M, m_0, \varsigma^{\mathtt{n}}, \varsigma^{\mathtt{u}})$ be a winning strategy for Eve in G'. We define a gracious strategy $(\{\mathsf{p}, \mathsf{s}\} \times Q \times M, (\mathsf{p}, q_{in}, m_0), \sigma^{\mathtt{n}}, \sigma^{\mathtt{u}})$ for Eve in G. This strategy distinguishes two cases to decide whether to use $\varsigma^{\mathtt{n}}$ as defined on the $play$ vertices or on the $show$ vertices. These two cases depend on the behaviour of Adam. If Adam makes a move in G from a vertex v that corresponds to the move of $\varsigma^{\mathtt{n}}$ from the vertex $(v, show)$ in G', then $\sigma^{\mathtt{u}}$ updates the first component of the memory to s, i.e., $\sigma^{\mathtt{n}}$ starts simulating $\varsigma^{\mathtt{n}}$ as if the play is in the $show$ part of G'. If Adam makes a move that is not of this kind, then $\sigma^{\mathtt{u}}$ updates the first component of the memory to p and $\sigma^{\mathtt{n}}$ simulates the behaviour of $\varsigma^{\mathtt{n}}$ on the $play$ part of G'.

We first give the definition of the next move function $\sigma^{\mathtt{n}}$, which is quite straightforward:

$\sigma^{\mathtt{n}}(u, \langle \mathsf{p}, q_{in}, m \rangle) = v$ with $\varsigma^{\mathtt{n}}((u, play), m) = (v, play, q_{in})$,
$\sigma^{\mathtt{n}}(u, \langle \mathsf{s}, q, m \rangle) = v$ if $\varsigma^{\mathtt{n}}((u, show, q), m) = (v, play, q')$ for some q'.

The definition of the memory update function $\sigma^{\mathtt{u}}$ is a bit more involved since we have to distinguish the different behaviours of player 1 as explained above. Below, we define the update of the memory for a move from u to v in G for different memory contents. If $u \in V_\circ$, we assume that v is the vertex that is chosen by the next move function $\sigma^{\mathtt{n}}$ because otherwise the move from u to v cannot occur in a play according to the strategy.

(i) If $u \in V_\circ$, then $\sigma^{\mathtt{u}}(u, \langle \mathsf{p}, q_{in}, m \rangle, v) = \langle \mathsf{p}, q_{in}, m' \rangle$ with

$$m' = \varsigma^{\mathtt{u}}((u, play), m, (v, choose, q_{in}))$$

and $\sigma^{\mathtt{u}}(u, \langle \mathsf{s}, q, m \rangle, v) = \langle \mathsf{p}, q', m' \rangle$ with

$$m' = \varsigma^{\mathtt{u}}((u, show, q), m, (v, choose, q'))$$

and $\varsigma^{\mathtt{n}}((u, show, q), m) = (v, choose, q')$ (here we use the assumption that $\sigma^{\mathtt{n}}(u, \langle \mathsf{s}, q, m \rangle) = v$, i.e., v is the target of the next move function).

(ii) If $u \in V_\Box$ and $\varsigma^{\mathtt{n}}((u, show, q), \varsigma^{\mathtt{u}}((u, choose, q), m, (u, show, q))) = (v, show, q')$, then $\sigma^{\mathtt{u}}(u, \langle \mathsf{x}, q, m \rangle, v) = \langle \mathsf{s}, q', m' \rangle$ with

$$m' = \varsigma^{\mathtt{u}}_+((u, play, q)(u, show, q)(v, show, q'), m)$$

for all $\mathsf{x} \in \{\mathsf{p}, \mathsf{s}\}$. This is the case where the move from u to v of Adam in G corresponds to the move that Eve would have made in his place in G'. To obtain m' we look at how the memory would have evolved in G' in the move sequence in which Adam gives the choice to Eve.

(iii) If $u \in V_\Box$ and $\varsigma^{\mathtt{n}}((u, show, q), \varsigma^{\mathtt{u}}((u, choose, q), m, (u, show, q))) = (v', show, q')$ for some $v' \neq v$, then $\sigma^{\mathtt{u}}(u, \langle \mathsf{x}, q, m\rangle, v) = \langle \mathsf{p}, q_{in}, m'\rangle$ with

$$m' = \varsigma^{\mathtt{u}}((u, choose, q), m, (v, play))$$

for all $\mathsf{x} \in \{\mathsf{p}, \mathsf{s}\}$. This is the case where Adam makes a choice different from the one that Eve would have made on his behalf in G'.

We now show that this strategy is indeed gracious in G. From the definition of $\sigma^{\mathtt{n}}$ and $\sigma^{\mathtt{u}}$ one can see that for every $\sigma^{\mathtt{n}}$-play ρ there exists a corresponding $\varsigma^{\mathtt{n}}$-play ρ' that is obtained from $play$ by inserting appropriate vertices from $V_\Box \times \{show\} \times Q$ at those positions where $\sigma^{\mathtt{u}}$ updates the first component of the memory to s, i.e., if (ii) in the definition of $\sigma^{\mathtt{u}}$ is applied.

To formalize this let $\rho = v_0 v_1 v_2 \cdots$ be a $\sigma^{\mathtt{n}}$-play and let

$$\langle \mathsf{x}_0, q_0, m_0\rangle \langle \mathsf{x}_1, q_1, m_1\rangle \langle \mathsf{x}_2, q_2, m_2\rangle \cdots \in (\{\mathsf{p}, \mathsf{s}\} \times Q \times M)^\omega$$

be the corresponding sequence of memory contents according to $\sigma^{\mathtt{u}}$.

Similar to the operation del from the first implication of the proof we now define an operation ins that transforms ρ into a corresponding play based on the sequence of memory contents. By abuse of notation we also define the operation ins to work on tuples of nodes by inserting the necessary information (we assume for simplicity that the play starts in $V_\circ$):

$$ins(\rho) = (v_0, play)\, ins(v_0, v_1)\, ins(v_1, v_2)\, ins(v_2, v_3) \cdots$$

with

$$ins(v_i, v_{i+1}) = \begin{cases} (v_{i+1}, play) & \text{if } \mathsf{x}_{i+1} = \mathsf{p} \text{ and } v_{i+1} \in V_\circ, \\ (v_{i+1}, choose, q_{i+1}) & \text{if } v_{i+1} \in V_\Box, \\ (v_i, show, q_i)(v_{i+1}, show, q_{i+1}) & \text{if } x_{i+1} = \mathsf{s} \text{ and } v_{i+1} \in V_\circ. \end{cases}$$

Now one can verify that a σ-play ρ in G is transformed by ins into a ς-play ρ' in G'. Therefore, ρ satisfies Φ because the colour sequences from Γ_Φ of ρ and ρ' are the same except for some insertions of the neutral colour $-$. Furthermore, at each position of a play in G, Adam has the possibility to move so that Eve updates her memory content to an element with s in the first component: for a σ-play x in G Adam checks what would have been the move of Eve according to ς in G' for the play $ins(x)$ extended by Adam's move to the show part of the game. If Adam always copies these ς moves to G from some point onwards, then the resulting play ρ satisfies Ψ because $ins(\rho)$ is a ς-play in G' that does visit $V_\circ \times \{play\}$ only finitely often and hence satisfies Υ. This means that the simulated run of $\mathcal{M}$ on the play is accepting and therefore the corresponding play in G satisfies Ψ. This shows that σ is indeed a gracious strategy. $\square$

Lemma 2 provides a reduction of obliging games to standard games. This notion is formalised as follows. We say that an obliging game G can be reduced to a standard game G' with memory m if:

1. there is a mapping ι from the vertices of G to the vertices of G' such that for each vertex v of G Eve has a gracious strategy from v in G if, and only if, Eve has a winning strategy from $\iota(v)$ in G';
2. given a winning strategy for Eve from $\iota(v)$ in G' with memory of size n, one can compute a gracious strategy for Eve from v in G with memory of size $m \cdot n$.

We also use this notion in connection with classes of games. A class $\mathcal{K}$ of games can be reduced to a class $\mathcal{K}'$ of games with memory m if each game G in $\mathcal{K}$ can be reduced to a game G' in $\mathcal{K}'$ with memory m. The time complexity of such a reduction is the time needed to compute G' from G, to compute the mapping ι, and to compute the strategy in G from the strategy in G'.

We can now instantiate Lemma 2 for several types of obliging games to obtain results on their complexity. The first instantiation is for general Muller conditions using the fact that the winning sequences for a condition Ψ can be recognised by a one state ω-automaton which itself uses the condition Ψ.

Theorem 3. *There is a linear time reduction with memory* 2 *from* Φ/Ψ *obliging games to standard* $(\Phi \wedge (\Psi \vee B)))$ *games for a Büchi condition* B.

The point of using a non-deterministic ω-automaton in the formulation of Lemma 2 is illustrated by the following result.

Theorem 4. *There is a polynomial time reduction with memory* $2(\ell+1)k$ *from* $2k$*-parity/*2ℓ*-parity obliging games to standard* $(2k+2)$*-parity games.*

Proof. We apply Lemma 2 with a Büchi automaton accepting L_Ψ for the 2ℓ-parity condition Ψ. Such a Büchi automaton is easily constructed using $(\ell + 1)$ states. On the first state the automaton loops and outputs $\bot$ for each input priority. Using the other ℓ states the automaton can guess at any point that $2i$ is the minimal priority which appears infinitely often in the input sequence. It moves to state i and outputs $\top$ whenever priority $2i$ appears on the input. For greater priorities it outputs $\bot$, and for priorities smaller than $2i$ there is no transition. One easily verifies that this automaton accepts L_Ψ.

Lemma 2 yields a reduction with memory $2(\ell + 1)$ to a ($2k$-parity $\wedge$ Büchi) game (using the fact that a disjunction of two Büchi conditions is equivalent to a single Büchi condition). Analysing the Zielonka tree [Zie98,DJW97] of a ($2k$-parity $\wedge$ Büchi) condition shows that it has k leafs and the technique from [DJW97] gives a reduction to $2k+2$-parity game with memory k. The composition of these two reductions gives the claimed reduction. One can note that this proof also works if the weak condition is a Rabin condition with ℓ pairs. □

Since parity games are determined with memoryless strategies (see, e.g., [Tho97] or [Zie98]), Theorem 4 directly gives an upper bound on the memory required for a gracious strategy in parity/parity obliging games.

Corollary 5. *If Eve has a gracious strategy in a $2k$-parity/2ℓ-parity obliging game, then she has a gracious strategy with memory of size at most $2(\ell+1)k$.*

In the case $\ell = 1$, we have rather tight lower bound for the required memory. Indeed, it is possible to construct a $2k$-parity/Büchi obliging game where Eve needs k memory states. The case $k = 6$ is depicted in Figure 3 (in order to improve readability, there are some vertices of Adam from where two edges lead to the same target).

Eve has a gracious strategy with k memory states that works as follows: if Adam just played $2i$, she plays $2i+1$; otherwise, she plays $2(k-1)$. This strategy clearly ensures the parity condition. Furthermore, Adam can get an infinite number of visits to the $\top$ edge, by always answering $2(i-1)$ to $2i+1$.

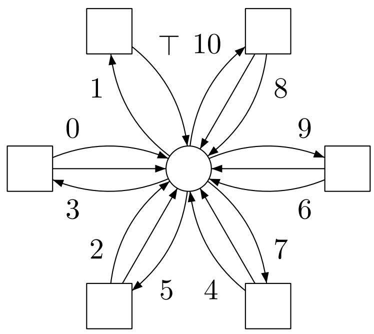

Fig. 3. At least 6 memory states

There is no gracious strategy for Eve with less than k states: as there are k successors of the central vertex, one of them is never visited. Thus, Eve can ultimately not propose the lower ones safely, and either does not guarantee the parity condition or eventually forbids the Büchi condition.

4 Obliging Streett games

Streett games are a very natural setting for obligingness questions. Indeed, the Streett condition allows Eve to win by either granting requests or denying Adam the possibility to make them. It is thus natural to consider k-Streett/k-generalised Büchi objectives, where the objectives of the weak condition are exactly the requests of the strong one. We call them simply obliging Streett games.

As a generalised Büchi condition can be recognised by a Streett automaton with only one state, we can use Lemma 2 to reduce an obliging k-Streett game with n vertices to a classical $2k$-Streett game with $2n$ vertices. As classical Streett games can also be reduced to obliging Streett games (by always allowing Adam to go to a vertex where all the pairs are forever requested and granted) and classical Streett games problem is co-NP complete [EJ88], it follows that the obliging Streett games problem is co-NP complete:

Theorem 6. *The decision problem of existence of a gracious strategy for Eve in obliging Streett games is co-*NP *complete.*

In the cases where Eve does not have a gracious strategy, we might be interested in knowing how many simultaneous requests she can allow. This can be defined as a threshold problem: "Given $\ell \leq k$, is it possible to allow Adam to visit at least ℓ different requests?"; or as an optimisation problem: "What is the highest ℓ such that Eve can allow Adam to visit at least ℓ different requests?".

Theorem 7. *The threshold problem of obliging Streett games is co-*NP *complete; and the optimisation problem of obliging Streett games is in* FNP.

Proof. As the optimal number of request that Eve can allow is between -1 and k, the second statement follows directly from the first one. Furthermore, it is clear that the threshold problem is co-NP hard since it generalises both classical Streett games (for $\ell = 0$) and obliging Streett games (for $\ell = k$).

In order to show that the threshold problem belongs to co-NP, we use once more Lemma 2: we just need a non-deterministic automaton recognising the words where at least ℓ different colours are visited infinitely often. We describe such an automaton in Figure 4, with the following conventions: the alphabet is $\{1, \ldots, k\}$, and for each i, $R_i = \{i\}$; there is an unmarked loop on each state; unmarked edges are enabled for each letter and are labelled $\perp$. □

5 Multi-Player obliging games

An interesting feature of obliging games is that they provide new and interesting semantics for multi-player games. In this setting, Eve has more than one opponent and each must be allowed to satisfy his weak condition, regardless of what the others do.

The definitions are similar to the two-player case, *mutatis mutandis*: a *n-player arena* A is a finite directed graph (V, E) without deadlocks whose vertices are partitioned in n subsets, $V_\circ, V_1, \ldots, V_{n-1}$; a *$n$-player obliging game* is a n-player arena and as many colourings and conditions: $\gamma_\circ, \Phi; \gamma_1, \Psi_1; \ldots; \gamma_{n-1}, \Psi_{n-1}$. A gracious strategy σ for Eve in such a game is such that:

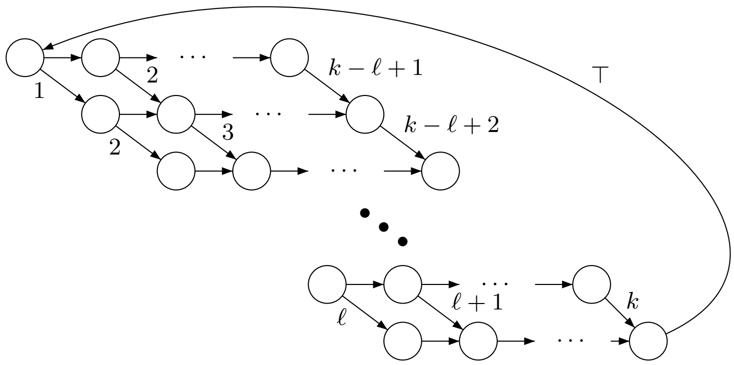

Fig. 4. Büchi automaton recognising repeated ℓ-out-of-k

- any infinite σ-play ρ satisfies Φ;
- for any $1 \leq i < n$, for any finite σ-play x, there is a strategy τ_i for Player i consistent with x such that any infinite play consistent with both σ and τ_i satisfies Ψ_i.

We can solve n-player obliging games by reduction to classical two-player games, in a way similar to the two-player case. However, we do not use automata to check whether the play satisfies the weak conditions, for two reasons: first, we cannot use non-deterministic automata: even if one opponent yields control of his moves, the others can still interfere so Eve cannot simply "choose" a correct run; second, we would have to remember the current state of each automaton, leading to an exponential blow-up in the size of the arena.

Theorem 8. *Let $G = (A, \gamma_\Phi, \Phi, \gamma_1, \Psi_1, \ldots; \gamma_{n-1}, \Psi_{n-1})$ be a n-player obliging game with arena $A = (V, E, V_\circ, V_1, \ldots, V_{n-1})$. We can compute, in time linear in the size of G, a game $G' = (A', \gamma_\Upsilon, \Upsilon)$ of size linear in the size of G and a mapping $\iota : V \to V'$ with the following properties:*

1. *$\Upsilon = \Phi \wedge (\Psi_1 \vee B_1) \wedge \ldots (\Psi_{n-1} \vee B_{n-1})$, where $B_1, \ldots, B_{n-1}$ are Büchi conditions.*
2. *For each vertex v in A, Eve has a gracious strategy from v in G if, and only if, she has a winning strategy from the vertex $\iota(v)$ in G'.*

Proof. The construction of G' is similar to its counterpart in the proof of Lemma 2. Each opponent has the possibility to leave Eve choose his move in his stead. If one of them eventually always does so, the play has to satisfy his weak condition; otherwise, the corresponding Büchi condition allows Eve to ignore it. The proof is even simpler, as there is no need to keep track of a run of an automaton. □

6 Conclusion

In this work we introduced the notion of obliging games and presented a linear time reduction to classical games for all ω-regular objectives specified as Muller objectives. We also presented a complete analysis for the reduction and memory requirement when the specifications are given as parity objectives. We studied the important class of fairness (Streett) conditions, and showed that obligingness Streett games are co-NP complete. We also studied a natural quantitative optimization problem for obliging Streett games and proved inclusion in FNP. We showed extension of the notion of obligingness games to multi-player games and how it leads to new and interesting semantics. In future work we will explore how the solution of obliging games can be used to synthesize more desirable controllers.

References

[BJW02] J. Bernet, D. Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *Theoretical Informatics and Applications*, 36(3):261–275, 2002.

[BL69] J.R. Büchi and L.H. Landweber. Solving Sequential Conditions by Finite-State Strategies. *Transactions of the AMS*, 138:295–311, 1969.

[BSL04] Y. Bontemps, P-Yv. Schobbens, and C. Löding. Synthesis of Open Reactive Systems from Scenario-Based Specifications. *Fundamenta Informaticae*, 62(2):139–169, 2004.

[Chu62] A. Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.

[CMJ04] K. Chatterjee, R. Majumdar, and M. Jurdziński. On Nash Equilibria in Stochastic Games. In *Proceedings of CSL, LNCS 3210*, pages 26–40. Springer, 2004.

[DJW97] S. Dziembowski, M. Jurdziński, and I. Walukiewicz. How Much Memory is Needed to Win Infinite Games? In *Proceedings of LICS*, pages 99–110. IEEE, 1997.

[EJ88] E.A. Emerson and C.S. Jutla. The Complexity of Tree Automata and Logics of Programs. In *Proceedings of FOCS*, pages 328–337. IEEE, 1988.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *TCS*, 27(3):333–354, 1983.

[PR89] A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *Proceedings of POPL*, pages 179–190, ACM, 1989.

[RW87] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[Sti01] C. Stirling. *Modal and Temporal Properties of Processes*. Graduate Texts in Computer Science. Springer, 2001.

[Tho97] W. Thomas. Languages, Automata, and Logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.

[Umm08] M. Ummels. The Complexity of Nash Equilibria in Infinite Multiplayer Games. In *Proceedings of FoSSaCS, LNCS 4962*, pages 20 – 34. Springer, 2008.

[Zie98] W. Zielonka. Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.