# Verification of Broadcasting Multi-Agent Systems against an Epistemic Strategy Logic

**Francesco Belardinelli**
Laboratoire IBISC, UEVE
and IRIT Toulouse
belardinelli@ibisc.fr

**Alessio Lomuscio**
Department of Computing
Imperial College London
a.lomuscio@imperial.ac.uk

**Aniello Murano** and **Sasha Rubin**
DIETI
Università degli Studi di Napoli
murano@na.infn.it
rubin@unina.it

## Abstract

We study a class of synchronous, perfect-recall multi-agent systems with imperfect information and broadcasting, i.e., fully observable actions. We define an epistemic extension of strategy logic with incomplete information and the assumption of uniform and coherent strategies. In this setting, we prove that the model checking problem, and thus rational synthesis, is non-elementary decidable. We exemplify the applicability of the framework on a rational secret-sharing scenario.

## 1 Introduction

Epistemic logic has a long tradition in knowledge representation and reasoning, multi-agent systems (MAS), and more broadly in artificial intelligence [Meyer and Hoek, 1995]. A significant line of research over the past twenty years has concerned its combination with various temporal logics such as LTL, CTL, and the like [Clarke *et al.*, 2002]. The resulting syntax can express a wide range of properties of multi-agent systems, including the knowledge agents have about the world, about each other's knowledge, how this evolves over time and whether sophisticated epistemic states such as common knowledge are acquired in a system's run [Halpern and Vardi, 1989].

Temporal-epistemic properties of multi-agent systems have been studied under a variety of assumptions, including synchronicity, asynchronicity, perfect recall, bounded recall, no learning, and observational semantics [Fagin *et al.*, 1995]. These aspects are now known to impact the resulting axiomatisations [Halpern *et al.*, 2003; Belardinelli and Lomuscio, 2009] as well as the complexity of the verification problem [Meyden and Shilov, 1999]. For these reasons, a key aspect in this line of work has been the identification of expressive fragments with relatively low complexity.

Recently there has been considerable interest in the extension of the formalisms above to languages sufficiently expressive to capture strategic abilities of agents. Towards this aim, alternating-time temporal logic (ATL) [Alur *et al.*, 2002] and strategy logic (SL) [Mogavero *et al.*, 2014] have been put forward and combined with epistemic modalities and uniform strategies [Hoek and Wooldridge, 2003; Belardinelli, 2014; Huang and Meyden, 2014; Čermák *et al.*, 2014; Berthon *et al.*, 2017a].

Reasoning about strategic abilities of MAS under imperfect information is known to be difficult. For example, model checking MAS against ATL specifications under incomplete information goes from PTIME-complete to $\Delta_2^p$-complete under memoryless strategies (i.e., imperfect recall) [Jamroga and Dix, 2006] and is undecidable under memoryfull strategies (i.e., perfect recall) [Dima and Tiplea, 2011]. For this reason it is of interest to identify expressive classes of MAS for which the model checking problem is decidable. The aim of this paper is to make a contribution in this direction.

Specifically, we introduce ESL, an epistemic extension of SL based on synchronous perfect-recall strategies (Section 2). The language introduced can express rational synthesis [Fisman *et al.*, 2010; Wooldridge *et al.*, 2016; Kupferman *et al.*, 2016], but its model-checking problem is undecidable. However, we identify a significant class BA-iCGS of systems: those having broadcast (i.e., fully observable) actions (Section 2.2) and prove that model checking BA-iCGS against ESL is non-elementary decidable (Section 4). This is a tight result as a matching lower-bound already holds in the perfect-information case. We illustrate our formalism on a rational secret-sharing scenario with broadcast actions (Section 3).

**Related Work.** As mentioned above, several approaches have been put forward to reason about strategies and knowledge in the context of MAS.

SL and knowledge have been combined before in the context of MAS. In [Cermák, 2014; Čermák *et al.*, 2014], an epistemic variant of SL [Mogavero *et al.*, 2014] was introduced. However, this was limited to epistemic sentences, whereas we consider the full combined language, and the approach assumed observational semantics, whereas we here consider synchronous perfect recall. Although not studied in these papers these formalisms have an undecidable model checking problem if evaluated under synchronous perfect recall. Also, [Berthon *et al.*, 2017b] defines a variant of SL with uniform strategies. They achieve decidability by a variation of the tradition of assuming a hierarchy on the observations. In this paper we do not make any hierarchical assumptions.

A key aspect of the work here presented is that it relies on broadcasting to achieve decidability in the context of a very expressive specification language. The notion of broadcast has already been studied in the context of knowledge [Fa-

gin *et al.*, 1995; Lomuscio *et al.*, 2000]. A further important result in this area is that for broadcast systems the synthesis problem of specifications in LTL and knowledge is decidable [Meyden and Wilke, 2005]. However, ESL is strictly more expressive and synthesis, which in our case can be expressed via model checking, can also be shown to be decidable. An approach to reasoning about strategies and knowledge under broadcast was also recently presented in [Belardinelli *et al.*, 2017]. However, their logic is considerably less expressive than ours, as it is based on ATL and not SL. In particular, it cannot express Nash equilibria and rational synthesis, which are essential features of this contribution.

Rational synthesis has been studied before in the context of perfect information. In [Kupferman *et al.*, 2016] the strong-rational synthesis problem with LTL objectives (and aggregation of finitely many objectives), is shown to be 2EXPTIME-complete.In [Gutierrez *et al.*, 2017], Equilibrium Logic is introduced to reason about Nash equilibria in games with LTL and CTL objectives. However, both cases assume perfect information of the agents. Synthesis under imperfect information has been first tackled in [Gutierrez *et al.*, 2016] albeit for a restricted class of CGS, viz. *reactive modules*. In this paper we explore synthesis in CGS under imperfect information.

## 2 Strategy Logic with Imperfect Information

In this section we present strategy logic (SL) (see [Mogavero *et al.*, 2014] for a definition of SL) in an imperfect information setting. In particular, we introduce the class of imperfect information concurrent game structures (iCGS) with broadcast actions only (BA-iCGS). We start with some preliminaries. For an infinite or non-empty finite sequence $u \in X^\omega \cup X^+$ of elements in $X$, we write $u_i$ for the $(i+1)$-th element of $u$, i.e., $u = u_0 u_1 \ldots$. For $i \geq 0$, $u_{\leq i}$ is the prefix of $u$ of length $i+1$, i.e., $u_{\leq i} = u_0 u_1 \ldots u_i$. The empty sequence is denoted as $\epsilon$. The length of a finite sequence $u \in X^*$ is denoted as $|u|$. For a vector $v \in \prod_i X_i$ we denote the $i$-th co-ordinate of $v$ by $v(i)$. In particular, for $F \in \prod_i (X_i)^Y$ we may write $F(i) \in X_i^Y$ and $F(i)(y) \in X_i$.

### 2.1 iCGS

Hereafter we consider concurrent game structures enriched with indistinguishability relations. These are the standard setting for agent-based logics under imperfect information [Jamroga and van der Hoek, 2004; Bulling and Jamroga, 2014].

**Definition 1** (iCGS)**.** *An* imperfect information concurrent game structure (iCGS) *is a tuple* $\mathsf{S} = \langle Ag, AP, \{Act_a\}_{a\in Ag}, S, S_0, \mathsf{tr}, \{\sim_a\}_{a\in Ag}, \lambda\rangle$*, where:*

1. *$Ag$ is the finite non-empty set of* agent names*.*
2. *$AP$ is the finite non-empty set of* atomic propositions*.*
3. *$Act_a$ is the finite non-empty set of* actions *for $a \in Ag$; for $A \subseteq Ag$, let $Act_A = \cup_{a\in A} Act_a$, and let $Act = Act_{Ag}$.*
4. *$S$ is the finite non-empty set of* states *and $S_0 \subseteq S$ is the non-empty set of* initial states*.*
5. *$\mathsf{tr} : S \times \mathsf{ACT} \to S$ is the* transition function*, where $\mathsf{ACT} = \prod_{a\in Ag} Act_a$ is the set of all* joint actions*.*
6. *$\sim_a \subseteq S^2$ is the* indistinguishability relation *for agent $a$, which is an equivalence relation.*
7. *$\lambda : AP \to 2^S$ is the* labelling *function that assigns to each atom $p$ the set of states $\lambda(p)$ in which $p$ holds.*

A *concurrent game structure (CGS)* is an iCGS for which $\sim_a = \{(s,s) : s \in S\}$ for all $a \in \mathrm{Ag}$. This corresponds to the perfect-information setting [Alur *et al.*, 2002].

We now define what it means for an agent to have *synchronous perfect-recall* in an iCGS S. A *history* in S is a non-empty finite sequence $h_0 h_1 \ldots$ in $S^+$ such that for all $i \geq 0$, there exists a joint action $J_i \in \mathsf{ACT}$ such that $h_{i+1} \in \mathsf{tr}(h_i, J_i)$. The set of all histories in S is denoted as $\mathsf{hist}(\mathsf{S})$, and the set of histories $h'$ that extend history $h$ is denoted as $\mathsf{hist}(\mathsf{S}, h)$, that is, $h'_{\leq |h|} = h$.

Hereafter we use the following notation: if $\sim$ is a binary relation on $S$, we define the extension of $\sim$ to histories as the binary relation $\equiv$ on $\mathsf{hist}(\mathsf{S})$ such that $h \equiv h'$ iff $|h| = |h'|$ (i.e., synchronicity) and $h_j \sim h'_j$ for all $0 \leq j \leq |h|$ (i.e., perfect recall). We consider three instantiations for individual, common and distributed knowledge respectively. If $\sim_a$ is the indistinguishability relation for agent $a$, then two histories $h, h'$ are *indistinguishable to agent $a$*, if $h \equiv_a h'$. For $A \subseteq \mathrm{Ag}$, let $\sim_A^{\mathbb{C}} = (\cup_{a\in A} \sim_a)^*$, where $*$ denotes the reflexive and transitive closure (w.r.t. relation composition), and its extension to histories is denoted $\equiv_A^{\mathbb{C}}$. For $A \subseteq \mathrm{Ag}$, let $\sim_A^{\mathbb{D}} = \cap_{a\in A} \sim_a$, and its extension to histories is denoted $\equiv_A^{\mathbb{D}}$.

A *deterministic memoryfull strategy*, or simply *strategy*, is a function $\sigma : \mathsf{hist}(\mathsf{S}) \to \mathrm{Act}$ (recall that $\mathrm{Act} = \cup_{a\in \mathrm{Ag}} \mathrm{Act}_a$). The set of all strategies is denoted $\Sigma(\mathsf{S})$. Further, a strategy $\sigma_a$ is *coherent for $a$* if for every $h \in \mathsf{hist}(\mathsf{S})$, $\sigma_a(h) \in \mathrm{Act}_a$; while $\sigma_a$ is *uniform for $a$* if for all $h, h' \in \mathsf{hist}(\mathsf{S})$, $h \equiv_a h'$ implies $\sigma_a(h) = \sigma_a(h')$. Then, a *joint full strategy* is a function $\sigma_{\mathrm{Ag}} : \mathrm{Ag} \to \Sigma(\mathsf{S})$ that associates to each agent $a \in \mathrm{Ag}$ a strategy that is both coherent and uniform for $a$. We write $\sigma_{\mathrm{Ag}}(a) = \sigma_a$. For every $s_0 \in S_0$, a joint full strategy $\sigma_{\mathrm{Ag}}$ defines a unique infinite sequence $\pi(\sigma_{\mathrm{Ag}}) = s_0 s_1 \ldots$ of states, i.e., for all $i \geq 0$, $s_{i+1} = \mathsf{tr}(s_i, \sigma_{\mathrm{Ag}}(s_0 s_1 \ldots s_i))$. A history $h$ is *consistent* with $\sigma_{\mathrm{Ag}}$ if $h$ is a prefix of $\pi(\sigma_{\mathrm{Ag}})$. Given $h \in \mathsf{hist}(\mathsf{S})$, define the set $\mathsf{out}(h, \sigma_{\mathrm{Ag}})$ of *outcomes of $\sigma_{\mathrm{Ag}}$ from $h$* as the set of histories $h' \in \mathsf{hist}(\mathsf{S}, h)$ that extend $h$ and are consistent with $\sigma_{\mathrm{Ag}}$. Notice that for every $i \geq 0$, there is unique $h' \in \mathsf{out}(h, \sigma_{\mathrm{Ag}})$ of length $|h| + i$. Thus, write $\pi(h, \sigma_{\mathrm{Ag}}) \in S^\omega$ for the infinite sequence all of whose prefixes are in $\mathsf{out}(h, \sigma_{\mathrm{Ag}})$.

### 2.2 BA-iCGS— iCGS with Broadcast Actions only

In this paper we focus on a particular class of iCGS, those having broadcast actions only. This section is reported from [Belardinelli *et al.*, 2017] (where these were called iCGS with public actions only).

**Definition 2** (BA-iCGS)**.** *An iCGS* S only has broadcast actions *if for every agent $a \in Ag$, states $s, s' \in S$, and joint actions $J, J' \in \mathsf{ACT}$, if $J \neq J'$ and $s \sim_a s'$ then $\mathsf{tr}(s, J) \not\sim_a \mathsf{tr}(s', J')$. In this case we call* S *a* broadcast iCGS*. We write* BA-iCGS *for the set of broadcast iCGS.*

Broadcast iCGS arise naturally in several MAS scenarios, including epistemic puzzles (e.g., the muddy children puzzle) and games (e.g., battleship). In Section 3 we discuss an application to rational synthesis.

We define the following natural encoding of histories.

**Definition 3.** *Let* $\mathsf{S}$ *be an iCGS. Define the* encoding function $\mu : S_0 \times \mathsf{ACT}^* \to \mathsf{hist}(\mathsf{S})$ *that maps* $(s_0, u)$ *to the history* $h$ *of length* $|u|+1$ *and such that* $h_0 = s_0$ *and* $h_j = \mathsf{tr}(h_{j-1}, u_{j-1})$ *for* $1 \leq j \leq |u|$.

In case $\mathsf{S}$ is a BA-iCGS, then $\mu$ is a bijection, i.e., for every $h \in \mathsf{hist}(\mathsf{S})$ there exists a unique $(s_h, u_h) \in S_0 \times \mathsf{ACT}^*$ such that $\mu(s_h, u_h) = h$. Moreover, the moment different joint actions are taken, two histories become distinguishable:

**Lemma 1.** *Let* $\mathsf{S}$ *be a BA-iCGS. For all* $a \in Ag$, $u, u' \in \mathsf{ACT}^*$ *and* $s, s' \in S_0$, *if* $\mu(s,u) \equiv_a \mu(s',u')$ *then* $u = u'$.

*Proof.* Indeed, if $\mu(s,u) \equiv_a \mu(s',u')$ then $|u| = |u'|$ and, for all $0 \leq j \leq |u|$, $\mu(s,u)_j \sim_a \mu(s',u')_j$. By the definition of having only broadcast actions, $u_j = u'_j$ for all $j < |u|$. □

The next characterisation of uniformity in BA-iCGS follows from Lemma 1 and is central to our decidability result:

**Proposition 1.** *Let* $\mathsf{S}$ *be a BA-iCGS, and let* $\sigma$ *be a coherent strategy for agent* $a$. *Then* $\sigma$ *is uniform for agent* $a$ *if and only if for all* $v \in \mathsf{ACT}^*$, $s, s' \in S_0$ *we have that* $\mu(s,v) \equiv_a \mu(s',v)$ *implies* $\sigma(\mu(s,v)) = \sigma(\mu(s',v))$.

### 2.3 The Logic ESL

We now introduce ESL, an epistemic extension of SL. We interpret it on iCGS with history-based semantics.

**Syntax**. Fix a finite set of *atomic propositions (atoms)* $AP$, a finite set of *agents* Ag, and an infinite set Var of strategy variables $x_0, x_1, \ldots$. The *formulas over* $AP$, Ag, *and Var* are built according to the following grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \langle\!\langle x \rangle\!\rangle \varphi \mid (x,a)\varphi \mid \mathbb{K}_a\varphi \mid \mathbb{C}_A\varphi \mid \mathbb{D}_A\varphi$, where $p \in AP$, $x \in$ Var, $a \in$ Ag, and $A \subseteq$ Ag. The set of ESL *formulas* is the one generated by the grammar. The *temporal operators* are X (read "next") and U (read "until"). The *strategy quantifier* is $\langle\!\langle x \rangle\!\rangle$ ("for some strategy $x$, ..."") and the binding operator $(x,a)$ ("by using strategy $x$, agent $a$ can enforce ..."); whilst the *epistemic operators* are $\mathbb{K}_a$ ("agent $a$ knows that"), $\mathbb{C}_A$ ("it is common-knowledge amongst $A$ that"), and $\mathbb{D}_A$ ("the agents in $A$ distributively know that"). We use the usual shorthands, e.g., $\mathsf{true}$ for $p \vee \neg p$, $[[x]]\varphi$ for $\neg\langle\!\langle x \rangle\!\rangle\neg\varphi$, and $\mathbb{E}_A\varphi$ for $\wedge_{a\in A}\mathbb{K}_a\varphi$. The sets *free*$(\varphi)$ and *bnd*$(\varphi)$ of free and bound variables appearing in a formula $\varphi$ are defined as standard [Mogavero *et al.*, 2014]. Intuitively, $x \in$ *free*$(\varphi)$ if $x$ does not occur in $\varphi$ within the scope of any strategy quantifier of the form $\langle\!\langle x \rangle\!\rangle$; while $a \in$ *free*$(\varphi)$ if some temporal operator is not in the scope of a binding $(x,a)$ for agent $a$. Hereafter we assume that sets *free*$(\varphi)$ and *bnd*$(\varphi)$ are disjoint. Moreover, as in [Cermák, 2014] we assume that every variable is quantified at most once in a given formula. All these properties can be ensured w.l.o.g. by renaming bound variables. A *sentence* is a formula $\varphi$ with *free*$(\varphi) = \emptyset$. Finally, we define $shr(x,\varphi) = \{a \in Ag \mid (x,a)\psi \text{ is a subformula of } \varphi\}$ as the set of agents using strategy $x$ in evaluating $\varphi$.

**Semantics**. Fix an iCGS $\mathsf{S}$. An *assignment* is a function $\chi$ : Var $\cup$ Ag $\to \Sigma(\mathsf{S})$ such that for every agent $a \in$ Ag, the strategy $\chi(a)$ is coherent and uniform for $a$. For $x \in$ Var$\cup$Ag and $\sigma \in \Sigma(\mathsf{S})$, the *variant* $\chi^x_\sigma$ is the assignment that maps $x$ to $\sigma$ and coincides with $\chi$ on all other variables and agents. Moreover, if $x = a \in$ Ag then we require $\sigma$ to be coherent and uniform for $a$. An assignment $\chi$ is $\varphi$*-compatible* if, for every $x \in$ Var, the strategy $\chi(x)$ is coherent and uniform for every agent in $shr(x,\varphi)$.

We define $(\mathsf{S}, h, \chi) \models \varphi$ where $h \in \mathsf{hist}(\mathsf{S})$, $\varphi$ is a formula, $\chi$ is a $\varphi$-compatible assignment, and $\pi := \pi(h, \chi|_{\mathrm{Ag}})$ is the unique infinite sequence that extends $h$ by following the restriction of $\chi$ to Ag:

$(\mathsf{S},h,\chi) \models p$ iff $last(h) \in \lambda(p)$, for $p \in AP$
$(\mathsf{S},h,\chi) \models \neg\varphi_1$ iff it is not the case that $(\mathsf{S},h,\chi) \models \varphi_1$
$(\mathsf{S},h,\chi) \models \varphi_1 \wedge \varphi_2$ iff $(\mathsf{S},h,\chi) \models \varphi_i$ for $i \in \{1,2\}$
$(\mathsf{S},h,\chi) \models \langle\!\langle x \rangle\!\rangle \varphi_1$ iff there exists a strategy $\sigma$ that is uniform and coherent for every agent in $shr(x,\varphi_1)$ such that $(\mathsf{S},h,\chi^x_\sigma) \models \varphi_1$
$(\mathsf{S},h,\chi) \models (x,a)\varphi_1$ iff $(\mathsf{S},h,\chi^a_{\chi(x)}) \models \varphi_1$
$(\mathsf{S},h,\chi) \models \mathbb{K}_a\varphi_1$ iff for every history $h' \in \mathsf{hist}(\mathsf{S})$, $h' \equiv_a h$ implies $(\mathsf{S},h',\chi) \models \varphi_1$
$(\mathsf{S},h,\chi) \models \mathbb{C}_A\varphi_1$ iff for every history $h' \in \mathsf{hist}(\mathsf{S})$, $h' \equiv^{\mathbb{C}}_A h$ implies $(\mathsf{S},h',\chi) \models \varphi_1$
$(\mathsf{S},h,\chi) \models \mathbb{D}_A\varphi_1$ iff for every history $h' \in \mathsf{hist}(\mathsf{S})$, $h' \equiv^{\mathbb{D}}_A h$ implies $(\mathsf{S},h',\chi) \models \varphi_1$
$(\mathsf{S},h,\chi) \models \mathsf{X}\varphi_1$ iff $(\mathsf{S},\pi_{\leq|h|+1},\chi) \models \varphi_1$
$(\mathsf{S},h,\chi) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists $i \geq |h|$ s.t. $(\mathsf{S},\pi_{\leq i},\chi) \models \varphi_2$, for all $j$ with $|h| \leq j < i$, $(\mathsf{S},\pi_{\leq j},\chi) \models \varphi_1$,

The following says that the satisfaction is well-defined:

**Lemma 2.** *In the expressions* $(S,h,\chi') \models \varphi'$ *on the right-hand sides,* $\chi'$ *is always a* $\varphi'$*-compatible assignment.*

For a history formula $\varphi$, we write $\mathsf{S} \models \varphi$ to mean that $(\mathsf{S},s,\chi) \models \varphi$ for every $s \in S_0$ and assignment $\chi$ (observe that states are histories of length 1). One can prove (as usual) that the satisfaction of ESL-formulas depends only on their free variables and agents, that is, if assignments $\chi$ and $\chi'$ coincide on *free*$(\varphi)$, then $(\mathsf{S},h,\chi) \models \varphi$ iff $(\mathsf{S},h,\chi') \models \varphi$. Thus, e.g., if $\varphi$ is a sentence, then $\mathsf{S}, s \models \varphi$ iff $(\mathsf{S},s,\chi) \models \varphi$ for some assignment $\chi$.

Clearly ESL extends SL. Indeed, one restricts the syntax of ESL to the epistemic-free fragment, and the models to CGS (i.e., $\sim_a := \{(s,s) : s \in \mathsf{S}\}$ for every $a \in$ Ag):

**Proposition 2.** *For every* SL *sentence* $\varphi$ *there is an* ESL *sentence* $\hat{\varphi}$ *s.t. for all CGS* $\mathsf{S}$, *we have that* $\mathsf{S} \models \varphi$ *iff* $\mathsf{S} \models \hat{\varphi}$.

The proof simply requires to show that the state-based semantics of SL in [Mogavero *et al.*, 2014] can be captured by our history-based semantics.

The next proposition says that ATL$^*$K embeds in ESL (see [Jamroga and van der Hoek, 2004] for the definitions of ATL$^*$K). Although the embedding is as expected, the proof that it is correct is subtle.

**Proposition 3.** *For every* ATL$^*$K *formula* $\varphi$ *there is an* ESL *sentence* $\hat{\varphi}$ *s.t. for all iCGS* $\mathsf{S}$, *we have that* $\mathsf{S} \models \varphi$ *iff* $\mathsf{S} \models \hat{\varphi}$.

*Proof.* The main difficulty is to translate the ATL$^*$ operator $\langle\!\langle A \rangle\!\rangle$ in ESL, which we illustrate. Consider Ag $= \{a_1, a_2, a_3, a_4\}$, $A = \{1,2\}$, and the ATL$^*$K formula $\varphi = \langle\!\langle A \rangle\!\rangle \psi$. The formula says that for every set of uniform strategies, one for each agent in $A$, every path consistent with these strategies satisfies $\psi$. Consider the ESL formula $\hat{\varphi} = \langle\!\langle x_1 \rangle\!\rangle \langle\!\langle x_2 \rangle\!\rangle [[x_3]][[x_4]](x_1,a_1)(x_2,a_2)(x_3,a_3)(x_4,a_4)\hat{\psi}$. Clearly, $\varphi$ logically implies $\hat{\varphi}$ since the paths consistent with $x_1, x_2$ include those generated by uniform strategies $x_1, x_2, x_3, x_4$. On the

other hand, let $\pi$ be any path consistent with strategies $\sigma_1, \sigma_2$ (for agents $a_1, a_2$). It is sufficient to show that there exist uniform strategies, $\sigma_3$ for agent $a_3$ and $\sigma_4$ for agent $a_4$, such that $\pi(s, \sigma_{\mathrm{Ag}}) = \pi$. Indeed, for every $n \geq 0$, let $J \in \mathsf{ACT}$ be a joint action such that i) $\sigma_i(\pi_{\leq n}) = J(i)$ for $i = 1, 2$, and ii) $\mathsf{tr}(\pi_n, J) = \pi_{n+1}$. Define $\sigma_i(\pi_{\leq n}) = J(i)$ for $i = 3, 4$. Since the uniformity condition only restricts pairs of histories of the same length, we can extend $\sigma_3$ and $\sigma_4$ to uniform strategies. Note that $\pi(s, \sigma_{\mathrm{Ag}}) = \pi$, as required. □

We now introduce the main decision problem of this work.

**Definition 4** (Model Checking)**.** *Let $\mathcal{C}$ be a class of iCGS and $\mathcal{F}$ a sublanguage of* ESL. Model checking $\mathcal{C}$ against $\mathcal{F}$ specifications *is the following decision problem: given $\mathsf{S} \in \mathcal{C}$ and $\varphi \in \mathcal{F}$ as input, decide whether $\mathsf{S} \models \varphi$.*

Model checking iCGS against ATL is undecidable [Dima and Tiplea, 2011]. Thus, applying Proposition 3, we get:

**Proposition 4.** *Model checking iCGS against* ESL *is undecidable.*

Indeed, it is undecidable even if $\mathcal{C}$ consists of all iCGS with $|\mathrm{Ag}| = 3$ and $\mathcal{F}$ contains just the ATL formula $\langle\langle\{1,2\}\rangle\rangle \mathsf{G}\, p$, see [Dima and Tiplea, 2011]. The source of the undecidability is the interplay between two assumptions: a) $\sim_1$ and $\sim_2$ are incomparable under the refinement-order on equivalence relations, and b) agent 3 can privately communicate with agents 1 and 2. In the sequel we prove that model checking is decidable assuming all agents only have broadcast actions. Thus, we keep property a) while dropping property b).

## 3 Rational Synthesis under Imperfect Information

In this section we show how to express central game-theoretic properties in ESL, e.g., the existence of Nash equilibria in multi-player games of imperfect information with epistemic objectives. Moreover, we illustrate that ESL can be used to reason about rational secret-sharing, i.e., rational agents that communicate by broadcast actions in order to learn a secret whose "shares" have been distributed amongst them.

### 3.1 Expressing Rational Synthesis in ESL

Several questions in computer science can be cast as the problem of deciding if there exists a joint winning strategy for a coalition of agents against a coalition of adversarial agents (and computing one if it exists). In the verification literature this problem is called *synthesis*.

However, as argued in [Wooldridge *et al.*, 2016; Kupferman *et al.*, 2016; Abraham *et al.*, 2011], the partition of agents into "good" and "bad" is often insufficient, and it is more appropriate to view agents as rational. That is, agents have preferences over outcomes and act in a way that increases their own utility. Then, instead of reasoning about winning strategies, one should reason about *rational* strategy profiles, i.e., that satisfy some notion of equilibrium. Application domains include rational distributed computing and rational cryptography [Abraham *et al.*, 2011], and negotiating systems with self-interested agents [Jennings and others, 2001]. Technically, suppose we are given an iCGS $\mathsf{S}$ representing the multi-agent system, and LTLK-formulas $\gamma_a$ representing the objective of agent $a \in \mathrm{Ag}$. Here, LTLK is the logic consisting of the set of path-formulas of $\mathsf{ATL}^*\mathsf{K}$. We can then talk about Nash equilibria $\overline{\sigma}$ in games of the form $G = \langle \mathsf{S}, \{\gamma_a\}_{a \in \mathrm{Ag}} \rangle$[1]. Rational synthesis considers the following decision problem (sometimes called *E-NASH*) :

**Definition 5** (Rational Synthesis for LTLK objectives, cf. [Kupferman *et al.*, 2016])**.** *Given an iCGS $\mathsf{S}$, LTLK-formulas $\gamma_a$ for every $a \in Ag$, and an LTLK-formula $\varphi$, decide whether there exists a Nash equilibrium $\overline{\sigma}$ in the game $G = \langle \mathsf{S}, \{\gamma_a\}_{a \in Ag} \rangle$ such that the path induced by $\overline{\sigma}$ satisfies $\varphi$.*

Intuitively, $\varphi$ represents some global property that the designer wants to ensure given that agents are self-interested. In case $\varphi = \mathsf{true}$, this simply asks if there exists a Nash-equilibrium. Moreover, if there is such a Nash equilibrium, the synthesis problem concerns deriving one such strategy profile $\overline{\sigma}$. The dual problem, called *Strong Rational Synthesis* (sometimes called *A-NASH*), concerns deciding whether all Nash equilibria induce a path that satisfies $\varphi$ [Kupferman *et al.*, 2016].

We now show that rational synthesis for LTLK objectives reduces to model checking against ESL. Suppose $\mathrm{Ag} = \{a_1, a_2, \ldots, a_n\}$, and let $\overline{x}$ be an $n$-tuple of variables. Let $\beta$ be the expression $(x_1, a_1)(x_2, a_2)\ldots(x_n, a_n)$ that binds agent $a_i$ to strategy $x_i$. The following formula $\mathsf{RatSyn}_{\varphi}(\overline{x})$ in ESL expresses that $\overline{x}$ is a Nash equilibrium whose induced execution satisfies $\varphi$: $(x_1, a_1)\ldots(x_n, a_n)\left[\varphi \wedge \bigwedge_{a \in \mathrm{Ag}} (\langle\langle y \rangle\rangle (y, a)\gamma_a \rightarrow \gamma_a)\right]$.

In words, if agent $a_i$ uses $x_i$ then the resulting execution satisfies $\varphi$, and no agent has an incentive to unilaterally deviate from the strategy profile $\overline{x}$. Then:

**Lemma 3.** *Rational synthesis for* LTLK *objectives is reducible to model checking against the* ESL*-formula $\langle\langle x_1 \rangle\rangle \ldots \langle\langle x_n \rangle\rangle \mathsf{RatSyn}_{\varphi}(\overline{x})$.*

A universally quantified formula is used for Strong Rational Synthesis. It is important to observe that ESL can express other equilibrium concepts such as subgame-perfect equilibria, concepts that capture deviations by groups of players such as $k$-resilience and $t$-immunity, and the combination $(k, t)$-robustness that captures fault-tolerance [Abraham *et al.*, 2011]. Also, ESL is able to express the existence of Nash equilibria w.r.t. epistemic objectives, which, to the best of our knowledge, has not yet been considered in the literature. We illustrate this last point in the next section.

### 3.2 Rational Secret-Sharing with Broadcast

We illustrate the model-checking problem for BA-iCGS against ESL with a simple scenario inspired by [Abraham *et al.*, 2006] that uses broadcast. In the classic $m$-out-of-$n$ secret-sharing problem, for $\mathrm{Ag} = \{1, 2, \ldots, n\}$, initially each agent $i \in \mathrm{Ag}$ privately holds a "share" $f_i$ of a secret $f_0$, and any $m$ "good" agents can collaborate to learn the secret

[1]The framework can also support every agent having finitely-many Boolean objectives aggregated by means of a reward function such as $\max$, cf. [Kupferman *et al.*, 2016].

in spite of the remaining $n-m$ "bad" agents[2]. In the rational version of this scenario, the objective of each agent is to learn the secret, i.e., she prefers to learn the secret rather than not to learn it. Richer, non-binary, preferences can also be handled, including the fact that an agent may prefer that the least number of other agents learn the secret. For simplicity we do not consider such extensions here.

We can model this scenario as an iCGS as follows. The secret is the value of a variable $s$ initially hidden from all agents (formally, a variable $v$ with finite domain $D$ is modelled as $|D|$-many atomic propositions); agent $i$'s share is modelled as a private variable $f_i$; each agent has a private variable $s_i$ that represents what she thinks the secret is; at every step, every agent broadcasts a message (from some fixed finite set of $M$ messages). Finally, the objective $\gamma_i$ of each agent $i$ can be formalised as the LTLK-formula $\mathsf{F}\,\mathsf{G}\,\mathbb{K}_i(s_i = s)$: *from some point on, agent $i$ knows the secret*. Thus, the ESL-formula $\langle\!\langle x_1\rangle\!\rangle \dots \langle\!\langle x_n\rangle\!\rangle \mathsf{RatSyn}_{\varphi}(\overline{x})$ expresses that there is a Nash equilibrium satisfying $\varphi$ in the rational secret-sharing scenario. For instance, one can use $\varphi$ to express that agents make "true" statements, e.g., that if agent $i$ broadcasts "my share is $x$", then indeed $f_i = x$. Observe that by using ESL specifications we can naturally express secrecy and strategic concepts.

## 4 Model Checking BA-iCGS against ESL

In this section we prove the main technical result of this paper.

**Theorem 6.** *Model checking BA-iCGS against* ESL *specifications is decidable and non-elementary complete.*

For the non-elementary lower-bound we use the observation that model-checking SL on CGS (i.e., with perfect-information) is non-elementary [Mogavero *et al.*, 2014], together with the fact that by encoding the last joint action into the states, one can translate a CGS $\mathsf{S}$ into a BA-iCGS $\mathsf{S}'$ such that for all sentences $\varphi$ in ESL, we have that $\mathsf{S} \models \varphi$ iff $\mathsf{S}' \models \varphi$ (the same procedure is used in [Belardinelli *et al.*, 2017]).

For the non-elementary upper-bound, we reduce the model-checking problem of BA-iCGS against ESL specifications to model checking regular-trees against Monadic Second-Order Logic (MSO). The naïve approach is to code every tuple $(\mathsf{S}, h, \chi)$ by a tuple of functions $(\widehat{\mathsf{S}}, \widehat{h}, \widehat{\chi})$ each of whose domain is the set $\mathsf{ACT}^*$ of finite sequences of joint actions, and whose ranges are finite (to be specified later). This encoding allows us to build, for every ESL-sentence $\varphi$, an MSO-formula $\Phi$, such that $(\mathsf{S}, h, \chi) \models \varphi$ iff $T \models \Phi(\widehat{\mathsf{S}}, \widehat{h}, \widehat{\chi})$, where $T$ is the infinite ACT-ary tree generated by $\widehat{\mathsf{S}}$, $\widehat{h}$, and $\widehat{\chi}$. The latter problem is decidable if $\widehat{\mathsf{S}}$, $\widehat{h}$, and $\widehat{\chi}$ are regular functions (a function $f : D^* \to L$ is *regular* if, for each $l \in L$, the set $f^{-1}(l) \subseteq D^*$ is accepted by a finite automaton). Since $\varphi$ is a sentence we can choose $\chi$ arbitrarily, in particular so that it is regular (on the other hand, both $\widehat{\mathsf{S}}$ and $\widehat{h}$ are always regular).

[2] In Shamir's scheme this is implemented by an initially unknown polynomial $f$ of degree $m-1$ in some finite field $F$ with $|F| > n$ and $f(0) \neq 0$; the secret is $f(0)$, each share is $f(i)$, and thus any $m$ shares uniquely determine the secret (by interpolation).

**Monadic Second-Order Logic**. Below we summarise MSO, which extends first-order logic with variables for sets, and recall the fundamental theorem, i.e., that MSO is decidable on regular-trees [Rabin, 1969]. The *syntax* of MSO includes Boolean operators $\neg$ and $\wedge$; individual variables $u, v, w, \dots$; set variables $U, V, W, \dots$; quantifiers over these variables $\exists u, \exists U, \dots$; binary relation symbols $\in$, $=$, and $\preceq$; and unary function symbols $\mathsf{suc}_d$ for every $d$ in a finite set $\Delta$ of *directions*. We denote formulas of MSO by $\Phi, \Psi, \dots$. The *semantics* of MSO is defined over the structure $\mathsf{T}_\Delta = \langle \Delta^*, \{\mathsf{suc}_d\}_{d\in\Delta}\rangle$, called the *unlabelled $\Delta$-ary tree*. The interpretation of individual variables are elements in $\Delta^*$, of set variables are subsets of $\Delta^*$; Boolean operators and quantifiers are interpreted as usual; atoms $u \in U$ and $u = v$ as usual; while $u \preceq v$ is the prefix relation, and $\mathsf{suc}_d(u) = ud$ for $d \in \Delta, u \in \Delta^*$. We will often think of $u \in \Delta^*$ as the singleton set $U = \{u\} \subseteq \Delta^*$. Formulas $\Phi(\overline{U})$ with free variables $\overline{U}$ are interpreted in expanded structures $(\mathsf{T}_\Delta, \overline{A})$, called *labelled $\Delta$-ary trees*, where each $A_i \subseteq \Delta^*$. Instead of writing $(\mathsf{T}_\Delta, \overline{A}) \models \Phi(\overline{U})$, we may write $\mathsf{T}_\Delta \models \Phi(\overline{A})$, or simply $\overline{A} \models \Phi$. A labelled-tree $(\mathsf{T}_\Delta, \overline{A})$ is *regular* if each $A_i \subseteq \Delta^*$ is accepted by a finite automaton.

**Theorem 7.** *[Rabin, 1969] There is a non-elementary time algorithm that, given an* MSO*-formula $\Phi(\overline{U})$ and a regular labelled-tree $(\mathsf{T}_\Delta, \overline{A})$, decides whether $\mathsf{T}_\Delta \models \Phi(\overline{A})$. Also, if $\mathsf{T}_\Delta \models \exists \overline{U}\, \Phi(\overline{U})$ then there is a regular labelled-tree $(\mathsf{T}_\Delta, \overline{A})$ such that $\mathsf{T}_\Delta \models \Phi(\overline{A})$, and the finite automata for all $A_i \subseteq \Delta^*$ are computable.*

We use standard shorthands, e.g., $\epsilon$ for the root; $X = Y$ for $\forall v(v \in X \leftrightarrow v \in Y)$, etc. Say that $\overline{A'}$ *is definable from* $\overline{A}$ if for each $i$ there is an MSO-formula $\varphi_i(x)$ such that $\overline{A} \models \forall x(\varphi_i(x) \leftrightarrow x \in A'_i)$. For an MSO-formula $\Phi(\overline{U})$, we define $\Phi[\overline{A'} \leftarrow \overline{A}]$ for the MSO-formula formed from $\Phi$ in which every variable $U_i$ is replaced by the definition $\varphi_i$ of $A'_i$. Then $\overline{A'} \models \Phi$ iff $\overline{A} \models \Phi[\overline{A'} \leftarrow \overline{A}]$. We now introduce some abbreviations, i.e., variables for functions with finite ranges:

**Definition 8** (Unary Function Variables)**.** *Let $\Theta$ be a finite set of* sorts*. Associate with each type $\theta \in \Theta$ a* finite *set of* labels *$L_\theta$. For every sort $\theta$, we introduce unary function variables $\alpha, \beta, \dots$ of that sort, and quantification, i.e, $\exists\alpha, \exists\beta, \dots$. Define the interpretation of variable $\alpha$ of sort $\theta$ by a function of the form $\alpha : \Delta^* \to L_\theta$. We write $\overline{\alpha} \models \Phi$ to denote $(\mathsf{T}_\Delta, \overline{\alpha}) \models \Phi$. If $\alpha'$ is definable from $\alpha$, we write $\Phi[\alpha' \leftarrow \alpha]$ for the substitution as above.*

We remark that this extension does not add expressive power. Indeed, we can replace the function variable $\alpha$ of sort $\theta$ by a $|L_\theta|$-tuple of set variables $\overline{X}$, and replace every term $\alpha(v) = d$ by the expression $v \in X_d$.

**Directions and sorts**. Fix a BA-iCGS $\mathsf{S}$, the direction set $\Delta = \mathsf{ACT}$, and the set $\Theta$ to consists of four sorts:

- $D$ with labels $L_D = S_0 \to S$ (for representing iCGS);
- $H$ with $L_H = S_0 \cup \{\bot\}$ (for histories);
- $R$ with $L_R = S_0 \to$ Act (for strategies);
- $K$ with $L_K = S_0 \to ((\text{Var} \cup \text{Ag}) \to \text{Act})$ (for assignments).

**Encoding $(\mathsf{S}, h, \chi)$ by functions**. Recall the bijection $\mu : \mathsf{hist}(\mathsf{S}) \to S_0 \times \mathsf{ACT}^*$ in Def. 3 and that we write $h =$

$\mu(s_h, u_h)$ (Section 2.2). The structure $\mathsf{S}$ is encoded by a function $\hat{\mathsf{S}}$ of sort $D$; a history $h$ by a function $\hat{h}$ of sort $H$; an assignment $\chi$ by a function $\hat{\chi}$ of sort $K$; and a strategy $\sigma$ by a function $\hat{\sigma}$ of sort $R$, as follows[3]:

- $\widehat{\mathsf{S}}(v)(t) = \mathsf{tr}(t, v)$ for all $v \in \mathsf{ACT}^*, t \in S_0$.
- $\widehat{h}(u_h) = s_h$, and $\widehat{h}(v) = \bot$ for all $v \in \mathsf{ACT}^*$ with $v \neq u_h$.
- $\widehat{\sigma}(v)(t) = \sigma(\mu(t, v))$ for all $v \in \mathsf{ACT}^*, t \in S_0$.
- $\widehat{\chi}(v)(t)(x) = \chi(x)(\mu(t, v))$ for all $v \in \mathsf{ACT}^*$, $t \in S_0$, $x \in \mathrm{Var} \cup \mathrm{Ag}$.

**Expressing ESL in MSO.** We now show how to express in MSO that a function variable $\alpha$ of a given sort is a valid encoding. First, we can express that a function variable $\alpha$ of sort $H$ is of the form $\widehat{h}$ for some history $h$, i.e., $\exists x(\alpha(x) \in S_0 \wedge \forall y(y \neq x \rightarrow (\alpha(y) = \bot)))$. Second, we can express that a function variable $\alpha$ of sort $D$ is of the form $\widehat{\mathsf{S}}$, i.e., $\bigwedge_{t \in S}(\alpha(\epsilon)(t) = t \wedge \alpha(\mathsf{suc}_d(v))(t) = \mathsf{tr}(\alpha(v)(t), d))$. Third, for every ESL formula $\varphi$, we can express that a function variable of sort $K$ is of the form $\widehat{\chi}$ for some $\varphi$-compatible assignment $\chi$. To do this, it is sufficient to express, for $a \in Ag$, that a function variable $\alpha$ of sort $R$ is of the form $\widehat{\sigma}$ for some strategy $\sigma$ that is coherent and uniform for agent $a$. Coherency is easy: $C_a(\alpha) := \forall v \bigwedge_{s \in S_0} \alpha(v)(s) \in \mathrm{Act}_a$. For uniformity, we use the characterisation in Proposition 1: $U_a(\alpha) := \bigwedge_{s,s' \in S_0} \forall v \left(E^a_{s,s'}(v) \rightarrow (\alpha(v)(s) = \alpha(v)(s'))\right)$ where $E^a_{s,s'}(v)$ is $\forall w(w \preceq v \rightarrow (\widehat{\mathsf{S}}(w)(s) \sim_a \widehat{\mathsf{S}}(w)(s')))$.

The remainder of the proof is by structural induction.

**Inductive hypothesis**. For every ESL-sentence $\varphi$ and BA-iCGS $\mathsf{S}$ one can construct an MSO-formula $\Phi$ such that $(\mathsf{S}, h, \chi) \models \varphi$ if and only if $(\widehat{\mathsf{S}}, \widehat{h}, \widehat{\chi}) \models \Phi$ (for all $h, \chi$).

*Atomic predicate* $\varphi = p$. Define $\Phi$ by $\bigvee_{s_0 \in S_0, s \in \lambda^{-1}(p)} \exists v(\widehat{h}(v) = s_0 \wedge \widehat{S}(v)(s_0) = s)$.

*Boolean operators.* For $\varphi = \neg\varphi_1$ define $\Phi = \neg\Phi_1$; and for $\varphi = \varphi_1 \wedge \varphi_2$ define $\Phi = \Phi_1 \wedge \Phi_2$.

*Strategic operator* $\varphi = \langle\langle x \rangle\rangle \varphi_1$. Define $\Phi$ by $\exists \alpha \bigwedge_{a \in shr(x,\varphi)}(C_a(\alpha) \wedge U_a(\alpha) \wedge \Phi'_1)$, where $\Phi'_1$ is $\Phi_1$ in which $\widehat{\chi}(v)(s)(x)$ is replaced by $\alpha(v)(s)$.

*Binding operator* $\varphi = (x, a)\varphi_1$. Define $\Phi$ by $\Phi_1[\widehat{\chi'} \leftarrow \widehat{\chi}]$ where, writing $\chi'$ for $\chi^a_{\chi(x)}$, the encoding $\widehat{\chi'}$ is definable from $\widehat{\chi}$ as follows: $\widehat{\chi'}(v)(t)(y)$ equals $\widehat{\chi}(v)(t)(y)$ if $y \neq a$, and equals $\widehat{\chi}(v)(t)(x)$ if $y = a$.

*Epistemic operator* $\varphi = \mathbb{K}_a\varphi_1$. The formula $\Phi$ is $\bigwedge_{s,t \in S_0} \forall u \left((\widehat{h}(u) = s \wedge E^a_{s,t}(u)) \rightarrow \Phi_1[\widehat{h'} \leftarrow \widehat{h}]\right)$, where $h' = \mu(t, u)$ (note that $\widehat{h'}$ is definable from $u$ and $t$). The other epistemic operators are treated similarly.

*Next operator* $\varphi = \mathsf{X}\varphi_1$. It is sufficient to note that, writing $h' = \pi(h, \chi|_{\mathrm{Ag}})_{\leq |h|+1}$, the encoding $\widehat{h'}$ is definable from $\widehat{\chi}$ and $\widehat{h}$ as follows: $\widehat{h'}(v) = t$ if $\widehat{h}(u) = t$ and $v = \mathsf{suc}_J(u)$ and $J(a) = \widehat{\chi}(u)(t)(a)$, else $\widehat{h}(v) = \bot$.

*Until operator* $\varphi = \varphi_1 \mathsf{U} \varphi_2$. Note that a) for every $s \in S_0$ there is an MSO-formula $P_s(U, u)$ that says that $U$ is an infinite branch, $u \in U$, and that after $u$ the branch $U$ continues by following the joint full strategy induced by $\widehat{\chi}$ from initial state $s$; and b) $\mu(s, u) \in \mathsf{out}(\mathsf{S}, h)$ can be expressed in terms of $\widehat{h}$ by $\exists x(u \preceq x \wedge \widehat{h}(x) = s)$. Thus, the translation of $\mathsf{F}\varphi_2 \equiv \mathsf{true}\,\mathsf{U}\,\varphi_2$ (the full operator U is similar) is $\bigvee_{s \in S_0} \exists U \exists u(P_s(U, u) \wedge \widehat{h}(u) = s \wedge \exists \widehat{h'} \exists v \in U(u \preceq v \wedge \widehat{h'}(v) = s \wedge \Phi_2[\widehat{h'} \leftarrow \widehat{h}]))$. This completes the induction.

The size of $\Phi$ is polynomial in the size of the input (i.e., $\varphi, \mathsf{S}$). Applying Theorem 7 we get the stated non-elementary upper-bound. This completes the proof of Theorem 6.

**Application to Rational Synthesis**. By the discussion in Section 3.1, we immediately get the first part of the following:

**Corollary 1.** *Rational synthesis for* LTLK *objectives on BA-iCGS is decidable. Moreover, if a given instance returns "yes", then a finite-state Nash equilibrium can be computed.*

For the second part, apply the translation presented above to the ESL-formula $\mathsf{RatSyn}_{\mathsf{true}}(\overline{x})$ and a BA-iCGS $\mathsf{S}$ (say with $|Ag| = n$) to get an MSO-formula $\Phi(\overline{U})$ such that for all strategy profiles $\overline{\sigma}$, we have that $\mathsf{S} \models \mathsf{RatSyn}_{\mathsf{true}}(\overline{x})$ iff $(\widehat{\sigma_1}, \ldots, \widehat{\sigma_n}) \models \Phi$. Now, by Theorem 7 applied to $\Phi$, one can compute regular languages $A_i \subseteq \mathsf{ACT}^*$ such that $\overline{A} \models \Phi$. Since these languages code strategies, we have computed finite-state strategies $\sigma_i$ such that $\mathsf{S} \models \mathsf{RatSyn}_{\mathsf{true}}(\overline{\sigma})$.

## 5 Conclusions

One of the key problems in reasoning about strategic abilities in MAS under incomplete information and perfect recall is that the model checking and synthesis problems are undecidable even for relatively weak logics such as ATL. Yet, MAS applications require specifications that are more expressive than ATL, e.g., capable of expressing solution concepts such as Nash equilibria. Identifying classes of systems for which these two desiderata can be combined remains a challenge. In this paper we have made a contribution towards this aim.

Specifically, we defined ESL, a combination of Strategy Logic and Epistemic Logic. We observed that model checking and synthesis are undecidable under synchronous perfect-recall semantics. However, we showed that a noteworthy subclass of systems, those that admit only broadcast actions, admit decidable model checking and synthesis, and identified tight bounds for the model-checking problem.

We have illustrated the expressivity of the formalism by phrasing rational synthesis under incomplete information, a previously unexplored set-up, as an instance of model checking for ESL. This has the noteworthy consequence that rational synthesis is decidable in the framework. It follows that we can decide expressive strategic properties of rational secret-sharing scenarios like the one presented in Section 3.2 under the assumption of non-randomised strategies. We leave the exploration of other scenarios for future work.

## Acknowledgements

This research was partly supported by EPSRC (grant EP/I00529X), INdAM (grant "Logica e Automi per il Model Checking"), the ANR JCJC (project SVeDaS) and an INdAM Marie Curie fellowship to S. Rubin. The authors thank Benjamin Aminof for fruitful discussions.

[3]Hereafter $\mathsf{tr} : S_0 \times \mathsf{ACT}^* \rightarrow S$ is defined by $\mathsf{tr}(s, \epsilon) = s$ and $\mathsf{tr}(s, vx) = \mathsf{tr}(\mathsf{tr}(s, v), x)$ for $v \in \mathsf{ACT}^*, x \in \mathsf{ACT}$.

## References


[Abraham *et al.*, 2006] I. Abraham, D. Dolev, R. Gonen, and J. Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *PODC'06*, pages 53–62, 2006.

[Abraham *et al.*, 2011] I. Abraham, L. Alvisi, and J.Y. Halpern. Distributed computing meets game theory: combining insights from two fields. *SIGACT News*, 42(2):69–76, 2011.

[Alur *et al.*, 2002] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[Belardinelli and Lomuscio, 2009] F. Belardinelli and A. Lomuscio. Quantified epistemic logics for reasoning about knowledge in multi-agent systems. *Art. Intell.*, 173(9-10):982–1013, 2009.

[Belardinelli *et al.*, 2017] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with imperfect information and public actions. In *AAMAS'17*, pages 1268–1276, 2017.

[Belardinelli, 2014] F. Belardinelli. Reasoning about knowledge and strategies: Epistemic strategy logic. In *SR'14*, pages 27–33, 2014.

[Berthon *et al.*, 2017a] R. Berthon, B. Maubert, and A. Murano. Decidability results for ATL* with imperfect information and perfect recall. In *AAMAS'17*, pages 1250–1258, 2017.

[Berthon *et al.*, 2017b] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Vardi. Hierarchical strategic reasoning. In *LICS'17*, 2017. To appear.

[Bulling and Jamroga, 2014] N. Bulling and W. Jamroga. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *JAAMAS*, 28(3):474–518, 2014.

[Čermák *et al.*, 2014] P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *CAV'14*, pages 524–531, 2014.

[Cermák, 2014] P. Cermák. A model checker for strategy logic. Master's thesis, Dept. of Comp., Imperial, 2014.

[Clarke *et al.*, 2002] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking.* MIT Press, 2002.

[Dima and Tiplea, 2011] C. Dima and F.L. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[Fagin *et al.*, 1995] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT, 1995.

[Fisman *et al.*, 2010] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *TACAS'10*, pages 190–204, 2010.

[Gutierrez *et al.*, 2016] J. Gutierrez, G. Perelli, and M. Wooldridge. Imperfect information in reactive modules games. In *KR'15*, pages 390–400, 2016.

[Gutierrez *et al.*, 2017] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Reasoning about equilibria in game-like concurrent systems. *Ann. Pure Appl. Log.*, 168(2):373–403, 2017.

[Halpern and Vardi, 1989] J. Halpern and M. Vardi. The complexity of reasoning about knowledge and time. I. Lower bounds. *J. Comp. Sys. Sci.*, 38(1):195–237, 1989.

[Halpern *et al.*, 2003] J. Halpern, R. van der Meyden, and M. Y. Vardi. Complete axiomatisations for reasoning about knowledge and time. *SIAM Comp.*, 33(3):674–703, 2003.

[Hoek and Wooldridge, 2003] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[Huang and Meyden, 2014] X. Huang and R. van der Meyden. A temporal logic of strategic knowledge. In *KR'14*, 2014.

[Jamroga and Dix, 2006] W. Jamroga and J. Dix. Model checking abilities under incomplete information is indeed $\Delta_p^2$-complete. In *EUMAS'06*, pages 14–15, 2006.

[Jamroga and van der Hoek, 2004] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fund. Inf.*, 62:1–35, 2004.

[Jennings and others, 2001] N. Jennings et al. Automated negotiation: Prospects, methods, and challenges. *Int. J. of Gr. Dec and Neg.*, 10(199-215):199–215, 2001.

[Kupferman *et al.*, 2016] O. Kupferman, G. Perelli, and M.Y. Vardi. Synthesis with rational environments. *Ann. Math. Art. Int.*, 78(1):3–20, 2016.

[Lomuscio *et al.*, 2000] A. Lomuscio, R. van der Meyden, and M. Ryan. Knowledge in multi-agent systems: Initial configurations and broadcast. *ACM Trans. Comp. Log.*, 1(2):246–282, 2000.

[Meyden and Shilov, 1999] R. van der Meyden and H. Shilov. Model checking knowledge and time in systems with perfect recall. In *FST&TCS'99*, pages 432–445, 1999.

[Meyden and Wilke, 2005] R. van der Meyden and T. Wilke. Synthesis of distributed systems from knowledge-based specifications. In *CONCUR'05*, pages 562–576, 2005.

[Meyer and Hoek, 1995] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, 1995.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comp. Log.*, 15(4):34:1–34:47, 2014.

[Rabin, 1969] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. of the AMS*, 141:1–35, 1969.

[Wooldridge *et al.*, 2016] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In *AAAI'16*, pages 4184–4191, 2016.

Information and Computation 254 (2017) 195–216

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

# First-cycle games ☆

Benjamin Aminof [a], Sasha Rubin [b,*,1]

[a] Technische Universität Wien, Austria
[b] Università degli Studi di Napoli "Federico II", Italy

ARTICLE INFO

*Article history:*
Received 21 November 2014
Available online 4 November 2016

*Keywords:*
Graph games
Cycle games
Memoryless determinacy
Parity games
Mean-payoff games
Energy games

ABSTRACT

First-cycle games (FCG) are played on a finite graph by two players who push a token along the edges until a vertex is repeated, and a simple cycle is formed. The winner is determined by some fixed property $Y$ of the sequence of labels of the edges (or nodes) forming this cycle. These games are intimately connected with classic infinite-duration games such as parity and mean-payoff games. We initiate the study of FCGs in their own right, as well as formalise and investigate the connection between FCGs and certain infinite-duration games.

We establish that (for efficiently computable $Y$) the problem of solving FCGs is Pspace-complete; we show that the memory required to win FCGs is, in general, $\Theta(n)!$ (where $n$ is the number of nodes in the graph); and we give a full characterisation of those properties $Y$ for which all FCGs are memoryless determined.

We formalise the connection between FCGs and certain infinite-duration games and prove that strategies transfer between them. Using the machinery of FCGs, we provide a recipe that can be used to very easily deduce that many infinite-duration games, e.g., mean-payoff, parity, and energy games, are memoryless determined.



## 1. Introduction

Infinite-duration games are studied in computer science in the context of decidability of logical theories as well as verification and synthesis in formal methods. In particular, reactive systems, which consist of an ongoing interaction between a program and its environment, may be formalised as a game played on a graph, by two players, who push a token along the edges of the graph, resulting in an infinite path, called a play. The winner of the play is determined by some winning condition. For example, in (one version of) mean-payoff games each edge in the graph carries a real number, called a weight, and Player 0 wins the play if and only if the limit-supremum of the running averages of the weights is positive; and otherwise Player 1 wins. Other types of games from the formal verification literature are reachability games, Büchi games, parity games, Muller games, energy games, etc.

Intuitively, certain games on finite graphs can be won using greedy reasoning. For instance, to win a mean-payoff game, it is sufficient for Player 0 to ensure that the average weight of each cycle that is formed is positive. This, in turn, can be ensured by enforcing the average weight of the *first cycle* formed to be positive (because Player 0 could then "forget" that the cycle was formed, and play another cycle with positive average weight, and so on). Thus, in some cases, one can

☆ Some of the results in this paper were reported in the Proceedings of the Second International Workshop on Strategic Reasoning (SR), April 2014.

* Corresponding author.
*E-mail addresses:* benj@forsyte.at (B. Aminof), rubin@unina.it (S. Rubin).

[1] Marie Curie fellow of the Istituto Nazionale di Alta Matematica.

http://dx.doi.org/10.1016/j.ic.2016.10.008
0890-5401/

reduce reasoning about infinite-duration games to *first-cycle games*, i.e., games in which the winner is determined by the first cycle on the play. Such reasoning appears in [6], where first-cycle mean-payoff games were defined and used to prove that mean-payoff games can be won using memoryless strategies (i.e., the next move of a player does not depend on the full history up till now, but only on the current node of the play). Memoryless strategies are extremely useful, e.g., they are used to prove deep results in the theory of automata (e.g., Rabin's theorem that automata on infinite trees can be complemented), and to prove upper bounds on the complexity of solving certain classes of games (e.g., that solving parity games is in NP $\cap$ co-NP).

In this paper we study first-cycle games in their own right and formalise the greedy reasoning above which connects first-cycle games with certain infinite-duration games. We now discuss our main contributions.

**First-cycle games.** We define first-cycle games (FCGs). These games are played on a finite graph (called an arena) by two players who push a token along the edges of the graph until a cycle is formed. Player 0 wins the play if the sequence of labels of the edges (or nodes) of the cycle is in some fixed set $Y$, and otherwise Player 1 wins. The set $Y$ is called a cycle property, and we say that a cycle satisfies $Y$ if its sequence of labels is in $Y$. For example, if every vertex is labelled by an integer priority, and $Y = \mathsf{cyc\text{-}Parity}$ comprises sequences whose largest priority is even, then a cycle satisfies cyc-Parity if the largest priority occurring on the cycle is even. For a fixed cycle property $Y$, we write $\textsc{Fcg}(Y)$ for the family of games over all possible arenas with this winning condition.

**Complexity and memory requirements.** We give a simple example showing that first cycle games (FCGs) are not necessarily memoryless determined. We then show that, for a graph with $n$ nodes, whereas no winning strategy needs more than $n!$ memory (since this is enough to remember the whole history of the game), some winning strategies require at least $(\frac{n-1}{3})!$ memory (Proposition 1, Page 201). We analyse the complexity of solving FCGs and show that it is Pspace-complete. More specifically, we show that if one can decide in Pspace whether a given cycle satisfies the property $Y$, then solving the games in $\textsc{Fcg}(Y)$ is in Pspace; and that there is a trivially computable cycle property $Y$ for which solving the games in $\textsc{Fcg}(Y)$ is Pspace-hard (Theorem 1, Page 202).

**First-cycle games and infinite-duration games (Section 5).** The central object that connects cycle games with infinite-duration games is the *cycles-decomposition* of a path (used for example by [12] to derive the value of a mean-payoff game). Informally, a path is decomposed by pushing the edges of the path onto a stack and, as soon as a cycle is detected in the stack, the cycle is output, popped, and the algorithm continues. This decomposes all but finitely many edges of the path into a sequence of simple cycles.

In order to connect FCGs with infinite-duration games we define the following notion: a winning condition $W$ (such as the parity winning condition) is *$Y$-greedy on arena $\mathcal{A}$* if, in the game on arena $\mathcal{A}$ with winning condition $W$, Player 0 is guaranteed to win by ensuring that every cycle in the cycles-decomposition of the play satisfies $Y$, and Player 1 is guaranteed to win by ensuring that every cycle in the cycles-decomposition does not satisfy $Y$ (Definition 5, Page 208). We prove a *Strategy Transfer Theorem:* if $W$ is $Y$-greedy on $\mathcal{A}$ then the winning regions in the following two games on arena $\mathcal{A}$ coincide, and memoryless winning strategies transfer between them: the infinite-duration game with winning condition $W$, and the FCG with cycle property $Y$ (Theorem 7, Page 209).

To illustrate the usefulness of being $Y$-greedy, we instantiate the definition to well-studied infinite-duration games such as parity, mean-payoff, and energy games.

**Memoryless determinacy of first-cycle games.** We next address the fundamental question: for which cycle properties $Y$ is every game in $\textsc{Fcg}(Y)$ memoryless determined (i.e., no matter the arena)? We provide sufficient and necessary conditions for all games in $\textsc{Fcg}(Y)$ to be memoryless determined (Theorem 6, Page 208). Although applying this characterisation may not be hard, it involves reasoning about arenas, which is sometimes inconvenient. Therefore, we also provide the following easy-to-check conditions on $Y$ that ensure memoryless determinacy for all games in $\textsc{Fcg}(Y)$ (Theorem 9, Page 210): $Y$ is closed under cyclic permutations (i.e., if $ab \in Y$ then $ba \in Y$), and both $Y$ and its complement are closed under concatenation (a set of strings $X$ is closed under concatenation if $a, b \in X$ implies $ab \in X$). We demonstrate the usefulness of these conditions by observing that natural cycle properties are easily seen to satisfy them, e.g., cyc-Parity, $\mathsf{cyc\text{-}MeanPayoff}_\nu$ (which states that the limsup average of the weights is at most $\nu$), and cyc-Energy (which states that the sum of the weights is positive).

**Easy-to-follow recipe.** We integrate the previous results by providing an easy-to-follow recipe that allows one to deduce memoryless determinacy of all classic games that are memoryless determined and many others besides (Section 8). The recipe states that, given a winning condition $W$, first "finitise" $W$ to get a cycle property $Y$ that is closed under cyclic permutations, then prove that $W$ is $Y$-greedy on the arenas of interest (usually all arenas), and finally, either show that both $Y$ and its complement are closed under concatenation, or show that $W$ is prefix-independent. For example, if $W$ is the parity winning condition (i.e., the largest priority occurring infinitely often is even), the natural "finitisation" of $W$ is the cycle property $Y = \mathsf{cyc\text{-}Parity}$ (i.e., sequences of priorities whose largest priority is even), and it is almost completely trivial to apply the recipe in this case.

**Related work.** The relationship of our work with that in [6] is as follows. First, our paper deals with qualitative games (i.e., a play is either won or lost) whereas [6] consider quantitative (i.e., a play is assigned a real number, which Player 0 wants to minimize and Player 1 wants to maximize) mean-payoff games. Their main result states that mean-payoff games

are memoryless determined. However, they simultaneously prove that first-cycle games with $Y = \text{cyc-MeanPayoff}_v$ are memoryless determined. We generalise (in the qualitative setting) both of these facts and their proofs.

Referring to their proofs, [6, Page 111] state: "Our proofs are roundabout, we use the infinite game F to establish facts about the finite game G and vice versa. Perhaps more direct proofs would be desirable." In contrast, the proof of the characterisation of memoryless determined FCGs (Theorems 4, 5 and 6) is direct, and does not go through infinite-duration games. Nonetheless, we use their idea of inducting on the choice nodes of the players, and of employing "reset" nodes in the arena.

We briefly discuss other related work. We point out (in Theorem 3, Page 204) that first-cycle games are not necessarily memoryless determined, even if the cycle property $Y$ is closed under cyclic permutations contrary to the claim in [4, Page 370]. Our work thus also supplies a correct proof Lemma 4 in [5] which relied on this incorrect claim (see Remark 2, Page 210). Conditions that ensure (or characterise) which winning conditions always admit memoryless strategies appear in [3,8,10]. However, none of these exploit the connection to first-cycle games. For example, [8] give a full characterization of winning conditions for which all infinite-duration games are memoryless determined. Unlike our framework, the main objects of study in their work are winning conditions (i.e., languages of infinite sequences of labels) and one cannot use their results to reason about cycles in an arena since every cycle in an arena induces a cycle in the sequences of labels, but not vice versa. Their characterisation of those winning conditions which admit memoryless determined games is very "infinite" in nature, as it involves reasoning about preference relations among infinite words, and their properties concerning all infinite subsets of words recognizable by nondeterministic automata. On the other hand, their result, that if all solitaire games with a given winning condition are memoryless determined then so are all the two player games, provides a much more useful tool.

The present paper differs from the preliminary version of this work [1] mainly in the following respects: we have re-organised some of the definitions, supplied all proofs, established a full characterisation of those cycle properties $Y$ such that every FCG over $Y$ and $Y$-greedy games are memoryless determined, and extended the easy-to-follow recipe to include the case that the winning condition is prefix-independent.

## 2. Games

In this paper all games are two-player turn-based games of perfect information played on finite graphs. The players are called Player 0 and Player 1.

**Arenas.** An *arena* is a labelled directed graph $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$ where

1. $V_0$ and $V_1$ are disjoint sets of *vertices* (alternatively, *nodes*) of Player 0 and Player 1, respectively; the set of vertices of the arena is $V := V_0 \cup V_1$, and is assumed to be finite and non-empty.
2. $E \subseteq V \times V$ is a set of *edges* with no dead-ends (i.e., for every $v \in V$ there is some edge $(v, w) \in E$).
3. $\mathbb{U}$ is a set of possible *labels* (typical choices for $\mathbb{U}$ are $\mathbb{Q}$ and $\mathbb{N}$).
4. $\lambda : E \to \mathbb{U}$ is a *labelling function* (which will be used by the winning condition).

A *sub-arena* of $\mathcal{A}$ is an arena of the form $(V_0, V_1, E', \mathbb{U}, \lambda')$ where $E' \subseteq E$ and $\lambda'(e) = \lambda(e)$ for $e \in E'$, i.e., it may have fewer edges. If $V_0 = \emptyset$ or $V_1 = \emptyset$ then $\mathcal{A}$ is called a *solitaire arena*.

**Remark 1.** Note that all the results in this paper also hold for vertex labelled arenas, by labelling every edge $(v, w)$ by the label of its source $v$. In particular, we use vertex labelling in some of our examples, with this transformation in mind. Also note that transforming a vertex labelled arena to an edge-labelled one can be done by inserting an intermediate node in the middle of every edge, and giving it the edge label. However, since this transformation changes the structure of the arena, some properties that hold for vertex labelled arenas do not always transfer to edge-labelled ones.

**Sequences.** Let $\mathbb{N}$ denote the positive integers. The $i$th element (for $i \in \mathbb{N}$) in a sequence $u$ is denoted $u_i$. The *length* of $u$, denoted $|u|$, is the cardinality of the sequence $u$. For $1 \le i \le j \le |u|$, write $u[i, j]$ for the sub-sequence $u_i u_{i+1} \cdots u_j$. The empty sequence is denoted $\epsilon$. By convention, $u[i, j] = \epsilon$ if $j < i$. The set of infinite sequences over alphabet $X$ is written $X^\omega$, the set of finite sequences is written $X^*$. We write concatenation as $u \cdot v$ or simply as $uv$.

**Paths, simple paths, and node-paths.** For an edge $e = (v, w)$ we call $v$ the *start* and $w$ the *end* of $e$, and write $\mathsf{start}(e) := v$ and $\mathsf{end}(e) := w$; we say that $v$ and $w$ *appear on the edge* $e$. A *path* $\pi$ in $\mathcal{A}$ is a finite or infinite sequence of edges $\pi_1 \pi_2 \cdots \in E^* \cup E^\omega$ such that $\mathsf{end}(\pi_i) = \mathsf{start}(\pi_{i+1})$ for $1 \le i < |\pi|$. The set of paths in $\mathcal{A}$ is denoted $paths(\mathcal{A})$. We extend start and end to paths in the natural way: $\mathsf{start}(\pi) := \mathsf{start}(\pi_1)$, and if $\pi$ is of length $\ell \in \mathbb{N}$ then $\mathsf{end}(\pi) := \mathsf{end}(\pi_\ell)$. We say that vertex $v$ *appears on the path* $\pi$ if $v$ appears on some edge $\pi_i$ of $\pi$. We extend $\lambda$ from edges to paths point-wise: $\lambda(\pi)$ is defined to be the string of labels $\lambda(\pi_1)\lambda(\pi_2)\cdots$. A path $\pi$ is called *simple* if $\mathsf{start}(\pi_i) = \mathsf{end}(\pi_j)$ implies $i = j + 1$.

A sequence of nodes $v \in V^* \cup V^\omega$ is called a *node-path* if $(v_i, v_{i+1}) \in E$ for all $i$. If $\pi \in E^* \cup E^\omega$ is a path then $\mathsf{nodes}(\pi)$ is a node-path, where $\mathsf{nodes}(\pi)$ is defined to be the sequence $\mathsf{start}(\pi_1)\mathsf{start}(\pi_2)\cdots$ if $\pi$ is infinite, and $\mathsf{start}(\pi_1)\cdots\mathsf{start}(\pi_{|\pi|})\mathsf{end}(\pi_{|\pi|})$ if it is finite. Every node-path is either a singleton sequence $v$ or of the form $\mathsf{nodes}(\pi)$

for some path $\pi$. For a finite non-empty sequence $u$ of vertices (such as a finite node-path), we overload notation and write $\mathsf{end}(u)$ for the last node in $u$.

**Histories and strategies.** A strategy for a player is a function that tells the player what node to move to given the history, i.e., the sequence of nodes visited so far. Define the set $H_\sigma(\mathcal{A})$ of *histories* for Player $\sigma \in \{0,1\}$ by $H_\sigma(\mathcal{A}) := \{u \in V^*V_\sigma : (u_n, u_{n+1}) \in E, 1 \le n < |u|\}$. In other words, $H_\sigma(\mathcal{A})$ is the set of node-paths ending in $V_\sigma$. A *strategy for Player* $\sigma$ is a function $S : H_\sigma(\mathcal{A}) \to V$ such that $(\mathsf{end}(u), S(u)) \in E$ for all $u \in H_\sigma(\mathcal{A})$. A strategy $S$ for Player $\sigma$ is *memoryless* if $S(u) = S(u')$ for all $u, u' \in H_\sigma(\mathcal{A})$ with $\mathsf{end}(u) = \mathsf{end}(u')$. Hence, we usually consider a memoryless strategy as a function $S : V_\sigma \to V$. A node-path $u \in V^* \cup V^\omega$ is *consistent* with a strategy $S$ for Player $\sigma$ if whenever $u_n \in V_\sigma$ (for $n < |u|$) then $u_{n+1} = S(u[1,n])$. A path $\pi$ is *consistent* with a strategy $S$ if $\mathsf{nodes}(\pi)$ is consistent with $S$.

A strategy $S$ for Player $\sigma$ is *generated by a Mealy machine* $\langle M, m_I, \delta, \mu\rangle$ if there exists a finite set $M$ of *memory states*, an *initial state* $m_I \in M$, a *memory update* function $\delta : V \times M \to M$, and a *next-move function* $\mu : V_\sigma \times M \to V$, such that for a history $u = u_1u_2\cdots u_l \in H_\sigma(\mathcal{A})$ we have $S(u) = \mu(u_l, m_l)$ where $m_l$ is defined inductively by $m_1 = m_I$ and $m_{i+1} = \delta(u_i, m_i)$. A strategy $S$ is *finite-memory* if it is generated by some Mealy machine. A strategy $S$ *uses memory at most* $k$ if it is generated by some Mealy machine with $|M| \le k$, and it *uses memory at least* $k$ if every Mealy machine generating $S$ has $|M| \ge k$.

**Notational abuse.** We sometimes, for a path $\pi \in E^*$, write $S(\pi)$ instead of $S(\mathsf{nodes}(\pi))$.

**Plays.** An infinite path in $\mathcal{A}$ is called a *play*.[2] We denote the set of all plays of $\mathcal{A}$ by $plays(\mathcal{A})$. For a node $v \in V$, and strategy $S$, we write $plays_{\mathcal{A}}(S, v)$ for the set of plays $\pi$ that start with $v$ and are consistent with $S$; we write $reach_{\mathcal{A}}(S, v)$ for the set of vertices in $V$ that appear on the plays in $plays_{\mathcal{A}}(S, v)$. We may drop the subscript $\mathcal{A}$ when the arena is clear from the context.

**Games and winning.** A *game* is a pair $(\mathcal{A}, O)$ consisting of an arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$ and an *objective* $O \subseteq plays(\mathcal{A})$. Objectives are usually determined by winning conditions or cycle properties (defined later). A play $\pi$ in a game $(\mathcal{A}, O)$ is *won by Player* 0 if $\pi \in O$, and otherwise it is *won by Player* 1. A strategy $S$ for Player $\sigma$ is *winning from* a node $v \in V$ (in the game $(\mathcal{A}, O)$) if every play in $plays_{\mathcal{A}}(S, v)$ is won by Player $\sigma$. If Player $\sigma$ has a winning strategy from $v$ we say that Player $\sigma$ *wins from* $v$. A game $(\mathcal{A}, O)$ is said to be *determined* if, for every $v \in V$, one of the players wins from $v$.

The *winning region (resp. memoryless winning region)* of Player $\sigma$ is the set of vertices $v \in V$ such that Player $\sigma$ has a winning strategy (resp. memoryless winning strategy) from $v$. We denote the winning region in the game $(\mathcal{A}, O)$ of Player $\sigma$ by $\mathsf{WR}^\sigma(\mathcal{A}, O)$.

**Solitaire games and memoryless strategies.** If either $V_0$ or $V_1$ is empty, then the game $(\mathcal{A}, O)$ is called a *solitaire game*.

A simple property of memoryless strategies that is often used is the following. In a game over an arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$, every memoryless strategy $S$ for Player $\sigma$ induces a sub-arena $\mathcal{A}^{|S} = (V_0, V_1, E^{|S}, \mathbb{U}, \lambda^{|S})$, obtained by deleting all edges $(v, w) \in E$ where $v \in V_\sigma$ and $w \neq S(v)$. Observe that in $\mathcal{A}^{|S}$ all Player $\sigma$ nodes have exactly one outgoing edge (namely the edge specified by $S$), and thus Player $\sigma$ has no choices to make in this arena. For this reason, many times one considers the solitaire arena $\mathcal{A}^{\|S}$ in which all the nodes are assigned to Player $1-\sigma$, i.e., $\mathcal{A}^{\|S} = (V_0^{\|S}, V_1^{\|S}, E^{|S}, \mathbb{U}, \lambda^{|S})$, where $V_\sigma^{\|S} = \emptyset$, and $V_{1-\sigma}^{\|S} = V$. The main connection between $\mathcal{A}, \mathcal{A}^{|S}$ and $\mathcal{A}^{\|S}$ is that every path in $\mathcal{A}^{|S}$ ($\mathcal{A}^{\|S}$) is a path in $\mathcal{A}$ that is also consistent with $S$, and vice versa. We can thus reason about paths in $\mathcal{A}^{|S}$ (or $\mathcal{A}^{\|S}$) instead of paths in $\mathcal{A}$ that are consistent with $S$.

**Memoryless determinacy.** A game is *memoryless from* $v$ if the player that wins from $v$ has a memoryless strategy that is winning from $v$.

A game is *point-wise memoryless for Player* $\sigma$ if the memoryless winning region for Player $\sigma$ coincides with the winning region for Player $\sigma$. A game is *uniform memoryless for Player* $\sigma$ if there is a memoryless strategy for Player $\sigma$ that is winning from every vertex in that player's winning region.

A game is *point-wise memoryless determined* if it is determined and it is point-wise memoryless for both players. A game is *uniform memoryless determined* if it is determined and uniform memoryless for both players.

**Winning conditions.** A *winning condition* is a set $W \subseteq \mathbb{U}^\omega$. If $W$ is a winning condition and $\mathcal{A}$ is an arena, the objective $O^{\mathcal{A}}(W)$ induced by $W$ is defined as follows: $O^{\mathcal{A}}(W) = \{\pi \in plays(\mathcal{A}) : \lambda(\pi) \in W\}$. For ease of readability we may drop the superscript $\mathcal{A}$ and write $O(W)$ instead of $O^{\mathcal{A}}(W)$.

Here are some standard winning conditions:

- The *parity condition* consists of those infinite sequences $c_1c_2\cdots \in \mathbb{N}^\omega$ such that the largest label occurring infinitely often is even.
- For $\nu \in \mathbb{R}$, the *$\nu$-mean-payoff condition* consists of those infinite sequences $c_1c_2\cdots \in \mathbb{R}$ such that $\limsup_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} c_i$ is at most $\nu$.

[2] For ease of presentation, we consider plays of both finite- and infinite-duration games to be infinite. Obviously, in games finite-duration, games such as FCGs, the winner is determined by a finite prefix of the play, and the moves after this prefix are immaterial.

- The *energy condition*, for a given *initial credit* $r \in \mathbb{N}$, consists of those infinite sequences $c_1 c_2 \cdots \in \mathbb{Z}^\omega$ such that $r + \sum_{i=1}^{k} c_i \geq 0$ for all $k \geq 1$.
- The *energy-parity condition (for a given initial credit r)* is defined as consisting of $(c_1, d_1)(c_2, d_2) \cdots \in \mathbb{N} \times \mathbb{Z}$ such that $c_1 c_2 \cdots$ satisfies the parity condition and $d_1 d_2 \cdots$ satisfies the energy condition with initial credit $r$.

## 3. Cycles and games

In this section we define the cycles-decomposition of a path, as well as the first-cycle game, and the infinite-duration all-cycles game.

**Cycles-decomposition.** A *cycle* in an arena $\mathcal{A}$ is a finite path $\pi$ such that $\mathsf{start}(\pi) = \mathsf{end}(\pi)$. Note that, like paths, cycles are ordered. Hence, the cycles $(v_1, v_2)(v_2, v_1)$ and $(v_2, v_1)(v_1, v_2)$ are not identical.

**Algorithm 1** CycDec$(s, \pi)$.

**Require:** $s$ is a finite (possibly empty) simple path ▷ initial stack content
**Require:** $\pi$ is a finite or infinite path $\pi_1 \pi_2 \cdots$ ▷ the path to decompose
**Require:** If $s$ is non-empty then $\mathsf{end}(s) = \mathsf{start}(\pi)$ ▷ $s\pi$ must form a path
  $step = 1$
  **while** $step \leq |\pi|$ **do** ▷ Start a step
    Append $\pi_{step}$ to $s$ ▷ Push current edge into stack
    Say $s = e_1 e_2 \cdots e_m$
    **if** $\exists i : e_i e_{i+1} \cdots e_m$ is a cycle **then** ▷ If stack has a cycle
      **Output** $e_i e_{i+1} \cdots e_m$ ▷ output the cycle
      $s := e_1 \cdots e_{i-1}$ ▷ Pop the cycle from the stack
    **end if**
    $step := step + 1$ ▷ advance to next input edge
  **end while**

The code appearing in Algorithm 1 defines an algorithm CycDec that takes as input a (usually empty) simple path $s$, which is treated as the initial contents of a stack, and a path $\pi$ (finite or infinite). At step $j \geq 1$, the edge $\pi_j$ is pushed onto the stack and if, for some $k$, the top $k$ edges on the stack form a cycle, this cycle is output, then popped, and the procedure continues to step $j + 1$.

Note that CycDec may take a finite path $\pi$ and non-empty stack $s$ as input. Moreover, it halts if and only if $\pi$ is a finite path.

The sequence of cycles output by this procedure when input the empty stack $s = \epsilon$ and path $\pi$, denoted $cycles(\pi)$, is called the *cycles-decomposition of* $\pi$. The *first cycle of* $\pi$, written $first(\pi)$, is the first cycle in $cycles(\pi)$. For example, if

$$\pi = (v, w)(w, x)(x, w)(w, v)(v, s)(s, x)[(x, y)(y, z)(z, x)]^\omega,$$

then

$$cycles(\pi) = (w, x)(x, w), (v, w)(w, v), (x, y)(y, z)(z, x), (x, y)(y, z)(z, x), \ldots,$$

and the first cycle of $\pi$ is $(w, x)(x, w)$.

It is easy to see that, if the algorithm CycDec is run on a path $\pi$ in an arena with $n$ nodes, then at most $n - 1$ edges of $\pi$ are pushed but never popped (like the edges $(v, s)$ and $(s, x)$ in the example above).[3] So we have:

**Lemma 1.** *For every path $\pi$ in arena $\mathcal{A}$ with vertex set $V$, there are at most $|V| - 1$ indices $i$ such that $\pi_i$ does not appear in any of the cycles in $cycles(\pi)$.*

Given a play $\pi$ in $\mathcal{A}$, an initial stack content $s$, and $i \geq 0$, we write $stack^i(s, \pi)$ for the contents of the stack of algorithm CycDec$(s, \pi)$ at the beginning of step $i + 1$ (i.e., just before the edge $\pi_{i+1}$ is pushed). Note that if $stack^i(s, \pi)$ is not empty then it ends with the vertex $\mathsf{start}(\pi_{i+1})$, whether or not a cycle was popped during step $i$. For $i \geq 1$, we define $cycle^i(s, \pi)$ to be the cycle output by the algorithm CycDec$(s, \pi)$ during step $i$, or $\epsilon$ if no cycle was output at step $i$. We may drop $s$ when $s = \epsilon$, and we may drop $i$ when $i = |\pi|$. For instance, $stack(\pi)$ is the stack content at the end of the algorithm CycDec$(\epsilon, \pi)$ for a finite $\pi$; and $cycle^i(\pi)$ is the cycle output during step $i$ of the algorithm CycDec$(\epsilon, \pi)$.

Given that, for every $i \geq 1$, the behaviour of the algorithm CycDec$(s, \pi)$ from the end of step $i$ onwards is completely determined by $stack^i(s, \pi)$, and the suffix $\pi_{i+1} \pi_{i+2} \ldots$ of $\pi$, the following lemma is immediate:

**Lemma 2.** *Let $\pi$ be a play in $\mathcal{A}$, let $i \geq 0$, let $\mathrm{w} = stack^i(\pi)$, and let $\pi' = \pi_{i+1} \pi_{i+2} \ldots$ and $\pi'' = \mathrm{w} \cdot \pi'$. Then for every $j \geq 0$ we have that*

[3] As we show in Section 8, this allows one to reason, for instance, about the initial credit problem for energy games.

1. $stack^{i+j}(\pi) = stack^{j}(\mathrm{w}, \pi') = stack^{|\mathrm{w}|+j}(\pi'')$, *and*
2. $cycle^{i+j}(\pi) = cycle^{j}(\mathrm{w}, \pi') = cycle^{|\mathrm{w}|+j}(\pi'')$.

*Furthermore, for every* $0 \leq l \leq |\mathrm{w}|$, *we have that* $stack^{l}(\pi'') = \mathrm{w}_1 \dots \mathrm{w}_l$, *and* $cycle^{l}(\pi'') = \epsilon$.

**Cycle properties.** For a given $\mathbb{U}$, a *cycle property* is a set $Y \subseteq \mathbb{U}^*$, used later on to define winning conditions for games.[4] Here are some cycle properties that we refer to in the rest of the article:

1. Let cyc-EvenLen be those sequences $c_1 c_2 \cdots c_k \in \mathbb{U}^*$ such that $k$ is even.
2. Let cyc-Parity be those sequences $c_1 \cdots c_k \in \mathbb{N}^*$ such that $\max_{1 \leq i \leq k} c_i$ is even.
3. Let cyc-Energy be those sequences $c_1 \cdots c_k \in \mathbb{Z}^*$ such that $\sum_{i=1}^{k} c_i \geq 0$.
4. Let cyc-GoodForEnergy be those sequences $(c_1, d_1) \cdots (c_k, d_k) \in (\mathbb{N} \times \mathbb{Z})^*$ such that $\sum_{i=1}^{k} d_i > 0$, or both $\sum_{i=1}^{k} d_i = 0$ and $c_1 \cdots c_k \in$ cyc-Parity.
5. Let cyc-MeanPayoff$_\nu$ (for $\nu \in \mathbb{R}$) be those sequences $c_1 \cdots c_k \in \mathbb{R}^*$ such that $\frac{1}{k} \sum_{i=1}^{k} c_i \leq \nu$.
6. Let cyc-MaxFirst be those sequences $c_1 \cdots c_k \in \mathbb{N}^*$ such that $c_1 \geq c_i$ for all $i$ with $1 \leq i \leq k$.
7. Let cyc-EndsZero be those sequences $c_1 \cdots c_k \in \mathbb{N}_0^*$ such that $c_k = 0$.

If $Y \subseteq \mathbb{U}^*$ is a cycle property, write $\neg Y$ for the cycle property $\mathbb{U}^* \setminus Y$. We will usually use $Y$ to define goals for Player 0 (hence Player 1's goals would be naturally associated with $\neg Y$). It is convenient to define $Y^0 = Y$, and $Y^1 = \neg Y$. This allows us to refer to the goals of Player $\sigma$ in terms of $Y^\sigma$.

We isolate two important classes of cycle properties (the first is inspired by [4]):

1. Say that $Y$ is *closed under cyclic permutations* if $ab \in Y$ implies $ba \in Y$, for all $a \in \mathbb{U}, b \in \mathbb{U}^*$.
2. Say that $Y$ is *closed under concatenation* if $a \in Y$ and $b \in Y$ imply that $ab \in Y$, for all $a, b \in \mathbb{U}^*$.

Note, for example, that the cycle properties 1–5 above are closed under cyclic permutations and concatenation; and that ¬cyc-EvenLen is closed under cyclic permutations but not under concatenation.

If $\mathcal{A}$ is an arena with labelling $\lambda$, and $C$ is a cycle in $\mathcal{A}$, we say that a *cycle* $C$ *satisfies* $Y$ if $\lambda(C) \in Y$.

**First-cycle games and all-cycles games.**

For arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$ and cycle property $Y \subseteq \mathbb{U}^*$, we define two objectives (subsets of $plays(\mathcal{A})$):

1. $\pi \in O^{\mathcal{A}}_{\mathsf{first}}(Y)$ if $first(\pi)$ satisfies $Y$.
2. $\pi \in O^{\mathcal{A}}_{\mathsf{all}}(Y)$ if every cycle in $cycles(\pi)$ satisfies $Y$.

To ease readability, we drop the superscript $\mathcal{A}$ when the arena is clear from the context and write, for example, $O_{\mathsf{all}}(Y)$ instead of $O^{\mathcal{A}}_{\mathsf{all}}(Y)$.

The game $(\mathcal{A}, O_{\mathsf{first}}(Y))$ is called a *first-cycle game (over* $Y$*)*, and the family of all first-cycle games over $Y$ (i.e., taking all possible arenas) is denoted $\textsc{Fcg}(Y)$. Similarly, we write $\textsc{Acg}(Y)$ for the family of *all-cycles games* over $Y$. For instance, $\textsc{Fcg}$(cyc-Parity) consists of those games such that Player 0 wins iff the largest label occurring on the first cycle is even.

A game $(\mathcal{A}, O)$ is *finitary* if every play is already won after a finite number of steps, i.e., for every $\pi \in plays(\mathcal{A})$ there exists $K_\pi \in \mathbb{N}$ such that, if Player $\sigma$ wins the play $\pi$, then Player $\sigma$ also wins every play of $\mathcal{A}$ starting with the prefix $\pi[1, K_\pi]$. Clearly FCGs are finitary since the winner is already determined by the first-cycle of the play (recall that arenas are finite). A basic result states that finitary games (of perfect information) are determined.[5] We recap the proof because we use it in Theorem 1 to determine the computational complexity of deciding which player has a winning strategy in a given FCG.

**Lemma 3.** *Every finitary game* $(\mathcal{A}, O)$ *is determined.*

**Proof.** Suppose $(\mathcal{A}, O)$ is finitary. Fix a starting node $v$ and note that the set of finite node-paths starting in $v$ form a tree, which we call the *game-tree*. Plays in $\mathcal{A}$ correspond to (infinite) branches in the game-tree. Prune every branch $\pi$ of the game-tree at its $K_\pi$th node to get the pruned game-tree $T$. By Kőnig's Tree Lemma, the pruned game-tree $T$ is finite since it is finitely branching and contains no infinite paths. Turn $T$ into an And–Or tree: label an internal node $\rho$ by $\vee$ in case $\mathsf{end}(\rho) \in V_0$, and by $\wedge$ in case $\mathsf{end}(\rho) \in V_1$; label a leaf $\rho$ by *true* if every play extending $\rho$ is won by Player 0, and by *false* if every play extending $\rho$ is won by Player 1. Note that every leaf gets a label. It is easy to see that Player 0 has a winning strategy from $v$ in the game $(\mathcal{A}, O)$ if and only if the And–Or tree evaluates to *true*. □

[4] When $\mathbb{U}$ is clear from the context, we will not mention it.

[5] This is usually attributed to Zermelo, but also follows from the fact that open games are determined, e.g., [7].

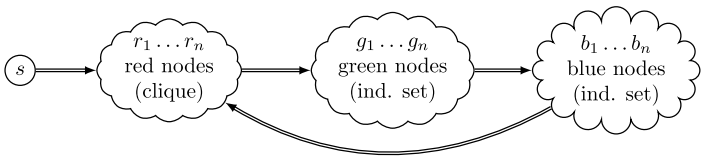



**Fig. 1.** double arrows represent one edge from every node to every node.

## 4. First-cycle games

### 4.1. Memory bounds and complexity

In this section we discuss the amount of memory required to win FCGs and the computational complexity of deciding which player has a winning strategy.

We begin by showing that while strategies with memory $2^{O(|V|\log|V|)}$ always suffice, there are FCGs that require $2^{\Omega(|V|\log|V|)}$ memory.

**Proposition 1.**

1. *For a FCG on an arena with $n$ vertices, if Player $\sigma$ wins from $v$, then every winning strategy for Player $\sigma$ starting from $v$ uses memory at most $n!$.*
2. *For every $n \in \mathbb{N}$ there exists a FCG on an arena with $3n+1$ vertices, and a vertex $v$, such that every winning strategy for Player 0 from $v$ uses memory at least $n!$.*

**Proof.** For the upper bound, note that it is enough to remember the whole history of the play until a cycle is formed, i.e., all histories of length at most $n-1$. To store all histories of length $k$ requires $\sum_{i\leq k} i! < (k+1)!$ many states. Thus, $n!$ memory suffices.

We now turn to the lower bound.

**Sketch.** We define a game in which Player 1 can select any possible permutation of $\{1,\dots,n\}$ and, in order to win, Player 0 must remember this permutation. Consider the arena in Fig. 1. Intuitively, the red nodes are used by Player 1 to select a permutation of $\{1,\dots,n\}$, and the green and blue nodes are used by her to query Player 0 as to the relative order of any pair of indices $i, j$ in this permutation. The outgoing edges from the blue nodes are used by Player 0 to respond to the query by going back to either $r_i$ or $r_j$. Player 0 wins the game if the cycle contains both $r_i$ and $r_j$, hence, he must remember the order of the pair $i, j$ in the permutation. Since this is true for every such pair of indices, he must use at least $n!$ memory to store the permutation. The winning condition captures the above intuition, and also ensures that Player 1 loses if she deviates from the above protocol.

**Details.** The arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$ has $n$ red nodes $r_1,\dots,r_n$, $n$ green nodes $g_1,\dots,g_n$, $n$ blue nodes $b_1,\dots,b_n$, and an initial node $s$. The red nodes form a clique (i.e., $(r_i, r_j) \in E$ for every $1 \leq i \neq j \leq n$), the green nodes form an independent set, and so do the blue nodes (i.e., $(g_i, g_j) \notin E$ and $(b_i, b_j) \notin E$ for every $1 \leq i, j \leq n$). In addition, for every $1 \leq i, j \leq n$ we have the edges $(s, r_j), (r_i, g_j), (g_i, b_j), (b_i, r_j)$. Player 0 owns the blue nodes ($V_0 = \{b_1,\dots,b_n\}$), and the rest belong to Player 1. In order to correctly describe the winning condition, we label the nodes as follows: for every $1 \leq i \leq n$, the nodes $r_i, g_i, b_i$ are labelled $(Red, i), (Green, i), (Blue, i)$, respectively; and the node $s$ is labelled by 0. In other words, $\mathbb{U} = \{0\} \cup (\{Red, Green, Blue\} \times \{1,\dots,n\})$, and the induced edge labelling is: for every $(v, w) \in E$, we have that $\lambda(v, w) = (Red, i)$ if $v = r_i$, $\lambda(v, w) = (Green, i)$ if $v = g_i$, $\lambda(v, w) = (Blue, i)$ if $v = b_i$, and otherwise $\lambda(v, w) = 0$.

We now define the winning condition. Player 0 wins a play iff the first cycle $(v_1, v_2)\dots(v_k, v_{k+1})$ on the play satisfies one of the following three conditions[6]:

1. the node just before the end is not blue (i.e., $v_k \notin \{b_1,\dots,b_n\}$); or
2. it does not have exactly 1 green node and 1 blue node; or
3. exactly 1 green node $g_j$, and 1 blue node $b_l$ appear on it, and $b_l = v_k$, and
   (a) it has red nodes labelled with the same numbers as $g_j$ and $b_l$ (i.e., $r_j, r_l$ are on the cycle), and
   (b) it starts with a red node labelled with a number equal to that of the green or the blue node (i.e., $v_1 \in \{r_j, r_l\}$).

Informally, the first two conditions above ensure that Player 0 can win if Player 1 does not select a permutation followed by a query, whereas the third condition ensures that if Player 1 does, then Player 0 can win but only if he remembers the whole permutation.

[6] For simplicity, we state the winning condition in terms of the nodes on the cycle and not in terms of the labels of the edges. However, since we essentially label an edge by the name of its source node, the latter can be easily done.

We first prove that Player 0 has a winning strategy.[7] We distinguish between two types of plays:

(i) Player 1 selects a permutation through the red nodes, and then queries Player 0 by going to some green node $g_j$, followed by a blue node $b_l$; i.e., $u = (s, r_{i_1})(r_{i_1}, r_{i_2}) \dots (r_{i_{n-1}}, r_{i_n})(r_{i_n}, g_j)(g_j, b_l)$ are the first $n+2$ edges of the play, and for every $1 \leq x \leq n$ we have that $x = i_h$ for some $1 \leq h \leq n$. In this case, let $1 \leq \tilde{j}, \tilde{l} \leq n$ be such that $i_{\tilde{j}} = j$ and $i_{\tilde{l}} = l$, and observe that Player 0 can win by taking the edge $(b_l, r_{i_h})$ where $h = min(\tilde{j}, \tilde{l})$ (i.e., by taking the edge back to the node among $r_i, r_j$ that appears sooner on $u$). Indeed, the first cycle of the play will then be $(r_{i_h}, r_{i_{h+1}}) \dots (r_{i_{n-1}}, r_{i_n})(r_{i_n}, g_j)(g_j, b_l)$, and it will satisfy the third option in the winning condition (in particular, by our choice of $h$, both $r_i$ and $r_j$ are on the cycle, and $i_h \in \{j, l\}$).

(ii) The play never reaches a blue node, or when the play reaches a blue node for the first time it does not conform to the pattern given in case $(i)$ above. If the play never reaches a blue node then Player 0 wins since the first cycle formed satisfies the first option in the winning condition. This is also true if a cycle was formed before the play reaches a blue node for the first time. Hence, we are left with the option that the prefix of the play is of the form $u = (s, r_{i_1})(r_{i_1}, r_{i_2}) \dots (r_{i_{t-1}}, r_{i_t})(r_{i_t}, g_j)(g_j, b_l)$, where $t < n$, and for $x \neq y$ we have $r_{i_x} \neq r_{i_y}$. In this case, Player 0 first takes the edge $(b_l, r_m)$, where $1 \leq m \leq n$ is the index of a red node that did not yet appear on the play (i.e., $i_x \neq m$ for all $1 \leq x \leq t$). Note that this does not yet form a cycle and the play continues. Now, the game can proceed in three ways, all losing for Player 1: the first is by keeping the play forever away from blue nodes, thus closing a cycle satisfying option 1 in the winning condition; the second is by closing the first cycle by going back to $b_l$, again losing by option 1 (since then $v_k$ is green); the third is by reaching some blue node $b_h$ different from $b_l$ without yet forming a cycle. In this last case, Player 0 wins (using the second option in the winning condition) by taking the edge $(b_h, r_{i_t})$ and thus forming a cycle that contains two different blue nodes: $b_l, b_h$.

We now show that every strategy for Player 0 that uses less than $n!$ memory is not winning. Let $\xi$ be a Player 0 strategy that is implemented using a Mealy machine $\mathcal{M}$ with less than $n!$ states. Hence, there are two different permutations $p = i_1, \dots i_n$ and $p' = i'_1, \dots, i'_n$ of $\{1, \dots, n\}$, such that $\mathcal{M}$ is in the same state $m$ after reading the history $s \cdot r_{i_1} \dots r_{i_n}$, as after reading the history $s \cdot r_{i'_1} \dots r_{i'_n}$. Let $h$ be the smallest index such that $i_h \neq i'_h$, and let $j := i_h$ and $l := i'_h$, and let $1 \leq \tilde{j}, \tilde{l} \leq n$ be such that $i'_{\tilde{j}} = j$ and $i_{\tilde{l}} = l$. Simply put, $j$ appears in position $h$ in $p$ and position $\tilde{j}$ in $p'$, whereas $l$ appears in position $h$ in $p'$ and position $\tilde{l}$ in $p$. The minimality of $h$ implies that $\tilde{j}, \tilde{l} > h$, and thus $l$ appears after $j$ in $p$, but before $j$ in $p'$.

Observe that the two histories $\rho = s \cdot r_{i_1} \dots r_{i_n} \cdot g_j \cdot b_l$, and $\rho' = s \cdot r_{i'_1} \dots r_{i'_n} \cdot g_j \cdot b_l$ also put $\mathcal{M}$ in the same state and thus, $\xi$ responds to both histories with the same move, say by taking the edge $(b_l, r_x)$. Observe that, since $p, p'$ are permutations, every red node already appears on $\rho, \rho'$. Hence, in both cases, every possible move of Player 0 from $b_l$ closes a cycle that has exactly one green and one blue node, and the blue node appears just before the end. It follows that, in both cases, in order to win Player 0 must satisfy items 3.*a* and 3.*b* of the winning condition. In order to satisfy 3.*a* he must choose $r_x = r_l$ or $r_x = r_j$. However, that prevents him from satisfying item 3.*b* in both cases since $r_j$ appears on $\rho$ before $r_l$, but after it on $\rho'$. More formally, consider first the option $r_x = r_l$ and the history $\rho$. Recall that $i_{\tilde{l}} = l$, that $j = i_h$, and that $\tilde{l} > h$. Hence, the first cycle formed is $(r_{i_{\tilde{l}}}, r_{i_{\tilde{l}+1}}) \dots (r_{i_{n-1}}, r_{i_n})(r_{i_n}, g_j)(g_j, b_l)(b_l, r_{i_{\tilde{l}}})$, and it does not contain the required $r_j$. Similarly, if $r_x = r_j$ consider the history $\rho'$, and note that the first cycle formed is $(r_{i'_{\tilde{j}}}, r_{i'_{\tilde{j}+1}}) \dots (r_{i'_{n-1}}, r_{i'_n})(r_{i'_n}, g_j)(g_j, b_l)(b_l, r_{i'_{\tilde{j}}})$, and it does not contain the required $r_l$. It follows that $\xi$ is not a winning strategy. □

We now address the complexity of solving FCGs with efficiently computable cycle properties. Given a game, and a starting node $v$, solving the game is the problem of deciding whether Player 0 or Player 1 wins from $v$. We show that this problem is in general Pspace-complete.

**Theorem 1.**

1. *If $Y$ is a cycle property for which solving membership is in* Pspace *then the problem of solving games in* Fcg$(Y)$ *is in* Pspace.
2. *There exist a cycle property $Y$, that is computable in linear-time, such that the problem of solving games in* Fcg$(Y)$ *is* Pspace*-hard.*

**Proof.** Let $\mathcal{A}$ be an arena with $n$ nodes. Following the proof of Lemma 3, for the special case of FCGs, prune a branch of the game-tree once the first cycle is formed, and label the game-tree to get an And–Or tree. Every node in the And–Or tree is of depth at most $n$, and has at most $n$ children. Hence, evaluating the tree can be done, using a depth-first algorithm, in space polynomial in $n$ plus the maximal space required to compute the labels of the leaves $\rho$ in the tree. Note that the label of a leaf $\rho$ is determined by whether $\lambda(c) \in Y$ or not (where $c$ is the first-cycle on $\rho$). By our assumption on $Y$ this can be done in polynomial space, hence the upper bound follows.

[7] We define the strategy only for histories that are consistent with it. Obviously, it can be arbitrarily defined on any other history.

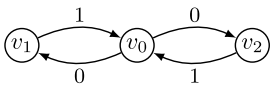



**Fig. 2.** Sample arena $A$. Circles are Player 0 nodes, solid lines are edges.

The lower bound follows immediately by reducing the PSPACE-hard problem QBF to solving FCG($Y$), where $Y$ is the set of all sequences whose last two elements are identical. The proof uses the same arena (which is vertex-labelled – see Remark 1) as in the reduction of QBF to Generalised Geography (e.g., [11, Theorem 8.11]). To quickly recap, the basic idea there is to view QBF as a game in which Player 0 (resp. Player 1) assigns the values of an existentially (resp. universally) quantified variable $x_i$ by picking a node labelled by the literal $x_i$ or the literal $\neg x_i$; after all variables are thus assigned, Player 1 picks a clause (challenging Player 0 to show that it is true), and finally Player 0 responds by picking a literal of this clause (that he claims is true). The structure of the arena is such that the only outgoing edge from the node picked by Player 0 in this last step is the node labelled by the same literal that was available during the value-assigning phase. Thus, if indeed that literal was picked in the assignment phase, then the next move closes a cycle that satisfies $Y$ and otherwise, taking this edge does not close a cycle, forcing Player 0 to close a losing cycle in the next step. Overall, the QBF formula is true iff Player 0 wins this FCG. □

### 4.2. The relation between point-wise and uniform memoryless determinacy

In this section we consider the difference between point-wise memoryless determinacy and uniform memoryless determinacy in FCGs. We begin by considering the simple case of solitaire games.

**Theorem 2.** *All solitaire FCGs are point-wise memoryless determined. However, some solitaire FCG's are not uniform memoryless determined.*

**Proof.** The fact that a solitaire FCG is point-wise memoryless determined is simply because once a node repeats the game is effectively over, and thus the player makes at most one meaningful move from any node. In other words, Player $\sigma$ can win from a node $v$ iff there is a play $\pi$ starting in $v$ such that the first cycle $\pi_i \dots \pi_j$ on $\pi$ satisfies $Y^\sigma$. Traversing the prefix $\pi_1 \dots \pi_j$ requires no memory since each vertex on it is the source of exactly one edge. For the second item, consider the arena in Fig. 2. Observe that, for the cycle property $Y = \mathsf{cyc\text{-}EndsZero}$, no memoryless strategy is winning from both $v_1$ and $v_2$. □

We now consider two-player games. In contrast with solitaire games, some two-player FCGs are not point-wise memoryless determined. Indeed, any solitaire game (in which all nodes belong to Player 0) that is not uniform memoryless determined, can be turned into a two player game in which Player 0 wins from some node $w$, but requires memory to do so: simply add a single Player 1 node $w$ that has outgoing edges to all nodes in the original game.

As we later show (Corollary 1, Page 205), if a cycle property $Y$ is closed under cyclic permutations then all solitaire games in FCG($Y$) are uniform memoryless determined. Unfortunately, for two player games, this is not enough even for point-wise memoryless determinacy. Indeed, Theorem 3 shows that closure of $Y$ under cyclic permutations is a necessary condition for having all games in FCG($Y$) be point-wise memoryless determined, but it is not a sufficient one.[8]

We also show that, for cycle properties $Y$ that are closed under cyclic permutations, if a game in FCG($Y$) is point-wise memoryless for a player then it is also uniform memoryless for this player (Theorem 3 Item 3). The proof of this fact uses the following Lemma.

**Lemma 4.** *Suppose $Y$ is closed under cyclic permutations, and let $S$ be a strategy for Player $\sigma$ in arena $\mathcal{A}$. If $S$ is winning from $v$, then $S$ is winning from every node $w \in reach_{\mathcal{A}}(S, v)$.*

**Proof.** We begin with the intuition.

**Sketch.** If $c$ is the first cycle of some play $\pi \in plays(S, w)$, then we can construct a path $\pi'$ starting in $v$ and consistent with $S$ whose first cycle is some cyclic permutation $c'$ of $c$, as follows: we proceed along some simple path from $v$ to $w$ until we hit a node on $\pi$; if this node is not on $c$, we continue along $\pi$ until we reach a node on $c$; and finally, we cycle along the nodes of $c$ until we return to where we started, forming a cycle $c'$. Note that $c'$ is not necessarily identical to $c$ since in the first stage we may have hit $\pi$ somewhere in the middle of $c$. Since $c' = first(\pi') \in Y^\sigma$, and $Y^\sigma$ is closed under cyclic permutations (note that $Y$ is closed under cyclic permutations if and only if $\neg Y$ is closed under cyclic permutations), we get that $\pi$ is won by Player $\sigma$.

**Details.** It is enough to show that in the arena $\mathcal{A}^{|S}$, for every play $\pi$ starting in $w$, there is a path $\pi'$ starting in $v$, such that the first cycle of $\pi'$ is a cyclic permutation of the first cycle $\pi_m \dots \pi_n$ of $\pi$. We construct $\pi'$ as follows. Let $\rho$ be a path

[8] This result corrects a mistake in [4] that claims that this is a sufficient condition (note that they do not address the question of whether it is necessary).

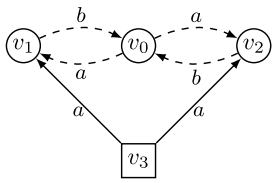



**Fig. 3.** Sample arena $B$. Circles are Player 0 nodes, squares are Player 1 nodes, solid lines are edges, dashed lines represent (sequences of edges that are) simple paths.

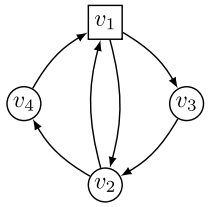



**Fig. 4.** Sample arena $C$. Circles are Player 0 nodes, squares are Player 1 nodes, solid lines are edges.

in $\mathcal{A}^{|S}$ from $v$ to $w$ (such a path exists since $w \in \mathit{reach}(S, v)$). Let $\ell$ be the smallest index such that $\rho_\ell$ intersects $\pi[1, n]$, i.e., such that $\mathsf{end}(\rho_\ell) = \mathsf{start}(\pi_j)$ for some $j \leq n$. Note that $\ell$ is well defined since $\mathsf{end}(\rho) = w = \mathsf{start}(\pi_1)$. We consider two cases (depending on whether or not $\rho$ first intersects $\pi$ before $\pi$ starts traversing $\mathit{first}(\pi)$).

Case $j \leq m$. Let $\pi' = \rho_1 \dots \rho_\ell \pi_j \dots \pi_n$, and note that $\mathit{first}(\pi') = \mathit{first}(\pi)$.

Case $m < j \leq n$. Let $\pi' = \rho_1 \dots \rho_\ell \pi_j \dots \pi_n \pi_m \dots \pi_{j-1}$, and note that $\mathit{first}(\pi') = \pi_j \dots \pi_n \pi_m \dots \pi_{j-1}$ which is a cyclic permutation of $\mathit{first}(\pi)$. □

**Theorem 3.**

1. *If $Y$ is not closed under cyclic permutations then there is a game in* $\textsc{Fcg}(Y)$ *that is not point-wise memoryless determined.*
2. *There exists a cycle property $Y$, closed under cyclic permutations, and a game in* $\textsc{Fcg}(Y)$ *that is not point-wise memoryless determined.*
3. *If a cycle property $Y$ is closed under cyclic permutations, then every game in* $\textsc{Fcg}(Y)$ *that is point-wise memoryless for Player $\sigma$ is also uniform memoryless for Player $\sigma$.*

**Proof.** For the first item, assume that $Y$ is not closed under cyclic permutations, and let $a, b \in \mathbb{U}^*$ be such that $ab \in Y$ but $ba \notin Y$. Consider the arena in Fig. 3. Observe that Player 0 wins from $v_3$, but in order to do so he needs to remember if the play arrived to $v_0$ from $v_1$ or from $v_2$.

For the second item, consider the arena in Fig. 4, and the cycle property $Y = \mathsf{cyc\text{-}EvenLen}$. Obviously, $Y$ is closed under cyclic permutations. However, starting at $v_1$, Player 0 has a winning strategy, but no memoryless winning strategy – when choosing the outgoing edge from $v_2$ the player needs to remember if the previous node was $v_1$ (in which case he should return to $v_1$), or $v_3$ (in which case he should go to $v_4$).

For the third item, write $W_\sigma := \mathsf{WR}^\sigma(\mathcal{A}, O)$, and assume that the game is point-wise memoryless for Player $\sigma$, and for every $v \in W_\sigma$ let $S_v$ be a memoryless winning strategy for Player $\sigma$ from $v$.

**Sketch.** The idea is to define a memoryless strategy $S$ for Player $\sigma$ that is winning from every node $v \in W_\sigma$ by considering the nodes of $W_\sigma$ (in some arbitrary order), and if $v$ is the next node in $W_\sigma$ for which $S(v)$ is not yet defined, then define $S(w) := S_v(w)$ for all $w \in \mathit{Reach}(S_v, v)$ that have not yet been defined. Thus, $S$ is memoryless by construction. The main work is to show that it is winning (this is where Lemma 4 is used).

**Details.** Fix some arbitrary linear ordering $v_1 \prec v_2 \prec \dots \prec v_n$ of the nodes in $W_\sigma$. We iteratively build a memoryless strategy $S$ for Player $\sigma$ that is winning from every node $v \in W_\sigma$. At each round $j \geq 0$ of this construction, we write $V^j \subseteq V$ for the set of nodes considered by the end of that round, and have that $S$ is defined for every node in $V^j \cap V_\sigma$. At round 0, we begin with $V^0 = \emptyset$, and with $S(v)$ undefined for all $v \in V_\sigma$. At round $j > 0$, if $V^{j-1} = V$ then we are done, and otherwise we proceed as follows:

(1) If $W_\sigma \not\subseteq V^{j-1}$, let $v$ be the smallest vertex (by the ordering $\prec$) such that $v \in W_\sigma \setminus V^{j-1}$. Set $V^j := V^{j-1} \cup \mathit{reach}(S_v, v)$, and for every $w \in (V_\sigma \cap \mathit{reach}(S_v, v)) \setminus V^{j-1}$ define $S(w) := S_v(w)$.
(2) If $W_\sigma \subseteq V^{j-1}$, then set $V^j = V$, and for every node $v \in V_\sigma \setminus V^{j-1}$ define $S(v)$ arbitrarily. Intuitively, moves from these nodes are unimportant since they are not reachable from any node in $W_\sigma$ on any play consistent with $S$.

It is easy to see that $S$ is well defined and memoryless.

It remains to show that $S$ is winning from every $w \in W_\sigma$. The proof is by induction on the round number $j$. The induction hypothesis is that *(i)* if $w \in V^j$ then $reach(S, w) \subseteq V^j$ and; *(ii)* if $W_\sigma \not\subseteq V^{j-1}$ then $S$ is winning from every $w \in V^j$.

Observe that, by taking the maximal $j$ for which $W_\sigma \not\subseteq V^{j-1}$, item (ii) in the induction hypothesis implies that $S$ is winning from every $w \in W_\sigma$.

For $j = 0$, the hypothesis is trivially true. Assume that the hypothesis holds for all $0 \leq l < j$, and consider round $j$. If $W_\sigma \subseteq V^{j-1}$, then (i) is true since the construction uses case (2) and sets $V^j = V$, and (ii) is trivially true. If, on the other hand, $W_\sigma \not\subseteq V^{j-1}$ (i.e., the construction uses cases (1)), then let $w \in reach(S_v, v)$ be some node added at round $j$. Note that to prove that (i) holds, it is enough to show that every node $w'$, that is reachable in $\mathcal{A}^{|S}$ from $w$, was added before or at round $j$. In other words, we have to show that $reach(S, w) \subseteq (V^{j-1} \cup reach(S_v, v))$. This can be easily done by inducting on the length of the node-path $\rho = v_1 \dots v_k \in V^*$ from $w$ to $w'$ in $\mathcal{A}^{|S}$. For $k = 0$ this is trivially true. Assume it is true for $\rho$ of length $k$, and consider $\rho$ of length $k+1$. Now, if $\rho_k$ was added at a previous round, then so did $\rho_{k+1}$ by (i) in the induction hypothesis applied to $\rho_k$. Otherwise, $\rho_k$ was added at this round (and thus $\rho_k \in reach(S_v, v)$), in which case if $\rho_k$ belongs to the opponent (i.e., $\rho_k \in V_{1-\sigma}$) then all its successors, and in particular $\rho_{k+1}$, are in $reach(S_v, v)$; and if $\rho_k \in V_\sigma$ then $\rho_{k+1} = S(\rho_k) = S_v(\rho_k) \in reach(S_v, v)$. To complete the proof, we have to show that (ii) holds, i.e., that $S$ is winning from $w$.

To see that $S$ is winning from $w$, take any play $\pi \in plays(S, w)$, and let $\pi_m \dots \pi_n$ be the first cycle of $\pi$. There are two options: either the prefix $\pi_1 \dots \pi_n$ is consistent with $S_v$, or it isn't. If it is, since $S_v$ is winning for Player $\sigma$ from every node in $reach(S_v, v)$ (Lemma 4), and thus in particular from $w$, we have that $\pi$ is won by Player $\sigma$. Assume then that $\pi_1 \dots \pi_n$ is not consistent with $S_v$. Note that by our choice of $\pi$ it is consistent with $S$ and thus, for some $1 \leq h < n$, we have that $S_v(\mathsf{start}(\pi_h)) \neq \mathsf{end}(\pi_h) = S(\mathsf{start}(\pi_h))$. Let $k \leq n$ be the smallest index such that $\mathsf{start}(\pi_k) \in V^{j-1}$. Such a $k$ exists by the above inequality and the fact that (by construction) $S$ agrees with $S_v$ on all nodes added at round $j$. Observe that if $k > m$ then all the nodes appearing on $\pi_m \dots \pi_n$ are in $reach(S, \pi_k)$, simply by following the cycle $\pi_k \dots \pi_n \pi_m \dots \pi_{k-1}$. Hence, since by item (i) in the induction hypothesis $reach(S, \mathsf{start}(\pi_k)) \subseteq V^{j-1}$, the minimality of $k$ implies that $k \leq m$. We conclude that the suffix $\pi' = \pi_k \pi_{k+1} \dots$ of $\pi$, which is a play consistent with $S$ starting from $\mathsf{start}(\pi_k) \in V^{j-1}$, satisfies $first(\pi') = first(\pi)$. By item (ii) in the induction hypothesis, $\pi'$ is won by Player $\sigma$, hence so is $\pi$, which completes the proof. □

Theorems 2 and 3 give us the following corollary:

**Corollary 1.** *If a cycle property $Y$ is closed under cyclic permutations, then every solitaire game in* $\textsc{Fcg}(Y)$ *is uniform memoryless determined.*

## 5. Memoryless determinacy of first-cycle games

In this section we give a necessary and sufficient condition for a FCG to be memoryless determined. In Section 7 we give an easy-to check condition that is sufficient, but not necessary.

### 5.1. A full characterisation of memoryless determinacy of all games in $\textsc{Fcg}(Y)$

We begin by introducing some useful shorthand notation. Given an arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$, and a node $z \in V$, for a path $\pi \in E^* \cup E^\omega$, define $N_z(\pi) \in \mathbb{N} \cup \{\infty\}$ to be the index of the first edge that starts with $z$, if one exists. Formally, $N_z(\pi) := \infty$ if $z$ does not occur on $\pi$, and otherwise $N_z(\pi) := \min\{j : \mathsf{start}(\pi_j) = z\}$. Also, define $head_z(\pi) := \pi[1, N_z(\pi) - 1]$ to be the prefix of $\pi$ before $N_z(\pi)$, and $tail_z(\pi) := \pi[N_z(\pi), |\pi|]$ to be the suffix of $\pi$ starting at $N_z(\pi)$. By convention, if $N_z(\pi) = \infty$ then $head_z(\pi) = \pi$ and $tail_z(\pi) = \epsilon$.

We now define a game, that is very similar to the first-cycle game, except that one of the nodes of the arena is designated as a "reset" node:

**Definition 1.** Fix an arena $\mathcal{A}$, a vertex $z \in V$, and a cycle property $Y$. Define the objective $O^{\mathcal{A}}_{z\text{-first}}(Y) \subseteq plays(\mathcal{A})$ to consist of all plays $\pi$ satisfying the following property: if $head_z(\pi)$ is not a simple path then $first(\pi) \in Y$, and otherwise $first(tail_z(\pi)) \in Y$.

Playing the game with objective $O_{z\text{-first}}(Y)$ is like playing the first-cycle game over $Y$, however, if no cycle is formed before reaching $z$ for the first time, the prefix of the play up to that point is ignored. Thus, in a sense, the game is reset. Also note that if play starts from $z$, then the game reduces to a first-cycle game. It turns out that we may assume that a strategy of $(\mathcal{A}, O_{z\text{-first}}(Y))$ makes the same move every time it reaches $z$:

**Definition 2** *(Forgetful at z from v).* For an arena $\mathcal{A}$, a vertex $v \in V$, a Player $\sigma \in \{0, 1\}$, and a vertex $z \in V_\sigma$ belonging to Player $\sigma$, we call a strategy $T$ for Player $\sigma$ *forgetful at z from v* if there exists $z' \in V$ such that $(z, z') \in E$ and for all $\pi \in plays(T, v)$, and all $n \in \mathbb{N}$, if $\mathsf{start}(\pi_n) = z$ then $\mathsf{end}(\pi_n) = z'$.

**Lemma 5** *(Forgetful at z from v). Fix an arena $\mathcal{A}$, a vertex $v \in V$, a Player $\sigma \in \{0,1\}$, and a vertex $z \in V_\sigma$ belonging to Player $\sigma$. In the game $(\mathcal{A}, O_{z\text{-first}}(Y))$, if Player $\sigma$ has a strategy $S$ that is winning from $v$, then Player $\sigma$ has a strategy $T$ that is winning from $v$ and that is forgetful at $z$ from $v$.*

**Proof.** We begin with the intuition.

**Sketch.** The second time $z$ appears on a play, the winner is already determined, and so the strategy is free to repeat the first move it made at $z$. On the other hand, when a play visits $z$ the first time, the strategy can make the same move regardless of the history of the play before $z$, because the winning condition ignores this prefix.

**Details.** We may suppose that $z$ appears on some play of $plays(S, v)$, otherwise we can take $T$ to be $S$. Let $\rho$ be a simple path from $v$ to $z$ that is consistent with $S$. Let $z' := S(\rho)$. Define the strategy $T$ as follows:

$$T(u) := \begin{cases} S(u) & \text{if } z \text{ does not appear on } u, \\ z' & \text{if } \mathsf{end}(u) = z, \\ S(\rho \cdot tail_z(u)) & \text{otherwise.} \end{cases}$$

By definition, $T$ is forgetful at $z$ from $v$. It remains to show that every $\pi \in plays(T, v)$ is won by Player $\sigma$.

First consider the case that $head_z(\pi)$ is not a simple path. By definition, $S$ and $T$ agree on $head_z(\pi)$, and since $S$ is winning, the first cycle on $head_z(\pi)$ (and thus also on $\pi$) satisfies $Y^\sigma$, and $\pi$ is won by Player $\sigma$.

Now consider the case that $head_z(\pi)$ is a simple path. We need to show that $first(tail_z(\pi))$ is in $Y^\sigma$. Define $\pi' := \rho \cdot tail_z(\pi)$. It is easy to see that $\pi'$ is consistent with $T$. We claim that $\pi' \in plays(S, v)$. Indeed, the prefix $\rho \cdot (z, z')$ is consistent with $S$; and for every $j \geq |\rho| + 1$, such that $\mathsf{end}(\pi'_j) \in V_\sigma$, we have, by the third case in the definition of $T$, that $T(\pi'[1, j]) = S(\rho \cdot tail_z(\pi'[1, j])) = S(\pi'[1, j])$ (the second equality holds since, by the definition of $tail_z$, $\rho \cdot tail_z(\pi'[1, j]) = \pi'[1, j]$). Now, since $\pi'$ is consistent with $T$, we have that $T(\pi'[1, j]) = \mathsf{end}(\pi'_{j+1})$, and thus $S(\pi'[1, j]) = \mathsf{end}(\pi'_{j+1})$. This completes the proof of the claim.

To finish the proof, note that $head_z(\pi')$ is a simple path (it is $\rho$), and that $tail_z(\pi') = tail_z(\pi)$. Hence, since $head_z(\pi')$ is a simple path, and $S$ is winning from $v$, we know that $first(tail_z(\pi'))$ satisfies $Y^\sigma$. Thus, since $head_z(\pi)$ is a simple path and $first(tail_z(\pi))$ satisfies $Y^\sigma$, we can conclude that $\pi$ is won by Player $\sigma$. □

We now define the basic notion behind our necessary and sufficient condition for games in $\textsc{Fcg}(Y)$ to be memoryless determined.

**Definition 3.** For an arena $\mathcal{A}$, and a node $v$ in $\mathcal{A}$, say that $\mathcal{A}$ is *$Y$-resettable from $v$* if for every node $z$, the same player wins from $v$ in both $(\mathcal{A}, O_{\mathsf{first}}(Y))$ and $(\mathcal{A}, O_{z\text{-}\mathsf{first}}(Y))$.

First, we show that $Y$-resettability is a sufficient condition for having memoryless strategies.

**Theorem 4.** *Given an arena $\mathcal{A}$, if a node $v$ is such that every sub-arena of $\mathcal{A}$ is $Y$-resettable from $v$, then the game $(\mathcal{A}', O_{first}(Y))$ is memoryless from $v$ for every sub-arena $\mathcal{A}'$ of $\mathcal{A}$.*

**Proof.** A node $z \in V$ is a *choice node* of an arena $\mathcal{B} = (V_0, V_1, E^{\mathcal{B}}, \mathbb{U}, \lambda)$, if there are at least two distinct vertices $v', v'' \in V$ such that $(z, v') \in E^{\mathcal{B}}$ and $(z, v'') \in E^{\mathcal{B}}$.

**Sketch.** Fix a sub-arena $\mathcal{A}'$ of $\mathcal{A}$. Suppose Player $\sigma$ has a winning strategy from $v$ in $\mathcal{A}'$. We induct on the number of choice nodes of Player $\sigma$. Let $z$ be a choice node for Player $\sigma$ (if there are none, the result is immediate). Since $\mathcal{A}'$ is $Y$-resettable from $v$, Player $\sigma$ also wins the game with objective $O_{z\text{-}\mathsf{first}}(Y)$ from $v$. By Lemma 5, Player $\sigma$ has a strategy $S$ that is winning from $v$ and that is also forgetful at $z$ from $v$. Thus we may form a sub-arena $\mathcal{B}$ of $\mathcal{A}'$ (and hence of $\mathcal{A}$) by removing all edges from $z$ that are not taken by $S$. Observe that $S$ is winning from $v$ in $(\mathcal{B}, O_{z\text{-}\mathsf{first}}(Y))$. Since the sub-arena $\mathcal{B}$ is $Y$-resettable from $v$, Player $\sigma$ also wins $(\mathcal{B}, O_{\mathsf{first}}(Y))$ from $v$. But $\mathcal{B}$ has less choice nodes for Player $\sigma$, and thus, by induction, Player $\sigma$ has a memoryless winning strategy from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$. This memoryless strategy is also winning from $v$ in $\mathcal{A}'$.

**Details.** Fix a sub-arena $\mathcal{A}'$ of $\mathcal{A}$, and let $\sigma \in \{0, 1\}$ be such that $v \in \mathsf{WR}^\sigma(\mathcal{A}', O_{\mathsf{first}}(Y))$. The $n$th inductive hypothesis states: for every sub-arena $\mathcal{B}$ (of $\mathcal{A}'$), in which Player $\sigma$ has exactly $n$ choice nodes, if Player $\sigma$ has a winning strategy from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$, then Player $\sigma$ has a *memoryless* winning strategy from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$.

The base case is when $n = 0$, i.e, Player $\sigma$ has no choice nodes in $\mathcal{B}$. In this case, Player $\sigma$ has a single strategy: given a history $u \in H_\sigma(\mathcal{B})$ with $end(u) = w$, then take the unique edge $(w, w') \in E^{\mathcal{B}}$. Note that this strategy is memoryless.

Let $n > 0$, and suppose the inductive hypothesis holds for $n - 1$. We will prove it holds for $n$. To this end, let $\mathcal{B} = (V_0, V_1, E^{\mathcal{B}}, \mathbb{U}, \lambda)$ be a sub-arena (of $\mathcal{A}'$) with $n$ choice nodes for Player $\sigma$. Fix one such choice node, and call it $z$. Suppose Player $\sigma$ has a winning strategy (not necessarily memoryless) from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$. Since, by assumption, $\mathcal{B}$ is $Y$-resettable from $v$, there is also a winning strategy $S$ for Player $\sigma$ from $v$ in $(\mathcal{B}, O_{z\text{-}\mathsf{first}}(Y))$. We will use $S$ to prove Player $\sigma$ has a memoryless winning strategy from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$. By Lemma 5, we may assume that $S$ is forgetful at

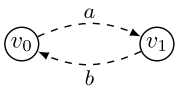



**Fig. 5.** The dashed lines represent simple paths.

$z$ from $v$, i.e., there exists $z' \in V$ such that $(z, z') \in E^{\mathcal{B}}$ and for all $\pi \in plays_{\mathcal{B}}(S, v)$, and all $i \in \mathbb{N}$, if $\mathsf{start}(\pi_i) = z$ then $\mathsf{end}(\pi_i) = z'$.

Define the sub-arena $\mathcal{B}_z$ to be the same as $\mathcal{B}$ but with all edges out of $z$ removed except for $(z, z')$. That is, $\mathcal{B}_z := (V_0, V_1, E_z, \mathbb{U}, \lambda')$ where $E_z := E^{\mathcal{B}} \setminus \{(z, x) : x \neq z'\}$ and $\lambda'$ is $\lambda$ restricted to $E_z$. Then $S$ is well-defined on $\mathcal{B}_z$ (i.e., by our assumption it never says to move from $z$ to a node other than $z'$). Since $S$ is winning for Player $\sigma$ from $v$ in the game $(\mathcal{B}, O_{z\text{-first}}(Y))$ and node $z$ belongs to Player $\sigma$, conclude that $S$ is winning for Player $\sigma$ from $v$ in $(\mathcal{B}_z, O_{z\text{-first}}(Y))$. Being a sub-arena of $\mathcal{A}$, by assumption, the arena $\mathcal{B}_z$ is $Y$-resettable from $v$. Hence, Player $\sigma$ wins from $v$ in $(\mathcal{B}_z, O_{\mathsf{first}}(Y))$. Since $\mathcal{B}_z$ is a sub-arena of $\mathcal{A}'$ and has $n-1$ choice nodes for Player $\sigma$, we can apply the induction hypothesis to $\mathcal{B}_z$ and obtain that Player $\sigma$ has a memoryless strategy $S_{mem}$ winning from $v$ in $(\mathcal{B}_z, O_{\mathsf{first}}(Y))$. Recall that $\mathcal{B}_z$ was obtained from $\mathcal{B}$ by removing outgoing edges from $z \in V_\sigma$ (i.e., by providing Player $\sigma$ with less freedom of movement), and conclude that $S_{mem}$ must be winning for Player $\sigma$ from $v$ in the game $(\mathcal{B}, O_{\mathsf{first}}(Y))$. This completes the inductive step. □

It is worth noting that the assumption in Theorem 4, that every sub-arena of $\mathcal{A}$ is $Y$-resettable from $v$, cannot be replaced by the weaker requirement that only $\mathcal{A}$ is $Y$-resettable from $v$. Consider for example the arena $\mathcal{A}$ in Fig. 3 (page 204), and a cycle property $Y \subseteq \mathbb{U}^*$, such that $ab \in Y$ but $ba \notin Y$, for some $a, b \in \mathbb{U}^*$. As argued in Theorem 3, the game $(\mathcal{A}, O_{\mathsf{first}}(Y))$ is not memoryless starting at $v_3$. While the sub-arena $\mathcal{A}'$, obtained by dropping the edge $(v_0, v_1)$, is not $Y$-resettable from $v_3$ (since for $z = v_0$ Player 0 wins $(\mathcal{A}', O_{z\text{-first}}(Y))$ but $(\mathcal{A}', O_{\mathsf{first}}(Y))$ is won by Player 1) the arena $\mathcal{A}$ is $Y$-resettable from $v_3$.[9]

Before we provide a converse for Theorem 4, we show that an additional assumption, namely that $Y$ is closed under cyclic permutations, is needed:

**Lemma 6.** *If $Y$ is not closed under cyclic permutations then there is an arena $\mathcal{A}$, and a node $v$, for which the game $(\mathcal{A}', O_{first}(Y))$ is memoryless from $v$ for every sub-arena $\mathcal{A}'$ of $\mathcal{A}$, but $\mathcal{A}$ is not $Y$-resettable from $v$.*

**Proof.** Assume that $Y$ is not closed under cyclic permutations, and let $a, b \in \mathbb{U}^*$ be such that $ab \in Y$ but $ba \notin Y$. Consider the arena in Fig. 5, containing only Player 0 nodes, where the path from $v_0$ to $v_1$ is labelled by $a$, and the path from $v_1$ to $v_0$ is labelled by $b$. Observe that $\mathcal{A}$ has only one sub-arena (itself), and there is a single strategy possible in the game $(\mathcal{A}, O_{\mathsf{first}}(Y))$, and it is memoryless. However, $\mathcal{A}$ is not $Y$-resettable from $v_0$ since, starting at $v_0$, Player 0 wins the game $(\mathcal{A}, O_{\mathsf{first}}(Y))$ but not the game $(\mathcal{A}, O_{z\text{-first}}(Y))$ for $z = v_1$. □

We now prove that if $Y$ is closed under cyclic permutations then the converse of Theorem 4 is also true.

**Theorem 5.** *Let $Y$ be closed under cyclic permutations, let $\mathcal{A}$ be an arena, and let $v$ be a node such that the game $(\mathcal{B}, O_{first}(Y))$ is memoryless from $v$ for every sub-arena $\mathcal{B}$ of $\mathcal{A}$. Then every sub-arena $\mathcal{B}$ of $\mathcal{A}$ is $Y$-resettable from $v$.*

**Proof.** We prove that, for every arena $\mathcal{B}$ and a node $v$ in it, if $(\mathcal{B}, O_{\mathsf{first}}(Y))$ is memoryless from $v$ then $\mathcal{B}$ is $Y$-resettable from $v$. The theorem follows by taking $\mathcal{B}$ to be any sub-arena of $\mathcal{A}$.

Let $z$ be a node in $\mathcal{B}$. Lemma 3 (Page 200) implies that FCGs are determined. Thus, it is enough to show that if Player $\sigma$ wins from $v$ in the game $(\mathcal{B}, O_{\mathsf{first}}(Y))$, then he wins from $v$ in the game $(\mathcal{B}, O_{z\text{-first}}(Y))$. So, fix a Player $\sigma \in \{0, 1\}$ and assume that Player $\sigma$ wins from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$. Since by our assumption this game is memoryless determined, there is a memoryless strategy $S$ for Player $\sigma$ winning from $v$ in $(\mathcal{B}, O_{\mathsf{first}}(Y))$. Consider the sub-arena $\mathcal{B}^{\|S}$ induced by $S$, and recall that every path in $\mathcal{B}^{\|S}$ is consistent with $S$. We claim that every simple cycle $\pi = \pi_1 \dots \pi_k$ (of some length $k$) in $\mathcal{B}^{\|S}$, that is reachable from $v$, satisfies $Y^\sigma$. Let $\rho$ be a path in $\mathcal{B}^{\|S}$ of minimal length that starts in $v$ and ends in $\mathsf{start}(\pi_i)$ for some $1 \leq i \leq k$. Consider the path $\pi' = \rho\pi_i \dots \pi_k\pi_1 \dots \pi_{i-1}$. Since $\pi' \in plays_{\mathcal{B}}(S, v)$, and $S$ is winning from $v$, we have that the first cycle $c$ on $\pi'$ satisfies $Y^\sigma$. Observe that (by our choice of $\rho$) $c = \pi_i \dots \pi_k\pi_1 \dots \pi_{i-1}$, and is thus a cyclic permutation of $\pi$. Since $Y^\sigma$ is closed under cyclic permutations (recall that if $Y$ is closed under cyclic permutations then so is $\neg Y$) then $\pi$ satisfies $Y^\sigma$, and the claim is true. In other words, for every play $\rho' \in plays_{\mathcal{B}}(S, v)$, and every simple cycle $c' = \rho'_i \dots \rho'_j$ (for some $i, j \in \mathbb{N}$) on $\rho'$, we have that $c'$ satisfies $Y^\sigma$. Hence, $S$ is also winning from $v$ in the game $(\mathcal{B}, O_{z\text{-first}}(Y))$. □

If an arena $\mathcal{A}$ is $Y$-resettable from $v$, for every node $v$, then we simply say that it is $Y$-resettable. In other words:

[9] One can also come up with such an example (albeit a more complicated one) for $Y = \mathsf{cyc\text{-}EvenLen}$ which is closed under cyclic permutations.

**Definition 4.** An arena $\mathcal{A}$ is *$Y$-resettable* if for every $\sigma \in \{0,1\}$, and every node $z$, we have that $\mathsf{WR}^\sigma(\mathcal{A}, O_{z\text{-first}}(Y)) = \mathsf{WR}^\sigma(\mathcal{A}, O_{\mathsf{first}}(Y))$.

We conclude with this full characterisation:

**Theorem 6** *(Memoryless determinacy characterisation of FCGs). The following are equivalent for every cycle property $Y$:*

1. *$Y$ is closed under cyclic permutations, and every arena $\mathcal{A}$ is $Y$-resettable.*
2. *Every game in $\textsc{Fcg}(Y)$ is uniform memoryless determined.*

**Proof.** Suppose $Y$ is closed under cyclic permutations. Theorem 3 (part 3) says that every game in $\textsc{Fcg}(Y)$ that is point-wise memoryless determined is uniform memoryless determined. But since every arena is assumed $Y$-resettable from every $v$, Theorem 4 implies that every game in $\textsc{Fcg}(Y)$ is point-wise memoryless determined.

Conversely, suppose every game in $\textsc{Fcg}(Y)$ is uniform memoryless determined. By Theorem 3 (part 1), $Y$ must be closed under cyclic permutations. This also means we can apply Theorem 5, and conclude that every arena is $Y$-resettable. □

## 6. Strategy Transfer Theorem: infinite-duration cycle games and first-cycle games

In this section we define the connection between first-cycle games and games of infinite duration (such as parity games, etc.), namely the concept of $Y$-greedy games. We then prove the Strategy Transfer Theorem, which says, roughly, that for every arena, the winning regions of the First-Cycle Game over $Y$ and a $Y$-greedy game coincide, and that memoryless winning strategies transfer between these two games.

**Definition 5** *(Greedy).* Say that a *game* $(\mathcal{A}, O)$ *is $Y$-greedy* if

$$O^{\mathcal{A}}_{\mathsf{all}}(Y) \subseteq O \text{ and } O^{\mathcal{A}}_{\mathsf{all}}(\neg Y) \subseteq E^\omega \setminus O.$$

Intuitively, a game $(\mathcal{A}, O)$ is $Y$-greedy means that Player 0 can win the game $(\mathcal{A}, O)$ if he ensures that every cycle in the cycles-decomposition of the play is in $Y$, and Player 1 can win if she ensures that every cycle in the cycles-decomposition of the play is not in $Y$.

An equivalent formulation is that $(\mathcal{A}, O)$ is $Y$-greedy if

$$O^{\mathcal{A}}_{\mathsf{all}}(Y) \subseteq O \subseteq O^{\mathcal{A}}_{\mathsf{exist}}(Y),$$

where $O^{\mathcal{A}}_{\mathsf{exist}}(Y)$ consists of all plays $\pi$ such that some cycle in $\mathit{cycles}(\pi)$ satisfies $Y$.

Here are some examples.

1. Every all-cycles game $(\mathcal{A}, O_{\mathsf{all}}(Y))$ is $Y$-greedy.
2. Every parity game is cyc-Parity-greedy.
3. Every game with $\nu$-mean-payoff winning condition is $\mathsf{cyc\text{-}MeanPayoff}_\nu$-greedy.

Before proving the Strategy Transfer Theorem we need a lemma that states that one can pump a strategy $S$ that is winning for the first-cycle game to get a strategy $S^{\circlearrowleft}$ that is winning for the all-cycles game by following $S$ until a cycle is formed, removing that cycle from the history, and continuing. The fact that every winning strategy in the first-cycle game of $Y$ can be pumped to obtain a winning strategy in a $Y$-greedy game, is why we call such games "greedy".

Recall from the Definitions that for a finite path $\pi \in E^*$, the stack content at the end of $\textsc{CycDec}(\epsilon, \pi)$ (Algorithm 1) is denoted $\mathit{stack}(\pi)$. Recall our notational abuse (Page 198) that for a finite path $\rho$, we may write $S(\rho)$ instead of $S(\mathsf{nodes}(\rho))$.

**Definition 6** *(Pumping strategy).* Fix an arena $\mathcal{A}$, a Player $\sigma \in \{0,1\}$, and a strategy $S$ for Player $\sigma$. Let the *pumping strategy of* $S$ be the strategy $S^{\circlearrowleft}$ for Player $\sigma$, defined, on any input $u = v_1 \dots v_k \in H_\sigma(\mathcal{A})$, as follows:

*a.* $S^{\circlearrowleft}(u) := S(v_k)$ if $k = 1$ or $\mathit{stack}(\pi) = \epsilon$, and otherwise
*b.* $S^{\circlearrowleft}(u) = S(\mathit{stack}(\pi))$,

where $\pi \in E^*$ is the path corresponding to $u$, i.e., $\mathsf{nodes}(\pi) = u$.

Note that $S^{\circlearrowleft}$ is well-defined since if $\mathit{stack}(\pi) \neq \epsilon$ then $\mathit{stack}(\pi)$ ends with $v_k \in V_\sigma$ and so $\mathit{stack}(\pi)$ is in the domain of $S$. Also note that if $S$ is memoryless then the pumping strategy $S^{\circlearrowleft} = S$.

**Lemma 7.** *Let $\mathcal{A}, \sigma, S$ and $S^{\circlearrowleft}$ be as in Definition 6, and let $v \in V$. If $\pi \in \mathit{plays}(S^{\circlearrowleft}, v)$ then for every cycle $C$ in $\mathit{cycles}(\pi)$ there exists a finite path $\rho$ consistent with $S$ and starting with $v$ such that:*

- *The first cycle on $\rho$ is $C$ (thus, in particular, if $S$ is winning from $v$ in the game $(\mathcal{A}, O_{first}(Y))$ then $C$ satisfies $Y^\sigma$);*
- *$\rho$ only contains edges from $\pi$.*

**Proof. Sketch.** The strategy $S^{\circlearrowleft}$ says to follow $S$, and when a cycle is popped by CycDec, remove that cycle from the history and continue. Thus, for every cycle $C$ that is popped, let $l$ be the time at which the first edge of $C$ is being pushed, and note that the stack up to time $l$ followed by $C$ is a path consistent with $S$ whose first cycle is $C$.

**Details.** Fix $\pi \in plays(S^{\circlearrowleft}, v)$. Every cycle $C \in cycles(\pi)$ was output by the run of CycDec$(\epsilon, \pi)$ at some time $j \in \mathbb{N}$, and is thus of the form $C = cycle^j(\pi)$. Let $\rho := stack_{j-1}(\pi) \cdot \pi_j$, and observe that $C$ is the first cycle on $\rho$, and the second item in the statement of the lemma is true.

For the first item, it is sufficient to prove, for all $k \in \mathbb{N}$, that $stack_{k-1}(\pi) \cdot \pi_k$ is consistent with $S$ and starts with $v$. We remind the reader that, by definition, $\rho$ is consistent with $S$ iff:

1. $S(\mathsf{start}(\rho_1)) = \mathsf{end}(\rho_1)$ if $\mathsf{start}(\rho_1) \in V_\sigma$, and
2. $S(\rho[1,n]) = \mathsf{end}(\rho_{n+1})$ for $1 \leq n < |\rho|$ with $\mathsf{end}(\rho[1,n]) \in V_\sigma$.

We prove that $stack_{k-1}(\pi) \cdot \pi_k$ is consistent with $S$ and starts with $v$, by induction on $k \in \mathbb{N}$.

The base is when $k = 1$. Since $stack_0(\pi) = \epsilon$, we show that the path consisting of the single edge $\pi_1$, which starts with $v$ by definition, is consistent with $S$. Note that we are in item 1. of the consistency condition with $\rho = \pi_1$. So suppose $\mathsf{start}(\pi_1) \in V_\sigma$. Then:

$$\begin{aligned} \mathsf{end}(\pi_1) &= S^{\circlearrowleft}(\mathsf{start}(\pi_1)) && \text{since } \pi \text{ is consistent with } S^{\circlearrowleft}, \\ &= S(\mathsf{start}(\pi_1)) && \text{by definition of } S^{\circlearrowleft}, \text{ part a).} \end{aligned}$$

For the inductive step, let $k > 1$ and suppose the inductive hypothesis holds for $k-1$. We prove it holds for $k$. There are two cases.

i) Suppose $stack_{k-1}(\pi) = \epsilon$. To show $\pi_k$ is consistent with $S$, note that we are again in item 1. of the consistency condition, but this time with $\rho = \pi_k$. Repeat the argument in the base case with $\pi_k$ replacing $\pi_1$. To show that $\pi_k$ starts with $v$ argue as follows. The fact that $stack_{k-1}(\pi) = \epsilon$ means that after pushing $\pi_{k-1}$ onto the stack during step $k-1$ of the algorithm CycDec$(\epsilon, \pi)$, the resulting stack forms a cycle. In other words, $\mathsf{end}(\pi_{k-1}) = \mathsf{start}(stack_{k-2}(\pi) \cdot \pi_{k-1})$. But $\mathsf{start}(\pi_k) = \mathsf{end}(\pi_{k-1})$ since $\pi$ is a path, and $\mathsf{start}(stack_{k-2}(\pi) \cdot \pi_{k-1}) = v$ by the induction hypothesis. Thus $\mathsf{start}(\pi_k) = v$.

ii) Suppose $stack_{k-1}(\pi) \neq \epsilon$. The induction hypothesis states that the path $stack_{k-2}(\pi) \cdot \pi_{k-1}$ is consistent with $S$ and starts with $v$. Observe that $stack_{k-1}(\pi)$ is a (non-empty) prefix of $stack_{k-2}(\pi) \cdot \pi_{k-1}$. So, $stack_{k-1}(\pi)$ starts with $v$, and, being a prefix of a path consistent with $S$, is consistent with $S$. Thus we only need to consider $\pi_k$. That is, we are in item 2. of the consistency condition, with $\rho = stack_{k-1}(\pi) \cdot \pi_k$ and $n = |stack_{k-1}(\pi)|$. If $\mathsf{end}(\rho[1,n]) \in V_\sigma$ then

$$\begin{aligned} \mathsf{end}(\rho_{n+1}) &= \mathsf{end}(\pi_k) && \text{since } \rho_{n+1} = \pi_k, \\ &= S^{\circlearrowleft}(\pi[1,k-1]) && \text{since } \pi \text{ is consistent with } S^{\circlearrowleft}, \\ &= S(stack_{k-1}(\pi)) && \text{by definition of } S^{\circlearrowleft}, \text{ part b),} \\ &= S(\rho[1,n]) && \text{since } \rho[1,n] = stack_{k-1}(\pi). \end{aligned}$$

This completes the inductive step and the proof. □

**Corollary 2.** *Fix Player $\sigma \in \{0,1\}$ and let $(\mathcal{A}, O)$ be a $Y$-greedy game. If $S$ is a strategy for Player $\sigma$ that is winning from $v$ in $(\mathcal{A}, O_{first}(Y))$ then $S^{\circlearrowleft}$ is winning from $v$ in $(\mathcal{A}, O)$.*

**Proof.** Suppose $S$ is winning from $v$ in the game $(\mathcal{A}, O_{\mathsf{first}}(Y))$. Then, by Lemma 7, for every play $\pi \in plays(S^{\circlearrowleft}, v)$, every cycle in $cycles(\pi)$ satisfies $Y^\sigma$, i.e., $\pi \in O^{\mathcal{A}}_{\mathsf{all}}(Y^\sigma)$. By definition of $Y$-greedy, this means that $S^{\circlearrowleft}$ is winning from $v$ in the game $(\mathcal{A}, O)$. □

We now have the ingredients for the proof of the Strategy Transfer Theorem:

**Theorem 7** *(Strategy transfer). Let $(\mathcal{A}, O)$ be a $Y$-greedy game, and let $\sigma \in \{0,1\}$.*

1. *The winning regions for Player $\sigma$ in the games $(\mathcal{A}, O)$ and $(\mathcal{A}, O_{first}(Y))$ coincide.*
2. *For every memoryless strategy $S$ for Player $\sigma$, and vertex $v \in V$ in arena $\mathcal{A}$: $S$ is winning from $v$ in the game $(\mathcal{A}, O)$ if and only if $S$ is winning from $v$ in the game $(\mathcal{A}, O_{first}(Y))$.*

**Proof.** Let $Y \subseteq \mathbb{U}^*$ be a cycle property and $\mathcal{A}$ an arena. Suppose that $(\mathcal{A}, O)$ is $Y$-greedy. We begin by proving the first item. Use Corollary 2 to get that for $\sigma \in \{0, 1\}$,

$$\mathsf{WR}^\sigma(\mathcal{A}, O_{\mathsf{first}}(Y)) \subseteq \mathsf{WR}^\sigma(\mathcal{A}, O).$$

Since first-cycle games are determined (Lemma 3), the winning regions $\mathsf{WR}^0(\mathcal{A}, O_{\mathsf{first}}(Y))$ and $\mathsf{WR}^1(\mathcal{A}, O_{\mathsf{first}}(Y))$ partition $V$. Thus, since $\mathsf{WR}^0(\mathcal{A}, O)$ and $\mathsf{WR}^1(\mathcal{A}, O)$ are disjoint, the containments above are equalities, as required for item 1.

We prove the second item. Since $S = S^{\circlearrowleft}$ if $S$ is memoryless, conclude by Corollary 2: if $S$ is winning from $v$ in the game $(\mathcal{A}, O_{\mathsf{first}}(Y))$ then it is winning from $v$ in the game $(\mathcal{A}, O)$. For the other direction, assume by contraposition that $S$ is not winning from $v$ in the game $(\mathcal{A}, O_{\mathsf{first}}(Y))$. Since $S$ is memoryless, plays of $\mathcal{A}$ consistent with $S$ are exactly infinite paths in the induced sub-arena $\mathcal{A}^{\|S}$. Hence, there is a path $\pi$ in the induced solitaire arena $\mathcal{A}^{\|S}$ for which the first cycle, say $\pi[i, j]$, satisfies $Y^{1-\sigma}$. Define the infinite path $\pi' := \pi[1, i-1] \cdot (\pi[i, j])^\omega$ and note that, being a path in $A^{\|S}$, it is a play of $\mathcal{A}$ consistent with $S$. Moreover, $\pi'$ has the property that every cycle in its cycles-decomposition (i.e., $\pi[i, j]$) satisfies $Y^{1-\sigma}$. Since $(\mathcal{A}, O)$ is $Y$-greedy, $S$ is not winning from $v$ in the game $(\mathcal{A}, O)$. □

Since FCGs are determined (Lemma 3, Page 200) use Theorem 7 to get:

**Corollary 3.** *Every $Y$-greedy game $(\mathcal{A}, O)$ is determined, has the same winning regions as $(\mathcal{A}, O_{first}(Y))$, and is point-wise (uniform) memoryless determined if and only if the game $(\mathcal{A}, O_{first}(Y))$ is point-wise (uniform) memoryless determined.*

We now state our main result concerning $Y$-greedy games.

**Theorem 8** *(Memoryless determinacy characterisation of greedy games). For every cycle property $Y$, the following are equivalent:*

1. *$Y$ is closed under cyclic permutations, and every arena is $Y$-resettable.*
2. *Every $Y$-greedy game is uniform memoryless determined.*

**Proof.** This follows from Corollary 3, Theorem 6, and the fact that for every arena $\mathcal{A}$ there is a $Y$-greedy game with arena $\mathcal{A}$, for example, the game $(\mathcal{A}, O_{\mathsf{all}}(Y))$. □

## 7. An easy to check sufficient condition on $Y$ ensuring memoryless determinacy of FCGs

As we have seen $Y$-resettability together with closure under cyclic permutations provides a full characterization of those cycle properties $Y$ for which the games in $\mathrm{Fcg}(Y)$, as well as any $Y$-greedy games, are memoryless determined. Even though, in many cases, checking whether $Y$ is such that every arena $\mathcal{A}$ (or just the arena(s) of interest) is $Y$-resettable is not too difficult, we believe that in practice it is much easier to use the following condition on $Y$ which avoids reasoning about arenas altogether. The condition we suggest is the following:

$$Y \text{ and } \neg Y \text{ are closed under concatenation.}$$

As we later show, this condition (together with closure under cyclic permutations) ensures that every arena $\mathcal{A}$ is $Y$-resettable, and thus by Theorem 6, that all games in $\mathrm{Fcg}(Y)$ are memoryless determined. On the other hand, as we discuss in Section 8, there is a cycle property $Y$ which is closed under cyclic permutations and for which every arena $\mathcal{A}$ is $Y$-resettable, even though $Y$ does not satisfy the condition. Thus, this condition cannot replace $Y$-resettability as a necessary condition for memoryless determinacy of all games in $\mathrm{Fcg}(Y)$. However, it is applicable in a wide variety of cases, and is usually very easy to check. Consider for example the cycle properties given in Section 2. For each of these properties $Y$, while proving that every arena is $Y$-resettable may not be hard, checking whether $Y$ satisfies the condition above is almost completely trivial. Note that cyc-EvenLen, which is the only property among these which is closed under cyclic permutations but fails to satisfy this condition, would also fail to satisfy any other condition that guarantees memoryless determinacy since it actually admits FCGs that are not memoryless determined (cf. Theorem 3, Page 204).

Our goal in the rest of this section is to prove the following theorem:

**Theorem 9** *(Easy to check). Let $Y \subseteq \mathbb{U}^*$ be a cycle property. If $Y$ is closed under cyclic permutations, and both $Y$ and $\neg Y$ are closed under concatenation, then every arena $\mathcal{A}$ is $Y$-resettable.*

By Theorem 6 we get:

**Corollary 4.** *Let $Y \subseteq \mathbb{U}^*$ be a cycle property. If $Y$ is closed under cyclic permutations, and both $Y$ and $\neg Y$ are closed under concatenation, then every game in $\mathrm{Fcg}(Y)$ is uniform memoryless determined.*

**Remark 2.** Consider the cycle property $Y = \text{cyc-GoodForEnergy}$ (recall from the definitions this means that either the energy level is positive, or it is zero and the largest priority occurring is even). Note that $Y$ is closed under cyclic permutations, and both $Y$ and $\neg Y$ are closed under concatenation. Conclude that every game in $\mathrm{Fcg}(Y)$ is pointwise-memoryless determined. This fact allows one to obtain a proof of Lemma 4 in [5] that no longer relies on the incorrect result from [4].

We begin by introducing a new kind of objective $O^{\mathcal{A}}_{\mathsf{tail}}(Y)$:

**Definition 7.** For an arena $\mathcal{A}$ and cycle property $Y$, define the objective $O^{\mathcal{A}}_{\mathsf{tail}}(Y)$ to consist of all plays $\pi$ such that there is a suffix $\pi'$ of $\pi$ with the property that every cycle in $\mathit{cycles}(\pi')$ satisfies $Y$.

Note that this is not the same as saying that $\lambda(C) \in Y$ for all but finitely many cycles $C$ in $\mathit{cycles}(\pi)$.[10]

**Definition 8.** An arena $\mathcal{A}$ is *$Y$-unambiguous* if $O^{\mathcal{A}}_{\mathsf{tail}}(Y) \cap O^{\mathcal{A}}_{\mathsf{tail}}(\neg Y) = \emptyset$.

In other words, $\mathcal{A}$ is $Y$-unambiguous if no play in $\mathcal{A}$ has two suffixes, $\pi, \pi'$ such that every cycle in the cyclic decomposition of $\pi$ is in $Y$, and every cycle in the cyclic decomposition of $\pi'$ is not in $Y$.

In order to prove Theorem 9 we have to show that, for $Y$ satisfying the assumptions of the theorem, for every arena $\mathcal{A}$, every Player $\sigma \in \{0,1\}$, and every node $z$ in $\mathcal{A}$, we have that $\mathsf{WR}^{\sigma}(\mathcal{A}, O_{\mathsf{first}}(Y)) = \mathsf{WR}^{\sigma}(\mathcal{A}, O_{z\text{-}\mathsf{first}}(Y))$. The proof is in three parts:

*Part* 1. If $Y$ is closed under cyclic permutations, and both $Y$ and $\neg Y$ are closed under concatenation, then every arena $\mathcal{A}$ is $Y$-unambiguous.
*Part* 2. If $\mathcal{A}$ is $Y$-unambiguous then $\mathsf{WR}^{\sigma}(\mathcal{A}, O_{\mathsf{first}}(Y)) = \mathsf{WR}^{\sigma}(\mathcal{A}, O_{\mathsf{tail}}(Y))$.
*Part* 3. If $\mathcal{A}$ is $Y$-unambiguous then $\mathsf{WR}^{\sigma}(\mathcal{A}, O_{z\text{-}\mathsf{first}}(Y)) = \mathsf{WR}^{\sigma}(\mathcal{A}, O_{\mathsf{tail}}(Y))$.

Observe that parts 2 and 3 imply that if an arena $\mathcal{A}$ is $Y$-unambiguous then it is $Y$-resettable.

We first need a couple of definitions and a few easy lemmas. Say that $Y$ is *closed under insertions* if it is closed under concatenation and: $ac \in Y$ and $b \in Y$ imply that $abc \in Y$, for all $a, b, c \in \mathbb{U}^*$. The *closure of $Y$ under insertions*, is defined to be the smallest subset of $\mathbb{U}^*$ (with respect to set containment) that contains $Y$ and is closed under insertions.

**Lemma 8.** *If $Y$ is closed under cyclic permutations and under concatenation then $Y$ is closed under insertions.*

**Proof.** Assume that $ac, b \in Y$. We have that $ac \in Y \Longrightarrow ca \in Y \Longrightarrow cab \in Y \Longrightarrow abc \in Y$. The middle implication is since $Y$ is closed under concatenation, and the other two implications are since $Y$ is closed under cyclic permutations. □

Fix an arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$. Given a simple path $s \in E^*$, and a path $u \in E^*$ such that $\mathsf{end}(s) = \mathsf{start}(u)$, write $\mathit{labels}(s,u) = \{\lambda(C) \mid C \in \mathit{cycles}(s,u)\}$ for the set of the labels of the cycles output by CycDec$(s,u)$.

**Lemma 9.** *Given an arena $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$, let $s, u$ be paths in $\mathcal{A}$ such that $u$ is a cycle and $\mathsf{end}(s) = \mathsf{start}(u) = v$. We have that:*

1. *if $\mathit{stack}(s,u) = s$ then the insertion closure of $\mathit{labels}(s,u)$ contains $\lambda(u)$;*
2. *if $v$ is the only node that appears on both $s$ and $u$ then $\mathit{stack}(s,u) = s$.*

**Proof. Sketch.** For the first item, note that the assumption $\mathit{stack}(s,u) = s$ implies that the cycles in $\mathit{cycles}(s,u)$ contain exactly the edges of $u$. Hence, it is not hard to see that $\lambda(u)$ is in the insertion closure of $\mathit{labels}(s,u)$. Intuitively, the algorithm output the cycles in $\mathit{cycles}(s,u)$ by "popping them out" of $u$. Thus, we can reverse this process and reconstruct $u$ using insertion operations on the elements of $\mathit{cycles}(s,u)$.

**Details.** More formally, let $C^1, \ldots C^m$ be the elements of $\mathit{cycles}(s,u)$ in the order they were output. We iteratively construct a string $w \in E^*$ until we get $w = u$. We start with $w$ being the empty string, and count down from $m$ to 1. At step $i$ in the count, we insert the cycle $C^i$ into the correct position in $w$, as follows: let $j$ be such that $u_j = C^i_0$ (i.e., the first edge of $C^i$), and set $\mathrm{w} := \mathrm{w}_1 \ldots \mathrm{w}_h \cdot C^i \cdot \mathrm{w}_{h+1} \ldots \mathrm{w}_{|\mathrm{w}|}$, where $h$ is the maximal index such that all the edges $\mathrm{w}_1, \ldots, \mathrm{w}_h$ are in $\{u_1, \ldots, u_{j-1}\}$.

It is easy to see that when the construction ends w contains exactly the edges appearing in $C^1, \ldots C^m$ and thus, as noted before, exactly the edges of $u$. We claim that the edges are also ordered correctly (i.e., if $a < b$ then $\mathrm{w}_a$ appears in $u$ before $\mathrm{w}_b$) and thus, $\mathrm{w} = u$ as required. Assume by way of contradiction that the correct ordering in w was violated for the first time when $C^i$ was inserted. Let $u_j$ be the first edge of $C^i$ and $h$ be the position where $C^i$ was inserted into w, as defined before. Recall that after the insertion the constructed string is $\mathrm{w}_1 \ldots \mathrm{w}_h \cdot C^i \cdot \mathrm{w}_{h+1} \ldots \mathrm{w}_{|\mathrm{w}|}$. Observe that since all edges of $C_i$ are internally ordered correctly, and all of them correctly appear (by our choice of $h$) after $\mathrm{w}_1 \ldots \mathrm{w}_h$, the only possible violation is that some edge $u_t$ in $C^i$ incorrectly appears before the edge $\mathrm{w}_{h+1} = u_r$, i.e., that $j < r < t$. Also note that at the step where $C^i$ was output, all edges above $u_j$ that were on the stack were popped, and all edges up to $u_t$ were already processed. Hence, it can not be that the edge $u_r$ was output by the algorithm after $C^i$, which is a contradiction to the fact that $u_r$ is already in w before the insertion of $C^i$. This completes the proof of the claim.

[10] For instance, consider the arena in Fig. 4, take $Y$ to be cyc-EvenLen, and let $\pi := [(v_1,v_2)(v_2,v_1)(v_1,v_3)(v_3,v_2)(v_2,v_4)(v_4,v_1)]^{\omega}$. Note that i) decomposing the suffix $\pi'$ starting with the second edge results in all cycles having odd length, and ii) it is not the case that almost all cycles in the cycles-decomposition of $\pi$ (i.e., starting with the first edge) have odd length – in fact, they all have even length.

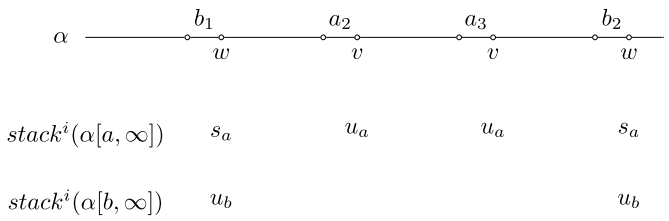



**Fig. 6.** Visualisation of the processing of the play $\alpha$.

For the second item, observe that the assumption made there implies that, while processing $u$, the algorithm CycDec$(s,u)$ cannot pop any edge in $s$. Hence, $stack(s,u)=s$ does not hold only if there exists $j$ such that the edge $u_j=(v,v')$ is pushed on top of $s$ but never popped. Observe that $v'\neq v=src(u)=trg(u)$ (the first inequality is since a self-loop is always popped, the rest by our assumption that $u$ is a cycle that starts in $v$), and thus $u_j$ is not the last edge of $u$. But this is a contradiction since the (non-empty) suffix $u_{j+1}\dots u_{|u|}$ of $u$ contains at least one edge $e'$ with $\mathsf{end}(e')=v$ (namely $u_{|u|}$), and the algorithm would have popped the edge $u_j$ when it encountered the first edge that ends in $v$. □

We are now ready to show part 1 in the proof of Theorem 9.

**Proposition 2.** *Let $Y\subseteq\mathbb{U}^*$ be a cycle property. If $Y$ is closed under cyclic permutations, and both $Y$ and $\neg Y$ are closed under concatenation, then every arena $\mathcal{A}$ is $Y$-unambiguous.*

**Proof.** First, note that since $Y$ is closed under cyclic permutations then so is $\neg Y$. By Lemma 8, we have that $Y$, as well as $\neg Y$, are closed under insertions.

Assume by way of contradiction that there is an arena $\mathcal{A}=(V_0,V_1,E,\mathbb{U},\lambda)$ which is not $Y$-unambiguous, and take a play $\alpha$ in the intersection of $O^{\mathcal{A}}_{\mathsf{tail}}(Y)$ and $O^{\mathcal{A}}_{\mathsf{tail}}(\neg Y)$. Let $a,b\in\mathbb{N}$ be such that every cycle in $cycles(\alpha[a,\infty])$ satisfies $Y$, and every cycle in $cycles(\alpha[b,\infty])$ satisfies $\neg Y$.

For $i,j$ such that $a\leq i<j$, write $Out(i,j)=\{cycle^l(\alpha[a,\infty])\mid i\leq l\leq j\}\setminus\{\epsilon\}$ to be the set of cycles output by CycDec$(\epsilon,\alpha[a,\infty])$ while processing the edges $\alpha_i\dots\alpha_j$, and note that for every $C\in Out(i,j)$ we have that $\lambda(C)\in Y$. Similarly, for $i,j$ such that $b\leq i<j$, write $Out^{\neg}(i,j)=\{cycle^l(\alpha[b,\infty])\mid i\leq l\leq j\}\setminus\{\epsilon\}$ to be the set of cycles output by CycDec$(\epsilon,\alpha[b,\infty])$ while processing the edges $\alpha_i\dots\alpha_j$, and note that for every $C\in Out^{\neg}(i,j)$ we have that $\lambda(C)\in\neg Y$.

Since, for every $i\geq 1$, the stack content $stack^i(\alpha[a,\infty])$ is of length at most $|V|-1$, there is at least one stack content that appears infinitely often. Thus, let $u_a\in E^*$ be a path of minimal length such that the set $A=\{i\in\mathbb{N}\mid u_a=stack^i(\alpha[a,\infty])\}$ is infinite. Similarly, let $u_b\in E^*$ be a path of minimal length such that $B=\{i\in\mathbb{N}\mid u_b=stack^i(\alpha[b,\infty])\}$ is infinite.

We assume w.l.o.g. (otherwise we simply replace $A$ and $B$ by appropriate infinite subsets of themselves until each of the following conditions is satisfied) that †:

(i) $u_a=stack^i(\alpha[a,\infty])$ for every $i\in A$, and $u_b=stack^i(\alpha[b,\infty])$ for every $i\in B$;
(ii) $\mathsf{end}(\alpha_i)=\mathsf{end}(\alpha_j)$ for all $i,j\in A$ and $\mathsf{end}(\alpha_i)=\mathsf{end}(\alpha_j)$ for all $i,j\in B$ (recall that the nodes on $\alpha$ come from the finite set $V$);
(iii) there exists $s_a\in E^*$ such that $s_a=stack^i(\alpha[a,\infty])$ for all $i\in B$.

Pick indices $b_1,b_2\in B$ and $a_2,a_3\in A$ in such a way that $b_1<a_2<a_3<b_2$. Fig. 6 may aid in visualising the current state of affairs. In this figure, $w:=\mathsf{end}(\alpha_{b_1})=\mathsf{end}(u_b)=\mathsf{end}(\alpha_{b_2})$ and $v:=\mathsf{end}(\alpha_{a_2})=\mathsf{end}(u_a)=\mathsf{end}(\alpha_{a_3})$.

Our aim now is to show that the above state of affairs leads to a contradiction, and thus deduce that it can not be that $\alpha\in O^{\mathcal{A}}_{\mathsf{tail}}(Y)\cap O^{\mathcal{A}}_{\mathsf{tail}}(\neg Y)$. The contradiction we will show is that $\lambda(\alpha[b_1+1,b_2])$ is both in $Y$ and in its complement $\neg Y$, which is impossible. We will do that by showing how to build $\lambda(\alpha[b_1+1,b_2])$ in two different ways: the first by using cyclic permutations and concatenations of strings that are the labels of cycles that satisfy $Y$, and the second by doing the same but with cycles that satisfy $\neg Y$. Since by the assumption of the proposition $Y$ and $\neg Y$ are closed under these string operations, the contradiction is reached.

We first show that $\alpha[b_1+1,b_2]$ (which by †(*ii*) is a cycle) satisfies $\neg Y$. Intuitively (refer to Fig. 6), this follows from the fact that all cycles output by the algorithm CycDec$(\epsilon,\alpha[b,\infty])$ while processing $\alpha[b_1+1,b_2]$ satisfy $\neg Y$, and the fact that before and after processing $\alpha[b_1+1,b_2]$ the stack of this algorithm is $u_b$, and thus by Lemma 9 we have that $\lambda(\alpha[b_1+1,b_2])$ is in the insertion closure of the labels of these cycles, and thus also in $\neg Y$ (recall that $\neg Y$ is closed under insertions).

We next show that $\lambda(\alpha[b_1+1,b_2])\in Y$. Observe (refer to Fig. 6) that $\alpha[b_1+1,b_2]=\alpha[b_1+1,a_2]\alpha[a_2+1,a_3])\alpha[a_3+1,b_2]$. Since $Y$ is closed under concatenation and cyclic permutations, to show that $\lambda(\alpha[b_1+1,b_2])\in Y$ it is enough to show that $\lambda(\alpha[a_2+1,a_3])\in Y$ and $\lambda(x)\in Y$, where $x:=\alpha[a_3+1,b_2]\alpha[b_1+1,a_2]$. The fact that $\lambda(\alpha[a_2+1,a_3])\in Y$ follows from a symmetric argument that mimics the one used above to prove that $\alpha[b_1+1,b_2]$ satisfies $\neg Y$. It remains to show that $\lambda(x)\in Y$.

To see that $\lambda(x) \in Y$, note that when the algorithm CycDec$(\epsilon, \alpha[a, \infty])$ finishes processing $\alpha[a_3 + 1, b_2]$ it has the same stack content (namely $s_a$) that it had when it previously started processing $\alpha[b_1 + 1, a_2]$. Thus, if we imagine that after processing $\alpha[a_3 + 1, b_2]$, instead of continuing to process $\alpha[b_2 + 1, \infty]$, we feed the algorithm again with $\alpha[b_1 + 1, a_2]$ (i.e., we let it process the second part of $x$), we will get (by Lemma 2) that it will behave exactly the same as when it first processed $\alpha[b_1 + 1, a_2]$ as part of the prefix $\alpha[a, a_2]$. We thus obtain that while processing $x$ this way starting with stack $u_a$, the algorithm ends with stack $u_a$ and outputs only cycles that satisfy $Y$ (recall that all cycles output by CycDec$(\epsilon, \alpha[a, \infty])$ do). Thus, by Lemma 9, we have that $\lambda(x)$ is in the insertion closure of the labels of cycles that satisfy $Y$, and thus also in $Y$ (recall that $Y$ is closed under insertions). □

The following Proposition deals with Part 2 in the proof of Theorem 9.

**Proposition 3.** *Fix a $Y$-unambiguous arena $\mathcal{A}$. Then for $\sigma \in \{0, 1\}$,*

$$WR^{\sigma}(\mathcal{A}, O_{first}(Y)) = WR^{\sigma}(\mathcal{A}, O_{tail}(Y)).$$

**Proof.** By Theorem 7 (item 1) it is sufficient to show that if $\mathcal{A}$ is $Y$-unambiguous then the game $(\mathcal{A}, O_{\text{tail}}(Y))$ is $Y$-greedy. So, fix a play $\pi$ in $\mathcal{A}$. If every cycle in the cycles-decomposition of $\pi$ satisfies $Y$ then certainly $\pi \in O_{\text{tail}}(Y)$ (just take $\pi$ itself as the required suffix). On the other hand, if every cycle in the cycles-decomposition of $\pi$ satisfies $\neg Y$ then for the same reason $\pi \in O_{\text{tail}}(\neg Y)$. However, since $\mathcal{A}$ is $Y$-unambiguous, $\pi \notin O_{\text{tail}}(Y)$, as required. □

The following Proposition deals with Part 3.

**Proposition 4.** *Fix a $Y$-unambiguous arena $\mathcal{A}$ and vertex $z \in V$. Then for $\sigma \in \{0, 1\}$,*

$$WR^{\sigma}(\mathcal{A}, O_{z\text{-}first}(Y)) = WR^{\sigma}(\mathcal{A}, O_{tail}(Y)).$$

**Proof.** Let $\mathcal{A} = (V_0, V_1, E, \mathbb{U}, \lambda)$. We first claim that for every $\sigma \in \{0, 1\}$, we have that $\mathsf{WR}^{\sigma}(\mathcal{A}, O_{z\text{-first}}(Y)) \subseteq \mathsf{WR}^{\sigma}(\mathcal{A}, O_{\text{tail}}(Y))$. To see that the Proposition follows from this claim note that $\mathsf{WR}^0(\mathcal{A}, O_{z\text{-first}}(Y))$ and $\mathsf{WR}^1(\mathcal{A}, O_{z\text{-first}}(Y))$ partition $V$ since the game $(\mathcal{A}, O_{z\text{-first}}(Y))$, being finitary, is determined (by Lemma 3). Since $\mathsf{WR}^0(\mathcal{A}, O_{\text{tail}}(Y))$ and $\mathsf{WR}^1(\mathcal{A}, O_{\text{tail}}(Y))$ are disjoint, the containments above must be equalities.

To prove the claim, recall that since $\mathcal{A}$ is $Y$-unambiguous then $O^{\mathcal{A}}_{\text{tail}}(Y) \cap O^{\mathcal{A}}_{\text{tail}}(\neg Y) = \emptyset$. Hence, a play $\pi$ in the game $(\mathcal{A}, O_{\text{tail}}(Y))$ is won by Player 0 (resp. Player 1) if $\pi$ has a suffix $\pi'$ for which every cycle in $cycles(\pi')$ is in $Y$ (resp. $\neg Y$). It is thus enough to show that: (†) for every $\sigma \in \{0, 1\}$, and every node $v$ in $\mathcal{A}$, given a strategy $S$ that is winning from $v$ for Player $\sigma$ in the game $(\mathcal{A}, O_{z\text{-first}}(Y))$, we can construct a strategy $T$ for Player $\sigma$ in the game $(\mathcal{A}, O_{\text{tail}}(Y))$, such that every play $\pi \in plays(T, v)$ has a suffix $\pi'$ for which every cycle in $cycles(\pi')$ satisfies $Y^{\sigma}$.

Consider first the case where $z \notin reach(S, v)$, or that $z \in reach(S, v)$ but that every path in $plays(S, v)$ that visits $z$ contains a cycle before the first occurrence of $z$ on the path. Observe that this implies that $S$ is winning from $v$ in the game $(\mathcal{A}, O_{\text{first}}(Y))$. In this case we let $T = S^{\circlearrowleft}$, where $S^{\circlearrowleft}$ is the pumping strategy of $S$ (Definition 6). By Lemma 7 we have that, for every $\pi \in plays(T, v)$, all the cycles in $cycles(\pi)$ satisfy $Y^{\sigma}$, and thus (†) holds with $\pi' = \pi$.

Consider now the remaining case that $z \in reach(S, v)$ and that there is a simple path $\rho$ from $v$ to $z$ which is consistent with $S$. Define a strategy $S_z$ for Player $\sigma$ in the game $(\mathcal{A}, O_{\text{first}}(Y))$ as follows: for every $u \in V^*V_{\sigma}$, let $S_z(u) := S(\rho u)$ if $u$ starts in $z$, and otherwise define it arbitrarily (i.e., $S_z(u) = w$, where $w$ is some node with $(\mathsf{end}(u), w) \in E$). Observe that, for every $\pi \in plays(S_z, z)$, the play $\rho\pi$ is in $plays(S, v)$, and since $\rho$ is a simple path that ends in $z$, Player $\sigma$ wins the play $\rho\pi$ in the game $(\mathcal{A}, O_{z\text{-first}}(Y))$ iff the first cycle on $\pi$ satisfies $Y^{\sigma}$. Thus, since $S$ is winning from $v$, we have that $S_z$ is winning from $z$ in the game $(\mathcal{A}, O_{\text{first}}(Y))$.

We can now construct the desired strategy $T$ for Player $\sigma$ in the game $(\mathcal{A}, O_{\text{tail}}(Y))$. The strategy $T$ works as follows: as long as a play does not touch the node $z$ the strategy $T$ behaves like the pumping strategy $S^{\circlearrowleft}$ of $S$; however, once (and if) $z$ is reached, $T$ erases its memory and switches to behave like the pumping strategy $(S_z)^{\circlearrowleft}$ of $S_z$ (starting from $z$). Recall that we have to show that every play $\pi \in plays(T, v)$ has a suffix $\pi'$ for which every cycle in $cycles(\pi')$ satisfies $Y^{\sigma}$. Informally, if $z$ does not appear on $\pi$ then $\pi$ is consistent with $S^{\circlearrowleft}$, and by Lemma 7 every cycle in $cycles(\pi)$ satisfies $Y^{\sigma}$, and we can take $\pi' = \pi$. On the other hand, if $z$ appears on $\pi$ then $tail_z(\pi)$ is consistent with $(S_z)^{\circlearrowleft}$, and thus by Lemma 7 every cycle in $cycles(tail_z(\pi))$ satisfies $Y^{\sigma}$, and we can take $\pi' = tail_z(\pi)$.

Formally, define the strategy $T$ as follows: for every $u \in V^*V_{\sigma}$,

$$T(u) := \begin{cases} S^{\circlearrowleft}(u) & \text{if } z \text{ does not appear on } u, \\ (S_z)^{\circlearrowleft}(tail_z(u)) & \text{otherwise.} \end{cases}$$

Given $\pi \in plays(T, v)$, either $z$ appears on $\pi$ or not. If it does not, then $\pi \in plays(S^{\circlearrowleft}, v)$, and by Lemma 7 every cycle $C$ in $cycles(\pi)$ is the first cycle of some path $\rho$ consistent with $S$ and starting with $v$, that only uses edges from $\pi$. Thus, $z$ does not appear on $\rho$, and since $S$ is winning from $v$ in the game $(\mathcal{A}, O_{z\text{-first}}(Y))$, we have that $C$ satisfies $Y^{\sigma}$. If $z$ does appear

on $\pi$, we argue that $tail_z(\pi) \in plays((S_z)^{\circlearrowleft}, z)$, i.e., that $tail_z(\pi)$ is consistent with $(S_z)^{\circlearrowleft}$. Indeed, if $m = |head(\pi)|$, then for every $j \geq 1$ for which $\mathsf{start}(tail_z(\pi)_j) \in V_\sigma$ we have:

$$\begin{aligned}\mathsf{end}(tail_z(\pi)_j) &= T(\pi[1, m+j]) \\ &= (S_z)^{\circlearrowleft}(tail_z(\pi[1, m+j])) \\ &= (S_z)^{\circlearrowleft}(\pi[m+1, m+j]) \\ &= (S_z)^{\circlearrowleft}((tail_z(\pi))[1, j]).\end{aligned}$$

By Lemma 7, since $tail_z(\pi) \in plays((S_z)^{\circlearrowleft}, z)$, every cycle in $cycles(tail_z(\pi))$ satisfies $Y^\sigma$. Overall, in the first case (†) holds with $\pi' = \pi$, and in the second with $\pi' = tail_z(\pi)$, which completes the proof of the claim. □

This concludes the proof of Theorem 9.

## 8. A recipe for proving that a game is memoryless determined

We synthesise the results of the previous section and provide a practical way of deducing uniform memoryless determinacy of many infinite-duration games.

First, we get the following sufficient condition for memoryless determinacy from Theorems 8 and 9.

**Theorem 10.** *Let $Y$ be a cycle property that is closed under cyclic permutations, and such that both $Y$ and $\neg Y$ are closed under concatenation. Let $W$ be a winning condition. Every $Y$-greedy game $(\mathcal{A}, O(W))$ is uniform memoryless determined.*

Second, many winning conditions for infinite-duration games (for example parity and mean-payoff) are such that the outcome of a play does not depend on any finite prefix of the play, but only on some "infinite" property of the play. Such winning conditions are usually called *prefix-independent*. Formally:

**Definition 9.** Say that a winning condition $W \subseteq \mathbb{U}^\omega$ is *prefix-independent* if $c_1 c_2 \ldots \in W$ implies that the suffix $c_i c_{i+1} \ldots \in W$ for every $i \in \mathbb{N}$.

In practice, showing whether or not a given $W$ is prefix-independent is usually very easy. The relevance of prefix-independence to our work is captured by the following theorem.

**Theorem 11.** *Let $W$ be a prefix-independent winning condition, and let $Y$ be a cycle-property that is closed under cyclic permutations. For every arena $\mathcal{A}$, if the game $(\mathcal{A}, O(W))$ is $Y$-greedy then it is uniform memoryless determined.*

**Proof.** By Theorem 8 it is sufficient to prove that every arena $\mathcal{A}$ is $Y$-resettable. By Propositions 3 and 4 it is thus sufficient to prove that $\mathcal{A}$ is $Y$-unambiguous. So, assume by way of contradiction there is a play $\pi$ and indices $a, b \in \mathbb{N}$ be such that every cycle in $cycles(\pi[a, \infty])$ satisfies $Y$, and every cycle in $cycles(\pi[b, \infty])$ satisfies $\neg Y$. Since $(\mathcal{A}, O(W))$ is $Y$-greedy on $\mathcal{A}$ we get that, in the game $(\mathcal{A}, O(W))$, Player 0 wins $\pi[a, \infty]$, and Player 1 wins $\pi[b, \infty]$. But this is a contradiction to the assumption that $W$ is prefix-independent. □

Given a winning condition $W$ and a set of arenas of interest $\mathcal{C}$ (in many cases $\mathcal{C}$ is taken to be all arenas), Theorems 10 and 11 suggest the following recipe for proving that the game $(\mathcal{A}, O(W))$ is uniform memoryless determined for every $\mathcal{A} \in \mathcal{C}$.

Step 1. Finitise the winning condition $W \subseteq \mathbb{U}^\omega$ to get a cycle property $Y \subseteq \mathbb{U}^*$ that is closed under cyclic permutations.
Step 2. Show that the game $(\mathcal{A}, O(W))$ is $Y$-greedy for every $\mathcal{A} \in \mathcal{C}$.
Step 3. Show that either:
  (a) Both $Y$ and $\neg Y$ are closed under concatenation; or that:
  (b) $W$ is prefix independent.

We illustrate the use of the recipe with some examples.

**Example 1.** We will use the recipe to prove that every parity game is memoryless determined. For Step 1, a natural finitisation of the parity condition $W \subset \mathbb{Z}^\omega$ — which says that the largest priority occurring infinitely often is even — is the cycle property $\mathsf{cyc\text{-}Parity} \subset \mathbb{Z}^*$ which says that the largest priority occurring is even. This property is clearly closed under

cyclic permutations. For Step 2, it is easy to verify that every parity game is cyc-Parity-greedy, and for Step 3, it is immediate that both cyc-Parity and ¬cyc-Parity are closed under concatenation (as it happens, it is also easy to check that $W$ is prefix-independent).

**Example 2.** We now consider a slightly more subtle application of the recipe. Consider the following game,[11] played on an arena $\mathcal{A}$ whose vertices are labelled[12] using the alphabet $\mathbb{U} = \{a, b\}$. The winning condition $W \subset \mathbb{U}^\omega$ consists of those infinite sequences that contain infinitely many $a$'s and infinitely many $b$'s. The natural finite version of $W$ is the cycle property $Y \subset \mathbb{U}^*$ consisting of strings which contain at least one $a$ and at least one $b$, and it is clearly closed under cyclic permutations. To see that $W$ is $Y$-greedy on $\mathcal{A}$, observe that if all cycles in $cycles(\pi)$ satisfy $Y$ then certainly $\pi \in W$. On the other hand, if all cycles in $cycles(\pi)$ satisfy $\neg Y$ then no edge that appears on such a cycle goes from a node labelled $a$ to a node labelled $b$. Thus by Lemma 1 (Page 199) $\pi$ does not satisfy $W$.

Unfortunately, $\neg Y$ is not closed under concatenation, which prevents us from applying Step 3(a). However, since $W$ is clearly prefix-independent, we can apply Step 3(b) and conclude that $(\mathcal{A}, O(W))$ is memoryless determined.

**Example 3.** We conclude with a more sophisticated use of the recipe, applied to the initial credit problem of energy games. We show that either there is an initial credit with which Player 0 (the "energy" player) wins, or that for every initial credit Player 1 wins. In both cases, we show that the winner can use a memoryless strategy. A natural finitisation of the energy condition $W_r \subset \mathbb{Z}^\omega$ — which says that at every point along a play, the sum of the initial credit $r$ and the labels of the edges already traversed is not negative — is the cycle property cyc-Energy $\subset \mathbb{Z}^*$ which says that the sum of the numbers is non-negative. This property is clearly closed under cyclic permutations. Given an arena $\mathcal{A}$, consider the game $(\mathcal{A}, O_{\text{all}}(\text{cyc-Energy}))$. We first claim that if the initial credit is at least $r = -t(|V| - 1)$, where $t$ is the minimum amongst the negative weights of the arena, the winning regions of the energy game and the game $(\mathcal{A}, O_{\text{all}}(\text{cyc-Energy}))$ coincide and winning strategies transfer from the latter to the former. To see that, observe that a play won by Player 1 in $(\mathcal{A}, O_{\text{all}}(\text{cyc-Energy}))$ is also won by her in the energy game since the energy along the play tends to $-\infty$. On the other hand, let $\pi$ be a play won by Player 0 in $(\mathcal{A}, O_{\text{all}}(\text{cyc-Energy}))$, and consider some prefix $\pi'$ of it. Recall that, by Lemma 1, at most $|V| - 1$ edges are not on $cycles(\pi')$, and thus the energy level at the end of $\pi'$ is at least the initial credit plus $t(|V| - 1)$. Hence, an initial credit of $r$ suffices for $\pi$ to be winning for Player 0 also in the energy game. It remains to show, using the recipe, that $(\mathcal{A}, O_{\text{all}}(\text{cyc-Energy}))$ is memoryless determined. As noted before (see Example 1 after Definition 5) every $(\mathcal{A}, O_{\text{all}}(Y))$ is $Y$-greedy, which completes Step 2. Since step 3$a$ is immediate for cyc-Energy we are done.

## 9. Discussion and future work

The central algorithmic problem for a class of determined graph games is to decide, given an arena and a starting vertex, which of the players has a winning strategy. We have seen a PSPACE upper bound on the complexity of solving games in FCG$(Y)$ (assuming $Y$ is computable in PSPACE), and that there are very simple cycle properties $Y$ for which the complexity is PSPACE-complete. What about other $Y$s such as cyc-Parity and cyc-MeanPayoff$_\nu$? Our Strategy Transfer result (Theorem 7) implies that the complexity of solving FCG$(Y)$ is the same as that of solving $Y$-greedy games. For instance, since parity games are cyc-Parity-greedy, and the complexity of solving them is not known to be in PTIME — except on restricted classes of arenas, e.g., bounded DAG-width ([2]) and bounded trap-depth ([9]) — the same is true of the complexity of solving games from FCG(cyc-Parity).

On the other hand, since there are cycle properties $Y$ such that some games in FCG$(Y)$ are memoryless determined and some are not (for instance, take $Y =$ cyc-EvenLen), the following algorithmic problem naturally presents itself: what is the complexity of deciding, given (a finite description of) $Y$ and an arena $\mathcal{A}$, whether or not the game $(\mathcal{A}, O_{\text{first}}(Y))$ is memoryless determined? We believe that this problem can be addressed using techniques developed in this paper.

Finally, this paper has dealt with qualitative games (i.e., a player either wins or loses). The exact nature of the theory of quantitative first-cycle games over general cycle properties $Y$ is still to be explored. To what extent do our techniques generalise to quantitative games? or to stochastic ones?

## Acknowledgments

We thank Erich Grädel for stimulating feedback which led to an improvement of the recipe in Section 8. We thank the referees for their useful suggestions and careful reading. Benjamin Aminof was supported by the Austrian National Research Network S11403-N23 (RiSE) of the Austrian Science Fund (FWF) and by the Vienna Science and Technology Fund (WWTF) through grant ICT12-059.

[11] This example was kindly brought to our attention by Erich Grädel, and it prompted us to enhance the recipe that was published in the preliminary work [1] to include Step 3($b$).

[12] Recall Remark 1 that states that all the results in this paper also apply to vertex labelling.

## References

[1] B. Aminof, S. Rubin, First cycle games, in: Proceedings 2nd International Workshop on Strategic Reasoning, SR 2014, in: EPTCS, vol. 146, 2014, pp. 83–90.
[2] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, Dag-width and parity games, in: B. Durand, W. Thomas (Eds.), Symposium on Theoretical Aspects of Computer Science, STACS 2006, in: Lecture Notes in Computer Science, vol. 3884, Springer, 2006, pp. 524–536.
[3] A. Bianco, M. Faella, F. Mogavero, A. Murano, Exploring the boundary of half-positionality, Ann. Math. Artif. Intell. 62 (1–2) (2011) 55–77.
[4] H. Björklund, S. Sandberg, S.G. Vorobyov, Memoryless determinacy of parity and mean payoff games: a simple proof, Theor. Comput. Sci. 310 (1–3) (2004) 365–378.
[5] K. Chatterjee, L. Doyen, Energy parity games, Theor. Comput. Sci. 458 (2012) 49–60.
[6] A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, Int. J. Game Theory 8 (2) (1979) 109–113.
[7] D. Gale, F.M. Stewart, Infinite games with perfect information, in: H. Kuhn, A. Tucker (Eds.), Contributions to the Theory of Games, Volume II, in: Annals of Mathematics Studies, vol. AM-28, Princeton University Press, 1953, pp. 245–266.
[8] H. Gimbert, W. Zielonka, Games where you can play optimally without any memory, in: M. Abadi, L. de Alfaro (Eds.), International Conference on Concurrency Theory, CONCUR 2005, in: Lecture Notes in Computer Science, vol. 3653, 2005, pp. 428–442.
[9] A. Grinshpun, P. Phalitnonkiat, S. Rubin, A. Tarfulea, Alternating traps in Muller and parity games, Theor. Comput. Sci. 521 (2014) 73–91.
[10] E. Kopczynski, Half-positional determinacy of infinite games, in: International Colloquium on Automata, Languages and Programming, ICALP 2006, 2006, pp. 336–347.
[11] M. Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.
[12] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theor. Comput. Sci. 158 (1&2) (1996) 343–359.

# Imperfect-Information Games and Generalized Planning

**Giuseppe De Giacomo**
SAPIENZA Università di Roma
Rome, Italy
degiacomo@dis.uniroma1.it

**Aniello Murano, Sasha Rubin, Antonio Di Stasio**
Università degli Studi di Napoli "Federico II"
Naples, Italy
first.last@unina.it

## Abstract

We study a generalized form of planning under partial observability, in which we have multiple, possibly infinitely many, planning domains with the same actions and observations, and goals expressed over observations, which are possibly temporally extended. By building on work on two-player (non-probabilistic) games with imperfect information in the Formal Methods literature, we devise a general technique, generalizing the belief-state construction, to remove partial observability. This reduces the planning problem to a game of perfect information with a tight correspondence between plans and strategies. Then we instantiate the technique and solve some generalized-planning problems.

## 1 Introduction

Automated planning is a fundamental problem in Artificial Intelligence. Given a deterministic dynamic system with a single known initial state and a goal condition, automated planning consists of finding a sequences of actions (the plan) to be performed by agents in order to achieve the goal [M. Ghallab and Traverso, 2008]. The application of this notion in real-dynamic worlds is limited, in many situations, by three facts: i) the number of objects is neither small nor predetermined, ii) the agent is limited by its observations, iii) the agent wants to realize a goal that extends over time. For example, a preprogrammed driverless car cannot know in advance the number of obstacles it will enter in a road, or the positions of the other cars not in its view, though it wants to realize, among other goals, that every time it sees an obstacle it avoids it. This has inspired research in recent years on *generalized forms of planning* including conditional planning in partially observable domains [Levesque, 1996; Rintanen, 2004], planning with incomplete information for temporally extended goals [De Giacomo and Vardi, 1999; Bertoli and Pistore, 2004] and generalized planning for multiple domains or infinite domains [Levesque, 2005; Srivastava *et al.*, 2008; Bonet *et al.*, 2009; Hu and Levesque, 2010; Hu and De Giacomo, 2011; Srivastava *et al.*, 2011; Felli *et al.*, 2012; Srivastava *et al.*, 2015].

We use the following running example, taken from [Hu and Levesque, 2010], to illustrate a generalized form of planning:

*Example* 1 (Tree Chopping). The goal is to chop down a tree, and store the axe. The number of chops needed to fell the tree is unknown, but a look-action checks whether the tree is up or down. Intuitively, a plan solving this problem alternates looking and chopping until the tree is seen to be down, and then stores the axe. This scenario can be formalized as a partially-observable planning problem on a single infinite domain (Example 2, Figure 1), or on the disjoint union of infinitely many finite domains (Section 3).

The standard approach to solve planning under partial observability for *finite* domains is to reduce them to planning under complete observability. This is done by using the *belief-state construction* that removes partial observability and passes to the *belief-space* [Goldman and Boddy, 1996; Bertoli *et al.*, 2006]. The motivating problem of this work is to solve planning problems on infinite domains, and thus we are naturally lead to the problem of removing partial-observability from infinite domains.

In this paper we adopt the Formal Methods point of view and consider generalized planning as a game $\mathbf{G}$ of imperfect information, i.e., where the player under control has partial observability. The game $\mathbf{G}$ may be infinite, i.e., have infinitely many states.

Our technical contribution (Theorem 4.5) is a general sound and complete mathematical technique for removing imperfect information from a possibly infinite game $\mathbf{G}$ to get a game $\mathbf{G}^\beta$, possibly infinite, of perfect information. Our method builds on the classic belief-state construction [Reif, 1984; Goldman and Boddy, 1996; Raskin *et al.*, 2007], also adopted in POMDPs [Kaelbling *et al.*, 1998; LaValle, 2006].[1] The classic belief-state construction fails for certain infinite games, see Example 3. We introduce a new component to the classic belief-state construction that isolates only those plays in the belief-space that correspond to plays in $\mathbf{G}$. This new component is necessary and sufficient to solve the general case and capture all infinite games $\mathbf{G}$.

We apply our technique to the decision problem that asks, given a game of imperfect information, if the player under control has a winning strategy (this corresponds to deciding if there is a plan for a given planning instance). We remark that we consider strategies and plans that may de-

[1] However, our work considers nondeterminism rather than probability, and qualitative objectives rather than quantitative objectives.

pend on the history of the observations, not just the last observation. Besides showing how to solve the running Tree Chopping example, we report two cases. The first case is planning under partial observability for temporally extended goals expressed in LTL in finite domains (or a finite set of infinite domains sharing the same observations). This case generalizes well-known results in the AI literature [De Giacomo and Vardi, 1999; Rintanen, 2004; Bertoli *et al.*, 2006; Hu and De Giacomo, 2011; Felli *et al.*, 2012]. The second case involves infinite domains. Note that because game solving is undecidable for computable infinite games (simply code the configuration space of a Turing Machine), solving games with infinite domains requires further computability assumptions. We focus on games generated by pushdown automata; these are infinite games that recently attracted the interest of the AI community [Murano and Perelli, 2015; Chen *et al.*, 2016]. In particular these games have been solved assuming perfect information. By applying our technique, we extend their results to deal with imperfect information under the assumption that the stack remains observable (it is known that making the stack unobservable leads to undecidability [Azhar *et al.*, 2001]).

## 2 Generalized-Planning Games

In this section we define generalized-planning (GP) games, known as games of imperfect information in the Formal Methods literature [Raskin *et al.*, 2007], that capture many generalized forms of planning.

Informally, two players (agent and environment) play on a transition-system. Play proceeds in rounds. In each round, from the current state $s$ of the transition-system, the agent observes $obs(s)$ (some information about the current state), and picks an action $a$ from the set of actions $\mathsf{Ac}$, and then the environment picks an element of $\mathsf{tr}(s,a)$ ($\mathsf{tr}$ is the transition function of the transition-system) to become the new current state. Note that the players are asymmetric, i.e., the agent picks actions and the environment resolves non-determinism.

*Notation.* Write $X^\omega$ for the set of infinite sequences whose elements are from the set $X$, write $X^*$ for the finite sequences, and $X^+$ for the finite non-empty sequences. If $\pi$ is a finite sequence then $Last(\pi)$ denotes its last element. The positive integers are denoted $\mathbb{N}$, and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

**Linear-temporal logic**. We define $\mathsf{LTL}$ over a finite set of letters $\Sigma$.[2] The *formulas of* $\mathsf{LTL}$ *(over* $\Sigma$*)* are generated by the following grammar: $\varphi ::= x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi$ where $x \in \Sigma$. We introduce the usual abbreviations for, e.g., $\vee, \mathsf{F}$. Formulas of $\mathsf{LTL}$ (over $\Sigma$) are interpreted over infinite words $\alpha \in \Sigma^\omega$. Define the satisfaction relation $\models$ as follows: i) $(\alpha, n) \models x$ iff $\alpha_n = x$; ii) $(\alpha, n) \models \varphi_1 \wedge \varphi_2$ iff $(\alpha, n) \models \varphi_i$ for $i = 1, 2$; iii) $(\alpha, n) \models \neg\varphi$ iff it is not the case that $(\alpha, n) \models \varphi$; iv) $(\alpha, n) \models \mathsf{X}\varphi$ iff $(\alpha, n+1) \models \varphi$; v) $(\alpha, n) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists $i \geq n$ such that $(\alpha, i) \models \varphi_2$ and for all $j \in [n, i)$, $(\alpha, j) \models \varphi_1$. Write $\alpha \models \varphi$ if $(\alpha, 0) \models \varphi$ and say that $\alpha$ *satisfies* the $\mathsf{LTL}$ formula $\varphi$.

[2] This is without loss of generality, since if $\mathsf{LTL}$ were defined over a set of atomic propositions AP we let $\Sigma = 2^{\mathrm{AP}}$ and replace atoms $p \in \mathrm{AP}$ by $\bigvee_{p \in x} x$ to get equivalent $\mathsf{LTL}$ formulas over $\Sigma$.

**Arenas**. An *arena of imperfect information*, or simply an *arena*, is a tuple $\mathbf{A} = (\mathsf{S}, I, \mathsf{Ac}, \mathsf{tr}, \mathsf{Obs}, obs)$, where $\mathsf{S}$ is a (possibly infinite) set of *states*, $I \subseteq \mathsf{S}$ is the set of *initial states*, $\mathsf{Ac}$ is a finite set of *actions*, and $\mathsf{tr} : \mathsf{S} \times \mathsf{Ac} \to 2^{\mathsf{S}} \setminus \{\emptyset\}$ is the *transition function*, $\mathsf{Obs}$ is a (possibly infinite) set of *observations*, and $obs : \mathsf{S} \to \mathsf{Obs}$, the *observation function*, maps each state to an observation. We extend $\mathsf{tr}$ to sets of states: for $\emptyset \neq Q \subseteq \mathsf{S}$, let $\mathsf{tr}(Q, a)$ denote the set $\cup_{q \in Q} \mathsf{tr}(q, a)$.

Sets of the form $obs^{-1}(x)$ for $x \in \mathsf{Obs}$ are called *observation sets*. The set of all observation sets is denoted $\mathsf{ObsSet}$. Non-empty subsets of observation sets are called *belief-states*. Informally, a belief-state is a subset of the states of the game that the play could be in after a given finite sequence of observations and actions.

**Finite and finitely-branching**. An arena is *finite* if $\mathsf{S}$ is finite, and *infinite* otherwise. An arena is *finitely-branching* if i) $I$ is finite, and ii) for every $s, a$ the cardinality of $\mathsf{tr}(s, a)$ is finite. Clearly, being finite implies being finitely-branching.

**Strategies**. A *play* in $\mathbf{A}$ is an infinite sequence $\pi = s_0 a_0 s_1 a_1 s_2 a_2 \ldots$ such that $s_0 \in I$ and for all $i \in \mathbb{N}_0$, $s_{i+1} \in \mathsf{tr}(s_i, a_i)$. A *history* $h = s_0 a_0 \ldots s_{n-1} a_{n-1} s_n$ is a finite prefix of a play ending in a state. The set of plays is denoted $\mathsf{Ply}(\mathbf{A})$, and the set of histories is denoted $\mathsf{Hist}(\mathbf{A})$ (we drop $\mathbf{A}$ when it is clear from the context). For a history or play $\pi = s_0 a_0 s_1 a_1 \ldots$ write $obs(\pi)$ for the sequence $obs(s_0) a_0 obs(s_1) a_1 \ldots$. A *strategy* (for the agent) is a function $\sigma : \mathsf{Hist}(\mathbf{A}) \to \mathsf{Ac}$. A strategy is *observational* if $obs(h) = obs(h')$ implies $\sigma(h) = \sigma(h')$. In Section 3 we will briefly mention an alternative (but essentially equivalent) definition of observational strategy, i.e., as a function $\mathsf{Obs}^+ \to \mathsf{Ac}$. We do not define strategies for the environment. A play $\pi = s_0 a_0 s_1 a_1 \ldots$ is *consistent* with a strategy $\sigma$ if for all $i \in \mathbb{N}$ we have that if $h \in \mathsf{Hist}(\mathbf{A})$ is a prefix of $\pi$, say $h = s_0 a_0 \ldots s_{n-1} a_{n-1} s_n$, then $\sigma(h) = a_{n+1}$.

**GP Games**. A *generalized-planning (GP) game*, is a tuple $\mathbf{G} = \langle \mathbf{A}, W \rangle$ where the *winning objective* $W \subseteq \mathsf{Obs}^\omega$ is a set of infinite sequences of observation sets. A *GP game with restriction* is a tuple $\mathbf{G} = \langle \mathbf{A}, W, F \rangle$ where, in addition, $F \subseteq \mathsf{S}^\omega$ is the *restriction*. Note that unlike the winning objective, the restriction need not be closed under observations. A GP game is *finite* (resp. *finitely branching*) if the arena $\mathbf{A}$ is finite (resp. finitely branching).

**Winning**. A strategy $\sigma$ is *winning* in $\mathbf{G} = \langle \mathbf{A}, W \rangle$ if for every play $\pi \in \mathsf{Ply}(\mathbf{A})$ consistent with $\sigma$, we have that $obs(\pi) \in W$. Similarly, a strategy is winning in $\mathbf{G} = \langle \mathbf{A}, W, F \rangle$ if for every play $\pi \in \mathsf{Ply}(\mathbf{A})$ consistent with $\sigma$, if $\pi \in F$ then $obs(\pi) \in W$. Note that a strategy is winning in $\langle \mathbf{A}, W, \mathsf{Ply}(\mathbf{A}) \rangle$ if and only if it is winning in $\langle \mathbf{A}, W \rangle$.

**Solving a GP game**. A central decision problem is the following, called *solving a GP game*: given a (finite representation of a) GP game of imperfect information $\mathbf{G}$, decide if the agent has a winning observational-strategy.

**GP games of perfect information**. An arena/GP game has *perfect information* if $\mathsf{Obs} = \mathsf{S}$ and $obs(s) = s$ for all $s$. We thus suppress mentioning $\mathsf{Obs}$ and $obs$ completely, e.g., we write $\mathbf{A} = (\mathsf{S}, I, \mathsf{Ac}, \mathsf{tr})$ and $W, F \subseteq \mathsf{S}^\omega$. Note that in a GP game of perfect information every strategy is observational.

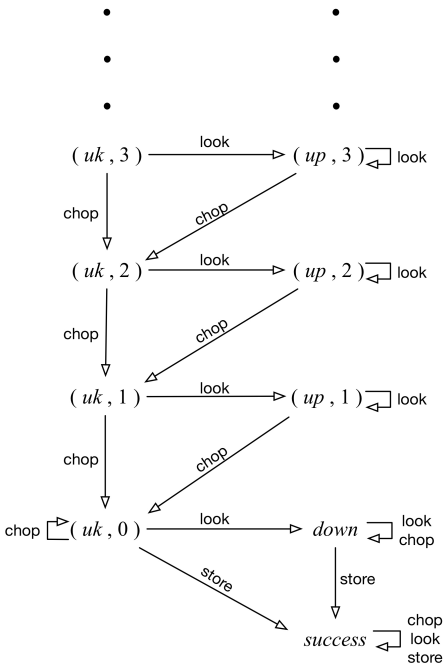



Figure 1: Part of the arena $\mathbf{A}_{\mathsf{chop}}$ (missing edges go to the $failure$ state). The numbers correspond to the number of chops required to fell the tree. The arena is not finitely-branching since it has infinitely many initial states $\{uk\}\times\mathbb{N}_0$.

*Example* 2 (continued). We formalize the tree-chopping planning problem. Define the GP game $\mathbf{G}_{\mathsf{chop}} = \langle \mathbf{A}_{\mathsf{chop}}, W\rangle$ where $\mathbf{A}_{\mathsf{chop}} = \langle \mathsf{S}, I, \mathsf{Ac}, \mathsf{tr}, \mathsf{Obs}, obs\rangle$, and:

- $\mathsf{S} = \{down, success, failure\}\cup(\{uk\}\times\mathbb{N}_0)\cup(\{up\}\times\mathbb{N})$,
- $\mathsf{Ac} = \{\mathsf{chop}, \mathsf{look}, \mathsf{store}\}$, $I = \{uk\}\times\mathbb{N}$,
- $\mathsf{tr}$ is illustrated in Figure 1,
- $\mathsf{Obs} = \{\mathsf{DN}, \checkmark, \times, \mathsf{UK}, \mathsf{UP}\}$,
- $obs$ maps $down \mapsto \mathsf{DN}$, $(up, i) \mapsto \mathsf{UP}$ for $i \in \mathbb{N}$, $(uk, i) \mapsto \mathsf{UK}$ for $i \in \mathbb{N}_0$, $failure \mapsto \times$, and $success \mapsto \checkmark$, and
- the objective $W$ is defined as $\alpha \in W$ iff $\alpha \models \mathsf{F}\checkmark$.

The mentioned plan is formalized as the observational-strategy $\sigma_{\mathsf{chop}}$ that maps any history ending in $(uk, i)$ to look (for $i \in \mathbb{N}_0$), $(up, i)$ to chop (for $i \in \mathbb{N}$), $down$ to store, and all others arbitrarily (say to store).

Note: $\sigma_{\mathsf{chop}}$ is a winning strategy, i.e., no matter which initial state the environment chooses, the strategy ensures that the play (it is unique because the rest of the GP game is deterministic) reaches the state $success$ having observation $\checkmark$.

## 3 Generalized-Planning Games and Generalized Forms of Planning

In this section we establish that generalized-planning (GP) games can model many different types of planning from the AI literature, including a variety of generalized forms of planning:

1. planning on finite transition-systems, deterministic actions, actions with conditional effects, partially observable states, incomplete information on the initial state, and temporally extended goals [De Giacomo and Vardi, 1999];
2. planning under partial observability with finitely many state variables, nondeterministic actions, reachability goals, and partial observability [Rintanen, 2004];
3. planning on finite transition systems, nondeterministic actions, looking for strong plans (i.e., adversarial nondeterminism) [Bertoli *et al.*, 2006];
4. generalized planning, consisting of multiple (possibly infinitely many) related finite planning instances [Hu and Levesque, 2010; Hu and De Giacomo, 2011].

We discuss the latter in detail. Following [Hu and De Giacomo, 2011], a *generalized-planning problem* $\mathfrak{P}$ is defined as a sequence of related classical planning problems. In our terminology, fix finite sets $\mathsf{Ac}, \mathsf{Obs}$ and let $\mathfrak{P}$ be a countable sequence $\mathbf{G}_1, \mathbf{G}_2, \ldots$ where each $\mathbf{G}_n$ is a finite GP game of the form $\langle \mathsf{S}_n, \{\iota_n\}, \mathsf{Ac}, \mathsf{tr}_n, \mathsf{Obs}, obs_n, W_n\rangle$. In [Hu and De Giacomo, 2011], a plan is an observational-strategy $p : \mathsf{Obs}^+ \to \mathsf{Ac}$, and a solution is a single plan that solves all of the GP games in the sequence. Now, we view $\mathfrak{P}$ as a single infinite GP game as follows. Let $\mathbf{G}_{\mathfrak{P}}$ denote the *disjoint* union of the GP games in $\mathfrak{P}$. Formally, $\mathbf{G}_{\mathfrak{P}} = \langle \mathsf{S}, I, \mathsf{Ac}, \mathsf{tr}, \mathsf{Obs}, obs, W\rangle$ where

- $\mathsf{S} = \{(s, n) : s \in \mathsf{S}_n, n \in \mathbb{N}\}$,
- $I = \{(\iota_n, n) : n \in \mathbb{N}\}$,
- $\mathsf{tr}((s, n), a) = \{(t, n) : t \in \mathsf{tr}_n(s, a)\}$,
- $obs(s, n) = obs_n(s)$,
- $W = \cup_n W_n$.

Then: there is a correspondence between solutions for $\mathfrak{P}$ and winning observational-strategies in $\mathbf{G}_{\mathfrak{P}}$.

For example, consider the tree-chopping problem as formalized in [Hu and Levesque, 2010; Hu and De Giacomo, 2011]: there are infinitely many planning instances which are identical except for an integer parameter denoting the number of chops required to fell the tree. The objective for all instances is to fell the tree. Using the translation above we get a GP game with an infinite arena which resembles (and, in fact, can be transformed to) the GP game in Example 2.

## 4 Generalized Belief-State Construction

In this section we show how to remove imperfect information from generalized-planning (GP) games $\mathbf{G}$. That is, we give a transformation of GP games of imperfect information $\mathbf{G}$ to GP games of perfect information $\mathbf{G}^\beta$ such that the agent has a winning observational-strategy in $\mathbf{G}$ if and only if the agent has a winning strategy in $\mathbf{G}^\beta$. The translation is based on the classic belief-state construction [Reif, 1984; Raskin *et al.*, 2007]. Thus, we begin with a recap of that construction.

**Belief-state Arena**.[3] Let $\mathbf{A} = (\mathsf{S}, I, \mathsf{Ac}, \mathsf{tr}, \mathsf{Obs}, obs)$ be an arena (not necessarily finite). Recall from Section 2 that *observation sets* are of the form $obs^{-1}(x)$ for $x \in \mathsf{Obs}$, and are collectively denoted $\mathsf{ObsSet}$. Define the arena of perfect information $\mathbf{A}^\beta = (\mathsf{S}^\beta, I^\beta, \mathsf{Ac}, \mathsf{tr}^\beta)$ where,

- $\mathsf{S}^\beta$ is the set of belief-states, i.e., the non-empty subsets of the observation-sets,
- $I^\beta$ consists of all belief-states of the form $I \cap X$ for $X \in \mathsf{ObsSet}$,
- $\mathsf{tr}^\beta(Q, a)$ consists of all belief-states of the form $\mathsf{tr}(Q, a) \cap X$ for $X \in \mathsf{ObsSet}$.

The idea is that $Q \in \mathsf{S}^\beta$ represents a refinement of the observation set: the agent, knowing the structure of $\mathbf{G}$ ahead of time, and the sequence of observations so far in the game, may deduce that it is in fact in a state from $Q$ which may be a strict subset of its corresponding observation set $X$.[4]

**NB**. Since $\mathbf{A}^\beta$ is an arena, we can talk of its histories and plays. Although we defined $\mathsf{S}^\beta$ to be the set of all belief-states, only those belief-states that are reachable from $I^\beta$ are relevant. Thus, overload notation and write $\mathsf{S}^\beta$ for the set of reachable belief-states, and $\mathbf{A}^\beta$ for the corresponding arena. This notation has practical relevance since if $\mathbf{A}$ is countable there are uncountably many belief-states; but in many cases only countably many (or, as in the running example, finitely many) reachable belief-states.

The intuition for the rest of this section is illustrated in the next example.

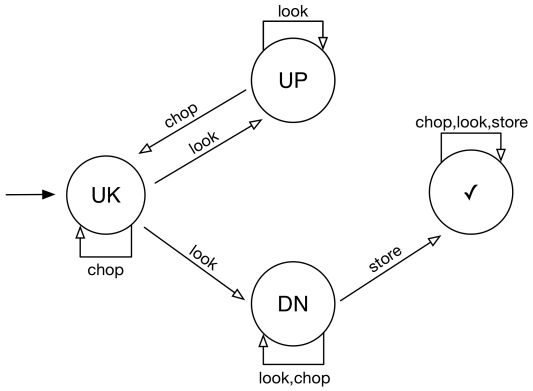



Figure 2: Part of the arena $\mathbf{A}^\beta_{\mathsf{chop}}$ (missing edges go the the $failure$ state). Each circle is a belief-state. The winning objective is $\mathsf{F}\checkmark$, and the restriction is $\neg\,\mathsf{G}\,\mathsf{F}[\mathsf{UK} \wedge \mathsf{X}\,\mathsf{look} \wedge \mathsf{X}\,\mathsf{X}\,\mathsf{UP}]$.

*Example* 3 (continued). Figure 2 shows the arena $\mathbf{A}^\beta_{\mathsf{chop}}$ corresponding to the arena from tree-chopping game $\mathbf{G}_{\mathsf{chop}}$, i.e.,

- $\mathsf{S}^\beta$ are the following belief-states: $\{(uk, n) \mid n \in \mathbb{N}_0\}$, denoted UK; $\{(up, n) \mid n \in \mathbb{N}\}$, denoted UP; $\{down\}$, denoted DN; $\{success\}$, denoted ✓; and $\{failure\}$.
- $I^\beta$ is the belief-state UK,
- and $\mathsf{tr}^\beta$ is shown in the figure.

Note that the agent does not have a winning strategy in the GP game with arena $\mathbf{A}^\beta_{\mathsf{chop}}$ and winning condition $\mathsf{F}\checkmark$. The informal reason is that the strategy $\sigma_{\mathsf{chop}}$ (which codifies "alternately look and chop until the tree is sensed to be down, and then store the axe"), which wins in $\mathbf{G}$, does not work. The reason is that after every look the opponent can transition to UP (and never DN), resulting in the play $\rho = (\mathsf{UK}\ \mathsf{look}\ \mathsf{UP}\ \mathsf{chop})^\omega$, i.e., the repetition of $(\mathsf{UK}\ \mathsf{look}\ \mathsf{UP}\ \mathsf{chop})$ forever. Such a play of $\mathbf{A}^\beta_{\mathsf{chop}}$ does not correspond to any play in $\mathbf{A}_{\mathsf{chop}}$. This is a well known phenomena of the standard belief-set construction [Sardiña *et al.*, 2006], which our construction overcomes by adding a restriction that removes from consideration plays such as $\rho$ (as discussed in Example 5).

The following definition is central. It maps a history $h \in \mathsf{Hist}(\mathsf{A})$ to the corresponding history $h^\beta \in \mathsf{Hist}(\mathbf{A}^\beta)$ of belief-states.

**Definition 4.1.** For $h \in \mathsf{Hist}(\mathbf{A})$ define $h^\beta \in \mathsf{Hist}(\mathbf{A}^\beta)$ inductively as follows.

- For $s \in I$, define $s^\beta \in I^\beta$ to be $I \cap obs^{-1}(obs(s))$. In words, $s^\beta$ is the set of initial states the GP game could be in given the observation $obs(s)$.
- If $h \in \mathsf{Hist}(\mathbf{A}), a \in \mathsf{Ac}, s \in \mathsf{S}$, then $(has)^\beta := h^\beta aB$ where $B := \mathsf{tr}(Last(h^\beta), a) \cap obs^{-1}(obs(s))$. In words, $B$ is the set of possible states the GP game could be in given the observation sequence $obs(has)$.

In the same way, for $\pi \in \mathsf{Ply}(\mathsf{A})$ define $\pi^\beta \in \mathsf{Ply}(\mathsf{A}^\beta)$. Extend the map pointwise to sets of plays $P \subseteq \mathsf{Ply}(\mathsf{A})$, i.e., define $P^\beta := \{\pi^\beta \in \mathsf{Ply}(\mathsf{A}^\beta) \mid \pi \in P\}$. Finally, we give notation to the special case that $P = \mathsf{Ply}(\mathsf{A})$: write $\mathsf{Im}(\mathbf{A})$ for the set $\{\pi^\beta \mid \pi \in \mathsf{Ply}(\mathsf{A})\}$, called the *image of* $\mathbf{A}$.

By definition, $\mathsf{Im}(\mathbf{A}) \subseteq \mathsf{Ply}(\mathbf{A}^\beta)$. However, the converse is not always true.

*Example* 4 (continued). There is a play of $\mathbf{A}^\beta_{\mathsf{chop}}$ that is not in $\mathsf{Im}(\mathbf{A}_{\mathsf{chop}})$, e.g., $\rho = (uk\ \mathsf{look}\ up\ \mathsf{chop})^\omega$. Indeed, suppose $\pi^\beta = \rho$ and consider the sequence of counter values of $\pi$. Every look action establishes that the current counter value in $\pi$ is positive (this is the meaning of the tree being $up$), but every chop action reduces the current counter value by one. This contradicts that counter values are always non-negative.

*Remark* 4.2. If $\mathbf{A}$ is finitely-branching then $\mathsf{Im}(\mathbf{A}) = \mathsf{Ply}(\mathsf{A}^\beta)$. To see this, let $\rho$ be a play in $\mathbf{A}^\beta$, and consider the forest whose nodes are the histories $h$ of $\mathbf{A}$ such that $h^\beta$ is a prefix of $\rho$. Each tree in the forest is finitely branching (because $\mathbf{A}$ is), and at least one tree in this forest is infinite. Thus, by Kőnig's lemma, the tree has an infinite path $\pi$. But $\pi$ is a play in $\mathbf{A}$ and $\pi^\beta = \rho$.

**Definition 4.3.** For $\rho \in \mathsf{Ply}(\mathbf{A}^\beta)$, say $\rho = B_0 a_0 B_1 a_1 \ldots$, define $obs(\rho)$ to be the sequence $obs(q_0)a_0 obs(q_1)a_1 \ldots$

[3] In the AI literature, this is sometimes called the *belief-space*.

[4] To illustrate simply, suppose there is a unique initial state $s$, and that it is observationally equivalent to other states. At the beginning of the game the agent can deduce that it must be in $s$. Thus, its initial belief-state is $\{s\}$ and not its observation-set $obs^{-1}(obs(s))$. This belief can (and, in general, must) be exploited if the agent is to win.

where $q_i \in B_i$ for $i \in \mathbb{N}_0$ (this is well defined since, by definition of the state set $\mathsf{S}^\beta$, each $B_i$ is a subset of a unique observation-set).

The classic belief-state construction transforms $\langle \mathbf{A}, W\rangle$ into $\langle \mathbf{A}^\beta, W\rangle$. Example 3 shows that this transformation may not preserve the agent having a winning strategy if $\mathbf{A}$ is infinite. We now define the generalized belief-state construction and the main technical theorem of this work.

**Definition 4.4.** Let $\mathbf{G} = \langle \mathbf{A}, W\rangle$ be a GP game. Define $\mathbf{G}^\beta = \langle \mathbf{A}^\beta, W, \mathsf{Im}(\mathbf{A})\rangle$, a GP game of perfect information with restriction. The restriction $\mathsf{Im}(\mathbf{A}) \subseteq \mathsf{Ply}(\mathbf{A}^\beta)$ is the image of $\mathsf{Ply}(\mathbf{A})$ under the map $\pi \mapsto \pi^\beta$.

**Theorem 4.5.** *Let $\mathbf{A}$ be a (possibly infinite) arena of imperfect information, $\mathbf{A}^\beta$ the corresponding belief-state arena of perfect information, and $\mathsf{Im}(\mathbf{A}) \subseteq \mathsf{Ply}(\mathbf{A}^\beta)$ the image of $\mathbf{A}$. Then, for every winning objective $W$, the agent has a winning observational-strategy in the GP game $\mathbf{G} = \langle \mathbf{A}, W\rangle$ if and only if the agent has a winning strategy in the GP game $\mathbf{G}^\beta = \langle \mathbf{A}^\beta, W, \mathsf{Im}(\mathbf{A})\rangle$. Moreover, if $\mathbf{A}$ is finitely-branching then $\mathbf{G}^\beta = \langle \mathbf{A}^\beta, W\rangle$.*[5]

*Proof.* The second statement follows from the first statement and Remark 4.2. For the first statement, we first need some facts that immediately follow from Definition 4.1.

1. $(h_1)^\beta = (h_2)^\beta$ if and only if $obs(h_1) = obs(h_2)$.
2. For every $h \in \mathsf{Hist}(\mathbf{A}^\beta)$ that is also a prefix of $\pi^\beta$ there is a history $h' \in \mathsf{Hist}(\mathbf{A})$ that is also a prefix of $\pi$ such that $(h')^\beta = h$. Also, for every $h' \in \mathsf{Hist}(\mathbf{A})$ that is also a prefix of $\pi$ there is a history $h \in \mathsf{Hist}(\mathbf{A}^\beta)$ that is also a prefix of $\pi^\beta$ such that $(h')^\beta = h$.

Second, there is a natural correspondence between observational strategies of $\mathbf{A}$ and strategies of $\mathbf{A}^\beta$.

- If $\sigma$ is a strategy in $\mathbf{A}^\beta$ then define the strategy $\omega(\sigma)$ of $\mathbf{A}$ as mapping $h \in \mathsf{Hist}(\mathbf{A})$ to $\sigma(h^\beta)$. Now, $\omega(\sigma)$ is observational by Fact 1. Also, if $\pi$ is consistent with $\omega(\sigma)$ then $\pi^\beta$ is consistent with $\sigma$. Indeed, let $h$ be a history that is also a prefix of $\pi^\beta$. We need to show that $h\sigma(h)$ is a prefix of $\pi^\beta$. Suppose that $\sigma(h) = a$. Take prefix $h'$ of $\pi$ such that $(h')^\beta = h$ (Fact 2). Then $\omega(\sigma)(h') = \sigma((h')^\beta) = \sigma(h) = a$. Since $\pi$ is assumed consistent with $\omega(\sigma)$, conclude that $h'a$ is a prefix of $\pi$. Thus $ha$ is a prefix of $\pi^\beta$.
- If $\sigma$ is an observational strategy in $\mathbf{A}$ then define the strategy $\kappa(\sigma)$ of $\mathbf{A}^\beta$ as mapping $h \in \mathsf{Hist}(\mathbf{A}^\beta)$ to $\sigma(h')$ where $h'$ is any history such that $h'^\beta = h$. This is well-defined by (†) and the fact that $\sigma$ is observational. Also, if $\rho$ is consistent with $\kappa(\sigma)$, then every $\pi$ with $\pi^\beta = \rho$ (if there are any) is consistent with $\sigma$. Indeed, let $h'$ be a history of $\pi$ and take a prefix $h$ of $\pi^\beta$ such that $(h')^\beta = h$ (Fact 2). Then $\kappa(\sigma)(h) = \sigma(h')$, call this action $a$. But $\pi^\beta$ is assumed consistent with $\kappa(\sigma)$, and thus $ha$ is a prefix of $\pi^\beta$. Thus $h'a$ is a prefix of $\pi$.

We now put everything together and show that the agent has a winning observational-strategy in $\mathbf{G}$ iff the agent has a winning strategy in $\mathbf{G}^\beta$.

[5]The case that $\mathbf{A}$ is finite appears in [Raskin *et al.*, 2007].

Suppose $\sigma$ is a winning strategy in $\mathbf{G}^\beta$. Let $\pi \in \mathsf{Ply}(\mathbf{A})$ be consistent with the observational strategy $\omega(\sigma)$ of $\mathbf{G}$. Then $\pi^\beta \in \mathsf{Im}(\mathbf{A})$ is consistent with $\sigma$. But $\sigma$ is assumed to be winning, thus $obs(\pi^\beta) \in W$. But $obs(\pi) = obs(\pi^\beta)$. Conclude that $\omega(\sigma)$ is a winning strategy in $\mathbf{G}$.

Conversely, suppose $\sigma$ is a winning strategy in $\mathbf{G}$. Let $\rho \in \mathsf{Im}(\mathbf{A})$ be consistent with the strategy $\kappa(\sigma)$ of $\mathbf{G}^\beta$, and take $\pi \in \mathsf{Ply}(\mathsf{A})$ be such that $\pi^\beta = \rho$ (such a $\pi$ exists since we assumed $\rho \in \mathsf{Im}(\mathbf{A})$). Then $\pi$ is consistent with $\sigma$. But $\sigma$ is assumed to be winning, thus $obs(\pi) \in W$. But $obs(\rho) = obs(\pi^\beta) = obs(\pi)$. Conclude that $\kappa(\sigma)$ is a winning strategy in $\mathbf{G}^\beta$. □

*Remark* 4.6. The proof of Theorem 4.5 actually shows how to transform strategies between the GP games, i.e., $\sigma \mapsto \omega(\sigma)$ and $\sigma \mapsto \kappa(\sigma)$, and moreover, these transformations are inverses of each other.

We end with our running example:

*Example* 5 (Continued). Solving $\mathbf{G}_{\mathsf{chop}}$ (Figure 1) is equivalent to solving the finite GP game $\mathbf{G}^\beta_{\mathsf{chop}}$ of perfect information, i.e., $\langle \mathbf{A}^\beta_{\mathsf{chop}}, W, \mathsf{Im}(\mathbf{A}_{\mathsf{chop}})\rangle$, where the arena $\mathbf{A}$ is shown in Figure 2. To solve this we should understand the structure of $\mathsf{Im}(\mathbf{A}_{\mathsf{chop}})$. It is not hard to see that a play $\rho \in \mathsf{Ply}(\mathbf{A}^\beta_{\mathsf{chop}})$ is in $\mathsf{Im}(\mathbf{A}_{\mathsf{chop}})$ if and only if it contains only finitely many infixes of the form "UK look UP". This property is expressible in LTL by the formula $\neg\,\mathsf{G}\,\mathsf{F}[\mathsf{UK} \wedge \mathsf{X}\,\mathsf{look} \wedge \mathsf{X}\,\mathsf{X}\,\mathsf{UP}]$. Thus we can apply the algorithm for solving finite games of perfect information with LTL objectives (see, e.g., [Pnueli and Rosner, 1989; de Alfaro *et al.*, 2001]) to solve $\mathbf{G}^\beta_{\mathsf{chop}}$, and thus the original GP game $\mathbf{G}_{\mathsf{chop}}$.

## 5 Application of the Construction

We now show how to use generalized-planning (GP) games and our generalized belief-state construction to obtain effective planning procedures for sophisticated problems. For the rest of this section we assume Obs is finite ($\mathbf{A}$ may be infinite) so that we can consider LTL temporally extended goals over the alphabet Obs. For instance, LTL formulas specify persistent surveillance missions such as "get items from region A, drop items at region B, infinitely often, and always avoid region C".

**Definition 5.1.** Let $\varphi$ be an LTL formula over $\mathsf{Obs} \times \mathsf{Ac}$. For an arena $\mathbf{A}$, define $[[\varphi]] = \{\pi \in \mathsf{Ply}(\mathbf{A}) \mid obs(\pi) \models \varphi\}$.

The following is immediate from Theorem 4.5 and the fact that solving finite LTL games of perfect information is decidable [Pnueli and Rosner, 1989; de Alfaro *et al.*, 2001]:

**Theorem 5.2.** *Let $\mathbf{G} = \langle \mathbf{A}, [[\varphi]]\rangle$ be a GP game with a finite arena (possibly obtained as the disjoint union of several arenas sharing the same observations), and $\varphi$ be an LTL winning objective. Then solving $\mathbf{G}$ can be reduced to solving the finite GP game $\mathbf{G}^\beta = \langle \mathbf{A}^\beta, [[\varphi]]\rangle$ of perfect information, which is decidable.*

Although we defined winning objectives to be observable, one may prefer general winning conditions, i.e., $W \subseteq S^\omega$. In this case, for finite arenas there is a translation from parity-objectives to observable parity-objectives [Chatterjee

and Doyen, 2010]; moreover, for reachability objectives, a plan reaches a goal $T \subseteq \mathsf{S}$ iff it reaches a belief-state $B \subseteq T$ [Bertoli *et al.*, 2006].

Next we look a case where the arena is actually infinite. Recently, the AI community has considered games generated by pushdown-automata [Murano and Perelli, 2015; Chen *et al.*, 2016]. However, the games considered are of perfect information and cannot express generalized-planning problems or planning under partial observability. In contrast, our techniques can solve these planning problems on pushdown domains assuming that the stack is not hidden (we remark that if the stack is hidden, then game-solving becomes undecidable [Azhar *et al.*, 2001]):

**Theorem 5.3.** *Let $\mathbf{G} = \langle \mathbf{A}, [[\varphi]] \rangle$ be a GP game with a pushdown-arena with observable stack, and $\varphi$ is an* LTL *formula. Then solving $\mathbf{G}$ can be reduced to solving $\mathbf{G}^{\beta} = \langle \mathbf{A}^{\beta}, [[\varphi]] \rangle$, a GP game with pushdown-arena with perfect information, which is decidable.*

*Proof.* Let $P$ be a pushdown-automaton with states $Q$, initial state $q_0$, finite input alphabet $\mathsf{Ac}$, and finite stack alphabet $\Gamma$. We call elements of $\Gamma^*$ stacks, and denote the empty stack by $\epsilon$. Also, fix an observation function on the states, i.e., $f : Q \to \mathsf{Obs}$ for some set $\mathsf{Obs}$ (we do not introduce notation for the transition function of $P$). A pushdown-arena $\mathbf{A}_P = (\mathsf{S}, I, \mathsf{Ac}, \mathsf{tr}, \mathsf{Obs}, obs)$ is generated by $P$ as follows: the set of states $\mathsf{S}$ is the set of configurations of $P$, i.e., pairs $(q, \gamma)$ where $q \in Q$ and $\gamma \in \Gamma^*$ is a stack-content of $P$; the initial state of $\mathbf{A}$ is the initial configuration, i.e., $I = \{(q_0, \epsilon)\}$; the transition function of $\mathbf{A}$ is defined as $\mathsf{tr}((q, \gamma), a) = (q', \gamma')$ if $P$ can move in one step from state $q$ and stack content $\gamma$ to state $q'$ and stack content $\gamma'$ by consuming the input letter $a$; the observation function $obs$ maps a configuration $(q, \gamma)$ to $f(q)$ (i.e., this formalizes the statement that the stack is observable). Observe now that: (1) the GP game $\mathbf{A}$ is finitely-branching; (2) the GP game $\mathbf{A}^{\beta}$ is generated by a pushdown automaton (its states are subsets of $Q$). Thus we can apply Theorem 4.5 to reduce solving $\mathbf{A}$, a GP-game with imperfect information and pushdown arena, to $\mathbf{A}^{\beta}$, a GP-game with perfect information and pushdown arena. The latter is decidable [Walukiewicz, 2001]. □

## 6 Related work in Formal Methods

Games of imperfect information on *finite* arenas have been studied extensively. Reachability winning-objectives were studied in [Reif, 1984] from a complexity point of view: certain games were shown to be universal in the sense that they are the hardest games of imperfect information, and optimal decision procedures were given. More generally, $\omega$-regular winning-objectives were studied in [Raskin *et al.*, 2007], and symbolic algorithms were given (also for the case of randomized strategies).

To solve (imperfect-information) games on infinite arenas one needs a finite-representation of the infinite arena. One canonical way to generate infinite arenas is by parametric means. In this line, [Jacobs and Bloem, 2014] study the synthesis problem for distributed architectures with a parametric number of finite-state components. They leverage results from the Formal Methods literature that say that for certain types of token-passing systems there is a cutoff [Emerson and Namjoshi, 1995], i.e., an upper bound on the number of components one needs to consider in order to synthesize a protocol for any number of components. Another way to generate infinite arenas is as configuration spaces of pushdown automata. These are important in analysis of software because they capture the flow of procedure calls and returns in reactive programs. Module-checking pushdown systems of imperfect information [Bozzelli *et al.*, 2010; Aminof *et al.*, 2013] can be thought of as games in which the environment plays non-deterministic strategies. Although undecidable, by not hiding the stack (cf. Theorem 5.3) decidability of module-checking is regained.

Finally, we note that synthesis of distributed systems has been studied in the Formal Methods literature using the techniques of games, starting with [Pnueli and Rosner, 1990]. Such problems can be cast as multi-player games of imperfect information, and logics such as ATL with knowledge can be used to reason about strategies in these games. However, even for three players, finite arenas, and reachability goals, the synthesis problem (and the corresponding model checking problem for ATLK) is undecidable [Dima and Tiplea, 2011].

## 7 Critical Evaluation and Conclusions

Although our technique for removing partial observability is sound and complete, it is, necessarily, not algorithmic: indeed, no algorithm can always remove partial observability from computable infinite domains and result in a solvable planning problem (e.g, one with a finite domain).[6]

The main avenue for future technical work is to establish natural classes of generalized-planning problems that can be solved algorithmically. We believe the methodology of this paper will be central to this endeavor. Indeed, as we showed in Section 5, we can identify $\mathsf{Im}(\mathbf{A})$ in a number of cases. We conjecture that one can do the same for all of the one-dimensional planning problems of [Hu and Levesque, 2010; Hu and De Giacomo, 2011].

The framework presented in this paper is non-probabilistic, but extending it with probabilities and utilities associated to agent choices [Kaelbling *et al.*, 1998; LaValle, 2006; Bonet and Geffner, 2009; Geffner and Bonet, 2013] is of great interest. In particular, POMDPs with temporally-extended winning objectives (e.g., LTL, Büchi, parity) have been studied for finite domains [Chatterjee *et al.*, 2010]. We leave for future work the problem of dealing with such POMDPs over infinite domains.

## Acknowledgments

We thank the reviewers for their constructive comments. This research was partially supported by the Sapienza project "Immersive Cognitive Environments". Aniello Murano was supported in part by GNCS 2016 project: Logica, Automi e Giochi per Sistemi Auto-adattivi. Sasha Rubin is a Marie Curie fellow of the Istituto Nazionale di Alta Matematica.

[6] In fact, there is no algorithm solving games of perfect observation on computable domains with reachability objectives.

## References


[Aminof *et al.*, 2013] B. Aminof, A. Legay, A. Murano, O. Serre, and M. Y. Vardi. Pushdown module checking with imperfect information. *Inf. Comput.*, 223, 2013.

[Azhar *et al.*, 2001] S. Azhar, G. Peterson, and J. Reif. Lower bounds for multiplayer non-cooperative games of incomplete information. *J. Comp. Math. Appl.*, 41, 2001.

[Bertoli and Pistore, 2004] P. Bertoli and M. Pistore. Planning with extended goals and partial observability. In *Proc. of ICAPS 2004*, 2004.

[Bertoli *et al.*, 2006] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Strong planning under partial observability. *Artif. Intell.*, 170(4-5), 2006.

[Bonet and Geffner, 2009] B. Bonet and H. Geffner. Solving POMDPs: Rtdp-bel vs. point-based algorithms. In *Proc. of IJCAI 2009*, 2009.

[Bonet *et al.*, 2009] B. Bonet, H. Palacios, and H. Geffner. Automatic derivation of memoryless policies and finite-state controllers using classical planners. In *Proc. of ICAPS 2009*, 2009.

[Bozzelli *et al.*, 2010] L. Bozzelli, A. Murano, and A. Peron. Pushdown module checking. *FMSD*, 36(1), 2010.

[Chatterjee and Doyen, 2010] K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *Proc. of LPAR 2010*, 2010.

[Chatterjee *et al.*, 2010] K. Chatterjee, L. Doyen, and T.A. Henzinger. Qualitative analysis of partially-observable markov decision processes. In *Proc. of MFCS*, 2010.

[Chen *et al.*, 2016] T. Chen, F. Song, and Z. Wu. Global model checking on pushdown multi-agent systems. In *Proc. of AAAI 2016*, 2016.

[de Alfaro *et al.*, 2001] L. de Alfaro, T. A. Henzinger, and R. Majumdar. From verification to control: Dynamic programs for omega-regular objectives. In *Proc. of LICS 2001*, 2001.

[De Giacomo and Vardi, 1999] G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *Proc. of ECP 1999*, 1999.

[Dima and Tiplea, 2011] C. Dima and F. L. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoR*, abs/1102.4225, 2011.

[Emerson and Namjoshi, 1995] E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In *Proc. of POPL 1995*, 1995.

[Felli *et al.*, 2012] P. Felli, G. De Giacomo, and A. Lomuscio. Synthesizing agent protocols from LTL specifications against multiple partially-observable environments. In *Proc. of KR 2012*, 2012.

[Geffner and Bonet, 2013] H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool, 2013.

[Goldman and Boddy, 1996] R. P. Goldman and M. S. Boddy. Expressive planning and explicit knowledge. In *Proc. of AIPS 1996*, 1996.

[Hu and De Giacomo, 2011] Y. Hu and G. De Giacomo. Generalized planning: Synthesizing plans that work for multiple environments. In *Proc. of IJCAI 2011*, 2011.

[Hu and Levesque, 2010] Y. Hu and H. J. Levesque. A correctness result for reasoning about one-dimensional planning problems. In *Proc. of KR 2010*, 2010.

[Jacobs and Bloem, 2014] S. Jacobs and R. Bloem. Parameterized synthesis. *LMCS*, 10(1), 2014.

[Kaelbling *et al.*, 1998] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2), 1998.

[LaValle, 2006] S. M. LaValle. *Planning algorithms*. Cambridge, 2006.

[Levesque, 1996] H. J. Levesque. What is planning in the presence of sensing. In *Proc. of AAAI 1996*, 1996.

[Levesque, 2005] H. Levesque. Planning with loops. In *Proc. of IJCAI 2005*, 2005.

[M. Ghallab and Traverso, 2008] D. Nau M. Ghallab and P. Traverso. *Automated Planning: Theory & Practice*. Elsevier, 2008.

[Murano and Perelli, 2015] A. Murano and G. Perelli. Pushdown multi-agent system verification. In *Proc. of IJCAI 2015*, 2015.

[Pnueli and Rosner, 1989] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *Proc. of ICALP 1989*, 1989.

[Pnueli and Rosner, 1990] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. of STOC 1990*, 1990.

[Raskin *et al.*, 2007] J. Raskin, K. Chatterjee, L. Doyen, and T. A. Henzinger. Algorithms for omega-regular games with imperfect information. *LMCS*, 3(3), 2007.

[Reif, 1984] J. H. Reif. The complexity of two-player games of incomplete information. *JCSS*, 29(2), 1984.

[Rintanen, 2004] J. Rintanen. Complexity of planning with partial observability. In *Proc. of ICAPS 2004*, 2004.

[Sardiña *et al.*, 2006] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the limits of planning over belief states under strict uncertainty. In *Proc. of KR 2016*, 2006.

[Srivastava *et al.*, 2008] S. Srivastava, N. Immerman, and S. Zilberstein. Learning generalized plans using abstract counting. In *Proc. of AAAI 2008*, 2008.

[Srivastava *et al.*, 2011] S. Srivastava, N. Immerman, and S. Zilberstein. A new representation and associated algorithms for generalized planning. *Artif. Intell.*, 175(2), 2011.

[Srivastava *et al.*, 2015] S. Srivastava, S. Zilberstein, A. Gupta, P. Abbeel, and S. J. Russell. Tractability of planning with loops. In *Proc. of AAAI 2015*, 2015.

[Walukiewicz, 2001] I. Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2), 2001.

# Parameterized Model Checking of Token-Passing Systems

Benjamin Aminof[1], Swen Jacobs[2], Ayrat Khalimov[2], Sasha Rubin[1,3] ⋆

[1]IST Austria (first.last@ist.ac.at), [2]TU Graz (first.last@iaik.tugraz.at), [3]TU Wien

**Abstract.** We revisit the parameterized model checking problem for token-passing systems and specifications in indexed $\mathsf{CTL}^*\backslash\mathsf{X}$. Emerson and Namjoshi (1995, 2003) have shown that parameterized model checking of indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ in uni-directional token rings can be reduced to checking rings up to some *cutoff* size. Clarke et al. (2004) have shown a similar result for general topologies and indexed $\mathsf{LTL}\backslash\mathsf{X}$, provided processes cannot choose the directions for sending or receiving the token.

We unify and substantially extend these results by systematically exploring fragments of indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ with respect to general topologies. For each fragment we establish whether a cutoff exists, and for some concrete topologies, such as rings, cliques and stars, we infer small cutoffs. Finally, we show that the problem becomes undecidable, and thus no cutoffs exist, if processes are allowed to choose the directions in which they send or from which they receive the token.

## 1 Introduction

As executions of programs and protocols are increasingly distributed over multiple CPU cores or even physically separated computers, correctness of concurrent systems is one of the primary challenges of formal methods today. Many concurrent systems consist of an arbitrary number of identical processes running in parallel. The parameterized model checking problem (PMCP) for concurrent systems is to decide if a given temporal logic specification holds irrespective of the number of participating processes.

The PMCP is undecidable in many cases. For example, it is undecidable already for safety specifications and finite-state processes communicating by passing a binary-valued token around a uni-directional ring [16,7]. However, decidability may be regained by restricting the communication primitives, the topologies under consideration (i.e., the underlying graph describing the communication paths between the processes), or the specification language. In particular, previous results have shown that parameterized model checking can sometimes be reduced to model checking a finite number of instances of the system, up to some *cutoff* size.

⋆ This work was supported by the Austrian Science Fund through grant P23499-N23 and through the RiSE network (S11403, S11405, S11406, S11407-N23); ERC Starting Grant (279307: Graph Games); Vienna Science and Technology Fund (WWTF) grants PROSEED, ICT12-059, and VRG11-005.

For token-passing systems (TPSs) with uni-directional ring topologies, such cutoffs are known for specifications in the prenex fragment of indexed $\mathsf{CTL}^*$ without the next-time operator ($\mathsf{CTL}^*\backslash\mathsf{X}$) [9,7]. For token-passing in general topologies, cutoffs are known for the prenex fragment of indexed $\mathsf{LTL}\backslash\mathsf{X}$, provided that processes are not allowed to choose the direction to which the token is sent or from which it is received [4]. In this paper we generalize these results and elucidate what they have in common.

**Previous Results.** In their seminal paper [7], Emerson and Namjoshi consider systems where the token does not carry messages, and specifications are in prenex indexed temporal logic — i.e., quantifiers $\forall$ and $\exists$ over processes appear in a block at the front of the formula. They use the important concept of a *cutoff* — a number $c$ such that the PMCP for a given class of systems and specifications can be reduced to model checking systems with up to $c$ processes. If model checking is decidable, then existence of a cutoff implies that the PMCP is decidable. Conversely, if the PMCP is undecidable, then there can be no cutoff for such systems.

For uni-directional rings, Emerson and Namjoshi provide cutoffs for formulas with a small number $k$ of quantified index variables, and state that their proof method allows one to obtain cutoffs for other quantifier prefixes. In brief, cutoffs exist for the branching-time specification language prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ and the highly regular topology of uni-directional rings.

Clarke et al. [4] consider the PMCP for token-passing systems arranged in general topologies. Their main result is that the PMCP for systems with arbitrary topologies and $k$-indexed $\mathsf{LTL}\backslash\mathsf{X}$ specifications (i.e., specifications with $k$ quantifiers over processes in the prenex of the formula) can be reduced to combining the results of model-checking finitely many topologies of size at most $2k$ [4, Theorem 4]. Their proof implies that, for each $k$, the PMCP for linear-time specifications in $k$-indexed $\mathsf{LTL}\backslash\mathsf{X}$ and general topologies has a cutoff.

**Questions.** Comparing these results, an obvious question is: are there cutoffs for branching time temporal logics and arbitrary topologies (see Table 1)? Clarke et al. already give a first answer [4, Corollary 3]. They prove that there is no cutoff for token-passing systems with arbitrary topologies and specifications from 2-indexed $\mathsf{CTL}\backslash\mathsf{X}$. However, their proof makes use of formulas with unbounded nesting-depth of path quantifiers. This lead us to the first question.

| | Uni-Ring Topologies | Arbitrary Topologies |
|---|---|---|
| indexed $\mathsf{LTL}\backslash\mathsf{X}$ | – | [4] |
| indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ | [7] | this paper |

Table 1: Direction-Unaware TPSs.

| | Bi-Ring Topologies | Arbitrary Topologies |
|---|---|---|
| indexed $\mathsf{LTL}\backslash\mathsf{X}$ | [6] | this paper |
| indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ | this paper | this paper |

Table 2: Direction-Aware TPSs.

*Question* 1. Is there a way to stratify $k$-indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ such that for each level of the stratification there is a cutoff for systems with arbitrary topologies? In particular, does stratification by nesting-depth of path quantifiers do the trick?

Cutoffs for $k$-indexed temporal logic fragments immediately yield that for each $k$ there is an algorithm (depending on $k$) for deciding the PMCP for $k$-indexed temporal logic. However, this does not imply that there is an algorithm that can compute the cutoff for a given $k$. In particular, it does not imply that PMCP for full prenex indexed temporal logic is decidable.

*Question* 2. For which topologies (rings, cliques, all?) can one conclude that the PMCP for the full prenex indexed temporal logic is decidable?

Finally, an important implicit assumption in Clarke et al. [4] is that processes are not *direction aware*, i.e., they cannot sense or choose in which direction the token is sent, or from which direction it is received. In contrast, Emerson and Kahlon [6] show that cutoffs exist for certain direction-aware systems in bi-directional rings (see Section 8). We were thus motivated to understand to what extent existing results about cutoffs can be lifted to direction-aware systems, see Table 2.

*Question* 3. Do cutoffs exist for direction-aware systems on arbitrary topologies and $k$-indexed temporal logics (such as $\mathsf{LTL}\backslash\mathsf{X}$ and $\mathsf{CTL}\backslash\mathsf{X}$)?

**Our contributions.** In this paper, we answer the questions above, unifying and substantially extending the known cutoff results:

*Answer to Question* 1. Our main positive result (Theorem 7) states that for arbitrary parameterized topologies $\mathbf{G}$ there is a cutoff for specifications in $k$-indexed $\mathsf{CTL}^*_d\backslash\mathsf{X}$ — the cutoff depends on $\mathbf{G}$, the number $k$ of the process quantifiers, and the nesting depth $d$ of path quantifiers. In particular, indexed $\mathsf{LTL}\backslash\mathsf{X}$ is included in the case $d = 1$, and so our result generalizes the results of Clarke et al. [4].

*Answer to Question* 2. We prove (Theorem 14) that there exist topologies for which the PMCP is undecidable for specifications in prenex indexed $\mathsf{CTL}\backslash\mathsf{X}$ or $\mathsf{LTL}\backslash\mathsf{X}$. Note that this undecidability result does not contradict the existence of cutoffs (Theorem 7), since cutoffs may not be computable from $k, d$ (see the note on decidability in Section 2.4). However, for certain topologies our positive result is constructive and we can compute cutoffs given $k$ and $d$ (Theorem 15). To illustrate, we show that rings have a cutoff of $2k$, cliques of $k+1$, and stars of $k+1$ (independent of $d$). In particular, PMCP is decidable for these topologies and specifications in prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$.

*Answer to Question* 3. The results just mentioned assume that processes are not direction-aware. Our main negative result (Theorem 17) states that if processes can control at least one of the directions (i.e., choose in which direction to send or from which direction to receive) then the PMCP for arbitrary topologies and $k$-indexed logic (even $\mathsf{LTL}\backslash\mathsf{X}$ and $\mathsf{CTL}\backslash\mathsf{X}$) is undecidable, and therefore does not have cutoffs. Moreover, if processes can control both in- and out-directions, then the PMCP is already undecidable for bi-directional rings and 1-indexed $\mathsf{LTL}\backslash\mathsf{X}$.

*Technical contributions relative to previous work.* Our main positive result (Theorem 7) generalizes proof techniques and ideas from previous results [7,4]. We observe that in both of these papers the main idea is to abstract a TPS by simulating the quantified processes exactly and simulating the movement of the token between these processes. The relevant information about the movement of the token is this: whether there is a direct edge, or a path (through unquantified processes) from one quantified process to another. This abstraction does not work for $\mathsf{CTL}^*_d\backslash\mathsf{X}$ and general topologies since the formula can express branching properties of the token movement. Our main observation is that the relevant information about the branching-possibilities of the token can be expressed in $\mathsf{CTL}^*_d\backslash\mathsf{X}$ over the topology itself. We develop a composition-like theorem, stating that if two topologies (with $k$ distinguished vertices) are indistinguishable by $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas, then the TPSs based on these topologies and an arbitrary process template $P$ are indistinguishable by $\mathsf{CTL}^*_d\backslash\mathsf{X}$ (Theorem 9). The machinery involves a generalization of stuttering trace-equivalence [15], a notion of $d$-contraction that serves the same purpose as the connection topologies of Clarke et al. [4, Proposition 1], and also the main simulation idea of Emerson and Namjoshi [7, Theorem 2].

Our main negative result, undecidability of PMCP for direction-aware systems (Theorem 17), is proven by a reduction from the non-halting problem for 2-counter machines (as is typical in this area [7,10]). Due to the lack of space, full proofs are omitted, and can be found in the full version [1].

## 2 Definitions and Existing Results

Let $\mathbb{N}$ denote the set of positive integers. Let $[k]$ for $k \in \mathbb{N}$ denote the set $\{1, \ldots, k\}$. The concatenation of strings $u$ and $w$ is written $uw$ or $u \cdot w$.

Let $\mathsf{AP}$ denote a countably infinite set of *atomic propositions* or *atoms*. A *labeled transition system (LTS) over* $\mathsf{AP}$ is a tuple $(Q, Q_0, \Sigma, \delta, \lambda)$ where $Q$ is the set of *states*, $Q_0 \subseteq Q$ are the *initial states*, $\Sigma$ is the set of *transition labels* (also called *action labels*), $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*, and $\lambda : Q \to 2^{\mathsf{AP}}$ is the *state-labeling* and satisfies that $\lambda(q)$ is finite (for every $q \in Q$). Transitions $(q, \sigma, q') \in \delta$ may be written $q \xrightarrow{\sigma} q'$.

A *state-action path* of an LTS $(Q, Q_0, \Sigma, \delta, \lambda)$ is a finite sequence of the form $q_0\sigma_0q_1\sigma_1 \ldots q_n \in (Q\Sigma)^*Q$ or an infinite sequence $q_0\sigma_0q_1\sigma_1 \cdots \in (Q\Sigma)^\omega$ such that $(q_i, \sigma_i, q_{i+1}) \in \delta$ (for all $i$). A *path* of an LTS is the projection $q_0q_1 \ldots$ of a state-action path onto states $Q$. An *action-labeled path* of an LTS is the projection $\sigma_0\sigma_1 \ldots$ of a state-action path onto transition labels $\Sigma$.

### 2.1 System Model (Direction-Unaware)

In this section we define the LTS $P^G$ — it consists of replicated copies of a process $P$ placed on the vertices of a graph $G$. Transitions in $P^G$ are either internal (in which exactly one process moves) or synchronized (in which one

process sends the token to another along an edge of $G$). The token starts with the process that is at the initial vertex of $G$.

Fix a countably infinite set of (local) atomic propositions $\mathsf{AP}_{\mathsf{pr}}$ (to be used by the states of the individual processes).

**Process Template $P$.** Let $\Sigma_{\mathsf{int}}$ denote a finite non-empty set of *internal-transition labels*. Define $\Sigma_{\mathsf{pr}}$ as the disjoint union $\Sigma_{\mathsf{int}} \cup \{\mathsf{rcv}, \mathsf{snd}\}$ where $\mathsf{rcv}$ and $\mathsf{snd}$ are new symbols.

A *process template* $P$ is a LTS $(Q, Q_0, \Sigma_{\mathsf{pr}}, \delta, \lambda)$ over $\mathsf{AP}_{\mathsf{pr}}$ such that:

i) the state set $Q$ is finite and can be partitioned into two non-empty sets, say $T \cup N$. States in $T$ are said to *have the token*.
ii) The initial state set is $Q_0 = \{\iota_t, \iota_n\}$ for some $\iota_t \in T, \iota_n \in N$.
iii) Every transition $q \xrightarrow{\mathsf{snd}} q'$ satisfies that $q$ has the token and $q'$ does not.
iv) Every transition $q \xrightarrow{\mathsf{rcv}} q'$ satisfies that $q'$ has the token and $q$ does not.
v) Every transition $q \xrightarrow{\mathsf{a}} q'$ with $\mathsf{a} \in \Sigma_{\mathsf{int}}$ satisfies that $q$ has the token if and only if $q'$ has the token.
vi) The transition relation $\delta$ is total in the first coordinate: for every $q \in Q$ there exists $\sigma \in \Sigma_{\mathsf{pr}}, q' \in Q$ such that $(q, \sigma, q') \in \delta$ (i.e., the process $P$ is non-terminating).
vii) Every infinite action-labeled path $a_0 a_1 \ldots$ is in the set $(\Sigma_{\mathsf{int}}^* \ \mathsf{snd} \ \Sigma_{\mathsf{int}}^* \ \mathsf{rcv})^\omega \cup (\Sigma_{\mathsf{int}}^* \ \mathsf{rcv} \ \Sigma_{\mathsf{int}}^* \ \mathsf{snd})^\omega$ (i.e., $\mathsf{snd}$ and $\mathsf{rcv}$ actions alternate continually along every infinite action-labeled path of $P$). [1]

The elements of $Q$ are called *local states* and the transitions in $\delta$ are called *local transitions (of $P$)*. A local state $q$ such that the only transitions are of the form $q \xrightarrow{\mathsf{snd}} q'$ (for some $q'$) is said to be *send-only*. A local state $q$ such that the only transitions are of the form $q \xrightarrow{\mathsf{rcv}} q'$ (for some $q'$) is said to be *receive-only*.

**Topology $G$.** A *topology* is a directed graph $G = (V, E, x)$ where $V = [k]$ for some $k \in \mathbb{N}$, vertex $x \in V$ is the *initial vertex*, $E \subseteq V \times V$, and $(v, v) \notin E$ for every $v \in V$. Vertices are called *process indices*.

We may also write $G = (V_G, E_G, x_G)$ if we need to disambiguate.

**Token-Passing System $P^G$.** Let $\mathsf{AP}_{\mathsf{sys}} := \mathsf{AP}_{\mathsf{pr}} \times \mathbb{N}$ be the *indexed atomic propositions*. For $(p, i) \in \mathsf{AP}_{\mathsf{sys}}$ we may also write $p_i$. Given a process template $P = (Q, Q_0, \Sigma_{\mathsf{pr}}, \delta, \lambda)$ over $\mathsf{AP}_{\mathsf{pr}}$ and a topology $G = (V, E, x)$, define the *token-passing system (TPS) $P^G$* as the finite LTS $(S, S_0, \Sigma_{\mathsf{int}} \cup \{\mathsf{tok}\}, \Delta, \Lambda)$ over atomic propositions $\mathsf{AP}_{\mathsf{sys}} := \mathsf{AP}_{\mathsf{pr}} \times \mathbb{N}$, where:

- The set $S$ of *global states* is $Q^V$, i.e., all functions from $V$ to $Q$. If $s \in Q^V$ is a global state then $s(i)$ denotes the local state of the process with index $i$.
- The set of *global initial states* $S_0$ consists of the unique global state $s_0 \in Q_0^V$ such that only $s_0(x)$ has the token (here $x$ is the initial vertex of $G$).
- The labeling $\Lambda(s) \subset \mathsf{AP}_{\mathsf{sys}}$ for $s \in S$ is defined as follows: $p_i \in \Lambda(s)$ if and only if $p \in \lambda(s(i))$, for $p \in \mathsf{AP}_{\mathsf{pr}}$ and $i \in V$.
- The *global transition relation* $\Delta$ is defined to consist of the set of all internal transitions and synchronous transitions:

[1] This restriction was introduced by Emerson and Namjoshi in [7]. Our positive results that cutoffs exist also hold for a more liberal restriction (see Section 7).

- An *internal transition* is an element $(s, \mathsf{a}, s')$ of $S \times \Sigma_{\mathsf{int}} \times S$ for which there exists a process index $v \in V$ such that
  i) $s(v) \xrightarrow{\mathsf{a}} s'(v)$ is a local transition of $P$, and
  ii) for all $w \in V \setminus \{v\}$, $s(w) = s'(w)$.
- A *token-passing transition* is an element $(s, \mathsf{tok}, s')$ of $S \times \{\mathsf{tok}\} \times S$ for which there exist process indices $v, w \in V$ such that $(v, w) \in E$ and
  i) $s(v) \xrightarrow{\mathsf{snd}} s'(v)$ is a local transition of $P$,
  ii) $s(w) \xrightarrow{\mathsf{rcv}} s'(w)$ is a local transition of $P$, and
  iii) for every $u \in V \setminus \{v, w\}$, $s'(u) = s(u)$.

In words, the system $P^G$ can be thought of the asynchronous parallel composition of $P$ over topology $G$. The token starts with process $x$. At each time step either exactly one process makes an internal transition, or exactly two processes synchronize when one process sends the token to another along an edge of $G$.

### 2.2 System Model (Direction-Aware)

Inspired by direction-awareness in the work of Emerson and Kahlon [6], we extend the definition of TPS to include additional labels on edges, called *directions*. The idea is that processes can restrict which directions are used when they send or receive the token.

Fix finite non-empty disjoint sets $\mathsf{Dir}_{\mathsf{snd}}$ of *sending directions* and $\mathsf{Dir}_{\mathsf{rcv}}$ of *receiving directions*. A *direction-aware token-passing system* is a TPS with the following modifications.

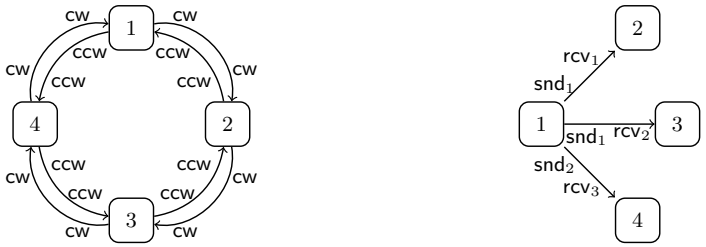
**Direction-aware Topology.** A *direction-aware topology* is a topology $G = (V, E, x)$ with labeling functions $\mathsf{dir}_{\mathsf{rcv}} : E \to \mathsf{Dir}_{\mathsf{rcv}}$, $\mathsf{dir}_{\mathsf{snd}} : E \to \mathsf{Dir}_{\mathsf{snd}}$.

**Direction-aware Process Template.** For process templates of direction-aware systems, transition labels are taken from $\Sigma_{\mathsf{pr}} := \Sigma_{\mathsf{int}} \cup \mathsf{Dir}_{\mathsf{snd}} \cup \mathsf{Dir}_{\mathsf{rcv}}$. The definition of a *direction-aware process template* is like that in Section 2.1, except that in item iii) $\mathsf{snd}$ is replaced by $\mathsf{d} \in \mathsf{Dir}_{\mathsf{snd}}$, in iv) $\mathsf{rcv}$ is replaced by $\mathsf{d} \in \mathsf{Dir}_{\mathsf{rcv}}$, and in vii) $\mathsf{snd}$ is replaced by $\mathsf{Dir}_{\mathsf{snd}}$ and $\mathsf{rcv}$ by $\mathsf{Dir}_{\mathsf{rcv}}$ .

**Direction-aware Token Passing System.** Fix $\mathsf{Dir}_{\mathsf{snd}}$ and $\mathsf{Dir}_{\mathsf{rcv}}$, let $G$ be a direction-aware topology and $P$ a direction-aware process template. Define the *direction-aware token-passing system* $P^G$ as in Section 2.1, except that token-passing transitions are now direction-aware: *direction-aware token-passing transitions* are elements $(s, \mathsf{tok}, s')$ of $S \times \{\mathsf{tok}\} \times S$ for which there exist process indices $v, w \in V$ with $(v, w) \in E$, $\mathsf{dir}_{\mathsf{snd}}(v, w) = \mathsf{d}$, and $\mathsf{dir}_{\mathsf{rcv}}(v, w) = \mathsf{e}$, such that:

i) $s(v) \xrightarrow{\mathsf{d}} s'(v)$ is a local transition of $P$.
ii) $s(w) \xrightarrow{\mathsf{e}} s'(w)$ is a local transition of $P$.
iii) For every $u \in V \setminus \{v, w\}$, $s'(u) = s(u)$.

**Notations $\mathcal{P}_{\mathsf{u}}$, $\mathcal{P}_{\mathsf{snd}}$, $\mathcal{P}_{\mathsf{rcv}}$, $\mathcal{P}_{\mathsf{sndrcv}}$.** Let $\mathcal{P}_u$ denote the set of all process templates for which $|\mathsf{Dir}_{\mathsf{snd}}| = |\mathsf{Dir}_{\mathsf{rcv}}| = 1$. In this case $P^G$ degenerates to a direction-unaware TPS as defined in Section 2.1. If we require $|\mathsf{Dir}_{\mathsf{rcv}}| = 1$, then processes cannot choose from which directions to receive the token, but possibly in which




(a) Bi-directional ring with directions.

(b) Directed topology with multiple edges of same direction.

Fig. 1: Direction-aware Topologies.

direction to send it. Denote the set of all such process templates by $\mathcal{P}_{snd}$. Similarly define $\mathcal{P}_{rcv}$ to be all process templates where $|\mathsf{Dir}_{\mathsf{snd}}| = 1$ — processes cannot choose where to send the token, but possibly from which direction to receive it. Finally, let $\mathcal{P}_{sndrcv}$ be the set of all direction-aware process templates.

**Examples.** Figure 1(a) shows a bi-directional ring with directions $\mathsf{cw}$ (clockwise) and $\mathsf{ccw}$ (counterclockwise). Every edge $e$ is labeled with an outgoing direction $dir_{\mathsf{snd}}(e)$ and an incoming direction $dir_{\mathsf{rcv}}(e)$.[2] Using these directions, a process that has the token can choose whether he wants to send it in direction $\mathsf{cw}$ or $\mathsf{ccw}$. Depending on its local state, a process waiting for the token can also choose to receive it only from direction $\mathsf{cw}$ or $\mathsf{ccw}$.

Figure 1(b) depicts a topology in which process 1 can choose between two outgoing directions. If it sends the token in direction $\mathsf{snd}_1$, it may be received by either process 2 or 3. If however process 2 blocks receiving from direction $\mathsf{rcv}_1$, the token can only be received by process 3. If 3 additionally blocks receiving from $\mathsf{rcv}_2$, then this token-passing transition is disabled.

### 2.3 Indexed Temporal Logics

*Indexed temporal logics (ITL)* were introduced in [3,8,7] to model specifications of certain distributed systems. Subsequently one finds a number of variations of indexed temporal-logics in the literature (linear vs. branching, restrictions on the quantification). Thus we introduce Indexed-$\mathsf{CTL}^*$ which has these variations (and those in [4]) as syntactic fragments.

**Syntactic Fragments of $\mathsf{CTL}^*$, and $\equiv_{\mathsf{TL}}$.** We assume the reader is familiar with the syntax and semantics of $\mathsf{CTL}^*$, for a reminder see [2]. For $d \in \mathbb{N}$ let $\mathsf{CTL}^*_d\backslash\mathsf{X}$ denote the syntactic fragment of $\mathsf{CTL}^*\backslash\mathsf{X}$ in which the nesting-depth of path quantifiers is at most $d$ (for a formal definition see [15, Section 4]).

Let $\mathsf{TL}$ denote a temporal logic (in this paper these are fragments of $\mathsf{CTL}^*\backslash\mathsf{X}$). For temporal logic $\mathsf{TL}$ and LTSs $M$ and $N$, write $M \equiv_{\mathsf{TL}} N$ to mean that for every formula $\phi \in \mathsf{TL}$, $M \models \phi$ if and only if $N \models \phi$.

[2] For notational simplicity, we denote both outgoing direction $\mathsf{snd}_{\mathsf{cw}}$ and incoming direction $\mathsf{rcv}_{\mathsf{cw}}$ by $\mathsf{cw}$, and similarly for $\mathsf{ccw}$.

**Indexed-CTL$^*$.** Fix an infinite set $\mathsf{Vars} = \{x, y, z, \dots\}$ of index variables, i.e., variables with values from $\mathbb{N}$. These variables refer to vertices in the topology.

*Syntax.* The *Indexed-CTL$^*$ formulas over variable set Vars and atomic propositions AP* are formed by adding the following rules to the syntax of $\mathsf{CTL}^*$ over atomic propositions $\mathsf{AP} \times \mathsf{Vars}$. We write $p_x$ instead of $(p, x) \in \mathsf{AP} \times \mathsf{Vars}$.

If $\phi$ is an indexed-$\mathsf{CTL}^*$ state (resp. path) formula and $x, y \in \mathsf{Vars}, Y \subset \mathsf{Vars}$, then the following are also indexed-$\mathsf{CTL}^*$ state (resp. path) formulas:

- $\forall x.\ \phi$ and $\exists x.\phi$ (i.e., for all/some vertices in the topology, $\phi$ should hold),
- $\forall x.\ x \in Y \rightarrow \phi$ and $\exists x.x \in Y \wedge \phi$ (for all/some vertices that that are designated by variables in $Y$),
- $\forall x.\ x \in E(y) \rightarrow \phi$ and $\exists x.x \in E(y) \wedge \phi$ (for all/some vertices to which there is an edge from the vertex designated by the variable $y$).

*Index Quantifiers.* We use the usual shortands (e.g., $\forall x \in Y.\ \phi$ is shorthand for $\forall x.\ x \in Y \rightarrow \phi$). The quantifiers introduced above are called called *index quantifiers*, denoted $Qx$.

*Semantics.* Indexed temporal logic is interpreted over a system instance $P^G$ (with $P$ a process template and $G$ a topology). The formal semantics are in the full version of the paper [1]. Here we give some examples. The formula $\forall i.\, \mathsf{E}\,\mathsf{F}\, p_i$ states that for every process there exists a path such that, eventually, that process is in a state that satisfies atom $p$. The formula $\mathsf{E}\,\mathsf{F}\,\forall i.p_i$ states that there is a path such that eventually all processes satisfy atom $p$ simultaneously. We now define the central fragment that includes the former example and not the latter.

**Prenex indexed TL and $\{\forall, \exists\}^k$-TL.** *Prenex indexed temporal-logic* is a syntactic fragment of indexed temporal-logic in which all quantifiers are at the front of the formula, e.g., prenex indexed $\mathsf{LTL}\backslash\mathsf{X}$ consists of formulas of the form $(Q_1 x_1) \dots (Q_k x_k)\ \varphi$ where $\varphi$ is an $\mathsf{LTL}\backslash\mathsf{X}$ formula over atoms $\mathsf{AP} \times \{x_1, \dots, x_k\}$, and the $Q_i x_i$s are index quantifiers. Such formulas with $k$ quantifiers will be referred to as *$k$-indexed*, collectively written $\{\forall, \exists\}^k$-TL. The union of $\{\forall, \exists\}^k$-TL for $k \in \mathbb{N}$ is written $\{\forall, \exists\}^*$-TL and called (full) prenex indexed TL.

### 2.4 Parameterized Model Checking Problem, Cutoffs, Decidability

A *parameterized topology* $\mathbf{G}$ is a countable set of topologies. E.g., the set of unidirectional rings with all possible initial vertices is a parameterized topology.

**$\mathsf{PMCP}_{\mathbf{G}}(-,-)$.** The *parameterized model checking problem (PMCP)* for parameterized topology $\mathbf{G}$, processes from $\mathcal{P}$, and parameterized specifications from $\mathcal{F}$, written $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$, is the set of pairs $(\varphi, P) \in \mathcal{F} \times \mathcal{P}$ such that for all $G \in \mathbf{G}$, $P^G \models \varphi$. A *solution* to $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is an algorithm that, given a formula $\varphi \in \mathcal{F}$ and a process template $P \in \mathcal{P}$ as input, outputs 'Yes' if for all $G \in \mathbf{G}$, $P^G \models \varphi$, and 'No' otherwise.

**Cutoff.** A *cutoff* for $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is a natural number $c$ such that for every $P \in \mathcal{P}$ and $\varphi \in \mathcal{F}$, the following are equivalent:

- $P^G \models \varphi$ for all $G \in \mathbf{G}$ with $|V_G| \leq c$;
- $P^G \models \varphi$ for all $G \in \mathbf{G}$.

Thus $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ *does not have a cutoff* iff for every $c \in \mathbb{N}$ there exists $P \in \mathcal{P}$ and $\varphi \in \mathcal{F}$ such that $P^G \models \varphi$ for all $G \in \mathbf{G}$ with $|V_G| \leq c$, and there exists $G \in \mathbf{G}$ such that $P^G \not\models \varphi$.

**Observation 1.** *If $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ has a cutoff, then $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is decidable*

Indeed: if $c$ is a cutoff, let $G_1, \ldots, G_n$ be all topologies $G$ in $\mathbf{G}$ such that $|V_G| \leq c$. The algorithm that solves PMCP takes $P, \varphi$ as input and checks whether or not $P^{G_i} \models \varphi$ for all $1 \leq i \leq n$.

**Note About Decidability.** The following statements are not, a priori, equivalent (for given parameterized topology $\mathbf{G}$ and process templates $\mathcal{P}$):

- For every $k \in \mathbb{N}$, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^k\text{-}\mathsf{TL})$ is decidable.
- $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^*\text{-}\mathsf{TL})$ is decidable.

The first item says that for every $k$ there *exists* an algorithm $A_k$ that solves the PMCP for $k$-indexed $\mathsf{TL}$. This does not imply the second item, which says that there exists an algorithm that solves the PMCP for $\cup_{k\in\mathbb{N}}\{\forall, \exists\}^k\text{-}\mathsf{TL}$. If the function $k \mapsto A_k$ is also *computable* (e.g., Theorem 15) then indeed the second item follows: given $P, \varphi$, extract the size $k$ of the prenex block of $\varphi$, *compute* a description of $A_k$, and run $A_k$ on $P, \varphi$.

For instance, the result of Clarke et al. — that there are cutoffs for $k$-index $\mathsf{LTL}\backslash\mathsf{X}$ and arbitrary topologies — does not imply that the PMCP for $\{\forall, \exists\}^*\text{-}\mathsf{LTL}\backslash\mathsf{X}$ and arbitrary topologies is decidable. Aware of this fact, the authors state (after Theorem 4) "Note that the theorem does not provide us with an effective means to find the reduction [i.e. algorithm]...".

In fact, we prove (Theorem 14) that there is some parameterized topology such that PMCP is undecidable for prenex indexed $\mathsf{LTL}\backslash\mathsf{X}$.

**Existing Results.** We restate the known results using our terminology.
A *uni-directional ring* $G = (V, E, x)$ is a topology with $V = [n]$ for some $n \in \mathbb{N}$, there are edges $(i, i+1)$ for $1 \leq i \leq n$ (arithmetic is modulo $n$), and $x \in V$. Let $\mathbf{R}$ be the parameterized topology consisting of all uni-directional rings.

**Theorem 2 (Implict in [7]).** *For every $k \in \mathbb{N}$, there is a cutoff for the problem $PMCP_{\mathbf{R}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{CTL}^*\backslash\mathsf{X})$.* [3]

Although Clarke et al. [4] do not explicitly state the following theorem, it follows from their proof technique, which we generalize in Section 3.

**Theorem 3 (Implicit in [4]).** *For every parameterized topology $\mathbf{G}$, and every $k \in \mathbb{N}$, the problem $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{LTL}\backslash\mathsf{X})$ has a cutoff.* [4].

**Theorem 4 ([4, Corollary 3]).** *There exists a parameterized topology $\mathbf{G}$ and process $P \in \mathcal{P}_u$ such that the problem $\mathsf{PMCP}_{\mathbf{G}}(\{P\}, \{\exists\}^2\text{-}\mathsf{CTL}\backslash\mathsf{X})$ does not have a cutoff.*

[3] The paper explicitly contains the result that 4 is a cutoff for $\forall\forall\text{-}\mathsf{CTL}^*\backslash\mathsf{X}$ on rings. However the proof ideas apply to get the stated theorem.

[4] Khalimov et al. [13, Corollary 2] state that $2k$ is a cutoff if $\mathbf{G}$ is taken to be $\mathbf{R}$. However this is an error: $2k$ is a cutoff only for formulas with no quantifier alternations. See Remark 16 in Section 5.

The proof of this theorem defines $\mathbf{G}$, process $P$, and for every $c \in \mathbb{N}$ a formula $\varphi_c$, such that if $G \in \mathbf{G}$ then $P^G \models \varphi_c$ if and only if $|V_G| \leq c$. The formula $\varphi_c$ is in 2-indexed $\mathsf{CTL}\backslash\mathsf{X}$ and has nesting depth of path quantifiers equal to $c$.

## 3 Method for Proving Existence of Cutoffs

We give a method for proving cutoffs for direction-*un*aware TPSs that will be used to prove Theorem 7.

In a $k$-indexed $\mathsf{TL}$ formula $Q_1 x_1 \dots Q_k x_k.\ \varphi$, every valuation of the variables $x_1, \dots, x_k$ designates $k$ nodes of the underlying topology $G$, say $\bar{g} = g_1, \dots, g_k$. The formula $\varphi$ can only talk about (the processes in) $\bar{g}$. In order to prove that the PMCP has a cutoff, it is sufficient (as the proof of Theorem 5 will demonstrate) to find conditions on two topologies $G, G'$ and $\bar{g}, \bar{g}'$ that allow one to conclude that $P^G$ and $P^{G'}$ are indistinguishable with respect to $\varphi$.

We define two abstractions for a given TPS $P^G$. The first abstraction simulates $P^G$, keeping track only of the local states of processes indexed by $\bar{g}$. We call it the *projection* of $P^G$ onto $\bar{g}$.[5] The second abstraction only simulates the movement of the token in $G$, restricted to $\bar{g}$. We call it the *graph LTS* of $G$ and $\bar{g}$.

*Notation.* Let $\bar{g}$ denote a tuple $(g_1, \dots, g_k)$ of *distinct* elements of $V_G$, and $\bar{g}'$ a $k$-tuple of distinct elements of $V_{G'}$. Write $v \in \bar{g}$ if $v = g_i$ for some $i$.

**The projection $P^G|\bar{g}$.** Informally, the *projection* of $P^G$ onto a tuple of process indices $\bar{g}$ is the LTS $P^G$ and a new labeling that, for every $g_i \in \bar{g}$, replaces the indexed atom $p_{g_i}$ by the atom $p@i$; all other atoms are removed. Thus $p@i$ means that the atom $p \in \mathsf{AP_{pr}}$ holds in the process with index $g_i$. In other words, process indices are replaced by their *positions* in $\bar{g}$.

Formally, fix process $P$, topology $G$, and $k$-tuple $\bar{g}$ over $V_G$. Define the *projection of* $P^G = (S, S_0, \Sigma_{\mathsf{int}} \cup \{\mathsf{tok}\}, \Delta, \Lambda)$ *onto* $\bar{g}$, written $P^G|\bar{g}$ as the LTS $(S, S_0, \Sigma_{\mathsf{int}} \cup \{\mathsf{tok}\}, \Delta, L)$ over atomic propositions $\{p@i : p \in \mathsf{AP_{pr}}, i \in [k]\}$, where for all $s \in S$ the labeling $L(s)$ is defined as follows: $L(s) := \{p@i : p_{g_i} \in \Lambda(s), i \in [k]\}$.

**The graph LTS $G|\bar{g}$.** Informally, $G|\bar{g}$ is an LTS where states are nodes of the graph $G$, and transitions are edges of $G$. The restriction to $\bar{g}$ is modeled by labeling a state with the position of the corresponding node in $\bar{g}$.

Let $G = (V, E, x)$ be a topology, and let $\bar{g}$ be a $k$-tuple over $V_G$. Define the *graph LTS* $G|g$ as the LTS $(Q, Q_0, \Sigma, \Delta, \Lambda)$ over atomic propositions $\{1, \dots, k\}$, with state set $Q := V$, initial state set $Q_0 := \{x\}$, action set $\Sigma = \{\mathsf{a}\}$, transition relation with $(v, \mathsf{a}, w) \in \Delta$ iff $(v, w) \in E$, and labeling $\Lambda(v) := \{i\}$ if $v = g_i$ for some $1 \leq i \leq k$, and $\emptyset$ otherwise.

Fix a non-indexed temporal logic $\mathsf{TL}$, such as $\mathsf{CTL}^*\backslash\mathsf{X}$. We now define what it means for $\mathsf{TL}$ to have the reduction property and the finiteness property. Informally, the reduction property says that if $G$ and $G'$ have the same connectivity

[5] Emerson and Namjoshi [7, Section 2.1] define the related notion "LTS projection".

(with respect to $\mathsf{TL}$ and only viewing $k$-tuples $\bar{g}, \bar{g}'$) then $P^G$ and $P^{G'}$ are indistinguishable (with respect to $\mathsf{TL}$-formulas over process indices in $\bar{g}, \bar{g}'$).

**REDUCTION property for $\mathsf{TL}$**[6]

For every $k \in \mathbb{N}$, process $P \in \mathcal{P}_u$, topologies $G, G'$, $k$-tuples $\bar{g}, \bar{g}'$, if $G|\bar{g} \equiv_{\mathsf{TL}} G'|\bar{g}'$ then $P^G|\bar{g} \equiv_{\mathsf{TL}} P^{G'}|\bar{g}'$.

**FINITENESS property for $\mathsf{TL}$**

For every $k \in \mathbb{N}$, there are finitely many equivalence classes $[G|\bar{g}]_{\equiv_{\mathsf{TL}}}$ where $G$ is an arbitrary topology, and $\bar{g}$ is a $k$-tuple over $V_G$.

**Theorem 5 (REDUCTION & FINITENESS $\implies$ Cutoffs for $\{\forall,\exists\}^k$-$\mathsf{TL}$).** *If $\mathsf{TL}$ satisfies the* REDUCTION *and the* FINITENESS *property, then for every $k, \mathbf{G}$, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall,\exists\}^k\text{-}\mathsf{TL})$ has a cutoff.*

*Proof.* Fix quantifier prefix $Q_1x_1 \dots Q_kx_k$. We prove that there exist finitely many topologies $G_1, \cdots, G_N \in \mathbf{G}$ such that for every $G \in \mathbf{G}$ there is an $i \leq N$ such that for all $P \in \mathcal{P}_u$, and all $\mathsf{TL}$-formulas $\varphi$ over atoms $\mathsf{AP}_{\mathsf{pr}} \times \{x_1, \cdots, x_k\}$

$$P^G \models Q_1x_1 \dots Q_kx_k.\ \varphi \iff P^{G_i} \models Q_1x_1 \dots Q_kx_k.\ \varphi$$

In particular, $\max\{|V_{G_i}| : 1 \leq i \leq N\}$ is a cutoff for $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall,\exists\}^k\text{-}\mathsf{TL})$.

Suppose for simplicity of exposition that $Q_ix_i$ is a quantifier that also expresses that the value of $x_i$ is different from the values of $x_j \in \{x_1, \dots, x_{i-1}\}$.[7] Fix representatives of $[G|\bar{g}]_{\equiv_{\mathsf{TL}}}$ and define a function $r$ that maps $G|\bar{g}$ to the representative of $[G|\bar{g}]_{\equiv_{\mathsf{TL}}}$. Define a function $rep$ that maps $P^G|\bar{g}$ to $P^H|\bar{h}$, where $r(G|\bar{g}) = H|\bar{h}$.

For every $\equiv_{\mathsf{TL}}$-representative $H|\bar{h}$ (i.e., $H|\bar{h} = r(G|\bar{g})$ for some topology $G$ and $k$-tuple $\bar{g}$), introduce a new Boolean proposition $q_{H|\bar{h}}$. By the FINITENESS property of $\mathsf{TL}$ there are finitely many such Boolean propositions, say $n$.

Define a valuation $e_\varphi$ (that depends on $\varphi$) of these new atoms by

$$e_\varphi(q_{H|\bar{h}}) := \begin{cases} \top & \text{if } P^H|\bar{h} \models \varphi[p_{x_j} \mapsto p@j] \\ \bot & \text{otherwise.} \end{cases}$$

For every $G \in \mathbf{G}$ define Boolean formula $B_G := (\overline{Q_1}g_1 \in V_G) \dots (\overline{Q_k}g_k \in V_G)\ q_{r(G|\bar{g})}$, where $\overline{Q}$ is the Boolean operation corresponding to $Q$, e.g., $\overline{\exists} g_i \in V_G$ is interpreted as $\bigvee_{g \in V_G \setminus \{g_1,\dots,g_{i-1}\}}$.[8]

[6] Properties of this type are sometimes named *composition* instead of *reduction*, see for instance [14].

[7] All the types of quantifiers defined in Section 2.3, such as $\exists x \in E(y)$, can be dealt with similarly at the cost of notational overhead.

[8] Note that in the Boolean propositions $G$ is fixed while $\bar{g} = (g_1, \dots, g_n)$ ranges over $(V_G)^k$ and is determined by the quantification.

Then (for all $P, G$ and $\varphi$)

$$\begin{aligned} & P^G \models Q_1 x_1 \dots Q_k x_k.\ \varphi \\ \iff & Q_1 g_1 \in V_G \dots Q_k g_k \in V_G : P^G \models \varphi[p_{x_j} \mapsto p_{g_j}] \\ \iff & Q_1 g_1 \in V_G \dots Q_k g_k \in V_G : P^G|\bar{g} \models \varphi[p_{x_j} \mapsto p@j] \\ \iff & Q_1 g_1 \in V_G \dots Q_k g_k \in V_G : rep(P^G|\bar{g}) \models \varphi[p_{x_j} \mapsto p@j] \\ \iff & e_\varphi(B_G) = \top \end{aligned}$$

Here $\varphi[p_{x_j} \mapsto p_{g_j}]$ is the formula resulting from replacing every atom in $\varphi$ of the form $p_{x_j}$ by the atom $p_{g_j}$, for $p \in \mathsf{AP_{pr}}$ and $1 \le j \le k$. Similarly $\varphi[p_{x_j} \mapsto p@j]$ is defined as the formula resulting from replacing (for all $p \in \mathsf{AP_{pr}}, j \le k$) every atom in $\varphi$ of the form $p_{x_j}$ by the atom $p@j$. The first equivalence is by the definition of semantics of indexed temporal logic; the second is by the definition of $P^G|\bar{g}$; the third is by the REDUCTION property of $\mathsf{TL}$; the fourth is by the definition of $e_\varphi$ and *rep*.

Fix $B_{G_1}, \dots, B_{G_N}$ (with $G_i \in \mathbf{G}$) such that every $B_G$ ($G \in \mathbf{G}$) is logically equivalent to some $B_{G_i}$. Such a finite set of formulas exists since there are $2^{2^n}$ Boolean formulas (up to logical equivalence) over $n$ Boolean propositions, and thus at most $2^{2^n}$ amongst the $B_G$ for $G \in \mathbf{G}$.

By the equivalences above conclude that for every $G \in \mathbf{G}$ there exists $i \le N$ such that $P^G \models Q_1 x_1 \dots Q_k x_k.\ \varphi$ if and only if $P^{G_i} \models Q_1 x_1 \dots Q_k x_k.\ \varphi$. Thus '$\forall G \in \mathbf{G}, P^G \models \varphi$' is equivalent to '$\bigwedge_{i \le N} e_\varphi(B_{G_i})$' and so the integer $c := \max\{|V_{G_i}| : 1 \le i \le N\}$ is a cutoff for $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{TL})$. □

*Remark 6.* The theorem implies that for every $k, \mathbf{G}$, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^k\text{-}\mathsf{TL})$ is decidable. Further, fix $\mathbf{G}$ and suppose that given $k$ one could compute the finite set $G_1, \cdots, G_N$. Then by the last sentence in the proof one can compute the cutoff $c$. In this case, $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \{\forall, \exists\}^*\text{-}\mathsf{TL})$ is decidable.

## 4 Existence of Cutoffs for $k$-indexed $\mathsf{CTL}^*_d\backslash\mathsf{X}$

The following theorem answers Question 1 from the introduction.

**Theorem 7.** *Let* $\mathbf{G}$ *be a parameterized topology. Then for all* $k, d \in \mathbb{N}$*, the problem* $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{CTL}^*_d\backslash\mathsf{X})$ *has a cutoff.*

**Corollary 8.** *Let* $\mathbf{G}$ *be a parameterized topology. Then for all* $k, d \in \mathbb{N}$*, the problem* $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall, \exists\}^k\text{-}\mathsf{CTL}^*_d\backslash\mathsf{X})$ *is decidable.*

To prove the Theorem it is enough, by Theorem 5, to show that the logic $\{\forall, \exists\}^k\text{-}\mathsf{CTL}^*_d\backslash\mathsf{X}$ has the REDUCTION property and the FINITENESS property.

**Theorem 9 (Reduction).** *For all* $d, k \in \mathbb{N}$*, topologies* $G, G'$*, processes* $P \in \mathcal{P}_u$*,* $k$*-tuples* $\bar{g}$ *over* $V_G$ *and* $k$*-tuples* $\bar{g}'$ *over* $V_{G'}$*:*

*If* $G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} G'|\bar{g}'$ *then* $P^G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} P^{G'}|\bar{g}'$.

The idea behind the proof is to show that paths in $P^G$ can be simulated by paths in $P^{G'}$ (and vice versa). Given a path $\pi$ in $P^G$, first project it onto $G$ to get a path $\rho$ that records the movement of the token, then take an equivalent path $\rho'$ in $G'$ which exists since $G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} G'|\bar{g}'$, and then lift $\rho'$ up to get a path $\pi'$ in $P^{G'}$ that is equivalent to $\pi$. This lifting step uses the assumption that process $P$ is in $\mathcal{P}_u$, i.e., $P$ cannot control where it sends the token, or from where it receives it. The proof can be found in the full version of the paper [1].

*Remark 10.* As immediate corollaries we get that the REDUCTION property holds with $\mathsf{TL} = \mathsf{LTL}\backslash\mathsf{X}$ (take $d = 1$), $\mathsf{CTL}^*\backslash\mathsf{X}$ (since if the assumption holds with $\mathsf{TL} = \mathsf{CTL}^*\backslash\mathsf{X}$ then the conclusion holds with $\mathsf{TL} = \mathsf{CTL}^*_d\backslash\mathsf{X}$ for all $d \in \mathbb{N}$, and thus also for $\mathsf{TL} = \mathsf{CTL}^*\backslash\mathsf{X}$) and, if $P$ is finite, also for $\mathsf{TL} = \mathsf{CTL}\backslash\mathsf{X}$ (since $\mathsf{CTL}\backslash\mathsf{X}$ and $\mathsf{CTL}^*\backslash\mathsf{X}$ agree on finite structures).

**Finiteness Theorem.** Theorem 4 ([4, Corollary 3]) states that there exists $\mathbf{G}$ such that the problem $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \exists\exists\text{-}\mathsf{CTL}^*\backslash\mathsf{X})$ does not have a cutoff. We observed that the formulas from their result have unbounded nesting depth of path quantifiers. This leads to the idea of stratifying $\mathsf{CTL}^*\backslash\mathsf{X}$ by nesting depth.

Recall from Section 2.3 that i) $\mathsf{CTL}^*_d\backslash\mathsf{X}$ denotes the syntactic fragment of $\mathsf{CTL}^*\backslash\mathsf{X}$ in which formulas have path-quantifier nesting depth at most $d$; ii) $M \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} N$ iff $M$ and $N$ agree on all $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas. Write $[M]_{\mathsf{CTL}^*_d\backslash\mathsf{X}}$ for the set of all LTSs $N$ such that $M \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} N$.

Following the method of Section 3 we prove that the following FINITENESS property holds (where $k$ represents the number of process-index quantifiers in the prenex indexed temporal logic formula).

*Remark 11.* For ease of exposition we sketch a proof under the assumption that path quantifiers in formulas ignore runs in which the token does not visit every process infinitely often. This is an explicit restriction in [4] and implicit in [7]. In the full version [1] we remove this restriction. For the purpose of this paper this restriction only affects the explicit cutoffs in Theorem 15.

**Theorem 12 (Finiteness).** *For all positive integers $k$ and $d$, there are finitely many equivalence classes $[G|\bar{g}]_{\equiv_{CTL^*_d\backslash X}}$ where $G$ is an arbitrary topology, and $\bar{g}$ is a $k$-tuple over $V_G$.*

*Proof Idea.* We provide an algorithm that given positive integers $k, d$, topology $G$, $k$-tuple $\bar{g}$ over $V_G$, returns a LTS $\textsc{con}_d G|\bar{g}$ such that $G|\bar{g} \equiv_{\mathsf{CTL}^*_d\backslash\mathsf{X}} \textsc{con}_d G|\bar{g}$. Moreover, we prove that for fixed $k, d$ the range of $\textsc{con}_d\text{-}|\text{-}$ is finite.

Recursively define a marking function $\mu_d$ that associates with each $v \in V_G$ a finite set (of finite strings over alphabet $\mu_{d-1}(V_G)$). For the base case define $\mu_0(v) := \Lambda(v)$, the labeling of $G|\bar{g}$. The marking $\mu_d(v)$ stores (representatives) of all strings of $\mu_{d-1}$-labels of paths that start in $v$ and reach some element in $\bar{g}$. The idea is that $\mu_d(v)$ determines the set of $\mathsf{CTL}^*_d\backslash\mathsf{X}$ formulas that hold in $G|\bar{g}$ with initial vertex $v$, as follows: stitch together these strings, using elements of $\bar{g}$ as stitching points, to get the $\mathsf{CTL}^*_d\backslash\mathsf{X}$ types of the infinite paths starting in

$v$. This allows us to define a topology, called the $d$-contraction $\textsc{con}_d G|\bar{g}$, whose vertices are the $\mu_d$-markings of vertices in $G$. In the full version [1] we prove that $G|\bar{g}$ is $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalent to its $d$-contraction, and that the number of different $d$-contractions is finite, and depends on $k$ and $d$.

**Definition of $d$-contraction $\textsc{con}_d G|\bar{g}$.** Next we define $d$-contractions.

*Marking $\mu_d$.* Fix $k, d \in \mathbb{N}$, topology $G$, and $k$-tuple $\bar{g}$ over $V_G$. Let $\Lambda$ be the labeling-function of $G|\bar{g}$, i.e., $\Lambda(v) = \{i\}$ if $v = g_i$, and $\Lambda(v) = \emptyset$ for $v \notin \bar{g}$. For every vertex $v \in V_G$ define a set $X(v)$ of paths of $G$ as follows: a path $\pi = \pi_1 \dots \pi_t$, say of length $t$, is in $X(v)$ if $\pi$ starts in $v$, none of $\pi_1, \dots, \pi_{t-1}$ is in $\bar{g}$, and $\pi_t \in \bar{g}$. Note that $X(g_i) = \{g_i\}$.

Define the marking $\mu_d$ inductively:

$$\mu_d(v) := \begin{cases} \Lambda(v) & \text{if } d = 0 \\ \{\mathsf{destutter}(\mu_{d-1}(\pi_1) \dots \mu_{d-1}(\pi_t)) : \pi_1 \dots \pi_t \in X(v), t \in \mathbb{N}\} & \text{if } d > 0, \end{cases}$$

where $\mathsf{destutter}(w)$ is the maximal substring $s$ of $w$ such that for every two consecutive letters $s_i$ and $s_{i+1}$ we have that $s_i \neq s_{i+1}$. Informally, remove identical consecutive letters of $w$ to get the 'destuttering' $\mathsf{destutter}(w)$.

The elements of $\mu_d(v)$ ($d > 0$) are finite strings over the alphabet $\mu_{d-1}(V_G)$. For instance, strings in $\mu_1(v)$ are over the alphabet $\{\{1\}, \{2\}, \dots, \{k\}, \emptyset\}$.

*Equivalence relation $\sim_d$.* Vertices $v, u \in V_G$ are *$d$-equivalent*, written $u \sim_d v$, if $\mu_d(v) = \mu_d(u)$. We say that $\sim_d$ *refines* $\sim_j$ if $u \sim_d v$ implies $u \sim_j v$.

**Lemma 13.** *If $0 \leq j < d$, then $\sim_d$ refines $\sim_j$.*

Indeed, observe that for all nodes $v$, all strings in $\mu_d(v)$ start with the letter $\mu_{d-1}(v)$. Thus $\mu_d(v) = \mu_d(u)$ implies that $\mu_{d-1}(v) = \mu_{d-1}(u)$. In other words, if $u \sim_d v$ then $u \sim_{d-1} v$, and thus also $u \sim_j v$ for $0 \leq j < d$.

*$d$-contraction $\textsc{con}_d G|\bar{g}$.* Define an LTS $\textsc{con}_d G|\bar{g}$ called the $d$-contraction of $G|\bar{g}$ as follows. The nodes of the contraction are the $\sim_d$-equivalence classes. Put an edge between $[u]_{\sim_d}$ and $[v]_{\sim_d}$ if there exists $u' \in [u]_{\sim_d}, v' \in [v]_{\sim_d}$ and an edge in $G$ from $u'$ to $v'$. The initial state is $[x]_{\sim_d}$ where $x$ is the initial vertex of $G$. The label of $[u]_{\sim_d}$ is defined to be $\Lambda(u)$ — this is well-defined because, by Lemma 13, $\sim_d$ refines $\sim_0$.

In the full version [1] we prove that $G|\bar{g}$ is $\mathsf{CTL}^*_d\backslash\mathsf{X}$-equivalent to its $d$-contraction, and that the number of different $d$-contractions is finite, and depends on $k$ and $d$. □

## 5 Cutoffs for $k$-index $\mathsf{CTL}^*\backslash\mathsf{X}$ and Concrete Topologies

The following two theorems answer Question 2 from the introduction, regarding the PMCP for specifications from $\{\forall, \exists\}^*$-$\mathsf{CTL}^*\backslash\mathsf{X}$.

First, the PMCP is undecidable for certain (pathological) parameterized topologies $\mathbf{G}$ and specifications from $\{\forall, \exists\}^*$-$\mathsf{CTL}^*\backslash\mathsf{X}$.

**Theorem 14.** *There exists a process $P \in \mathcal{P}_u$, and parameterized topologies* $\mathbf{G}$, $\mathbf{H}$, *such that the following PMCPs are undecidable*
1. $\mathit{PMCP}_{\mathbf{G}}(\{P\}, \{\forall, \exists\}^*\text{-}\mathit{LTL}\backslash\mathsf{X})$.
2. $\mathit{PMCP}_{\mathbf{H}}(\{P\}, \{\forall, \exists\}^2\text{-}\mathit{CTL}\backslash\mathsf{X})$.

*Moreover,* $\mathbf{G}$ *and* $\mathbf{H}$ *can be chosen to be computable sets of topologies.*

Second, PMCP is decidable for certain (regular) parameterized topologies and specifications from $\{\forall\}^*\text{-}\mathsf{CTL}^*\backslash\mathsf{X}$. This generalizes results from Emerson and Namjoshi [7] who show this result for $\{\forall\}^k\text{-}\mathsf{CTL}^*\backslash\mathsf{X}$ with $k = 1, 2$ and uni-directional ring topologies. By Remark 11, these cutoffs apply under the assumption that we ignore runs that do not visit every process infinitely often.

**Theorem 15.** *If* $\mathbf{G}$ *is as stated, then* $\mathit{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall\}^k\text{-}\mathit{CTL}^*\backslash\mathsf{X})$ *has the stated cutoff.*
1. *If* $\mathbf{G}$ *is the set of uni-directional rings, then* $2k$ *is a cutoff.*
2. *If* $\mathbf{G}$ *is the set of bi-directional rings, then* $2k$ *is a cutoff.*
3. *If* $\mathbf{G}$ *is the set of cliques, then* $k+1$ *is a cutoff.*
4. *If* $\mathbf{G}$ *is the set of stars, then* $k+1$ *is a cutoff.*

*Consequently, for each* $\mathbf{G}$ *listed,* $\mathit{PMCP}_{\mathbf{G}}(\mathcal{P}_u, \{\forall\}^*\text{-}\mathit{CTL}^*\backslash\mathsf{X})$ *is decidable.*

This theorem is proved following Remark 6: given $k, d$, we compute a set $G_1, \ldots, G_N \in \mathbf{G}$ such that every $B_G$ for $G \in \mathbf{G}$ is logically equivalent to some $B_{G_i}$, where the Boolean formula $B_G$ is defined as $\bigwedge_{\bar{g}} q_{\text{CON}_d G|\bar{g}}$. To do this, note that $B_G$ is logically equivalent to $B_H$ if and only if $\{\text{CON}_d G|\bar{g} : \bar{g} \in V_G\} = \{\text{CON}_d H|\bar{h} : \bar{h} \in V_H\}$ (this is where we use that there is no quantifier alternation). So it is sufficient to prove that, if $c$ is the stated cutoff,

$$|V_G|, |V_H| \geq c \implies \{\text{CON}_d G|\bar{g} : \bar{g} \in V_G\} = \{\text{CON}_d H|\bar{h} : \bar{h} \in V_H\}$$

To illustrate how to do this, we analyze the case of uni-directional rings and cliques (the other cases are similar).

*Uni-directional rings.* Suppose $\mathbf{G}$ are the uni-directional rings and let $G \in \mathbf{G}$. Fix a $k$-tuple of distinct elements of $V_G$, say $(g_1, g_2, \ldots, g_k)$. Define a function $f : V_G \to \{g_1, \ldots, g_k\}$ that maps $v$ to the first element of $\bar{g}$ on the path $v, v+1, v+2, \ldots$ (addition is mod $|V_G|$). In particular $f(g_i) = g_i$ for $i \in [k]$. In the terminology of the proof of Theorem 12, $X(v)$ consists of the simple path $v, v+1, \cdots, f(v)$.

We now describe $\mu_d$. Clearly $\mu_d(g_i) = \{\mu_{d-1}(g_i)\}$. By induction on $d$ one can prove that if $v \notin \bar{g}$ with $f(v) = g_j$ then $\mu_d(v) = \{\mu_{d-1}(v) \cdot \mu_{d-1}(g_j)\}$. So for every $d > 1$, the equivalences $\sim_d$ and $\sim_1$ coincide.

| $d$ | 0 | 1 | 2 | $\ldots$ |
|---|---|---|---|---|
| $\mu_d(v)$ for $v = g_i$ | $\{i\}$ | $\{\{i\}\}$ | $\{\{\{i\}\}\}$ | $\ldots$ |
| $\mu_d(v)$ if $v \notin \bar{g}$ and $f(v) = g_j$ | $\emptyset$ | $\{\emptyset \cdot \{j\}\}$ | $\{\{\emptyset \cdot \{j\}\} \cdot \{\{j\}\}\}$ | $\ldots$ |

Thus for every $k \in \mathbb{N}$, the $d$-contraction $\text{CON}_d G|\bar{g}$ is a ring of size at most $2k$ (in particular, it is independent of $d$). In words, the $d$-contraction of $G$ is the ring

resulting by identifying adjacent elements not in $\bar{g}$. It is not hard to see that if $G, H$ are rings such that $|V_G|, |V_H| \geq 2k$ then for every $\bar{g}$ there exists $\bar{h}$ such that $\text{CON}_d G|\bar{g} = \text{CON}_d H|\bar{h}$.

*Cliques.* Fix $n \in \mathbb{N}$. Let $G$ be a clique of size $n$. That is: $V_G = [n]$ and $(i,j) \in E_G$ for $1 \leq i \neq j \leq n$. Fix a $k$-tuple of distinct elements of $V_G$, say $(g_1, g_2, \ldots, g_k)$. We now describe $\mu_d(v)$. Clearly $\mu_d(g_i) = \{\mu_{d-1}(g_i)\}$ and for $v \notin \bar{g}$ we have $\mu_d(v) = \{\mu_{d-1}(v) \cdot \mu_{d-1}(j) : j \in [k]\}$. So for every $d > 1$, the equivalences $\sim_d$ and $\sim_1$ coincide, and the $d$-contraction $\text{CON}_d G|\bar{g}$ is the clique of size $k+1$. In words, the $d$-contraction of $G$ results from $G$ by identifying all vertices not in $\bar{g}$. It is not hard to see that if $G, H$ are cliques such that $|V_G|, |V_H| \geq k+1$ then for every $\bar{g}$ (of size $k$) there exists $\bar{h}$ such that $\text{CON}_d G|\bar{g} = \text{CON}_d H|\bar{h}$.

*Remark 16.* For cliques and stars, $k+1$ is also a cutoff for $\{\forall, \exists\}^k$-$\mathsf{CTL}^*\backslash\mathsf{X}$. Also, $2k$ is *not* a cutoff for uni-rings and $\{\forall, \exists\}^k$-$\mathsf{LTL}\backslash\mathsf{X}$ as stated in [13, Corollary 2]. To see this, let $tok_i$ express that the process with index $i$ has the token, and $adj(k,i) := tok_i \rightarrow tok_i \,\mathsf{U}\, tok_k \vee tok_k \rightarrow tok_k \,\mathsf{U}\, tok_i$. Then the formula $\exists i \exists j \forall k.\ adj(k,i) \vee adj(k,j)$, holds in the ring of size 6, but not 7.

## 6 There are No Cutoffs for Direction-Aware Systems

In the following, we consider systems where processes can choose which directions are used to send or receive the token, i.e., process templates are from $\mathcal{P}_{snd}$, $\mathcal{P}_{rcv}$, or $\mathcal{P}_{sndrcv}$. Let $\mathbf{B}$ be the parameterized topology of all bi-directional rings, with directions $\mathsf{cw}$ (clockwise) and $\mathsf{ccw}$ (counter-clockwise). The following theorem answers Question 3 from the introduction.

**Theorem 17.** *1. $\mathsf{PMCP}_{\mathbf{B}}(\mathcal{P}_{sndrcv}, \forall\text{-}\mathsf{LTL}\backslash\mathsf{X})$ is undecidable.*
*2. For $\mathcal{F}$ equal to $\{\forall\}^9$-$\mathsf{LTL}\backslash\mathsf{X}$ or $\{\exists\}^9$-$\mathsf{CTL}\backslash\mathsf{X}$, and $\mathcal{P} \in \{\mathcal{P}_{snd}, \mathcal{P}_{rcv}\}$, there exists a parameterized topology $\mathbf{G}$ such that $\mathsf{PMCP}_{\mathbf{G}}(\mathcal{P}, \mathcal{F})$ is undecidable.*

*Proof Idea.* We reduce the non-halting problem of two-counter machines (2CMs) to the PMCP. The idea is that one process, the *controller*, simulates the finite-state control of the 2CM. The other processes, arranged in a chain or a ring, are *memory processes*, collectively storing the counter values with a fixed memory per process. This allows a given system to simulate a 2CM with bounded counters. Since a 2CM terminates if and only if it terminates for some bound on the counter values, we can reduce the non-halting problem of 2CMs to the PMCP. The main work is to show that the controller can issue commands, such as 'increment counter 1' and 'test counter 1 for zero'. We give a detailed proof sketch for part 1 of the theorem, and then outline a proof for part 2.
*1. $\forall$-$\mathsf{LTL}\backslash\mathsf{X}$ and $\mathcal{P}_{sndrcv}$ in bi-directional rings.*
The process starting with the token becomes the controller, all others are memory, each storing one bit for each counter of the 2CM. The current value of a counter $c$ is the total number of corresponding bits ($c$-bits) set to 1. Thus, a system with $n$ processes can store counter values up to $n-1$.

Fix a numbering of 2CM-commands, say $0 \mapsto$ 'increment counter 1', $1 \mapsto$ 'decrement counter 1', $2 \mapsto$ 'test counter 1 for zero', etc. Every process has a command variable that represents the command to be executed when it receives the token from direction ccw.

If the controller sends the token in direction cw, the memory processes will increment (mod 6) the command variable, allowing the controller to encode which command should be executed. Every process just continues to pass the token in direction cw, until it reaches the controller again.

If the controller sends the token in direction ccw, then the memory processes try to execute the command currently stored. If it is an 'increment counter $c$' or 'decrement counter $c$' command, the memory process tries to execute it (by incrementing/decrementing its $c$-bit). If the process cannot execute the command (because the $c$-bit is already 1 for an increment, or 0 for a decrement), then it passes the token along direction ccw and remembers that a command is being tried. If the token reaches a memory process which can execute the command, then it does so and passes the token back in direction cw. The processes that remembered that a command is being tried will receive the token from direction cw, and know that the command has been successfully executed, and so will the controller. If the controller gets the token from ccw, the command failed. In this case, the controller enters a loop in which it just passes the token in direction cw (and no more commands are executed).

If the command stored in the memory processes is a 'test for zero counter $c$', then the processes check if their $c$-bit is 0. If this is the case, it (remembers that a command is being tried and) passes the token to the next process in direction ccw. If the token reaches a process for which the $c$-bit is 1, then this process sends the token back in direction cw. Other memory processes receiving it from cw (and remembering that the command is being tried), pass it on in direction cw. In this case, the controller will receive the token from cw and know that counter $c$ is not zero. On the other hand, if all memory processes store 0 in their $c$-bit, then they all send the token in direction ccw. Thus, the controller will receive it from ccw and knows that counter $c$ currently is zero. To terminate the command, it sends the token in direction cw, and all processes (which remembered that a command is being tried), know that execution of this command is finished.

With the description above, a system with $n-1$ memory processes can simulate a 2CM as long as counter values are less than $n$. Let $HALT$ be an atomic proposition that holds only in the controller's halting states. Then solving the PMCP for $\forall i\, \mathsf{G}\, \neg HALT_i$ amounts to solving the non-halting problem of the 2CM.

*2.* $\{\forall\}^9$*-LTL*$\backslash X$ *and* $\mathcal{P}_{snd}$.

We give a proof outline. In this case there are $2n$ memory processes, $n$ for each counter $c \in \{1, 2\}$. The remaining 9 processes are special and called 'controller', 'counter $c$ is zero', 'counter $c$ is not zero', 'counter $c$ was incremented', and 'counter $c$ was decremented'. When the controller wants to increment or decrement counter $c$, it sends the token non-deterministically to some memory process for counter $c$. When the controller wants to test counter $c$ for zero, it sends the token to the first memory process. When a memory process receives the token

it does not know who sent it, and in particular does not know the intended command. Thus, it non-deterministically takes the appropriate action for one of the possible commands. If its bit is set to 0 then it either i) increments its bit and sends the token to a special process 'counter $c$ was incremented', or ii) it sends the token to the next memory node in the chain, or to the special process 'counter $c$ is zero' if it is the last in the chain. If its bit is set to 1 then it either i) decrements its bit and sends the token to a special process 'counter $c$ was decremented', or ii) sends the token to a special process 'counter $c$ is not zero'.

Even though incoming directions are not available to the processes, we can write the specification such that, out of all the possible non-deterministic runs, we only consider those in which the controller receives the token from the expected special node (the formula requires one quantified index variable for each of the special nodes). So, if the controller wanted to increment counter $c$ it needs to receive the token from process 'counter $c$ was incremented'. If the controller receives the token from a different node, it means that a command was issued but not executed correctly, and the formula disregards this run. Otherwise, the system of size $2n+1$ correctly simulates the 2CM until one of the counter values exceeds $n$. □

## 7 Extensions

There are a number of extensions of direction-unaware TPSs for which the theorems that state existence of cutoffs (Theorems 7 and 15) still hold. We describe these in order to highlight assumptions that make the proofs work:

1. Processes can be infinite-state.
2. The EN-restriction on the process template $P$ can be relaxed: replace item *vii*) in Definition 2.1 by "For every state $q$ that has the token there is a finite path $q \dots q'$ such that $q'$ does not have the token, and for every $q$ that does not have the token there is a finite path $q \dots q'$ such that $q'$ has the token".
3. One can further allow *direction-sensing* TPSs, which is a direction-aware TPS with an additional restriction on the process template: "If $q \xrightarrow{\mathsf{d}} q' \in \delta$ for some direction $d \in \mathsf{Dir_{snd}}$, then for every $d \in \mathsf{Dir_{snd}}$ there exists a transition $q \xrightarrow{\mathsf{d}} q'' \in \delta$"; and a similar statement for $\mathsf{Dir_{rcv}}$. Informally: we can allow processes to change state according to the direction that the token is (non-deterministically) sent to or received, but the processes are not allowed to block any particular direction.
4. One can further allow the token to carry a value but with the strong restriction that from every state that has the token and every value $v$ there is a path of internal actions in $P$ which eventually sends the token with value $v$, and the same for receiving.

These conditions on $P$ all have the same flavor: they ensure that a process can not choose what information to send/receive, whether that information is a value on the token or a direction for the token.

## 8 Related work

Besides the results that this paper is directly based on [16,7,4], there are several other relevant papers.

Emerson and Kahlon [6] consider token-passing in uni- and bi-directional rings, where processes are direction-aware and tokens carry messages (but can only be changed a bounded number of times). However, the provided cutoff theorems only hold for specifications that talk about two processes (in a uni-directional ring) or one process (in a bi-directional ring), process templates need to be deterministic, an cutoffs depend on the size of the process implementation.

German and Sistla [11] provide cutoffs for the PMCP for systems with pairwise synchronization. Although pairwise synchronization can simulate token-passing, their cutoff results are restricted to cliques and 1-indexed $\mathsf{LTL}$. Moreover, their proof uses vector-addition systems with states and their cutoff depends on the process template and the specification formula.

Delzanno et al. [5] study a model of broadcast protocols on arbitrary topologies, in which a process can synchronize with all of its available neighbors 'at once' by broadcasting a message (from a finite set of messages). They prove undecidability of PMCP for systems with arbitrary topologies and 1-indexed safety properties, and that the problem becomes decidable if one restricts the topologies to 'graphs with bounded paths' (such as stars). Their proof uses the machinery of well-structured transitions systems, and no cutoffs are provided. They also show undecidability of the PMCP in the case of non-prenex indexed properties of the form $\mathsf{G}(\exists i.s(i) \in B)$ on general and the restricted topologies.

Rabinovich [14, Section 4] proves, using the composition method, that if monadic second-order theory of the set of topologies in $\mathbf{G}$ is decidable, then the PMCP is decidable for propositional modal logic. The systems considered are defined by a very general notion of product of systems (which includes our token passing systems as a subcase).

The PMCP for various fragments of non-prenex indexed $\mathsf{LTL}$ is undecidable, see German and Sistla [11, Section 6] for systems with pairwise synchronization, and John et al. [12, Appendix $A$] for systems with no synchronization at all.

## 9 Summary

The goal of this work was to find out under what conditions there are cutoffs for temporal logics and token-passing systems on general topologies. We found that stratifying prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$ by nesting-depth of path quantifiers allowed us to recover the existence of cutoffs; but that there are no cutoffs if the processes are allowed to choose the direction of the token. In all the considered cases where there is no cutoff we show that the PMCP problem is actually undecidable.

Our positive results are provided by a construction that generalizes and unifies the known positive results, and clearly decomposes the problem into two aspects: tracking the movement of the token through the underlying topology, and simulating the internal states of the processes that the specification formula

can see. The construction yields small cutoffs for common topologies (such as rings, stars, and cliques) and specifications from prenex indexed $\mathsf{CTL}^*\backslash\mathsf{X}$.

**Acknowledgments.** We thank Roderick Bloem for detailed comments on numerous drafts and Krishnendu Chatterjee for important comments regarding the structure of the paper. We thank Roderick Bloem, Igor Konnov, Helmut Veith, and Josef Widder for discussions at an early stage of this work that, in particular, pointed out the relevance of direction-unawareness in [4].

## References

1. Aminof, B., Jacobs, S., Khalimov, A., Rubin, S.: Parameterized Model Checking of Token-Passing Systems (2013), pre-print on arxiv.org
2. Baier, C., Katoen, J.P., et al.: Principles of model checking, vol. 26202649. MIT press Cambridge (2008)
3. Browne, M.C., Clarke, E.M., Grumberg, O.: Reasoning about networks with many identical finite state processes. Inf. Comput. 81, 13–31 (April 1989)
4. Clarke, E., Talupur, M., Touili, T., Veith, H.: Verification by network decomposition. In: CONCUR 2004. vol. 3170, pp. 276–291 (2004)
5. Delzanno, G., Sangnier, A., Zavattaro, G.: Parameterized verification of ad hoc networks. In: CONCUR. LNCS, vol. 6269, pp. 313–327 (2010)
6. Emerson, E.A., Kahlon, V.: Parameterized model checking of ring-based message passing systems. In: CSL. LNCS, vol. 3210, pp. 325–339. Springer (2004)
7. Emerson, E.A., Namjoshi, K.S.: On reasoning about rings. Int. J. Found. Comput. Sci. 14(4), 527–550 (2003)
8. Emerson, E.A., Sistla, A.P.: Symmetry and model checking. In: CAV. pp. 463–478 (1993)
9. Emerson, E., Namjoshi, K.: Reasoning about rings. In: POPL. pp. 85–94 (1995)
10. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. Logic in Computer Science, Symposium on 0, 352 (1999)
11. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39(3), 675–735 (1992)
12. John, A., Konnov, I., Schmid, U., Veith, H., Widder, J.: Counter attack on byzantine generals: Parameterized model checking of fault-tolerant distributed algorithms. CoRR abs/1210.3846 (2012)
13. Khalimov, A., Jacobs, S., Bloem, R.: Towards efficient parameterized synthesis. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI, LNCS, vol. 7737, pp. 108–127. Springer Berlin Heidelberg (2013)
14. Rabinovich, A.: On compositionality and its limitations. ACM Trans. Comput. Logic 8(1) (Jan 2007)
15. Shamir, S., Kupferman, O., Shamir, E.: Branching-depth hierarchies. ENTCS 39(1), 65 – 78 (2003)
16. Suzuki, I.: Proving properties of a ring of finite-state machines. Inf. Process. Lett. 28(4), 213–214 (Jul 1988)