Uniform Strategies, Rational Relations and Jumping Automata

Laura Bozzelli[a], Bastien Maubert[b], Sophie Pinchinat[b]

[a] *Facultad de Informática, UPM, Madrid, Spain*
[b] *Université de Rennes 1, IRISA, Rennes, France*

Abstract

A general concept of uniform strategies has recently been proposed as a relevant notion in game theory for computer science, which subsumes various notions from the literature. It relies on properties involving sets of plays in two-player turn-based arenas equipped with arbitrary binary relations between plays; these properties are expressed in a language based on CTL^* with a quantifier over related plays. There are two semantics for our quantifier, a *strict* one and a *full* one, that we study separately. Regarding the strict semantics, the existence of a uniform strategy is undecidable for *rational* binary relations, but introducing *jumping tree automata* and restricting attention to *recognizable* relations allows us to establish a 2-Exptime-complete complexity – and still capture a class of two-player imperfect-information games with epistemic temporal objectives. Regarding the full semantics, relying on *information set automata* we establish that the existence of a uniform strategy is decidable for rational relations and we provide a nonelementary synthesis procedure. We also exhibit an essentially optimal subclass of rational relations for which the problem becomes 2-Exptime-complete. Considering rich classes of relations makes the theory of uniform strategies powerful: it directly entails various results in logics of knowledge and time, some of them already known, and others new.

Keywords: Games, Imperfect information, Uniform strategies, Logics of knowledge and time, Rational relations, Jumping automata

1. Introduction

Infinite-duration game models have been intensively studied for their applications in computer science (Apt and Grädel, 2011) and logic (Grädel et al., 2002). First, infinite-duration games provide a natural abstraction of computing systems' non-terminating interaction (**?**) – think of a communication protocol between a printer and its users, or control systems. Second, infinite-duration games naturally occur as a tool to handle logical systems for the specification of non-terminating behaviors, such as for the propositional μ-calculus (Emerson and Jutla, 1991), leading to a powerful theory of automata, logics and infinite games (Grädel et al., 2002) and to the development of algorithms for the automatic verification ("model-checking") and synthesis of hardware and software systems. In all cases, solving games aims at computing a strategy (of some distinguished player) whose outcomes fulfill ω-regular conditions meant to describe some desirable property.

In essence, ω-regular conditions are evaluated on individual plays, independently of other plays that result from the strategy. However, turning to imperfect-information games raises the need to deal with *sets* of plays, as the strategic decision has to be the same in indistinguishable situations (Reif, 1984). This typical property of strategies in imperfect-information games is in general dealt with aside from the ω-regular winning conditions. However, this splitting is a real issue when considering properties of strategies that mix, *e.g.* knowledge and time.

In an attempt to study this problem in depth, Maubert and Pinchinat (2014) introduced a general notion of *uniform strategies* and showed that it captures a variety of settings from the literature. Uniformity properties of strategies are expressed in a logic that combines standard temporal modalities with two new quantifiers, $\boxminus$ and $\boxbox$, that universally quantify over "related" plays according to a binary relation between plays. The difference between the two quantifiers is their range. While the *strict* quantifier $\boxminus$ only quantifies over related plays that follow the strategy, the *full* quantifier $\boxbox$ ranges over all related plays in the arena.

These quantifiers generalize the knowledge operator K of epistemic temporal logics: classically, the semantics of K is a universal quantification over histories related to the actual one by some observational equivalence relation that captures the capabilities of the agent – perfect/imperfect recall, synchronous/asynchronous, ... (Halpern and Vardi, 1989). In contrast, the setting of Maubert and Pinchinat (2014) allows arbitrary binary relations as long as they are *rational*, *i.e.* recognized by *finite state transducers* (Eilenberg, 1974; Berstel, 1979). Noticeably, most equivalence relations used in epistemic temporal logics are recognized by very simple transducers. Additionally, rational relations need not be equivalences, and can capture relations used in belief revision, and for modelling plausibility, with K45 or KD45 axiomatization (Fagin et al., 1995).

In this work we extend the setting of Maubert and Pinchinat (2014) by founding our logical language $\mathcal{L}_{\rightsquigarrow}$ on the full branching time logic CTL^* instead of LTL, allowing us to capture *e.g.* games with CTL^* winning conditions and module-checking (Kupferman and Vardi, 1997). Constraining the uniformity properties to use only either the strict quantifier or the full quantifier yields the notions of *strictly-uniform strategies* and *fully-uniform strategies*, which we study separately as they require different techniques.

Relying on Maubert and Pinchinat (2013), we first establish the undecidability of the *strictly-uniform strategy problem* (i.e. the existence of a strictly-uniform strategy) when the whole class of rational relations is considered. More precisely, we show that this undecidability result holds even if we restrict attention to the subclass of regular[1] equivalence relations. To try and better understand the difficulty of this problem, we propose an automata-based approach inspired by Vardi (1991) for solving LTL games. We introduce and study *jumping tree automata (JTA)*, a class of tree automata which generalizes standard alternating tree automata. JTA are equipped with a binary relation between branches of trees and, in addition to the usual behaviour of alternating automata, they allow for jumps between related nodes of the input tree. Intuitively, the jumps of JTA "implement" the meaning of the $\boxminus$ quantifier in $\mathcal{L}_{\rightsquigarrow}$. We show that JTA capture the full logic $\mathcal{L}_{\rightsquigarrow}$, and that from a uniformity property we can build a JTA that accepts the tree unfoldings of strictly-uniform strategies.

[1] captured by synchronous transducers

Although the emptiness problem for JTA is unsurprisingly undecidable when considering arbitrary rational relations over branches of trees, we identify a decidable case when the class of binary relations between branches is confined to the well-known family of *recognizable* relations; basically, such relations only challenge a bounded amount of information in each branch. Decidability of JTA emptiness in this case is shown by an effective transformation of JTA with recognizable relations into equivalent two-way tree automata. The emptiness problem for JTA with recognizable relations is then Exptime-complete, and the strictly-uniform strategy problem for this class of relations is 2-Exptime-complete.

Concerning fully-uniform strategies, deciding their existence (the *fully-uniform strategy problem*) has been investigated by Maubert and Pinchinat (2014) for the case of linear time modalities. The problem is in k-Exptime for logical specifications that involve up to k nested $\boxbslash$ quantifiers – 2-Exptime if $k \leq 2$. We prove that these complexities still hold when the full branching time logic CTL^* is allowed, and we establish the matching lower bounds. We also introduce information set automata as a tool to compute a generalized notion of information sets for rational relations. Information set automata offer a modular proof of our decision procedure, and they enable us to identify a rich subclass of rational relations (K45NM) that still contains relations considered in epistemic temporal logics and games with imperfect information, and for which the fully-uniform strategy problem is 2-Exptime-complete.

At last, we generalize these results to the case of multiple relations $\rightsquigarrow_i$ with corresponding quantifiers $\boxminus_i$ and $\boxbslash_i$, and we describe how several problems from the literature fit in this generalized setting. We show that several known results find a unified proof in our work, and we fill some gaps; we finally take inspiration from other related results to discuss future work.

The rest of the paper is organized as follows. In Section 2, we recall some notions concerning words, trees, game arenas and rational relations. We present in Section 3 the language $\mathcal{L}_{\rightsquigarrow}$ to specify uniform strategies, and we define them, before giving in Section 4 two examples of problems that uniform strategies naturally capture. In Section 5 we study the strictly-uniform strategy problem: we prove that it is undecidable in general and, resorting to jumping tree automata, we establish its 2-Exptime-completeness in the case of recognizable relations. Section 6 starts with the statement of our nonelementary result for the fully-uniform strategy problem and the elementary case of K45NM relations, it continues with the exposition of information set automata, which we use to establish our upper bounds, and the section ends with the proofs for the lower bounds. We extend our results to the case of multiple relations and combinations of strict and full quantifiers in Section 7, in which we also state several corollaries and discuss related work. We conclude and give perspectives in Section 8.

Because of space limitation, some proofs are omitted, and others are just sketched. However, all details can be found in Maubert (2014).

2. Preliminaries

2.1. Words and trees

For an alphabet Σ, Σ^* is the set of all *finite words* over Σ, ϵ denotes the *empty word*, $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$ is the set of nonempty finite words and Σ^ω is the set of infinite words.

For a finite word w we note $|w|$ its *length*. For a nonempty finite word $w = a_1 \dots a_n$, $\text{last}(w) := a_n$ is its last letter. For two finite words $w = a_1 \dots a_n$ and $w' = b_1 \dots b_m$, $w \cdot w' := a_1 \dots a_n b_1 \dots b_m$ is the *concatenation* of w and w'. A finite word w is a *prefix* of a word w', written $w \preccurlyeq w'$, if there exists a word w'' such that $w \cdot w'' = w'$. Finally, for a word $w = a_1 \dots a_n$, we note $\overline{w} := a_n \dots a_1$ its *mirror* word.

A *tree alphabet* Υ is a finite set of *directions*. A tree is a subset of Υ^+ that is closed for nonempty prefixes, and such that all words in the tree start with the same direction, called the root. Note that in many works, the root of a tree is the empty word ϵ. However, when considering trees that represent strategies in a game, it is convenient for us to see their root as the initial position of the game. This justifies our slightly unusual definition. We also define forests which, intuitively, can be seen as unions of trees. Finally, because the games we define in the next section have only infinite plays, we consider *leafless* trees and forests, hence the last point of the following definition.

Definition 1. Given a tree alphabet Υ, a Υ-*tree* τ, or *tree* for short when Υ is clear from the context, is a set of words $\tau \subseteq \Upsilon^+$ such that:

1. there is a one symbol word $r = \tau \cap \Upsilon$, called the *root*, such that $r \preccurlyeq x$ for all $x \in \tau$,
2. if $x \cdot d \in \tau$ and $x \neq \epsilon$, then $x \in \tau$, and
3. if $x \in \tau$ then there exists $d \in \Upsilon$ such that $x \cdot d \in \tau$.

A Υ-*forest*, or *forest* when Υ is understood, is defined likewise, removing Point 1. In other words a forest is a union of trees.

The following notions of nodes, children, parents and branches, that we define for trees, are similar for forests. The elements of a tree τ are called *nodes*. If $x \neq \epsilon$ and $x \cdot d \in \tau$, we say that $x \cdot d$ is a *child* of x, and that x is the *parent* of $x \cdot d$. We will write $x \cdot \uparrow$ for the parent of a node x: $(x \cdot d) \cdot \uparrow = x$. Note that the root has no parent. The *arity* of a node x, written $\text{arity}(x)$, is the number of children of x. If all nodes have arity at most k, then τ is a k-ary tree. Note that Υ-trees are $|\Upsilon|$-ary trees.

A *branch* is an infinite sequence $\lambda = x_0 x_1 \dots$ of nodes such that for all i, x_{i+1} is a child of x_i. For a branch $\lambda = x_0 x_1 \dots$ and an integer $i \geq 0$, $\lambda[i] = x_i$ is the i-th node on the branch, and λ^i denotes the i-th suffix $x_i x_{i+1} x_{i+2} \dots$ Given a node x of a tree τ, we let $Branches(x)$ be the set of branches $\lambda = x_0 x_1 \dots$ that start in node x, *i.e.* such that $x_0 = x$. $Branches(\tau)$ is the set of all branches in τ.

We classically allow nodes of trees and forests to carry additional information. Given a labelling alphabet Σ and a tree alphabet Υ, a Σ-*labelled* Υ-*tree*, or (Σ, Υ)-*tree* for short, is a pair $t = (\tau, \ell)$, where τ is a Υ-tree and $\ell : \tau \to \Sigma$ is a *labelling*. For a node $x = d_1 d_2 \dots d_n$ in τ, we define its *node word* $w(x)$ made of the sequence of labels from the root to this node: $w(x) = \ell(d_1)\ell(d_1 d_2) \dots \ell(d_1 \dots d_n)$. The notion of (Σ, Υ)-*forest* $\mathcal{U} = (u, \ell)$ is defined likewise. Note that we use forests to represent the universe in the semantics of $\mathcal{L}_{\rightsquigarrow}$ (see Section 3), hence the notations $\mathcal{U}$ and u.

We finish this section by defining, given a forest and a node in the forest, the tree to which this node belongs, *i.e.* the set of nodes in the forest that have the same root.

Definition 2. Let u be a Υ-forest, and let $x = d_1 \dots d_n$ be a node of u. We define the tree u_x as the "greatest" tree in the forest u that contains the node x: $u_x = \{y \in u \mid d_1 \preccurlyeq y\}$. Similarly, given a (Σ, Υ)-forest $\mathcal{U} = (u, \ell)$ and a node $x \in u$, $\mathcal{U}_x = (u_x, \ell_x)$, where u_x is as above and ℓ_x is the restriction of ℓ to the tree u_x.

2.2. Two-player games played on graphs

We present the notions of arenas, plays and strategies for the classic framework of two-player turn based games played on graphs.

Arenas

A *game arena*, or *arena* for short, is a tuple $\mathcal{G} = (V, E, V_\iota, v_\iota)$, with a set of *positions* $V = V_1 \uplus V_2$ partitioned between positions belonging to Player 1 (V_1) and those belonging to Player 2 (V_2). $E \subseteq V \times V$ is a relation between positions, describing the possible *moves* between positions, $V_\iota \subseteq V$ is a set of *starting positions* and $v_\iota \in V_\iota$ is the *initial position*. V_ι has no role in the dynamics of the game, it rather is a lever to tune the range of our full quantifier, as explained in Section 3.2. For two positions v and v', $v\,E\,v'$ denotes that $(v, v') \in E$, and $E(v) = \{v' \mid v\,E\,v'\}$ is the set of successors of v. We will assume that in every position $v \in V$ there is at least one possible move, *i.e.* $E(v) \neq \emptyset$. We say that Player i *owns* a position v if $v \in V_i$ ($i \in \{1, 2\}$). Also, when there is no ambiguity we may write $v \to v'$ instead of $v\,E\,v'$.

We will often consider game arenas with additional information attached to positions, hence the following definition. For a (finite) alphabet Σ, a Σ-*labelled game arena* is a tuple $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, where (V, E, V_ι, v_ι) is a game arena, and $\mu : V \to \Sigma$ is a labelling function. Concretely, in this work, the alphabet will be the set of possible valuations over some finite set of atomic propositions AP, *i.e.* $\Sigma = 2^{AP}$. The propositions in AP represent the relevant information for the uniformity properties one wants to state. If the game models interacting systems, this information can be the value of some (Boolean) variables or some state of communication channels. In games with imperfect information, it can be the current observation or which action has just been played. μ is extended naturally to sequences of positions: $\mu(v_1 \dots v_n) = \mu(v_1) \dots \mu(v_n)$.

We define the *size* of a labelled game arena as its number of moves: $|\mathcal{G}| = |\,E\,|$.

Plays and paths

In a (labelled or unlabelled) arena, the player owning the initial position v_ι chooses a next position v such that $v_\iota \to v$, then it is to the player owning v to choose an accessible next position, and this process continues for ever, forming an infinite sequence of positions called a *play*. Formally, we define the set of (infinite) plays $Plays_\omega \subseteq V^\omega$ as the set of infinite words $\pi = v_0 v_1 \dots$ such that $v_0 = v_\iota$ and for each $i \geq 0$, $v_i \to v_{i+1}$. We also define the notion of *partial play*, or *finite play*: a finite sequence of positions $\rho = v_0 v_1 \dots v_n$ is a partial play if it is the prefix of a play, and we note $Plays_*$ the set of partial plays.

Finally, since we will be led to consider sequences of positions that do not start in the initial position of the arena, we define for each position v the sets $Paths_\omega(v)$ and $Paths_*(v)$ of infinite paths and finite paths starting in v just like we defined $Plays_\omega$ and $Plays_*$, except that the first position has to be v instead of v_ι. In particular, $Plays_* = Paths_*(v_\iota)$ and $Plays_\omega = Paths_\omega(v_\iota)$. We also define, for $V' \subseteq V$, $Paths_*(V') = \cup_{v \in V'} Paths_*(v)$ and $Paths_\omega(V') = \cup_{v \in V'} Paths_\omega(v)$ as the sets of all finite and infinite paths starting in V'. $Paths_* = Paths_*(V)$ and $Paths_\omega = Paths_\omega(V)$ are thus the sets of all possible paths in the arena. For an infinite path $\pi = v_0 v_1 \dots$ and an integer $i \geq 0$, we use the two following notations: $\pi[i] := v_i$ is the i-th position of the path, and $\pi[0, i] = v_0 \dots v_i$ is the i-th prefix of π; we use similar notation for finite paths.

Strategies

A *strategy* for Player i is a partial function $\sigma : Plays_* \to V$ that maps a finite play ending in $v \in V_i$ to some v' such that $v\,E\,v'$. A play is said to *follow* or to be *induced* by a strategy for Player i if in this play, every time it is Player i's turn to play, she chooses the next position prescribed by the strategy: $\pi \in Plays_\omega$ is induced by σ if for all $j \geq 0$ such that $\pi[j] \in V_i$, $\pi[j+1] = \sigma(\pi[0,j])$. The *outcome* of a strategy σ, noted $\mathrm{Out}(\sigma) \subseteq Plays_\omega$, is the set of all (infinite) plays that are induced by σ.

Strategies as trees

It will often be convenient to see a finite path in a game as a node in a tree, a strategy as a tree and more generally the set of finite paths as a forest. To make this correspondence clear, take a *finite* Σ-labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ and a position $v \in V$. The set $Paths_*(v)$ is a V-tree rooted in v, and we define its natural labelling $\ell : x \cdot v' \mapsto \mu(v')$. This way, for $V' \subseteq V$, $Paths_*(V')$ can be seen as a (Σ, V)-forest that contains one tree per starting position in V', and each finite path ρ in $Paths_*(V')$ is a node in the forest; also, observe that its node word is its sequence of labels: $w(\rho) = \mu(\rho)$. In the same fashion, a strategy for Player i can be seen as a subtree of $Plays_*$ (seen as a (Σ, V)-tree) obtained by pruning some moves of Player i. Formally, a *strategy tree* $t = (\tau, \ell)$ for Player i is a Σ-labelled V-tree with root v_ι such that for every $x \in \tau$, noting $v = \mathrm{last}(x)$, it holds that:

1. if $v \in V_i$ then x has a unique child $x \cdot v'$ with $v' \in E(v)$, and $\ell(x \cdot v') = \mu(v')$
2. if $v \in V_{3-i}$ then x has one child $x \cdot v'$ for each $v' \in E(v)$, and $\ell(x \cdot v') = \mu(v')$.

Every strategy σ defines a unique strategy tree that we shall write t_σ.

Parity winning condition

A *parity game* $G = (\mathcal{G}, C)$ is a game arena $\mathcal{G}$ together with a *colouring function* $C : V \to \mathbb{N}$ with finite codomain, that assigns a *colour* or *priority* to each position. A play π is *winning* for Player 1 if the least priority of positions seen infinitely often is even. Otherwise it is winning for Player 2.

Given a parity game $G = (\mathcal{G}, C)$, a strategy σ for Player i is a *winning strategy* if Player i wins all the plays induced by σ.

2.3. Rational relations

Relations over finite words are often infinite objects, and manipulating them requires finite representations. Rational languages is a classic example of a class of (potentially) infinite languages of words that can be finitely represented, and the corresponding notion concerning relations is *rational relations*. These relations are recognized by finite state machines called transducers, which allows to address algorithmic issues involving these relations. We define finite state transducers and recall basic results concerning rational relations as well as some subclasses of interest. Much more information concerning these matters can be found in Berstel (1979).

As in this work we only consider binary rational relations, we will not define transducers for relations of arbitrary arity but just for binary ones. One way to see such a transducer is to picture a nondeterministic automaton with two tapes. Given two input finite words, one on each tape, the automaton reads them according to a set of possible

transitions, and when it has reached the end of both words, it accepts the pair if it is in an accepting state. Notice that the transducer in general does not have to progress at the same pace on both tapes.

Another way to see a transducer which will be useful is to consider that the first tape is an *input* tape and the second is an *output* tape. The transducer reads an input finite word on its input tape and writes out a finite word on its output tape. This machine being in general nondeterministic, it may have several outputs for a given input word.

Definition 3. A *Finite State Transducer* (FST) is a tuple $T = (\Sigma, \Gamma, Q, \Delta, q_\iota, Q_F)$, where Σ is an *input alphabet* and Γ an *output alphabet*, Q is a finite set of states, $q_\iota \in Q$ is the *initial state*, $Q_F \subseteq Q$ is a set of *accepting states*, and $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q$ is a finite set of *transitions*. The transducer is called *synchronous* if $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$.

Intuitively, $(q, a, b, q') \in \Delta$ means that the transducer can move from state q to state q' by reading a and writing b (both possibly ϵ, except for synchronous transducers).

We also define the *extended transition relation* $\Delta^* \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$, which is the smallest relation such that:

- for all $q \in Q$, $(q, \epsilon, \epsilon, q) \in \Delta^*$, and
- if $(q, w, w', q') \in \Delta^*$ and $(q', a, b, q'') \in \Delta$, then $(q, w \cdot a, w' \cdot b, q'') \in \Delta^*$.

For $q, q' \in Q$, $w \in \Sigma^*$ and $w' \in \Gamma^*$, the notation $q \xrightarrow{-[w/w']\rightarrow} q'$ means that $(q, w, w', q') \in \Delta^*$. The *relation recognized by* T is:

$$[T] := \{(w, w') \mid w \in \Sigma^*, w' \in \Gamma^*, \exists q \in Q_F,\ q_\iota \xrightarrow{-[w/w']\rightarrow} q\}.$$

In other words, a pair (w, w') is in the relation recognized by T if there is an accepting execution of T that reads w and writes w'.

Definition 4 (Rational relations)**.** Let Σ and Γ be two alphabets. A binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Gamma^*$ is *rational* if there is a finite state transducer T such that $[T] = \rightsquigarrow$. A binary relation is *regular* if it can be recognized by a synchronous transducer. [2]

Rat and Reg are respectively the set of rational relations and the set of regular relations. We will often consider transducers that have the same alphabet Σ for input and output, and in this case we shall just talk about transducers over Σ and omit the output alphabet in the description of the transducer. The size of a transducer $T = (\Sigma, \Gamma, Q, \Delta, q_\iota, Q_F)$ is its number of transitions: $|T| = |\Delta|$.

Definition 5 (Recognizable relations)**.** Let Σ and Γ be two alphabets. A binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Gamma^*$ is *recognizable* if there are two finite families of regular languages $\mathcal{L}_1, \ldots, \mathcal{L}_n \subseteq \Sigma^*$ and $\mathcal{L}'_1, \ldots, \mathcal{L}'_n \subseteq \Gamma^*$ such that $\rightsquigarrow = \bigcup_{i=1}^{n} \mathcal{L}_i \times \mathcal{L}'_i$.

We note Rec the set of recognizable relations. The following inclusions are well known (Frougny and Sakarovitch, 1993): Rec $\subsetneq$ Reg $\subsetneq$ Rat.

[2] These are normally theorems, the real definitions are in terms of rational and regular subsets of the monoid $\Sigma^* \times \Gamma^*$. We choose to take these definitions instead as we are not interested in algebraic considerations.

3. Uniform strategies

In this section we first define the logic $\mathcal{L}_{\rightsquigarrow}$, which can be seen as a generalization of logics of knowledge and branching time, with arbitrary relations between nodes of trees and two different semantics for the "knowledge-like" quantifiers. We then define our notion of uniform strategies, where the uniformity constraints are expressed in this logic.

3.1. The logic $\mathcal{L}_{\rightsquigarrow}$

Because we believe the logical approach to be intuitive and well suited for automation, we choose to represent uniformity requirements by means of a logical language, that we call $\mathcal{L}_{\rightsquigarrow}$. In order to talk about the dynamics of strategy trees, $\mathcal{L}_{\rightsquigarrow}$ contains the classic operators of the full branching time logic CTL^*, and as for CTL^* a state formula is interpreted in a node of a labelled tree. The logic also contains quantifiers $\boxminus$ and $\boxbox$, that universally quantify over "related" nodes of labelled trees (representing paths in a game arena). The semantics of $\mathcal{L}_{\rightsquigarrow}$ is therefore parameterized by a binary relation $\rightsquigarrow$ between nodes of trees.

The difference between the two quantifiers $\boxminus$ and $\boxbox$ lies in the domain over which they range. Sometimes one wants to quantify only over those related paths that are inside the strategy under consideration. This is the role of $\boxminus$, called the *strict* quantifier. On the other hand it is sometimes natural to consider related paths that are not generated by the strategy, and that do not even start in the initial position of the game. This is achieved by $\boxbox$, called the *full* quantifier; its semantics is parameterized by a forest of labelled trees called the *universe*, and it quantifies over all $\rightsquigarrow$-related nodes in this universe. We now present the syntax and semantics of $\mathcal{L}_{\rightsquigarrow}$. In the remaining of this work, AP will always be a *finite* set of *atomic propositions*.

Note that in Maubert and Pinchinat (2014) the logical language basis was LTL. We choose here to extend it to the full branching time logic CTL^* as it is clearly more expressive (for example we can specify CTL^* winning conditions and capture module checking (Kupferman and Vardi, 1997)), it is also more natural as we see strategies as trees, and furthermore our decidability and complexity results are identical.

Syntax

The set of well-formed $\mathcal{L}_{\rightsquigarrow}$ formulas is given by the following grammar:

$$\begin{array}{rl} \text{State formulas:} & \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{A}\psi \mid \boxminus\varphi \mid \boxbox\varphi \\ \text{Path formulas:} & \psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\psi \mid \psi\mathbf{U}\psi, \end{array}$$

where $p \in AP$. Classically, we define $\texttt{true}$ as $p \vee \neg p$, $\texttt{false}$ as $\neg\texttt{true}$, and we define the Boolean conjunction $\wedge$ for both types of formulas. Also, for each path formula ψ, we write $\mathbf{E}\psi$ for the state formula $\neg\mathbf{A}\neg\psi$, $\mathbf{F}\psi$ for $\texttt{true}\mathbf{U}\psi$, $\mathbf{G}\psi$ for $\neg\mathbf{F}\neg\psi$ and for each state formula φ, we write $\diamondminus\varphi$ for $\neg\boxminus\neg\varphi$ and $\diamondbox\varphi$ for $\neg\boxbox\neg\varphi$.

For a formula $\varphi \in \mathcal{L}_{\rightsquigarrow}$, we define its *size* $|\varphi|$ as the number of symbols it contains, and $Sub(\varphi)$ as the set of subformulas of φ.

Semantics

Let Υ be a finite set of directions, and let $\Sigma = 2^{AP}$ be the set of possible valuations (for some finite set AP of atomic propositions). As we said, like for CTL^*, an $\mathcal{L}_{\rightsquigarrow}$ state formula (resp. path formula) is interpreted in a node (resp. branch) of a Σ-labelled Υ-tree, but the semantics is parameterized by, first, a binary relation $\rightsquigarrow$ between finite words over Σ, and second, a forest of Σ-labelled Υ-trees.

Let $\rightsquigarrow$ be a binary relation over Σ^* and $\mathcal{U}$ be a (Σ, Υ)-forest. For two nodes $x, y \in \mathcal{U}$ we let $x \rightsquigarrow y$ denote that $w(x) \rightsquigarrow w(y)$ (that is their node words are related by $\rightsquigarrow$).

Given a (Σ, Υ)-tree $t = (\tau, \ell)$, we define the semantics of $\mathcal{L}_{\rightsquigarrow}$ as follows, where $x \in \tau$ is a node and λ is a branch in τ. $t, x \models \varphi$ means that the $\mathcal{L}_{\rightsquigarrow}$ state formula φ holds at the node x of the labelled tree t, and $t, \lambda \models \psi$ means that the $\mathcal{L}_{\rightsquigarrow}$ path formula ψ holds on the branch λ of t. Rigorously we should write $\rightsquigarrow, \mathcal{U}, t, x \models \varphi$ but since the relation and the universe will always be clear from the context we may omit them.

$t, x \models p$ if $p \in \ell(x)$

$t, x \models \neg\varphi$ if $t, x \not\models \varphi$

$t, x \models \varphi_1 \vee \varphi_2$ if $t, x \models \varphi_1$ or $t, x \models \varphi_2$

$t, x \models \mathbf{A}\psi$ if for all $\lambda \in \mathit{Branches}(x)$, $t, \lambda \models \psi$

$t, x \models \boxminus\varphi$ if for all $y \in t$ such that $x \rightsquigarrow y$, $t, y \models \varphi$

$t, x \models \boxbox\varphi$ if for all $y \in \mathcal{U}$ such that $x \rightsquigarrow y$, $\mathcal{U}_y, y \models \varphi$ [3]

$t, \lambda \models \varphi$ if $t, \lambda[0] \models \varphi$

$t, \lambda \models \neg\psi$ if $t, \lambda \not\models \psi$

$t, \lambda \models \psi_1 \vee \psi_2$ if $t, \lambda \models \psi_1$ or $t, \lambda \models \psi_2$

$t, \lambda \models \mathbf{X}\psi$ if $t, \lambda^1 \models \psi$

$t, \lambda \models \psi_1 \mathbf{U} \psi_2$ if there exists $i \geq 0$ such that $t, \lambda^i \models \psi_2$ and for all $0 \leq j < i$, $t, \lambda^j \models \psi_1$

We shall use the notation $t \models \varphi$ for $t, r \models \varphi$, where r is the root of t. In particular, $t \models \mathbf{A}\psi$ if every branch of t that starts at the root satisfies ψ. Also, for a 2^{AP}-labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, a position $v \in V$ and a state formula $\varphi \in \mathsf{CTL}^*$, $v \models \varphi$ classically means $\mathit{Paths}_*(v) \models \varphi$.

3.2. Uniform strategies

Let $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ be a finite 2^{AP}-labelled game arena for some finite set AP. Let us note $\Sigma = 2^{AP}$, and let $\rightsquigarrow$ be a binary relation over Σ^*. Finally let φ be an $\mathcal{L}_{\rightsquigarrow}$ formula. The universe, *i.e.* the range of the full quantifier, is determined by the set V_ι of possible starting positions. We let $\mathcal{U}$ be the (Σ, V)-forest of all paths in the arena starting from a position in V_ι: $\mathcal{U} = \mathit{Paths}_*(V_\iota)$.

Definition 6 (Uniform strategies)**.** A strategy σ is $(\rightsquigarrow, \varphi)$-*uniform* if the strategy tree of σ satisfies φ, *i.e.* $t_\sigma \models \varphi$.

In the following section we give two examples of relevant problems that are naturally expressed by means of uniform strategies.

[3] Recall that $\mathcal{U}_y$ is the tree in $\mathcal{U}$ that contains y (see Definition 2).

4. Examples of uniform strategies

The first example describes how the strict quantifier enables us to capture the notion of observation-based strategy in games with imperfect information. The second example, relying on a notion of opacity in games, illustrates the usefulness of the full quantifier to characterize winning strategies when the winning condition involves the knowledge of the opponent.

4.1. Games with imperfect information

We first define in our setting a classic formalism of two-player turn-based games with imperfect information. We show that the standard notion of "uniform strategy" or "observation-based strategy" in these games, and hence the essence of imperfect information, is a quite simple instance of our general notion of uniform strategy. This is true no matter which assumptions are made on the player's observational power.

We consider the framework of two-player imperfect-information games studied for example in Reif (1984); Chatterjee et al. (2006); Berwanger and Doyen (2008). In these games, Player 1 only partially observes the positions of the game, such that some positions are indistinguishable to her, while Player 2 has perfect information (the asymmetry is due to the fact that we consider all possible outcomes of a strategy, and to the focus being on the existence of strategies for Player 1). Arenas are directed graphs with *actions* on edges. The game is played in *rounds*: in each round, if the position is a node v, Player 1 chooses an available action a, and Player 2 chooses a next position v' reachable from v through an a-edge.

We reformulate this framework in a way that fits our definition of game arenas by putting Player 1's actions inside the positions. Intuitively, in a position v, Player 1 choosing an action a is simulated by a move to position (v, a), after what Player 2 chooses a position v' reachable from v through a.

Formally, we define an *imperfect-information game arena* as a structure $\mathcal{G}^{imp} = (\mathcal{G}, \mathrm{obs})$ where $\mathcal{G} = (V, E, \{v_\iota\}, v_\iota, \mu)$ is a labelled perfect-information game arena where positions of the two players are of a different nature: there is a finite set Act of *actions* such that $V_2 = V_1 \times \mathrm{Act}$. So positions in V_1 are of the form v while positions in V_2 are of the form (v, a). The additional component $\mathrm{obs} : V_1 \to \mathrm{Obs}$ is an *observation function* mapping positions of Player 1 to a finite set Obs of *observations*. For a position $(v, a) \in V_2$, we let $(v, a).act := a$ denote the action it contains. The players must play in turns: $E \subseteq V_1 \times V_2 \cup V_2 \times V_1$. Also, because a move of Player 1 represents her choosing an action and not changing of position, we add the following requirement that $v\, E(v', a)$ implies $v = v'$; and because a game starts with Player 1 choosing an action, the initial position v_ι is in V_1. We add the classic requirement that the same actions must be available in indistinguishable positions: for all $v, v' \in V_1$, if $\mathrm{obs}(v) = \mathrm{obs}(v')$ then $v\, E(v, a)$ if and only if $v'\, E(v', a)$. This reflects the assumption that a player is able to distinguish positions with different alternatives of actions.

Also, we assume that AP contains a proposition p_a for each action $a \in \mathrm{Act}$ and a proposition p_o for each observation $o \in \mathrm{Obs}$, and we also assume that it contains a special proposition p_1 that marks positions of Player 1: $\mu(v) = \{p_1, p_{\mathrm{obs}(v)}\}$ for $v \in V_1$ and $\mu(v, a) = \{p_a\}$ for $(v, a) \in V_2$.

We now define the *observational equivalence relation* on finite plays $\approx$. This relation depends on Player 1's assumed capacities. Following a classic approach (*e.g.* Reif (1984);

Chatterjee et al. (2006); Berwanger and Doyen (2008)), we suppose that Player 1 has perfect recall, meaning that she remembers the whole sequence of observations she makes during a play. We also assume that she remembers her own actions. This leads to the following definition: we first extend the observation function to finite plays, by letting

$$\begin{aligned} \mathrm{obs}(v_0(v_0,a_1)v_1\dots(v_{n-1},a_n)) &:= \mathrm{obs}(v_0)a_1\mathrm{obs}(v_1)\dots a_n \text{ and} \\ \mathrm{obs}(v_0(v_0,a_1)v_1\dots(v_{n-1},a_n)v_n) &:= \mathrm{obs}(v_0)a_1\mathrm{obs}(v_1)\dots a_n\mathrm{obs}(v_n), \end{aligned}$$

and two plays are observationally equivalent if they have the same observation:

for $\rho,\rho' \in Plays_*$, $\rho \approx \rho'$ if $\mathrm{obs}(\rho) = \mathrm{obs}(\rho')$.

Observe that if $\rho \approx \rho'$ and ρ ends in V_1, then so does ρ'.

Definition 7. A strategy σ for Player 1 is *observation-based* if for all finite plays $\rho,\rho' \in \mathrm{Out}(\sigma)$ that end in V_1, if $\rho \approx \rho'$ then $\sigma(\rho).act = \sigma(\rho').act$.

Remark 1. In this definition we restrict attention to plays that follow the strategy. This is no limitation, because the definition of a strategy on finite plays that cannot be reached with this strategy does not affect its outcome. However, should we want to consider how strategies are defined on unreachable plays, strategies would no longer be represented as trees but as forests with infinitely many roots, which would pose problems, notably for the use of tree automata techniques.

Observation-based strategies as uniform strategies

First, let us define the $\mathcal{L}_{\rightsquigarrow}$ formula

$$\texttt{SameAct} := \mathbf{AG}(p_1 \to \bigvee_{a\in\mathrm{Act}} \boxminus\mathbf{EX}p_a),$$

which holds of a strategy of Player 1 if whenever it is Player 1's turn to play, some action a is chosen by the strategy in all related finite plays.

It remains to define the binary relation $\rightsquigarrow$ over $(2^{AP})^*$ that relates observationally equivalent plays. Note that the projection of the node word $w(\rho)$ on $2^{AP\setminus\{p_1\}}$ is exactly $\mathrm{obs}(\rho)$, thence two plays are observationally equivalent if, and only if, their node words are identical. Writing Id for the identity relation over $(2^{AP})^*$, we obtain the following result.

Proposition 1. *A strategy σ for Player 1 is observation-based if, and only if, it is $(Id, \texttt{SameAct})$-uniform.*

Note first that formula `SameAct` does not involve any full quantifier, so that the set of possible starting positions V_ι (only used to define the range of full quantifiers) is indifferent here. Now to understand why it is the strict quantifier that we need in `SameAct`, consider Figure 1, which represents the (first levels of) the unraveling of some imperfect-information arena with two actions, a and b. Circles represent positions of Player 1, squares positions of Player 2, and each position contains its propositional valuation, except for those propositions representing observations, rather represented with colours: Player 1 confuses positions v_1 and v_2, as well as positions v_3 and v_4.

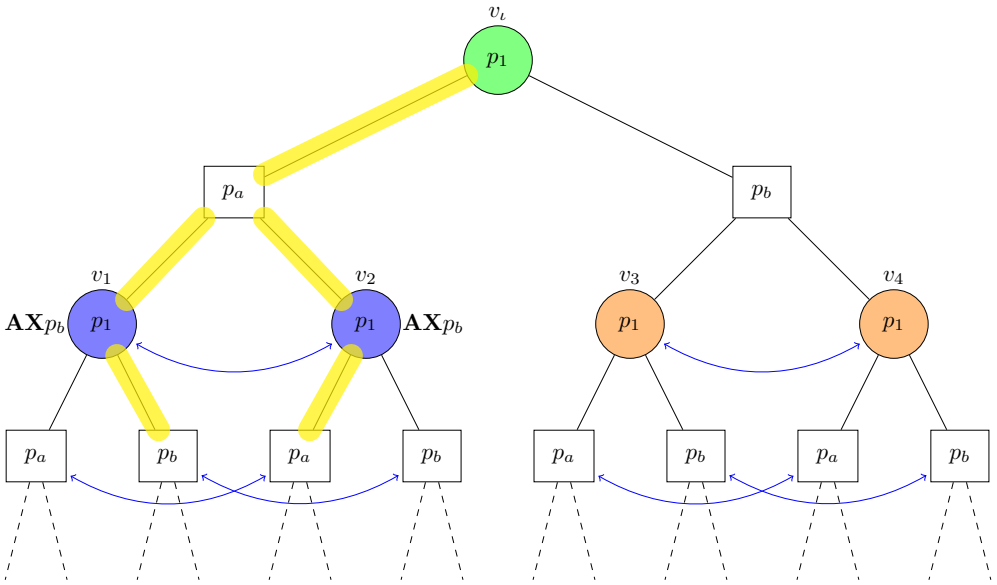

Figure 1: An example of strategy in an imperfect-information arena.

Finally, blue arrows represent the observational equivalence relation $\approx$ (self loops are omitted). Let x_1 (resp. x_2) be the node ending in v_1 (resp. v_2), and note that $x_1 \approx x_2$.

The yellow subtree is a strategy tree of Player 1 on which we evaluate `SameAct`. Since this strategy is not observation-based – it plays different actions in x_1 and x_2 – it should not verify `SameAct`. Therefore, because x_1 is a node of the strategy that verifies p_1, for each action $c \in \{a, b\}$, $\boxminus\mathbf{EX}p_c$ should not hold in this node. And indeed the only successor of x_1 *in the strategy tree* is labelled by p_a while the only successor of x_2 *in the strategy tree* is labelled by p_b. If we replaced in `SameAct` the strict quantifier by the full one, according to the semantics of $\boxminus$ nodes v_1 and v_2 would be seen as nodes of the full tree unfolding of the arena, and as such both have two successors, one with p_a and one with p_b, which makes sure that the formula holds. In fact, by using the full quantifier we would lose track of the strategy under evaluation, and the formula would be true of any strategy.

In this example we assume that Player 1 has perfect recall, and that she can observe her own actions, which determines the definition of $\approx$. It is easy to see that we may capture the same notion of observation-based strategy for any other assumptions on the player's capacities (like asynchronicity, imperfect recall...), by simply replacing the identity relation *Id* in Proposition 1 with an appropriate one.

4.2. Games with opacity condition

We have seen with the previous example that as soon as the desired property only concerns plays that follow the strategy, the appropriate quantifier is the strict one. *Games with opacity condition*, as studied in Maubert et al. (2011), give an example of a problem

that intrinsically requires the full quantifier. They also illustrate that the notion of uniform strategies enables us to express winning conditions with epistemic features.

Games with opacity condition are based on two-player imperfect-information arenas with a particular winning condition, called the *opacity condition*, that involves the knowledge of the player who has imperfect information. In such games, some positions are "secret", in the sense that they reveal some critical information. The player who has imperfect information (Attacker) aims at obtaining the certainty that the current position is a secret one, while her opponent (Defender) wants to prevent Attacker from obtaining this certainty.

Assume that some proposition $p_S \in AP$ is dedicated to the marking of secret positions. Let $\mathcal{G}^{imp} = (\mathcal{G} = (V, E, V_\iota, v_\iota, \mu), \mathrm{obs})$ be an imperfect-information arena (as defined in Section 4.1), with a distinguished set of positions $S \subseteq V_1$ that denotes the secret, and such that $\mu^{-1}(\{p_S\}) = S$ (positions labeled by p_S are exactly positions in S). We define the observational equivalence relation $\approx$ as in Section 4.1.

After a finite play ρ, the *knowledge* or *information set* $I(\rho)$ of Attacker is the set of positions that she considers possible according to the observation she has of ρ. Formally, for each play $\rho \in Plays_*$ such that $\mathrm{last}(\rho) \in V_1$, we let

$$I(\rho) := \{\mathrm{last}(\rho') \mid \rho' \in Plays_*, \rho \approx \rho'\}.$$

Here it is assumed that Attacker knows the initial position. For this reason, we let $V_\iota = \{v_\iota\}$, and in the definition of information sets, only equivalent *plays* are considered, instead of all equivalent paths in the arena. We could consider instead that Attacker has incomplete information and does not know the exact initial position; to do so we would just define V_ι and information sets differently.

An infinite play π is winning for Defender if Attacker never knows the secret. More formally, Defender wins play π if for every finite prefix ρ of π, not all the positions considered possible by Attacker after ρ are in the secret, *i.e.* $I(\rho) \not\subseteq S$.

We define the formula

$$\texttt{NeverKnowsS} := \mathbf{AG}\neg\boxbslash p_S,$$

which says that in all plays, Attacker never knows the secret. Again, letting Id be the identity relation over $(2^{AP})^*$, it can easily be shown that:

Proposition 2. *A strategy for Defender is winning iff it is* $(Id, \texttt{NeverKnowsS})$*-uniform.*

We give an intuitive explanation of why the full quantifier is the correct one here. According to the definition of Attacker's information sets, she considers that all observationally equivalent plays are possible, even those that are not induced by Defender's strategy. In other words, given a tree representing a strategy for Defender, evaluating whether Attacker knows the secret in some node requires to consider equivalent nodes outside the strategy tree, which is precisely what the full quantifier achieves. Notice that, intuitively, this definition of Attacker's knowledge implies that she "ignores" Defender's strategy. This assumption is natural in this precise setting, and corresponds to the fact that the imperfect information arises from the Attacker not seeing exactly which positions are chosen by Defender.

5. Strictly-uniform strategies

In this section, we investigate the problem of deciding the existence of a strictly-uniform strategy, *i.e.* a uniform strategy for some uniformity property that uses only strict quantifiers. Since the input of the problem must be finite, we choose to consider the rich class of rational relations, representable by finite state transducers.

Let $\mathcal{SL}_{\rightsquigarrow}$ be the language of *strict uniformity* constraints, *i.e.* the sublanguage of $\mathcal{L}_{\rightsquigarrow}$ where the full quantifier ($\boxbslash$) is not allowed. We consider the following decision problem[4]:

Definition 8.

$$\mathsf{SUS} := \left\{ (\mathcal{G}, T, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena, } T \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{SL}_{\rightsquigarrow}, \text{ and there exists a} \\ ([T], \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

We first show that SUS in undecidable. Then, in an attempt to understand the inherent difficulty of the problem, we introduce the notion of jumping tree automata that generalize alternating tree automata by allowing jumps between nodes of different branches in the input tree. The jumps "implement" the $\boxbslash$ quantifiers, and we prove that the satisfiability problem for $\mathcal{SL}_{\rightsquigarrow}$ reduces to the non-emptiness of jumping tree automata. It is then easy to see that SUS also reduces to the non-emptiness of these automata, by taking the product of an automaton that accepts trees representing strategies in arena $\mathcal{G}$ with the jumping automaton – equipped with relation $[T]$ – that accepts models of the formula Φ. We establish that for *recognizable* relations, jumping tree automata can be transformed into equivalent two-way tree automata, whose non-emptiness problem is decidable in EXPTIME (Vardi, 1998). This yields decidability and a tight 2-EXPTIME upper bound complexity for SUS with recognizable relations.

5.1. Undecidability for rational relations

It can be proven, by reduction of the Post Correspondence Problem, that SUS is undecidable, but we propose here the stronger result.

Theorem 1. *The strictly-uniform strategy problem is undecidable for the class of regular equivalence relations.*

Because the class of rational relations contains the class of regular equivalence relations, undecidability of SUS follows.

Corollary 1. SUS *is undecidable.*

The rest of this section is dedicated to the proof of Theorem 1.

We reduce the *distributed strategy problem for three-player games with safety objectives*, as addressed by Peterson et al. (2001); Berwanger and Kaiser (2010). We present the problem as stated in Berwanger and Kaiser (2010), in which two players with imperfect information (Player A and Player B) play against nature (the third player). Let Act_A (resp. Act_B) be a finite set of available actions for Player A (resp. Player B), and

[4] We use Φ for the input formula, and φ for subformulas.

Obs_A (resp. Obs_B) be a finite set of observations for Player A (resp. Player B). We assume that Act_A and Act_B are disjoint and we write $\mathrm{Act} = \mathrm{Act}_A \times \mathrm{Act}_B$.

A finite *three player imperfect-information game with safety objective* is a tuple $\mathcal{G}_3^{imp} = (V, E, v_\iota, o_A, o_B)$. V is a finite set of positions with a designated subset $Bad \subseteq V$ of "bad" positions that Player A and Player B should avoid. $E \subseteq V \times \mathrm{Act} \times V$ is a set of transitions and $o_X : V \to \mathrm{Obs}_X$ is an observation function ($X \in \{A, B\}$). In each round, Player X chooses an action $c_X \in \mathrm{Act}_X$, which gives an action profile $x = (c_A, c_B)$, and nature chooses a next position in $E(v, x) = \{v' \mid (v, x, v') \in E\}$. We suppose that all actions are allowed in every position: for all $v \in V$, $a \in \mathrm{Act}_A$ and $b \in \mathrm{Act}_B$, we have $E(v, (a, b)) \neq \emptyset$. The observation functions are extended to finite plays $\rho = v_0 x_0 v_1 \dots x_{n-1} v_n$ by letting $o_X(\rho) = o_X(v_0) o_X(v_1) \dots o_X(v_n)$. Note that actions are not observed.

A strategy for Player X is a partial mapping $\sigma_X : (V \cdot \mathrm{Act})^* \cdot V \to \mathrm{Act}_X$ that assigns an action to each finite play. It must be observation-based: for any finite plays ρ and ρ' such that $o_X(\rho) = o_X(\rho')$, $\sigma_X(\rho) = \sigma_X(\rho')$. A *distributed strategy* is a pair (σ_A, σ_B) of strategies for Player A and Player B. The outcome of a distributed strategy is the set of infinite plays that are both induced by σ_A and σ_B, and a distributed strategy is winning if no play in the outcome ever visits a position in Bad.

It is well known (Peterson et al., 2001; Berwanger and Kaiser, 2010) that the following problem is undecidable : *given a safety imperfect-information game, does there exist a winning distributed strategy?*

We explain how to reduce it to SUS. We fix an imperfect-information arena $\mathcal{G}_3^{imp} = (V, E, v_\iota, o_A, o_B)$ with observations Obs_A and Obs_B and bad states Bad, and we build a labelled game arena $\mathcal{G} = (V', E', V'_\iota, v'_\iota, \mu)$ in which Player 1 plays for both Player A and Player B, and Player 2 plays for nature. Figure 2 shows how each transition in $\mathcal{G}_3^{imp}$ is transformed into a widget in $\mathcal{G}$.

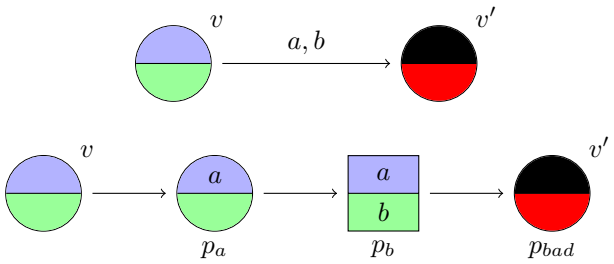

Figure 2: Encoding in $\mathcal{G}$ a transition $(v, (a, b), v')$ of $\mathcal{G}_3^{imp}$, with $v' \in Bad$. Colours represent the observations of Player A and Player B.

The set of positions $V' = V_1^A \uplus V_1^B \uplus V_2$ is split into three: in positions of $V_1^A = V$, Player 1 simulates moves of Player A, in positions of $V_1^B = V \times \mathrm{Act}_A$, Player 1 simulates moves of Player B, and in positions of $V_2 = V \times \mathrm{Act}_A \times \mathrm{Act}_B$, Player 2 simulates moves of nature. Hence for all v, v', a, b, we have $(v, (v, a)) \in E$, $((v, a), (v, a, b)) \in E$, and if $(v, (a, b), v') \in E$ then $((v, a, b), v') \in E$. For each action $c \in \mathrm{Act}_X$, p_c labels positions in which the last move was Player 1 simulating the choice of action c by Player X. In addition, "bad" positions are marked with proposition p_{bad}. Formally, we consider the set

of atomic propositions $\{p_c \mid c \in \mathrm{Act}_A \cup \mathrm{Act}_B\} \cup \{p_{bad}\}$, and we label the arena as follows: if $v \in Bad$, then $\mu(v) = \{p_{bad}\}$, $\mu(v,a) = \{p_a, p_{bad}\}$ and $\mu(v,a,b) = \{p_b, p_{bad}\}$; otherwise, $\mu(v) = \emptyset$, $\mu(v,a) = \{p_a\}$ and $\mu(v,a,b) = \{p_b\}$. We let $v'_\iota = v_\iota$, and because players are assumed to know the initial position, we let $V'_\iota = \{v'_\iota\}$. Finally the observation functions are defined on this new arena as follows: for a finite play $\rho = v_0(v_0,a_0)(v_0,a_0,b_0)v_1\ldots$, we note $o_X(\rho) = o_X(v_0)o_X(v_1)\ldots$

Clearly, since Player 1 plays for the coalition $\{A,B\}$, we expect each branch of her strategy to satisfy the following path formula:

$$\psi_{Safe} := \mathbf{G}\neg p_{bad}$$

We want to enforce that when Player 1 simulates a move of Player X, her choice is only based on Player X's observation. To do so, we define the symmetric and transitive relation $\rightsquigarrow$ over V'^* that relates two sequences of positions if they end in positions belonging to the same player, and are observationally equivalent for this player:

$$\rightsquigarrow := \left\{ (\rho,\rho') \;\middle|\; \begin{array}{l} \mathrm{last}(\rho) \in V_1^A \text{ and } \mathrm{last}(\rho') \in V_1^A \text{ and } o_A(\rho) = o_A(\rho'), \text{ or} \\ \mathrm{last}(\rho) \in V_1^B \text{ and } \mathrm{last}(\rho') \in V_1^B \text{ and } o_B(\rho) = o_B(\rho') \end{array} \right\}$$

Note that for the sake of clarity we defined $\rightsquigarrow$ on V^* instead of $(2^{AP})^*$, but by labelling each position v with an atomic proposition p_v and working with the alphabet $\Sigma = \{\{p_v\} \cup \mu(v) \mid v \in V\}$, one can rephrase relation $\rightsquigarrow$ as a binary relation over Σ^*.

The following path formula states that whenever Player 1 simulates a move of Player X, she chooses the same action in all plays observationally equivalent for Player X:

$$\psi_{Obs} := \mathbf{G} \bigwedge_{c \in \mathrm{Act}_A \cup \mathrm{Act}_B} \mathbf{X}p_c \to \boxminus\mathbf{E}\mathbf{X}p_c$$

We get the following reduction:

Lemma 1. *There is a winning distributed strategy in $\mathcal{G}_3^{imp}$ if, and only if, there is a $(\rightsquigarrow, \mathbf{A}(\psi_{Obs} \wedge \psi_{Safe}))$-uniform strategy for Player 1 in $\mathcal{G}$.*

We show that $\rightsquigarrow$ is regular. Consider the synchronous transducer $T_{A,B}$ of Figure 3. State q_ι is the initial state (ingoing arrow), q_1^A and q_2^B are final states (doubly circled). Transducer $T_{A,B}$ works as follows: before reading a word w, the transducer guesses whether we are interested in Player A or Player B's observation. In the first case it goes to the left, reads w and writes a word w' observationally equivalent for Player A (remember that actions are not observed). The pair (w,w') is accepted if w (and w') indeed ends in a position where it is Player A's turn to play. The second case is symmetric. Note that $\rightsquigarrow$ is not reflexive – words ending in V_2 are related to no word – but its reflexive closure $\sim$ is also regular (plug in $T_{A,B}$ the synchronous transducer for the identity relation). Lemma 1 would also hold for $\sim$, which concludes the proof of Theorem 1.

5.2. Intermezzo: jumping tree automata

Let Υ be a finite set of directions. We recall the notions of alternating tree automata and two-way tree automata on Υ-trees, and we define *jumping tree automata (JTA)*, all with parity acceptance condition. For an introduction to the theory of automata on infinite trees see Thomas (1990) or Löding (2014).

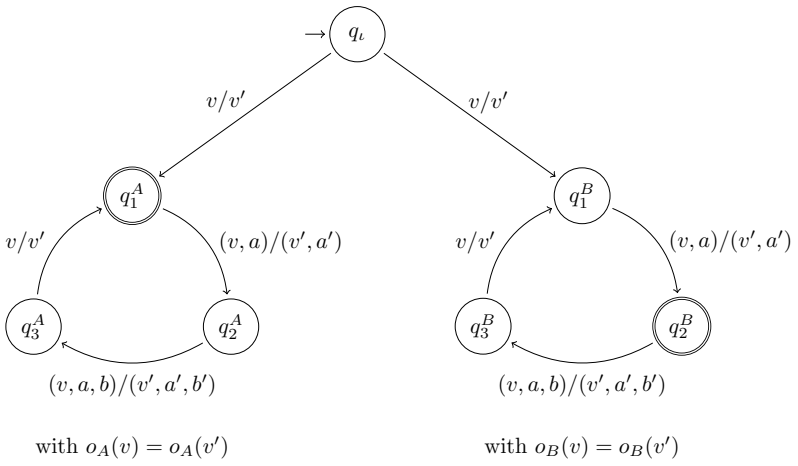

Figure 3: The synchronous transducer $T_{A,B}$.

For a set X, $\mathbb{B}^+(X)$ is the set of positive boolean formulas over X, *i.e.* formulas built with elements of X as atomic propositions and using only connectives $\vee$ and $\wedge$. As usual, we also allow for formulas `true` and `false`, and $\wedge$ has precedence over $\vee$. Elements of $\mathbb{B}^+(X)$ are denoted by $\alpha, \beta \ldots$ Let $Dir \subseteq \Upsilon \cup \{\epsilon, \uparrow, \diamondsuit, \boxminus\}$ be a set of *transition directions*. A Dir-automaton is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ where Σ is a finite alphabet, Q a finite set of states, $q_\iota \in Q$ an initial state, $C : Q \to \mathbb{N}$ a colouring function, and $\delta : Q \times \Sigma \to \mathbb{B}^+(Dir \times Q)$ a transition function. If Dir contains $\diamondsuit$ or $\boxminus$ then we additionally require the automaton to be equipped with a binary relation $\rightsquigarrow$ over Σ^*. We note $Dir_\uparrow = \Upsilon \cup \{\epsilon, \uparrow\}$ and $Dir_\rightsquigarrow = \Upsilon \cup \{\diamondsuit, \boxminus\}$. Finally, we define the size of a Dir automaton $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ as the sum of the sizes of formulas in its transition function: $|\mathcal{A}| = \sum_{q \in Q} \sum_{a \in \Sigma} |\delta(q, a)|$.

Definition 9. Υ-automata are *alternating tree automata*, $Dir_\uparrow$-automata are *two-way tree automata*, and $Dir_\rightsquigarrow$-automata are *jumping tree automata (JTA)*.

Most of the time in this section, identifying the precise child of the current node in which an automaton must send a copy of itself is unnecessary, and specifying instead that it must send a copy in at least one child or in all children is enough. Therefore, for the sake of readability, we replace the set of directions Υ with the set of abstract directions $\{\Diamond, \Box\}$ as in alternating automata on graphs (Bojanczyk, 2002; Piterman and Vardi, 2004). $[\Diamond, q]$ can be seen as a macro for $\bigvee_{d \in \Upsilon}[d, q]$, and similarly $[\Box, q]$ stands for $\bigwedge_{d \in \Upsilon}[d, q]$. Note however that the results we establish in this section also hold in the more general case where concrete directions are allowed.

Following a classic approach (Muller and Schupp, 1987; Löding, 2014), acceptance of a (Σ, Υ)-tree $t = (\tau, \ell)$ in a designated node $x_\iota \in \tau$ by a Dir-automaton $\mathcal{A}$ is defined on a two-player game between Eve (the proponent) and Adam (the opponent). Let $t = (\tau, \ell)$ be a (Σ, Υ)-tree, $x_\iota \in \tau$, and $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ be a Dir-automaton for some set of transition directions Dir. We define the parity game $\mathcal{G}^{x_\iota}_{\mathcal{A},t} = (V, E, v_\iota, C')$: the set of positions is $V = \tau \times Q \times \mathbb{B}^+(Dir \times Q)$, the initial position is $(x_\iota, q_\iota, \delta(q_\iota, \ell(x_\iota)))$, and a

position (x, q, α) belongs to Eve if α is of the form $\alpha_1 \vee \alpha_2$, $[\Diamond, q']$ or $[\diamondsuit, q']$; otherwise it belongs to Adam. Moves in $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$ are defined by Rules (1a)-(1e); for clarity we shall abuse notations, and for a node x of t and a state q of $\mathcal{A}$ we write $\delta(q, x)$ for $\delta(q, \ell(x))$.

$$(x, q, \alpha_1 \dagger \alpha_2) \to (x, q, \alpha_i) \quad \text{where } \dagger \in \{\vee, \wedge\} \text{ and } i \in \{1, 2\} \tag{1a}$$
$$(x, q, [\bigcirc, q']) \to (y, q', \delta(q', y)) \quad \text{where } \bigcirc \in \{\Diamond, \Box\} \text{ and } y \text{ is a child of } x \tag{1b}$$
$$(x, q, [\epsilon, q']) \to (x, q', \delta(q', x)) \tag{1c}$$
$$(x, q, [\uparrow, q']) \to (y, q', \delta(q', y)) \quad \text{where } y \text{ is } x\text{'s parent} \tag{1d}$$
$$(x, q, [\ominus, q']) \to (y, q', \delta(q', y)) \quad \text{where } \ominus \in \{\diamondsuit, \boxminus\} \text{ and } x \rightsquigarrow y \tag{1e}$$

Positions of the form $(x, q, \texttt{true})$ and $(x, q, \texttt{false})$ are sink positions, winning for Eve and Adam respectively. Positions of the form $(r, q, [\uparrow, q'])$ where r is the root are also sink positions as the root of a tree has no parent; they are wining for Adam. The colouring is inherited from the one of the automaton: $C'(x, q, \alpha) = C(q)$, except for sink positions, which are assigned an even (resp. odd) priority if they are winning for Eve (resp. Adam).

Most of the time the starting node x_ι will be the root r of the tree, and in this case we simply write $\mathcal{G}_{\mathcal{A},t}$ instead of $\mathcal{G}^r_{\mathcal{A},t}$. A tree t is *accepted* by $\mathcal{A}$ if Eve has a winning strategy in $\mathcal{G}_{\mathcal{A},t}$, and we denote by $\mathcal{L}(\mathcal{A})$ the set of trees accepted by $\mathcal{A}$.

We first prove that, just like alternating automata, the class of JTA is closed by complementation. To this aim, for a formula $\alpha \in \mathbb{B}^+(Dir_\rightsquigarrow \times Q)$ we define its *dualization* $\widetilde{\alpha}$ by induction as follows: $\widetilde{\texttt{true}} = \texttt{false}$, $\widetilde{\texttt{false}} = \texttt{true}$, $\widetilde{\alpha \vee \beta} = \widetilde{\alpha} \wedge \widetilde{\beta}$, $\widetilde{\alpha \wedge \beta} = \widetilde{\alpha} \vee \widetilde{\beta}$, $\widetilde{[\Diamond, q]} = [\Box, q]$, $\widetilde{[\Box, q]} = [\Diamond, q]$, and, as expected, $\widetilde{[\diamondsuit, q]} = [\boxminus, q]$ and $\widetilde{[\boxminus, q]} = [\diamondsuit, q]$.

Definition 10. Let $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ be a jumping tree automaton. We define *the complement* of $\mathcal{A}$ by $\widetilde{\mathcal{A}} = (\Sigma, Q, \widetilde{\delta}, q_\iota, \widetilde{C})$, where $\widetilde{C}(q) = C(q) + 1$, and $\widetilde{\delta}(q, a) = \widetilde{\delta(q, a)}$.

Lemma 2. *Eve wins $\mathcal{G}^{x_\iota}_{\widetilde{\mathcal{A}},t}$ if, and only if, Eve loses $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$.*

Proof. The arenas of both games are isomorphic, and if a position belongs to Eve in $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$ then its counterpart in $\mathcal{G}^{x_\iota}_{\widetilde{\mathcal{A}},t}$ belongs to Adam, and vice versa. Also, a play is winning for a player in one game if and only if its counterpart in the other game is winning for the opponent. From this we have that a winning strategy for a player in one game gives a winning strategy for its opponent in the other, and because parity games are determined (Zielonka, 1998), the result follows. □

We now establish that JTA capture $\mathcal{SL}_\rightsquigarrow$.

Proposition 3. *Let φ be an $\mathcal{SL}_\rightsquigarrow$ formula, and let $\rightsquigarrow$ be a binary relation over $(2^{AP})^*$. There exists a jumping tree automaton $\mathcal{A}_\varphi$ equipped with $\rightsquigarrow$ and of size $2^{O(|\varphi|)}$ such that $t \in \mathcal{L}(\mathcal{A}_\varphi)$ if, and only if, $t \models \varphi$.*

Proof. The construction is a simple adaptation of Kupferman et al. (2000), that inductively builds an alternating tree automaton for a CTL^* formula. We only give the inductive case $\varphi = \boxminus\varphi'$.

If $\varphi = \boxminus\varphi'$, with φ' a state formula, let $\mathcal{A}_{\varphi'} = (\Sigma, Q, \delta', q'_\iota, C)$ be the jumping automaton associated to φ'. We define $\mathcal{A}_\varphi = (\Sigma, Q \cup \{q_\iota\}, \delta, q_\iota, C)$, where $q_\iota \notin Q$, $\delta(q, a) = \delta'(q, a)$ if $q \in Q$, and $\delta(q_\iota, a) = [\boxminus, q'_\iota]$. □

We prove that JTA are adequate machines for the decision problem SUS:

Proposition 4. *Let $(\mathcal{G}, T, \Phi)$ be an instance of* SUS*. There is a jumping tree automaton $\mathcal{A}$ equipped with $[T]$ such that σ is a $([T], \Phi)$-uniform strategy if, and only if, $t_\sigma \in \mathcal{L}(\mathcal{A})$. Moreover, $\mathcal{A}$ can be chosen of size $|\mathcal{A}| = |\mathcal{G}|^2 + 2^{O(|\Phi|)}$.*

Proof. One builds from $\mathcal{G}$ a nondeterministic tree automaton $\mathcal{A}_\mathcal{G}$ (using concrete transition directions $Dir = V$) that accepts the set of strategy trees for Player 1 in $\mathcal{G}$. $\mathcal{A}_\mathcal{G}$ is of size at most $|\mathcal{G}|^2$. Then, by Proposition 3, one can build a JTA $\mathcal{A}_\varphi$ equipped with $[T]$ that accepts the 2^{AP}-labelled V-trees that verify Φ. This automaton is of size $|\mathcal{A}_\Phi| = 2^{O(|\Phi|)}$. Because JTA are trivially closed by language intersection, one can build in time linear in the sizes of $\mathcal{A}_\mathcal{G}$ and $\mathcal{A}_\Phi$ a JTA that accepts precisely the strategy trees that verify Φ. □

The following is a direct consequence of Theorem 1 and Proposition 4.

Corollary 2. *The emptiness problem for jumping tree automata with regular equivalence relations is undecidable.*

5.3. The special case of recognizable relations

We have seen that SUS is undecidable for regular relations (Theorem 1). We establish that when restricted to recognizable relations the problem becomes decidable and, more precisely, 2-EXPTIME-complete. To achieve this result we prove that JTA equipped with recognizable relations can be effectively transformed into equivalent two-way tree automata of linear size; this implies that the emptiness problem for JTA with recognizable relations is decidable in exponential time.

We gave in Section 2.3 the definition of recognizable relations, we now give a useful characterization in terms of regular finite word languages. While in general transducers that recognize rational or regular binary relations have to parse the two input words in parallel, a recognizable relation can be recognized by a finite word automaton that starts by reading entirely the first word (or its mirror), and then the second one, before deciding whether these two words are related or not.

Proposition 5 (Carton et al. (2006))**.** *A relation $\rightsquigarrow \subseteq \Sigma^* \times \Gamma^*$ is recognizable if, and only if, the language $\{\overline{u}\#v \mid u \rightsquigarrow v\}$ is regular (accepted by a finite word automaton), where $\# \notin \Sigma \cup \Gamma$ is a fresh symbol.*

Given a recognizable relation $\rightsquigarrow$, we write $\mathcal{B}_\rightsquigarrow = (\Sigma \cup \{\#\}, Q_\rightsquigarrow, \delta_\rightsquigarrow, s_\iota, F_\rightsquigarrow)$ for the minimal deterministic word automaton of the language $\{\overline{u}\#v \mid u \rightsquigarrow v\}$ (Proposition 5), with standard interpretation of the components of $\mathcal{B}_\rightsquigarrow$.

Remark 2. In games with imperfect information, a strategy is called *information-based* if it assigns the same move to finite plays that share the same information set. This is a stronger constraint than being observation-based as two plays can have different observations and still yield the same information set. However it is well known (Berwanger et al., 2010) that in two-player turn-based games with imperfect information equipped, for example, with parity winning conditions, there is a winning observation-based strategy for a player if, and only if, she has a winning information-based strategy, which is usually proved by reduction to a powerset perfect-information game. This result implies that in these games, looking for an information-based strategy is sufficient; interestingly, the binary relation involved to represent information-based strategies as uniform strategies is recognizable, and the powerset arena gives a finite word automaton that recognizes it.

Proposition 6. *If $\mathcal{A}$ is a jumping tree automaton equipped with a recognizable relation $\rightsquigarrow$, then there is a two-way tree automaton $\widehat{\mathcal{A}}$ of size $O(|\mathcal{A}|\cdot|\mathcal{B}_\rightsquigarrow|)$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.*

Proof sketch. When JTA $\mathcal{A}$ goes down along a branch of a tree, $\widehat{\mathcal{A}}$ behaves likewise. The critical points are the jump instructions of $\mathcal{A}$, say in a node x of the tree. At this point, $\widehat{\mathcal{A}}$ stops behaving like $\mathcal{A}$ and enters a *jump mode* that simulates this jump: $\widehat{\mathcal{A}}$ triggers automaton $\mathcal{B}_\rightsquigarrow$ and goes up to the root while running $\mathcal{B}_\rightsquigarrow$ on the reversed branch. When reaching the root, $\mathcal{B}_\rightsquigarrow$ has read $\overline{w(x)}$, the mirror of the node word of x, and $\widehat{\mathcal{A}}$ feeds $\mathcal{B}_\rightsquigarrow$ with the # symbol. Then $\widehat{\mathcal{A}}$ goes down along some or all (depending on the jump: respectively existential or universal) branch(es) of the tree while still running $\mathcal{B}_\rightsquigarrow$. Each time $\mathcal{B}_\rightsquigarrow$ reaches a final state in a node y, it has read $\overline{w(x)}\#w(y)$, and by Proposition 5 it means that $x \rightsquigarrow y$; the automaton then (existentially or universally) chooses to continue or to exit the jump mode, in which case $\widehat{\mathcal{A}}$ resumes the simulation of $\mathcal{A}$. □

From Proposition 6 follows directly

Corollary 3. *The non-emptiness problem for jumping tree automata with recognizable relations is decidable in time exponential in the size of the jumping automaton and of the word automaton recognizing the relation.*

Proof. This is a direct consequence of Proposition 6 along with the EXPTIME complexity of the non-emptiness problem for two-way tree automata Vardi (1998). □

Finally, combining Proposition 4 with Corollary 3 proves that the strictly uniform problem for recognizable relations is decidable. We define this problem formally.

$$\mathsf{SUS}_{\mathrm{Rec}} := \left\{ (\mathcal{G}, \mathcal{B}_\rightsquigarrow, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena, } \Phi \in \mathcal{SL}_\rightsquigarrow, \\ \rightsquigarrow \text{ is a recognizable relation over } 2^{AP}, \text{ and there exists a} \\ (\rightsquigarrow, \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

The size of an instance $(\mathcal{G}, \mathcal{B}_\rightsquigarrow, \Phi)$ of $\mathsf{SUS}_{\mathrm{Rec}}$ is the sum of the sizes of its components, plus the number of atomic propositions used: $|(\mathcal{G}, \mathcal{B}_\rightsquigarrow, \Phi)| = |\mathcal{G}| + |\mathcal{B}_\rightsquigarrow| + |\Phi| + |AP|$.

Theorem 2. $\mathsf{SUS}_{\mathrm{Rec}}$ *is* 2-EXPTIME*-complete.*

Proof sketch. Let $(\mathcal{G}, \mathcal{B}_\rightsquigarrow, \Phi)$ be an instance of $\mathsf{SUS}_{\mathrm{Rec}}$. By Proposition 3, there is a JTA $\mathcal{A}$ of size $2^{O(|\Phi|)}$), equipped with relation $\rightsquigarrow$, whose language is the set of trees that verify Φ. By Proposition 6, one can build an equivalent two-way tree automaton $\widehat{\mathcal{A}}$ of size $O(|\mathcal{A}|\cdot|\mathcal{B}_\rightsquigarrow|)$. By Vardi (1998), there is an equivalent nondeterministic tree automaton $\mathcal{A}^n$ with $2^{\mathrm{poly}(|\widehat{\mathcal{A}}|)} = 2^{\mathrm{poly}(|\mathcal{A}|\cdot|\mathcal{B}_\rightsquigarrow|)} = 2^{\mathrm{poly}(2^{O(|\Phi|)}\cdot|\mathcal{B}_\rightsquigarrow|)}$ states and an acceptance condition of size $O(|\widehat{\mathcal{A}}|) = O(|\mathcal{A}| \cdot |\mathcal{B}_\rightsquigarrow|) = O(2^{O(|\Phi|)} \cdot |\mathcal{B}_\rightsquigarrow|)$. From $\mathcal{G}$, one builds a nondeterministic tree automaton $\mathcal{A}_\mathcal{G}$ with $O(|\mathcal{G}|)$ states that accepts strategy trees for Player 1 in $\mathcal{G}$. The final step is to test the emptiness of the product $\mathcal{A}^n \times \mathcal{A}_\mathcal{G}$. Because checking for emptiness of a nondeterministic parity tree automaton over alphabet Σ with n states and c colors on d-ary trees can be done in time $(|\Sigma|\cdot n^{O(d)})^{O(c)}$, and because here $\Sigma = 2^{AP}$, we obtain a decision procedure that runs in time $(|\mathcal{G}|^d \cdot 2^{|AP|+d\cdot|\mathcal{B}_\rightsquigarrow|})^{2^{O(|\Phi|)}}$, where d is the maximum branching degree in $\mathcal{G}$.

The 2-EXPTIME-hardness of $\mathsf{SUS}_{\mathrm{Rec}}$ is inherited from the 2-EXPTIME-completeness of solving CTL^* games (Kupferman and Vardi, 1997). □

Synthesis of strictly-uniform strategies. We have seen that the problem $\mathsf{SUS}_{\mathrm{Rec}}$ is decidable. In fact the associated synthesis problem can be solved with the same time complexity. Indeed, a uniform strategy (if any) can be synthesized when checking the non-emptiness of the JTA that accepts uniform strategy trees: first, build the linear size equivalent two-way tree automaton of Proposition 6, then turn it into an equivalent non-deterministic parity tree automaton with exponentially many states but a linear number of priorities. Classic decision procedures for the emptiness of parity tree automata can be adapted to provide, in case the language is not empty, a regular tree in the language (Löding, 2014). This regular tree represents a uniform strategy, and it can be seen as a finite state automaton that implements this uniform strategy. The number of states represents the memory needed by the strategy, and it is the same as in the nondeterministic tree automaton. To conclude, a memory polynomial in $|\mathcal{G}|$, exponential in $|\mathcal{B}_{\sim}|$ and doubly exponential in $|\Phi|$ is enough for strictly-uniform strategies with recognizable relations.

6. Fully-uniform strategies

We turn to the dual case where only full quantifiers are allowed. We consider the decision problem of the existence of a fully-uniform strategy, but like in the decidable case for strictly-uniform strategies, our decision procedures can be adapted with no additional cost to effectively provide a uniform strategy whenever there exists one. Once again we consider the class of rational relations.

We established in Maubert and Pinchinat (2014), with LTL as base language, that the existence of a fully-uniform strategy is decidable for the class of rational relations, and we gave a non-elementary decision procedure. More precisely, we proved that if the maximum nesting of $\boxbslash$ quantifiers in the uniformity properties is k then the problem is in k-Exptime – 2-Exptime for $k \leq 2$. The first contribution of this section is to generalize this result to uniformity properties with linear *and* branching time connectives, and to establish the matching lower-bounds. As a second contribution we exhibit a rich subclass of rational relations, called K45NM, for which the problem has the same complexity as the strategy problem for CTL^* games, *i.e.* 2-Exptime-complete.

We first formally define the decision problems considered and state our results. Next we introduce the notion of *information set automata*, which is central in our decision procedures. Then we adapt the proof from Maubert and Pinchinat (2014) for the k-Exptime upper bounds, and make it more modular by using information set automata. Again relying on these automata we establish that the fully-uniform strategy problem for relations in K45NM is in 2-Exptime, and we finish with the matching lower bounds.

6.1. Main results

First, let $\mathcal{FL}_{\sim}$ denote the language of *full uniformity* constraints, *i.e.* the sublanguage of $\mathcal{L}_{\sim}$ that does not allow strict quantifiers ($\boxbslash$) but only full ones ($\boxbslash$).

The $\boxbslash$*-depth* of a $\mathcal{FL}_{\sim}$ formula φ, written $d(\varphi)$, is the maximum number of nested $\boxbslash$ quantifiers in φ, defined inductively as follows:

$$d(p) = 0 \qquad d(\neg\varphi) = d(\varphi) \quad d(\varphi \vee \varphi') = \max(d(\varphi), d(\varphi')) \quad d(\boxbslash\varphi) = 1 + d(\varphi)$$
$$d(\mathbf{A}\psi) = d(\psi) \quad d(\mathbf{X}\psi) = d(\psi) \quad d(\psi \mathbf{U} \psi') = \max(d(\psi), d(\psi'))$$

For $k \in \mathbb{N}$, we let $\mathcal{FL}^k_{\rightsquigarrow} := \{\varphi \in \mathcal{FL}_{\rightsquigarrow} \mid d(\varphi) \leq k\}$ and we define the decision problem FUS_k.[5]

Definition 11. For each $k \in \mathbb{N}$, we let

$$\mathsf{FUS}_k := \left\{ (\mathcal{G}, T, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena, } T \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{FL}^k_{\rightsquigarrow}, \text{ and there exists a} \\ ([T], \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

The *fully-uniform strategy problem* is $\mathsf{FUS} := \bigcup_{k \in \mathbb{N}} \mathsf{FUS}_k$. We let the size of an instance be the sum of the sizes of its components, plus the number of atomic propositions used: $|(\mathcal{G}, T, \Phi)| = |\mathcal{G}| + |T| + |\Phi| + |AP|$.

Theorem 3. FUS_k *is* k-EXPTIME*-complete if* $k \geq 2$*, otherwise it is* 2-EXPTIME*-complete.*

Corollary 4. *The fully-uniform strategy problem is nonelementary.*

We now turn to our second result. First, recall that K45 is the set of transitive and Euclidean relations[6], which are "almost" equivalence relations, and are relevant for example in the context of belief revision and modelling plausibility (Fagin et al., 1995).

We define in our setting the classic notion of *No Miracles* (Pacuit and van Benthem, 2006):

Definition 12. A binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$ satisfies the *No Miracles* property if for all $u, v, w \in \Sigma^*$, $u \rightsquigarrow v$ implies $u \cdot w \rightsquigarrow v \cdot w$.

We note K45NM for the set of transitive, Euclidean and No Miracles rational relations, and we call $\mathsf{FUS}_{\mathrm{K45NM}}$ the restriction of FUS to K45NM relations, *i.e.* for an input $(\mathcal{G}, T, \Phi)$ of $\mathsf{FUS}_{\mathrm{K45NM}}$, we assume that $[T]$ is transitive, Euclidean and verifies No Miracles.

Theorem 4. $\mathsf{FUS}_{\mathrm{K45NM}}$ *is* 2-EXPTIME*-complete.*

This result is of interest as most relations used to represent uncertainty fall into this class, like synchronous or asynchronous perfect recall relations, and in general all equivalence relations induced by alphabetic morphisms.

We now introduce information set automata, and then we use them to prove the upper bounds of Theorem 3 and Theorem 4. We prove the lower bounds at the end of the section.

6.2. Information set automaton

The notion of information set is classic in games with imperfect information. After a finite play, a player's information set is the set of positions considered possible by this player according to what she has observed so far. We introduce a general notion of information set: informally, given a binary relation over finite words, the information set of a word is the set of all last letters of related words. We then describe how, when given a transducer that recognizes a binary relation, one can build a deterministic word

[5] We use Φ for the input formula, and φ for subformulas.

[6] A relation $\rightsquigarrow$ is Euclidean if $u \rightsquigarrow v$ and $u \rightsquigarrow w$ implies $v \rightsquigarrow w$.

automaton that computes the information set of its input word. We also introduce a bisimulation relation on states of this automaton, called the *information set bisimulation*, and we consider the quotient automaton. We prove that for relations in K45NM, two related plays take the quotient automaton to the same state. This result is crucial for the upper bound of Theorem 4.

Definition 13. Let Σ be an alphabet, and $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$ be a binary relation over Σ^*. For $w \in \Sigma^*$, the *information set* after the word w is:

$$I(w) = \{a \in \Sigma \mid w \rightsquigarrow w' \cdot a \text{ for some } w' \in \Sigma^*\}$$

For the rest of Section 6.2 we fix a transducer $T = (\Sigma, Q, \Delta, Q_\iota, Q_F)$ over alphabet Σ. We describe a powerset construction to transform T into a deterministic automaton that computes information sets for $[T]$. Transducers cannot be determinized in general, but because we only aim at computing information sets we can afford to forget the output tape, except for the last letter. This allows us to obtain a deterministic automaton $\mathcal{A}^T$ that we call the *information set automaton*.

Computing information sets in the classic framework of imperfect-information games (see Section 4.1) is simple: the player only needs to remember the current information set, and update it with each newly observed position. In the case of rational relations in general, information sets cannot be inferred from the new position and the previous information set only, and more information needs to be stored in the states of the information set automaton.

While reading a word w, we remember two things: first, the set of states q that the transducer may have reached by nondeterministic runs on input w, and second, for each such state q, the set $Last(q)$ of all last letters of output words in a run on input w that ends in q. Therefore, states of $\mathcal{A}^T$ are pairs of the form $(S, Last)$, with $S \subseteq Q$ and $Last : Q \to 2^\Sigma$ (recall that Q is the transducer's set of states).

Definition 14 (Information set automaton)**.** The deterministic *information set automaton* for $T = (\Sigma, Q, \Delta, q_\iota, Q_F)$ is $\mathcal{A}^T = (\Sigma, A, \delta, \alpha_\iota)$, where

- $A = 2^Q \times (Q \to 2^\Sigma)$
- $\alpha_\iota = (S_\iota, Last_\iota)$, with
  - $S_\iota = \{q \mid \exists w \in \Sigma^*,\ q_\iota \text{ -}[\epsilon/w]\!\to q\}$ and
  - $Last_\iota(q) = \{a \mid \exists w \in \Sigma^*, q_\iota \text{ -}[\epsilon/w \cdot a]\!\to q\}$
- $\delta((S, Last), a) = (S', Last')$, with
  - $S' = \{q' \mid \exists q \in S, \exists w \in \Sigma^*,\ q \text{ -}[a/w]\!\to q'\}$ and
  - $Last'(q') = \{a' \mid \exists q \in S,\ \exists w \in \Sigma^*, q \text{ -}[a/w \cdot a']\!\to q'$, or $q \text{ -}[a/\epsilon]\!\to q'$ and $a' \in Last(q)\}$

First, observe the initial state $(S_\iota, Last_\iota)$: S_ι is the set of states that can be reached internally (*i.e.* by reading nothing) from q_ι, and for each state $q \in S_\iota$, $Last_\iota(q)$ is the set of letters found at the end of the output tape in runs that internally reach q.

We now detail how for a current state $(S, Last)$ and a new input letter a, we build the successor state $\delta((S, Last), a) = (S', Last')$. First, S' is made of all the states q' that can be reached from a state $q \in S$ with a transition of the form $q \text{ -}[a/w]\!\to q'$ with $w \in \Sigma^*$.

If $w = w' \cdot b$, then b is added to $Last'(q')$, otherwise $w = \epsilon$ and each $b \in Last(q)$ can still be the last letter on the output tape after reading a and reaching q', therefore we let $b \in Last'(q')$. Note that by definition $Last'(q') = \emptyset$ for each $q' \notin S'$.

Since $\mathcal{A}^T$ is complete, $\delta(\alpha_\iota, w)$ is defined (in the classic way) for all $w \in \Sigma^*$. We define the size of an information set automaton as its number of transitions: $|\mathcal{A}^T| = |\delta|$. Note that because $\mathcal{A}^T$ is complete and deterministic, $|\delta| = |A| \cdot |\Sigma|$.

Lemma 3. *$\mathcal{A}^T$ is of size $2^{O(|T|\cdot|\Sigma|)}$ and can be computed in time $2^{O(|T|\cdot|\Sigma|)}$.*

Finally, for every state $\alpha = (S, Last)$ of $\mathcal{A}^T$, we define:

$$\alpha.I := \bigcup_{q \in S \cap Q_F} Last(q)$$

We prove that the components S and $Last$ actually capture what they are meant to:

Lemma 4. *Let $w \in \Sigma^*$. If $(S, Last) = \delta(\alpha_\iota, w)$, then:*

1. *$S = \{q \mid \exists w' \in \Sigma^*, q_\iota \xrightarrow{[w/w']} q\}$, and*
2. *for each $q \in S$, $Last(q) = \{a' \mid \exists w' \in \Sigma^*, q_\iota \xrightarrow{[w/w' \cdot a']} q\}$.*

Proposition 7. *For every word $w \in \Sigma^*$, $\delta(\alpha_\iota, w).I = I(w)$.*

Proof. Let $w \in \Sigma^*$, and let $(S, Last) = \delta(\alpha_\iota, w)$. Recall that $I(w) = \{a \in \Sigma \mid \exists w' \cdot a \in \Sigma^*,\ w \rightsquigarrow w' \cdot a\}$ (Definition 13).

We start with the left-to-right inclusion. Let $a \in (S, Last).I$. By definition, $a \in Last(q)$ for some $q \in S \cap Q_F$. By Lemma 4, there exists $w' \in \Sigma^*$ such that $q_\iota \xrightarrow{[w/w' \cdot a]} q$, and because $q \in Q_F$, we have that $(w, w' \cdot a) \in [T] = \rightsquigarrow$, hence $a \in I(w)$.

For the right-to-left inclusion, take $a \in I(w)$. There exists w' such that $w \rightsquigarrow w' \cdot a$. Since $\rightsquigarrow = [T]$, there exists $q \in Q_F$ such that $q_\iota \xrightarrow{[w/w' \cdot a]} q$. By Lemma 4, $q \in S$, and $a \in Last(q)$. Since $q \in S \cap Q_F$, $a \in (S, Last).I$. □

Information set bisimulation and quotient automaton.

We define a bisimulation relation over the states of $\mathcal{A}^T$, the information set automaton for T, that we call the *information set bisimulation*. Informally, two states are information set bisimilar if reading a word starting from one state or the other leads to the same information set.

Definition 15 (Information set bisimulation)**.** A binary relation $R \subseteq A \times A$ is an *information set bisimulation* if, for all α, α' in A,

- $\alpha \, R \, \alpha' \to \alpha.I = \alpha'.I$ and
- for all $a \in \Sigma$, $\delta(\alpha, a) \, R \, \delta(\alpha', a)$ (recall that $\mathcal{A}^T$ is deterministic and complete).

Two states $\alpha, \alpha' \in A$ are *information set bisimilar*, written $\alpha \leftrightarrows_I \alpha'$, if there is an information set bisimulation R such that $\alpha \, R \, \alpha'$, and we call $\leftrightarrows_I$ the *information set bisimilarity* relation. Note that $\leftrightarrows_I$ is an equivalence relation. For a state α, $[\alpha]_{\leftrightarrows_I}$ denotes the equivalence class of α. We first state the following straightforward lemma:

Lemma 5. *If $\alpha \leftrightarrows_I \alpha'$ then for all $w \in \Sigma^*$, $\delta(\alpha, w) \leftrightarrows_I \delta(\alpha', w)$.*

We also establish a useful characterization of information set bisimilar states:

Lemma 6. *For $\alpha, \alpha' \in A$, $\alpha \leftrightarrows_I \alpha'$ if, and only if, for all $w \in \Sigma^*$, $\delta(\alpha, w).I = \delta(\alpha', w).I$.*

We now show that quotienting the information set automaton $\mathcal{A}^T$ with $\leftrightarrows_I$ yields an automaton that still computes the information sets for T.

Definition 16. The *quotient automaton* is $\mathcal{A}^T_{\leftrightarrows_I} = (A_{\leftrightarrows_I}, \delta_{\leftrightarrows_I}, [\alpha_\iota]_{\leftrightarrows_I})$, where:

- $A_{\leftrightarrows_I}$ is the set of equivalence classes of $\leftrightarrows_I$,
- for $a \in \Sigma$, $\delta_{\leftrightarrows_I}([\alpha]_{\leftrightarrows_I}, a) = [\delta(\alpha, a)]_{\leftrightarrows_I}$, and
- $[\alpha]_{\leftrightarrows_I}.I = \alpha.I$.

Lemma 7. *$\mathcal{A}^T_{\leftrightarrows_I}$ is well defined and can be computed in time $O(|\mathcal{A}^T|^2)$.*

Proof. For $\alpha, \alpha' \in A$ such that $\alpha \leftrightarrows_I \alpha'$, we have that $\alpha.I = \alpha'.I$, so the information set of a state $[\alpha]_{\leftrightarrows_I}$ is well defined. Also for $\alpha, \alpha' \in A$ and $a \in \Sigma$, $\delta(\alpha, a) \leftrightarrows_I \delta(\alpha', a)$, hence $[\delta(\alpha, a)]_{\leftrightarrows_I} = [\delta(\alpha', a)]_{\leftrightarrows_I}$ and $\delta_{\leftrightarrows_I}$ is well defined.

The relation $\leftrightarrows_I$ can be computed in time $O(|\mathcal{A}^T|^2)$ (Kanellakis and Smolka, 1990), and from $\leftrightarrows_I$ the quotient automaton $\mathcal{A}^T_{\leftrightarrows_I}$ is computed in linear time. □

The following lemma is easily proved by induction:

Lemma 8. *For all $w \in \Sigma^*$, $\delta_{\leftrightarrows_I}([\alpha_\iota]_{\leftrightarrows_I}, w) = [\delta(\alpha_\iota, w)]_{\leftrightarrows_I}$.*

It follows that the quotient automaton still computes information sets correctly:

Proposition 8. *For all $w \in \Sigma^*$, $\delta_{\leftrightarrows_I}([\alpha_\iota]_{\leftrightarrows_I}, w).I = I(w)$.*

We now turn to the proofs of Theorems 3 and 4.

6.3. Upper bounds

We establish the upper bounds for Theorem 3 and Theorem 4. We first observe that in the degenerate case FUS_0, the formula $\Phi \in \mathcal{FL}^0_{\sim}$ being a CTL^* formula, the transducer is irrelevant, and in fact the problem FUS_0 is exactly the problem of solving games with CTL^* winning condition. This problem is known to be 2-EXPTIME-complete (Kupferman and Vardi, 1997), and we rephrase the precise upper-bound in our context.

Proposition 9. *Let $\mathcal{G}$ be a 2^{AP}-labelled arena and $\Phi \in \mathsf{CTL}^*$. Solving the CTL^* game $(\mathcal{G}, \Phi)$ can be done in time $(|\mathcal{G}|^d \cdot 2^{|AP|})^{2^{O(|\Phi|)}}$, where d is the maximum branching degree in $\mathcal{G}$.*

The general case

Let us fix for this section an instance $(\mathcal{G}, T, \Phi)$ of FUS_{k+1}, where $k \geq 0$, $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, and let us note once more $\Sigma = 2^{AP}$ and $\rightsquigarrow = [T]$. We describe a powerset construction that, relying strongly on the information set automaton, builds an instance of FUS_k of exponential size that is equivalent to $(\mathcal{G}, T, \Phi)$ as regards the existence of uniform strategies. Iterating this powerset construction yields an equivalent instance of FUS_0, which can be solved in time doubly exponential in the size of the formula (Proposition 9). In addition, a winning strategy in the latter CTL^* game straightforwardly provides a uniform strategy in the original instance.

Because the semantics of the full quantifier depends only on the universe and the binary relation, and not on the particular strategy Φ is evaluated on, we can use a bottom-up evaluation process in the formula before addressing the existence of a strategy. Informally, like in the classic powerset construction for games with imperfect information (Reif, 1984), we build an arena with information sets in the positions, in which formulas of the form $\boxbslash\varphi$ can be evaluated positionally if $\varphi \in \mathsf{CTL}^*$. According to the semantics of $\boxbslash$, after a finite play, the information set that we require is the set of last positions of related *finite paths in the universe*.

First, observe that T works on alphabet Σ, hence using $\mathcal{A}^T$ we would get sets of valuations instead of sets of positions. We remedy this technical problem by building a transducer T_V that recognizes the relation on V^* induced by $\rightsquigarrow$. Formally, $T_V = T_{V\to\Sigma} \circ T \circ T_{\Sigma\to V}$, where $T_{V\to\Sigma}$ is a one-state deterministic transducer with $|V|$ transitions that outputs the valuations of the positions it reads, $T_{\Sigma\to V}$ is its (nondeterministic) inverse of same size, and $\circ$ is the composition operator on transducers (Elgot and Mezei, 1965). We obtain $[T_V] = [T_{V\to\Sigma}]\circ[T]\circ[T_{\Sigma\to V}]$, *i.e.* $[T_V] = \{(w, w') \mid w, w' \in V^* \text{ and } \mu(w) \rightsquigarrow \mu(w')\}$, and $|T_V| = |T_{V\to\Sigma}| \cdot |T| \cdot |T_{\Sigma\to V}| = O(|T| \cdot |\mathcal{G}|^2)$.

The second technical problem is that $[T_V]$ may relate sequences of positions that are not paths of the universe $\mathcal{U} = Paths_*(V_\iota)$. To fix this, we define $T_\mathcal{G} = T_V \circ T_\mathcal{U}$, where $T_\mathcal{U}$ is a transducer that recognizes the identity relation over the regular language $\mathcal{U}$. We have that $|T_\mathcal{G}| = |T_V| \cdot |T_\mathcal{U}| = O(|T_V| \cdot |\mathcal{G}|) = O(|T| \cdot |\mathcal{G}|^3)$, and $[T_\mathcal{G}] = [T_V] \cap (V^* \times \mathcal{U})$. Let $\mathcal{A}^{T_\mathcal{G}} = (A, \delta, \alpha_\iota)$ be the information set automaton for $T_\mathcal{G}$.

Lemma 9. *For all $w \in V^*$, $\delta(\alpha_\iota, w).I = \{v \mid \exists \rho \cdot v \in \mathcal{U}, w \rightsquigarrow \rho \cdot v\}$.*

Proof. This follows directly from the definition of $T_\mathcal{G}$ and Proposition 7. □

We describe the synchronization of the arena $\mathcal{G}$ with $\mathcal{A}^{T_\mathcal{G}}$, which yields an arena $\widehat{\mathcal{G}}$ that has the same dynamics as $\mathcal{G}$, but in addition computes the information sets required to evaluate fully-quantified formulas.

Definition 17. Let $\widehat{\mathcal{G}} := (\widehat{V}, \widehat{E}, \widehat{V}_\iota, \widehat{v}_\iota, \widehat{\mu})$, with

- $\widehat{V} = V \times A$,
- for all $(v, \alpha) \in \widehat{V}$, for all $v' \in V$ such that $v \to v'$, $(v, \alpha) \mathrel{\widehat{\to}} (v', \delta(\alpha, v'))$
- $\widehat{V}_\iota = \{(v, \delta(\alpha_\iota, v)) \mid v \in V_\iota\}$
- $\widehat{v}_\iota = (v_\iota, \delta(\alpha_\iota, v_\iota))$ and
- $\widehat{\mu}(v, \alpha) = \mu(v)$

Each path in $\mathcal{G}$ defines a unique path in $\widehat{\mathcal{G}}$, and vice versa. To avoid confusion we shall use a "hat" version of each notation when it refers to the powerset arena $\widehat{\mathcal{G}}$. Consider the following function:

$$f : Paths_\omega(V_\iota) \to \widehat{Paths}_\omega(\widehat{V}_\iota)$$
$$\pi \mapsto \widehat{\pi} \text{ where for each } i \geq 0, \widehat{\pi}[i] = (\pi[i], \delta(\alpha_\iota, \pi[0, i])).$$

Clearly, f is a bijection between $Paths_\omega(V_\iota)$ and $\widehat{Paths}_\omega(\widehat{V}_\iota)$, *i.e.* between universes $\mathcal{U}$ and $\widehat{\mathcal{U}}$. When π is given, we shall write $\widehat{\pi}$ for $f(\pi)$, and when $\widehat{\pi}$ is given, π shall denote $f^{-1}(\widehat{\pi})$, and similarly for finite paths.

The next step is to eliminate all subformulas of Φ of the form $\boxminus\varphi$ with $d(\varphi) = 0$, *i.e.* $\varphi \in \mathsf{CTL}^*$. For each such subformula $\boxminus\varphi$ and each position $\widehat{v} = (v, \alpha)$ of $\widehat{\mathcal{G}}$, we model-check φ in all positions of the information set $\alpha.I$. Since φ holds in all these positions iff $\boxminus\varphi$ holds in (every finite path ending in) $\widehat{v}$, we can mark $\widehat{v}$ with a fresh atomic proposition $p_{\boxminus\varphi}$ when appropriate. This procedure, which we shall refer to as the *marking phase*, is described in Algorithm 1. From now on, $\widehat{\mathcal{G}}$ refers to the powerset arena after its labelling has been enriched by this marking phase.

foreach $\boxminus\varphi \in \mathit{Sub}(\Phi)$ *such that* $d(\varphi) = 0$ **do**
  foreach $\widehat{v} = (v, \alpha) \in \widehat{V}$ **do**
    if $\forall v' \in \alpha.I,\ v' \models \varphi$ **then**
      $\widehat{\mu}(\widehat{v}) := \widehat{\mu}(\widehat{v}) \cup \{p_{\boxminus\varphi}\}$;
    end
  end
end

Algorithm 1: Marking the positions of $\widehat{\mathcal{G}}$.

For a state formula φ, we define $\widehat{\varphi}$ as the formula obtained by replacing each innermost subformula of the form $\boxminus\varphi'$ with $p_{\boxminus\varphi'}$, and similarly for path formulas. For example, if $\varphi = \boxminus p \wedge \mathbf{AG}\boxminus\mathbf{EX}\boxminus q$, then $\widehat{\varphi} = p_{\boxminus p} \wedge \mathbf{AG}\boxminus\mathbf{EX}p_{\boxminus q}$.

It just remains to define the transducer $\widehat{T}$ such that $\rho \rightsquigarrow \rho'$ if and only if $\widehat{\rho} \mathrel{\widehat{\rightsquigarrow}} \widehat{\rho}'$, where $\mathrel{\widehat{\rightsquigarrow}} = [\widehat{T}]$. By Definition 17, a position (v, α) in $\widehat{\mathcal{G}}$ has the same valuation as the underlying position v in $\mathcal{G}$, except for the fresh atomic propositions added during the marking phase. So, noting $\widehat{AP}$ the set AP augmented with these fresh atomic propositions, we just modify T so that it works on alphabet $2^{\widehat{AP}}$ but ignores these additional propositions. Thus we define $\widehat{T} = T_{2^{\widehat{AP}} \to 2^{AP}} \circ T \circ T_{2^{AP} \to 2^{\widehat{AP}}}$, where $T_{2^{\widehat{AP}} \to 2^{AP}}$ is a one state deterministic transducer that reads valuations on $\widehat{AP}$ and outputs underlying valuations on AP by erasing fresh propositions, and $T_{2^{AP} \to 2^{\widehat{AP}}}$ is its inverse. The number of fresh atomic propositions is bounded by $|\Phi|$, so $|\widehat{T}| = O(2^{|\widehat{AP}|} \cdot |T| \cdot 2^{|\widehat{AP}|}) = |T| \cdot 2^{O(|\mathcal{G}|+|\Phi|)}$, and noting $\mathrel{\widehat{\rightsquigarrow}} = [\widehat{T}]$ we have that for $\rho, \rho' \in \mathit{Paths}_*(V_\iota)$, $\rho \rightsquigarrow \rho'$ if and only if $\widehat{\rho} \mathrel{\widehat{\rightsquigarrow}} \widehat{\rho}'$.

Since $d(\widehat{\Phi}) = d(\Phi) - 1 = k$, $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi})$ is an instance of FUS_k, and we prove that:

Proposition 10. $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_{k+1}$ *if, and only if,* $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}) \in \mathsf{FUS}_k$.

We establish Lemma 10 below, and Proposition 10 follows from the fact that the bijection f between $\mathit{Paths}_\omega(V_\iota)$ and $\widehat{\mathit{Paths}}_\omega(\widehat{V}_\iota)$ induces a bijection between strategy trees in $\mathcal{G}$ and strategy trees in $\widehat{\mathcal{G}}$.

More precisely, f induces a bijection between the trees of paths in $\mathcal{G}$ and those in $\widehat{\mathcal{G}}$. For any $t \subseteq \mathit{Paths}_*(v)$ where $v \in V_\iota$, we define $\widehat{t} \subseteq \widehat{\mathit{Paths}}_*(v, \delta(\alpha_\iota, v))$ by $\widehat{t} = \{\widehat{\rho} \mid \rho \in t\}$, with labelling $\widehat{\ell}(\widehat{\rho}) = \widehat{\mu}(\mathrm{last}(\widehat{\rho}))$. Notice that a node $x \in t$ being a finite path, $\widehat{x}$ is defined.

Lemma 10. *For any state formula* $\varphi \in \mathit{Sub}(\Phi)$ *and for any labelled tree* $t \subseteq \mathit{Paths}_*(V_\iota)$, *we have: for all* $x \in t$, $t, x \models \varphi$ *if, and only if,* $\widehat{t}, \widehat{x} \models \widehat{\varphi}$.

Lemma 10 shows that each powerset construction yields an equivalent FUS instance with a strictly smaller $\boxminus$-depth, and iterating the process we obtain an equivalent CTL^*

game that it remains to solve, which is a 2-EXPTIME-complete problem (Kupferman and Vardi, 1997) (see also Proposition 9).

We can now establish the upper bound for Theorem 3. For convenience, we introduce iterated exponential functions as follows:

Definition 18. For all $k, n \in \mathbb{N}$, $\exp^0(n) = n$ and $\exp^{k+1}(n) = 2^{\exp^k(n)}$.

Proposition 11. *Let $(\mathcal{G}, T, \Phi)$ be an instance of FUS_k for some $k \geq 0$, and note d the maximum branching degree in $\mathcal{G}$. Deciding whether $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_k$ can be done in time $\exp^k(|\mathcal{G}, T, \Phi|^{O(1)})^{d \cdot |AP| \cdot 2^{O(|\Phi|)}}$.*

Proof. The proof is by induction on k.

Case $k = 0$. Let $(\mathcal{G}, T, \Phi)$ be an instance of FUS_0. FUS_0 is exactly the strategy problem for Player 1 in CTL^* games, and by Proposition 9 this problem can be solved in time $(|\mathcal{G}|^d \cdot 2^{|AP|})^{2^{O(|\Phi|)}}$, which is less than $\exp^0(|\mathcal{G}, T, \Phi|)^{d \cdot |AP| \cdot 2^{O(|\Phi|)}}$.

Case $k + 1$. Let $(\mathcal{G}, T, \Phi)$ be an instance of FUS_{k+1}, with $k \geq 0$. By Proposition 10, deciding whether $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_{k+1}$ is equivalent to deciding whether $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}) \in \mathsf{FUS}_k$. Letting d be the maximum branching degree in $\widehat{\mathcal{G}}$, by induction hypothesis this can be done in time $\exp^k(|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}|^{O(1)})^{d \cdot |\widehat{AP}| \cdot 2^{O(|\Phi|)}}$. Observe that by construction of $\widehat{\mathcal{G}}$, d is also the maximum branching degree in $\mathcal{G}$. Also, the number of fresh atomic propositions used in the marking phase is bounded by $|\Phi|$, such that $|\widehat{AP}| \leq |AP| + |\Phi|$ and $\exp^k(|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}|^{O(1)})^{d \cdot |\widehat{AP}| \cdot 2^{O(|\Phi|)}}$ is in $\exp^k(|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}|^{O(1)})^{d \cdot |AP| \cdot 2^{O(|\Phi|)}}$.
We prove that the instance $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi})$ is of size $|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}| = 2^{|\mathcal{G}, T, \Phi|^{O(1)}}$, hence solving it takes time $\exp^{k+1}(|\mathcal{G}, T, \Phi|^{O(1)})^{d \cdot |AP| \cdot 2^{O(|\Phi|)}}$. We also prove that computing the powerset instance $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi})$ takes time less than $2^{|\mathcal{G}, T, \Phi|^{O(1)}}$, so that the decision procedure runs in time $\exp^{k+1}(|\mathcal{G}, T, \Phi|^{O(1)})^{d \cdot |AP| \cdot 2^{O(|\Phi|)}}$.
First, by Lemma 3, $\mathcal{A}^{T_\mathcal{G}}$ is of size $2^{O(|T_\mathcal{G}| \cdot |\Sigma|)}$, which is also the time needed to compute it. Because $T_\mathcal{G}$ works on alphabet $\Sigma = V$ and is of size $|T| \cdot |\mathcal{G}|^3$, we have that $|\mathcal{A}^{T_\mathcal{G}}| = 2^{O(|T_\mathcal{G}| \cdot |\Sigma|)} = 2^{(|\mathcal{G}| + |T|)^{O(1)}}$. Then, computing $\widehat{\mathcal{G}}$ from $\mathcal{G}$ and $\mathcal{A}^{T_\mathcal{G}}$ takes time $|\mathcal{G}| \cdot |\mathcal{A}^{T_\mathcal{G}}| = 2^{(|\mathcal{G}| + |T|)^{O(1)}}$, which is also the size of $\widehat{\mathcal{G}}$. Performing the marking phase requires to model-check at most $|\Phi|$ CTL^* formulas $|\mathcal{G}|$ times. Model-checking CTL^* is in PSPACE hence in EXPTIME, thus the marking phase can be done in time $2^{(|\mathcal{G}| + |\Phi|)^{O(1)}}$. Finally, we have seen that $|\widehat{T}| \leq |T| \cdot 2^{O(|AP|)} \cdot 2^{O(|\Phi|)}$, and it is also the time needed to compute it. Summing up everything yields the required results.

□

By Proposition 11, the complexity bounds are as follows:

- FUS_0 can be solved in time $|\mathcal{G}, T, \Phi|^{d \cdot |AP| \cdot 2^{O(|\Phi|)}}$,
- FUS_1 can be solved in time $2^{|\mathcal{G}, T, \Phi|^{O(1)} \cdot 2^{O(|\Phi|)}}$ (because $d \leq |\mathcal{G}|$ and $|AP| \leq |\mathcal{G}, T, \Phi|$),
- and for $k \geq 2$, FUS_k can be solved in time $\exp^k(|\mathcal{G}, T, \Phi|^{O(1)})$.

This establishes the upper bounds for Theorem 3.

The elementary case

We prove Theorem 4 which considers binary relations in K45NM. For such relations, all the $\boxbslash$ quantifiers can be eliminated in the formula with a single powerset construction.

Consider the marking phase described in Algorithm 1. To evaluate whether $\boxbslash\varphi$ holds in a position $\widehat{v}$ reached after some finite play $\widehat{\rho}$, we have to evaluate formula φ in all paths ρ' such that $\rho \rightsquigarrow \rho'$. If φ does not contain any subformula of the form $\boxbslash\varphi'$, *i.e.* φ is a CTL^* state formula, knowing the last position of each such ρ' is sufficient to evaluate φ; the information set contained in $\widehat{v}$ provides it. But if φ has subformulas of the form $\boxbslash\varphi'$, evaluating φ in a related path ρ' requires not only the last position of ρ', but also the state $\delta(\alpha_\iota, \rho')$ of $\mathcal{A}^{T_\mathcal{G}}$ after reading ρ'.

In the general case, as described in the proof of Proposition 11, we perform a new powerset construction, such that information sets are sets of positions of $\widehat{\mathcal{G}}$, *i.e.* an element of an information set is a pair (v, α) where v is the last position of a related path, and α the state of $\mathcal{A}^{T_\mathcal{G}}$ after reading this path.

The reason why $\mathsf{FUS}_{\mathrm{K45NM}}$ is elementary lies in the fact that if $\rightsquigarrow$ is in K45NM and $\rho \rightsquigarrow \rho'$, then $\delta(\alpha_\iota, \rho)$ and $\delta(\alpha_\iota, \rho')$ are information set bisimilar. To take advantage of this we do not build $\widehat{\mathcal{G}}$ from $\mathcal{A}^{T_\mathcal{G}}$ directly, but we first quotient it according to Definition 16. Thus in Algorithm 1, when evaluating $\boxbslash\varphi$ in a position $\widehat{v} = (v, \alpha)$ reached after some play $\widehat{\rho}$, we know that for each path ρ' such that $\rho \rightsquigarrow \rho'$, $\delta(\alpha_\iota, \rho') = \alpha$, so that we can evaluate φ directly in position $\widehat{v'} = (\mathrm{last}(\rho'), \alpha)$ of $\widehat{\mathcal{G}}$.

Let $(\mathcal{G}, T, \Phi)$ be an instance of $\mathsf{FUS}_{\mathrm{K45NM}}$, note V the set of positions in $\mathcal{G}$ and $\rightsquigarrow = [T]$ (recall that $\rightsquigarrow$ is a K45NM relation). We describe how, with one powerset construction only, we obtain an equivalent CTL^* game.

First, like in the general case, let us define T_V the transducer that relates two words in V^* if the corresponding sequences of valuations are related by T, and define $T_\mathcal{G}$ the restriction of T_V that only outputs words in $\mathcal{U} = Paths_*(V_\iota)$. Now we build the information set automaton $\mathcal{A}^{T_\mathcal{G}}$ for $T_\mathcal{G}$ and we quotient it by its bisimilarity relation $\leftrightarrows_I$. Let $\mathcal{A}^{T_\mathcal{G}}_{\leftrightarrows_I} = (A, \delta, \alpha_\iota)$ be this quotient information set automaton.

We first state that this quotient automaton computes the information we need. It follows directly from Lemma 9 and Proposition 8:

Lemma 11. *For all $w \in V^*$, $\delta(\alpha_\iota, w).I = \{v \mid \exists \rho \cdot v \in \mathcal{U}, w \rightsquigarrow \rho \cdot v\}$.*

The following lemma is the crucial point that makes the problem elementary for K45NM relations.

Lemma 12. *For all $w, w' \in V^*$, if $w \rightsquigarrow w'$ then $\delta(\alpha_\iota, w) = \delta(\alpha_\iota, w')$.*

We define the powerset arena as in Definition 17, except that we synchronize $\mathcal{G}$ with $\mathcal{A}^{T_\mathcal{G}}_{\leftrightarrows_I}$ instead of $\mathcal{A}^{T_\mathcal{G}}$. We note it $\widetilde{\mathcal{G}} = (\widetilde{V}, \widetilde{E}, \widetilde{V}_\iota, \widetilde{v}_\iota, \widetilde{\mu})$.

The bijection f between $\mathcal{U}$ and $\widetilde{\mathcal{U}}$ is defined as in the general case: for an infinite path π, $f(\pi) = \widetilde{\pi}$ where for each $i \geq 0$, $\widetilde{\pi}[i] = (\pi[i], \delta(\alpha_\iota, \pi[0, i]))$, and similarly for finite paths and for trees. To avoid confusion with the general case we denote the image of an object by f with a tilde instead of a hat ($\widetilde{\rho}$, $\widetilde{t}$, ...).

For a $\mathcal{L}_{\rightsquigarrow}$ formula φ we define its *flattening* $\widetilde{\varphi}$ as the CTL^* formula obtained by recursively replacing each subformula $\boxbslash\varphi'$ with $p_{\boxbslash\varphi'}$, starting from innermost subformulas. For example,

$$\widetilde{\boxbslash \mathbf{EX} \boxbslash q} = p_{\boxbslash \mathbf{EX} p_{\boxbslash q}}.$$

We describe in Algorithm 2 the new marking procedure, which evaluates *all* subformulas of the form $\boxminus\varphi$, starting with the innermost ones, and at the same time computes the flattening of Φ. From now on, $\widetilde{\mathcal{G}}$ refers to the powerset arena after its labelling has been enriched by this marking phase.

$\widetilde{\Phi} := \Phi$;
while $d(\widetilde{\Phi}) > 0$ **do**
  foreach $\boxminus\varphi \in Sub(\widetilde{\Phi})$ *such that* $d(\varphi) = 0$ **do**
    foreach $\widetilde{v} = (v, \alpha) \in \widetilde{V}$ **do**
      if $\forall v' \in \alpha.I,\ (v', \alpha) \models \varphi$ **then**
        $\widetilde{\mu}(\widetilde{v}) := \widetilde{\mu}(\widetilde{v}) \cup \{p_{\boxminus\varphi}\}$;
      end
    end
    $\widetilde{\Phi} := \widetilde{\Phi}[p_{\boxminus\varphi}/\boxminus\varphi]$
  end
end

Algorithm 2: Marking the positions of $\widetilde{\mathcal{G}}$.

Lemma 13. *For all state formula $\varphi \in Sub(\Phi)$, for all $v \in V_\iota$, for all labelled tree $t \subseteq Paths_*(v)$ and for all $x \in t$, $t, x \models \varphi$ if, and only if, $\widetilde{t}, \widetilde{x} \models \widetilde{\varphi}$.*

Proposition 12. *For an instance $(\mathcal{G}, T, \Phi)$ of $\mathsf{FUS}_{\mathrm{K45NM}}$, whether $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_{\mathrm{K45NM}}$ can be decided in time $2^{(|\mathcal{G},T,\Phi|)^{O(1)}.2^{O(|\Phi|)}}$.*

Proposition 12 gives the upper bound for Theorem 4. Note that the complexity is doubly exponential in the size of the formula, but for a fixed formula the problem falls in Exptime.

Synthesis of fully-uniform strategies. Regarding the problem of synthesizing uniform strategies, both in the general and the elementary case, the final step of the decision procedure consists in solving the final CTL^* game, and this is done by deciding the nonemptiness of a parity tree automaton. As mentioned in Remark 5.3, decision procedures for the emptiness of parity tree automata can synthesize a finitely represented tree in the language if it is not empty. This regular tree represents a finite memory winning strategy in the CTL^* game. Assuming that each position v in the original arena is identified with a dedicated atomic proposition p_v enables to keep track in the successive powerset constructions of which original position underlies a powerset one. This way a regular winning strategy in the CTL^* game readily gives a fully-uniform strategy in the initial game. This regular strategy has the same number of states as the parity tree automaton. In the elementary case, this would be doubly exponential in the size of the formula. In the general case it would be the same for a $\boxminus$-depth below 2, and above it would be k-exponential in the size of the original problem.

6.4. Lower bounds

Regarding Theorem 4, we easily have:

Proposition 13. $\mathsf{FUS}_{\mathrm{K45NM}}$ *is* 2-EXPTIME*-hard.*

Proof. The problem of solving CTL^* games, which is 2-EXPTIME-complete (Kupferman et al., 2001), readily reduces to $\mathsf{FUS}_{\mathrm{K45NM}}$. □

In the rest of the section we establish the lower bounds for Theorem 3.

FUS, like $\mathsf{FUS}_{\mathrm{K45NM}}$, contains the problem of solving CTL^* games, which gives the 2-EXPTIME lower bound for FUS_0 and FUS_1. It remains to prove the k-EXPTIME lower bounds for FUS_k with $k \geq 2$, which is done by the following proposition.

Proposition 14. *For $k \in \mathbb{N}$, FUS_{k+1} is $(k+1)$-*EXPTIME*-hard even if the $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula is assumed to be fixed and the transducer is assumed to be synchronous.*

For each $k \in \mathbb{N}$, let $\exp[k]$ denote the class of functions $f : \mathbb{N} \to \mathbb{N}$ such that for some constant $c \geq 1$, $f(n) = \exp^k(n^c)$ for all $n \in \mathbb{N}$.

Fix $k \geq 0$. Proposition 14 is proved by a polynomial-time reduction from the word problem for $\exp[k]$-space bounded *alternating* Turing Machines with a binary branching degree and without halting configurations, which is a well-known $(k+1)$-EXPTIME-complete problem (Chandra et al., 1981). Fix such an alternating Turing Machine $\mathcal{M} = (A, Q = Q_\exists \cup Q_\forall, q_\iota, \delta, F)$ over the input alphabet A, where the set of states Q is partitioned into a set $Q_\exists$ of existential states and a set $Q_\forall$ of universal states, q_ι is the initial state, F is the set of accepting states, and the transition function is of type

$$\delta : Q \times A \to (Q \times A \times \{\leftarrow, \rightarrow\}) \times (Q \times A \times \{\leftarrow, \rightarrow\})$$

(in each step, $\mathcal{M}$ overwrites the tape cell being scanned, and the tape head moves one position to the left – $\leftarrow$ – or to the right – $\rightarrow$). Fix an input α and let $n = |\alpha|$.

Note that a configuration of $\mathcal{M}$, or *TM configuration*, can be seen as a word $\alpha_1 \cdot (q, a) \cdot \alpha_2$ in $A^* \cdot (Q \times A) \cdot A^*$, where $\alpha_1 \cdot a \cdot \alpha_2$ denotes the tape content, q the current state, and the reading head is at position $|\alpha_1| + 1$. A TM configuration is *well-formed* if it has length exactly $\exp^k(n)$. Since $\mathcal{M}$ is $\exp[k]$-space bounded, without loss of generality, we can assume that each reachable TM configuration from the fixed input α is well-formed. In particular, the *initial TM configuration* is the unique well-formed TM configuration having the form $(q_\iota, \alpha(1))\,\alpha(2) \ldots \alpha(n)\# \ldots \#$, where $\#$ is the blank symbol. For a TM configuration $C = \alpha_1 \cdot (q, a) \cdot \alpha_2$, the *left* (resp., *right*) successor of C is the successor of C obtained by choosing the left (resp., the right) triple in $\delta(q, a)$. A *computation tree of* $\mathcal{M}$ (over α) is an infinite binary tree whose nodes are labelled by *well-formed* TM configurations and that verifies: (i) the root is labelled by the initial TM configuration, (ii) each node labelled by an *existential* TM configuration C (i.e., the associated state is in $Q_\exists$) has a unique child, labelled by some successor of C, and (iii) each node labelled by an *universal* TM configuration C (i.e., the associated state is in $Q_\forall$) has two children, labelled by the left and right successors of C, respectively. A computation tree of $\mathcal{M}$ is accepting if each infinite branch from the root visits some accepting TM configuration. $\mathcal{M}$ accepts α *iff* there is an accepting computation tree of $\mathcal{M}$.

Proposition 14 directly follows from the following proposition.

Proposition 15. *There is a* fixed *$\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula φ (independent of n and $\mathcal{M}$) such that one can build – in time polynomial in n and the size of $\mathcal{M}$ – a 2^{AP}-labelled arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ and a finite-state synchronous transducer T over the alphabet 2^{AP} so that $\mathcal{M}$ accepts α* iff *Player 1 in $\mathcal{G}$ admits a $([T], \varphi)$-uniform strategy.*

Proof of Proposition 15. We assume that $k \geq 1$ (the case $k = 0$ being simpler). First, we define a suitable encoding of (well-formed) TM configurations, obtained by using the following set AP of atomic propositions:

$$AP := A \cup (Q \times A) \cup \{\$_1, \ldots, \$_k, \$, \$_{acc}, 0, 1, L, R, \exists, \forall\} \cup \{inc, =, good\}$$

For each cell of a well-formed TM configuration C, we keep track of the content of the cell together with a suitable encoding of the cell number which is a natural number in $[0, \exp^k(n) - 1]$. Thus, for all $1 \leq h \leq k$, we define the notions of *h-block* and *well-formed h-block*. Essentially, for $h < k$, well-formed h-blocks are finite words over $\{\$_1, \ldots, \$_h, 0, 1\}$ which encode integers in $[0, \exp^h(n) - 1]$, while well-formed k-blocks are finite words over $A \cup (Q \times A) \cup \{\$_1, \ldots, \$_k, 0, 1\}$ which encode the cells of well-formed TM configurations. In particular, for $h > 1$, a well-formed h-block encoding a natural number $m \in [0, \exp^h(n) - 1]$ is a sequence of $\exp^{h-1}(n)$ $(h-1)$-blocks, where the i^{th} $(h-1)$-block encodes both the value and (recursively) the position of the i^{th}-bit in the binary representation of m. Formally, the set of (well-formed) h-blocks is defined by induction on h as follows:

Base Step: $h = 1$. The notions of 1-block and well-formed 1-block coincide, and a 1-block is a finite word bl having the form $bl = \$_1 \tau bit_1 \ldots bit_n \$_1$ such that $bit_1, \ldots, bit_n \in \{0, 1\}$ and $\tau \in \{0, 1\}$ if $1 < k$, and $\tau \in A \cup (Q \times A)$ otherwise. We say that bit_i (for $1 \leq i \leq n$) is the i^{th} bit of bl. The *content* of bl is τ, and the *index* of bl is the natural number in $[0, \exp^1(n) - 1]$ (recall that $\exp^1(n) = 2^n$) whose binary code is $bit_1 \ldots bit_n$. The 1-block bl is *initial* (resp., *final*) if $bit_i = 0$ (resp., $bit_i = 1$) for all $1 \leq i \leq n$.

Induction Step: $1 < h \leq k$. An *h-block* is a finite word bl having the form $\$_h \cdot \tau \cdot bl_0 \ldots bl_j \cdot \$_h$ such that $j > 0$, $bl_0, \ldots, bl_j$ are $(h-1)$-blocks, and $\tau \in \{0, 1\}$ if $h < k$, and $\tau \in A \cup (Q \times A)$ otherwise. Additionally, we require that bl_0 is initial, bl_j is final, and for all $0 < i < j$, bl_i is not final. The *content* of bl is τ. The h-block bl is *initial* (resp., *final*) if the content of bl_i is 0 (resp., 1) for all $0 \leq i \leq j$. The h-block bl is *well-formed* if additionally, the following holds: $j = \exp^{h-1}(n) - 1$ and for all $0 \leq i \leq j$, bl_i is well-formed and has index i. If bl is well-formed, then its *index* is the natural number in $[0, \exp^h(n) - 1]$ whose binary code is given by $bit_0, \ldots, bit_j$, where bit_i is the content of the sub-block bl_i for all $0 \leq i \leq j$.

Encoding of (well-formed) TM configurations. Let $C = C(0) \ldots C(j)$ be a TM configuration of length at least 2. A *TM configuration code* (for C) is a word over $AP \setminus \{inc, =, good, L, R, \exists, \forall\}$ of the form $code = \tau \cdot bl_0 \ldots bl_j \cdot \tau$ satisfying the following:

- $\tau = \$$ if C is not accepting and $\tau = \$_{acc}$ otherwise;
- for all $0 \leq i \leq j$, bl_i is a k-block whose content is $C(i)$;
- bl_0 is initial and bl_j is the unique final k-block.

We say that *code* is *initial* if C is of the form $(q_\iota, \alpha(0))\, \alpha(1) \ldots \alpha(n) \# \ldots \#$ (note that we do not require that C is the well-formed initial TM configuration). Moreover, we say that *code* is *well-formed* if additionally, $j = \exp^k(n) - 1$ (hence, C is well-formed) and for all $0 \leq i \leq j$, bl_i is well-formed and has index i. Note that there is exactly one well-formed TM configuration code associated with a well-formed TM configuration.

Encoding of TM computations. We use the four additional symbols L, R, $\exists$, and $\forall$ to encode single computations of $\mathcal{M}$. Intuitively, the symbol $\exists$ (resp., $\forall$) is used to delimit an existential (resp., universal) configuration, while the symbol L (resp., R) is used to delimit the left (resp., right) successor of a TM configuration. A *TM computation code* is an infinite sequence ν of the form $\nu = \mathit{code}_0 \cdot Q_0 \cdot \mathit{dir}_1 \cdot \mathit{code}_1 \cdot Q_1 \cdot \mathit{dir}_2 \ldots$ such that for all $i \geq 0$, code_i is a TM configuration code which is initial if $i = 0$, $\mathit{dir}_{i+1} \in \{L, R\}$, and $Q_i = \exists$ if the TM configuration C_i associated with code_i is existential, and $Q_i = \forall$ otherwise. The TM computation code ν is well-formed if code_i is well-formed for all $i \geq 0$, and ν is accepting if ν visits some accepting configuration code. Moreover, ν is *fair* if additionally, C_{i+1} is the left successor of C_i if $\mathit{dir}_{i+1} = L$, and the right successor of C_i otherwise. Thus, the accepting, fair, and well-formed TM computation codes encode all the possible accepting TM computations of $\mathcal{M}$ over α.

Note that we have not used the symbols in $\{\mathit{inc}, =, \mathit{good}\}$ in the encoding of TM computations. These extra symbols are used to mark positions along a TM computation code and are crucial for the implementation of transducer satisfying Proposition 15. So, we give the following additional definitions. For a word w over 2^{AP}, the *content* of w is the word over $2^{AP \setminus \{\mathit{inc}, =, \mathit{good}\}}$ obtained by removing from each letter in w the extra symbols in $\{\mathit{inc}, =, \mathit{good}\}$. An *extended TM computation code* is an infinite word over 2^{AP} whose content corresponds to a TM computation code. For an arena with labelling over 2^{AP} and a play π, the *content* of π is the content of its labelling $\mu(\pi)$.

Construction of the arena $\mathcal{G}$ in Proposition 15. The following is straightforward:

Lemma 14. *One can construct in time polynomial in n and the size of $\mathcal{M}$, a 2^{AP}-labelled arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ such that $V_\iota = \{v_\iota\}$ and the following holds:*

1. *for each extended TM computation code ν, there is a play whose labelling is ν;*
2. *for each finite play ρ, the labelling ν_ρ of ρ is the prefix of some extended TM computation code; moreover, for each extended TM computation code ν having ν_ρ as prefix, ρ can be extended to an infinite play whose labelling is ν;*
3. *the set of positions of Player 2 is the set of positions labelled by proposition $\forall$.*

In the following, let $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ be the arena of Lemma 14. Note that in a position labelled by $\forall$, Player 2 chooses between a next position labelled with L and one labelled with R, hence she simulates the universal choices of $\mathcal{M}$; similarly Player 1 simulates the existential choices. Moreover, by Properties 1–3 of Lemma 14, we easily obtain the following.

Remark 3. $\mathcal{M}$ accepts α *iff* Player 1 has a strategy σ in $\mathcal{G}$ such that for all $\pi \in \mathrm{Out}(\sigma)$, the content of $\mu(\pi)$ is a well-formed, fair and accepting TM computation code.

Construction of the finite-state transducer T in Proposition 15. Let ρ be a finite play of $\mathcal{G}$ and $1 \leq h \leq k$. We say that ρ is *not* tagged if each position along ρ is not labelled by the extra symbols in $\{\mathit{inc}, =\}$. Moreover, we say that ρ is a $(h, =)$-*tagged* (resp., (h, inc)-*tagged*) play if exactly two positions along ρ are labelled by some proposition in $\{=, \mathit{inc}\}$, this proposition is $=$ (resp., inc), and these two positions correspond to the initial positions of two h-blocks along ρ. Additionally, for a (h, inc)-tagged play, we require that the two tagged h-blocks are adjacent. Assuming that the two tagged h-blocks bl_1 and bl_2 are well-formed, then the tag $=$ is used to check that bl_1 and bl_2

have the same index, while the tag *inc* is used to check that the indices of bl_1 and bl_2 are *consecutive* (i.e, bl_1 is not final and the index of bl_2 is the index of bl_1 plus one).

A $\boxdot$-*propositional* $\mathcal{FL}_{\rightsquigarrow}$ *formula* contains only modality $\boxdot$, boolean connectives, and atomic propositions. Let $\mathcal{U}$ be the universe of $\mathcal{G}$. Note that since V_ι is a singleton (Lemma 14), $\mathcal{U}$ is a tree. For a transducer T, a finite play ρ of $\mathcal{G}$, and a $\mathcal{FL}_{\rightsquigarrow}$ state formula φ, we write $\rho \models \varphi$ to mean that $\mathcal{U}, \rho \models \varphi$ with relation $[T]$. Note that if φ is $\boxdot$-propositional, then for each strategy σ of $\mathcal{G}$ and $\rho \in t_\sigma$, $t_\sigma, \rho \models \varphi$ iff $\mathcal{U}, \rho \models \varphi$.

The core result in the proposed reduction is represented by the following lemma.

Lemma 15. *One can construct in time polynomial in n and the size of the TM $\mathcal{M}$ a synchronous finite-state transducer T over 2^{AP} such that there are two* fixed $\boxdot$*-propositional $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formulas φ_{conf} and φ_{fair} over* $\{\text{good}\}$*, and for all $1 \leq h \leq k$, three* fixed $\boxdot$*-propositional $\mathcal{FL}^{h}_{\rightsquigarrow}$ formulas $\varphi^h_{=}$, φ^h_{inc}, and φ^h_{bl} over* $\{\text{good}\}$ *so that the following holds.*

1. *Let ρ be an $(h, =)$-tagged finite play of $\mathcal{G}$. If the two tagged h-blocks* bl_1 *and* bl_2 *of ρ are well-formed, then $\rho \models \varphi^h_{=}$* iff bl_1 *and* bl_2 *have the same index.*
2. *Let ρ be an (h, inc)-tagged finite play of $\mathcal{G}$. If the two tagged h-blocks* bl_1 *and* bl_2 *of ρ are well-formed and* bl_1 *precedes* bl_2*, then $\rho \models \varphi^h_{\text{inc}}$* iff *the indices of* bl_1 *and* bl_2 *are consecutive.*
3. *Let ρ be a non-tagged finite play ρ of $\mathcal{G}$ leading to an h-block* bl*. If each sub-block of* bl *is well-formed, then $\rho \models \varphi^h_{\text{bl}}$* iff bl *is well-formed.*
4. *Let ρ be a non-tagged finite play of ρ leading to a TM configuration code* – code – *followed by either an $\exists$-position or a $\forall$-position. Then, if each k-block of* code *is well-formed, $\rho \models \varphi_{\text{conf}}$* iff code *is well-formed.*
5. *Let ρ be a non-tagged finite play of $\mathcal{G}$ having a suffix whose content has the form* $\text{code} \cdot Q \cdot \text{dir} \cdot \text{code}'$*, where $Q \in \{\exists, \forall\}$,* $\text{dir} \in \{L, R\}$*, and* code *and* code' *are two TM configuration codes associated with two TM configurations C and C'. If* code *and* code' *are well-formed, then $\rho \models \varphi_{\text{fair}}$* iff *$C'$ is the* dir*-successor of C.*

Proof sketch: The main idea, for each point, is to decompose the verification of a property in layers implementable with polynomially many states in the transducer, and invoking other layers thanks to the $\boxdot$ quantifier. We illustrate this by sketching the idea of the construction for point 1. When it reads a $(h, =)$-tagged play ρ for $h > 1$, the transducer outputs a $(h-1, =)$-tagged play with same content as ρ: for each $=$-tagged h-block in ρ it nondeterministically tags one $(h-1)$-sub-block with $=$ and checks that these sub-blocks have the same content, in which case it tags the last position of the output with *good*. It just remains to ask for the $(h-1)$-sub-blocks tagged with $=$ to have the same content if they have same index, which is done by letting $\varphi^h_{=} := \boxdot(\varphi^{h-1}_{=} \rightarrow \mathit{good})$. For $h = 1$ the transducer directly guesses an index $1 \leq j \leq n$ and verifies that the $j-th$ bit is the same in both $=$-tagged block, and we let $\varphi^1_{=} = \boxdot \mathit{good}$. This behavior can be implemented with a number of states polynomial in n and the size of $\mathcal{M}$.

Construction of the *fixed* $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula φ in Proposition 15 and proof of Proposition 15. Let T be the synchronous finite-state transducer and φ_{conf}, φ_{fair}, φ^1_{bl}, ..., φ^k_{bl} be the *fixed* $\boxdot$-propositional $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formulas satisfying Lemma 15. Then, the *fixed* $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula φ is given by $\varphi := \mathbf{A}\psi$, the path formula ψ being defined as follows, where $\varphi' := \$_1 \vee \ldots \vee \$_k \vee \$ \vee \$_{acc}$:

$$\psi := \begin{array}{c} \underbrace{\mathbf{G}\big(\neg inc \wedge \neg =\big) \wedge \mathbf{GF}(\$ \vee \$_{acc})}_{\text{the infinite play corresponds to a non-tagged TM computation code}} \\ \wedge \\ \underbrace{\bigwedge_{h=1}^{k} \mathbf{G}((\$_h \wedge \mathbf{X}\varphi') \rightarrow \varphi^h_{bl}) \wedge \mathbf{G}((\exists \vee \forall) \rightarrow \varphi_{conf})}_{\text{the TM computation code is well-formed}} \\ \wedge \\ \underbrace{\mathbf{G}\big((\exists \vee \forall) \rightarrow \mathbf{XG}(\mathbf{X}(\exists \vee \forall) \rightarrow \varphi_{fair})\big)}_{\text{the well-formed TM sequence code is fair}} \\ \wedge \\ \underbrace{\mathbf{F}\,\$_{acc}}_{\text{the fair well-formed TM computation code is accepting}} \end{array}$$

By Remark 3 and Lemma 15, it easily follows that $\mathcal{M}$ accepts α *iff* Player 1 has a $([T], \mathbf{A}\psi)$-uniform strategy in $\mathcal{G}$, which proves Proposition 15.

7. Extensions

We generalize our language to different relations, we show how to combine the techniques developed for strictly-uniform and fully-uniform strategies, and we discuss some consequences of our results on the model checking problem for logics of knowledge and time. Because for $n = 0$ the language that we consider collapses to CTL^*, we generalise our setting to n relations with $n \geq 1$.

7.1. Generalization to n relations

For $n \geq 1$, the set of well-formed $n\mathcal{L}_{\rightsquigarrow}$ formulas is:

$$\begin{array}{ll} \text{State formulas:} & \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{A}\psi \mid \boxminus_i \varphi \mid \boxbslash_i \varphi \\ \text{Path formulas:} & \psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\psi \mid \psi \mathbf{U} \psi, \end{array}$$

where $p \in AP$ and $i \in \{1, \ldots, n\}$.

The models now include one relation $\rightsquigarrow_i$ for each pair of quantifiers $\boxminus_i, \boxbslash_i$. The semantics of formulas $\boxminus_i \varphi$ or $\boxbslash_i \varphi$ becomes – given a family of relations $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$, a universe $\mathcal{U}$, a 2^{AP}-labelled tree t and a node $x \in t$:

- $t, x \models \boxminus_i \varphi$ if for all $y \in t$ such that $x \rightsquigarrow_i y$, $t, y \models \varphi$
- $t, x \models \boxbslash_i \varphi$ if for all $y \in \mathcal{U}$ such that $x \rightsquigarrow_i y$, $\mathcal{U}_y, y \models \varphi$

Definition 19. For $n \geq 0$, let $\mathcal{S}n\mathcal{L}_{\rightsquigarrow}$ and $\mathcal{F}n\mathcal{L}_{\rightsquigarrow}$ be the sublanguages of $n\mathcal{L}_{\rightsquigarrow}$ that use only one kind of quantifier: respectively, $\boxminus_i$ quantifiers and $\boxbslash_i$ quantifiers. We call $n\mathsf{SUS}, n\mathsf{SUS}_{\mathrm{Rec}}, n\mathsf{FUS}, n\mathsf{FUS}_k$ and $n\mathsf{FUS}_{\mathrm{K45NM}}$ the natural adaptations of (respectively) $\mathsf{SUS}, \mathsf{SUS}_{\mathrm{Rec}}, \mathsf{FUS}, \mathsf{FUS}_k$ and $\mathsf{FUS}_{\mathrm{K45NM}}$ to the case of uniformity constraints in $n\mathcal{L}_{\rightsquigarrow}$.

Theorem 5. $n\mathsf{SUS}$ *is undecidable and* $n\mathsf{SUS}_{\mathrm{Rec}}$ *is* 2-EXPTIME*-complete.*

Proof. Because SUS is an undecidable subproblem of $n\mathsf{SUS}$, $n\mathsf{SUS}$ is also undecidable. For the 2-Exptime membership of $n\mathsf{SUS}_{\mathrm{Rec}}$, observe that all the results on jumping tree automata established in Section 5 generalize to the case of n relations. In particular, for a jumping tree automaton $\mathcal{A}$ equipped with a family of recognizable relations $\{\rightsquigarrow_i\}_{1\leq i\leq n}$, there is an equivalent two-way automaton of size $O(|\mathcal{A}| \cdot \sum_{i=1}^{n} |\mathcal{B}_i|)$. This automaton works as described in the proof of Proposition 6: when it starts to simulate a jump for relation $\rightsquigarrow_i$ it triggers the associated recognizer $\mathcal{B}_i$. The 2-Exptime-hardness of $\mathsf{SUS}_{\mathrm{Rec}}$ extends to $n\mathsf{SUS}_{\mathrm{Rec}}$. □

Theorem 6. *$n\mathsf{FUS}_k$ is k-Exptime-complete for $k \geq 2$, 2-Exptime-complete otherwise.*

Proof. The lower bounds are directly inherited from FUS_k. For the upper bound, we explain informally how to adapt the construction in Section 6.3. From each transducer T_i (recognizing relation $\rightsquigarrow_i$), build the associated information set automaton $\mathcal{A}^{T_i}$, and synchronize the arena $\mathcal{G}$ with $\mathcal{A}^{T_1}, \ldots, \mathcal{A}^{T_n}$. This gives a powerset arena of size $2^{(|\mathcal{G}|+|T_1|+\ldots+|T_n|)^{O(1)}}$ in which each innermost subformula of the form $\boxtimes_i\varphi$ can be evaluated positionally, using the information computed by $\mathcal{A}^{T_i}$. The rest of the proof is similar to the case of one relation. □

Corollary 5. *$n\mathsf{FUS}$ is nonelementary.*

So far, generalizing to n relations does not change the decidability/complexity of the different problems we introduced. Concerning $n\mathsf{FUS}_{\mathrm{K45NM}}$, the fully-uniform strategy problem for K45NM relations, allowing for several relations raises the complexity from 2-Exptime-complete to nonlementary. However the complexity is still better than for the case of arbitrary rational relations, as the height of the tower of exponentials is determined by the number of *alternations* between quantifiers for different relations in the formula, instead of the total $\boxtimes$-depth. We inductively define the *alternation depth* of a formula $\varphi \in \mathcal{F}n\mathcal{L}_{\rightsquigarrow}$, noted $ad(\varphi)$, as follows:

$$\begin{array}{lll} ad(p) = 0 & ad(\neg\varphi) = ad(\varphi) & ad(\varphi \vee \varphi') = \max(ad(\varphi), ad(\varphi')) \\ ad(\mathbf{A}\psi) = ad(\psi) & ad(\mathbf{X}\psi) = ad(\psi) & ad(\psi\mathbf{U}\psi') = \max(ad(\psi), ad(\psi')) \\ \multicolumn{3}{c}{ad(\boxtimes_i\varphi) = 1 + \max\{ad(\boxtimes_j\varphi') \mid \boxtimes_j\varphi' \in Sub(\varphi) \text{ and } i \neq j\}} \end{array}$$

Definition 20. For $h, n \geq 0$, $n\mathsf{FUS}^h_{\mathrm{K45NM}}$ is the restriction of $n\mathsf{FUS}_{\mathrm{K45NM}}$ to $\mathcal{F}n\mathcal{L}_{\rightsquigarrow}$ formulas of alternation depth at most h.

Theorem 7. *$n\mathsf{FUS}^h_{\mathrm{K45NM}}$ is h-Exptime-complete if $h \geq 2$, 2-Exptime-complete otherwise.*

Proof. For the upper bound it is relatively easy to see how the elementary procedure for $\mathsf{FUS}_{\mathrm{K45NM}}$ can be adapted. Roughly speaking, using the fact that for each relation $\rightsquigarrow_i$ two related paths take the information set automaton $\mathcal{A}^{T_i}$ in information-set bisimilar states, one can evaluate on the first powerset arena all subformulas of the form $\boxtimes_i\varphi$ as long as there is no alternation. Then, in order to evaluate positionally formulas of the form $\boxtimes_i\varphi$ where φ contains a fresh proposition of the form $p_{\boxtimes_j\varphi'}$ with $i \neq j$, a new powerset construction is needed. Iterating the process gives the desired upper bounds.

For the lower bounds, the construction presented for FUS in Proposition 15 (Page 31) can be adapted. The behaviour of the transducer T built in the proof of Lemma 15

(Page 34) shows that the relation it recognizes is not in K45NM. First, we describe how to obtain K45NM relations by alternating between two transducers that each recognize a transitive and Euclidean relation. To do so, we make each transducer label its outputs with a special atomic proposition that identifies the transducer that produced the output. Say that the first transducer, T_0, uses p_{T_0} and T_1 uses p_{T_1}. First, observe that an input ρ labelled by some p_{T_i} has necessarily been output by T_i on some input ρ', and furthermore a transducer reading ρ can "know" exactly what was this ρ'. Indeed, the contents of ρ and ρ' are the same, and we can add finitely many atomic propositions to mark on an output what was the kind of the input (there are but a finite number of different kinds of plays). Therefore we can let T_i behave on a p_{T_i}-tagged play ρ as it would have behaved on the input ρ' that originated ρ. This ensures that on a given input of T_i, all the outputs are related together, so that $[T_i]$ is transitive and Euclidean. On a play that is tagged by neither p_{T_0} nor p_{T_1}, T_i just behaves normally and tags the outputs with p_{T_i}. On plays tagged with $p_{T_{1-i}}$, T_i does the same, but in addition it removes the tag $p_{T_{1-i}}$. In the formulas built in the proof, we alternate between full quantifiers for each transducer, such that each T_i always receives as input a play that is not tagged with p_{T_i}, and it therefore behaves normally.

Now, making the relations verify the No Miracles property can be done by adding loops on accepting states of the transducers with labels a/a for each a in the alphabet but, as a side effect, insignificant related plays are added. This issue can however be dealt with by adding two new atomic propositions, $valid_0$ and $valid_1$, used by transducers to signal which related plays are relevant and which ones are added to achieve the No Miracles property. More precisely, when T_i reads a partial play, in addition to its normal behaviour (described above) it also marks each position in the output with $valid_i$, except in loop transitions added for the No Miracles property, that still just copy the input. Making sure that transducer T_i never takes as input a word already marked with $valid_i$ ensures that the relevant outputs of T_i and only them are marked with $valid_i$. This is achieved by letting each transducer erase the marker of the other, by relativising the $\boxminus_i$-quantifications to $valid_i$-marked plays, and specifying in the main formula that the plays in the strategy must be originally unmarked (see Lemma 15). □

Corollary 6. *$n\mathsf{FUS}_{\mathrm{K45NM}}$ is nonelementary.*

7.2. Mixing strict and full quantifiers

The fact that $n\mathcal{L}_{\rightsquigarrow}$ allows for quantifiers over different relations (when $n > 1$) makes it possible to define a relevant class of uniformity constraints that combine strict and full quantifiers. To give the idea, consider an $\mathcal{L}_{\rightsquigarrow}$ formula φ that contains no strict quantifier in the scope of a full quantifier. This means that if $\boxminus_i\varphi' \in \mathit{Sub}(\varphi)$, then there is no formula of the form $\boxminus_j\varphi''$ in $\mathit{Sub}(\varphi')$, for any j. If the relations attached to the full quantifiers in φ are rational, then we can iterate powerset constructions and subformula elimination of Section 6.3 to remove all full quantifiers. Recall that there is a bijection between plays in an arena and plays in its powerset construction. Because this bijection preserves relations, the truth of strict quantifiers is also preserved between a tree of the original arena and a tree of the powerset arena. Therefore, eliminating the full quantifiers yields an equivalent instance of $n\mathsf{SUS}$. This instance can in turn be solved using jumping tree automata, provided the relations attached to the strict quantifiers are recognizable.

We let $\mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$ be the set of formulas in $n\mathcal{L}_{\rightsquigarrow}$ such that there is no strict quantifier in the scope of a full quantifier. For example, $\mathbf{AG}[\!]_2 q \in \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$, $\langle\!\rangle_1 \mathbf{EF}[\!]_2[\!]_1 p \in \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$, but $[\!]_1 \mathbf{AX}\langle\!\rangle_2 p \notin \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$.

Definition 21. For $n \geq 1$, $n\mathsf{SFUS}$ is the uniform strategy problem for uniformity properties in $\mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$ such that if $\langle\!\rangle_i$ appears in the formula, $\rightsquigarrow_i$ is a recognizable relation given by its recognizer $\mathcal{B}_{\sim_i}$, and if $[\!]_i$ appears in the formula, $\rightsquigarrow_i$ is a rational relation given by a transducer T_i.

We generalize the notions of *nesting depth* and *alternation depth* to $\mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$ formulas by simply ignoring the strict quantifiers. We also define the following problems: $n\mathsf{SFUS}_k$ is the restriction of $n\mathsf{SFUS}$ to formulas of nesting depth at most k, $n\mathsf{SFUS}_{\mathrm{K45NM}}$ is the restriction of $n\mathsf{SFUS}$ to the case where the transducers recognize K45NM relations, and $n\mathsf{SFUS}^h_{\mathrm{K45NM}}$ restricts $n\mathsf{SFUS}_{\mathrm{K45NM}}$ to formulas of alternation depth at most h.

The following results are obtained by combining the complexity results previously established for strictly and fully-uniform strategies. For the upper-bounds, the decision procedures consist in piping the ones for fully-uniform strategies (quantifier elimination) with the one for strictly-uniform strategies (jumping automata). Observe in particular that, as established in the proof of Theorem 2, our decision procedure to solve the strictly-uniform strategy problem runs in time doubly exponential in the size of the formula, but only polynomial in the size of the arena. For this reason, the two first powerset constructions do not impact the overall complexity.

Theorem 8. *$n\mathsf{SFUS}_k$ is k-*EXPTIME*-complete for $k \geq 2$, 2-*EXPTIME*-complete otherwise.*

Corollary 7. *$n\mathsf{SFUS}$ is nonelementary.*

Theorem 9. *For $n = 1$, $n\mathsf{SFUS}^h_{\mathrm{K45NM}}$ is 2-*EXPTIME*-complete for all h. For $n > 1$, $n\mathsf{SFUS}^h_{\mathrm{K45NM}}$ is h-*EXPTIME*-complete for $h \geq 2$, 2-*EXPTIME*-complete otherwise.*

Corollary 8. *For $n > 1$, $n\mathsf{SFUS}_{\mathrm{K45NM}}$ is nonelementary. Otherwise it is 2-*EXPTIME*-complete.*

7.3. Related work

We first point out two results from the literature on logics of knowledge and time that are direct corollaries of our results on uniform strategies.

Theorem 10 (van der Meyden and Shilov (1999))**.** *Model-checking LTLK_n with synchronous perfect-recall is decidable.*

Theorem 11 (Dima (2008))**.** *Model-checking CTLK_n with synchronous perfect-recall is decidable.*

Because our base language is CTL^* and synchronous perfect-recall relations are in K45NM, both problems are easily reduced to $n\mathsf{FUS}_{\mathrm{K45NM}}$. Let $\mathcal{M}$ be a model and φ a formula of either LTLK_n or CTLK_n. Each uncertainty relation $\sim_i$ is induced by an equivalence relation on states of the model. A transducer T_i with one state q which has a transition $q \dashv[s/s']\rightarrow q$ whenever states s and s' of $\mathcal{M}$ have the same observation clearly recognizes $\sim_i$. Being an equivalence relation, $\sim_i$ clearly is in K45, and one easily checks

that it verifies the No Miracles property. Then, one simply sees the model $\mathcal{M}$ as an arena whose positions all belong to Player 2. Then Player 1 has only one trivial strategy, and all possible runs in the model are in the outcome. So φ is true in the model if and only if the only trivial strategy of Player 1 is $(\{[T_i]\}_{1\leq i\leq n}, \varphi)$-uniform.

The algorithms given in van der Meyden and Shilov (1999) and Dima (2008) are in k-EXPSPACE for formulas of knowledge nesting depth k. Our results improve this upper bound: Theorem 7 gives an h-EXPTIME upper-bound, where h is the alternation depth of the formula instead of the total nesting of knowledge quantifiers. Note that for $h = 1$ we manage to obtain an EXPTIME upper-bound instead of the 2-EXPTIME one we have for $n\mathsf{FUS}^1_{\mathrm{K45NM}}$. This is because the automaton $\mathcal{A}_{\mathcal{G}}$ that recognizes the strategy trees of Player 1 in the above reduction is trivial as Player 1 never plays. In fact we can combine this automaton with the hesitant alternating automaton $\mathcal{A}_{\widehat{\Phi}}$ that accepts models of the final CTL^* formula, the automaton we obtain is still hesitant, and its emptiness can be tested in linear time (Kupferman et al., 2000). $\mathcal{A}_{\widehat{\Phi}}$ being of size exponential in $|\Phi|$, the result follows. In fact we have the following general result:

Theorem 12. *Model-checking $\mathsf{CTL}^*\mathsf{K}_n$ with rational epistemic relations on runs is in k-*EXPTIME*, where k is the modal depth of the formula. If, in addition, the epistemic relations verify transitivity, Euclideanity and No Miracles, then the problem is in h-*EXPTIME*, where h is the alternation depth of the formula.* K45NM *relations include,* e.g. *synchronous and asynchronous perfect-recall relations.*

Note that the same upper-bounds for asynchronous perfect recall and distributed knowledge have also been proved by Aucher (2013).

Uniform strategies also naturally capture the synthesis problem from knowledge-based specifications, as addressed in van der Meyden and Vardi (1998) or van der Meyden and Wilke (2005). The knowledge operators in this framework have – to use the vocabulary of uniform strategies – strictly-uniform semantics: the system in which formulas are interpreted is restricted to behaviors that are induced by the protocol/strategy they synthesize. van der Meyden and Vardi (1998) establish that the case of one agent with synchronous perfect-recall is decidable; because perfect-recall relations are not recognizable relations, this result indicates that there should be a class of relations other than Rec for which the strictly-uniform strategy problem is decidable. On the other hand, when more than one agent are involved the synthesis problem for knowledge-based specifications is undecidable; yet it is well known that making additional assumptions on information flows yields decidable cases. For instance van der Meyden and Wilke (2005) establish that the problem is decidable in *broadcast environments*, where information flows between players by means of public broadcasts only. This also suggests that there should be relations not in Rec for which, under some assumptions on the connections between relations for each quantifier $\boxminus_i$, $n\mathsf{SUS}$ is decidable. It would be very interesting to identify for which class of relations and which constraints on their inter-connections this result holds.

8. Conclusion

We have generalized the framework proposed by Maubert and Pinchinat (2014) by extending the theory from linear-time to branching-time, and we have studied in depth

the decidability and complexity of uniform strategy synthesis for both kinds of quantifiers – the strict one and the full one – and various subclasses of rational relations. We then have generalized the theory by allowing for several relations, and for each result of the single-relation case we have established its counterpart in the multiple-relation setting.

The results concerning strictly-uniform strategies shed light on phenomena in games with imperfect information. While observation-based strategies require the player to play identically in observationally equivalent plays, the notion of *knowledge-based* strategy asks the player to play identically in all situations that yield the same information set. Although being rational, the observation-based equivalence relation is not recognizable, unlike the information-set-based equivalence relation. Interestingly, in two-player games with ω-regular objectives, the existence of an observation-based strategy implies the existence of an information-set-based strategy, hence looking for the latter is enough; but this no longer holds for more players. We believe that our results (on undecidability/decidability), by distinguishing arbitrary rational relations from recognizable ones, give a new insight on the frontier between imperfect-information games with at most two players and games with more than two players.

About imperfect-information games where players have bounded memory instead of perfect recall, they are clearly decidable for ω-regular objectives as only finitely many strategies are available for each player. Still, because indistinguishability relations for bounded-memory agents are recognizable, uniform strategies and jumping tree automata provide a well-suited tool to solve such games, and also allow for arbitrary epistemic temporal winning conditions. Note also that recognizable relations do not need to be equivalence relations in general; our results thus allow for more involved indistinguishability relations, for modelling *e.g.* beliefs and plausibility. In order to handle richer strictly-uniform properties, and thus players with capacities more powerful than just bounded memory, we seek for a class of relations for which jumping tree automata languages would exceed the line of ω-regularity, and still enjoy a decidable emptiness problem.

To finish with jumping tree automata, we want to study their relationship with $\mathbf{R}\mathsf{L}_\mu$, *i.e.* the extension of the full μ-calculus with the $\boxminus$ quantifier. We conjecture that, just like classic tree automata correspond exactly to the modal μ-calculus, jumping tree automata correspond to $\mathbf{R}\mathsf{L}_\mu$, but further investigation is required.

On the other hand, our results concerning fully-uniform strategies provide a unified proof of several existing results in the domain of model-checking logics of knowledge and time with various observational powers of agents. In addition, our generic result settles the matter for any new type of capabilities that one would be led to consider, as long as the induced relations are rational.

Also, we believe that the notion of information set automaton could lead to a generic powerset-construction for solving two-player games with imperfect information and rational relations, which would capture both the classic one of Reif (1984) for synchronous perfect recall and the recent generalization to asynchronous perfect recall by Puchala (2010). This may allow us to solve a potentially vast class of games with imperfect recall, the relevance of which has recently been advocated in Berwanger et al. (2012).

Finally, the notion of uniform strategy generalizes to the case of concurrent game structures, and investigating which results are preserved is yet another interesting perspective, as well as possible extensions to strategy logics with quantifiers over uniform strategies. This would generalize existing approaches, and may provide a clean setting to address questions of strategy context (see Lopes et al. (2010)) with imperfect information.

Bibliography

Apt, K., Grädel, E., 2011. Lectures in Game Theory for Computer Scientists. Cambridge University Press. doi:path10.1017/CBO9780511973468.

Aucher, G., 2013. Infinite Games in Epistemic Temporal Logic via Supervisory Control Theory. Rapport de recherche RR-8374. INRIA. URL: `http://hal.inria.fr/hal-00866155`.

Berstel, J., 1979. Transductions and context-free languages. volume 4. Teubner Stuttgart. doi:path10.1007/978-3-663-09367-1.

Berwanger, D., Chatterjee, K., Wulf, M.D., Doyen, L., Henzinger, T.A., 2010. Strategy construction for parity games with imperfect information. Inf. Comput. 208, 1206–1220.

Berwanger, D., Doyen, L., 2008. On the power of imperfect information, in: Hariharan, R., Mukund, M., Vinay, V. (Eds.), FSTTCS, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. pp. 73–82.

Berwanger, D., Kaiser, L., 2010. Information tracking in games on graphs. Journal of Logic, Language and Information 19, 395–412.

Berwanger, D., Kaiser, L., Leßenich, S., 2012. Solving counter parity games, in: Rovan, B., Sassone, V., Widmayer, P. (Eds.), MFCS, Springer. pp. 160–171.

Bojanczyk, M., 2002. Two-way alternating automata and finite models, in: Widmayer, P., Ruiz, F.T., Bueno, R.M., Hennessy, M., Eidenbenz, S., Conejo, R. (Eds.), ICALP, Springer. pp. 833–844.

Carton, O., Choffrut, C., Grigorieff, S., 2006. Decision problems among the main subfamilies of rational relations. ITA 40, 255–275.

Chandra, A.K., Kozen, D., Stockmeyer, L.J., 1981. Alternation. Journal of the ACM 28, 114–133.

Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.F., 2006. Algorithms for omega-regular games with imperfect information' , in: Ésik, Z. (Ed.), CSL, Springer. pp. 287–302.

Dima, C., 2008. Revisiting satisfiability and model-checking for CTLK with synchrony and perfect recall, in: Fisher, M., Sadri, F., Thielscher, M. (Eds.), CLIMA, Springer. pp. 117–131.

Eilenberg, S., 1974. Automata, languages, and machines. volume 1. Elsevier.

Elgot, C., Mezei, J., 1965. On relations defined by generalized finite automata. IBM Journal of Research and Development 9, 47–68. doi:path10.1147/rd.91.0047.

Emerson, E.A., Jutla, C.S., 1991. Tree automata, mu-calculus and determinacy (extended abstract), in: FOCS, IEEE Computer Society. pp. 368–377.

Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y., 1995. Reasoning about knowledge. volume 4. MIT press Cambridge.

Frougny, C., Sakarovitch, J., 1993. Synchronized rational relations of finite and infinite words. Theor. Comput. Sci. 108, 45–82.

Grädel, E., Thomas, W., Wilke, T. (Eds.), 2002. Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]. volume 2500 of *Lecture Notes in Computer Science*. Springer.

Halpern, J.Y., Vardi, M.Y., 1989. The complexity of reasoning about knowledge and time. 1. Lower bounds. Journal of Computer and System Sciences 38, 195–237. doi:path10.1145/12130.12161.

Kanellakis, P.C., Smolka, S.A., 1990. Ccs expressions, finite state processes, and three problems of equivalence. Information and Computation 86, 43–68.

Kupferman, O., Vardi, M.Y., 1997. Module checking revisited, in: Grumberg, O. (Ed.), CAV, Springer. pp. 36–47.

Kupferman, O., Vardi, M.Y., Wolper, P., 2000. An automata-theoretic approach to branching-time model checking. J. ACM 47, 312–360.

Kupferman, O., Vardi, M.Y., Wolper, P., 2001. Module checking. Inf. Comput. 164, 322–344.

Löding, C., 2014. Automata on infinite trees, in: Pin, J.E. (Ed.), preliminary version for the handbook Automata: from Mathematics to Applications. To appear.

Lopes, A.D.C., Laroussinie, F., Markey, N., 2010. Atl with strategy contexts: Expressiveness and model checking, in: Lodaya, K., Mahajan, M. (Eds.), FSTTCS, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. pp. 120–132.

Maubert, B., 2014. Logical foundations of games with imperfect information: uniform strategies. Ph.D. thesis. Université de Rennes 1. URL: `http://www.theses.fr/en/2014REN1S001`. directed by Pinchinat, Sophie and Aucher, Guillaume.

Maubert, B., Pinchinat, S., 2013. Jumping automata for uniform strategies, in: Seth, A., Vishnoi, N. (Eds.), Proceedings of the 33rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13), Leibniz-Zentrum für Informatik, Guwahati, India. pp. 287–298. doi:path10.4230/LIPIcs.FSTTCS.2013.287.

Maubert, B., Pinchinat, S., 2014. A general notion of uniform strategies. International Game Theory Review 16. doi:path10.1142/S0219198914400040.
Maubert, B., Pinchinat, S., Bozzelli, L., 2011. Opacity issues in games with imperfect information, in: D'Agostino, G., Torre, S.L. (Eds.), GandALF, pp. 87–101. doi:path10.4204/EPTCS.54.7.
van der Meyden, R., Shilov, N.V., 1999. Model checking knowledge and time in systems with perfect recall (extended abstract), in: Rangan, C.P., Raman, V., Ramanujam, R. (Eds.), FSTTCS, Springer. pp. 432–445.
van der Meyden, R., Vardi, M.Y., 1998. Synthesis from knowledge-based specifications, in: CONCUR'98 Concurrency Theory. Springer, pp. 34–49.
van der Meyden, R., Wilke, T., 2005. Synthesis of distributed systems from knowledge-based specifications, in: Abadi, M., de Alfaro, L. (Eds.), CONCUR, Springer. pp. 562–576.
Muller, D.E., Schupp, P.E., 1987. Alternating automata on infinite trees. Theoretical computer science 54, 267–276.
Pacuit, E., van Benthem, J., 2006. The tree of knowledge in action: Towards a common perspective. Proceedings of Advances in Modal Logic Volume 6 , 87–106.
Peterson, G., Reif, J.H., Azhar, S., 2001. Lower bounds for multiplayer noncooperative games of incomplete information. Computers & Mathematics with Applications 41, 957–992.
Piterman, N., Vardi, M.Y., 2004. Global model-checking of infinite-state systems, in: Alur, R., Peled, D. (Eds.), CAV, Springer. pp. 387–400.
Puchala, B., 2010. Asynchronous omega-regular games with partial information, in: Hlinený, P., Kucera, A. (Eds.), MFCS, Springer. pp. 592–603.
Reif, J.H., 1984. The complexity of two-player games of incomplete information. Journal of computer and system sciences 29, 274–301. doi:path10.1016/0022-0000(84)90034-5.
Thomas, W., 1990. Automata on infinite objects, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B). Elsevier, pp. 133–192.
Vardi, M.Y., 1991. Verification of concurrent programs: The automata-theoretic framework. Annals of Pure and Applied Logic 51, 79–98.
Vardi, M.Y., 1998. Reasoning about the past with two-way automata, in: Larsen, K.G., Skyum, S., Winskel, G. (Eds.), ICALP, Springer. pp. 628–641.
Zielonka, W., 1998. Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. 200, 135–183.